STUDIES IN *FUZZINESS*
AND *SOFT COMPUTING*

James J. Buckley
Leonard J. Jowers

Monte Carlo Methods in Fuzzy Optimization

Springer

James J. Buckley and Leonard J. Jowers

Monte Carlo Methods in Fuzzy Optimization

Studies in Fuzziness and Soft Computing, Volume 222

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

James J. Buckley and Leonard J. Jowers

Monte Carlo Methods in Fuzzy Optimization

Springer

Authors

Prof. James J. Buckley
Mathematics Department
University of Alabama at Birmingham
Birmingham, AL 35294
USA
E-mail: buckley@math.uab.edu

Dr. Leonard J. Jowers
Department of Computer and
Information Sciences
University of Alabama at Birmingham
Birmingham, AL 35294
USA
E-mail: LJJowers@uab.edu

ISBN 978-3-540-76289-8 e-ISBN 978-3-540-76290-4

Studies in Fuzziness and Soft Computing ISSN 1434-9922

Library of Congress Control Number: 2007938042

Typesetting by the authors and Scientific Publishing Services Pvt. Ltd.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To Julianne and Helen,

To Paula and “The Kids”

Contents

Part I: Introduction

1 Introduction

1.1 Introduction

The objective of this book is to introduce Monte Carlo methods to find good approximate solutions to fuzzy optimization problems. Many crisp (nonfuzzy) optimization problems have algorithms to determine solutions. This is not true for fuzzy optimization. There are other things to discuss in fuzzy optimization, which we will do later on in the book, like $\leq$ and $<$ between fuzzy numbers since there will probably be fuzzy constraints, and how do we evaluate $max/min\overline{Z}$ for $\overline{Z}$ the fuzzy value of the objective function.

This book is divided into four parts: (1) Part I is the Introduction containing Chapters 1-5; (2) Part II, Chapters 6-16, has the applications of our Monte Carlo method to obtain approximate solutions to fuzzy optimization problems; (3) Part III, comprising Chapters 17-27, outlines our "unfinished business" which are fuzzy optimization problems for which we have not yet applied our Monte Carlo method to produce approximate solutions; and (4) Part IV is our summary, conclusions and future research.

1.1.1 Part I

First we need to be familiar with fuzzy sets. All you need to know about fuzzy sets for this book comprises Chapter 2. For a beginning introduction to fuzzy sets and fuzzy logic see [2]. Three other items related to fuzzy sets, needed in this book, are also in Chapter 2: (1) in Section 2.5 we discuss how we have dealt in the past with determining $max/min(\overline{Z})$ for $\overline{Z}$ a fuzzy set representing the value of an objective function in a fuzzy optimization problem; (2) in Section 2.6 we present the three methods we will use in this book to determine which of the following possibilities $\overline{M} < \overline{N}$, $\overline{M} > \overline{N}$ or $\overline{M} \approx \overline{N}$ is true, for two fuzzy numbers $\overline{M}$, $\overline{N}$; and (3) in Section 2.7 we investigate dominated, and undominated, fuzzy vectors.

Chapter 3 introduces the random number generators that we will be using in the rest of the book. We will need sequences of crisp random vectors $v =$

$(x_1, ..., x_n)$ with $x_i \in [0, 1]$, $n \geq 3$. If we use a pseudo-random number generator to produce the v it is well known (see Chapter 3) that when we plot these points in $[0, 1]^n$ we can get empty regions and clustering. We need a method of getting sequences of v that will uniformly fill $[0, 1]^n$. Such a method already exists. They are called quasi-random number generators, introduced and discussed in Chapter 3.

Next we need to randomly generate sequences of fuzzy numbers and sequences of fuzzy vectors. We usually use triangular fuzzy numbers (TFNs) and quadratic fuzzy numbers (Section 2.2.1). The quadratic fuzzy numbers we use are called quadratic Bézier generated fuzzy numbers (QBGFNs) which are defined in Chapter 4. We show in this chapter how we use the sequences $v = (x_1, ..., x_n)$, from a quasi-random number generator, to get sequences of TFNs/QBGFNs and sequences of vectors of TFNs/QBGFNs. We use these results to show that the three methods of evaluating $\leq$ and $<$ between fuzzy numbers given in Sections 2.6.1 - 2.6.3 quite often give the same results.

Chapter 5 is about testing our sequences of fuzzy numbers/vectors for randomness. There are many tests for randomness for sequences of crisp numbers, but most are not applicable to fuzzy numbers. However, the run test can be extended to fuzzy numbers and our results are presented in this chapter. We have one randomness test for sequences of fuzzy vectors. But we do argue that our method of producing sequences of fuzzy vectors will uniformly fill the search space in a fuzzy optimization problem. That is exactly what is needed for a sequence of fuzzy vectors used to get an approximate solution to a fuzzy optimization problem.

1.1.2 Part II

Part II contains our applications of Monte Carlo methods to generating approximate solutions to fuzzy optimization problems. Fuzzy linear programming is in Chapters 6-9. Getting solutions, or approximate solutions, to fuzzy equations is the topic of Chapter 10. Applications to fuzzy regression is the theme for Chapters 11-14. The last two applications are to fuzzy game theory (Chapter 15) and to fuzzy queuing theory (Chapter 16).

1.1.3 Part III

The chapters here describe more fuzzy optimization problems that do not have algorithms that give an exact fuzzy solution. Therefore they are candidates for an approximate Monte Carlo solution. In each case we first outline the problem and then discuss how we might use Monte Carlo to generate approximate solutions, but we do not do it. We leave it for future research by the authors or interested readers.

1.1.4 Part IV

This consists of one short chapter containing a summary, our suggestions for future research and our conclusions.

1.2 Notation

It is difficult, in a book with a lot of mathematics, to achieve a uniform notation without having to introduce many new specialized symbols. Our basic notation is presented in Chapter 2. What we have done is to have a uniform notation within each chapter. What this means is that we may use the letters "a" and "b" to represent a closed interval $[a, b]$ in one chapter but they could stand for something else in another chapter. We will have the following uniform notation throughout the book: (1) we place a "bar" over a letter to denote a fuzzy set ($\overline{A}$, $\overline{B}$, etc.), and all our fuzzy sets will be fuzzy subsets of the real numbers; and (2) an α-cut of a fuzzy set (Chapter 2) is always denoted by "α".

We use the abbreviations: FN for fuzzy number; TFN for triangular fuzzy number; TrFN for trapezoidal fuzzy number, and QBGFN for quadratic Bézier generated fuzzy number (defined in Chapter 4). All fuzzy arithmetic is performed using α-cuts and interval arithmetic and not by using the extension principle (Chapter 2).We also use TBC to mean "to be completed" in Part III.

The term "crisp" means not fuzzy. A crisp set is a regular set and a crisp number is a real number. There is a potential problem with the symbol "$\leq$". It sometimes means "fuzzy subset" as $\overline{A} \leq \overline{B}$ stands for $\overline{A}$ is a fuzzy subset of $\overline{B}$ (defined in Chapter 2). However, also in Chapter 2, $\overline{A} \leq \overline{B}$ means that fuzzy set $\overline{A}$ is less than or equal to fuzzy set $\overline{B}$. The meaning of the symbol "$\leq$" should be clear from its use.

1.3 Previous Research

Mathematica has added random fuzzy numbers [11]. It can create "random" trapezoidal, Gaussian and triangular fuzzy numbers. They are represented by thin vertical bars similar to a histogram. We would need the functional expressions for the sides of the fuzzy numbers and it is not clear how we could get that information from Mathematica. The web site does not tell the user how these "random" fuzzy numbers are generated.

Most authors ([1],[5],[6],[8]-[10]) have used the following method to define random fuzzy numbers. Consider LR fuzzy numbers which we write as $(a, b, c)_{LR}$. Let m, l and r be three real-valued random variables with $l > 0, r > 0$. Then a random LR-fuzzy number is $(m, l, r)_{LR}$. The functions L and R are called the left and right membership functions, m is where the membership value equals one (vertex point) and $l(r) \geq 0$ is the left (right) spread of the fuzzy number. So, once you pick and fix L and R, the randomness is in the end points of the $\alpha = 0$ cut and the vertex point of the fuzzy number. We think the randomness should also be in the shape of the fuzzy number. That is, we should also be able to randomly change L and R.

Next, the paper [4] has another way to construct random fuzzy numbers. Let $F_i(x)$, $i = 1, 2, 3$, be probability distribution functions. Randomly choose $y \in [0, 1]$, then a random triangular fuzzy number has base $[F_1^{-1}(y), F_3^{-1}(y)]$ and vertex point at $F_2^{-1}(y)$. We have assumed that $F_1^{-1}(y) < F_2^{-1}(y) < F_3^{-1}(y)$

for y in $[0, 1]$. This just randomly produces a triangular fuzzy number but it always has the same shape (a triangle). [12] uses Gaussian fuzzy numbers to generate random fuzzy numbers but employs crisp Monte Carlo methods. The book [3] has one very short section on fuzzy Monte Carlo simulation

Finally, [7] generates a random triangular fuzzy number as $(m - 6/m/m + 2)$ for m uniform on $[1, 3]$ and a random trapezoidal fuzzy number as $(m - 4/m - 2, m + 4/m + 6)$ for m a standard normal random variable. Again the shape is always the same, straight line segments for the sides of the fuzzy number.

To complete a search for random fuzzy numbers/vectors we suggest putting the following items into your search engine: fuzzy random numbers, random fuzzy numbers, fuzzy Monte Carlo,... The authors believe that there are no other publications that cover in our detail generating random fuzzy numbers/vectors to give approximate solutions to fuzzy optimization problems. Our random fuzzy numbers will have random base, random vertex point and also (limited) random shape. We believe this gives a better picture of random fuzzy numbers for fuzzy Monte Carlo methods.

Beginning in Chapter 6 many different topics are covered until Chapter 27. Most chapters have a list of references at the end of the chapter. We give only a few key references to the topic were the interested reader may find other references. This book will not do a complete literature search on each area for you, but we do hope that we have given you a good start.

1.4 MATLAB/C++ Programs

Computer programs were written in MATLAB or C++. Only one MATLAB program is given in the book and it is at the end of Chapter 14. However, any of these programs can be obtained from the authors.

References

1. Bandemer, H., Gebhardt, A.: Bayesian Fuzzy Kriging, `www-stat.uni-klu.ac.at/~agebhard/fuba/`
2. Buckley, J.J., Eslami, E.: An Introduction to Fuzzy Logic and Fuzzy Sets. Springer, Heidelberg (2002)
3. El-Hawary, M.E. (ed.): Electric Power Applications of Fuzzy Systems. IEEE Power Engineering Series. Wiley-IEEE Press (1998)
4. Ferson, S., Ginzburg, L.: Hybrid Arithmetic. In: Proceedings ISUMA-NAFIPS, pp. 619–623 (1995)
5. Hong, D.H., Ahn, C.H.: Equivalent Conditions for Laws of Large Numbers for T-Related L-R Fuzzy Numbers. Fuzzy Sets and Systems 136, 387–395 (2003)
6. Jimenez-Gamero, M.D., Pino-Mejias, R., Rojas-Medar, M.A.: A Bootstrap Test for the Expectation of Fuzzy Random Variables. Comput. Statist. Data Anal. (2004)
7. Korner, R.: An Asymptotic $\alpha-$test for the Expectation of Random Fuzzy Variables. J. Statistical Planning and Inference 83, 331–346 (2000)
8. Liu, Y.-K., Chen, Y.-J.: Multicriteria Optimization Problem in Fuzzy Random Decision Systems. In: Proc. Third Int. Conf. Machine Learning and Cybernetics, Shanghai, China (2004)

9. Nather, W.: Linear Statistical Inference for Fuzzy Data. In: 3rd Int. Symposium Uncertainty Modeling and Analysis (ISUMA-NAFIPS), pp. 71–74 (1995)
10. Nather, W., Korner, R.: Linear Regression with Random Fuzzy Numbers. In: Ayyub, B.M., Gupta, M.M. (eds.) Uncertainty Analysis in Engineering and the Sciences, Kluwer, Boston (1998)
11. www.wolfram.com/products/mathematica
12. Zonouz, S.A., Miremadi, S.G.: A Fuzzy-Monte Carlo Simulation Approach for Fault Tree Analysis. In: IEEE 52nd Annual Reliability and Maintainability Symposium (RAMS), Newport Beach, CA, pp. 428–433 (January 2006)

2 Fuzzy Sets

2.1 Introduction

In this chapter we have collected together the basic ideas from fuzzy sets and fuzzy functions needed for the book. Any reader familiar with fuzzy sets, fuzzy numbers, the extension principle, α-cuts, interval arithmetic, and fuzzy functions may go on and have a look at Sections 2.5-2.7. In Section 2.5 we present a method that we have used in the past of maximizing/minimizing a fuzzy number $\overline{Z}$ which represents the value of some objective function in a fuzzy optimization problem. In Section 2.6 we are concerned with ordering a finite set of fuzzy numbers from smallest to largest to be used in our fuzzy Monte Carlo studies. Basically, given two fuzzy numbers $\overline{M}$ and $\overline{N}$, we need a method of deciding which of the following three possibilities is true: $\overline{M} < \overline{N}$, $\overline{M} \approx \overline{N}$, $\overline{M} > \overline{N}$. Three methods are discussed in Section 2.6. Section 2.7 discusses dominated and undominated fuzzy vectors needed in Chapter 9. Fuzzy vectors are vectors made up of fuzzy numbers. A good general reference for fuzzy sets and fuzzy logic is [4] and [19].

Our notation specifying a fuzzy set is to place a "bar" over a letter. So $\overline{A}$, $\overline{B}, \ldots$, $\overline{X}$, $\overline{Y}, \ldots$, $\overline{\alpha}$, $\overline{\beta}, \ldots$, will all denote fuzzy sets.

2.2 Fuzzy Sets

If Ω is some set, then a fuzzy subset $\overline{A}$ of Ω is defined by its membership function, written $\overline{A}(x)$, which produces values in $[0, 1]$ for all x in Ω. So, $\overline{A}(x)$ is a function mapping Ω into $[0, 1]$. If $\overline{A}(x_0) = 1$, then we say x_0 belongs to $\overline{A}$, if $\overline{A}(x_1) = 0$ we say x_1 does not belong to $\overline{A}$, and if $\overline{A}(x_2) = 0.6$ we say the membership value of x_2 in $\overline{A}$ is 0.6. When $\overline{A}(x)$ is always equal to one or zero we obtain a crisp (non–fuzzy) subset of Ω. For all fuzzy sets $\overline{B}$, $\overline{C}, \ldots$ we use $\overline{B}(x)$, $\overline{C}(x), \ldots$ for the value of their membership functions at x. Most of the fuzzy sets we will be using will be fuzzy numbers.

The term "crisp" will mean not fuzzy. A crisp set is a regular set. A crisp number is just a real number. A crisp matrix (vector) has real numbers as its components. A crisp function maps real numbers (or real vectors) into real numbers.

A crisp solution to a problem is a solution involving crisp sets, crisp numbers, crisp functions, etc.

2.2.1 Fuzzy Numbers

A general definition of a fuzzy number may be found in ([4],[19]), however our fuzzy numbers will be almost always triangular (shaped), or trapezoidal (shaped), fuzzy numbers. A triangular fuzzy number (TFN) $\overline{N}$ is defined by three numbers $a < b < c$ where the base of the triangle is the interval $[a, c]$ and its vertex is at $x = b$. Triangular fuzzy numbers will be written as $\overline{N} = (a/b/c)$. A triangular fuzzy number $\overline{N} = (1.2/2/2.4)$ is shown in Figure 2.1. We see that $\overline{N}(2) = 1$, $\overline{N}(1.6) = 0.5$, etc.

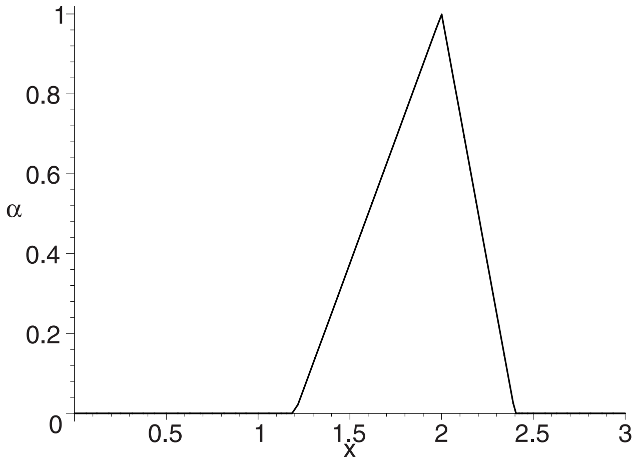


Fig. 2.1. Triangular Fuzzy Number $\overline{N}$

A trapezoidal fuzzy number $\overline{M}$ is defined by four numbers $a < b < c < d$ where the base of the trapezoid is the interval $[a, d]$ and its top (where the membership equals one) is over $[b, c]$. We write $\overline{M} = (a/b, c/d)$ for trapezoidal fuzzy numbers. Figure 2.2 shows $\overline{M} = (1.2/2, 2.4/2.7)$.

A triangular shaped fuzzy number $\overline{P}$ is given in Figure 2.3. $\overline{P}$ is only partially specified by the three numbers 1.2, 2, 2.4 since the graph on $[1.2, 2]$, and $[2, 2.4]$, is not a straight line segment. To be a triangular shaped fuzzy number we require the graph to be continuous and: (1) monotonically increasing on $[1.2, 2]$; and (2) monotonically decreasing on $[2, 2.4]$. For triangular shaped fuzzy number $\overline{P}$ we use the notation $\overline{P} \approx (1.2/2/2.4)$ to show that it is partially defined by the three numbers 1.2, 2, and 2.4. If $\overline{P} \approx (1.2/2/2.4)$ we know its base is on the interval $[1.2, 2.4]$ with vertex (membership value one) at $x = 2$. Similarly we define trapezoidal shaped fuzzy number $\overline{Q} \approx (1.2/2, 2.4/2.7)$ whose base is $[1.2, 2.7]$ and top is over the interval $[2, 2.4]$. The graph of $\overline{Q}$ is similar to $\overline{M}$ in Figure 2.2 but it has continuous curves for its sides.

