AD-A012 886

AN INTERACTIVE COMPUTER PROGRAM FOR ASSESSING AND USING MULTIATTRIBUTE UTILITY FUNCTIONS

Alan Sicherman

Massachusetts Institute of Technology

Prepared for:

Office of Naval Research

June 1975

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

AN INTERACTIVE COMPUTER PROGRAM FOR ASSESSING AND USING MULTIATTRIBUTE UTILITY FUNCTIONS

by

ALAN SICHERMAN

Technical Report No.111

OPERATIONS RESEARCH CENTER

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 1975

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report No. 111	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN INTERACTIVE COMPUTER PROGRAM FOR ASSESSING AND USING MULTIATTRIBUTE UTILITY FUNCTIONS		5. TYPE OF REPORT & PERIOD COVERED Technical Report June 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Alan Sicherman		8. CONTRACT OR GRANT NUMBER(s) N00014-67-A-0204-0056
9. PERFORMING ORGANIZATION NAME AND ADDRESS M.I.T. Operations Research Center 77 Massachusetts Avenue, Room 24-215 Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 047-104/06/09/71 #434 M.I.T./OSP 73787
11. CONTROLLING OFFICE NAME AND ADDRESS O.R. Branch, ONR, Navy Dept. 800 North Quincy Street Arlington, VA 22217		12. REPORT DATE June 1975 13. NUMBER OF PAGES [illegible]
14. MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Releasable without limitations on dissemination.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Interactive Computer Program
Assessment of Multiattribute Utility Functions
Utilization of Multiattribute Utility Functions
Decision Analysis

PRICES SUBJECT TO CHANGE

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents a computer package designed to facilitate the assessment and use of a decision maker's utility function for multiple objectives. The package provides routines for (1) specifying the decision maker's preferences over multiple criteria, (2) treating uncertainty in the consequences resulting from a decision, (3) ranking alternative courses of action in order of preference, and (4) studying the effects changes of preferences or uncertainty estimates may have upon the ranking of alternatives. The routines (U)

20. are designed to be applicable in a variety of problem contexts.

The paper is organized as follows. The decision analysis approach which provides the theoretical basis for the program is summarized. This is followed by a description of existing methods for multiattribute utility function assessment and use. Then the computer package is presented and compared with the aforementioned methods. Applications of the package to several problems are illustrated and areas for future improvement and research are suggested. (U)

AN INTERACTIVE COMPUTER PROGRAM
FOR ASSESSING AND USING
MULTIATTRIBUTE UTILITY FUNCTIONS

by

ALAN SICHERMAN

Technical Report No. 111

Work Performed Under

Contract N00014-67-0204-0056, Office of Naval Research
Decision Analysis Research
NR-104/06-09-71 #434 MIT/OSP 73787

Operations Research Center
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

June 1975

FOREWORD

The Operations Research Center at the Massachusetts Institute of Technology is an interdepartmental activity devoted to graduate education and research in the field of operations research. The work of the Center is supported, in part, by government contracts and industrial grants-in-aid. The work reported herein was supported (in part) by the Office of Naval Research under Contract N00014-67-A-0204-0056.

Alan Sicherman is a research assistant and doctoral student at the Operations Research Center at M.I.T.

ABSTRACT

This report presents a computer package designed to facilitate the assessment and use of a decision maker's utility function for multiple objectives. The package provides routines for (1) specifying the decision maker's preferences over multiple criteria, (2) treating uncertainty in the consequences resulting from a decision, (3) ranking alternative courses of action in order of preference, and (4) studying the effects changes in preferences or uncertainty estimates may have upon the ranking of alternatives. The routines are designed to be applicable in a variety of problem contexts.

The paper is organized as follows. The decision analysis approach which provides the theoretical basis for the program is summarized. This is followed by a description of existing methods for multiattribute utility function assessment and use. Then the computer package is presented and compared with the aforementioned methods. Applications of the package to several problems are illustrated and areas for future improvement and research are suggested.

ACKNOWLEDGEMENT

I wish to thank Professor Ralph L. Keeney who supervised my research and provided me with much guidance and many helpful suggestions in drafting this thesis.

I would also like to acknowledge my fellow graduate students who took an interest in using and testing some of my research results.

Finally, I want to thank Professor John D. C. Little for taking final responsibility for my thesis in the absence of Professor Keeney.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS AND TABLES

1. INTRODUCTION

Many decision-making problems are characterized by two sources of complexity. First, there are multiple objectives on the basis of which the decision should be made. In weighing alternative actions, the decision maker must consider the tradeoffs between the degree of achievement in one objective and the degree of achievement in others. Second, there is often uncertainty about the consequences which will result from any particular action.

Because of these complexities, there is a need for a formal approach to help in evaluating alternatives. Decision analysis is an approach which explicitly addresses the multiple objective and uncertainty issues. The theoretical basis for this is well established. However, many practical problems arise when one tries to apply decision analysis in particular situations. This thesis describes a computer package for overcoming some of these difficulties.

1.1 The Decision Analysis Approach

Raiffa [14] discusses the philosophy and techniques of decision analysis in detail. We can think of the decision analysis approach as consisting of four steps:

1. structuring the problem,
2. quantifying the uncertainties involved,

3. quantifying the decision maker's preferences, and
4. combining the first three steps to evaluate the alternatives.

Structuring includes identifying the decision maker and the problem objectives. Measures of effectiveness (attributes) indicating the degree to which each objective is achieved are also formulated. Let us designate our set of attributes as X_1, $X_2, \ldots, X_n$ and use x_i to indicate a specific amount of attribute X_i. For example, X_1 may be profit in 1975 measured in thousands of dollars and x_1 may be 188. A consequence will be denoted by $\underline{x} \equiv (x_1, x_2, \ldots, x_n)$ and indicates the level x_i of each attribute which results given that consequence.

Quantifying uncertainties involves describing the uncertainty in the possible consequences of any alternative. For each alternative A_j, a probability distribution $p_j(\underline{x})$ indicating which consequences might occur and their likelihood of occurrence is required. The p_j may be derived by means of some analytical or simulation model or by subjective assessments.

Quantifying preferences means assessing the decision maker's utility function $u(\underline{x}) \equiv u(x_1, x_2, \ldots, x_n)$ which assigns a number to each of the possible consequences. This function is called a multiattribute utility function and will be referred to by the mnemonic MUF. A MUF has two properties which make it useful in addressing the issues of uncertainty and tradeoffs between objectives. These properties are:

1. $u(\underline{x}') > u(\underline{x}'')$ if and only if $\underline{x}'$ is preferred to $\underline{x}''$ and

2. in situations with uncertainty, the expected value of u is the appropriate guide for making decisions, i.e., the alternative with the highest expected value is the most preferred.

This second property follows from the axioms of rational behavior postulated first in von Neumann and Morgenstern[18].

Evaluating alternatives involves calculating the expected utility for each of the alternatives using the p_j and u from the previous steps. Various parameters of the probability distributions and the utility function can be varied to see how these affect the expected utility of the alternatives, i.e., how "sensitive" the results are to changes in the parameters.

1.2 Statement of the Problem

A major practical problem arises when one tries to obtain a MUF that is "tractable" yet appropriate for a particular situation. The general approach has been to postulate assumptions about the decision maker's preferences and derive the restrictions they place on the functional form for u. Then, for any specific problem, the adequacy of the assumptions must be verified and the parameters for the utility function assessed and checked for internal consistency. Ideally, the functional form of the MUF would have the following properties:

1. be general enough to apply to many real problems,
2. require a minimal number of assessment questions to be asked of the decision maker,
3. require assessments which are reasonable for a decision maker to consider, and
4. be easy to use in evaluating alternatives and conducting sensitivity analysis with respect to various parameters.

Even with a convenient functional form for the MUF, the nature and magnitude of a problem can make the assessment, bookkeeping, and use of quantitative preference information a formidable task. The computer package described in this thesis is designed to handle this task for a variety of problem contexts.

1.3 Organization of the Thesis

Chapter 2 summarizes the theoretical development of the functional forms for MUF's upon which the computer package is based. Chapter 3 discusses existing methods for assessing and using MUF's and their difficulties. Chapter 4 describes the computer package and the manner in which it alleviates the difficulties mentioned in Chapter 3. Chapter 5 presents several applications of the package to different problems illustrating the use of the various package routines. Chapter 6 discusses suggestions for improving the package and for future research. Chapter 7 contains a summary and conclusions of the thesis.

Five appendices contain detailed information concerning understanding and use of the computer package. Appendix A is a concise summary of the package commands. Appendix B is a listing of the program. Appendix C describes some of the algorithms used in several of the package routines. Appendix D contains a discussion of the overall program design. Appendix E explores the tradeoff properties among the attributes implied by the functional forms used for the multiattribute utility function. It serves to explain the design and use of some of the package routines.

2. THE ADDITIVE AND MULTIPLICATIVE UTILITY FUNCTIONS

This chapter states the conditions which imply that a MUF is either additive or multiplicative. None of the conditions require the decision maker to consider preference tradeoffs between more than two attributes simultaneously or to consider lotteries (specifying various $\underline{x}$ and the probabilities of receiving them) with the level of more than one attribute being varied. Furthermore, the assessments needed to specify an n-attribute utility function are n single-attribute utility functions and n scaling constants. Some properties of these forms are discussed as well as their applicability to different classes of problems.

2.1 The Basic Assumptions

The two basic assumptions which we use for both additive and multiplicative utility functions are referred to as preferential independence and utility independence. These are defined as follows:

Preferential Independence: The pair of attributes (X_1,X_2) is preferentially independent of the other attributes $(X_3,\ldots,X_n)$ if preferences among (X_1,X_2) pairs given that $(X_3,\ldots,X_n)$ are held fixed, do not depend on the level where $(X_3,\ldots,X_n)$ are fixed.

Preferential independence implies that the tradeoffs between attributes X_1 and X_2 do not depend on $X_3,\ldots,X_n$.

Utility Independence: The attribute X_1 is utility independent of the other attributes $(X_2,\ldots,X_n)$ if preferences among lotteries over X_1 (i.e., lotteries with uncertainty about the level of X_1 only) given $X_2,\ldots,X_n$ are fixed, do not depend on the level where those attributes are fixed.

The main result can now be stated.

Theorem 1. For $n \geq 3$, if for some X_i, (X_i,X_j) is preferentially independent of the other attributes for all $j \neq i$ and X_i is utility independent of all the other attributes, then either

$$u(\underline{x}) = \sum_{i=1}^{n} k_i u_i(x_i) \quad , \tag{1}$$

or

$$1 + ku(\underline{x}) = \prod_{i=1}^{n} [1 + kk_i u_i(x_i)] \quad , \tag{2}$$

where

(i) u and u_i are utility functions scaled from zero to one,

(ii) the k_i's are scaling constants with $0 < k_i < 1$, and

(iii) $k > -1$ is a non-zero scaling constant satisfying the equation

$$1 + k = \prod_{i=1}^{n} (1 + kk_i) \quad . \tag{3}$$

The proof of this result is found in Keeney [9]. Alternative sets of assumptions leading to either form (1) or (2) are found in Fishburn [3], Pollak [12], and Meyer [11]. The functional form (1) is referred to as the additive utility function and (2) is the multiplicative utility function. For the case of two attributes, the following is proved in Keeney [7]:

Theorem 2. For $n = 2$, if X_1 is utility independent of X_2 and X_2 is utility independent of X_1, then the utility function $u(x_1, x_2)$ is either additive or multiplicative.

Using either (1) or (2), if $\sum_{i=1}^{n} k_i = 1$, the utility function is additive, and if $\sum_{i=1}^{n} k_i \neq 1$, it is multiplicative. When $\sum_{i=1}^{n} k_i > 1$, then $-1 < k < 0$, and when $\sum_{i=1}^{n} k_i < 1$, then $0 < k < \infty$. To use either the additive or multiplicative form, we need to obtain exactly the same information. We have to assess the n single-attribute utility functions $u_i(x_i)$ and the n scaling constants k_i.

2.2 Nesting Utility Functions

The results concerning the functional forms above are valid regardless of whether the X_i's are scalar attributes or vector attributes. This means that the x_i's can be either scalars or vectors. In the former case, the component utility function u_i is a uniattribute utility function, whereas in the latter case, u_i is itself a multiattribute utility function.

If X_i is a vector attribute, it is possible, subject to satisfying the requisite assumptions, to use Theorems 1 and 2 concerning u_i. In such a case, we will say u_i is a nested MUF. That is, u_i is a MUF nested within the MUF u. Our interest in nesting utility functions will become more apparent in the discussion concerning the applicability of the functional forms.

2.3 Applicability of the Functional Forms

In terms of the required assessments, the additive and multiplicative utility functions appear to be the practical ones for say $n \geq 4$. Discussions on this and the reasonableness of the assumptions can be found in Keeney [9]. Even when the requisite assumptions do not precisely hold, it may be a good approximation to assume they do. Furthermore, by nesting one MUF inside another, additional flexibility in the preference structure can be achieved.

The effect of nesting multiplicative forms is to create an extra degree of freedom in the problem by having an extra independent constant. Without nesting, the number of independent scaling constants is equal to the number of single attributes. However, suppose u_n is a MUF nested within u and that u_n has three single attributes. Then one would need n scaling constants for the "outer MUF" and three for the "inner MUF" for a total of n + 3, even though there are only n + 2 single attributes, $X_i,\ldots,X_{n-1}$ and the three single attributes in u_n. The degree of freedom afforded by the extra parameter permits tradeoffs between two attributes to be dependent on a third. Specifically, tradeoffs between any of the single attributes in u_n and those not in u_n depend upon the levels of the other single attributes in u_n. This is discussed in detail in Appendix E.

Using various nesting schemes, enough extra constants could be provided to model situations in which tradeoffs between many pairs of attributes depend on the level of other attributes. That is to say, situations in which the preferential independence assumption does not hold for all the single attributes can still be modeled using nesting.

In case of utility independence violations, the particular problem may be far more sensitive to the scaling constants or tradeoffs among the attributes than to the conditional single-attribute utility function variations. Thus, even in these cases, the additive or multiplicative form may provide an adequate model for the problem.

In summary, the additive and multiplicative utility functions are simple enough to be tractable and yet, especially with nesting, robust enough to adequately quantify preferences for many problems. In practice, however, assessing and using such MUF's is "easier said than done."

3. DIFFICULTIES WITH EXISTING METHODS FOR ASSESSMENT AND USE

In this chapter, existing methods for assessing and using MJF's are discussed. Difficulties encountered with these methods include:

(1) the necessity to ask "extreme value" questions to keep the computational requirements for specifying a utility function to a manageable level,

(2) the tedium of calculating component utility functions and scaling constants even in this case,

(3) the lack of immediate feedback to the decision maker of the implication of his preferences,

(4) the absence of convenient procedures for "updating" the decision maker's preferences and conducting sensitivity analysis.

In all that follows, we will assume that the assumptions implying that the MUF is either additive or multiplicative hold. The discussion is developed in terms of the steps customarily followed in assessing and using a MUF.

3.1 Specifying the Preference Functions over the Single Attributes

Techniques for assessing single-attribute utility functions have become fairly standard (Raiffa [14]), and sophisticated computer programs have been developed for fitting single-attribute utility functions (Schlaifer [16]).

Such programs provide quick feedback which assists the decision maker in checking if his assessments and their implications appear reasonable. There is difficulty in using these programs for multiattribute utility applications, since at present, they do not exist in conjunction with a multiattribute utility assessment package.

3.2 Assessing the Tradeoffs among Attributes

The issue of tradeoffs among the attributes is addressed by assessing the k_i's in the utility functions (1) or (2). In theory, the general method for doing this is very simple. If there are n attributes, we want to assess the n unknown k_i's by creating n independent equations with the n unknowns and solving. An equation is created by (i) having the decision maker indicate two options, where an option is either a consequence or a lottery, between which he is indifferent, and (ii) equating the expected utility of these options using either (1) or (2). For instance, if the decision maker finds $\underline{x}'$ and $\underline{x}''$ indifferent, then $u(\underline{x}') = u(\underline{x}'')$ provides one equation with at most n unknowns.

Because of the difficulty and tedium in manually solving n equations (which are not necessarily linear) with n unknowns, current practice in assessing the k_i's usually requires sets of equations which are simple to solve. This basically limits the assessment questions to two types. To indicate these, let us define $\underline{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ and $x_1^\circ = (x_1^\circ, x_2^\circ, \ldots, x_n^\circ)$ as the most desirable and least desirable consequences. Then, because of the scaling conventions given in Theorems 1 and 2,

$$u(\underline{x}^*) = 1 \quad , \quad u(\underline{x}^\circ) = 0 \quad , \tag{4}$$

and

$$u_i(x_i^*) = 1 \quad , \quad u_i(x_i^\circ) = 0 \quad , \quad i = 1,2,\ldots,n \quad . \tag{5}$$

Question I. For what probability p are you indifferent between

(i) the lottery giving a p chance at $\underline{x}^*$ and 1-p chance at $\underline{x}^\circ$, and

(ii) the consequence $(x^\circ_1, \ldots, x^\circ_{i-1}, x^*_i, x^\circ_{i+1}, \ldots, x^\circ_n)$.

If we define the decision maker's answer as p_i, then using (4), the expected utility of the lottery is p_i, and using either (1) or (2), the utility of the consequence is k_i. Equating the expected utilities, we find

$$k_i = p_i \tag{6}$$

The second type of question is illustrated by

Question II. Select a level of X_i, call it x_i', and a level of X_j, call it x_j', such that, for any fixed levels of all the other attributes, you are indifferent between

(i) a consequence yielding x_i' and x_j° together, and

(ii) a consequence yielding x_j' and x_i° together.

Using (5) and either the multiplicative or additive utility function, the utilities of these two indifferent consequences can be equated to yield

$$k_i u_i(x_i') = k_j u_j(x_j') \qquad (7)$$

Once the single-attribute utility functions u_i and u_j are assessed, both $u_i(x_i)$ and $u_j(x_j)$ are easily found, so (7) is a simple linear equation expressing the relationship between k_i and k_j.

A major shortcoming of questions of both types is the use of the extreme levels of the attributes, that is the x_i^*'s and x_i°'s. Since the range from x_i° to x_i^* must cover all the possible x_i's, the implications of, and hence preferences for, the extreme levels are usually very difficult for a decision maker to consider. A further difficulty with Question I is the fact that the effect due to varying all n attributes simultaneously in a lottery must be considered. Hence, for computational ease, we must force the decision maker to respond to questions much more difficult to evaluate than would be theoretically necessary.

A common practice in assessing the k_i's would be to use a question I to evaluate the largest k_i, and then use type II questions to evaluate the magnitude of the other k_j's relative to the largest k_i. Once we have the k_i's, the additive form holds if they sum to one. Otherwise, the k_i's are substituted into (3) to evaluate the k for the multiplicative form. This last task in itself can be difficult using only a calculator.

3.3 Evaluating Alternatives and Sensitivity Analysis

Manual calculations are clearly impractical for evaluating alternatives. With uncertainty, we need to evaluate the expected value of u using the probability distribution describing the possible consequences. Even with probabilistic independence among the X_i's. the computational task is large. It is also clear that sophisticated sensitivity analyses are out of the question without major computational help.

On the other hand, it is a large requirement to develop a special computer program to accommodate a particular problem. Such programming is often inflexible because of the special problem nature for which it is done. For instance, it would be difficult to add more attributes, to try different "nesting" schemes, or explore the preference structure for "hints" of creative new alternatives to generate.

3.4 Summary of Existing Methods and Their Difficulties

Current methods for assessing and using MUF's require asking very difficult assessment questions, yield little feedback once given the responses requested and are tedious to implement computationally. These drawbacks can often result in abandoning the decision theoretic approach in favor of less explicit and theoretically well-established but more expedient methods for dealing with specific problems. The computer package to be described in the next chapter is designed to remedy some of these drawbacks.

4. THE COMPUTER PACKAGE

This chapter describes the major features of a computer package designed to alleviate some of the shortcomings with existing methods for assessing and using multiattribute utility functions. The package is referred to by the mnemonic MUFCAP standing for "multiattribute utility function calculation and assessment package." Steps customarily followed in obtaining and using a MUF are presented with a description of the MUFCAP commands appropriate in performing the particular step. Command usage is illustrated in Chapter 5. A concise summary of these commands is in Appendix A and the program listing is in Appendix B.