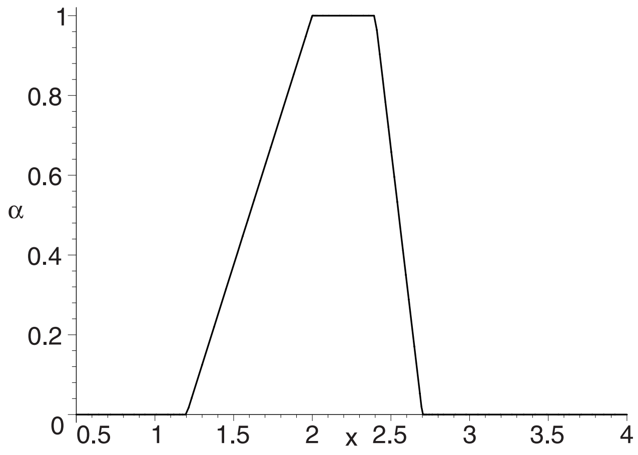


Fig. 2.2. Trapezoidal Fuzzy Number $\overline{M}$

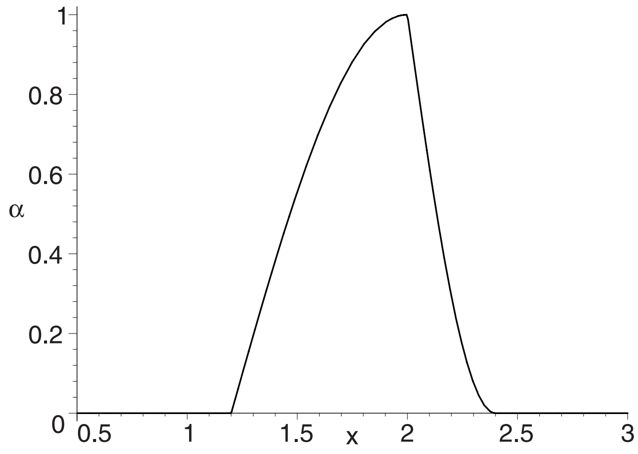


Fig. 2.3. Triangular Shaped Fuzzy Number $\overline{P}$

We will use special triangular shaped fuzzy numbers where their sides are defined by quadratic functions. These will be called quadratic fuzzy numbers (coded QBGFNs) and they are defined in Chapter 4.

Although we will be using triangular, trapezoidal(shaped) and quadratic fuzzy numbers throughout the book, many results can be extended to more general fuzzy numbers, but we shall be content to work with only these special fuzzy numbers.

2.2.2 Alpha-Cuts

Alpha-cuts are slices through a fuzzy set producing regular (nonfuzzy) sets. If $\overline{A}$ is a fuzzy subset of some set Ω, then an α-cut of $\overline{A}$, written $\overline{A}[\alpha]$ is defined as

$$\overline{A}[\alpha] = \{x \in \Omega | \overline{A}(x) \geq \alpha\}, \tag{2.1}$$

for all α, $0 < \alpha \leq 1$. The $\alpha = 0$ cut, or $\overline{A}[0]$, must be defined separately.

Let $\overline{N}$ be the fuzzy number in Figure 2.1. Then $\overline{N}[0] = [1.2, 2.4]$. Notice that using equation (2.1) to define $\overline{N}[0]$ would give $\overline{N}[0]$ = all the real numbers. Similarly, $\overline{M}[0] = [1.2, 2.7]$ from Figure 2.2 and in Figure 2.3 $\overline{P}[0] = [1.2, 2.4]$. For any fuzzy set $\overline{A}$, $\overline{A}[0]$ is called the support, or base, of $\overline{A}$. Many authors call the support of a fuzzy number the open interval (a, b) like the support of $\overline{N}$ in Figure 2.1 would then be $(1.2, 2.4)$. However in this book we use the closed interval $[a, b]$ for the support (base) of the fuzzy number.

The core of a fuzzy number is the set of values where the membership value equals one. If $\overline{N} = (a/b/c)$, or $\overline{N} \approx (a/b/c)$, then the core of $\overline{N}$ is the single point $x = b$. However, if $\overline{M} = (a/b, c/d)$, or $\overline{M} \approx (a/b, c/d)$, then the core of $\overline{M} = [b, c]$.

For any fuzzy number $\overline{Q}$ we know that $\overline{Q}[\alpha]$ is a closed, bounded, interval for $0 \leq \alpha \leq 1$. We will write this as

$$\overline{Q}[\alpha] = [q_1(\alpha), q_2(\alpha)], \tag{2.2}$$

where $q_1(\alpha)$ ($q_2(\alpha)$) will be an increasing (decreasing) function of α with $q_1(1) \leq q_2(1)$. If $\overline{Q}$ is a triangular shaped or a trapezoidal shaped fuzzy number then: (1) $q_1(\alpha)$ will be a continuous, monotonically increasing function of α in $[0, 1]$; (2) $q_2(\alpha)$ will be a continuous, monotonically decreasing function of α, $0 \leq \alpha \leq 1$; and (3) $q_1(1) = q_2(1)$ ($q_1(1) < q_2(1)$ for trapezoids). We sometimes check monotone increasing (decreasing) by showing that $dq_1(\alpha)/d\alpha > 0$ ($dq_2(\alpha)/d\alpha < 0$) holds.

For the $\overline{N}$ in Figure 2.1 we obtain $\overline{N}[\alpha] = [n_1(\alpha), n_2(\alpha)]$, $n_1(\alpha) = 1.2 + 0.8\alpha$ and $n_2(\alpha) = 2.4 - 0.4\alpha$, $0 \leq \alpha \leq 1$. Similarly, $\overline{M}$ in Figure 2.2 has $\overline{M}[\alpha] = [m_1(\alpha), m_2(\alpha)]$, $m_1(\alpha) = 1.2 + 0.8\alpha$ and $m_2(\alpha) = 2.7 - 0.3\alpha$, $0 \leq \alpha \leq 1$. The equations for $n_i(\alpha)$ and $m_i(\alpha)$ are backwards. With the y–axis vertical and the x–axis horizontal the equation $n_1(\alpha) = 1.2 + 0.8\alpha$ means $x = 1.2 + 0.8y$, $0 \leq y \leq 1$. That is, the straight line segment from $(1.2, 0)$ to $(2, 1)$ in Figure 2.1 is given as x a function of y whereas it is usually stated as y a function of x. This is how it will be done for all α-cuts of fuzzy numbers.

2.2.3 Inequalities

Let $\overline{N} = (a/b/c)$. We write $\overline{N} \geq \delta$, δ some real number, if $a \geq \delta$, $\overline{N} > \delta$ when $a > \delta$, $\overline{N} \leq \delta$ for $c \leq \delta$ and $\overline{N} < \delta$ if $c < \delta$. We use the same notation for triangular shaped and trapezoidal (shaped) fuzzy numbers whose support is the interval $[a, c]$.

If $\overline{A}$ and $\overline{B}$ are two fuzzy subsets of a set Ω, then $\overline{A} \leq \overline{B}$ means $\overline{A}(x) \leq \overline{B}(x)$ for all x in Ω, or $\overline{A}$ is a fuzzy subset of $\overline{B}$. $\overline{A} < \overline{B}$ holds when $\overline{A}(x) < \overline{B}(x)$, for all x. There is a potential problem with the symbol $\leq$. In some places in the book, for example see Section 2.6, $\overline{M} \leq \overline{N}$, for fuzzy numbers $\overline{M}$ and $\overline{N}$, means that $\overline{M}$ is less than or equal to $\overline{N}$. It should be clear on how we use "$\leq$" as to which meaning is correct.

2.2.4 Discrete Fuzzy Sets

Let $\overline{A}$ be a fuzzy subset of Ω. If $\overline{A}(x)$ is not zero only at a finite number of x values in Ω, then $\overline{A}$ is called a discrete fuzzy set. Suppose $\overline{A}(x)$ is not zero only at x_1, x_2, x_3 and x_4 in Ω. Then we write the fuzzy set as

$$\overline{A} = \{\frac{\mu_1}{x_1}, \cdots, \frac{\mu_4}{x_4}\}, \tag{2.3}$$

where the μ_i are the membership values. That is, $\overline{A}(x_i) = \mu_i$, $1 \leq i \leq 4$, and $\overline{A}(x) = 0$ otherwise. We can have discrete fuzzy subsets of any space Ω. Notice that α-cuts of discrete fuzzy sets of $\mathbb{R}$, the set of real numbers, do not produce closed, bounded, intervals.

2.3 Fuzzy Arithmetic

If $\overline{A}$ and $\overline{B}$ are two fuzzy numbers we will need to add, subtract, multiply and divide them. There are two basic methods of computing $\overline{A}+\overline{B}$, $\overline{A}-\overline{B}$, etc. which are: (1) extension principle; and (2) α-cuts and interval arithmetic.

2.3.1 Extension Principle

Let $\overline{A}$ and $\overline{B}$ be two fuzzy numbers. If $\overline{A}+\overline{B} = \overline{C}$, then the membership function for $\overline{C}$ is defined as

$$\overline{C}(z) = \sup_{x,y}\{\min(\overline{A}(x), \overline{B}(y))|x + y = z\}. \tag{2.4}$$

If we set $\overline{C} = \overline{A} - \overline{B}$, then

$$\overline{C}(z) = \sup_{x,y}\{\min(\overline{A}(x), \overline{B}(y))|x - y = z\}. \tag{2.5}$$

Similarly, $\overline{C} = \overline{A} \cdot \overline{B}$, then

$$\overline{C}(z) = \sup_{x,y}\{\min(\overline{A}(x), \overline{B}(y))|x \cdot y = z\}, \tag{2.6}$$

and if $\overline{C} = \overline{A}/\overline{B}$,

$$\overline{C}(z) = \sup_{x,y}\{\min(\overline{A}(x), \overline{B}(y))|x/y = z\}. \tag{2.7}$$

In all cases $\overline{C}$ is also a fuzzy number [19]. We assume that zero does not belong to the support of $\overline{B}$ in $\overline{C} = \overline{A}/\overline{B}$. If $\overline{A}$ and $\overline{B}$ are triangular (trapezoidal) fuzzy numbers then so are $\overline{A} + \overline{B}$ and $\overline{A} - \overline{B}$, but $\overline{A} \cdot \overline{B}$ and $\overline{A}/\overline{B}$ will be triangular (trapezoidal) shaped fuzzy numbers.

We should mention something about the operator "sup" in equations (2.4) – (2.7). If Ω is a set of real numbers bounded above (there is a M so that $x \leq M$, for all x in Ω), then $\sup(\Omega)$ = the least upper bound for Ω. If Ω has a maximum member, then $\sup(\Omega) = \max(\Omega)$. For example, if $\Omega = [0,1)$, $\sup(\Omega) = 1$ but if $\Omega = [0,1]$, then $\sup(\Omega) = \max(\Omega) = 1$. The dual operator to "sup" is "inf". If Ω is bounded below (there is a M so that $M \leq x$ for all $x \in \Omega$), then $\inf(\Omega)$ = the greatest lower bound. For example, for $\Omega = (0,1]$ $\inf(\Omega) = 0$ but if $\Omega = [0,1]$, then $\inf(\Omega) = \min(\Omega) = 0$.

Obviously, given $\overline{A}$ and $\overline{B}$, equations (2.4) – (2.7) appear quite complicated to compute $\overline{A} + \overline{B}$, $\overline{A} - \overline{B}$, etc. So, we now present an equivalent procedure based on α-cuts and interval arithmetic. First, we present the basics of interval arithmetic.

2.3.2 Interval Arithmetic

We only give a brief introduction to interval arithmetic. For more information the reader is referred to ([21],[22]). Let $[a_1, b_1]$ and $[a_2, b_2]$ be two closed, bounded, intervals of real numbers. If $\circ$ denotes addition, subtraction, multiplication, or division, then $[a_1, b_1] \circ [a_2, b_2] = [\alpha, \beta]$ where

$$[\alpha, \beta] = \{a \circ b | a_1 \leq a \leq b_1, a_2 \leq b \leq b_2\}. \tag{2.8}$$

If $*$ is division, we must assume that zero does not belong to $[a_2, b_2]$. We may simplify equation (2.8) as follows:

$$[a_1, b_1] + [a_2, b_2] = [a_1 + a_2, b_1 + b_2]\,, \tag{2.9}$$

$$[a_1, b_1] - [a_2, b_2] = [a_1 - b_2, b_1 - a_2]\,, \tag{2.10}$$

$$[a_1, b_1] \,/\, [a_2, b_2] = [a_1, b_1] \cdot \left[\frac{1}{b_2}, \frac{1}{a_2}\right], \tag{2.11}$$

and

$$[a_1, b_1] \cdot [a_2, b_2] = [\alpha, \beta], \tag{2.12}$$

where

$$\alpha = \min\{a_1 a_2, a_1 b_2, b_1 a_2, b_1 b_2\}, \tag{2.13}$$

$$\beta = \max\{a_1 a_2, a_1 b_2, b_1 a_2, b_1 b_2\}. \tag{2.14}$$

Multiplication and division may be further simplified if we know that $a_1 > 0$ and $b_2 < 0$, or $b_1 > 0$ and $b_2 < 0$, etc. For example, if $a_1 \geq 0$ and $a_2 \geq 0$, then

$$[a_1, b_1] \cdot [a_2, b_2] = [a_1 a_2, b_1 b_2], \tag{2.15}$$

and if $b_1 < 0$ but $a_2 \geq 0$, we see that

$$[a_1, b_1] \cdot [a_2, b_2] = [a_1 b_2, a_2 b_1]. \tag{2.16}$$

Also, assuming $b_1 < 0$ and $b_2 < 0$ we get

$$[a_1, b_1] \cdot [a_2, b_2] = [b_1 b_2, a_1 a_2], \tag{2.17}$$

but $a_1 \geq 0$, $b_2 < 0$ produces

$$[a_1, b_1] \cdot [a_2, b_2] = [a_2 b_1, b_2 a_1]. \tag{2.18}$$

2.3.3 Fuzzy Arithmetic

Again we have two fuzzy numbers $\overline{A}$ and $\overline{B}$. We know α-cuts are closed, bounded, intervals so let $\overline{A}[\alpha] = [a_1(\alpha), a_2(\alpha)]$, $\overline{B}[\alpha] = [b_1(\alpha), b_2(\alpha)]$. Then if $\overline{C} = \overline{A} + \overline{B}$ we have

$$\overline{C}[\alpha] = \overline{A}[\alpha] + \overline{B}[\alpha]. \tag{2.19}$$

We add the intervals using equation (2.9). Setting $\overline{C} = \overline{A} - \overline{B}$ we get

$$\overline{C}[\alpha] = \overline{A}[\alpha] - \overline{B}[\alpha], \tag{2.20}$$

for all α in $[0, 1]$. Also

$$\overline{C}[\alpha] = \overline{A}[\alpha] \cdot \overline{B}[\alpha], \tag{2.21}$$

for $\overline{C} = \overline{A} \cdot \overline{B}$ and

$$\overline{C}[\alpha] = \overline{A}[\alpha] / \overline{B}[\alpha], \tag{2.22}$$

when $\overline{C} = \overline{A}/\overline{B}$, provided that zero does not belong to $\overline{B}[\alpha]$ for all α. This method is equivalent to the extension principle method of fuzzy arithmetic [19]. Obviously, this procedure, of α-cuts plus interval arithmetic, is more user (and computer) friendly.

Example 2.3.3.1

Let $\overline{A} = (-3/-2/-1)$ and $\overline{B} = (4/5/6)$. We determine $\overline{A} \cdot \overline{B}$ using α-cuts and interval arithmetic. We compute $\overline{A}[\alpha] = [-3+\alpha, -1-\alpha]$ and $\overline{B}[\alpha] = [4+\alpha, 6-\alpha]$. So, if $\overline{C} = \overline{A} \cdot \overline{B}$ we obtain $\overline{C}[\alpha] = [(\alpha - 3)(6 - \alpha), (-1 - \alpha)(4 + \alpha)]$, $0 \leq \alpha \leq 1$. The graph of $\overline{C}$ is shown in Figure 2.4.

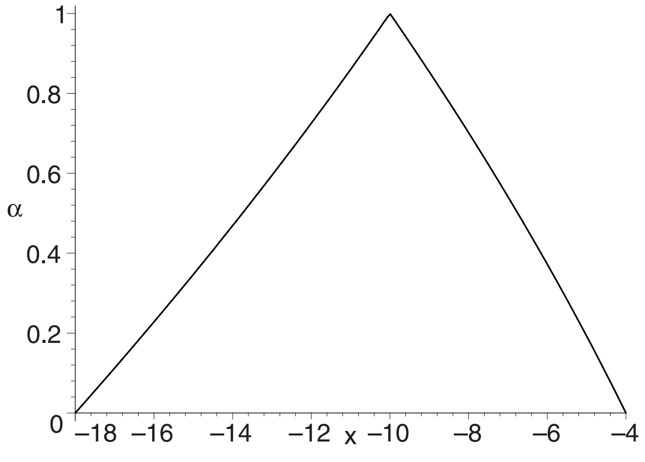


Fig. 2.4. The Fuzzy Number $\overline{C} = \overline{A} \cdot \overline{B}$

2.4 Fuzzy Functions

In this book a fuzzy function is a mapping from fuzzy numbers into fuzzy numbers. We write $H(\overline{X}) = \overline{Z}$ for a fuzzy function with one independent variable $\overline{X}$. Usually $\overline{X}$ will be a triangular (trapezoidal) fuzzy number and then we usually obtain $\overline{Z}$ as a triangular (trapezoidal) shaped fuzzy number. For two independent variables we have $H(\overline{X}, \overline{Y}) = \overline{Z}$.

Where do these fuzzy functions come from? They are usually extensions of real–valued functions. Let $h : [a,b] \rightarrow \mathbb{R}$. This notation means $z = h(x)$ for x in $[a,b]$ and z a real number. One extends $h : [a,b] \rightarrow \mathbb{R}$ to $H(\overline{X}) = \overline{Z}$ in two ways: (1) the extension principle; or (2) using α-cuts and interval arithmetic.

2.4.1 Extension Principle

Any $h : [a,b] \rightarrow \mathbb{R}$ may be extended to $H(\overline{X}) = \overline{Z}$ as follows

$$\overline{Z}(z) = \sup_{x} \left\{ \ \overline{X}(x) \mid h(x) = z, \ a \leq x \leq b \ \right\}. \tag{2.23}$$

Equation (2.23) defines the membership function of $\overline{Z}$ for any triangular (trapezoidal) fuzzy number $\overline{X}$ in $[a,b]$.

If h is continuous, then we have a way to find α-cuts of $\overline{Z}$. Let $\overline{Z}[\alpha] = [z_1(\alpha), z_2(\alpha)]$. Then [8]

$$z_1(\alpha) = \min\{ \ h(x) \mid x \in \overline{X}[\alpha] \ \}, \tag{2.24}$$

$$z_2(\alpha) = \max\{ \ h(x) \mid x \in \overline{X}[\alpha] \ \}, \tag{2.25}$$

for $0 \leq \alpha \leq 1$.

If we have two independent variables, then let $z = h(x,y)$ for x in $[a_1, b_1]$, y in $[a_2, b_2]$. We extend h to $H(\overline{X}, \overline{Y}) = \overline{Z}$ as

$$\overline{Z}(z) = \sup_{x,y} \left\{ \min\left(\overline{X}(x), \overline{Y}(y)\right) \ \mid h(x,y) = z \ \right\}, \tag{2.26}$$

for $\overline{X}$ ($\overline{Y}$) a triangular or trapezoidal fuzzy number in $[a_1, b_1]$ ($[a_2, b_2]$). For α-cuts of $\overline{Z}$, assuming h is continuous, we have

$$z_1(\alpha) = \min\{ \ h(x,y) \mid x \in \overline{X}[\alpha], \ y \in \overline{Y}[\alpha] \ \}, \tag{2.27}$$

$$z_2(\alpha) = \max\{ \ h(x,y) \mid x \in \overline{X}[\alpha], \ y \in \overline{Y}[\alpha] \ \}, \tag{2.28}$$

$0 \leq \alpha \leq 1$.

Applications

Let $f(x_1, ..., x_n; \theta_1, ..., \theta_m)$ be a continuous function. Then

$$I[\alpha] = \{ f(x_1, ..., x_n; \theta_1, ..., \theta_m) \mid \quad \mathbf{S} \quad \}, \tag{2.29}$$

for $\alpha \in [0,1]$ and $\mathbf{S}$ is the statement "$\theta_i \in \overline{\theta}_i[\alpha]$, $1 \leq i \leq m$", for fuzzy numbers $\overline{\theta}_i$, $1 \leq i \leq m$, defines an interval $I[\alpha]$. The endpoints of $I[\alpha]$ may be found as in equations (2.24),(2.25) and (2.27),(2.28). $I[\alpha]$ gives the α-cuts of $f(x_1, ..., x_n; \overline{\theta}_i, ..., \overline{\theta}_m)$.

2.4.2 Alpha-Cuts and Interval Arithmetic

All the functions we usually use in engineering and science have a computer algorithm which, using a finite number of additions, subtractions, multiplications and divisions, can evaluate the function to required accuracy [7]. Such functions can be extended, using α-cuts and interval arithmetic, to fuzzy functions. Let $h : [a, b] \rightarrow \mathbb{R}$ be such a function. Then its extension $H(\overline{X}) = \overline{Z}$, $\overline{X}$ in $[a, b]$ is done, via interval arithmetic, in computing $h(\overline{X}[\alpha]) = \overline{Z}[\alpha]$, α in $[0, 1]$. We input the interval $\overline{X}[\alpha]$, perform the arithmetic operations needed to evaluate h on this interval, and obtain the interval $\overline{Z}[\alpha]$. Then we put these α-cuts together to obtain the value $\overline{Z}$. The extension to more independent variables is straightforward.

For example, consider the fuzzy function

$$\overline{Z} = H(\overline{X}) = \frac{\overline{A}\;\overline{X} + \overline{B}}{\overline{C}\;\overline{X} + \overline{D}}, \tag{2.30}$$

for triangular fuzzy numbers $\overline{A}$, $\overline{B}$, $\overline{C}$, $\overline{D}$ and triangular fuzzy number $\overline{X}$ in $[0, 10]$. We assume that $\overline{C} \geq 0$, $\overline{D} > 0$ so that $\overline{C}\;\overline{X} + \overline{D} > 0$. This would be the extension of

$$h(x_1, x_2, x_3, x_4, x) = \frac{x_1 x + x_2}{x_3 x + x_4}. \tag{2.31}$$

We would substitute the intervals $\overline{A}[\alpha]$ for x_1, $\overline{B}[\alpha]$ for x_2, $\overline{C}[\alpha]$ for x_3, $\overline{D}[\alpha]$ for x_4 and $\overline{X}[\alpha]$ for x, do interval arithmetic, to obtain interval $\overline{Z}[\alpha]$ for $\overline{Z}$. Alternatively, the fuzzy function

$$\overline{Z} = H(\overline{X}) = \frac{2\overline{X} + 10}{3\overline{X} + 4}, \tag{2.32}$$

would be the extension of

$$h(x) = \frac{2x + 10}{3x + 4}. \tag{2.33}$$

2.4.3 Differences

Let $h : [a, b] \rightarrow \mathbb{R}$. Just for this subsection let us write $\overline{Z}^* = H(\overline{X})$ for the extension principle method of extending h to H for $\overline{X}$ in $[a, b]$. We denote $\overline{Z} = H(\overline{X})$ for the α-cut and interval arithmetic extension of h.

We know that $\overline{Z}$ can be different from $\overline{Z}^*$. But for basic fuzzy arithmetic in Section 2.3 the two methods give the same results. In the example below we show that for $h(x) = x(1-x)$, x in $[0, 1]$, we can get $\overline{Z}^* \neq \overline{Z}$ for some $\overline{X}$ in $[0, 1]$. What is known ([8],[21]) is that for usual functions in science and engineering $\overline{Z}^* \leq \overline{Z}$. Otherwise, there is no known necessary and sufficient conditions on h so that $\overline{Z}^* = \overline{Z}$ for all $\overline{X}$ in $[a, b]$. See also [20].

There is nothing wrong in using α-cuts and interval arithmetic to evaluate fuzzy functions. Surely, it is user, and computer friendly. However, we should be aware that whenever we use α-cuts plus interval arithmetic to compute

$\overline{Z} = H(\overline{X})$ we may be getting something larger than that obtained from the extension principle. The same results hold for functions of two or more independent variables.

Example 2.4.3.1

The example is the simple fuzzy expression

$$\overline{Z} = (1 - \overline{X})\ \overline{X}, \tag{2.34}$$

for $\overline{X}$ a triangular fuzzy number in $[0, 1]$. Let $\overline{X}[\alpha] = [x_1(\alpha), x_2(\alpha)]$. Using interval arithmetic we obtain

$$z_1(\alpha) = (1 - x_2(\alpha))x_1(\alpha), \tag{2.35}$$

$$z_2(\alpha) = (1 - x_1(\alpha))x_2(\alpha), \tag{2.36}$$

for $\overline{Z}[\alpha] = [z_1(\alpha), z_2(\alpha)]$, α in $[0, 1]$.

The extension principle extends the regular equation $z = (1-x)x$, $0 \leq x \leq 1$, to fuzzy numbers as follows

$$\overline{Z}^*(z) = \sup_x \left\{\overline{X}(x) | (1-x)x = z,\ 0 \leq x \leq 1\right\}. \tag{2.37}$$

Let $\overline{Z}^*[\alpha] = [z_1^*(\alpha), z_2^*(\alpha)]$. Then

$$z_1^*(\alpha) = \min\{(1-x)x | x \in \overline{X}[\alpha]\}, \tag{2.38}$$

$$z_2^*(\alpha) = \max\{(1-x)x | x \in \overline{X}[\alpha]\}, \tag{2.39}$$

for all $0 \leq \alpha \leq 1$. Now let $\overline{X} = (0/0.25/0.5)$, then $x_1(\alpha) = 0.25\alpha$ and $x_2(\alpha) = 0.50 - 0.25\alpha$. Equations (2.35) and (2.36) give $\overline{Z}[0.50] = [5/64, 21/64]$ but equations (2.38) and (2.39) produce $\overline{Z}^*[0.50] = [7/64, 15/64]$. Therefore, $\overline{Z}^* \neq \overline{Z}$. We do know that if each fuzzy number appears only once in the fuzzy expression, the two methods produce the same results ([8],[21]). However, if a fuzzy number is used more than once, as in equation (2.34), the two procedures can give different results.

2.5 Min/Max of a Fuzzy Number

In some fuzzy optimization problems we will want to determine the values of some decision variables $y = (x_1, ..., x_n)$ that will minimize (or maximize) a fuzzy function $\overline{E}(y)$. For each value of y we obtain a fuzzy number $\overline{E}(y)$. We have employed the method described below in previous publications and we will not use it in this book. We have included it so that the reader may understand our previous solution method when we compare it to our new fuzzy Monte Carlo procedure.

We can not minimize a fuzzy number so what we are going to do, which we have done before ([6],[9]-[13]), is first change $min\overline{E}(y)$ into a multiobjective problem

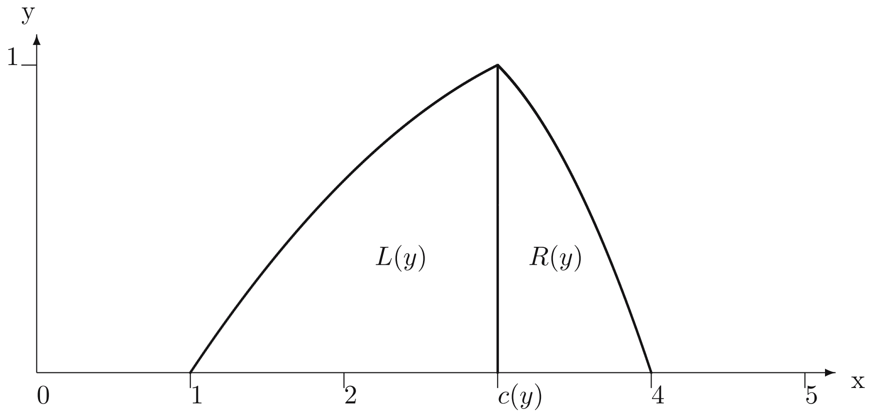


Fig. 2.5. Computations for the Minimum of a Fuzzy Number

and then translate the multiobjective problem into a single objective problem. This strategy is adopted from the finance literature where they had the problem of minimizing a random variable X whose values are constrained by a probability density function $g(x)$. They considered the multiobjective problem: (1) minimize the expected value of X; (2) minimize the variance of X; and (3) minimize the skewness of X to the right of the expected value. For our problem let: (1) $c(y)$ be the center of the core of $\overline{E}(y)$, the core of a fuzzy number is the interval where the membership function equals one, for each y; (2) $L(y)$ be the area under the graph of the membership function to the left of $c(y)$; and (3) $R(y)$ be the area under the graph of the membership function to the right of $c(y)$. See Figure 2.5. For $min\overline{E}(y)$ we substitute: (1) $min[c(y)]$; (2) $maxL(y)$, or maximize the possibility of obtaining values less than $c(y)$; and (3) $minR(y)$, or minimize the possibility of obtaining values greater then $c(y)$. So for $min\overline{E}(y)$ we have

$$V = (maxL(y), min[c(y)], minR(y)). \tag{2.40}$$

First let M be a sufficiently large positive number so that $maxL(y)$ is equivalent to $minL^*(y)$ where $L^*(y) = M - L(y)$. The multiobjective problem become

$$minV' = (minL^*(y), min[c(y)], minR(y)). \tag{2.41}$$

In a multiobjective optimization problem a solution is a value of the decision variable y that produces an undominated vector V'. Let $\mathcal{V}$ be the set of all vectors V' obtained for all possible values of the decision variable y. Vector $v_a = (v_{a1}, v_{a2}, v_{a3})$ dominates vector $v_b = (v_{b1}, v_{b2}, v_{b3})$, both in $\mathcal{V}$, if $v_{ai} \leq v_{bi}$, $1 \leq i \leq 3$, with one of the $\leq$ a strict inequality $<$. A vector $v \in \mathcal{V}$ is undominated if no $w \in \mathcal{V}$ dominates v. The set of undominated vectors in $\mathcal{V}$ is considered the general solution and the problem is to find values of the decision variables that produce undominated V'. The above definition of undominated was for a *min* problem, obvious changes need to be made for a *max* problem.

One way to explore the undominated set is to change the multiobjective problem into a single objective. The single objective problem is

$$min(\lambda_1[M - L(y)] + \lambda_2 c(y) + \lambda_3 R(y)), \tag{2.42}$$

where $\lambda_i > 0$, $1 \leq i \leq 3$, $\lambda_1 + \lambda_2 + \lambda_3 = 1$. You will get different undominated solutions by choosing different values of $\lambda_i > 0$, $\lambda_1 + \lambda_2 + \lambda_3 = 1$. It is known that solutions to this problem are undominated, but for some problems it will be unable to generate all undominated solutions [17]. The decision maker is to choose the values of the weights λ_i for the three minimization goals. Usually one picks different values for the λ_i to explore the solution set and then lets the decision maker choose an optimal y^* from this set of solutions.

2.6 Ordering Fuzzy Numbers

Given a finite set of fuzzy numbers $\overline{A}_1, ..., \overline{A}_n$ we would like to order them from smallest to largest. For a finite set of real numbers there is no problem in ordering them from smallest to largest. However, in the fuzzy case there is no universally accepted way to do this. There are probably more than 40 methods proposed in the literature of defining $\overline{M} \leq \overline{N}$, for two fuzzy numbers $\overline{M}$ and $\overline{N}$. Here the symbol $\leq$ means "less than or equal" and not "a fuzzy subset of". A few key references on this topic are ([1],[14]-[16],[18],[23],[24]) where the interested reader can look up many of these methods and see their comparisons.

In this section we will present three methods of defining $\overline{M} < \overline{N}$, $\overline{M} \approx \overline{N}$ and $\overline{M} \leq \overline{N}$ for two fuzzy numbers $\overline{M}$ and $\overline{N}$ which we will be using in this book.

2.6.1 Buckley's Method

For this book we have named this procedure Buckley's Method because we have used it before ([2],[3]). But note that different definitions of $\leq$ between fuzzy numbers can give different orderings. We first define $<$ between two fuzzy numbers $\overline{M}$ and $\overline{N}$. Define

$$v(\overline{M} \leq \overline{N}) = max\{min(\overline{M}(x), \overline{N}(y)) | x \leq y\}, \tag{2.43}$$

which measures how much $\overline{M}$ is less than or equal to $\overline{N}$. We write $\overline{N} < \overline{M}$ if $v(\overline{N} \leq \overline{M}) = 1$ but $v(\overline{M} \leq \overline{N}) < \eta$, where η is some fixed fraction in $(0, 1]$. In this book we will usually use $\eta = 0.8$ or $\eta = 0.9$. Then $\overline{N} < \overline{M}$ if $v(\overline{N} \leq \overline{M}) = 1$ and $v(\overline{M} \leq \overline{N}) < 0.8$. We then define $\overline{M} \approx \overline{N}$ when both $\overline{N} < \overline{M}$ and $\overline{M} < \overline{N}$ are false. $\overline{M} \leq \overline{N}$ means $\overline{M} < \overline{N}$ or $\overline{M} \approx \overline{N}$. Now this $\approx$ may not be transitive. If $\overline{N} \approx \overline{M}$ and $\overline{M} \approx \overline{O}$ implies that $\overline{N} \approx \overline{O}$, then $\approx$ is transitive. However, it can happen that $\overline{N} \approx \overline{M}$ and $\overline{M} \approx \overline{O}$ but $\overline{N} < \overline{O}$ because $\overline{M}$ lies a little to the right of $\overline{N}$ and $\overline{O}$ lies a little to the right of $\overline{M}$ but $\overline{O}$ lies sufficiently far to the right of $\overline{N}$ that we obtain $\overline{N} < \overline{O}$.

But this ordering is still useful in partitioning the set of fuzzy numbers $\overline{A}_i$, $1 \leq i \leq n$, up into disjoint sets $H_1, ..., H_K$ where ([2],[3]): (1) given any $\overline{A}_i$ and

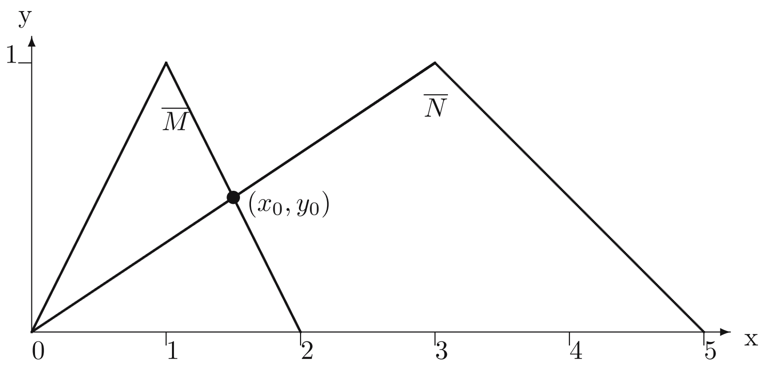


Fig. 2.6. Determining $v(\overline{N} \leq \overline{M})$

$\overline{A}_j$ in H_k, $1 \leq k \leq K$, then $\overline{A}_i \approx \overline{A}_j$; and (2) given $\overline{A}_i \in H_i$ and $i < j$, there is a $\overline{A}_j \in H_j$ with $\overline{A}_i < \overline{A}_j$. We say a fuzzy number $\overline{A}_i$ is dominated if there is another fuzzy number $\overline{A}_j$ so that $\overline{A}_i < \overline{A}_j$. So H_K will be all the undominated $\overline{A}_i$. Now H_K is nonempty and if it does not contain all the fuzzy numbers we then define H_{K-1} to be all the undominated fuzzy numbers after we delete all those in H_K. We continue this way to the last set H_1. Then the highest ranked fuzzy numbers lie in H_K, the second highest ranked fuzzy numbers are in H_{K-1}, etc. This result is easily seen if you graph all the fuzzy numbers on the same axis then those in H_K will be clustered together farthest to the right, proceeding from the H_K cluster to the left the next cluster will be those in H_{K-1}, etc.