4.1 Commands to Structure the Utility Function

Structuring a utility function consists of specifying a functional form, its attributes, and the ranges for each of the attributes. MUFCAP has several commands for structuring a preference function. The INPUT command requests a name for the utility function and asks for the number of attributes which are arguments of this function. The package then requests a name, and a range for scalar attributes. The range consists of two numbers which bound the amounts to be considered for each attribute. To specify a vector attribute, one inputs a range with one bound equal to the other bound such as 0,0. MUFCAP recognizes this as a signal for a vector attribute and notes that the u_i associated with that attribute is a nested MUF. The package then requests the number of attributes which are arguments of this nested MUF. For each of these, a name and range is solicited. Further levels of nesting could be specified if desired and the information requested would be analogous to the material above. After a nested MUF is completely specified, the program returns to ask for the names and ranges for whatever attributes have not yet been covered in the outer MUF. When all the attributes have been input, the structure is complete and MUFCAP requests a new command from the user.

The INPUT command provides for all the bookkeeping which will be necessary for information to follow. Each k_i and u_i (including those in a nested MUF), can be assessed using the name of the attribute with which it is associated. The INPUT command is quite flexible in having no logical limit to the degree of nesting allowed.

In addition to INPUT, the package has commands for adding or deleting attributes to or from the utility function. It also has a command to facilitate "regrouping" of the attributes into various "sub-MUF's." In this way, a model for a problem can be conveniently altered in terms of different nesting schemes.

4.2 Commands to Specify the Single Attribute Utility Functions

The next step in assessing a MUF involves specifying the u_i's for the single attributes. As noted in Section 3.1, sophisticated computer programs do exist for assessing single (scalar) attribute utility functions. One could incorporate these into MUFCAP. However, for simplicity, several less sophisticated routines for assessing unidimensional utility functions (referred to as UNIF's) were developed.

MUFCAP has a command UNISET for specifying any of three UNIF types; linear, exponential, and piecewise linear. Pratt [13] considers the implications of these forms. The linear utility function implies risk neutrality. This form requires no more information than the range of the attribute. The exponential form implies constant risk aversion or constant risk proneness. It requires the specification of a certainty equivalent for a single lottery. Given this, the exponential form is fitted and scaled automatically by the program. The piecewise linear utility function is specified by providing the abscissa and ordinate values for n points ($3 \leq n \leq 15$) of the utility function. This form can be used for non-monotonic or S-shaped utility functions. These three types provide the user with the means of specifying a UNIF appropriate for many situations. More forms can easily be added to the package in the future.

MUFCAP also has commands which enable a user to quickly display the assessed UNIF for purposes of checking its appropriateness. The command UNICAL calculates the utility for one or a series of attribute levels. INVERSE calculates the attribute level corresponding to a given utility value. LOTTERY evaluates the certainty equivalent for any lottery with n consequences and their associated probabilities over that attribute, where $2 \leq n \leq 15$.

To summarize, MUFCAP has commands to conveniently set those u_i's which are UNIF's and to display them for feedback purposes to check on their reasonableness.

4.3 Commands to Specify the Scaling Constants

Using the attribute names as identifiers, MUFCAP allows the user to set the scaling constants in the MUF corresponding to each attribute. If X_i is a vector attribute, the u_i associated with it is a MUF with its own internal scaling constants. By referring to the name of this vector attribute, the user can specify the internal scaling constants for the associated nested MUF. When all the k_i's for a particular MUF have been set, the program automatically calculates the corresponding k using (3).

Once the u_i's have been evaluated, the package has several commands useful for assessing the k_i's in any particular MUF. The command INDIF2 takes as input two pairs of two indifference consequences each. These consequences can vary only in terms of the two attributes whose k_i's are the object of assessment. Then, using (2), the program computes the relative k_i's (i.e., the ratio k_j/k_i for attributes i and j) implied by the indifference pairs. With INDIF2, the user is not limited to choosing consequences which have one attribute at a least desirable level in order to determine the relative k_i's.

Once we know the <u>relative</u> k_i's, we can assign k_i's in (2) by arbitrarily setting one k_i to a fixed value and the others in terms of the fixed k_i. The command INDIF1 can then be used. It takes as input a single pair of indifference consequences and computes the k, and the magnitude of the k_i's implied by that pair and the currently assigned k_i's. It does this by computing the factor by which the currently assigned k_i's need to be multiplied to be consistent with the indifference pair just given. MUFCAP provides a routine which allows the user to multiply the currently assigned k_i's for any MUF by any factor thus resetting them. In this way, INDIF1 enables the calculation of the magnitude of the k_i's using an indifference relation instead of a lottery over all the attributes at once. For consistency checks, a new indifference pair of consequences can be input using INDIF1, which then computes the factor described above. If this factor is close to 1, the indifference pair is consistent with the currently assigned scaling factors.

Once the k_i's for a MUF have been assigned, an indifference curve (see Appendix E) over any two attributes in that MUF can be calculated with the command IMAP. IMAP permits a user to get immediate feedback on the tradeoff implications of the k_i's or indifference pairs which he has specified. He can quickly see if the points "claimed" to be indifferent really appear so to him. If not, the k_i's can be changed or other indifference pairs solicited until they represent more accurately the user's preferences for tradeoffs between those attributes. If desired, IMAP can be used in conjunction with INDIF2 and other commands to produce indifference curves over two attributes before all the other k_i's have been assessed. This is discussed in Chapter 6 and Appendix E.

4.4 Commands for Evaluating Alternatives and Sensitivity Analysis

Once the u_i's and k_i's have been set, the utility function is completely specified and can be used to evaluate alternatives. MUFCAP has commands for specifying two kinds of alternatives; certain and uncertain. For certain alternatives, which are simply consequences, uniattribute amounts are solicited until the alternative is completely described. For uncertain alternatives, at present, MUFCAP assumes probabilistic independence and requests a probability distribution function for each scalar attribute. The probability distribution function currently used is a piecewise linear approximation to the cumulative probability distribution for X_i. The user supplies n abscissa-ordinate pairs, where $2 \leq n \leq 9$ to specify the cumulative distribution. The cumulative distribution was chosen rather than the probability density function because the fractile method of assessing probabilities (see Schlaifer [15]) yields points of the cumulative distribution Other forms of probability distributions such as the Gaussian as well as probabilistic dependencies could be added to the package in the future.

The specified alternatives are given names by the user. With these names, the user may add, change or delete alternatives. He may also choose the ones which are to be evaluated by listing their names with the appropriate commands about to be described.

The command EVAL is used to evaluate (i.e., compute the expected utility) for any alternative or group of alternatives. EVAL can compute the expected utility for the overall utility function or for the utility function associated with any particular attribute. In the latter case, attribute levels in an alternative which are not arguments of the particular utility function are ignored. Typically, EVAL can be used to evaluate alternatives for the current multiattribute model. Parameters such as the scaling constants or probability distributions can then be changed and the alternatives evaluated again. In this way, we can see how sensitive the rankings are to changes in certain parameters. In a group decision-making context, different utility functions and probability estimates of group members can be used to evaluate and rank the alternatives. This might help clarify differences of opinion and suggest certain creative compromises or areas where more precise probability estimates may be needed.

The command GRAD evaluates the gradient of a utility function at any number of specified alternatives. The gradient is defined as the vector $\left(\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_n}\right)$ and indicates the direction of steepest increase in the utility function at a specified point. The gradient component tells us which attribute level changes would yield large increases in utility. This could be useful in generating improved alternatives to the current one. Of course, one must keep in mind the scales of the attributes in interpreting the gradient.

In addition to the gradient, GRAD also computes the vector $\left(\frac{\partial u}{\partial u_1}, \frac{\partial u}{\partial u_2}, \ldots, \frac{\partial u}{\partial u_n}\right)$. Each component represents the rate of change of u with respect to a change in the utility u_i. These components reveal the attributes for which an increase in its utility will yield the largest increase in u. The advantage of calculating these quantities in addition to the gradient components are (a) components can be calculated for MUF's as well as UNIF's, and (b) the unit of measurement for a uniattribute does not distort the magnitude of the component. Thus in some cases, $\frac{\partial u}{\partial u_i}$ might give a better picture of possible improved alternatives than $\frac{\partial u}{\partial x_i}$. MUFCAP makes both available.

Summarizing, EVAL permits the evaluation of alternatives, and along with routines which alter parameters, provides for sensitivity analysis. GRAD makes use of the analytical formulation of the problem to calculate quantities useful in suggesting improved alternatives to the currently specified ones.

4.5 General Command Format and Commands for Facilitating Use of the Package

MUFCAP commands are designed to be concise and are for the most part no longer than three words. These words may initiate a dialogue when more information is necessary. The input format is free, i.e., words need not begin in a particular position on the page. For many commands, the user will be prompted if he has left out a necessary word.

Mistyping causing invalid numbers on input is handled automatically by the program and a correct number is requested. Provision is made for the user to terminate a lengthy dialogue by specifying the word QUIT for the next number to be input. A new command can then be entered. In the future, a help command could be easily implemented which would explain the syntax of any other command, give definitions of terms used in the program and make suggestions concerning what kinds of steps to perform in assessing and using the MUF.

In addition to these features, MUFCAP has the facility for saving the current status of the multiattribute utility structure and the current alternatives in a file of the user's choosing to be read in at a later time. This gives MUFCAP the capability for filing away several different MUF models as well as a large number of alternatives for the same problem. It also allows the user to build up his model over many different sessions at the terminal and restore any status he has saved away with which he wishes to calculate at any particular time.

Another feature of MUFCAP is the supplying of default settings when the INPUT command is used to structure the MUF for the problem. After INPUT, the default for all MUF's is the additive form, with all the k_i's equal to each other, and for all UNIF's, it is the linear utility function. With these defaults, the user is set to calculate immediately after input. Thus feedback can begin right away without requiring the user to completely specify everything first. Scaling constants and utility functions can then be altered after observing some feedback to refine the model for the problem.

Finally, MUFCAP provides commands to print out the current status of the assessments. There are routines to display the k_i's and k for any MUF, the range and type for any scalar attribute utility function, the probability distribution of any attribute for any alternative, the multiattribute utility function structure (i.e., nesting) and the currently defined alternatives. Commands are also provided for easily changing parameters such as individual k_i's or the components of any alternative.

5. APPLICATION OF THE PROGRAM TO DIFFERENT PROBLEMS

This chapter presents several applications designed to show how MUFCAP can be used in practice. Certain application descriptions contain computer printout illustrating the use of various MUFCAP commands. Each set of computer printout is followed by a comments section which summarizes the pertinent features illustrated by the printout. Reference to Appendix A when reading the printout and comments is recommended.

5.1 A Simulated Application of MUFCAP: The Mexico City Airport

The Mexico City Airport problem concerned the decision for developing the city's airport facilities in the most "effective" manner in a multiobjective sense. The analysis which was done is described in more detail in Keeney [8]. This problem was approached using the existing methods for MUF assessment and utilized special computer programming to aid in the calculations. This section presents what might have been done if MUFCAP had been available then.

5.1.1 Attributes for the Problem

The Mexico City Airport problem was defined in terms of the following attributes:

X_1 ≡ total cost in millions of pesos

X_2 ≡ the capacity in terms of the number of aircraft operations per hour

X_3 ≡ access time to and from the airport in minutes

X_4 ≡ number of people seriously injured or killed per aircraft accident

X_5 ≡ number of people displaced by airport development

X_6 ≡ number of people subject to a high noise level; (i.e., 90 CNR or more)

To incorporate time effects of building the airport, attributes were defined using present values or averages where appropriate. The capacity attribute X_2 had to be made a function of capacity for 1975, capacity for 1985, and capacity for 1995, and thus it was a vector attribute.

5.1.2 Summary of the Method Used in the Problem

After verifying assumptions concerning preferential and utility independence and ascertaining the appropriateness of the multiplicative model, assessments were begun. First, the fractile method was used to obtain probability distributions for all of the alternatives under consideration. Probabilistic independence was assumed to simplify calculations. Then uniattribute utility functions were assessed for all eight scalar attributes. The k_i's were assessed using the lottery over all the attributes illustrated by Question I in Section 3.2 for both the overall MUF and nested capacity MUF. Consistency checks on the relative k_i's involving tradeoffs of two attributes at a time (see Question II, Section 3.2) were also employed. Special computer programs and graphic displays were developed for evaluating alternatives and sensitivity analysis. For sensitivity analysis, the program allowed changes in (a) the endpoints for the fractile cumulative probability distributions and (b) in the scaling factors k_i. The shapes of the utility functions or the cumulative probability distributions could not be changed without programming adjustments.

5.1.3 A MUFCAP Approach to the Mexico City Problem

The MUFCAP approach would follow the existing methods scheme in making and verifying the preferential independence and utility independence assumptions. The INPUT command would structure the multiplicative function giving names such as "cost" and "access" to the various attributes along with ranges for the attribute amounts. Capacity would be put in as a nested MUF.

Alternatives would be specified by inputting the nine-point assessed fractile distribution for each uniattribute of a particular alternative. Utility functions for single attributes would be specified using any of the three forms available in MUFCAP.

Assessment of the k_i's could be accomplished without depending upon the supplying of the probability for a lottery over all the attributes as was done. Pairs of indifference points for two attributes would be fed into MUFCAP to immediately produce indifference curves for examination and verification by the decision maker. In this way, the relative k_i's would be established with the aid of feedback. The magnitude of the k_i's could be established using INDIF1 (see Section 4.3), so a lottery over all the attributes could be avoided for this purpose. A good consistency check would be provided by comparing the magnitude of the k_i's implied by each method. Using MUFCAP, all of the initial assessments could be made and stored for later use. The assessments would have been made with the aid of immediate feedback and with no need for very difficult lottery questions.

After the initial assessments, alternative evaluations and sensitivity analysis could be performed immediately with no need for special programming. Fractile distributions and utility function shapes could also be altered without programming adjustments. The different assessments of various individuals and groups could have been filed away for later reference using MUFCAP's filing capability.

In addition, other possibilities could have been explored with a minimum of extra effort. New attributes such as air pollution and political effects could be added into the analysis with no special programming. The gradient calculation capability may have been used to support other alternatives for exploration and development. If the preferential independence of some attributes are questioned, different nesting schemes could be tried to see if the ranking of the alternatives would be affected. Thus MUFCAP could have provided the analysis that was performed with no special programming and might have been used to explore variations of more parameters, other multiattribute nesting schemes, and additions of new attributes.

5.1.4 Mexico City Airport Illustrations

```
logon alan size(300) nono
ENTER PASSWORD FOR ALAN-

M20225.11940 ACCOUNT FUNDS ARE LOW. SEE USER ACCOUNTS.
ALAN LOGON IN PROGRESS AT 10:33:40 ON APRIL 29, 1975
NO BROADCAST MESSAGES
READY

allocate file(mexico) dataset(mexico)
READY
call mufcap
TEMPNAME ASSUMED AS A MEMBER NAME

COMMAND WORD AND FILE NAMES MUST BE IN CAPS .
COMMAND? :
```

Illustration 1

```
            READ MEXICO

COMMAND? : DEBUG

    STRUCTURE FOR mexico
 cost           0.480
 4.00000E+03                5.00000E+02            1
 capacity       0.600
 cap75          0.300
 5.00000E+01                1.30000E+02            1
 cap85          0.500
 8.00000E+01                2.00000E+02            1
 cap95          0.400
 1.00000E+02                2.50000E+02            1
 access         0.100
 9.00000E+01                1.20000E+01            1
 safety         0.350
 1.00000E+03                1.00000E+00            0
 displacement   0.130
 2.50000E+05                2.50000E+03            0
 noise          0.180
 1.50000E+03                2.00000E+00            1
COMMAND? :
```

Illustration 2

```
DISPLAY mexico

LISTING OF K FACTORS
 cost              0.480
 capacity          0.600
 access            0.100
 safety            0.350
 displacement      0.180
 noise             0.180
 BIGK=   -0.377 SUM K'S =    1.890
COMMAND? : DISPLAY capacity

LISTING OF K FACTORS
 cap75             0.300
 cap85             0.500
 cap95             0.400
 BIGK=   -0.453 SUM K'S =    1.200
COMMAND? : DISPLAY access

 RANGE:     90.000    12.000
UTYPE IS CONSTANT RISK   U(X)=B(1-EXP(-CX))
 B=   1.439 C=    1.188  VARIABLE NORMALIZED
 RISK AVERSE
COMMAND? :
```

Illustration 3

```
UNISET access CR

INPUT ANY 50-50 LOTTERY IN THE FORM OF C.E.,Q1 & Q2.  PLEASE
:
62 12 90

COMMAND? : UNICAL access
 U(     90.000)=     0.000
 U(     74.400)=     0.304
 U(     58.800)=     0.544
 U(     43.200)=     0.733
 U(     27.600)=     0.882
 U(     12.000)=     1.000
COMMAND? : INVERSE access 2

:
.25 .75
     77.463=INV(    0.250)
     41.617=INV(    0.750)
COMMAND? :
```

Illustration 4

```
LOTTERY access 3

LOTTERY ENDPTS. PLEASE?
:
20 40 60

CORRESP. PROBABILITIES PLEASE?
:
.3 .4 .3

 CE FOR LOTTERY=      41.816
COMMAND? :
```

Illustration 5

```
            ALTLIST
            allone        allhalf       a3

cost            500.000      2250.000       500.000
cap75           130.000        90.000       130.000
cap85           200.000       140.000       200.000
cap95           250.000       175.000       250.000
access           12.000        51.000        12.000
safety            1.000       500.500      1000.000
displaceme     2500.000    126250.000    250000.000
noise             2.000       751.000      1500.000

CERT EQUIV. TABLE FOR PROB ALTERN
NO PROB. ALTERN.
COMMAND? :
```

Illustration 6

```
           EVAL mexico
 allone          1.000
 allhalf         0.841
 a3              0.855
COMMAND? : EVAL mexico allhalf
 allhalf         0.841
COMMAND? : EVAL capacity
 allone          0.999
 allhalf         0.805
 a3              0.999
COMMAND? : EVAL access
 allone          1.000
 allhalf         0.644
 a3              1.000
COMMAND? :
```

Illustration 7

```
KSET mexico ADD

 BIGK=    0.000
COMMAND? : DISPLAY mexico

LISTING OF K FACTORS
 cost              0.254
 capacity          0.317
 access            0.053
 safety            0.185
 displacement      0.095
 noise             0.095
 BIGK=   0.000 SUM K'S =    1.000
COMMAND? : EVAL mexico
 allone             1.000
 allhalf            0.679
 a3                 0.624
COMMAND? :
```

Illustration 8

```
READ MEXICO

COMMAND? : ADDALT all-fourth .25

ALTERNATIVE all-fourth   SPECIF.
COMMAND? : EVAL mexico all-fourth
 all-fourth       0.616
COMMAND? : DROPALT all-fourth

COMMAND? :
```

Illustration 9

```
INDIF1 safety cost

INPUT AN INDIFFERENCE PAIR PLEASE
:
800 1000 300 2500

 IMPLIED NEW K'S FACTOR(S)     0.970 (    4.780)
 IMPLIED NEW BIGK=    -0.859
COMMAND? :
```

Illustration 10

```
INDIF2 safety cost

INPUT 2 INDIFFERENCE PAIRS PLEASE
:
800 1000 300 2500
+:200 3500 750 2500

 BIGK=  -0.267/K(safety       )
INDIF PAIR YIELDS INFO ABOUT REL K'S
 REL K CHECK. CURRENT RATIO cost         TO safety= 1.571
 IMPLIED RATIO =    1.397
COMMAND? :
```

Illustration 11

```
IMAP safety cost

INPUT INDIF PT. THROUGH WHICH CURVE WILL PASS: 500 2500

INPUT NUMBER OF PTS. FOR MAP: 5

 INPUT safety       VALUES FOR MAP
:
300 400 500 600 700

INDIFFERENCE PTS
 (   300.000,  2922.539)
 (   400.000,  2715.855)
 (   500.000,  2500.002)
 (   600.000,  2272.636)
 (   700.000,  2030.779)
 UTIL FOR CURVE WITH OTHER ATTR. AT 0       0.444
COMMAND? :
```

Illustration 12

```
INTERBK mexico

 capacity     BIGK=    -0.453  INTERBK=    -0.526
COMMAND? :
```

Illustration 13

5.1.5 Comments on Mexico City Airport Illustrations

Illustration 1

The user logs in, sets up a data file which will be used and invokes MUFCAP.

Illustration 2

The status of preferences and alternative specifications in the file MEXICO is read in. The multiattribute utility function structure is displayed.

Illustration 3

Characteristics of MUF's and UNIF's associated with various attribute names are displayed. Mexico and capacity have associated MUF's while access has an associated UNIF.

Illustration 4

An example of setting a UNIF is shown. The UNIF for access is assumed to be of the constant risk type. The UNIF is fitted in response to the 50-50 lottery certainty equivalent request. UNICAL tabulates the UNIF for various amounts of access. INVERSE tabulates the amounts of access having certain utility values. The amount of access having utility =.25 should correspond to the certainty equivalent for the 50-50 lottery between the amount of access having utility =.5 and that having utility = 0. A check with Keeney [8] shows that the fit for access appears to be very good.

Illustration 5

An example using the LOTTERY command is shown. A certainty equivalent for the 3-consequence lottery is output.

Illustration 6

Several "certain" alternatives are displayed. "allone" has all the attributes at their best levels. "a_3" has cost, capacity and access at their best, and safety, displacement and noise at their worst. "allhalf" has all the attributes halfway between their range limits. There are no uncertain alternatives in this current status.

Illustration 7

This illustrates the use of the EVAL command. The overall utility function mexico is evaluated for all the alternatives and then only for allhalf. The MUF associated with capacity is evaluated for all the alternatives. The UNIF associated with access is similarly evaluated.

Illustration 8

These lines illustrate a little sensitivity analysis. The KSET command makes the overall utility function "mexico" additive but maintains the same relative k_i's. The alternatives are then evaluated. Notice the change in rank between "allhalf" and "a_3" with the additive model as opposed to the original model.

Illustration 9

The original model is restored. An alternative all-fourth is added, evaluated and dropped.

Illustration 10

A check on the magnitude of the k_i's is performed using INDIF1 and a single indifference pair. The check shows that the current k_i's agree well with the indifference-pair check.

Illustration 11

An independent check is made on the relative k_i's concerning "cost" and "safety." The implied ratio agrees well with the current ratio.

Illustration 12

An indifference curve is tabulated between "cost" and "safety."

Illustration 13

A check is made on the necessity for nesting capacity as opposed to using the attributes cap75, cap85 and cap95 along with the others in a single 8-attribute multiplicative form. The check shows that <u>without</u> nesting the approximation to the tradeoffs among the attributes would be pretty good. (See Appendix E for a more detailed explanation.)

5.2 Evaluation of a Computer Time-Sharing System

This section concerns an example relevant to a manager of a time-sharing system in formulating a MUF to evaluate different courses of action. The data and formulation is based on Grochow [4]. This problem was also approached using existing methods and special computer programming. A possible MUFCAP approach is presented here.

5.2.1 Attributes for the Problem

The following attributes were used in the time-sharing problem:

A ≡ Availability measured in percentage of successful logins

RT ≡ Average response time to majority of trivial requests in seconds

RC ≡ Average response time to majority of compute-bound requests

5.2.2 Summary of the Method Used in the Problem

The first stage of analysis was to determine what utility independence relationships existed among the attributes. It was found that RC was utility independent of A, and RT was utility independent of A and RC. But A was not utility independent of RT or RC, and RC was not utility independent of RT. Examination of the attributes showed that certain forms of independence were not to be expected. For example, tradeoffs between RC and A may depend on RT since it hardly pays to be able to log in more often if RT is very bad.

Grochow's approach was to formulate an overall utility function involving seven conditional one-attribute utility functions and effectively assessing six scaling constants using existing methods.

5.2.3 A MUFCAP Approach

A possible MUFCAP approach to this problem would be to try, as an approximation, the following nesting scheme:

$$u(a,rt,rc) = u(u_a, u_r)$$

where $u_r = u_r(rt,rc)$ and $u_a = u_a(a)$

This is the multiplicative form with u_r as a nested MUF. There are four independent scaling constants possible in this formulation. The model is assuming as an approximation that the various violations of utility independence can be ignored but that preferences for tradeoffs between availability and any response time depend on the level of the other response time. This seems reasonable since tradeoffs between response times are of concern after the user has logged in. On the other hand, the value of logging in (e.g., the amount one is willing to trade to gain a faster RC) may depend on how good RT is.

To test out this MUFCAP approach, we can calibrate the MUFCAP model using the graphical data in Grochow [4]. This data provides enough information to attempt setting of the scaling constants for the MUFCAP model. In calibrating the scalar attribute utility functions, an "average" constant risk form for each attribute was estimated from the data.

After calibrating the model, various points in the attribute space (i.e., alternatives) were evaluated and ranked to see how closely they compared to the graphical data in Grochow [4]. The results illustrated in the computer printouts following this section were reasonably close to the graphical data and seemed to justify the MUFCAP approximation scheme. The agreement seemed reasonable in spite of the fact that constant risk forms were used for the scalar attribute utility functions. The graphical data exhibited "jumps" which could be modeled by piecewise linear forms in a more refined approximation.

If one is satisfied with the MUFCAP approximation, we can immediately proceed to perform gradient calculations showing which direction one should take for maximum improvement of the current state (in the attribute space) as Grochow suggests. Also, expanding the model to include more attributes (e.g., cost) seems easier with the MUFCAP scheme than with further conditional utility functions and "corner point" (i.e., extreme value) assessments for scaling constants.

To summarize, MUFCAP, with nesting, may be used to capture the essential features of situations which may not satisfy some of the independence assumptions. When the approximation can be used, gradient calculations, sensitivity analysis and expansion of the model to include more attributes become feasible using MUFCAP.

5.2.4 Computer Time-Sharing System Illustrations

```
INPUT grochow
HOW MANY ATTRIBUTES ARE IN THIS MUF? : 2

 INPUT NAME AND RANGE FOR ATTR  1 OF UTIL FUNC grochow
:
a .1 1

 INPUT NAME AND RANGE FOR ATTR  2 OF UTIL FUNC grochow
:
response 0 0

HOW MANY ATTR. ARE IN THIS MUF?: 2

 INPUT NAME AND RANGE FOR ATTR  1 OF UTIL FUNC response
:
rt 9 2

 INPUT NAME AND RANGE FOR ATTR  2 OF UTIL FUNC response
:
rc 120 2

COMMAND? : DEBUG

     STRUCTURE  FOR grochow
 a              0.500
 9.99999E-02              1.00000E+00              0
 response       0.500
 rt             0.500
 9.00000E+00              2.00000E+00              0
 rc             0.500
 1.20000E+02              2.00000E+00              0
COMMAND? :
```

<u>Illustration 14</u>

```
          UNISET a CR

INPUT ANY 50-50 LOTTERY IN THE FORM OF C.E.,Q1 & Q2.  PLEASE
:
.7 .1 1

COMMAND? : UNISET rt CR

INPUT ANY 50-50 LOTTERY IN THE FORM OF C.E.,Q1 & Q2.  PLEASE
:
5 9 2

COMMAND? : UNISET rc CR

INPUT ANY 50-50 LOTTERY IN THE FORM OF C.E.,Q1 & Q2.  PLEASE
:
20 120 2

COMMAND? :
```

Illustration 15

```
          INDIF1 rt rc

INPUT AN INDIFFERENCE PAIR PLEASE
:
5 120 9 2

INDIF PAIR YIELDS INFO ABOUT REL K'S
 REL K CHECK. CURRENT RATIO rc          TO rt = 1.000
 IMPLIED RATIO =      0.500
COMMAND? : KSET response
 rt             = :.667
 rc             = :.333

 BIGK=     0.000
COMMAND? :
```

Illustration 16

```
          INDIF1 rt rc

INPUT AN INDIFFERENCE PAIR PLEASE
:
5 2 2 120

 IMPLIED NEW K'S FACTOR(S)     1.000 (1254.905)
 IMPLIED NEW BIGK=     0.004
```

Illustration 17

```
            ADDALT a1

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a1          SPECIF.
a              =:.5

rt             =:5

rc             =:40

COMMAND? : ADDALT a2

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a2          SPECIF.
a              =:.4

rt             =:4

rc             =:40

COMMAND? : ADDALT a3

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a3          SPECIF.
a              =:.7

rt             =:6

rc             =:40

COMMAND? : ADDALT a4

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a4          SPECIF.
a              =:.8

rt             =:7

rc             =:40

COMMAND? :
```

Illustration 18

```
           EVAL a
 a1                  0.279
 a2                  0.191
 a3                  0.501
 a4                  0.641
COMMAND? : EVAL response
 a1                  0.409
 a2                  0.511
 a3                  0.315
 a4                  0.228
COMMAND? :
```

Illustration 19

```
           INDIF2 a response

INPUT UTILITY VALUES
INPUT 2 INDIFFERENCE PAIRS PLEASE
:
.28 .41 .19 .51
+:.5 .315 .64 .23

 BIGK=    1.850/K(a             )
INDIF PAIR YIELDS INFO ABOUT REL K'S
 REL K CHECK. CURRENT RATIO response     TO a  = 1.000
 IMPLIED RATIO =     1.345
COMMAND? : KSET grochow
 a              = :.25
 response       = :.34

 BIGK=    4.824
COMMAND? :
```

Illustration 20

```
           INDIF1 a response

INPUT UTILITY VALUES
INPUT AN INDIFFERENCE PAIR PLEASE
:
.501 .315 .64 .228

 IMPLIED NEW K'S FACTOR(S)     0.976 (  -2.301)
 IMPLIED NEW BIGK=    5.239
COMMAND? :
```

Illustration 21

```
              ADDALT a5

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a5              SPECIF.
a                =:.4

rt               =:3

rc               =:40

COMMAND? : EVAL grochow a3 a4 a5
 a3 .                0.297
 a4                  0.298
 a5                  0.308
COMMAND? :
```

Illustration 22

```
              CHANGE response K .31

COMMAND? : EVAL grochow a3 a4 a5
 a3                  0.292
 a4                  0.296
 a5                  0.293
COMMAND? : CHANGE response K .34

COMMAND? : KSET grochow .75

 BIGK=    11.660
COMMAND? : EVAL grochow a3 a4 a5
 a3                  0.262
 a4                  0.260
 a5                  0.261
COMMAND? : KSET grochow 1.33333

 BIGK=     4.824
COMMAND? :
```

Illustration 23

```
              GRAD grochow a1
 a1                  0.255
ATTRIB,UTIL. GRAD COMP. AND ATTR. GRAD COMP.
 a                   0.418   3.965E-01
 response            0.454
 rt                  0.303  -4.461E-02
 rc                  0.151  -1.401E-03
```

Illustration 24

```
              ADDALT a7

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a7                SPECIF.
a               =:.76

rt              =:9

rc              =:2

COMMAND? : ADDALT a8

IS ALT. PROB? (YES OR NO): NO

ALTERNATIVE a8                SPECIF.
a               =:.1

rt              =:2

rc              =:2

COMMAND? : EVAL grochow a7 a8
 a7                  0.338
 a8                  0.340
COMMAND? : CHANGEALT rc a7

rc              =:100

COMMAND? : CHANGEALT rc a8

rc              =:100

COMMAND? : EVAL grochow a7 a8
 a7                  0.148
 a8                  0.228
COMMAND? :
```

<u>Illustration 25</u>

```
            ADDALT a9

IS ALT. PROB? (YES OR NO): YES

ALTERNATIVE a9              SPECIF.
HOW MANY FRACTILE PTS. (INCL 0 AND 100%)  FOR a
(2<=N<=9) : 2
INPUT THE CUM FUNC F(X). X'S FIRST THEN F(X)'S
:
.1 1

:
0 1

HOW MANY FRACTILE PTS. (INCL 0 AND 100%)  FOR rt
(2<=N<=9) : 2
INPUT THE CUM FUNC F(X). X'S FIRST THEN F(X)'S
:
2 9

:
0 1

HOW MANY FRACTILE PTS. (INCL 0 AND 100%)  FOR rc
(2<=N<=9) : 2
INPUT THE CUM FUNC F(X). X'S FIRST THEN F(X)'S
:
2 120

:
0 1

COMMAND? : EVAL grochow a9
 a9                 0.281
COMMAND? : ADDALT a10 .5

ALTERNATIVE a10             SPECIF.
COMMAND? : EVAL grochow a9 a10
 a9                 0.281
 a10                0.232
COMMAND? :
```

Illustration 26

5.2.5 Comments on Computer Time-Sharing System Illustrations

Illustration 14

The INPUT command is used to structure the multiattribute utility function. "Response" is a nested MUF. The DEBUG command shows the defaults present after INPUT.

Illustration 15

All the UNIF's are set using the constant risk form.

Illustration 16

The relative k_i's are determined between "rt" and "rc" using INDIF1. Notice how INDIF1 can aid in calculation when a Type II Question (see Section 3.2) is asked. The KSET command sets the relative k_i's based on the output from INDIF1. The absolute k_i's are not yet known.

Illustration 17

INDIF1 is used to determine the magnitude of the k_i's. The results show that our current setting is close to the one implied by these indifference points. The nested MUF "response" has thus been assessed.

Illustration 18

Several alternatives are set up using ADDALT. These will be used in assessing the scaling constants for the MUF "grochow."

Illustration 19

The utility values for "a" and "response" are evaluated for the alternatives. These will be used in the subsequent commands; e.g., $u_a(.5) = .279$

$u_r(5, 40) = .409$

alternative a1 is the consequence (.5, 5. 40)

Illustration 20

INDIF2 is used to assess the relative k_i's between "a" and "response." We must use utility values in specifying indifference points because "response" is a vector attribute; e.g., to specify that (.5, 5, 40)~(.4, 4, 40) we say (.279, .409)~(.191, .511) (See Appendix A, Section A.2). The KSET command is used to set up the relative k_i's implied by the output from INDIF2.

Illustration 21

INDIF1 is used to assess the magnitude of the k_i's for the MUF "grochow." The results show that our current settings are reasonable. The MUF "grochow" is now set.

Illustration 22

EVAL is used to rank the alternatives. The rankings here are essentially the same as in Grochow.

Illustration 23

Some sensitivity analysis is performed. The CHANGE command alters the scaling constant for response. The alternatives are evaluated and the rankings have changed. The original model is restored and the magnitude of the k_i's for "grochow" are changed using KSET. Again, the rankings change from the original model. The original model is restored.

Illustration 24

The gradient for "grochow" is calculated at the alternative a_1.

Illustration 25

Two "indifferent" alternatives under the current model are set up using ADDALT. The CHANGEALT command is used to alter the common value of "rc" for the two alternatives. They are evaluated again and are no longer indifferent. This shows that tradeoffs between "a" and "rt" depend on the level of "rc." Our nesting scheme has captured this facet of the problem. The tradeoff value of logging in is degraded by the poorer "rc."

Illustration 26

A probabilistic alternative is input and evaluated. In this case, uniform distributions are implied by the cumulatives which are input.

Although not shown on the computer printout, the following table is a comparison between the MUFCAP approximation and the graphs in Grochow [4]. (The scales in Grochow [4] are not easy to interpret and the following uses my interpretation.)

Consequence (a, rt, rc)	U_{MUFCAP}	$U_{GROCHOW}$
(1,9,2)	500	500
(1,9,120)	250	290 (?)
(1,2,120)	750	750
(.5,9,2)	221	250
(.5,9,120)	70	60
(.5,2,120)	373	383
(.5,2,2)	524	494
(1,5,120)	500	490
(1,5,2)	750	740
(1,2,40)	807	915
(1,9,40)	306	282

Table 5.1

A Comparison of MUFCAP and Grochow Utility Functions

5.3 The Comparison of Dial-A-Ride Algorithms

This section presents elements of a MUFCAP application to decide between two algorithms used by a computer to schedule Dial-A-Ride service which is a mode of transportation being tried in certain cities today. The presentation is confined to aspects of the application which illustrate further features of MUFCAP.

5.3.1 Attributes for the Problem

The attributes of interest in this section are those for which preferences are not monotonic. These include:

pickup time deviation ≡ the difference in minutes between the promised pickup time and the actual pickup time

travel time deviation ≡ the difference in minutes between the promised delivery time and the actual delivery time

The utility functions for these attributes were assessed and input into MUFCAP making use of the piecewise linear form. Two other attributes along with these were used in making up the overall utility function (see Turnquist [17]).

The utility function parameters were assessed and several certainty alternatives were evaluated to check that the utility function reasonably represented the preferences of the person being assessed. For this application, however, the actual alternatives to be evaluated were outputs from a stochastic simulation program. One hundred outputs for each algorithm were evaluated using the utility function assessed via MUFCAP. That is, once the utility function was assessed, it was coded up in a separate program to process the output from the simulation runs. An estimate of the expected utility which was the criteria for choosing between the algorithm was obtained by taking the average of the one hundred output evaluations. This represents a way for evaluating the expected utility in a case where the attributes are not probabilistically independent of each other. Although the whole evaluation was not done through MUFCAP, this method for handling a case in which probabilistic independence did not hold was not too difficult. This was because sensitivity analysis could still be fairly easily performed since the utility function had been conveniently parameterized into the multiplicative form via MUFCAP. It is conceivable that MUFCAP could be given an option for reading an output file from a simulation model in a future version of the program. Then evaluations could be performed within MUFCAP.

The results of the evaluation showed that one algorithm was slightly superior to the other over a wide range of parameter variations and different simulation runs. Currently, a more ambitious effort is being undertaken to assess public preferences for attributes germane to this problem as opposed to one particular individual's preferences.

5.3.2 Dial-A-Ride Illustrations

```
UNISET pickdev PL
HOW MANY PTS. IN UTIL FUNC? : 5

INPUT THE FUNC., X'S FIRST THEN U(X)'S
:
-30 0 10 15 30

:
.75 1 .75 .5 0

COMMAND? : UNICAL pickdev
 U(     30.000)=      0.000
 U(     18.000)=      0.400
 U(      6.000)=      0.850
 U(     -6.000)=      0.950
 U(    -18.000)=      0.850
 U(    -30.000)=      0.750
COMMAND? :
```

Illustration 27

```
              INVERSE pickdev
      30.000=INV(    0.000)
      27.000=INV(    0.100)
      21.000=INV(    0.300)
      15.000=INV(    0.500)
      11.000=INV(    0.700)
       4.000=INV(    0.900)
       0.000=INV(    1.000)
COMMAND? :
```

Illustration 28

5.3.3 Comments on Dial-A-Ride Illustrations

Illustration 27

A non-monotonic utility function for pickup deviation is input using a piecewise linear utility function. Some sample utility function values are tabulated using UNICAL. The range of the function was input as 30, -30.

Illustration 28

The INVERSE function shows only positive deviations as attribute levels having certain utility values. This is because MUFCAP, for piecewise linear forms, searches the range from the 1st range value to the 2nd range value until it finds a level with the appropriate utility. This same feature holds true when an indifference curve is generated. This has no effect on the proper evaluation of alternatives.

5.4 A Sampling of Problems to which MUFCAP Has Been Applied

This section surveys some of the areas where MUFCAP has been used in a preliminary manner to develop multiattribute utility functions. In all these applications, the various commands and procedures already illustrated in previous sections were employed. Chapter 6 further discusses some of the things which were learned from these experiences.

5.4.1 Evaluating Health Plans

Four attributes were formulated for evaluating health plans. These were convenience, quality, cost and personalness of the service. Psychometric measures were developed for each of the attributes and questionnaire assessments were used to estimate the utility function parameters. MUFCAP was then used to calculate k in the multiplicative form and generate indifference curves between certain attributes (see Hauser and Urban [6]).

5.4.2 Evaluating Policies for Dealing with Prostitution in the Boston Area

A class project in a decision analysis course at MIT involved evaluating five options for dealing with the question of legalizing prostitution in the Boston area. These options were strict prohibition, toleration or benign neglect, regulation of prostitution, licensing of individual prostitutes and decriminalization. The attributes were chosen to reflect the prostitute's position, the public attitude, the economics of the options, the criminal justice system's opinion of the options and the political implications of the choices.