There is an easy way to determine if $\overline{M} < \overline{N}$, or $\overline{M} \approx \overline{N}$, for many fuzzy numbers. This will be all we need in randomness tests and Monte Carlo studies. First, it is easy to see that if the core of $\overline{N}$ lies completely to the right of the core of $\overline{M}$, then $v(\overline{M} \leq \overline{N}) = 1$. Also, if the core of $\overline{M}$ and the core of $\overline{N}$ overlap, then $\overline{M} \approx \overline{N}$. Now assume that the core of $\overline{N}$ lies to the right of the core of $\overline{M}$, as shown in Figure 2.6 for triangular fuzzy numbers, and we wish to compute $v(\overline{N} \leq \overline{M})$. The value of this expression is simply y_0 in Figure 2.6. In general, for triangular (shaped), and trapezoidal (shaped), fuzzy numbers $v(\overline{N} \leq \overline{M})$ is the height of their intersection when the core of $\overline{N}$ lies to the right of the core of $\overline{M}$.

2.6.2 Kerre's Method

We first need to present the fuzzy max (written $\overline{\max}$) of two fuzzy numbers. If $\overline{O} = \overline{\max}(\overline{M}, \overline{N})$, then

$$\overline{O}(z) = \sup\left\{\min(\overline{M}(x), \overline{N}(y)) | \max(x, y) = z\right\}. \tag{2.44}$$

The authors in [19] give a detailed study of the properties of $\overline{\max}$ and $\overline{\min}$ (fuzzy min).

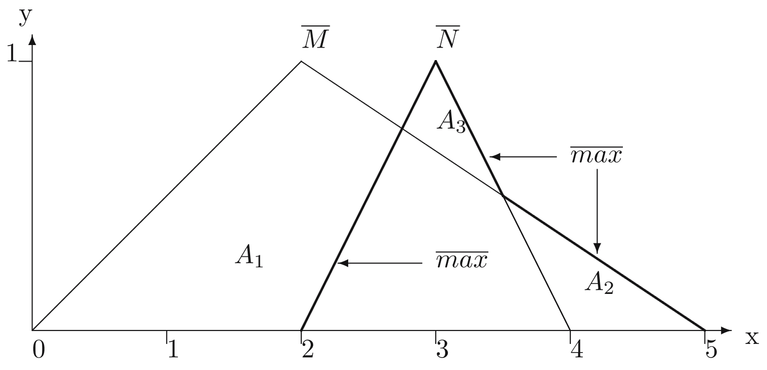


Fig. 2.7. Fuzzy Max

Next we define the Hamming distance between $\overline{M}$ and $\overline{N}$. The Hamming distance, $d(\overline{M}, \overline{N})$, is defined as

$$d(\overline{M}, \overline{N}) = \int_{-\infty}^{\infty} |\overline{M}(x) - \overline{N}(x)| dx. \tag{2.45}$$

Clearly, d is a metric (distance measure) on the space of continuous fuzzy numbers (those whose membership function is continuous).

Then we say $\overline{M} < \overline{N}$ is true whenever

$$d(\overline{N}, \overline{\max}(\overline{M}, \overline{N})) < d(\overline{M}, \overline{\max}(\overline{M}, \overline{N})). \tag{2.46}$$

This is simply a fuzzification of $x < y$ if and only if $\max(x, y) = y$ for real $x \neq y$. We write $\overline{M} \approx \overline{N}$ if you get equality in equation (2.46) and $\overline{M} \leq \overline{N}$ means $\overline{M} < \overline{N}$ or $\overline{M} \approx \overline{N}$. A numerical example showing $\overline{M} \leq \overline{N}$ by this method is in ([19], p. 407 - 408). We call this procedure for evaluating fuzzy inequalities Kerre's method [15].

Figure 2.7 shows the fuzzy max of two fuzzy numbers. We see that d $(\overline{M}, \overline{\max}(\overline{M}, \overline{N}))$ is the area of regions A_1 plus A_3 and $d(\overline{N}, \overline{\max}(\overline{M}, \overline{N}))$ is the area of region A_2. It appears that the area of region A_2 is less than the area of regions A_1 plus A_3 so $\overline{M} < \overline{N}$.

We point out from [23] that Kerre's $\leq$ is transitive.

2.6.3 Chen's Method

A third method of ranking fuzzy numbers we focus on was presented by Chen in [15]. A score is computed for each fuzzy number which is needed for ranking. The fuzzy set with the highest score is the largest fuzzy number. In order to rank triangular shaped fuzzy numbers $\overline{N} \approx (n_1/n_2/n_3)$ and $\overline{M} \approx (m_1/m_2/m_3)$ Chen defined a fuzzy max and a fuzzy min where the supports of fuzzy max and min is $[x_{\min}, x_{\max}]$ where

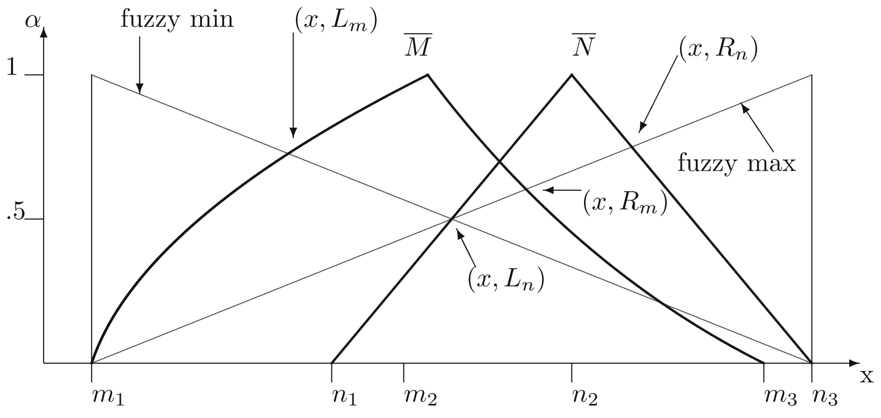


Fig. 2.8. Ranking Fuzzy Numbers Based on Chen's Method

$$x_{\min} = \min(n_1, m_1), \tag{2.47}$$

$$x_{\max} = \max(n_3, m_3). \tag{2.48}$$

Fuzzy min and fuzzy max are triangular fuzzy numbers with membership degree one at the left and the right limit of the support, respectively (see Figure 2.8). The membership functions are

$$\mu_{\min}(x) = \begin{cases} \frac{x - x_{\max}}{x_{\min} - x_{\max}} & : \; x_{\min} \leq x \leq x_{\max}, \\ 0 & : \; \text{otherwise} \end{cases} \tag{2.49}$$

$$\mu_{\max}(x) = \begin{cases} \frac{x - x_{\min}}{x_{\max} - x_{\min}} & : \; x_{\min} \leq x \leq x_{\max}, \\ 0 & : \; \text{otherwise.} \end{cases} \tag{2.50}$$

The intersection points between fuzzy max and $\overline{M}$ and $\overline{N}$ as well as the intersection points between fuzzy min and $\overline{M}$ and $\overline{N}$ are needed for computing the final scores. We compute

$$\mu_R(\overline{M}) = \sup_x (min(\mu_{\max}(x), \overline{M}(x))), \tag{2.51}$$

and

$$\mu_L(\overline{M}) = \sup_x (min(\mu_{\min}(x), \overline{M}(x))), \tag{2.52}$$

where $\mu_R(\overline{M})$ indicates the max of the intersection point between fuzzy max and $\overline{M}$ and $\mu_L(\overline{M})$ stands for the left score which is given by the max intersection point with fuzzy min. The larger $\mu_R(\overline{M})$ is, the higher $\overline{M}$ should be ranked. On the other hand a high value of $\mu_L(\overline{M})$ and $\overline{M}$ is close to the fuzzy min, and therefore should be ranked lower. By combining both scores we get the final rating

$$\mu_T(\overline{M}) = \frac{1}{2} \left(\mu_R(\overline{M}) + (1 - \mu_L(\overline{M}) \right). \tag{2.53}$$

Similarly, we get $\mu_T(\overline{N})$. We then say that $\overline{M} < \overline{N}$ is true if $\mu_T(\overline{M}) < \mu_T(\overline{N})$. In Figure 2.8 we used the notation $L_m = \mu_L(\overline{M})$, $R_m = \mu_R(\overline{M})$, $L_n = \mu_L(\overline{N})$ and $R_n = \mu_R(\overline{N})$. The labeling of L_m, R_m, L_n and R_n in Figure 2.8 may be a little misleading. These numbers are the y coordinates of the point indicated in the figure.

We write $\overline{M} \approx \overline{N}$ when $\mu_T(\overline{M}) = \mu_T(\overline{N})$ and as usual $\overline{M} \leq \overline{N}$ means $\overline{M} < \overline{N}$ or $\overline{M} \approx \overline{N}$.

We point out from [23] that Chen's $\leq$ is transitive.

2.6.4 Breaking Ties

We first adopt some method of deciding on $\leq$, $<$ and $\approx$ between fuzzy numbers. Assume we will use Buckley's Method. Sometimes in a fuzzy optimization problem, assume a max problem, we may get too many ties for maximum. Suppose we wish to $max \overline{Z} = f(\overline{X}_1, ..., \overline{X}_n)$ where the $\overline{X}_i$ are triangular fuzzy numbers and $\overline{Z}$ is a triangular shaped fuzzy number. Using our fuzzy Monte Carlo method we will generate a sequence $\overline{Z}_j$, $j = 1, 2, 3, ...$ Let H_K be the highest ranked fuzzy numbers in the sequence (Section 2.6.1). But H_K could contain 10, or 20, or 100 fuzzy numbers. Given $\overline{Z}_a$ and $\overline{Z}_b$ in H_K we know that $\overline{Z}_a \approx \overline{Z}_b$. What we can now do is rank the fuzzy numbers in H_K by their vertices. Let $\overline{Z}_a \approx (z_{a1}/z_{a2}/z_{a3})$ and $\overline{Z}_b \approx (z_{b1}/z_{b2}/z_{b3})$. We say $\overline{Z}_a < \overline{Z}_b$ if $z_{a2} < z_{b2}$, $\overline{Z}_a > \overline{Z}_b$ if $z_{a2} > z_{b2}$, and $\overline{Z}_a \approx \overline{Z}_b$ if $z_{a2} = z_{b2}$. The resulting highest ranked fuzzy numbers H_K^* should be more manageable. If we require a unique solution and we still have "ties" then we use the left (right) end points of the support. For example, if $\overline{Z}_a \approx \overline{Z}_b$ and: (1) $z_{b2} = z_{a2}$ but $z_{b1} < z_{a1}$ we say $\overline{Z}_b < \overline{Z}_a$; (2) $z_{b2} = z_{a2}$, $z_{b1} = z_{a1}$ and $z_{b3} < z_{a3}$ we say $\overline{Z}_b < \overline{Z}_a$; (3) $z_{b2} = z_{a2}$, $z_{b1} = z_{a1}$, $z_{b3} = z_{a3}$ we randomly discard one of them and declare the other the max (or min).

2.7 Undominated Fuzzy Vectors

We will first review the concept of undominated for crisp vectors. Consider a multiobjective optimization problem

$$max\ v = (v_1 = f_1(x), ..., v_m = f_m(x)), \tag{2.54}$$

where $x = (x_1, ..., x_n)$ is in the feasible set $\mathcal{F}$. Usually the x_i are non-negative. The optimization problem has constraints on the variables x_i and $\mathcal{F}$ is all x which satisfy these constraints. There will be certain changes for a min problem.

Let $\mathcal{V}$ be all vectors v from equation (2.54) obtained using all the $x \in \mathcal{F}$. Given $v_a = (v_{a1}, ..., v_{am})$ and $v_b = (v_{b1}, ..., v_{bm})$ in $\mathcal{V}$ we say v_a dominates v_b if $v_{ai} \geq v_{bi}$ all i with at least one of the $\geq$ is equal to $>$. The solution set $\mathcal{S}$ to the multiobjective max problem is all undominated $v \in \mathcal{V}$. The decision maker(s), depending on their preferences, would now choose certain $v \in \mathcal{S}$ as solutions to the optimization problem.

A way to generate undominated solutions is to consider the single objective optimization problem

$$max(\ \lambda_1 f_1(x) + ... + \lambda_m f_m(x)\ ), \tag{2.55}$$

for the $\lambda_i \in (0,1)$ all i and $\lambda_1 + ... + \lambda_m = 1$. It is known that all solutions are undominated but in certain problems we may not be able to obtain all undominated solutions by varying the values of the λ_i [17].

Now, as in Chapter 9, we consider a fuzzy multiobjective optimization problem

$$max\ \overline{V} = (\overline{V}_1 = f_1(\overline{X}), ..., \overline{V}_m = f_m(\overline{X})), \tag{2.56}$$

where $\overline{X} = (\overline{X}_1, ..., \overline{X}_n)$ is in the feasible set $\overline{\mathcal{F}}$. Assume we are using one of the three methods discussed above for evaluating $\leq$, $<$ and $\approx$ between fuzzy numbers. Usually the $\overline{X}_i \geq 0$ all i.

Let $\overline{\mathcal{V}}$ be all vectors $\overline{V}$ from equation (2.56) obtained using all the $\overline{X} \in \overline{\mathcal{F}}$. Given $\overline{V}_a = (\overline{V}_{a1}, ..., \overline{V}_{am})$ and $\overline{V}_b = (\overline{V}_{b1}, ..., \overline{V}_{bm})$ in $\overline{\mathcal{V}}$ we say $\overline{V}_a$ weakly dominates $\overline{V}_b$ if $\overline{V}_{ai} \geq \overline{V}_{bi}$ all i with at least one of the $\geq$ equal to $>$. We will call this definition of dominance "weak dominance". We will say $\overline{V}_a$ strongly dominates $\overline{V}_b$ if $\overline{V}_{ai} > \overline{V}_{bi}$ all i. We will employ both definitions of dominance. The solution set $\overline{\mathcal{S}}$ to the fuzzy multiobjective max problem is all (weakly, strongly) undominated $\overline{V} \in \overline{\mathcal{V}}$. Of course, we would like to show that this undominated set is nonempty.

Next we change the fuzzy multiobjective optimization problem into a single objective

$$max(\ \lambda_1 \overline{V}_1 + ... + \lambda_m \overline{V}_m\ ), \tag{2.57}$$

for $\lambda_i > 0$ all i and $\lambda_1 + ... + \lambda_m = 1$. We would now like to argue that any solution to equation (2.57) is (weakly, strongly) undominated. The argument depends on what definition for $\leq$, $<$ and $\approx$ you are using between fuzzy numbers. All that is needed is that if $\overline{V}_a$ (weakly, strongly) dominates $\overline{V}_b$ then

$$\sum_{i=1}^{m} \lambda_i \overline{V}_{ai} > \sum_{i=1}^{m} \lambda_i \overline{V}_{bi}. \tag{2.58}$$

If this is true one can easily obtain the desired result. Let us now prove this result for the special case of $m = 2$ and then we consider this problem for the three methods of defining $<$, $\leq$ and $\approx$ between fuzzy numbers discussed above.

We still consider the max problem with obvious changes for the min problem. Suppose the solution to

$$max(\ \lambda_1 \overline{X}_1 + \lambda_2 \overline{X}_2\ ) \tag{2.59}$$

is $\overline{X}_1^*$ and $\overline{X}_2^*$ for given (and fixed) $0 < \lambda_i < 1$, $i = 1, 2$ and $\lambda_1 + \lambda_2 = 1$. Assume $\overline{V}^* = (\overline{X}_1^*, \overline{X}_2^*)$ is not (weakly, strongly) undominated but $\overline{W} \in \overline{\mathcal{F}}$ (weakly, strongly) dominates $\overline{V}^*$. We now consider the two cases of weak and strong domination.

First we assume weak domination. So $\overline{X}_1^* < \overline{W}_1$ and $\overline{X}_2^* \leq \overline{W}_2$. Assume we are using a method of defining $\leq$, $<$ and $\approx$ between fuzzy numbers so that the following two results are true.

$$\lambda_1 \overline{X}_1^* < \lambda_1 \overline{W}_1, \ \lambda_2 \overline{X}_2^* \leq \lambda_2 \overline{W}_2, \tag{2.60}$$

$$\lambda_1 \overline{X}_1^* + \lambda_2 \overline{X}_2^* < \lambda_1 \overline{W}_1 + \lambda_2 \overline{W}_2. \tag{2.61}$$

Then $\overline{V}^*$ is not the optimal solution. A contradiction. So if $\leq$, $<$ and $\approx$ has the properties in equations (2.60) and (2.61) we get that the optimization problem in equation (2.59) only produces weakly undominated solutions.

Next we look at strong domination. So $\overline{X}_1^* < \overline{W}_1$ and $\overline{X}_2^* < \overline{W}_2$. Assume we are using a method of defining $\leq$, $<$ and $\approx$ between fuzzy numbers so that the following two results are true.

$$\lambda_1 \overline{X}_1^* < \lambda_1 \overline{W}_1, \ \lambda_2 \overline{X}_2^* < \lambda_2 \overline{W}_2, \tag{2.62}$$

$$\lambda_1 \overline{X}_1^* + \lambda_2 \overline{X}_2^* < \lambda_1 \overline{W}_1 + \lambda_2 \overline{W}_2. \tag{2.63}$$

Then $\overline{V}^*$ is not the optimal solution. A contradiction. So if $\leq$, $<$ and $\approx$ has the properties in equations (2.62) and (2.63) we get that the optimization problem in equation (2.59) only produces strongly undominated solutions.