The class divided into groups which concentrated on the specific attribute areas defined above. The groups assessed expected utility values for their individual attributes for each option. Pseudo-attributes consisting of the five attribute areas each measured by a utility value on a linear scale from 0 to 1 were then input into MUFCAP.* A sensitivity analysis concerning ranking of the options was then performed on the magnitude of the k_i's. It showed that regulation was the preferred policy for the particular relative k_i's used in this problem over a large range of their magnitudes.

This application illustrates how a complex problem can be subdivided into smaller problems and the outputs from these combined in an overall utility function. In some cases, the overall decision maker may not be familiar with the specific attributes used to represent the objectives of a particular area or group. If he has a "feel" for associating utility with that group's preferences, however, he may be able to estimate the scaling constants and conduct reasonable sensitivity analyses in a manner analogous to what was done in the class project on prostitution.

---

*Actually a very early version of MUFCAP. This application was repeated with a later version for validation of the results.

5.4.3 Evaluating Police Dispatching and Assignment Policies

Attributes for evaluating police assignment and dispatching strategies include cost per person per year, response time to various priority calls and distribution of the workload among the different police units. While models have been formulated to predict what workloads and response times will result from implementing certain strategies, work is just beginning on evaluating the tradeoffs between the various attributes in the problem which go into deciding upon a strategy. MUFCAP is now being used in preliminary attempts to structure a utility function for such strategy evaluations.

5.5 Other Problem Setting Amenable to MUFCAP

Many problems which can be cast as multiobjective decision making problems involving risk might be amenable to analysis using MUFCAP. This section presents some examples of current problems and how they might be structured for MUFCAP analysis.

5.5.1 Nuclear Power Plant Siting and Setting Standards for Air Pollution Control

This subsection mentions two areas which have been formulated as multiattribute decision-making problems in the literature. In Keeney and Nair [10], general objectives are described for a nuclear power plant siting decision. These include minimizing environmental damage, maximizing human health and safety, providing quality service for the customer and maximizing the economics of the company. Explicit attributes might be level of radiation per person for human safety and service interruption in days for quality of service to the customer.

Keeney and Ellis [1] describe the decision problem faced by New York City in legislating acceptable levels for sulfur content in fuel to be consumed by industry. The problem is organized in detail into a multiattribute utility function structure including attributes which reflect such objectives as the cost to the city of any plan, and effects on the health of the residents.

In both these cases, good descriptions of how to formulate the problem are available. The actual assessment in detail or implementation of the formulations appear to be possible through the use of MUFCAP.

5.5.2 Anti-Stagflation and Energy Policy Decisions

Two of the most important multiobjective problem areas facing the United States are how to deal with the economic and energy crises currently plaguing the country. A crucial aspect in these problems has been deciding what tradeoffs to make between apparently competing objectives.

In the economic area, some of the measures for objectives include the unemployment rate, the consumer price index and growth in the GNP. The energy area includes cost of fuel and degree of dependency upon other nations. In addition, the problem of sharing the burden equitably among the different groups in the United States such as labor, management, minorities, lower, middle and upper classes, residents of certain geographical regions, social security recipients, etc., lead to explicit consideration of the tradeoffs between these different groups in trying to decide upon a policy.

These problems appear to be very difficult and a formal analysis such as could be attempted with MUFCAP might shed some light on comparing alternative solutions. Perhaps as important, differences of opinion concerning tradeoffs among the objectives might also be clarified.

5.5.3 Multiobjective No-Risk Contexts

In situations where no uncertainty is present, multiattribute utility theory, of course, is still valid. In these situations, however, the theory of value functions (ordinal) rather than utility functions (cardinal) are applicable as well. With three or more attributes, preferential independence implies that an overall value function exists which is a weighted sum of the individual value functions assessed over the attributes. How one assesses value functions as opposed to utility functions will not be discussed here.

MUFCAP, while designed to implement utility theory, can nevertheless be used to implement a value function approach to a problem. The value functions for the individual attributes are input as if they were utility functions using the UNISET command. The scaling constants are input using the KSET command and the overall "value" function is deliberately made additive also using KSET.

MUFCAP can then be used to evaluate alternatives or generate indifference curves. Different functions based on the preferences of different people can be compared using MUFCAP's filing capability and sensitivity analysis varying the scaling constants and value functions can also be tried.

6. AREAS FOR IMPROVEMENT AND FUTURE RESEARCH

This chapter discusses various improvements which might be made to MUFCAP. Many of these were anticipated in the sense that MUFCAP should be considered a first edition or a basis on which to improve. In addition, through the use of MUFCAP, other new ideas for routines and commands emerged.

Besides technical improvements which can be made to the program, several theoretical and practical issues concerning types of assessment questions arose during the course of testing and using MUFCAP. These issues are also discussed in this chapter.

6.1 Ideas for Improving MUFCAP as a Computer Program

MUFCAP, being a computer program, can be improved in the ways that computer programs are generally improved. These encompass four general areas.

The first would be more testing and debugging of the existing routines. Currently, a bug exists in the LOTTERY command which was intended to perform a particular calculation when there is a 2-consequence lottery.. This bug can be easily corrected when a later version is compiled, hopefully including more than just the fix for this bug.

The second area concerns better program documentation. In programming MUFCAP, less attention was paid to documenting routines as opposed to getting them to work properly. Hopefully, this thesis and the program listing are sufficient for a knowledgeable programmer to successfully modify MUFCAP. In addition, the documentation for program usage could be expanded into a more extensive user's manual should MUFCAP ever attain the status of a standard package for decision analysis.

A third improvement involves making the program more "fail-safe" for the user. Many precautions have already been taken to "protect" the user against leaving out necessary input or making input mistakes. There remains room for improvement, however. One special area concerns generating an indifference map involving an attribute with a risk averse exponential form. With this form, there is a limit to the utility one could obtain even if one had an infinite amount of a desirable attribute. If an indifference point is given and another is desired having less of one desirable attribute but more of the risk averse one, it is possible that no amount of that attribute will make the new point indifferent to the old one. In this case, MUFCAP tries to extrapolate by taking the log of a negative number causing one to exit from the program. Thus, one should always save the status periodically so in case one is forced to exit from MUFCAP, the program can be invoked again and the status restored.

Finally, the output could be made more aesthetic and easy to understand. This improvement is a necessary complement to having better documentation.

6.2 Expanding Old and Adding New Routines

Several ideas for better routines concern the areas of generating indifference curves more automatically, expanding the number of available scalar attribute utility function forms, providing an easier way of specifying probabilistic distributions and providing for analysis of alternatives where probabilistic independence need not be assumed. There is also the area of more automatic sensitivity analysis.

One should be able to generate an indifference curve between two attributes which are preferentially independent of all the other attributes after obtaining two sets of indifference pairs. Currently, this can be done in MUFCAP in three stages. First, INDIF2 is used to obtain the relative scaling constants and k in terms of one of the scaling constants. Then KSET is used with the OVERIDE option to set one scaling constant arbitrarily, the second in terms of the first, and k in terms of the first. Then, IMAP is used to generate indifference curves. This procedure is one which is often requested because indifference curves are a valuable source of feedback. A needed improvement would be to have INDIF2 stay in an indifference curve generating mode and automatically generate indifference curves for the user right after input of the indifference pairs. This should be fairly easy to implement. (Alas, a computer program must be limited to some extent so a version can finally be produced.)

MUFCAP has three scalar attribute utility forms and more could be added. These might include decreasingly risk averse forms based on a single parameter which are very convenient to assess or multi-parameter forms.

Currently, specifying probabilistic alternatives, especially for a many-attribute problem is laborious. More automatic setups of these alternatives are possible. Suggestions include setting all attributes with uniform density functions over their ranges automatically or setting them all with normal distributions about their centers and having the range limits be several standard deviations away. Also, having set up a probabilistic alternative, one should be able to copy it into another alternative and then have the ability to change a particular component. A method of handling probabilistically dependent alternatives has already been discussed in Section 5.3. Another improvement would be provision for discrete probability functions for the scalar attributes.

Presently, in doing sensitivity analysis, a user must input the parameter changes and then evaluate alternatives. The program could be made to vary a parameter over a range and automatically evaluate alternatives, or generate other feedback. This would enable the user to perform sensitivity analysis more rapidly.

6.3 Making MUFCAP Easier to Use

MUFCAP requires an intermediate "decision analysis person" to operate the program, ask assessment questions, and discuss the feedback implied by the output. The program might be upgraded to (a) "prompt" what assessments should be made at various stages of the MUF development, and (b) print more interpretation about what certain output numbers mean. More will be mentioned in this vein in later sections of this chapter.

To develop an interface dialogue so that the program would be completely self-explanatory to decision makers in any field would take a lot of testing and work. This might not be desirable either since discussion with a decision analyst should not necessarily be avoided. I have found that users not "immersed" in multiattribute utility theory were nevertheless able to "order me" in rapid-fire succession about what to do next. Setting up the initial model is the hardest part. But sensitivity analysis should be fairly easy for a "layman" once he is reasonably satisfied with the initial model.

Another suggestion has been to put a graphics capability into MUFCAP. This would enable the program to draw utility functions and indifference curves displaying their shape to the user. Using a MUFCAP with graphics would be more stimulating in that information would be presented to the user in a more concise manner. Gradient vectors might even be presented on a representation of a utility surface. Also, changes to utility functions, indifference curves or parameters could be input via a light pen or a joystick cursor enabling the user to conduct sensitivity analysis with his hand. An advantage of the non-graphics current package is that it can be run on a portable terminal.

6.4 Assessment Question Issues

Although MUFCAP is a definite aid in MUF assessment, a great deal of discussion and patience is still necessary to solicit accurate information from the decision maker. The results output by MUFCAP are completely based upon the input information. In the early use of the package, it was tempting to input numbers which were not reasonably arrived at just to see some output from the package. The output was often nonsensical from the viewpoint of certain assumptions about the multiplicative form. For example, if two pairs of indifference points are input to INDIF2, MUFCAP essentially solves simultaneous equations of the form $Ax_1+By_1+Cx_1y_1 = Ax_2+By_2+Cx_2y_2$ where, for the multiplicative form, A corresponds to k_i, B to k_j, and C to kk_ik_j. In solving these equations, however, arbitrary input can lead to arbitrary values for k and k_j in terms of k_i. For example, sometimes the implied k is equal to $-2/k_i$ which is not allowed for the assumptions of the multiplicative form as defined in Keeney[9] since it is less than -1. When this happens, new pairs of indifference points should be input.

Besides leading to nonsensical output, certain forms of indifference pair inputs can given very inaccurate results. Indifference questions involving extreme attribute levels are very difficult to consider. However, indifference questions involving consequences which are not very different from each other in terms of attribute levels can give very inaccurate results. This is because it is hard to discriminate between what is preferred and what is indifferent. The best questions seem to be those in which the indifference points are spread about the middle of the attribute ranges and in which attribute amounts vary halfway between the middle and extreme end of the range. Also, specifying two indifference pairs which share a consequence point in common [e.g., $(a_1,b_1) \sim (a_2,b_2)$ and $(a_1,b_1) \sim (a_3,b_3)$] seem less prone to giving nonsensical results.

In using MUFCAP, certain indifference pairs appear to be more "robust" than others in terms of the implied relative scaling constants. For example, the type II question mentioned in Section 3.2 is very robust in the sense that if $[(x_i', x_j^\circ) \sim (x_i^\circ, x_j')]$ implies certain relative scaling constants, $[(x_i' + \delta x_i, x_j^\circ) \sim (x_i^\circ, x_j')]$ implies almost the same relative scaling constants provided δx_i is small compared to the range. This, however, is not always the case when INDIF2 is used with two sets of indifference pairs. In cases where the difference in the consequences is relatively small and it appears as if one of the scaling constants is more than twice the other, a δx_i which is small can lead to large changes in the implied relative scaling constants.

Fortunately, one can test the robustness of the relative scaling constants implied by two sets of indifference pairs using MUFCAP. One merely varies one of the attribute amounts by a small percentage and observes if the implied relative scaling constants are vastly different from those implied by the original sets of indifference pairs. A nice improvement to MUFCAP would be for the program to automatically test the robustness of certain inputs by performing the appropriate variations and displaying the results for the user. More about this will be discussed in the next section.

6.5 Areas for Future Research

One area for future research concerns the specification, from a theoretical point of view, of assessment questions involving indifference pairs which are "robust" as discussed in Section 6.4 A starting point might be to examine the indifference curves which are hyperbolas in the utility plane $u_i \times u_j$. (See Appendix E.) We could imagine having three points on an indifference curve and then displacing one of the points and plotting a new indifference curve. How much the new curve differs from the old might depend on the spread of the initial three points.

A second area for examination is how to interpret varying output during sensitivity analysis. When several pairs of indifference points are input, the implied k is often different. Interpreting what constitutes a significant difference is not very precisely defined. For example, is a k = -.50 significantly different from a k = -.80. Where the relative scaling constants are concerned, variations here are directly related to the size of the differences in attribute amounts necessary to maintain certain indifference relationships. But where k is concerned, it is difficult to tell where the differences will be because k = -.50 as opposed to k = -.80.

MUFCAP can be used to empirically examine what differences result when certain variations are perceived in the value of k. In addition to aiding in such sensitivity analysis, MUFCAP might also aid in researching the area of robust assessment questions and interpreting what constitutes significant variations in parameters implied by the answers to assessment questions.

A third topic for future research would be methods of verifying preferential and utility independence assumptions. In order to use the multiplicative form, we must test that the appropriate independence assumptions are satisfied. This can be done by asking a lot of tradeoff questions and lottery-type questions (see Keeney [8]). It can often be laborious to rigorously verify the requisite assumptions, however.

MUFCAP provides another means for testing preferential independence. If tradeoffs between attributes i and j imply a negative k, but tradeoffs between j and l imply a positive k, then obviously the set of attributes i, j and l cannot be combined into a single multiplicative form and are not preferentially independent. Earlier in this section, we discussed the problem of what constituted a significant difference in the value of k implied by indifference pair inputs. If this were known, preferential independence could be tested by seeing if several indifference pair inputs implied the same k within a certain "confidence interval." If so, we could assume more confidently that preferential independence was indeed present.

6.6 Summary of the Chapter

This chapter discussed a variety of areas for improving MUFCAP and for future research. These included improving and further documenting the computer code and expanding and adding new routines to improve feedback and make specifications easier. The issues in asking the "best" kind of assessment questions were discussed. These included asking questions which would have "robust" answers and not yield results too sensitive to small deviations in the answers. Areas for future research concerned these issues of robust yet reasonable assessment questions, how to interpret, in a statistical-like fashion, variations in parameters implied by certain indifference pair inputs and further ways of verifying certain independence assumptions.

7. SUMMARY AND CONCLUSIONS

This chapter summarizes the main aspects of the computer package MUFCAP. The current version provides the basic features necessary to assess and use multiattribute utility functions on complex decision problems. In particular, it permits one to use realistic and simple questions in assessing the decision maker's preferences, in addition to the "extreme value" types of questions previously used for computational reasons. MUFCAP provides for (a) a variety of immediate feedback of implications of the decision maker's responses, (b) evaluation of alternatives and sensitivity analysis, and (c) analyzing differences of preferences and judgements which constitute differing models of the same problem such as might arise among various individuals in a decision-making group.

The present MUFCAP should be considered a first edition, a basis on which to improve. In this regard, many possible improvements have been suggested in this thesis such as new routines for (a) providing more readable output, perhaps even graphical displays, (b) promoting easier feedback such as more automatic computation of the implications of certain input, and (c) providing more aid to the user as to what to do next. In addition, areas of research were suggested concerning what kind of assessment questions are the best to pursue with respect to the properties of being reasonable to answer, and having parameter implications not overly sensitive (i.e., robust) to the precision of the answer.

REFERENCES

1. Ellis, Howard M. and Keeney, R. L., "A Rational Approach for Government Decisions Concerning Air Pollution," in A. D. Drake, R. L. Keeney, and P. M. Morse (eds.), Analysis of Public Systems, M.I.T. Press, Cambridge, Mass., 1972.

2. Fike, C. T., PL/1 for Scientific Programmers, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1970.

3. Fishburn, P. C., "Independence in Utility Theory with Whole Product Sets," Operations Research, Vol. 13, 28-45 (1965).

4. Grochow, Jerrold M., "A Utility Theoretic Approach to Evaluation of a Time-Sharing System,in Walter Freiberger (ed.), Statistical Computer Performance Evaluation, Academic Press, Inc., New York, 1972.

5. Hamming, R. W., Introduction to Applied Numerical Analysis, McGraw Hill, New York, 33-52 (1971).

6. Hauser, John R. and Urban, Glen L., "A Normative Methodology for Modeling Consumer Response to Innovation," Tech. Rep. #109, Operations Research Center, M.I.T., Cambridge, Mass., May, 1975.

7. Keeney, R. L., "Utility Functions for Multiattributed Consequences," Management Science, Vol. 18, 276-87 (1972).

8. Keeney, R. L., "A Decision Analysis with Multiple Objectives: The Mexico City Airport," Bell Journal of Economics and Management Science, Vol. 4, 101-117, (1973).

9. Keeney, R. L., "Multiplicative Utility Functions," Operations Research, Vol. 22, 22-34 (1974).

10. Keeney, R. L., and Nair, Keshavan, "Decision Analysis for Siting of Nuclear Power Plants--The Relevance of Multiattribute Utility Theory," Tech. Rep. #96, Operations Research Center, M.I.T., Cambridge, Mass., June, 1974.

11. Meyer, R. F., "On the Relationship Among the Utility of Assets, the Utility of Consumption, and Investment Strategy in an Uncertain, but Time Invariant World," Proceedings of the Fourth IFORS Conference, Venice, Italy, 1969.

12. Pollak, R. A., "Additive von Neumann-Morgenstern Utility Functions," Econometrica, Vol. 35, 485-595 (1967).

13. Pratt, J. W., "Risk Aversion in the Small and in the Large," Econometrics, Vol. 32, 122-136 (1964).

14. Raiffa, H., Decision Analysis, Addison-Wesley, Reading, Mass., 1968.

15. Schlaifer, R. O., Analysis of Decisions Under Uncertainty, McGraw-Hill, New York, 1969.

16. Schlaifer, R. O., Computer Programs for Elementary Decision Analysis, Division of Research, Harvard Business School, Boston, Mass., 1971.

17. Turnquist, Mark A., "A Bayesian Approach to Simulation--Specific Experimental Design with Application to Modeling Transportation Systems," Ph.D. Dissertation, Department of Civil Engineering, M.I.T., May, 1975.

18. von Neumann, J. and Morgenstern, O., Theory of Games and Economic Behavior, 2nd Edition, Princeton University Press, Princeton, New Jersey, 1947.

APPENDIX A

LIST OF MUFCAP COMMANDS WITH BRIEF DESCRIPTIONS

A.1 Notation and Command Descriptions

CE - Certainty equivalent

MUF - Multiattribute Utility Function

UNIF - Uniattribute (scalar attribute) utility function

$[y_1, y_2, \ldots, y_R]$ - Brackets indicate the options which may be chosen. No option needs to be selected.

$(y_1, y_2, \ldots, y_R)$ - Parentheses indicate that a choice must be made among the options given.

INPUT name - Inputs the structure of the multiattribute utility function to be referred to by 'name.' The dialogue requests names for the attributes and their ranges. Ranges for attributes over which preferences are monotonic should be input with the least desirable end of the range first. A vector attribute, (and hence a nested MUF) is signalled by specifying a range whose lower and upper limits are the same. After INPUT, the default for all MUF's is the additive form with $k_i = k_j$ for all i, j. The default for all UNIF's is the linear utility function. The user is set to calculate immediately after INPUT.

SAVE filename - Saves the current preference and alternative specifications in file named 'filename.'

READ filename - Restores the information which was saved in 'filename.'

DEBUG - Lists all the attributes in the utility function structure including their names, scaling factors, ranges, and UNIF types (0, 1, and 2 indicate respectively linear, constant risk aversion, and piecewise linear). A vector attribute has its name and scaling factor listed and is followed by its component attributes.

ADDALT altname [factor] - Initiates dialogue to specify an alternative to be referred to by 'altname.' Either a probabilistic or certainty alternative may be specified. If the former is the case, a piecewise linear cumulative probability distribution is requested for each scalar attribute. (Abscissa values for the cumulative are input in ascending order.) The option 'factor' is a number which sets all of the scalar attributes at the factor level of their ranges, e.g., if factor = .1, all the scalar attributes are set at one-tenth of the way from the 1st range value to the 2nd range value.

DROPALT altname - Removes the alternative 'altname' from the status.

EVAL uname [A, B,...] - Evaluates the alternatives A,B,..., using the utility function associated with 'uname.' If no alternatives are specified, all alternatives in the status are evaluated and the results listed.

UNISET uname (LIN,CR,PL) - Sets the scalar attribute utility function associated with 'uname' to linear, constant risk averse, or piecewise linear form. For the piecewise linear form, the abscissa values are input in ascending order.

KSET mname [factor,ADD,OVERIDE] - Sets the scaling factors for the MUF associated with 'mname.' The number 'factor' causes the current scaling factors to be multiplied by that number. The program automatically calculates the k associated with the new scaling factors. If ADD is specified, the current factors are normalized to add to 1. The user may input k directly in response to the final prompt by the computer if OVERIDE has been specified.

GRAD uname [A,B,...] - Calculates the gradient components of the utility function associated with 'uname' for all or some of the alternatives A,B,... .

INDIF1 uname1 uname2 - In the uname1-uname2 attribute plane, given relative k_i's, (i.e., scaling factors with the appropriate ratio relationship to each other but not necessarily the appropriate absolute value) the k is specified by a single pair of indifference consequences.

INDIF1 requests a pair of indifference consequences and uses the current k_i's as the given relative k_i's. On output, the k is given along with the factor by which the current k_i's must be multiplied to yield the k (see KSET command with 'factor' option).

INDIF2 uname1 uname2 - In the uname1-uname2 attribute plane, with scaling factors denoted by k_1 and k_2, inputting two pairs of two indifference consequences each specifies the ratio k_1/k_2 and $k = \text{constant}/k_1$. After INDIF2, the KSET command may be used to fix k_1, and then k_2 and k in terms of k_1. The command IMAP can then be used to generate indifference curves in the uname1-uname2 plane. (For these indifference curves, the values of k_i, $i \neq 1,2$, are irrelevant).

UNICAL uname [n] - Prints a list of utilities using the UNIF associated with 'uname.' Once the number n is specified, the user supplies n attribute amounts and the program returns the n associated utilities.

INVERSE uname [n] - Prints a list of attribute amounts associated with utilities using the UNIF 'uname.' Once the number n is specified, the user supplies n utility amounts of 'uname' and the program returns the n associated attribute levels. If n is not specified, the program has a default printout.

CHANGEALT uname altname - Routine to change the 'uname' attribute component of the alternative 'altname' without changing the other components.

CHANGE uname (NAME,K, RANGE) param - Routine to change the name or scaling factor or range of the attribute 'uname' to param. When the range is changed, param is not required. The program requests respecification of the UNIF type when the range is changed. When the name is changed, param must not be left blank.

ALTLIST - Lists the current alternatives. The probabilistic alternatives are listed with their CE equivalent components.

DISPLAY uname - Displays the characteristics of the utility function associated with 'uname.' The scaling factors for the attribute arguments and their sum is listed for a MUF while the range and type is listed for a UNIF.

FRACTILE uname altname - Displays the cumulative distribution for 'uname' in the alternative 'altname.'

LOTTERY uname n - Calculates the CE for a lottery involving the scalar attribute 'uname.' The number n specifies the number of possible lottery consequences. These are solicited with their corresponding porbabilities and the CE is calculated.

IMAP unamel uname2 - Initiates a dialogue to generate an indifference 'curve' in the unamel-uname2 plane. A point through which the curve will pass is solicited. Then values of unamel are input and the uname2 values required to maintain indifference are output.

STOP - Thanks the user for using MUFCAP and exits from the program.

ADDU unamel uname2 - Initiates a dialogue which adds an attribute 'unamel' to the argument list of the MUF associated with 'uname2.'

DELU uname - Deletes the attribute 'uname' from the structure.

SWITCH uname uname2 - Adds current attribute 'uname' to the argument list of the MUF associated with 'uname2' and deletes 'uname' as an argument of the MUF to which it originally belonged.

INTERBK uname - If any attribute arguments of the MUF associated with 'uname' is a vector, its utility function is a nested MUF with its own internal constant k. INTERBK calculates the theoretical k for the nested MUF which would make the nesting of the inner attributes unnecessary and prints it along with the current internal k.

A.2 Further Notes on INDIF1, INDIF2 and IMAP

The INDIF1 command may be used with input to a Type II Question (see Section 3.2). It will then give the relative k_i's as output. An example of this is shown in Illustration 16 of Section 5.2.4.

For INDIF1, INDIF2 and IMAP, if either uname1 or uname2 is a vector attribute, consequences must be input as utility pairs rather than attribute value pairs. The utility for an attribute value is the result obtained when that attribute amount (vector or scalar) is evaluated using the utility function associated with the attribute name. An example of this is shown in Illustraions 19 through 21 of Section 5.2.4.

Uname1 and uname2 must be explicit arguments of the same MUF when using INDIF1, INDIF2 or IMAP. That is to say, (uname1, uname2) must be preferentially independent of the other attributes.

Finally, on output, INDIF1 prints a number in parenthesis as a second factor by which to multiply the current k_i's. If multiplied by this factor, the new k_i's will not be consistent with the indifference pair input. However, these new k_i's will yield a k identical to that of the new k_i's derived by using the non-parenthesized factor. In practice, although not consistent with the indifference pair input, the "alternative" k_i's come close to being consistent. Sometimes, the non-parethesized factor will yield k_i's which are not allowed in the multiplicative form; e.g., $k_i > 1$ for some i. When this happens, the parentesized factor can be tried instead. Using IMAP, with these alternate k_i's, we can see if the indifference pair consistent with these alternate k_i's is close enough to the original pair used in INDIF1 to justify use of the parenthesized factor.

APPENDIX B

MUFCAP PROGRAM LISTINGS

```
MUFCAP: PROC OPTIONS (MAIN);
DCL DUMMY AREA (12800) BASED (DPTR);
DCL (EMPTY,NULL,ONSOURCE) BUILTIN;
DCL  (BIGK,UNIPXP) ENTRY EXTERNAL;
DCL WORD(10) CHAR(12),PFLAG BIT(1),UF BIT(1);
DCL COMMAND(26) CHAR(12) INIT('INPUT','SAVE','READ',
 'DEBUG','ADDALT','DROPALT','EVAL','WTSET','KSET','GRAD',
 'INDIF1','INDIF2','UNICA ','INVERSE',
 'CHANGEALT','CHANGE','AL  IST','DISPLAY','FRACTILE',
     'LOTTERY','IMAP','STOP','DELU','ADDU','SWITCH','INTERRK');
DCL MUFILE FILE RECORD SEQUENTIAL;
DCL PROC(26) LABEL;
DCL CLIST (30) OFFSET(DUMMY) STATIC;
DCL UTNAME(30) STATIC  CHAR(12), NUTN STATIC;
DCL NGRAD STATIC,GRAD(30) STATIC,AF BIT(1);
DCL NSUB (30) STATIC , XIN(15),GP BIT(1);
DCL (NAT, NC,ICALT,IPALT) STATIC, JARY(2) FIXED;
DCL CARD CHAR(80),ANAME CHAR(12),FNAME CHAR(12);
DCL (J1,J2,K1,K2) FIXED,YIN(15);

/* DEFINE AREA FOR BASED ALLOCATIONS */
DCL 1 LIST BASED(LISTPTR),
      2 FIRST OFFSET (DUMMY),
      2 NAMPALT  (5) CHAR(12),
      2 NAMCALT  (10) CHAR(12),
      2 BODY AREA  (12800);

/* MULTIATTRIBUTE UTILITY FUNCTION STRUCTURE */
DCL MUFP PTR STATIC;
DCL 1 MUF BASED (MUFP),
      2 CAPK,
      2 MNAME CHAR (12),
      2 NUMAT,
      2 SUBAT(12),
           3 CHAINP OFFSET (DUMMY),
           3 SMALLK,
           3 UNIPTR OFFSET (DUMMY),
           3 UNAME CHAR(12);

/* UNIATTRIBUTE UTILITY FUNCTION STRUCTURE */
DCL UNIPP PTR  STATIC;
DCL 1 UNIF BASED(UNIPP),
      2 ULO, 2 UHI, 2 UTYPE FIXED,
      2 CALT(10), /* CERT. ALTERNATIVES */
           3 CALX, 3 PUC,
      2 UXP(15), 2 UYP(15), 2 NUP,
      2 PALT(5), /* PROB. ALTERNATIVES */
           3 NP, 3 XP(9), 3 CP(9), 3 EUP;

/* INITIALIZE */
ALLOCATE LIST; NUTN=0; DPTR=ADDR(BODY); NCOM=26; GP='0'B;
DO I=1 TO 5; NAMPALT(I)=' '; END;
DO I=1 TO 10; NAMCALT(I)=' '; END;
PUT SKIP LIST('COMMAND WORD AND FILE NAMES MUST BE IN CAPS');
ON CONVERSION BEGIN;
      DCL P FLOAT; IF ONSOURCE='QUIT' THEN DO;
      PUT SKIP LIST('POSSIBLE STATUS CHANGE. UNDO PARTIAL OP.');
```

0.820225.11940.MUPCAP.PLI

```
        GO TO GETCOM; END;
         PUT SKIP EDIT(ONSOURCE, ' IS NOT A VALID NUMBER.',
        'INPUT THE CORRECT NUMBER :')
        (COL(2),A(LENGTH(ONSOURCE)),A,A);
        GET LIST(B); ONSOURCE=B; END;
   ON UNDEFINEDFILE(MUFILE) BEGIN;
   PUT SKIP    ST('FILES MUST BE ALLOCATED AND FILE NAMES IN CAPS');
   GO TO GETCOM; END;

   /* COMMAND PROCESSOR SECTION */
   GETCOM:
   PUT SKIP LIST('COMMAND? :');
        CALL GETLINE;
        DO NC=1 TO NCOM;
        IF WORD(1)=COMMAND(NC) THEN GO TO PROC(NC); END;
   PUT SKIP EDIT (WORD(1),' IS NOT A VALID COMMAND.',
        ' (COMMAND WORD MUST BE IN CAPS)')
        (COL(2),A,A,A); GO TO GETCOM;

   PROC(1): /* 'INPUT' */
        IF WORD(2)=' ' THEN DO;
        PUT SKIP LIST('NAME FOR MUP PLEASE? :');
        CALL GETLINE; WORD(2)=WORD(1); GO TO PROC(1); END;
        PUT SKIP LIST('HOW MANY ATTRIBUTES ARE IN THIS MUP? :');
        GET LIST (NAT); IF NAT <=0 THEN GO TO GETCOM;
        DO I=1 TO 5; NAMPALT(I)=' '; END;
        DO I=1 TO 10; NAMCALT(I)=' '; END;
        BODY=EMPTY; ALLOCATE MUP IN (BODY); MNAME=WORD(2);
        FIRST=MUPP; CAPK=0; NUMAT=NAT; CALL GETMULT; CALL SETOFF;
        GO TO GETCOM;

   PROC(2): /* 'SAVE' */
        IF WORD(2)=' ' THEN DO;
        PUT SKIP LIST('FILE NAME FOR SAVE PLEASE? :');
        CALL GETLINE; WORD(2)=WORD(1); GO TO PROC(2); END;
        OPEN FILE(MUFILE) TITLE(WORD(2)) OUTPUT;
        WRITE FILE(MUFILE) FROM (LIST); CLOSE FILE(MUFILE);
        GO TO GETCOM;

   PROC(3): /* 'READ' */
        IF WORD(2)=' ' THEN DO;
        PUT SKIP LIST('FILE NAME FOR READ PLEASE? :');
        CALL GETLINE; WORD(2)=WORD(1); GO TO PROC(3); END;
        OPEN FILE(MUFILE) TITLE(WORD(2)) INPUT;
        READ FILE(MUFILE) INTO (LIST); CLOSE FILE(MUFILE);
        CALL SETOFF; GO TO GETCOM;

   PROC(4): /* 'DEBUG' */
        PUT SKIP EDIT('STRUCTURE FOR ',PNAME) (COL(5),A,A);
        DO I=1 TO NUTN;
   MUPP=OLIST(I); PUT SKIP EDIT(UTNAME(I),SUBAT(NSUB(I)).SMALLK)
     (COL(2),A,F(8,3)); IF SUBAT.CHAIIP(NSUB(I))=NULL THEN DO;
   UNIPP=SUBAT.UNIPTR(NSUB(I)); PUT SKIP LIST(ULO,UHI,UTYPE); END;END;
   GO TO GETCOM;

   PROC(5): /* 'ADDAL ' */
        IF WORD(2)=' ' THEN DO;
```

U.M20225.11940.MUFCAP.PLI

```
        PUT SKIP LIST('NAME FOR ALTERNATIVE PLEASE? :');
        CALL GETLINE; WORD(2)=WORD(1); GO TO PROC(5); END;
        ANAME=WORD(2); IF WORD(3)¬=' ' THEN DO; AF='1'B;
        PFLAG='0'B; APAC=WORD(3); GO TO CERT; END; ELSE AF='0'B;
        ANAME=WORD(2); PUT SKIP LIST('IS ALT. PROB? (YES OR NO):');
        REPLY5: CALL GETLINE; IF WORD(1)='YES' THEN PFLAG='1'B;
        ELSE IF WORD(1)='NO' THEN PFLAG='0'B;
        ELSE DO; PUT SKIP LIST('REPLY MUST BE YES OR NO IN CAPS ');
        GO TO REPLY5; END;
        IF PFLAG THEN DO I=1 TO 5;
        IF NAMPALT(I)=' ' THEN DO;
        NAMPALT(I)=ANAME; IPALT=I; GO TO PROC5B; END;
        IF I=5 THEN DO;
    PUT SKIP EDIT('ONLY 5 PROB. ALTERNATIVES ALLOWED. ',
    'ONE SHOULD BE DELETED. (STATUS CAN BE SAVED, TOO)')
    (COL(2),A,A);
        GO TO GETCOM; END; END;
        ELSE CERT: DO I=1 TO 10;
        IF NAMCALT(I)=' ' THEN DO;
        NAMCALT(I)=ANAME; ICALT=I; GO TO PROC5B; END;
        IF I=10 THEN DO;
        PUT SKIP LIST('ONLY 10 CERT. ALT. ALLOWED');
        GO TO GETCOM; END; END;
    PROC5B:    PUT SKIP EDIT ('ALTERNATIVE ',ANAME,' SPECIF.') (A,A,A);
        DO I=1 TO NUTN;
        NUPP=OLIST(I); J=NSUB(I);
        IF SUBAT(J).CHAINP=NULL THEN DO;
        UNIPP=SUBAT(J).UNIPTR;
        CALL ALTCOMP; END; END;GO TO GETCOM;
    ALTCOMP: PROC;
        /* NEEDS I,PFLAG,ICALT OR IPALT,AF AND UNIPP SET */
        /* SETS THE COMPONENT FOR AN ALTERNATIVE */
        IF PFLAG THEN DO;
        PUT SKIP EDIT('HOW MANY FRACTILE PTS. (INCL 0 AND 100%)  FOR ',
        UTNAME(I),' (2<=N<=9) : ') (A,A,A);
        GET LIST(N); PALT(IPALT).NF=N;
    PUT SKIP LIST('INPUT THE CUM FUNC F(X). X''S FIRST THEN F(X)''S');
        GET LIST((XIN(J) DO J=1 TO N));
        IF UHI<ULO THEN DO J=1 TO N; PALT(IPALT).XP(J)=
        (XIN(N-J+1)-ULO)/(UHI-ULO); END;
        ELSE DO J=1 TO N; PALT(IPALT).XP(J)=(XIN(J)-ULO)/
        (UHI-ULO); END;
        GET LIST((XIN(J) DO J=1 TO N));
        IF UHI<ULO THEN DO J=1 TO N;
        PALT(IPALT).CP(J)=1-XIN(N-J+1); END;
        ELSE DO J=1 TO N; PALT(IPALT).CP(J)=XIN(J); END;
    /* ABOVE INSURES THAT INTERNAL REP OF CUM FUNC IS OKAY */
        CALL UNIEU(IPALT,ANS); PALT(IPALT).EUP=ANS;
        END;
        ELSE IF ¬AF THEN DO; PUT SKIP EDIT(UTNAME(I),'=:') (A,A);
        GET LIST(X); X=(X-ULO)/(UHI-ULO); CALL UNICAL(X,ANS),
        CALT(ICALT).CALX=X; CALT(ICALT).EUC=ANS;
        END; ELSE DO; CALL UNICAL(APAC,EUC(ICALT));
        CALX(ICALT)=APAC; END; END ALTCOMP;

    PROC(6): /* 'DROPALT */
        IF WORD(2)=' ' THEN DO;
```

U.M20225.11980.MUPCAF.PLI

```
        PUT SKIP LIST('NAME FOR ALTERNATIVE PLEASE? :');
        CALL GETLINE; WORD(2)=WORD(1); GO TO PROC(6); END;
        ANAME=WORD(2); DO I=1 TO 10; IF NAMCALT(I)=ANAME THEN
        NAMCALT(I)=' '; END; DO I=1 TO 5; IF NAMPALT(I)=
        ANAME THEN NAMPALT(I)= ' '; END; GO TO GETCOM;

   PROC(7): /*EVAL */
   CALL PROC7A; GO TO GETCOM;
        PROC7A: PROC; IF WORD(2)=PNAME THEN DO; MUPP=FIRST;
        UP='0'B;
        GO TO PROC7C; END;
        DO I=1 TO NUTN; IF UTNAME(I)=WORD(2) THEN GO TO PROC7B; END;
        PUT SKIP LIST('ATTRIB NOT FOUND'); RETURN;
   PROC7B: MUPP=OLIST(I); IF SUBAT(NSUB(I)).CHAINP=NULL THEN DO;
        UNIFP=SUBAT(NSUB(I)).UNIFTP; UP='1'B; END;
        ELSE DO; UP='0'B; MUPP=SUBAT(NSUB(I)).CHAINP; END;
   PROC7C: IF WORD(3)=' ' THEN GO TO PROC7E;
        J=2;
   PROC7D: J=J+1; IF WORD(J)=' ' THEN RETURN;
        PFLAG='0'B; DO I=1 TO 10; ICALT=I; IF NAMCALT(I)=WORD(J)
        THEN DO; IF UP THEN ANS=CALT(I).EUC; ELSE CALL MULTEV(ANS);
        PUT EDIT(NAMCALT(I),ANS) (COL(2),A,X(1),F(8, ));
   IF (GPE(¬UP)) THEN CALL GETGRAD;
        GO TO PROC7D; END; END;
        PFLAG='1'B; DO I=1 TO 5; IPALT=I; IF NAMPALT(I)=WORD(J) THEN
        DO; IF UP THEN ANS=PALT(I).EUP; ELSE CALL MULTEV(ANS);
        PUT EDIT(NAMPALT(I),ANS) (COL(2),A,X(1),F(8,3));
   IF (GPE(¬UP)) THEN CALL GETGRAD; GO TO PROC7D;
        END; END;
   PROC7E: PFLAG='0'B; DO I=1 TO 10; ICALT=I; IF NAMCALT(I)¬=' ' THEN
        DO; IF UP THEN ANS=CALT(I).EUC; ELSE CALL MULTEV(ANS);
        PUT EDIT(NAMCALT(I),ANS) (COL(2),A,X(1),F(8,3));
   IF (GPE(¬UP)) THEN CALL GETGRAD;
        END;END;
        PFLAG='1'B; DO I=1 TO 5; IPALT=I; IF NAMPALT(I)¬=' ' THEN
        DO; IF UP THEN ANS=PALT(I).EUP; ELSE CALL MULTEV(ANS);
        PUT EDIT(NAMPALT(I),ANS) (COL(2),A,X(1),F(8,3));
   IF (GPE(¬UP)) THEN CALL GETGRAD;
        END; END; RETURN; END PROC7A;

   PROC(8): /* 'UNISET' */
        CALL UNIGET;
   PROC8C: IF WORD(3)=' ' THEN DO;
        PUT SKIP LIST('TYPE? :'); CALL GETLINE; WORD(3)=WORD(1);
        GO TO PROC8C; END;

        IF WORD(3)='CR' THEN DO;
        PUT SKIP EDIT('INPUT ANY 50-50 LOTTERY IN THE FORM OF
        ' C.E.,O1 & O2. PLEASE') (A,A);
        GET LIST(CE,X1,X2); IF UHI>ULO THEN DO;
        XLO=MIN(X1,X2); XHI=MAX(X1,X2); END;
        ELSE DO; XLO=MAX(X1,X2); XHI=MIN(X1,X2); END;
        P=(UHI-ULO); CE=(CE-ULO)/P; XLO=(XLO-ULO)/P;
        XHI=(XHI-ULO)/P; CALL UNIEXP(CE,XLO,XHI,UXP(1),UYP(1));
        UTYPE=1; END;
        ELSE IF WORD(3)='LIN' THEN UTYPE=0;
        ELSE IF WORD(3)='PL' THEN DO;
```