2.7.1 Buckley's Method

We will use strong domination and show equation (2.63) is true for Buckley's Method. Let $\overline{X}_i^* \approx (x_{i1}/x_{i2}/x_{i3})$ and $\overline{W}_i \approx (w_{i1}/w_{i2}/w_{i3})$ for $i = 1, 2$. Assume that $x_{12} < w_{12}$ and $x_{22} < w_{22}$ as in Figure 2.6 and we are using $\eta = 0.8$. Then $v(\overline{X}_i^*, \overline{W}_i) = \eta_i < 0.8$ and $v(\overline{W}_i, \overline{X}_i^*) = 1$ for $i = 1, 2$. Assume that $\eta_1 \leq \eta_2$. Now let $\overline{X}_i^*[\eta_2] = [x_{i1}(\eta_2), x_{i2}(\eta_2)]$ and $\overline{W}_i[\eta_2] = [w_{i1}(\eta_2), w_{i2}(\eta_2)]$ for $i = 1, 2$.

We know that $x_{22}(\eta_2) = w_{21}(\eta_2)$ and $x_{12}(\eta_2) \leq w_{11}(\eta_2)$. Now let $\overline{X}^* = \lambda_1 \overline{X}_1^* + \lambda_2 \overline{X}_2^*$ and $\overline{W} = \lambda_1 \overline{W}_1 + \lambda_2 \overline{W}_2$. We see that

$$\overline{X}^*[\eta_2] = [\lambda_1 x_{11}(\eta_2) + \lambda_2 x_{21}(\eta_2), \lambda_1 x_{12}(\eta_2) + \lambda_2 x_{22}(\eta_2)], \tag{2.64}$$

and

$$\overline{W}[\eta_2] = [\lambda_1 w_{11}(\eta_2) + \lambda_2 w_{21}(\eta_2), \lambda_1 w_{12}(\eta_2) + \lambda_2 w_{22}(\eta_2)]. \tag{2.65}$$

Therefore $\overline{X}^* < \overline{W}$ since

$$\lambda_1 x_{12}(\eta_2) + \lambda_2 x_{22}(\eta_2) < \lambda_1 w_{11}(\eta_2) + \lambda_2 w_{21}(\eta_2). \tag{2.66}$$

Now we can explain why we did now use weak domination. Because we can have $\overline{X}_1^* \approx \overline{W}_1$, so $\overline{X}_1^* \leq \overline{W}_1$ is true, and $\overline{X}_2^* < \overline{W}_2$ but for certain values of the λ_i we get $\overline{X}^* \approx \overline{W}$ so $\overline{X}^* < \overline{W}$ is not true.

2.7.2 Kerre's Method

First assume that we are using weak dominance. It was shown in [23] that if $\overline{X}_1^* < \overline{W}_1$ and $\overline{X}_2^* \leq \overline{W}_2$, then $\overline{X}_1^* + \overline{X}_2^* < \overline{W}_1 + \overline{W}_2$ may not be true. For this reason we will not use Kerre's Method in fuzzy multiobjective optimization problems in Chapter 9.

2.7.3 Chen's Method

First assume that we are using weak dominance. It was shown in [23] that if $\overline{X}_1^* < \overline{W}_1$ and $\overline{X}_2^* \leq \overline{W}_2$, then $\overline{X}_1^* + \overline{X}_2^* < \overline{W}_1 + \overline{W}_2$ may not be true. For this reason we will not use Chen's Method in fuzzy multiobjective optimization problems in Chapter 9.

References

1. Bortolon, G., Degani, R.: A Review of Some Methods for Ranking Fuzzy Subsets. Fuzzy Sets and Systems 15, 1–19 (1985)
2. Buckley, J.J.: Ranking Alternatives Using Fuzzy Numbers. Fuzzy Sets and Systems 15, 21–31 (1985)
3. Buckley, J.J.: Fuzzy Hierarchical Analysis. Fuzzy Sets and Systems 17, 233–247 (1985)
4. Buckley, J.J., Eslami, E.: Introduction to Fuzzy Logic and Fuzzy Sets. Physica-Verlag, Heidelberg (2002)
5. Buckley, J.J., Feuring, T.: Fuzzy and Neural: Interactions and Applications. Physica-Verlag, Heidelberg (1999)
6. Buckley, J.J., Feuring, T.: Evolutionary Algorithm Solutions to Fuzzy Problems: Fuzzy Linear Programming. Fuzzy Sets and Systems 109, 35–53 (2000)
7. Buckley, J.J., Hayashi, Y.: Can Neural Nets be Universal Approximators for Fuzzy Functions? Fuzzy Sets and Systems 101, 323–330 (1999)
8. Buckley, J.J., Qu, Y.: On Using α-cuts to Evaluate Fuzzy Equations. Fuzzy Sets and Systems 38, 309–312 (1990)
9. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
10. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems in Operations Research. J. Advanced Computational Intelligence 3, 171–176 (1999)
11. Buckley, J.J., Feuring, T., Hayashi, Y.: Multi-Objective Fully Fuzzified Linear Programming. Int. J. Uncertainty, Fuzziness and Knowledge Based Systems 9, 605–622 (2001)
12. Buckley, J.J., Feuring, T., Hayashi, Y.: Fuzzy Queuing Theory Revisited. Int. J. Uncertainty, Fuzziness and Knowledge Based Systems 9, 527–538 (2001)
13. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems in Operations Research: Inventory Control. Soft Computing 7, 121–129 (2002)
14. Chang, P.T., Lee, E.S.: Fuzzy Arithmetic and Comparison of Fuzzy Numbers. In: Delgado, M., Kacprzyk, J., Verdegay, J.L., Vila, M.A. (eds.) Fuzzy Optimization: Recent Advances, pp. 69–81. Physica-Verlag, Heidelberg (1994)
15. Chen, S.J., Hwang, C.L.: Fuzzy Multiple Attribute Decision Making. Springer, Heidelberg (1992)

16. Dubois, D., Kerre, E., Mesiar, R., Prade, H.: Fuzzy Interval Analysis. In: Dubois, D., Prade, H. (eds.) Fundamentals of Fuzzy Sets, The Handbook of Fuzzy Sets, pp. 483–581. Kluwer Acad. Publ., Dordrecht (2000)
17. Geoffrion, A.M.: Proper Efficiency and the Theory of Vector Maximization. J. Math. Analysis and Appl. 22, 618–630 (1968)
18. Gonzalez, A., Vila, M.A.: Dominance Relations on Fuzzy Numbers. Information Sciences 64, 1–16 (1992)
19. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall, Upper Saddle River, N.J. (1995)
20. Kreinovich, V., Longpre, L., Buckley, J.J.: Are There Easy-to-Check Necessary and Sufficient Conditions for Straightforward Interval Computations to be Exact? Reliable Computing 9, 349–358 (2003)
21. Moore, R.E.: Methods and Applications of Interval Analysis, SIAM Studies in Applied Mathematics, Philadelphia (1979)
22. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge, U.K. (1990)
23. Wang, X., Kerre, E.E.: Reasonable Properties for the Ordering of Fuzzy Quantities (I). Fuzzy Sets and Systems 118, 375–385 (2001)
24. Wang, X., Kerre, E.E.: Reasonable Properties for the Ordering of Fuzzy Quantities (II). Fuzzy Sets and Systems 118, 387–405 (2001)

3 Crisp Random Numbers and Vectors

3.1 Introduction

In this chapter we first discuss different ways to generate sequences of "random" numbers in some interval $[a, b]$. Usually the random numbers are first produced in $[0, 1]$ and then we perform a linear transformation to get them into $[a, b]$. Next we consider making sequences of random non-negative integers. We wish to produce sequences of random vectors $v = (x_1, ..., x_n)$ where the x_i are real numbers, and the randomness here means that the v will uniformly fill the space $[a, b]^n$. These random vectors will be used in the next chapter to generate sequences of random fuzzy numbers.

Subsequently, vectors of so-generated random fuzzy numbers are used for streams to feed fuzzy Monte Carlo optimization. As is shown in Chapter 4, with a 5-tuple we can generate a fuzzy number with quadratic membership functions. In some cases we evaluate using a vector of two or three fuzzy numbers generated from 5-tuples. In Chapters 6 and 9, we generate pairs of fuzzy numbers from Sobol quasi-random 10-tuples. In Chapters 7 and 8, vectors of three fuzzy numbers generated from Sobol 15-tuples are used. Other applications are in Chapters 10-16.

3.2 Random Numbers

We could have chosen to generate fuzzy numbers whenever they are needed; however, we wish to study the crisp numbers from which they are made, and we wish to study fuzzy numbers generated in various ways from those streams of crisp numbers. We expand upon a computer program from [2] to create streams of crisp numbers for which we simultaneously evaluate randomness. Our application from [2], `RNGenerator`, has several new features noted below. `RNGenerator` may be linked with any of several random number (RN) generator subroutines. The ones which we used were:

1. True Random: A million 8-bit (in binary notation) true random numbers were downloaded from `http://www.random.org`. This routine supplies one

16-bit true random integer (concatenate two 8-bit bytes), sequentially from that list, with each call.

2. Pseudo-Random: This routine supplies one 16-bit pseudo-random integer. From the C library's `rand()` function (Visual C++) with each call, we first obtain a pseudo-random integer in the interval [0,32767]. We multiply that value times 2 to create an even integer in [0,65534]. Since we will be scaling further to [0,1) for a χ^2 test, having even pseudo-random integers is not a concern.
3. Quasi-Random: Several quasi-random number routines from Burkardt [12] were used as the bases for quasi-random integer generators (Section 7.7, "Quasi-Random Sequences," from [10] provides background to Sobol sequences).
   The routines are designed to create n-tuples of crisp 16-bit integers, where n is user specified. To make their use compatible with the other random number generators, our generators release integers one at a time with each call. We are particularly interested in Sobol quasi-random integers because of our prior work ([1],[2]), and because Sobol sequences are reasonably well known and we have used them with MATLAB [9].

3.2.1 Quasi-random Sequences

Quasi-random numbers are also known as Low Discrepancy Points (LDP) or low discrepancy sequences. They are called quasi-random because they possess many attributes of random numbers, but they are truly not random. Rather they are designed to be less random and more uniformly distributed than Linear Congruential Generated (LCG) pseudo-random numbers. Their other name, Low Discrepancy Points, may be more appropriate though less catchy. The following excerpt from [7] is instructive to the goal of LDPs:

> [Begin] with a unit hypercube that is, a cube of more than three dimensions. Each edge of the cube has a length of 1 unit, so its volume is 1. Lets assume a large number of points are to be distributed within the cube. How can these points be distributed in such a way that, if any volume in the cube is selected, the proportion of the points within the volume is as close to the volume itself? . . . Points that provide, on average, a close fit between the volume and proportion numbers provide a *low discrepancy* thus, their name.

Many quasi-random number algorithms have been designed. The Van der Corput Sequence (1935) [11] generates LDPs in just one dimension [6]. Others have since been designed to provide LDPs in higher dimensions. Some of the best known are Halton (1960), Hammersley (1960), Sobol (1967), Faure (1980), and Niederreiter (1987) [11] Because quasi-random numbers provide more uniform coverage to a space than pseudo-random numbers, they are "at the forefront of financial mathematics" [8]. Algorithms for them are available from various sources including the Association for Computing Machinery (ACM).

Table 3.1. Random Number Generator χ^2 Tests, N=500,000, bins=10, 9 df

type	tuples	χ^2	Min	Max	Equal Pairs
Pseudo	1	9.793000	0	65534	10
True	1	7.073160	0	65535	6
Faure	2	0.007280	0	65534	829
Halton	2	0.011480	0	65534	10
Neiderreiter	2	0.007520	0	65534	864
Sobol	2	0.007600	0	65534	865
Faure	3	0.020880	0	65534	352
Halton	3	0.017680	0	65534	10
Neiderreiter	3	0.007400	0	65534	529
Sobol	3	0.006840	0	65534	530
Faure	5	0.082480	0	65534	339
Halton	5	0.017280	0	65534	10
Neiderreiter	5	0.003800	0	65534	332
Sobol	5	0.006520	0	65534	349
Faure	6	0.060400	0	65533	0
Halton	6	0.031800	0	65534	14
Neiderreiter	6	0.009360	0	65534	332
Sobol	6	0.014720	0	65534	411
Faure	9	0.700920	0	65531	0
Halton	9	0.057960	0	65533	11
Neiderreiter	9	0.007440	0	65534	356
Sobol	9	0.021720	0	65534	423
Faure	10	0.456000	0	65531	0
Halton	10	0.067400	0	65533	11
Neiderreiter	10	0.007840	0	65534	365
Sobol	10	0.013440	0	65534	411
Faure	15	4.328640	0	65534	0
Halton	15	0.203560	0	65533	13
Neiderreiter	15	0.020840	0	65534	411
Sobol	15	0.039640	0	65533	470

3.2.2 Random Number Generator

`RNGenerator` does statistics on the stream of RNs it generates. The 16-bit integers are scaled to [0,1) by division by 65536. A chi-square test is done for 10 bins (9 degrees of freedom) on 500,000 random numbers generated by each method. Though we do not use many of the streams later in this book, we provide our findings for comparison. In Table 3.1: (1) "type" is the type of generator used for the stream; (2) "tuple" is the number of integers the generator creates at a time; (3) "χ^2" is the value of the chi-square statistic; (4) "Min" is the smallest random number produced; (5) "Max" is the value of the largest random number generated; and (6) "Equal Pairs" means that two consecutively generated random numbers were equal.

The chi-square test was the standard randomness test applied to sequences of real numbers. The null hypothesis is H_0 that the sequence is random and the alternate hypothesis is H_1 that the sequence is not random. The significance level of the test was $\gamma = 0.05$. We place the random numbers into 10 equally spaced bins where, assuming H_0 is true, the expected number in each bin would be $500,000/10$. The critical value is $\chi^2 = 16.9190$ for 9 degrees of freedom. So the true random numbers and the pseudo-random numbers pass the randomness test (do not reject H_0). Moreover, the quasi-random numbers also pass the randomness test (do not reject H_0). The use of the quasi-random numbers will be explained in Section 3.4.

3.3 Random Non-negative Integers

Suppose we want a random sequence $z_1, z_2, z_3, ...$ of non-negative integers in some interval $[a, b]$, $a \geq 0$. We can use a pseudo-random number generator to first get a random sequence $x_1, x_2, x_3, ...$ in $[0, 1)$. We next transform the x_i into the interval $[a - 0.5, b + 0.5]$. Let $y_i = (b - a + 1)x_i + (a - 0.5)$, $i = 1, 2, 3, ...$ If the decimal part of y_i is less than 0.5 round y_i down to z_i and if the decimal part of y_i is greater than 0.5 round y_i up to z_i, $i = 1, 2, 3, ...$ In case the decimal part of y_i equals 0.5 round y_i to the nearest even integer for z_i. Even non-negative integers are $0, 2, 4, ...$ Random sequences of vectors whose components are non-negative integers, is discussed in Section 3.5.

3.4 Random Vectors: Real Numbers

Using quasi-random numbers in $[0, 1]$ we make vectors $v = (x_1, ..., x_n)$ that should uniformly fill the region $[0, 1]^n$. We can then easily adjust these vectors so that they uniformly fill the space $[a, b]^n$. It is well known ([3],[4],[10]) that if a pseudo-random number generator is used to produce sequences of vectors $v \in [a, b]^n$, and we plot their values, then there will be clusters and vacant regions in $[a, b]^n$. Quasi-random number generators are designed to avoid this problem and uniformly fill the space $[a, b]^n$.

Put "quasi-Monte Carlo simulation" into your search engine and get almost 700 web sites to visit. Another search phrase "low discrepancy numbers" could be used. We downloaded a MATLAB program for Sobol quasi-random vectors from [12]. When you run this program with small initial seeds it obviously does not start off "random". You need to discard the first few vectors and in [13] it is recommended that you delete the first 64 vectors. With large initial seeds we did not have this problem. We now generated N quasi-random vectors of length 7 using this MATLAB program, to be used in Chapter 4 to produce a random sequence of quadratic fuzzy numbers, and tested them for randomness.

We divided $[0, 1]$ up into four equal intervals, in each of the seven dimensions in $[0, 1]^7$, and we call these intervals $I_1 = [0.00, 0.25)$, $I_2 = [0.25, 0.5)$, $I_3 = [0.5, 0.75)$ and $I_4 = [0.75, 1.00]$. We then construct $K = 4^7$ boxes

$$B(ijklmnp) = I_i \times I_j \times ... \times I_p, \tag{3.1}$$

in $[0, 1]^7$. Each vector $v = (x_1, ..., x_7)$ will fall into a unique box and for N vectors let $O(ijklmnp)$ be the number of vectors that were in box $B(ijklmnp)$. In this statistical test the null hypothesis is H_0 that the sequence of vectors is random and the alternative hypothesis H_1 is that the sequence is not random. Let the significance level γ of the test be 0.05. This will be a chi-square goodness of fit test.

Under the randomness assumption of the null hypothesis the probability of an x_i in v being in an interval I_q, $q = 1, 2, 3, 4$, is $\frac{1}{4}$, for $i = 1, ..., 7$. So the expected number of vectors in any box is

$$E = \frac{N}{4^7}. \tag{3.2}$$

For the chi-square test we would like E to be at least five so we choose $N = 100,000$.

Let θ be the degrees of freedom of the test and then the critical value for the test will be cv so that the probability of a chi-square random variable χ^2, with degrees of freedom θ, exceeding cv is equal to $\gamma = 0.05$. The chi-square random variable for this test is

$$\chi^2 = \sum_{boxes} \frac{(O(ijklmnp) - E)^2}{E}. \tag{3.3}$$

Next we need to determine the degrees of freedom θ and the critical value cv. Now [5]

$$\theta = (4^7 - 1) - [(7)(3)] = 16362, \tag{3.4}$$

because: (1) we loose one degree of freedom because the sum of the $O(ijklnmp)$ must equal N; and (2) we loose three degrees of freedom for each dimension because we must specify three of the probabilities, since their sum is one, $p_i =$ the probability of x_i being in I_i, $i = 1, 2, 3$. Since the degrees of freedom is so large we must use the approximation [5]

$$cv = 0.5(z + \sqrt{2\theta - 1})^2, \tag{3.5}$$

where z is the corresponding critical value of the standard normal distribution. We calculate $cv = 16718$.