U.M20225.11940.MUFCAP.PLI

```
        PUT SKIP LIST('HOW MANY PTS. IN UTIL FUNC? :');
        GET LIST(N); PUT SKIP EDIT('INPUT THE FUNC., X''S FIRST ',
        'THEN U(X)''S') (A,A);
        GET LIST((XIN(I) DO I=1 TO N));
        GET LIST((UYP(I) DO I=1 TO N));
        NUP=N;
        IF UHI>ULO THEN DO I=1 TO N; UXP(I)=(XIN(I)-ULO)/(UHI-ULO);
        END; ELSE DO; DO I=1 TO N; UXP(I)=(XIN(N-I+1)-ULO)/
        (UHI-ULO); XIN(N-I+1)=UYP(I); END; DO I=1 TO N;
        UYP(I)=XIN(I); END; END;
        UTYPE=2; END; ELSE DO; PUT SKIP LIST('UNIF TYPE NOT VALID');
        GO TO GETCOM; END;
    /* UPDATE EXPECTED UTILITY FOR ALTERNATIVES */
        DO I=1 TO 5; IF NAMPALT(I)¬=' ' THEN CALL UNIEU(I,PALT(I).EUP);
        END; DO I=1 TO 10; IF NAMCALT(I)¬=' ' THEN CALL
        UNICAL(CALT(I).CALX,CALT(I).EUC); END; GO TO GETCOM;

    PROC(9): /* RESET */
        IF WORD(2)=FNAME THEN DO; MUFP=FIRST; GO TO PROC9C; END;
        DO I=1 TO NUTN; IF UTNAME(I)=WORD(2) THEN GO TO PROC9B; END;
        PUT SKIP LIST('ATTRIB NOT FOUND'); GO TO GETCOM;
    PROC9B: MUFP=OLIST(I); IF SUBAT(NSUB(I)).CHAINP=NULL THEN DO;
        PUT SKIP LIST('ATTRIB IS NOT A MUF'); GO TO GETCOM; END;
        MUFP=SUBAT(NSUB(I)).CHAINP;
    PROC9C:
         IF WORD(3)='ADD' THEN DO; SUMK=0; DO I=1 TO NUMAT;
        SUMK=SUMK+SUBAT(I).SMALLK; END; FACTOR=1./SUMK; END;
        ELSE IF WORD(3)='OVERIDE' THEN DO; WORD(4)=WORD(3);
        WORD(3)=' '; END;
        ELSE IF WORD(3)¬=' ' THEN FACTOR=WORD(3);
        DO I=1 TO NUMAT;
        IF WORD(3)=' ' THEN DO; PUT EDIT(SUBAT(I).UNAME,'= :') (COL(2),
        A,A);
        GET LIST(SUBAT(I).SMALLK); END; ELSE SUBAT(I).SMALLK=FACTOR*
        SUBAT(I).SMALLK; END; IF WORD(4)='OVERIDE' THEN GET LIST(CAPK);
        ELSE CAPK=BIGK(SMALLK,NUMAT); PUT SKIP EDIT('BIGK=',CAPK)
         (COL(2),A,X(1),F(8,3));
         GO TO GETCOM;

    PROC(10): /* GRAD */
        GF='1'B; CALL PROC7A; GF='0'B; GO TO GETCOM;

    /* PROCEDURE TO RESET OFFSET LIST */

    SETOFF: PROC; NUTN=0; MUFP=FIRST; FNAME=FNAME; CALL RESETOF;
        END SETOFF;
    RESETOF: PROC RECURSIVE;
        DCL TEMP PTR,I FIXED;
    DO I=1 TO NUMAT;
        NUTN=NUTN+1; UTNAME(NUTN)=SUBAT(I).UNAME;
        OLIST(NUTN)=MUFP; NSUB(NUTN)=I;
        IF SUBAT(I).CHAINP¬=NULL THEN DO;
        TEMP=MUFP; MUFP=SUBAT(I).CHAINP;
        CALL RESETOF;
    /* POPPING UP */
        MUFP=TEMP; END;
    END;
```

U.M20225.11940.MUPCAE.PLI

```
RETURN; END RESETOP;
/* PROCEDURE TO MAKE NEST */

GETMULT: PROC RECURSIVE;
     DCL TEMP PTR,I FIXED;
L1: DO I=1 TO NUMAT;
PUT SKIP EDIT ('INPUT NAME AND RANGE FOR ATTR ',I,'OF UTIL FUNC ',
MNAME) (COL(2),A,F(2),X(1),A,A);
CALL GETLINE; SUBAT(I).UNAME=WORD(1); R1=WORD(2);
     R2=WORD(3);

IF R1=R2 THEN DO;
PUT SKIP LIST('HOW MANY ATTR. ARE IN THIS MUP?:');
GET LIST (NAT); TEMP=MUPP;

/* CREATE A NEW MUP */
ALLOCATE MUP IN (BODY); NUMAT=NAT;
   TEMP->SUBAT(I).CHAINP=MUPP;
MNAME=TEMP->SUBAT(I).UNAME; CAPK=0;
/* RECURSIVE CALL */
CALL GETMULT;

/* POPPING UP AGAIN AFTER RECURSION */
MUPP=TEMP; END;

ELSE DO; SUBAT(I).CHAINP=NULL; ALLOCATE UNIP IN (BODY);
     SUBAT(I).UNIPTR=UNIPP; UTYPE=0; ULO=R1; UHI=R2;
     END;
SUBAT(I).SMALLK=1./NUMAT;
END L1;
     RETURN;
     END GETMULT;
/* FREE FORMAT READ CARD ROUTINE */
GETLINE: PROC;
DCL (I,J,K) FIXED;
DO I=1 TO 10; WORD(I)=' '; END;
GET EDIT (CARD) (A(80));
I=1; K=1;
DO WHILE(I<=80);
IF SUBSTR(CARD,I,1)=' ' THEN GO TO CONT;
J=1; IF I=80 THEN GO TO GOT;
MORE: IF SUBSTR(CARD,I+J,1)=' ' THEN GO TO GOT;
     J=J+1; IF (I+J=81) THEN GO TO GOT; GO TO MORE;
GOT: WORD(K)=SUBSTR(CARD,I,J); K=K+1; I=I+J-1;
CONT: I=I+1; END;
RETURN; END GETLINE;

UNICAL: PROC(X,ANS);
/* UNIPP IS ASSUMED POINTING AT THE PROPER UTILITY FUNCTION */
/* PROCEDURE TO CALCULATE THE UTILITY OF A VALUE */
DCL J FIXED;
IF UTYPE=0 THEN DO; ANS=X; RETURN; END; /* LINEAR U(X)=X */
/* CONSTANT RISK */
IF UTYPE=1 THEN DO; ANS=UXP(1)*(1-EXP(-UYP(1)*X)); RETURN; END;
/* PIECEWISE LINEAR */
IF UTYPE=2 THEN DO;
IF X>=1 THEN ANS=1.+(UYP(NUP)-UYP(NUP-1))/(UXP(NUP)-UXP(NUP-1))*
```

U.M20225.11940.MUPCAP.PLI

```
    (X-1.);
    ELSE IF X<=0 THEN ANS=(UYP(2)-UYP(1))/(UXP(2)-UXP(1))*X;
    ELSE IF (X<1 & X>0) THEN DO;
         DO J=1 TO NUP; IF UXP(J)>X THEN GO TO CAL; END;
         CAL: J=J-1;
    ANS=UYP(J)+(UYP(J+1)-UYP(J))/(UXP(J+1)-UXP(J))*(X-UXP(J));
    RETURN; END;
    END UNICAL;

    UNIEU:PROC (NALT,ANS);
    /* PROC TO CALCULATE EXP. UTIL FOR UNIAT. UNIFP ASSUMED SET */
    DCL (J,NX,JU,JP,NL) FIXED;
    DCL (SU(*),SP(*),XX(*),B(*)) CONTROLLED;
    DCL 1 ALT, 2 NPA, 2 XPA(9), 2 CPA(9), 2 FUA;
    ALT=PALT(NALT); ANS=0;
    IF UTYPE=0 THEN DO;
    DO J=2 TO NPA; ANS=ANS+(CPA(J)-CPA(J-1))/(XPA(J)-XPA(J-1))*
    (XPA(J)*XPA(J)-XPA(J-1)*XPA(J-1))/2.;
    END; RETURN; END;

    ELSE IF UTYPE=1 THEN DO;
    DO J=2 TO NPA; ANS=ANS+(CPA(J)-CPA(J-1))/(XPA(J)-XPA(J-1))*
    UXP(1)*(XPA(J)-XPA(J-1)+(EXP(-UYP(1)*XPA(J))-EXP(-UYP(1)*
    XPA(J-1)))/UYP(1)); END; RETURN; END;

    ELSE IF UTYPE=2 THEN DO;
    JP=1; DO JU=1 TO NUP; IF (UXP(JU)>XPA(JP)) THEN GO TO ALOC; END;
    /* INTEGRATE ONLY WHERE SOME PROB. IS */
    JU=NUP; /* THIS LAST STMT. IN CASE WE FALL OUT OF LOOP */
    ALOC: NL=NPA+NUP-JU; ALLOCATE SU(NL),SP(NL),XX(NL),B(NL);
    /* NL IS MAX NUMBER OF INTERVALS REQUIRED */
    NX=0;
    DO J=1 TO NL; /* PUT INTERVALS IN ORDER */
    NX=NX+1; IF XPA(JP)>1 THEN GO TO INCJP; IF UXP(JU)<=XPA(JP) THEN DO;
         XX(NX)=UXP(JU); JU=JU+1; IF UXP(JU-1)=XPA(JP) THEN
         JP=JP+1; END;
    ELSE INCJP: DO;
         XX(NX)=XPA(JP); JP=JP+1; END;
    IF (JP>NPA) THEN GO TO LPEND;
    IF(XX(NX)<1&XX(NX)>0) THEN DO;
         SU(NX)=(UYP(JU)-UYP(JU-1))/(UXP(JU)-UXP(JU-1));
         B(NX)=UYP(JU)-SU(NX)*UXP(JU); END;
    ELSE IF XX(NX)<=0 THEN DO;
         SU(NX)=(UYP(2)-UYP(1))/(UXP(2)-UXP(1));
         B(NX)=UYP(2)-SU(NX)*UXP(2); END;
    ELSE IF XX(NX)>=1 THEN DO;
         SU(NX)=(UYP(NUP)-UYP(NUP-1))/(UXP(NUP)-UXP(NUP-1));
         B(NX)=UYP(NUP)-SU(NX)*UXP(NUP); END;
    SP(NX)=(CPA(JP)-CPA(JP-1))/(XPA(JP)-XPA(JP-1));
     END;
    LPEND: DO J=2 TO NX;
    ANS=ANS+SU(J-1)*SP(J-1)*(XX(J)*XX(J)-XX(J-1)*XX(J-1))/2.;
         ANS=ANS+SP(J-1)*B(J-1)*(XX(J)-XX(J-1));
    /* INT(K(MX+B))=K(MX**2/2 +BX)   */
         END;
     FREE SU,SP,XX,B; RETURN; END;
    END UNIEU;
```

U.M20225.11940.MUPCAP.PLI

```
  MULTEV: PROC(ANS); NGRAD=0; CALL MULTCAL(ANS); RETURN ;END MULTEV;
  MULTCAL: PROC(ANS) RECURSIVE;
       DCL TEMP PTR, I FIXED, R FLOAT;
       DCL TEMPU(12),NGR(12);
       IF CAPK=0 THEN DO; /* ADDITIVE FORM */
       ANS=0; DO I=1 TO NUMAT; NGRAD=NGRAD+1; NGR(I)=NGRAD;
       IF SUBAT(I).CHAINP=NULL THEN IF PFLAG THEN
       R=SUBAT(I).UNIPTR->PALT(IPALT).EUP;
       ELSE R=SUBAT(I).UNIPTR->CALT(ICALT).EUC;
       ELSE DO; /* NEED TO EVAL A MUP */
       TEMP=MUPP; MUPP=SUBAT(I).CHAINP; CALL MULTCAL(R);
       /* POP UP */ MUPP=TEMP; END;
       ANS=ANS+SUBAT(I).SMALLK*R; END;
       DO I=1 TO NUMAT; GRAD(NGR(I))=SUBAT(I).SMALLK; END;
       RETURN; END;

       ELSE DO; /* MULT. FORM */
       ANS=1.; DO I=1 TO NUMAT; NGRAD=NGRAD+1; NGR(I)=NGRAD;
       IF SUBAT(I).CHAINP=NULL THEN IF PFLAG THEN
       R=SUBAT(I).UNIPTR->PALT(IPALT).EUP; ELSE
       R=SUBAT(I).UNIPTR->CALT(ICALT).EUC;
       ELSE DO;
       TEMP=MUPP; MUPP=SUBAT(I).CHAINP; CALL MULTCAL(R);
       MUPP=TEMP; END; ANS=ANS*(1+CAPK*SUBAT(I).SMALLK*R);
       TEMPU(I)=R; END;
       DO I=1 TO NUMAT; GRAD(NGR(I))=ANS/(1+CAPK*SUBAT(I).SMALLK*
       TEMPU(I))*SUBAT(I).SMALLK; END;
       ANS=(ANS-1)/CAPK;
       RETURN; END;
  END MULTCAL;

  GETGRAD: PROC;
       DCL TEMP PTR,FACTOR FLOAT,I FIXED,J FIXED;
       NGRAD=0; FACTOR=1; CALL SETGRAD(FACTOR);
       DO I=1 TO NUTN;
       J=I-1; TEMP=OLIST(I); IF MUPP=TEMP THEN GO TO GGRAD2; END;
  GGRAD2:
       IF ¬PFLAG THEN PUT SKIP LIST
       ('ATTRIB,UTIL. GRAD COMP. AND ATTR. GRAD COMP.');
       ELSE
       PUT SKIP LIST('ATTRIB & UTIL. GRADIENT COMPONENTS');
       DO I=1 TO NGRAD;
       IF ¬PFLAG & OLIST(J+I)->CHAINP(NSUB(I))=NULL
       THEN DO; UNIPP=OLIST(J+I)->UNIPTR(NSUB(I));
       CALL UNICAL(CALX(ICALT),R1); CALL UNICAL(CALX(ICALT)+.01,R2);
       DERIV=(R2-R1)/(CALX(ICALT)+.01-CALX(ICALT));
       DERIV=DERIV/(UHI-ULO); PUT EDIT(UTNAME(J+I),GRAD(I),
       GRAD(I)*DERIV) (COL(2),A,X(1),F(8,3),X(2),F(10,3));
        END;
       ELSE PUT EDIT(UTNAME(J+I),GRAD(I))
       (COL(2),A,X(1),F(8,3)); END; PUT SKIP(2); RETURN;
  END GETGRAD;

  SETGRAD: PROC(FACTOR) RECURSIVE;
       DCL TEMP PTR, I FIXED, FAC2 FLOAT;
       DO I=1 TO NUMAT; NGRAD=NGRAD+1;
```

U.M20225.11940.MUPCAP.PLI

```
        IF SUBAT(I).CHAINP¬=NULL THEN DO;
        TEMP=MUPP; MUPP=SUBAT(I).CHAINP; FAC2=FACTOR*GRAD(NGRAD);
        CALL SETGRAD(FAC2); MUPP=TEMP; END;
        GRAD(NGRAD)=FACTOR*GRAD(NGRAD); END;
        RETURN;
        END SETGRAD;

   UNIGET: PROC;
        DO I=1 TO NUTN; IF UTNAME(I)=WORD(2) THEN GO TO UGETB;
        END; UGETC: PUT SKIP LIST('UNIF NOT FOUND'); GO TO GETCOM;
        UGETB: MUPP=OLIST(I); IF SUBAT(NSUB(I)).CHAINP¬=NULL
        THEN GO TO UGETC; UNIFP=SUBAT(NSUB(I)).UNIPTR; RETURN;
        END UNIGET;

   PROC(11): PROC(12): /* INDIF1 AND INDIF2 */
        /* BASED ON EQUATION U(X1,Y1)=U(X2,Y2) WHEN
        (X1,Y1) IS INDIFFERENT TO (X2,Y2) */
        CALL GET2; GO TO PROC11C;
   GET2: PROC;
        /* ROUTINE GETS 2 U*'S AND SETS J1,J2,K1,K2,UP,MUPP
        JARY( ), AND I */
        JARY(1),JARY(2)=0; DO I=1 TO NUTN; DO J=1 TO 2;
        IF WORD(1+J)=UTNAME(I) THEN DO; JARY(J)=I;
        GO TO GET2B; END; END;
   GET2B:  IF (JARY(1)>0 & JARY(2)>0) THEN GO TO GET2C; END;
        PUT SKIP LIST('ATTR. NOT BOTH FOUND'); GO TO GETCOM;
   GET2C:J1=JARY(1);J2=JARY(2); IF (OLIST(J1)¬=OLIST(J2)) THEN DO;
        PUT SKIP LIST('ATTR. NOT IN SAME MUP'); GO TO GETCOM; END;
        MUPP=OLIST(J1);
        K1=NSUB(J1);K2=NSUB(J2); IF (CHAINP(K1)¬=NULL|CHAINP(K2)¬=NULL)
        THEN DO; UP='0'B; PUT SKIP LIST('INPUT UTILITY VALUES');
        END; ELSE UP='1'B; RETURN;  END GET2;
        PROC11C:MUPP=OLIST(I); R=SUBAT(NSUB(J2)).SMALLK/SMALLK(K1);
        IF WORD(1)='INDIF2' THEN GO TO PROC11F;
        PUT SKIP LIST('INPUT AN INDIFFERENCE PAIR PLEASE');
        GET LIST ((XIN(I) DO I=1 TO 4)); IF ¬UP THEN GO TO DCAL1;
        DO J=1 TO 2; UNIFP=SUBAT(NSUB(JARY(J))).UNIPTR; DO I=J,J+2;
        XIN(I)=(XIN(I)-ULO)/(UHI-ULO); CALL UNICAL(XIN(I),ANS);
        XIN(I)=ANS; END; END;
        DCAL1:DES1=XIN(1)*XIN(2)-XIN(3)*XIN(4);
        IF DES1=0 THEN GO TO PROC11D;
        RATK=(R*(XIN(4)-XIN(2))+XIN(3)-XIN(1))/(R*DES1);
        IF RATK=0 THEN DO; BK=0; GO TO PROC11F; END;
        BK=1; DO I=1 TO NUMAT; BK=BK*(1+RATK*SUBAT(I).SMALLK/
        SUBAT(NSUB(J1)).SMALLK); END; BK=BK-1;
        FAC1=RATK/(SMALLK(K1)*BK);
        FAC2=-(SMALLK(K1)+SMALLK(K2))/(BK*SMALLK(K1)*SMALLK(K2))-FAC1;
        PUT SKIP EDIT('IMPLIED NEW K''S FACTOR(S) ',FAC1,'(',FAC2,')')
        (COL(2),A,F(8,3),X(1),A,F(8,3),A);
   PROC11F:
        PUT SKIP EDIT('IMPLIED NEW BIGK= ',BK)  (COL(2),A,F(8,3));
        GO TO GETCOM;
   PROC11D:  PUT SKIP LIST('INDIF PAIR YIELDS INFO ABOUT REL K''S');
        PUT SKIP EDIT('REL K CHECK. CURRENT RATIO ', WORD(3),' TO ',
        WORD(2),' = ',R)  (COL(2),A,A,A,A,A,F(8,3));
        R=(XIN(3)-XIN(1))/(XIN(2)-XIN(4));
        PUT SKIP EDIT('IMPLIED RATIO = ',R)  (COL(2),A,F(8,3));
```

D.M20225.11940.MUPCAP.PLI

```
        GO TO GETCOM;
    PROC11E:  PUT SKIP LIST('INPUT 2 INDIFFERENCE PAIRS PLEASE');
        GET LIST((XIN(I) DO I=1 TO 8));
        IF ¬UP THEN GO TO DCAL2;
        DO J=1 TO 2; UNIPP=SUBAT(NSUB(JARY(J))).UNIPTR;
        DO I=J,J+2,J+4,J+6;
        XIN(I)=(XIN(I)-ULO)/(UHI-ULO); CALL UNICAL(XIN(I),ANS);
        XIN(I)=ANS; END; END;
    /* CHECK BOTH DESCRIMINANTS FOR REDUNDANCY */
        DCAL2:DES1=XIN(1)*XIN(2)-XIN(3)*XIN(4);
        DES2=XIN(5)*XIN(6)-XIN(7)*XIN(8);
        IF (DES1=0|DES2=0) THEN DO;
        PUT SKIP EDIT('ONE INDIF PAIR OBVIOUSLY YIELDS REL K''S. USE',
        ' THE COMMAND INDIF1 WITH IT TO COMPARE WITH CURRENT RATIO')
        (COL(2),A,A); GO TO GETCOM; END;
        Q1=XIN(7)-XIN(5)-(XIN(3)-XIN(1))*DES2/DES1;
        Q2=XIN(6)-XIN(8)+(XIN(4)-XIN(2))*DES2/DES1;
        IF(Q1=0|Q2=0) THEN DO;
        PUT SKIP LIST('CANNOT DETERMINE REL K''S FROM THESE PTS.');
        GO TO GETCOM; END;
        FAC1=(Q1/Q2)*(XIN(4)-XIN(2))+XIN(3)-XIN(1);
        FAC1=FAC1/((Q1/Q2)*DES1);
        PUT SKIP EDIT('BIGK=',FAC1,'/K(',WORD(2),')')
        (COL(2),A,F(8,3),A,A,A);
        XIN(3)=Q1; XIN(2)=Q2; XIN(4),XIN(1)=0; GO TO PROC11D;

    PROC(13): /* UNICAL */
        CALL UNIGET; IF WORD(3)=' ' THEN GO TO PROC13C;
        N=WORD(3); GET LIST((XIN(I) DO I=1 TO N));
    PROC13B:  DO I=1 TO N; P=(XIN(I)-ULO)/(UHI-ULO);
        CALL UNICAL(P,ANS); PUT EDIT('U(',XIN(I),')= ',ANS)
        (COL(2),A,F(10,3),A,F(8,3));
        END; GO TO GETCOM;
    PROC13C:  XIN(1)=ULO; DO I=2 TO 10 BY 2; XIN(1+I/2)=ULO+1.E-1*I*
        (UHI-ULO); END; N=6;  GO TO PROC13B;

    PROC(14): /* 'INVERSE' */
        CALL UNIGET;
        IF WORD(3)=' ' THEN GO TO PROC14C;
        N=WORD(3);
        GET LIST ((XIN(I) DO I=1 TO N));
    PROC14B:  DO I=1 TO N; CALL UNINV(XIN(I),ANS); ANS=ULO+(UHI-ULO)
        *ANS; PUT EDIT(ANS,'=INV(',XIN(I),')') (COL(2),F(10,3),A,
        F(8,3),A); END; GO TO GETCOM;
    PROC14C:  XIN(1)=0; DO I=1 TO 9 BY 2;
        XIN(2+I/2)=1.E-1*I; END; XIN(7)=1; N=7; GO TO PROC14B;

    UNINV: PROC(Y,ANS);
    /* PROC TO GET INVERSE OR ANS=X| U(X)=Y */
    DCL J FIXED;
        IF UTYPE=0 THEN DO; ANS=Y; RETURN; END;
        IF UTYPE=1 THEN DO;
        ANS=LOG(UXP(1)/(UXP(1)-Y))/UYP(1); RETURN; END;

        IF UTYPE=2 THEN DO;
        IF (Y>1|Y<0) THEN MESS: DO;
        PUT SKIP EDIT(Y,' IS OUT OF RANGE') (COL(2),F(8,3),A);
```

U.M20225.11940.MUPCAP.PLI

```
      ANS=-1; RETURN; END;
      DO J=2 TO NUP; IF (UYP(J-1)-Y)*(UYP(J)-Y)<=0 THEN
      GO TO GOT; END; GO TO MESS;
 GOT: ANS=(Y-UYP(J-1))/(UYP(J)-UYP(J-1))*(UXP(J)-UXP(J-1))
      +UXP(J-1); RETURN; END; END UNINV;

 PROC(15): /* 'CHANGEALT' */
      PFLAG='0'B; AF='0'B; DO I=1 TO 10;
      IF NAMCALT(I)=WORD(3) THEN DO; ICALT=I; GO TO CALLALT;
      END; END; PFLAG='1'B; DO I=1 TO 5; IF NAMPALT(I)=WORD(3)
      THEN DO; IPALT=I; GO TO CALLALT; END; END;
      PUT SKIP LIST('ALTERN. NOT FOUND'); GO TO GETCOM;
 CALLALT:  CALL UNIGET; /* SETS UNIFP AND I */
      CALL ALTCOMP; GO TO GETCOM;

 PROC(16): /* CHANGE */
      IF WORD(2)=FNAME THEN GO TO PROC16C;
      DO I=1 TO NUTN; IF UTNAME(I)=WORD(2) THEN GO TO PROC16B; END;
      PUT SKIP LIST('ATTRIB NOT FOUND'); GO TO  GETCOM;
 PROC16B: MUPP=OLIST(I); IF CHAINP(NSUB(I))=NULL THEN DO;
      UNIFP=UNIPTR(NSUB(I)); UP='1'B; END; ELSE UP='0'B;
      IF WORD(3)='NAME' THEN DO;
      UTNAME(I),UNAME(NSUB(I))=WORD(4); IF ¬UP THEN
      CHAINP(NSUB(I))->MNAME=WORD(4); END;
      ELSE IF  WORD(3)='K' THEN DO;
      SMALLK(NSUB(I))=WORD(4); CAPK=BIGK(SMALLK,NUMAT); END;
      ELSE IF (WORD(3)='RANGE' & UP) THEN DO;
      PUT SKIP LIST('ALTERNATIVE COMPONENTS NEED CHANGING');
      PUT SKIP LIST('RANGE PLEASE:'); GET LIST(ULO,UHI);
      WORD(3)=' '; GO TO PROC8C; END; GO TO GETCOM;
 PROC16C: MUPP=FIRST; IF WORD(3)='NAME' THEN FNAME,MNAME=WORD(4);
      GO TO GETCOM;

 PROC(17): /* ALTLIST */
      AF='0'B;
      J1=0; DO I=1 TO 10; IF NAMCALT(I)¬=' ' THEN DO;
      AF='1'B;
      J1=J1+1; PUT EDIT(NAMCALT(I))  (COL(12*J1),A); END; END;
      IF ¬AF THEN DO; PUT SKIP LIST('NO CERT. ALTERN.');
      GO TO LISTPRB; END;
      PUT SKIP(2); DO I=1 TO NUTN; MUPP=OLIST(I); IF CHAINP(NSUB(I))
      =NULL THEN DO; UNIFP=UNIPTR(NSUB(I)); PUT EDIT(UTNAME(I))
      (COL(2),A(10)); J1=0; DO J=1 TO 10; IF NAMCALT(J)¬=' ' THEN DO;
      J1=J1+1; X=ULO+CALX(J)*(UHI-ULO); PUT EDIT(X)
      (COL(12*J1),F(10,3)); END; END; END; END;
 LISTPRB:  AF='0'B;
      PUT SKIP(2) LIST('CERT EQUIV. TABLE FOR PROB ALTERN');
      J1=0; DO I=1 TO 5; IF NAMPALT(I)¬=' ' THEN DO;
      AF='1'B;
      J1=J1+1; PUT EDIT(NAMPALT(I))  (COL(12*J1),A); END; END;
      IF ¬AF THEN DO; PUT SKIP LIST('NO PROB. ALTERN.');
      GO TO GETCOM; END;
      PUT SKIP(2); DO I=1 TO NUTN; MUPP=OLIST(I); IF CHAINP(NSUB(I))
      =NULL THEN DO; UNIFP=UNIPTR(NSUB(I)); PUT EDIT  (UTNAME(I))
      (COL(2),A(10)); J1=0; DO J=1 TO 5; IF NAMPALT(J)¬=' ' THEN DO;
      J1=J1+1; CALL UNINV(EUP(J),X); X=ULO+X*(UHI-ULO);
      PUT EDIT(X)  (COL(12*J1),F(10,3)); END; END; END; END;
```

U.M20225.11940.MUPCAP.PLI

```
        GO TO GETCOM;

    PROC(18): /* DISPLAY */
        IF WORD(2)=FNAME THEN DO; MUPP=FIRST; GO TO PROC18C; END;
        DO I=1 TO NUTN; IF UTNAME(I)=WORD(2) THEN GO TO PROC18B; END;
        PUT SKIP LIST('ATTRIB NOT FOUND'); GO TO GETCOM;
    PROC18B:  MUPP=OLIST(I); IF CHAINP(NSUB(I))=NULL THEN DO;
        UNIPP=UNIPTR(NSUB(I)); PUT SKIP EDIT('RANGE:',ULO,UHI)
        (COL(2),A,(2)F(10,3)); IF UTYPE=0 THEN PUT SKIP LIST
        ('UTYPE IS LINEAR');
        ELSE IF UTYPE=1 THEN DO;
        PUT SKIP LIST('UTYPE IS CONSTANT RISK   U(X)=B(1-EXP(-CX))');
        PUT EDIT('B=',UXP(1),'C=',UYP(1),'VARIABLE NORMALIZED')
        (COL(2),A,F(8,3),X(1),A,F(8,3),X(2),A);
        IF UXP(1)>0 THEN ANAME='AVERSE'; ELSE ANAME='PRONE';
        PUT EDIT ('RISK',ANAME) (COL(2),A,X(1),A);
        END;
        ELSE IF UTYPE=2 THEN DO;
        PUT SKIP LIST('UTYPE IS PIECEWISE LINEAR');
        DO J=1 TO NUP; IF UHI<ULO THEN DO; X=UXP(NUP-J+1);
        Y=UYP(NUP-J+1); END; ELSE DO; X=UXP(J); Y=UYP(J); END;
        X=ULO+X*(UHI-ULO); PUT EDIT('U(',X,')=',Y)
        (COL(2),A,F(10,3),A,F(8,3)); END; END; GO TO GETCOM; END;
        MUPP=CHAINP(NSUB(I));
    PROC18C: PUT SKIP LIST('LISTING OF K FACTORS'); SUMK=0;
        DO J=1 TO NUMAT; SUMK=SUMK+SMALLK(J);
        PUT EDIT(UNAME(J),SMALLK(J)) (COL(2),A,F(8,3)); END;
        PUT SKIP EDIT('BIGK=',CAPK,'SUM K''S =',SUMK)
        (COL(2),A,F(8,3),X(1),A,F(8,3)); GO TO GETCOM;

    PROC(19): /* FRACTILE */
        CALL UNIGET; DO J=1 TO 5; IF NAMPALT(J)=WORD(3) THEN
        GO TO PROC19B; END;
        PUT SKIP LIST('ALTERN. NOT FOUND'); GO TO GETCOM;
    PROC19B:  PUT SKIP LIST('CUM DISTRIB FOR THE ALTERN.');
        DO I=1 TO NF(J); IF UHI<ULO THEN DO; X=PALT(J).XF(NF(J)-I+1);
        Y=1-PALT(J).CF(NF(J)-I+1); END; ELSE DO;
        X=PALT(J).XF(I); Y=PALT(J).CF(I); END;
        X=ULO+X*(UHI-ULO); PUT EDIT('F(',X,')=',Y)
        (COL(2),A,F(10,3),A,F(8,3)); END; GO TO GETCOM;

    PROC(20): /* LOTTERY */
        CALL UNIGET;
        IF WORD(3)¬='CE'|WORD(3)¬='P' THEN DO;
        N=WORD(3); IF N>1 THEN DO;
        PUT SKIP LIST('LOTTERY ENDPTS. PLEASE?');
        GET LIST((XIN(J) DO J=1 TO N));
        PUT SKIP LIST('CORRESP. PROBABILITIES PLEASE?');
        GET LIST((YIN(J) DO J=1 TO N)); X=0; DO J=1 TO N;
        XIN(J)=(XIN(J)-ULO)/(UHI-ULO); CALL UNICAL(XIN(J),ANS);
        X=X+ANS*YIN(J); END; GO TO PROC20B; END; END;
    PUT SKIP LIST('INPUT LOTTERY ENDPTS. (BOTTOM,TOP) AND THE CE OR P');
         GET LIST((XIN(J) DO J=1 TO 3));
        DO J=1 TO 2; XIN(J)=(XIN(J)-ULO)/(UHI-ULO);
        CALL UNICAL(XIN(J),ANS); YIN(J)=ANS; END;
        IF WORD(3)='CE' THEN DO;
        X=XIN(3)*XIN(2)+(1-XIN(3))*XIN(1); PROC20B: CALL UNINV(X,ANS);
```

U.M20225.11940.MUFCAP.PLI

```
        X=ULO+ANS*(UHI-ULO); PUT SKIP EDIT('CE FOR LOTTERY= ',X)
        (COL(2),A,F(10,3)); END;
        ELSE IF WORD(3)='P' THEN DO;
        XIN(3)=(XIN(3)-ULO)/(UHI-ULO); CALL UNICAL(XIN(3),ANS);
        X=(ANS-XIN(1))/(XIN(2)-XIN(1)); PUT SKIP EDIT
        ('P FOR LOTTERY= ',X) (COL(2),A,F(8,3)); END;
        GO TO GETCOM;

    PROC(21): /* IMAP */
        CALL GET2; MUFP=OLIST(I); PUT SKIP LIST
        ('INPUT INDIF PT. THROUGH WHICH CURVE WILL PASS:');
        GET LIST((XIN(J) DO J=1 TO 2));
        IF ¬UF THEN GO TO PROC21B; DO J=1 TO 2;
        UNIFP=SUBAT(NSUB(JARY(J))).UNIPTR;
        XIN(J)=(XIN(J)-ULO)/(UHI-ULO); CALL UNICAL(XIN(J),ANS);
        XIN(J)=ANS; END;
    PROC21B:  X=SMALLK(K1)*XIN(1)+SMALLK(K2)*XIN(2)+
        CAPK*SMALLK(K1)*SMALLK(K2)*XIN(1)*XIN(2);
        PUT SKIP LIST('INPUT NUMBER OF PTS. FOR MAP:');
        GET LIST(N); PUT SKIP EDIT('INPUT ',UNAME(K1),
        ' VALUES FOR MAP') (COL(2),A,A,A);
        GET LIST ((YIN(J) DO J=1 TO N));
        IF UF THEN DO; UNIFP=UNIPTR(K1); DO J=1 TO N;
        XIN(J)=(YIN(J)-ULO)/(UHI-ULO);
        CALL UNICAL(XIN(J),ANS); XIN(J)=ANS; END; END;
        ELSE DO J=1 TO N; XIN(J)=YIN(J); END;
        DO J=1 TO N;
        XIN(J)=(X-SMALLK(K1)*XIN(J))/(CAPK*SMALLK(K1)*SMALLK(K2)
        *XIN(J)+SMALLK(K2)); IF UF THEN DO;
        UNIFP=UNIPTR(K2); CALL UNINV(XIN(J),ANS);
        XIN(J)=ULO+ANS*(UHI-ULO); END; END;
        PUT SKIP LIST ('INDIFFERENCE PTS');
        PUT SKIP EDIT (('(',YIN(J),',',XIN(J),')' DO J=1 TO N))
        (COL(2),A,F(10,3),A,F(10,3),A);
        PUT SKIP EDIT('UTIL FOR CURVE WITH OTHER ATTR. AT 0',
        X) (COL(2),A,X(3),F(8,3));
        GO TO GETCOM;

    PROC(22): /* STOP */
        PUT SKIP LIST('THANKS FOR USING MUFCAP'); STOP;

    PROC(23): /* DELU */
        CALL DELUT(WORD(2)); CALL SETOFF;
        PUT SKIP LIST('K''S NEED NORMALIZING AND BIGK NEEDS SETTING');
        GO TO GETCOM;

    DELUT: PROC(TNAME);
        DCL TNAME CHAR(12),I FIXED, IS FIXED;
        DO I=1 TO NUMN; IS=NSUB(I); IF UTNAME(I)=TNAME THEN DO;
        MUFP=OLIST(I); GO TO FOUND; END; END;  PUT SKIP EDIT
        (TNAME,' NOT IN USE') (COL(2),A,A); GO TO GETCOM;
    FOUND: IF (NUMAT-1)=0 THEN DO; PUT SKIP LIST
        ('PLEASE DELETE THE MUF TO WHICH THIS ATTR. BELONGS');
        RETURN; END;
        NUMAT=NUMAT-1; DO I=1 TO NUMAT; IF(I>=IS) THEN SUBAT(I)=
        SUBAT(I+1); END;  RETURN;
    END DELUT;
```

U.M20225.11940.MUPCAP.PLI

```
    PROC(24): /* ADDU */
         CALL ADDUT(WORD(2),WORD(3)); CALL SETOPP;
         PUT SKIP LIST('ALTERN. COMP. MAY NEED SETTING');
         PUT SKIP LIST('K''S NEED NORMALIZING AND BIGK NEEDS SETTING');
         GO TO GETCOM;

    ADDUT: PROC(TN1,TN2);
         DCL (TN1,TN2) CHAR(12),I FIXED,IS FIXED,TEMP PTR;
         IF TN2=PNAME THEN DO; MUPP=FIRST; GO TO FOUND; END;
         DO I=1 TO NUTN; IF UTNAME(I)=TN2&OLIST(I)->CHAINP(NSUB(I))¬=
         NULL THEN DO; MUPP=OLIST(I)->CHAINP(NSUB(I)); GO TO FOUND; END;
         END;
         PUT SKIP EDIT(TN2,' NOT A MUP') (COL(2),A,A); GO TO GETCOM;
    FOUND: NUMAT=NUMAT+1; IS=NUMAT; UNAME(NUMAT)=TN1;
         SMALLK(NUMAT)=1.E0/NUMAT; PUT SKIP EDIT(
         'INPUT RANGE FOR ATTR. ',NUMAT,' OF UTIL FUNCTION ',TN2)
         (COL(2),A,F(2),A,A); GET LIST(R1,R2);
         IF R1=R2 THEN DO;PUT SKIP LIST('HOW MANY ATTR. IN THIS MUP?:');
         GET LIST (NAT); TEMP=MUPP; ALLOCATE MUP IN (BODY);
         NUMAT=NAT; TEMP->CHAINP(IS)=MUPP; MNAME=TEMP->UNAME(IS);
         CAPK=0; CALL GETMULT; MUPP=TEMP; END;
         ELSE DO; CHAINP(NUMAT)=NULL; ALLOCATE UNIP IN (BODY);
         UNIPTR(NUMAT)=UNIPP; UTYPE=0; ULO=R1; UHI=R2; END;
          RETURN; END ADDUT;

    PROC(25): /* SWITCH */
         CALL SWITCHU(WORD(2),WORD(3)); CALL SETOPP;
         PUT SKIP LIST('K''S IN BOTH MUPS NEED NORMALIZING');
         GO TO GETCOM;

    SWITCHU: PROC(TN1,TN2);
         DCL (TN1,TN2) CHAR(12),I FIXED, TEMP PTR;
         JARY(1),JARY(2)=0; DO I=1 TO NUTN; DO J=1 TO 2;
         IF WORD(1+J)=UTNAME(I) THEN DO; JARY(J)=I;
         GO TO SWB; END; END;
    SWB: IF (JARY(1)>0&JARY(2)>0) THEN GO TO SWC; END;
         IF (JARY(1)>0 &TN2=PNAME) THEN GO TO SWD;
         PUT SKIP LIST('ATTR. NOT BOTH FOUND'); GO TO GETCOM;;
    SWC: J1=JARY(1);J2=JARY(2);K1=NSUB(J1); MUPP=OLIST(J2);
         IF CHAINP(NSUB(J2))=NULL THEN DO;
         PUT SKIP EDIT(TN2,' IS NOT A MUP') (COL(2),A,A);
         GO TO GETCOM; END; MUPP=CHAINP(NSUB(J2));
         TEMP=OLIST(J1); GO TO SWE;
    SWD: J1=JARY(1); K1=NSUB(J1); MUPP=FIRST; TEMP=OLIST(J1);
    SWE: NUMAT=NUMAT+1; UNAME(NUMAT)=TN1; SMALLK(NUMAT)=
         1.E0/NUMAT;
         IF TEMP->CHAINP(K1)=NULL THEN DO;
         UNIPTR(NUMAT)=TEMP->UNIPTR(K1); CHAINP(NUMAT)=NULL; END;
         ELSE CHAINP(NUMAT)=TEMP->CHAINP(K1); CALL DELUT(TN1);
         RETURN; END SWITCHU;

    PROC(26): /* INTERPK */
         IF WORD(2)=PNAME THEN DO; MUPP=FIRST; GO TO PROC26D; END;
         DO I=1 TO NUTN; IF WORD(2)=UTNAME(I) THEN DO;
         MUPP=OLIST(I); GO TO PROC26B; END;END;
```