We wrote a program in MATLAB to run this test. The value we obtained for the test statistic χ^2 was $9935.7 < cv$ and we do not reject H_0. We ran the program again but this time the seed used to produce the vector of length seven was computed from the clock in the computer. The result was $\chi^2 = 10,036 < cv$ with no rejection of the null hypothesis. This does not prove "randomness" but it gives us confidence to use this MATLAB program to produce sequences of vectors in $[0, 1]^n$, $n \geq 2$, for our fuzzy Monte Carlo studies.

3.5 Random Vectors: Non-negative Integers

Now we want sequences of vectors $v = (x_1, ..., x_n)$, where each x_i is a non-negative integer in $[a, b]$, so that the v should uniformly fill the region $I \cap [a, b]^n$, $a \geq 0$, where I denotes integers. This may be used in Chapters 18, 20-23, and 25-26. Use a quasi-random number generator to get v with each $x_i \in [0, 1)$. Set $w = (a - b + 1)v + (a - 0.5)$ which puts each $x_i \in [a - 0.5, b + 0.5]$. Round the x_i to integers as follows: (1) if the decimal part of x_i is less than 0.5 round down to y_i; (2) if the decimal part of x_i is greater than 0.5 found up to y_i; and (3) if the decimal part of x_i equals 0.5 round to the nearest even integer y_i. The vector $u = (y_1, ..., y_n)$ is what we want.

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Linear Regression. Soft Computing 11, 991–996 (2007)
2. Buckley, J.J.: Fuzzy Probability and Statistics. Springer, Heidelberg (2006)
3. Deley, D.W.: Computer Generated Random Numbers, http://world.std.com/~franl/crypto/random-numbers.html
4. Henderson, S.G., Chiera, B.A., Cooke, R.M.: Generating “Dependent” Quasi-Random Numbers. In: Proceedings of the 2000 Winter Simulation Conference, Orlando, FL, December 10-13, 2000, pp. 527–536 (2000)
5. Lindgren, B.W.: Statistical Theory, 3rd edn. MacMillan, New York (1976)
6. Kimura, S., Matsumura, K.: Genetic algorithms using low-discrepancy sequences. In: Proceedings of GECCO 2005: 2005 conference on Genetic and evolutionary computation, New York, pp. 1341–1346 (2005)
7. Lord, G., Paskov, S., Vanderhoof, I.T.: Using Low-Discrepancy Points to Value Complex Financial Instruments, Contingencies, pp. 52–56 (September/October 1996)
8. Mango, D.: Random Number Generation Using Low Discrepancy Points. In: Proceedings of CAS Forum: 1999 Spring, Reinsurance Call Papers, Arlington, VA, USA, pp. 335–362 (1999)
9. MATLAB, The MathWorks, http://www.MathWorks.com
10. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C: The Art of Scientific Computing, 2nd edn. Cambridge Univ. Press, Cambridge (2002)
11. Reuillon, R., Hill, D.R.C.: Research Report LIMOS/RR-05-09: Unrolling optimization technique for quasi-random number generators. In: Proceeding of OICMS 2005: 1st Open International Conference on Modeling & Simulation, June 12-15, 2005, Blaise Pascal University, France (2005)
12. http://www.csit.fsu.edu/~burkardt/cpp_src/cpp_src.html
13. http://www.puc-rio.br/marco.ind/quasi_mc.html

4 Random Fuzzy Numbers and Vectors

4.1 Introduction

We first discuss how we plan to produce random triangular (trapezoidal) fuzzy numbers. Then we define quadratic fuzzy numbers and show how we can produce random sequences of these fuzzy numbers using sequences of random vectors of real numbers. Random quadratic fuzzy numbers are used in Chapters 10 and 27. In the last section we consider producing random sequences of fuzzy vectors $\overline{V} = (\overline{X}_1, ..., \overline{X}_n)$ where the $\overline{X}_i$ are triangular/trapezoidal fuzzy numbers or quadratic fuzzy numbers. These random sequences of fuzzy vectors have applications throughout these chapters; triangular fuzzy numbers (TFNs) in Chapters 11-12, 15-17; trapezoidal fuzzy numbers (TrFNs) in Chapter 13; and QBGFNs (defined below) in Chapters 6-10, 19 and 27. We also abbreviate "fuzzy numbers" as FNs. Trapezoidal shaped fuzzy numbers have applications in Chapters 13, 20-22 and 26.

4.2 Random Triangular/Trapezoidal Fuzzy Numbers

First consider producing a random sequence of triangular fuzzy numbers in an interval $[a, b]$. We want to create the sequence $\overline{X}_i = (x_{i1}/x_{i2}/x_{i3})$, $i = 1, 2, 3, ...,$ with $\overline{X}_i \in [a, b]$ all i. Using our Sobol quasi-random number generator (Chapter 3), we take three consecutively generated numbers and order them. Thus we make a sequence of random vectors $v_i = (z_{i1}, z_{i2}, z_{i3})$, $z_{i1} < z_{i2} < z_{i3}$, $i = 1, 2, 3, ...$ in $[0, 1]^3$. Then set $x_{ij} = (b - a)z_{ij} + a$, $j = 1, 2, 3$ and $i = 1, 2, 3, ...$

Now consider making a random sequence of trapezoidal fuzzy numbers in $[a, b]$. Assume the sequence is $\overline{X}_i = (x_{i1}/x_{i2}, x_{i3}/x_{i4})$, $i = 1, 2, 3, ...$ Using our Sobol quasi-random number generator (Chapter 3) we take four consecutively generated numbers and order them. Thus we make a sequence of random vectors $v_i = (z_{i1}, z_{i2}, z_{i3}, z_{i4})$, $z_{i1} < z_{i2} < z_{i3} < z_{i4}$, $i = 1, 2, 3, ...$ in $[0, 1]^4$. Then set $x_{ij} = (b - a)z_{ij} + a$, $j = 1, 2, 3, 4$ and $i = 1, 2, 3, ...$ We will use these random FNs in Section 4.5.

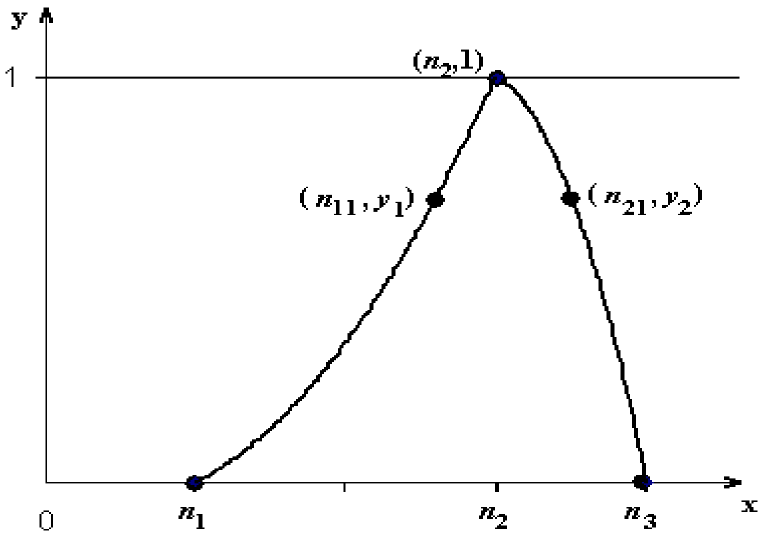


Fig. 4.1. Random Quadratic Fuzzy Number $\overline{N}$: Case 1

4.3 Random Quadratic Fuzzy Numbers

4.3.1 Generated from Implicit Quadratic Functions

Let $\overline{N} \approx (n_1/n_2/n_3)$, or $\approx (n_1/n_2, n_3/n_4)$, in $[0,1]$ be a triangular/trapezoidal shaped fuzzy number. In this section we will discuss how we plan to produce a sequence of random triangular/trapezoidal shaped fuzzy numbers in some interval $[a,b]$. We first make $\overline{N}$ in $[0,1]$ and multiply by $b-a$ and add a to get it into $[a,b]$. We always start out trying to make a triangular shaped fuzzy number but sometimes, as we show below, it turns out to be a trapezoidal shaped fuzzy number.

Let $y = f_1(x)$ denote the function that makes the left side of the membership function $y = \overline{N}(x)$, $0 \leq y \leq 1$, $n_1 \leq x \leq n_2$. We assume that $f_1(x)$ is continuous and strictly increasing with $f_1(n_1) = 0$ and $f_1(n_2) = 1$. Next let $y = f_2(x)$ denote the function that makes the right side of the membership function $y = \overline{N}(x)$, $0 \leq y \leq 1$, $n_2 \leq x \leq n_3$. We assume that $f_2(x)$ is continuous and strictly decreasing with $f_2(n_2) = 1$ and $f_2(n_3) = 0$. Notice that if we substitute α for y an α-cut of $\overline{N}$ can be written $[f_1^{-1}(\alpha), f_2^{-1}(\alpha)]$.

In this chapter we will use quadratic functions for the $f_i(x)$. Let $a_{i1}x^2 + a_{i2}x + a_{i3} = f_i(x)$, $i = 1,2$. We may extend the results to higher order polynomials. Now choose n_{11} and n_{21} so that $n_1 < n_{11} < n_2$, $n_2 < n_{21} < n_3$, and then choose y_1, y_2 in $(0,1)$. The left side of $\overline{N}$ will be determined by the three points $(n_1, 0)$, (n_{11}, y_1), $(n_2, 1)$ because these three points, assuming they do not lie in a straight line, uniquely determine the a_{1j} in $y = a_{11}x^2 + a_{12}x + a_{13}$. The right side of $\overline{N}$ will be determined by the three points $(n_2, 1)$, (n_{21}, y_2), $(n_3, 0)$ because these three points, assuming they do not lie in a straight line, uniquely determine the

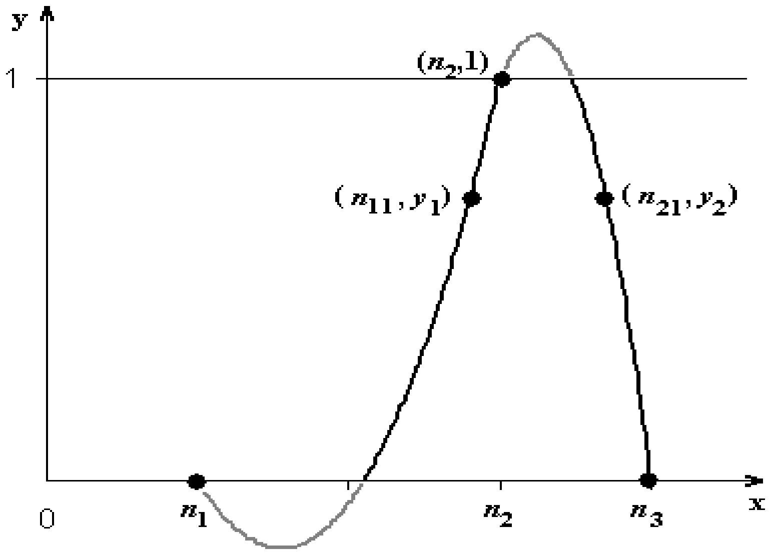


Fig. 4.2. Random Quadratic Fuzzy Number $\overline{N}$: Case 2

a_{2j} in $y = a_{21}x^2 + a_{22}x + a_{23}$. So we require the seven numbers n_1, n_{11}, y_1, n_3, n_{21}, y_2 and n_3 to construct our triangular shaped fuzzy number $\overline{N}$. We will call these fuzzy numbers quadratic fuzzy numbers because they have quadratic membership functions. In the case of Figure 4.1 it is also a triangular shaped fuzzy number. As shown in Figure 4.2 this method may result in a trapezoidal shaped fuzzy number or a triangular shaped fuzzy number. One can identify nine different cases of quadratic fuzzy numbers; we have shown one case which results in a trapezoidal shaped fuzzy number for which the left support and the right extent of the core must yet be determined.

To randomly generate a Case 1 $\overline{N}$ in $[0, 1]$ we randomly produce random numbers $x_1, ..., x_7$ in $[0, 1]$, using our Sobol quasi-random number generator (Chapter 3), giving the random vector $w = (x_1, ..., x_7) \in [0, 1]^7$. In w first randomly choose two values say, for example, x_3 and x_6. Then set $y_1 = x_3$ and $y_2 = x_6$. Now order the remaining five numbers from smallest to largest giving, for example, $x_5 < x_2 < x_7 < x_1 < x_4$. Then define $n_1 = x_5$, $n_{11} = x_2$, $n_2 = x_7$, $n_{21} = x_1$ and $n_3 = x_4$. We now have the five points to get triangular shaped fuzzy number $\overline{N}$. See Figure 4.1.

However, there are problems with constructing quadratic fuzzy numbers this way. The graph of the left side of the quadratic fuzzy number between n_1 and n_2 in Figure 4.1 may go above $y = 1$ or below $y = 0$, but not both. This may happen as in Figure 4.2. Then we use either $min\{f_1(x), 1\}$ or $max\{f_1(x), 0\}$ for the left side. Also, the graph of the right side of the quadratic fuzzy number between n_2 and n_3 in Figure 4.1 may go above $y = 1$ or below $y = 0$, but not both. This may happen as in Figure 4.2. Then we use either $min\{f_2(x), 1\}$ or $max\{f_2(x), 0\}$ for the right side. This produces eight more cases of quadratic fuzzy numbers, some trapezoidal shaped fuzzy numbers having quadratic sides, some triangular shaped fuzzy numbers with quadratic sides. But in all cases the sides of the fuzzy number are described by quadratic functions. These cases make up all our quadratic fuzzy numbers to be randomly generated from random vectors $w = (x_1, ..., x_7)$.

Because of the problems discussed above, curves going above (below) the horizontal line $y = 1$ ($y = 0$), we will not use this method of generating random quadratic fuzzy numbers in this book. Instead, we will employ the procedure outlined in the next section.

4.3.2 Generated from Parametric Quadratic Functions, Bézier Fuzzy Numbers

We now show that we can also generate random triangular shaped fuzzy numbers using random vectors $w = (x_1, ..., x_5) \in [0,1]^5$ of length five. The definition and properties of Bézier generated fuzzy numbers (BGFNs) is a result of research on random fuzzy numbers done by Leonard Jowers at the University of Alabama at Birmingham [2]. Bézier generated fuzzy numbers have a 100% yield of triangular shaped FNs. What happened in Figure 4.2 will not occur now. Methods for two types of BGFNs are given in [3]. Here we only discuss those which result in FNs which have quadratic membership functions, quadratic Bézier generated fuzzy numbers (QBGFNs) are FNs whose membership functions, left and right, are defined by parabolas generated from Bézier curves [1]. Yet another representation having membership functions consisting of hyperbolas, ellipses, a line and a parabola may be generated from weighted quadratic rational Bézier curves; however, those conic Bézier generated fuzzy numbers (CBGFNs) do not generate quadratic membership functions in their explicit forms and are not used in this book.

A full explanation of QBGFNs is lengthy and beyond the scope here. However, a full explanation is available in [3], which includes how one may generate QBGFNs (parabolic) or CBGFNs, and recover their explicit forms.

Creation of Bézier curves can be visualized through de Castlejau's Algorithm [1]. Given three control points, b_0, b_1, b_2, we create a curve as a moving point, b_0^2 (Figure 4.3), on a moving line, $\overline{b_0^1 b_1^1}$. Both move, maintaining the relationship $\overline{b_0 b_0^1}$ is to $\overline{b_0^1 b_1}$, as $\overline{b_1 b_1^1}$ is to $\overline{b_1^1 b_2}$, as $\overline{b_0^1 b_0^2}$ is to $\overline{b_0^2 b_1^1}$. The point b_0^2 is determined by repeated linear interpolation.

A quadratic Bézier curve is an arc length parameterization, t restricted to $[0,1]$, of a parabola. Where b_i is $\begin{bmatrix} x_i & y_i \end{bmatrix}^T$, we have

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = (1-t)^2 b_0 + 2(1-t)t b_1 + t^2 b_2. \tag{4.1}$$

b_0, b_1, and b_2 are coefficients to Bernstein polynomials, $B_{k,n}(t) = \binom{n}{k} t^k (1-t)^{n-k}$, of degree 2; that is, $B_{0,2} = (1-t)^2$, $B_{1,2} = 2(1-t)t$, and $B_{2,2} = t^2$. Bernstein polynomials of degree n are the terms of the expansion of $[(1-t)+t]^n$.

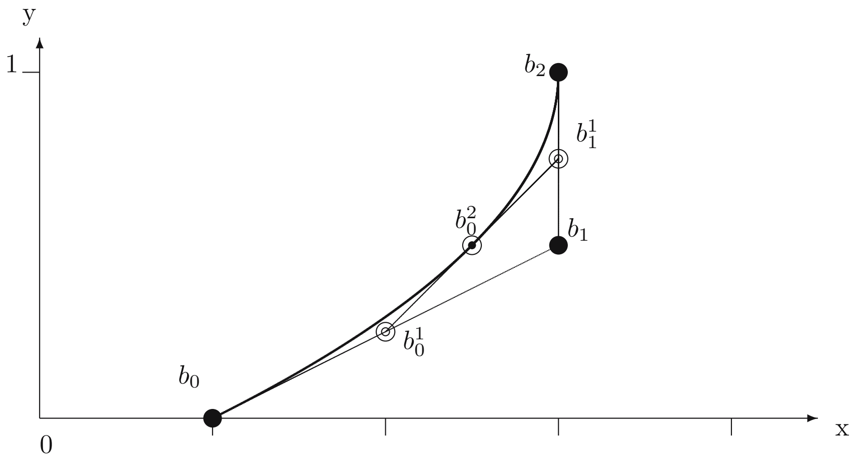


Fig. 4.3. A point, b_0^2, in de Castlejau's Algorithm

We now define quadratic Bézier generated fuzzy numbers (QBGFNs) as fuzzy numbers (FNs) whose membership function is a pair of quadratic Bézier curves [2]. This representation requires a vector of length 5. We produce a quadratic fuzzy number from a "random" vector $v = (z_1, ..., z_5)$ generated from some 'random' method. Three elements of a vector of length 5 define a TFN template of a QBGFN. The two additional elements define the shape of the left and right membership functions.

First we specify the range over which we allow the support, which we now assume to be $[0, M]$, $M > 0$. A stream of crisp random integers gives a vector v of length 5 and we first map each component into [0,1]. So assume $v = (z_1, ..., z_5)$, $z_i \in [0, 1]$ all i. Let the final "random" vector, used for the quadratic fuzzy number $\overline{A}$ be $w = (x_1, ..., x_5)$. However, our construction is to first determine the support and vertex of a TFN, then a parameter for left (right) side of the membership function. The first three elements of the vector v are sorted to create $z_2 < z_1 < z_3$, then $x_1 = Mz_2$, $x_2 = Mz_1$, $x_3 = Mz_3$ and the TFN is $(x_1/x_2/x_3)$ in $[0, M]$. Next z_4 (z_5) are transformed into x_4 (x_5) which produce the quadratic membership function for the left (right) side the quadratic fuzzy number $\overline{A}$.

The left membership function is defined in the following manner. z_4 is mapped to x_4 by $x_4 = (2z_4 - 1) \times ((x_2 - x_1) + 1)$ (similarly, x_5 is computed from z_5 as $x_5 = (2z_5 - 1) \times ((x_3 - x_2) + 1))$. x_4 will be in $[-(x_2 - x_1 + 1), (x_2 - x_1 + 1)]$. If $x_4 < 0$, x_4 defines a Bézier control point on the path counterclockwise from $(x_1, 0)$ to $(x_2, 0)$ to $(x_2, 1)$. $|x_4|$ is the Manhattan distance from $(x_1, 0)$ along that path. See b_0, b_1 and b_2 in Figure 4.4, the Bézier control points for the left membership function. One may see that, if $x_4 < 0$, the Bézier control points of the left membership are either $\{(x_1, 0), (|x_4|, 0), (x_2, 1)\}$ for $|x_4|$ the distance between $(x_1, 0)$ and $(x_2, 0)$, or $\{(x_1, 0), (x_2, |x_4 - (x_2 - x_1)|), (x_2, 1)\}$ for $|x_4|$ greater than the distance between $(x_1, 0)$ and $(x_2, 0)$, Otherwise, $x_4 \geq 0$ similarly defines a Bézier control point on the path clockwise from $(x_1, 0)$ to $(x_1, 1)$ to $(x_2, 1)$.

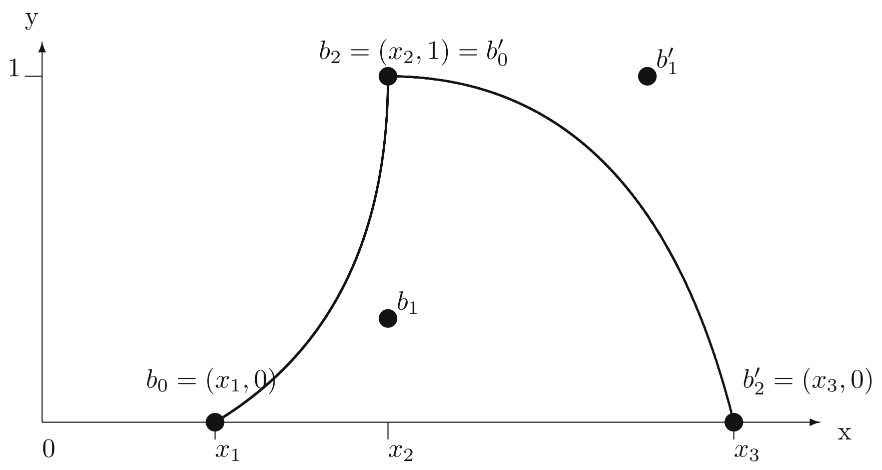


Fig. 4.4. A Parabolic QBGFN Fuzzy Number $\overline{A}$

The same process is used to determine the Bézier control points b'_0, b'_1 and b'_2 (see Figure 4.4) for the right membership function.

In [3] we show how a Bézier curve may be converted into its implicit form; i.e., $y = f(x)$. Additionally there, we show a quadratic Bézier is converted to an implicit quadratic polynomial function.

These QBGFNs are used extensively in Chapters 6-10 and their use is proposed in Chapters 19 and 27.

4.4 Comparison of Random Fuzzy Vectors

In our evaluation of streams of FNs, to be continued in the next chapter, we investigated our algorithms for comparing fuzzy numbers. For Buckley's method (Section 2.6.1) we evaluated the effect of choices of threshold values. Buckley thresholds of 0.7, 0.8, and 0.9 were evaluated. 0.7 generated an excessive number of "equal" results. 0.9 appeared too much like a vertex comparison. We determined that a Buckley threshold of 0.8 provided an acceptable comparison.

We also investigated differences among our three chosen comparison methods, Buckley, Kerre (Section 2.6.2), and Chen (Section 2.6.3). $10,000$ FNs created using various crisp input streams were evaluated. In Table 4.1 "Type FN" refers to the type of input stream; for example, "Sobol 10" is for QBGFNs generated from Sobol quasi-random numbers which had been generated 10 at a time (as 10-tuples). Given two fuzzy numbers $\overline{M}$ and $\overline{N}$, using any one of the three comparison methods, only one of the ordering results $\overline{M} < \overline{N}$, or $\overline{M} \approx \overline{N}$, or $\overline{M} > \overline{N}$ can be true. "All Agree" are the number times all three methods agreed upon the order of a pair of FNs. "Non-agree" represents the rest of the $10,000$. "B&K agree" are the number of times that Buckley and Kerre agree when Chen did not. "B&C agree" are the number of times that Buckley and Chen agree when Kerre did not. "K&C agree" are the number of times that Kerre and

Table 4.1. Comparison of Buckley (0.8), Kerre, and Chen Methods

Type FN	All agree	Non-agree	B&K agree	B&C agree	K&C agree	All disagree
QBGFN						
Pseudo	7984	2016	289	267	1424	36
True	8032	1968	249	309	1384	26
Faure 5	6402	3598	105	100	3363	30
Halton 5	8151	1849	240	275	1284	50
Niederreiter 5	8471	1529	180	176	1157	16
Sobol 5	8667	1333	86	121	1109	17
Faure 10	8019	1981	280	257	1410	34
Halton 10	8039	1961	275	259	1390	37
Niederreiter 10	7953	2047	286	267	1472	22
Sobol 10	7956	2044	283	288	1440	33
Faure 15	8155	1845	200	264	1356	25
Halton 15	7977	2023	307	270	1417	29
Niederreiter 15	7961	2039	300	270	1425	44
Sobol 15	8022	1978	276	267	1398	37
TFN						
Pseudo	8499	1501	203	202	1050	46
True	8461	1539	211	236	1065	27

Chen agree when Buckley did not. "All disagree" are the number of times that each gave a different result. An example of an "All disagree" situation could be: Buckley's method says $\overline{M} < \overline{N}$, Kerre's procedure implies $\overline{M} \approx \overline{N}$, but Chen's way gives $\overline{M} > \overline{N}$.

Table 4.1 shows that results did not greatly vary among FNs generated from our various streams of crisp random numbers. We also found that results did not vary greatly between QBGFNs and TFNs. We found that Kerre's method (Section 2.6.2) and Chen's method (Section 2.6.3) agreed on average for almost 95% of the comparisons. In Table 4.1, one may add columns "All agree" and "K&C agree", for any row, to see that for the $10,000$ comparisons about $9,500$ show the same result for Kerre's and Chen's methods. Now both Kerre's method and Chen's method are used in Chapters 6-8, so we will not be too surprised if they give similar results. Buckley's method is not used in Chapters 6-8 but will be employed in the rest of the book.

4.5 Random Fuzzy Vectors

We first look at obtaining sequences of random fuzzy vectors whose components are all quadratic fuzzy numbers. Suppose we want $\overline{V}_i = (\overline{X}_{i1}, \overline{X}_{i2}, \overline{X}_{i3})$, $i = 1, 2, 3, ...$ Using our quasi-random number generator we get a sequence of random vectors $v_i = (x_{i1}, ..., x_{i5})$, $i = 1, 2, 3, ...$ We take v_1 for $\overline{X}_{11}$, v_2 for $\overline{X}_{12}$, v_3 for $\overline{X}_{13}$, etc. We could also make v_i have length 15 and take the first five components for $\overline{X}_{i1}$, the second five components for $\overline{X}_{i2}$, etc. We will usually use the first method.

If we wanted a sequence of random fuzzy vectors $\overline{V}_i$ whose components are all TFNs we may use random crisp vectors v_i of length nine. Take the first three components to make $\overline{X}_{i1}$, etc. For random fuzzy vectors of trapezoidal fuzzy numbers we can use random crisp vectors of length 12 using the first four for $\overline{X}_{i1}$, etc.

References

1. Farin, G.: Curves and Surfaces for CADG, 5th edn., pp. 219–224. Academic Press, London (2002)
2. Jowers, L.J.: Fuzzy Numbers, A Novel Representation Using Bézier Curves. In: UAB Graduate Student Research Day 2005, Birmingham, AL (March 4, 2005)
3. Jowers, L.J., Buckley, J.J., Reilly, K.D.: Representation of Conic and Parabolic Fuzzy Numbers Using Bézier Curves. In: Proceedings IPMU 2006. 11th International Conference of Information Processing and Management of Uncertainty in Knowledge Based Systems, Paris, France, July 2, 2006, p. 6 (2006)

5 Tests for Randomness

5.1 Introduction

Any sequence claimed to be random, real numbers or fuzzy numbers, must be tested for randomness. We first test our sequence of fuzzy random numbers, obtained from Sobol quasi-random numbers, for randomness using a run test and then a frequency test. We identified two types of triangular shaped fuzzy numbers from Chapter 4: (1) quadratic fuzzy numbers generated from 7-tuples; and (2) quadratic Bézier generated fuzzy numbers (QBGFNs). For reasons given there we direct our attention to QBGFNs. A run test depends on what definition of $\leq$ between fuzzy numbers we are using. So we do the run test three times on the Bézier fuzzy numbers; first using Buckley's Method of $\leq$ (Section 2.6.1) next using Kerre's Method of $\leq$ (Section 2.6.2) and lastly using Chen's Method of $\leq$ (Section 2.6.3). We must also test our sequence of random fuzzy vectors for randomness. We have seen that sequences of random numbers can pass randomness tests but when they are used to build vectors the resulting sequence of vectors can fail randomness tests (Chapter 3). We will test our sequences of random vectors, whose components are all TFNs, for randomness using a chi-square test.

Actually, these randomness tests are not too important. We plan to use our sequences of random fuzzy numbers/vectors to generate approximate solutions to fuzzy optimizations problems. What is important is for our method to uniformly fill the search space to a fuzzy optimization problem. We argue that this is true in the last section in this chapter.

5.2 Random Fuzzy Numbers

The first test is the run test. A frequency (chi-square) test is at the end of this section.

5.2.1 Run Test

Assume we have a sequence of triangular shaped fuzzy numbers $\overline{N}_i$, $i = 1, 2, 3, ..., n$, in $[0, M]$ which we want to test for randomness. There are many randomness tests for sequences of real numbers ([3],[7]). But, most are not readily adapted to fuzzy numbers. The "run test" looks to be one of the easier to apply to fuzzy numbers and we will use it in this section. Make a new sequence using the symbols $+$, 0 and $-$ as follows: (1) if $\overline{N}_i < \overline{N}_{i+1}$ use $+$; (2) if $\overline{N}_i \approx \overline{N}_{i+1}$ use 0; and (3) if $\overline{N}_i > \overline{N}_{i+1}$ use $-$. From the original sequence of fuzzy numbers we get, for example

$$++0---++++++00-----....0++++++-----++++++------, \tag{5.1}$$

if $n = 56$. Now in the run test, applied to real number sequences, there will be no zeros. We will omit all the zeros and we obtain

$$++---++++++-----++++...++++++-----++++++------, \tag{5.2}$$

for $n = 50$. We count the total number of runs with the first run $++$, the second run $---$, third run $++++++$, etc. In our example above assume we get the total runs $s = 10$.

We do a statistical hypothesis test with null hypothesis H_0 the sequence of fuzzy numbers is random and the alternative hypothesis H_1 that the sequence is not random. We choose the level of significance (type I error) to be $\gamma = 0.05$. Under the null hypothesis the mean of s is $(2n-1)/3$ and the variance of s is $(16n-29)/90$. Also, we know that for large samples (say $n \geq 50$) the distribution of s is approximately normal ([3],[7]). In our example we have $n = 50$, the mean of s is 33 and the variance equals 8.5667. Doing a two sided test, and incorporating a continuity correction of 0.5, the left critical region is

$$(s + 0.5 - 33)/\sqrt{8.5667} \leq -1.96, \tag{5.3}$$

and the right critical region is

$$(s - 0.5 - 33)/\sqrt{8.5667} \geq 1.96. \tag{5.4}$$

So we reject H_0 when $s \leq 26$ or $s \geq 40$. In our example with $s = 10$ we reject H_0 and conclude that this sequence of fuzzy numbers is not random.

The left critical value guards against trends and the right critical value guards against cycles. A trend would be a sequences of increasing, or decreasing, fuzzy numbers leading to too few runs and $s \leq 26$. Cycles would produce something like $++--++--++--$.... and too many runs with $s \geq 40$.

There are two other variations on the run test that could be used. Let $\overline{M}$ be the median of the sequence $\overline{N}_i$, $i = 1, ..., n$. The fuzzy median of a finite sequence of fuzzy numbers would need to be defined. Make a new sequence using the symbols $+$, 0 and $-$ as follows: (1) if $\overline{N}_i < \overline{M}$ use $-$; (2) if $\overline{N}_i \approx \overline{M}$ use 0; and (3) if $\overline{N}_i > \overline{M}$ use $+$. Omit the zeros. Count the runs below the median, count the runs above the median and let s be the total number of runs. Then using a normal approximation, similar to equations (5.3) and (5.4), we can do the hypothesis test of H_0 it is a random sequence versus H_1 it is not random ([4],[7]). A third test involves using the first sequence of $+'s$ and $-'s$ described

Table 5.1. Run Test on QBGFNs: Buckley

Type	mean	σ^2	min	max	Reject H_0 $\gamma = 0.05\%$	Reject H_0 $\gamma = 0.01\%$
True	65.64	4.4256	57	79	5	1
Pseudo	65.47	3.9631	55	76	3	1
Sobol 5	76.68	3.0711	69	84	67	54
Faure 5	43.97	1.9871	39	49	100	100
Halton 5	66.47	3.1187	59	75	0	0
Niederreiter 5	73.82	3.5316	66	80	39	24
Sobol 10	58.13	3.4979	52	65	44	35
Faure 10	94.55	2.8047	87	100	100	100
Halton 10	73.63	3.9097	66	82	35	25
Niederreiter 10	58.36	3.8046	51	72	39	31
Sobol 15	66.35	5.2597	55	79	8	4
Faure 15	65.39	4.7798	32	69	6	4
Halton 15	65.46	2.6263	59	71	0	0
Niederreiter 15	65.94	7.4262	47	84	21	17

above, not constructed from the median, and counting the length of a run of $+'s$ or a run of $-'s$. An asymptotically chi-square distributed test statistic based on the number of runs of length $L = 1, 2, 3, 4, 5$ and $L \geq 6$ is given in [5]. However, the author in [5] suggests a sample size of $n \geq 4000$ for a good approximation. But we shall use only the first run test given in equations (5.3) and (5.4) in this chapter.

Run Test, Buckley's Method

Using streams of 500,000 "random" numbers generated by the various methods discussed in Chapter 3, we may create quadratic fuzzy numbers (Chapter 4) either from 7-tuples as implicitly generated quadratic fuzzy numbers, or from 5-tuples as quadratic Bézier generated fuzzy numbers, display them, and compute statistics for the run test. As explained in Section 4.3.2, to increase the yield of triangular shaped FNs, we work in Chapters 6-10 with quadratic Bézier generated fuzzy numbers (QBGFNs). As explained in Section 3.4 to reduce the effect of the "curse of dimensionality" on randomness for multi-dimensional vectors, we will use the quasi-random procedure to get random vectors that will define our fuzzy numbers.

In our computer program `FuzzyRunsTest`, as we create these QBGFNs, we apply the run test described above. Let the sequence of random fuzzy numbers be $\overline{A}_1$,.....,$\overline{A}_n$. The method used to decide on $\overline{A}_i < \overline{A}_{i+1}$, $\overline{A}_i \approx \overline{A}_{i+1}$ and $\overline{A}_i > \overline{A}_{i+1}$ is discussed in Section 2.6.1 using $\eta = 0.8$. If in the comparison of two fuzzy numbers in the sequence we get $\overline{A}_i \approx \overline{A}_{i+1}$, then we discard (reject) $\overline{A}_{i+1}$. Each test consists of evaluating QBGFNs until 100 are accepted. For a stream of 100 accepted fuzzy numbers, as we showed above for 50, the mean is 66.33334 and the variance is 17.45556. If $\gamma = 0.05$ the null hypothesis (that the stream is

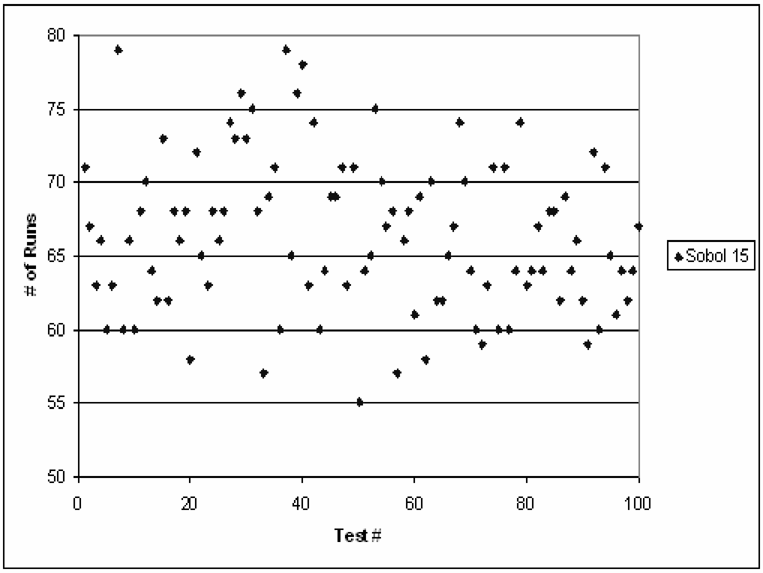


Fig. 5.1. Results on the Number of Runs Using 'Sobol 15' Quasi-Random Numbers & Buckley's Comparison Method

random) is rejected, if the number of runs is ≤ 57 or ≥ 76 (similar to equations (5.3) and (5.4)). When $\gamma = 0.01$, we reject the null hypothesis if the number of runs is ≤ 55 or ≥ 78.