U.M20225.11940.MUPCAP.PLI

```
PROC26C: PUT SKIP LIST('MUP NOT FOUND'); GO TO GETCOM;
PROC26B: IF CHAINP(NSUB(I))=NULL THEN GO TO PROC26C;
     MUPP=CHAINP(NSUB(I));
PROC26D: DO I=1 TO NUMAT; IF CHAINP(I)¬=NULL THEN
     PUT SKIP EDIT(UNAME(I),' BIGK= ',CHAINP(I)->CAPK,
     'INTERBK= ',CAPK*SMALLK(I))
     (COL(2),A,A,F(8,3),X(2),A,F(8,3)); END; GO TO GETCOM;
END MUPCAP; /*        */
```

U.M20225.11940.BIGK.PLT

```
  /* CALCULATE K IN MULT. FORM */
  BIGK: PROC (PK,NUMAT) RETURNS (FLOAT);
        DCL PK(*),ITERATE LABEL;

  /* CALCULATE SUM OF PK'S */
        SUMK=0; DO I=1 TO NUMAT; SUMK=PK(I)+SUMK; END;
        IF ABS(SUMK-1)<1.E-5 THEN RETURN(0.E0);
        IF SUMK<1. THEN GO TO POSK;

  /* -1 < K < 0. TRY BK=-.5 */
        NEGK: BK=-.5;  ADJ=-.5;  ITERATE=HOMEIN; GO TO TEST;

  /* 0<K . TRY BK=1. */
  POSK: BK=1.; ITERATE = POSK1; GO TO TEST;
  POSK1: IF SR>SL THEN GO  TO  POSK2;
        BK=BK+BK; GO TO TEST;
  POSK2: ITERATE=HOMEIN; ADJ=.25*BK; IF BK=1. THEN ADJ=.5;
        BK=BK-ADJ; GO TO TEST;

  HOMEIN: ADJ=.5*ADJ; IF SR < SL THEN BK=BK+ADJ; ELSE BK=BK-ADJ;

  /* EVALUATE SIDES OF 1+K=PROD(1+KK(I))   */
  TEST: SL=1. + BK; SR=1.; DO I=1 TO NUMAT;
        SR=SR*(1.+BK*PK(I)); END;
        IF ABS(SR-SL) <1.E-3 THEN RETURN(BK);
        GO TO ITERATE;
  END BIGK;
```

U.M20225.11940.UNI.PLI

```
/* FITS THE FORM U(X)=B*(1-EXP(-CX)) .
     FUNCTION IS MONOTONIC INCREASING ON THE INTERVAL
     0,1. C IS THE RISK AVERSION CONSTANT  */
UNIEXP: PROC (XMID,XLO,XHI,B,C) ;
     DCL ITERATE LABEL;
     SL=XMID;

/* CHECK ON RANGE TO  SEARCH FOR C */
     IF  (XMID-XLO)/(XHI-XLO)>.5 THEN CSIGN=-1.; ELSE CSIGN=1.;
/* TRY |C| =1  */
C=1.*CSIGN; ITERATE=RANGEFIND; GO TO TEST;

RANGEFIND: IF (SR-SL)*CSIGN<0. THEN GO TO RANGEFOUND;
     C=C+C; GO TO TEST;

RANGEFOUND: ITERATE=HOMEIN;
     IF ABS(C)=1 THEN ADJ=.5*C; ELSE ADJ=.25*C;
     C=C-ADJ; ADJ=ADJ*CSIGN; GO TO TEST;

HOMEIN: ADJ=.5*ADJ; IF SR<SL THEN C=C-ADJ; ELSE C=C+ADJ;

/* EVALUATE TEST FOR C */

TEST: SR=-LOG(.5*(EXP(-C*XLO)+EXP(-C*XHI)))/C;
     IF ABS(SR-SL)<1.E-3 THEN GO TO OUT; GO TO ITERATE;
OUT: B=1/(1-EXP(-C)); RETURN;
     END UNIEXP;
```

APPENDIX C

SOME ALGORITHMS USED IN MUFCAP

Apart from implementing the formula definitions necessary to calculate particular quantities, certain MUFCAP routines make use of some numerical analysis techniques or algorithms. These are discussed in the appendix.

C.1 Calculation of the Parameter k in the Multiplicative Utility Function

A subroutine called BIGK calculates the k in the multiplicative utility function using (3) described in Section 2.1. The algorithm employed is an iterative one suggested in Keeney [9]. Essentially, depending on the value of $\sum_{i=1}^{n} k_i$, an interval is isolated where the value of k must lie. Once a finite interval has been found where k lies, the bisection method for finding a real root as described in Hamming [5] is used to calculate k to the desired accuracy.

When $\sum_{i=1}^{n} k_i > 0$, we know $-1 < k < 0$ and we have our interval immediately. When $\sum_{i=1}^{n} k_i < 0$, BIGK tries successive powers of 2 until a comparison of the two sides of (3) indicates that a real root lies in the interval $(2^{n-1}, 2^n)$ where n is as large as necessary for the particular case. The bisection method is then used on this interval to calculate k to the desired accuracy.

Hamming [5] explains why the bisection method is a good one to use as opposed to other methods. Aside from being easy to implement, it is less vulnerable to ill-behavior and round-off error than other algorithms.

C.2 Calculation of the Constant Risk Scalar Utility Function

A subroutine called UNIEXP calculates the parameter c in the constant risk form $u(x) = a + b(1 - e^{-cx})$ where the conditions that $u(0) = 0$ and $u(1) = 1$ impose the values $a = 0$ and $b = 1/(1 - e^{-c})$. Internally, MUFCAP "normalizes" all scalar attributes to run between 0 and 1. For constant risk attributes, MUFCAP internally has the attribute increasing on the interval [0, 1]. On input and output, the appropriate scale conversions are always made so the internal normalization is transparent to the user except in displaying the parameters b and c.

One reason for normalization is that calculating utility values using the computer's exponential algorithm is made more accurate when the argument for the exponential function is not excessively large. This consideration is discussed in Schlaifer [16].

UNIEXP is very similar to BIGK in its algorithmic method. The equation used is similar to that in Schlaifer [16] where he discusses fitting constant risk forms. Again, the bisection method is used because of its nice "idiot-proof" properties.

C.3 Calculation of Gradient Components

The formula for the quantity $\frac{\partial u}{\partial u_i}$ is derived in a straightforward manner from either (1) or (2) in Section 2.1. The quantity $\left.\frac{\partial u}{\partial x_i}\right|_{\underline{x}}$, where x_i designates a scalar attribute amount and $\underline{x}$ is a "certainty" alternative, is calculated via the chain rule $\frac{\partial u}{\partial x_i} = \frac{\partial u}{\partial u_i} * \frac{du_i}{dx_i}$. Because of the various forms possible for u_i, the quantity $\frac{du_i}{dx_i}$ is calculated by using the approximation $\frac{du_i}{dx_i} = [u_i(x_i + .01) - u_i(x_i)]/[(x_i + .01) - (x_i)]$. Remember (as explained in C.2) that internally, MUFCAP scales all variables to run between 0 and 1. This approximation was felt to be adequate for the purpose of the program. When u_i is a piecewise linear form, the expression for the derivative when x_i is a breakpoint represents the change in the function when moving in the direction from the first range value to the second range value.

APPENDIX D

MUFCAP'S OVERALL PROGRAM DESIGN

This appendix gives an overview of the operating characteristics and programming design of MUFCAP.

D.1 Language and Operating System Considerations

The package is composed of three procedures which are compiled separately and then linked together. The main procedures is called MUFCAP and contains the bulk of the package making use of internal procedures sharing common data bases. The two external subroutines are BIGK and UNIEXP which are described in Appendix C.

The entire package is written in PL/1 using IBM's PL/1 optimizer compiler. Features of PL/1 which are used heavily are its based storage capabilities for managing linked lists and its recursive function capabilities for dealing with nested multiattribute utility functions. It is conceivable that a MUFCAP without nesting or a single level of nesting could be written in a language like FORTRAN, but a more powerful language such as PL/1 seems much more suitable for the general nature of this programming task. A helpful reference for PL/1 is Pike [2].

MUFCAP currently runs on an IBM 370/165 using IBM's Timesharing Option, TSO. It runs in a partition of 300K when using files for input and output although I believe it could get by with less memory. MUFCAP stores information on files with a fixed record format of blocksize 13000 bytes using IBM 3330 disk drives. These file characteristics correspond to a structure in the program designed to have room for roughly twenty scalar attributes. These can be adjusted if certain data structures in the program are made larger or smaller and if a track overflow option is used on the IBM system for blocksizes larger than 13000 bytes. To create a dataset for MUFCAP use, the following TSO commands work for the current version:

```
attrib trib recfm(f) blksize(13000) lrecl(13000)
allocate file(name1) dataset(name2) using(trib)
space(5 2) block(13000)
```

The parameter 'name1' is the name MUFCAP uses in the READ and SAVE commands. After a dataset has been created, new datasets may be more easily created by copying an old one into a new one using the TSO COPY command. Before using MUFCAP, all datasets which are to be read or saved should be allocated using the TSO ALLOCATE command. This is illustrated in Section 5.1.4.

MUFCAP is 861 cards long. Some estimates of relevant costs are:

compilation of program package	\$12 - \$15
linking the programs into a load module	\$2 - \$3
a one-hour assessment and use session	\$5

D.2 Data Structures in MUFCAP

There are two central data structures in MUFCAP; one is for MUF's and the other for UNIF's. For any MUF required during the program, a data structure is allocated with provision for the following information: the parameter k for the function, an associated function name and the number of attribute arguments of the function. Each MUF has room for 12 attribute arguments. For each of these arguments, the MUF structure contains the following information: a pointer to another MUF structure if an attribute argument is a vector, a pointer to a UNIF structure if the attribute arguments is a scalar, the k_i for that attribute and the name of that attribute.

When a scalar utility function or UNIF is required during the program, a data structure is allocated with provision for the following information: two range boundary values for the scalar attribute, the utility function type, room for 10 attribute amounts and the utilities of those amounts for "certain" alternatives, location for up to 30 parameters to specify the utility function (e.g., 15 abscissa and ordinate values) and room for 5 probabilistic alternatives each denoted by a cumulative piecewise-linear distribution which may be specified by as many as 9 points.

Along with these data structures are three arrays which contain the names of all the attributes, a pointer to the MUF where the attribute is "located" and the argument number of the attribute in that MUF. By scanning these arrays, the program finds the desired attribute name and then has pointers to all the information necessary to perform calculations involving that attribute name.

Data structures are allocated when needed in a designated area which can be written out on a file using the SAVE command. The relevant pointers are expressed as offsets to the beginning of this area.

D.3 Recursive Functions and Nesting

The data structures and PL/1's recursive procedure capability enable the same algorithms to handle any level of nesting. An example will illustrate the point. Suppose the program needs to evaluate a MUF. A routine is called for this purpose using (1) or (2) of Section 2.1 after a pointer has been set pointing to the appropriate MUF. Now, suppose during the course of evaluating (1) or (2), a vector attribute is encountered having an associated MUF of its own. At this point, the routine merely saves the pointer to the current MUF, sets up a pointer to the nested MUF, calls itself to evaluate the nested MUF and takes that value and uses it as it resumes its previous calculation. PL/1's recursive procedure capability handles all the appropriate bookkeeping. MUFCAP uses recursive routines to perform MUF evaluations, to calculate gradients, to chain through the multiattribute utility function structure in setting up the three arrays mentioned in Section D.2 and in setting up a nested MUF.

D.4 Evaluating Alternatives

As explained in Section D.2, each UNIF structure contains room for specifying the scalar component for each of the various alternatives. Whenever an alternative is specified or a scalar utility function is set or changed, MUFCAP automatically calculates the expected utility of that scalar attribute for the alternative affected. By saving the value of $E[u_i(x_i)]$ as well as x_i, MUFCAP saves a lot of redundant calculations when sensitivity analysis is performed involving only changes in the k_i's. There are separate routines for calculating expected utilities for scalar utility functions depending on the scalar utility function type.

Various flags in the program enable MUFCAP to keep track of when it is dealing with a certain alternative or a probabilistic alternative. The names for alternatives are contained in appropriate arrays and are saved when the SAVE command is used.

D.5 Program Flow

Program flow in MUFCAP revolves around the command processor section. This section determines what kind of command is requested and then transfers to the appropriate command execution section. After it is finished executing the command, the execution section transfers back to the command processor section for another command.

The execution sections are not internal procedures but invoke procedures as is necessary. Operations which are invoked by more than one execution section or are repeated fairly often are incorporated into internal procedures.

APPENDIX E

TRADEOFF PROPERTIES OF THE ADDITIVE AND MULTIPLICATIVE FORMS

Tradeoffs between attributes X_1 and X_2 with the other attributes $(X_3,\ldots,X_n)$ held fixed can be represented by an indifference map. An indifference map is a set of indifference curves each having the property that no point on a particular curve is preferred to any other point on that same curve. That is to say, all the points on a particular curve are indifferent to each other. The "points" here are consequences $\underline{x}$ with $(x_3,\ldots,x_n)$ held fixed but x_1 and x_2 allowed to vary. An indifference curve is generated when we choose a pair (x_1, x_2) and display all the allowable (x_1, x_2) pairs which are indifferent to it.

When the requisite assumptions to imply either (1) or (2) are satisfied (Section 2.1), an indifference curve is represented analytically by (x_1, x_2) pairs satisfying

$$k_1u_1(x_1) + k_2u_2(x_2) + kk_1k_2u_1(x_1)u_2(x_2) = \text{constant} \qquad \text{(E-1)}$$

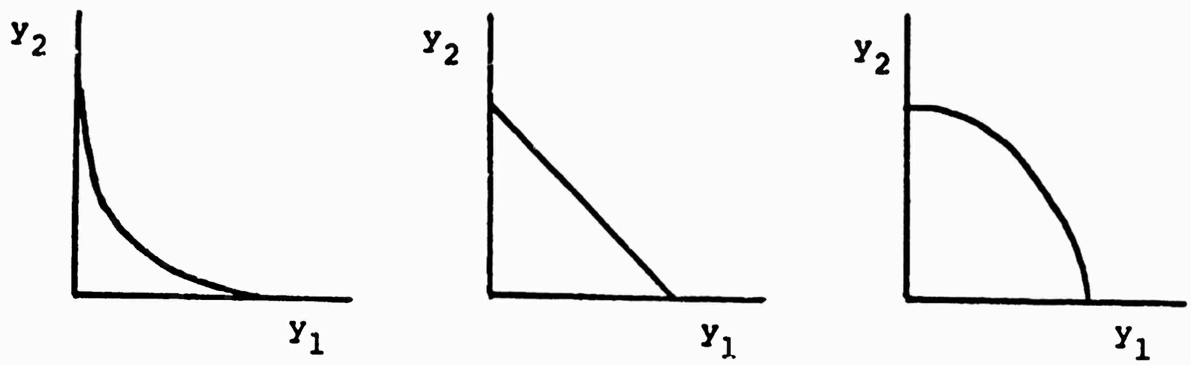
This equation results from the fact that when two consequences $\underline{x}'$ and $\underline{x}''$ are indifferent, $u(\underline{x}') = u(\underline{x}'')$. When $k = 0$ in (E-1), this corresponds to the additive form. When $k \neq 0$, this corresponds to the multiplicative form.

From (E-1) we can see that (x_1, x_2) pairs which are indifferent to each other remain indifferent regardless of the level at which $(x_3, \ldots, x_n)$ happen to be fixed. Suppose we wished to generate an indifference curve using only tradeoff information between X_1 and X_2. Since k in general depends on the other k_i via (3) (Section 2.1), we can generate two independent equations using two sets of indifference pairs varying x_1 and x_2. Using these, we can express k and k_2 in terms of k_1. Setting k_1 to an arbitrary number corresponds to setting the constant on the right hand side of (E-1) to an arbitrary constant. This does not affect which points are indifferent to each other. Thus, two sets of indifference pairs which are independent enables us to calculate the parameters of an equation for indifference curves. Then, if we are given any point (x_1, x_2), we can generate all the (x_1, x_2) pairs which are indifferent to it. To summarize, indifference curves representing tradeoffs between X_1 and X_2 can be generated using only information concerning preferences over (x_1, x_2) pairs and need not require any specific tradeoff information concerning the other attributes.

If we let $y_1 = u_1(x_1)$ and $y_2 = u_2(x_2)$, equation (E-1) becomes

$$k_1 y_1 + k_2 y_2 + k k_1 k_2 y_1 y_2 = \text{constant}$$

An indifference curve in (y_1, y_2) space as opposed to (x_1, x_2) space is always a hyperbola.



k > 0 k = 0 k < 0

Figure E-1

Indifference Curves in Utility Space

Now let us examine the effect of nesting on indifference curves. We will examine a three-attribute case of the form $u = u(u_a, u_b)$ where $u_a = u_a(a)$ and $u_b = (u_s, u_t)$. Thus, the three single attributes involved are A, S and T.

In the multiplicative form, we have, symbolically (where the arguments of the utility functions have been left out for more concise notation),

$$1 + ku = (1 + kk_a u_a)(1 + kk_b u_b) \qquad \text{(E-2)}$$

$$1 + k'u_b = (1 + k'k_s u_s)(1 + k'k_t u_t) \qquad \text{(E-3)}$$

Substituting (E-3) into (E-2) yields

$$1 + ku$$

$$= (1 + kk_a u_a)(1 + kk_b/k'[(1 + k'k_s u_s)(1 + k'k_t u_t) - 1]) \qquad \text{(E-4)}$$

Now, note what happens if $k' = kk_b$

We then obtain

$$1 + ku$$

$$= (1 + kk_a u_a)(1 + [(1 + kk_b k_s u_s)(1 + kk_b k_t u_t) - 1])$$

$$= (1 + kk_a u_a)(1 + kk'_s u_s)(1 + kk'_t u_t) \qquad \text{(E-5)}$$

where $k'_s = k_b k_s$

$k'_t = k_b k_t$

Equation (E-5) is nothing but the multiplicative form for three attributes. Thus, if $k' = kk_b$, any pair of attributes has the preferential independence property and the indifference curve properties of (E-1) apply. However, if $k' \neq kk_b$, this is no longer true. We can no longer factor the expression for 1 + ku into three factors each dealing with a single attribute. Because of this, if u(a',s',t) = u(a",s",t), it is not necessarily the case that u(a',s',t') = u(a",s",t') where $t' \neq t$. That is to say, indifference curves between a and s depend on t when there is nesting and $k' \neq kk_b$.

MUFCAP has a command INTERBK which calculates the quantity kk_b and compares it to k' where b is any vector attribute in a particular MUF and k_b, k and k' are the analogous parameters to those in our example. If $kk_b \approx k'$, then the nesting of attributes into their own internal MUF may be unnecessary. Section 5.1.4 has an illustration of the use of INTERBK.