We replicated the test 100 times using Buckley's comparison method. We show, in Table 5.1, results for streams of quadratic fuzzy numbers generated from various streams of crisp numbers. We used true random, pseudo-random, and four types of quasi-random numbers (Sobol, Faure, Halton, and Niederreiter). In Table 5.1 "Reject H_0" means the number of times, in 100 test runs, we rejected the null hypothesis of randomness. Notice from Table 5.1 how the null hypothesis of randomness shows a wide range of results for the various quasi-random numbers investigated. Using $\gamma = 0.05$ ($\gamma = 0.01$) we would expect rejection, assuming randomness, on the average 5 times (one time) for 100 tests. Others ([1],[2],[6]) have also noted sensitivity of quasi-random numbers to dimension and starting values.

We notice from Table 5.5 that the results for "Halton 5" and "Halton 15" are good but those for "Faure 5" and "Faure 10" are surprisingly non-random. The results for "Sobol 15" are not too bad. See also Figure 5.1. We will be using the Sobol quasi-random number generator in the rest of the book starting in the next chapter. The key question is: will this quasi-random number generator randomly produce sequences of random fuzzy numbers/vectors that uniformly fill the search space for a fuzzy optimization problem? This is discussed in Section 5.3.

Table 5.2. Run Test on QBGFNs: Kerre

Type	mean	σ^2	min	max	Reject H_0 $\gamma = 0.05\%$	Reject H_0 $\gamma = 0.01\%$
True	66.76	4.4384	56	75	2	1
Pseudo	67.75	4.1739	58	79	4	4
Sobol 5	79.57	2.5947	73	86	94	91
Faure 5	34.48	2.5445	28	40	100	100
Halton 5	67.05	3.0264	61	75	0	0
Niederreiter 5	74.78	2.1676	70	80	40	16
Sobol 10	57.71	3.0527	51	65	50	40
Faure 10	95.17	2.5783	90	100	100	100
Halton 10	72.67	4.4813	61	82	24	18
Niederreiter 10	55.57	2.9310	50	64	80	64
Sobol 15	67.13	5.8855	53	84	13	11
Faure 15	67.00	1.3707	56	70	1	1
Halton 15	66.67	2.2476	61	71	0	0
Niederreiter 15	67.43	8.8126	47	83	34	30

Run Test, Kerre's Method

We replicated the test 100 times using Kerre's comparison method. We show, in Table 5.2, results for streams of quadratic fuzzy numbers generated from various streams of crisp numbers. We used true random, pseudo-random, and four types of quasi-random numbers (Sobol, Faure, Halton, and Niederreiter). In Table 5.2 "Reject H_0" means the number of times, in 100 test runs, we rejected the null hypothesis of randomness.

Run Test, Chen's Method

We replicated the test 100 times using Chen's comparison method. We show, in Table 5.3, results for streams of quadratic fuzzy numbers generated from various streams of crisp numbers. We used true random, pseudo-random, and four types of quasi-random numbers (Sobol, Faure, Halton, and Niederreiter). In Table 5.3 "Reject H_0" means the number of times, in 100 test runs, we rejected the null hypothesis of randomness.

5.2.2 Frequency Test

Test for Randomness, QBGFNs

Suppose we have a sequence of fuzzy numbers $\overline{X}_i$ in some interval $[a, b]$ we wish to test for randomness. Assume the interval is $[0, 10]$. We compute the centroid of the fuzzy numbers producing a real sequence $\delta_1, ..., \delta_n, ...$ We will perform a frequency test on the δ sequence to test for randomness. With all the fuzzy numbers in $[0, 10]$ very few will have their centroid near zero or ten. We may get the left (right) end point of the support close to zero (ten) but not the centroid

Table 5.3. Run Test on QBGFNs: Chen

Type	mean	σ^2	min	max	Reject H_0 $\gamma = 0.05\%$	Reject H_0 $\gamma = 0.01\%$
True	67.43	4.4727	58	76	3	0
Pseudo	67.70	4.2200	56	78	2	2
Sobol 5	79.91	2.4499	73	86	95	93
Faure 5	36.53	2.3375	32	41	100	100
Halton 5	68.23	3.1362	61	77	2	1
Niederreiter 5	75.05	2.0120	72	80	45	20
Sobol 10	57.79	2.7972	49	64	45	32
Faure 10	95.37	2.5767	90	100	100	100
Halton 10	74.07	4.1736	64	86	37	29
Niederreiter 10	55.99	3.0434	50	68	75	58
Sobol 15	67.41	5.8657	55	80	15	9
Faure 15	67.28	0.9648	65	71	0	0
Halton 15	67.15	2.1760	61	73	0	0
Niederreiter 15	67.77	8.8064	47	85	35	31

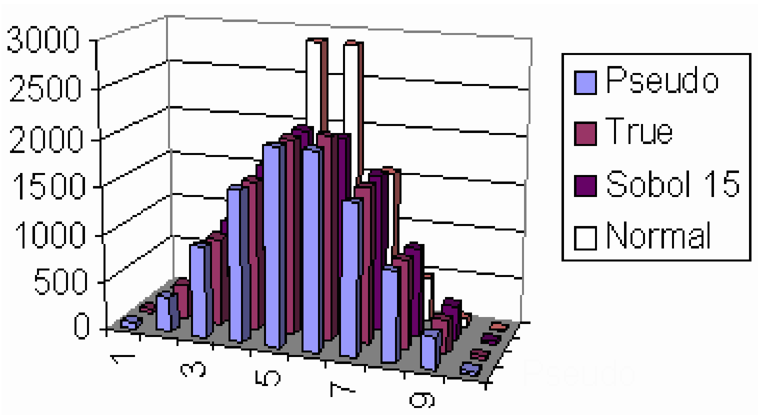


Fig. 5.2. Comparison of Frequency Test Results to Normal Distribution

which would be closer to the core. So we will use the interval $I = [1, 9]$ and divide it up into eight equal intervals $I_1 = [1, 2)$,...,$I_8 = [8, 9]$. Now we do a chi-square test, as in Section 3.4, on the δ sequence. The null hypothesis H_0 is that the δ sequence is random and the alternate hypothesis H_1 is that it is not random. Let the significant level $\gamma = 0.05$. O_i is the number of δ values in interval I_i, $i = 1, 2, ..., 8$. We reject any fuzzy number with centroid less than one or greater than nine. If N is the number of accepted fuzzy numbers in the sequence and H_0 is true, the expected number in each interval is $E = N/8$. We want E to be at least 5 so choose $N > 100$. The test statistic is

$$\chi^2 = \sum_{i=1}^{8} \frac{(O_i - E)^2}{E}, \tag{5.5}$$

with 7 degrees of freedom. The critical value cv is 14.1.

We chose to run this test on the Bézier fuzzy numbers, $N = 10,000$, generated from each of our crisp input streams. We obtain similar results for all tests (examples shown in Figure 5.2). We also test with TFNs generated from true random and pseudo-random crisp streams. For 'Sobol 15' QBGFNs, our ten bin counts are $\{52, 348, 978, 1623, 2047, 1987, 1641, 913, 367, 44\}$. The expected value (E) for the eight internal bins (total 9904) is $1238 = 9904/8$, so our test statistic (equation 5.5) was computed to be $\chi^2 = 2626$. Since this is greater than $cv = 14.1$ we do reject the null hypothesis of randomness. In all tests we rejected the null hypothesis of randomness. We conjecture (from Figure 5.2) that the distribution of the centroids is not approximately normal.

Test for Distribution Similarity, QBGFNs

Having rejected the null hypothesis of randomness, we select a null hypothesis of similarity of the sets of QBGFNs. Suppose we have a sequence of fuzzy numbers $\overline{X}_i$ in some interval $[a, b]$ we wish to test for similarity to QBGFNs generated from true random, crisp numbers. Assume the interval is $[0, 10]$. We compute the centroid of the fuzzy numbers producing a real sequence $\delta_1, ..., \delta_n, ...$. We perform a frequency test on the δ sequence to test for similarity.

So we will use the interval $I = [0, 10]$ and divide it up into ten equal intervals $I_0 = [0, 1)$,...,$I_9 = [9, 10]$. Now we do a chi-square test, as in Section 3.4, on the δ sequence. The null hypothesis H_0 is that the δ sequence has a similar distribution to that for QBGFNs generated from true random numbers and the alternate hypothesis H_1 is that it is does not have a similar distribution. Let the significant level $\gamma = 0.05$. O_i is the number of δ values in interval I_i, $i = 1, 2, ..., 10$. $N = 10,000$ is the number of fuzzy numbers in the sequence. The test statistic is

$$\chi^2 = \sum_{i=1}^{10} \frac{(O_i - E_i)^2}{E_i}, \tag{5.6}$$

with 9 degrees of freedom. The critical value cv is 19.9190.

We use the bin results from our tests for randomness in the previous section. For QBGFNs generated from a true random, crisp stream, our bin counts are $\{48, 368, 923, 1562, 2041, 2110, 1629, 927, 357, 35\}$. This data gives us the expected values E in the test statistic. For example $E_4 = 1562$. So we determine for 'Sobol 15' QBGFNs that $\chi^2 = 12.648$. Since this is less than $cv = 19.9190$ we do not reject the null hypothesis that the 'Sobol 15' QBGFNs distribution is similar to that for 'True Random' QBGFNs.

We also found that we do not reject this null hypothesis when we consider 'Pseudo-random' QBGFNs ($\chi^2 = 7.404$), 'Sobol 5' ($\chi^2 = 4.196$), or 'Sobol 10' ($\chi^2 = 12.264$). It appears that no matter how we produce a sequence of QBGFNs the distribution of the centroids are all roughly the same and this distribution is not uniform and not normal.

5.3 Search Space

The results of the run test and the frequency test are interesting but what we really want to show is that our method of generating sequences of QBGFNs, or triangular fuzzy numbers (TFNs), will uniformly fill the search space for a fuzzy optimization problem. We first look at sequences of QBGFNs and then sequences of TFNs. In both situations we will consider two cases. The first case is where the search space is an interval $[a, b]$ for $a < b$. This will be used in Chapter 10. The second case is for the search space a rectangular box in R^n. We will assume here that $n = 3$ so the box will be $\Omega = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$ for $a_i < b_i$, $i = 1, 2, 3$. This will be used in Chapters 7 an 8 where we will generate sequences of random fuzzy vectors $(\overline{X}_1, \overline{X}_2, \overline{X}_3)$ each $\overline{X}_i$ is a QBGFN in $[a_i, b_i]$, $i = 1, 2, 3$. In Chapters 6, 9 and 10 we use fuzzy vectors with only two components. Also, sequences of random vectors consisting of TFNs are used in Chapters 11,12,15,16. We need random sequences of trapezoidal fuzzy numbers in Chapter 13 and just random vectors of real numbers in Chapter 14.

We discuss the search space only for the Applications part of the book (Chapters 6 - 16). We do not consider uniformly filling the search space for the Future Research part of the book (Chapters 17 - 27).

5.3.1 Search Space $[a, b]$ for QBGFNs

Let $\mathbf{Q}$ = all QBGFNs in $[a, b]$. We will randomly generate a sequence of QBGFNs $\overline{X}_k$, $k = 1, 2, 3, ...$, in $[a, b]$ to uniformly fill $\mathbf{Q}$. Notice at this point this is an infinite sequence and we will later discuss finite sequences to be used in applications. We first discuss how we produce the sequence $\overline{X}_k$.

We use our Sobol quasi-random number generator to produce the sequence $v_k = (x_{1k}, ..., x_{5k}) \in [0, 1]^5$, $k = 1, 2, 3, ...$ Let $V = \{v_k | k = 1, 2, 3, ...\}$. For each v_k first choose the first three numbers in v_k. Transform these three numbers into the interval $[a, b]$. Let $z_{1k} = a+(b-a)x_{1k}$, $z_{2k} = a+(b-a)x_{2k}$,$z_{3k} = a+(b-a)x_{3k}$ all in $[a, b]$. Next order the z_{ik} from smallest to largest giving $d_k = z_{2k} < e_k = z_{1k} < f_k = z_{3k}$. Set $w_{1k} = x_{4k}$ and $w_{2k} = x_{5k}$. Then

$$u_k = (d_k, e_k, f_k, w_{1k}, w_{2k}) \tag{5.7}$$

defines a $\overline{X}_k$ in $\mathbf{Q}$. The construction of $\overline{X}_k$ is as follows: (1) the support (base) is $[d_k, f_k]$; (2) the vertex is at $x = e_k$; (3) as described in Section 4.3.2 w_{1k} defines the quadratic membership function for the left side; and (4) as described in Section 4.3.2 w_{2k} defines the quadratic membership function for the right side. This defines a mapping ψ of V into $\mathbf{Q}$. We know that V uniformly fills $[0, 1]^5$ (Section 3.4). This means that there will be no clusters of points and no "empty" regions in $[0, 1]^5$. We do not have a precise mathematical definition of "uniformly fills" but if you look at Figure 7.7.1, page 310, in [8] for $n = 2$, and only 1024 points, you will see the meaning of "uniformly fills". Also, since the sequence v_k passed the randomness test in Section 3.4 we feel that it uniformly fills $[0,1]^5$. Using a pseudo-random number generator it can produce clusters and empty regions in $[0,1]^n$ (Section 3.4). We now use this fact that V uniformly fills $[0,1]^5$ to argue that $\psi(V)$ will uniformly fill $\mathbf{Q}$.

Let $\mathbf{M} = \{(d,e,f,w_1,w_2)|d < e < f \in [a,b], w_1, w_2 \in [0,1]\}$. Now each $m \in \mathbf{M}$ uniquely defines a $\overline{X} \in \mathbf{Q}$ and each $\overline{X} \in \mathbf{Q}$ uniquely specifies an $m \in \mathbf{M}$. In this sense $\mathbf{M} = \mathbf{Q}$.

Now equation (5.12) shows the mapping ψ of V into $\mathbf{M}$. We easily extend it to all of $[0,1]^5$. For any vector in $[0,1]^5$ we first choose the first three components, map them into $[a,b]$, order them from smallest to largest, etc. and get a point in $\mathbf{M}$. So ψ maps $[0,1]^5$ onto $\mathbf{M}$. If V uniformly fills $[0,1]^5$ then surely $\psi(V)$ uniformly fills $\mathbf{M}$. If $\psi(V)$ uniformly fills $\mathbf{M}$ and $\mathbf{M} = \mathbf{Q}$ we have uniformly filled $\mathbf{Q}$.

We need to point out that there is an alternate method of producing the QBGFNs which may not uniformly fill the search space. Instead of always picking the first three members in v_k to get the base and vertex of the fuzzy number now assume we randomly choose three of the numbers in v_k. Let the tree numbers be x_{2k}, x_{3k} and x_{5k}. Map these three numbers into $[a,b]$ giving $z_{1k} = a+(b-a)x_{2k}$, $z_{2k} = a + (b-a)x_{3k}$ and $z_{3k} = a + (b-a)x_{5k}$. Now order these from smallest to largest producing $d_k = z_{2k} < e_k = z_{3k} < f_k = z_{1k}$. Set $w_{1k} = x_{1k}$ and $w_{2k} = x_{4k}$. Then we have defined u_k in equation (5.12) producing a member of $\mathbf{M}$ and a QBGFN $\overline{X}_k$ in $\mathbf{Q}$. But now we can not argue that the mapping from $[0,1]^5$ is onto $\mathbf{M}$ and may not uniformly fill $\mathbf{Q}$. Assume that the interval is $[0,10]$ and the point $v = (10r, 10s, 10t, w_1, w_2) \in \mathbf{M}$. Then $u = (r,s,t,w_1,w_2) \in [0,1]^5$. Let $V(i,j,k)$ be all the points in $[0,1]^5$ where we randomly choose the i^{th}, j^{th} and k^{th} points in v_k, $1 \leq i < j < k \leq 5$, to make a point in $\mathbf{M}$. There are ten (choose three from five) of these $V(i,j,k)$ giving $V(1,2,3)$, $V(1,2,4)$, ... ,$V(3,4,5)$. Whenever we choose three at random in v_k the first one, going from left to right in v_k, not picked we set equal to w_{1k} and other one not chosen becomes w_{2k}. Let u_{1j}, $j = 1,...,6$ be the six permutations of r,s,t in (r,s,t,w_1,w_2). So $u_{11} = (r,s,t,w_1,w_2)$,...,$u_{16} = (t,s,r,w_1,w_2)$. Assume that u_{1j} is not in $V(1,2,3)$ all j. If any u_{1j} is in $V(1,2,3)$ then it will be mapped to $v \in \mathbf{M}$. Next let u_{2j} be all six permutations of r,s,t in (r,s,w_1,t,w_2). Assume u_{2j} is not in $V(1,2,4)$ all j. Similarly we define v_{ij}, $i = 3,...,10$ and $j = 1,...,6$. In each case we assume that u_{ij} is not in the corresponding $V(i,j,k)$ all j. This means that it can happen that no point in $[0,1]^5$ maps to v. $\mathbf{M}$ can have empty regions and then so can $\mathbf{Q}$. Hence, we can not use this procedure to produce QBGFNs in $\mathbf{Q}$.

Now assume we use a finite sequence v_k, $k = 1,2,3,...,N$, in applications. We then plot these points in $[0,1]^5$. The sequence does not first uniformly fill a certain region in $[0,1]^5$, then move on the another region, etc. We see from Figure 7.7.1 in [8] for $[0,1]^2$ that the sequence of 1024 points wanders all around $[0,1]^2$ right from the beginning. For this reason we will assume that the finite sequence of QBGFNs $\overline{X}_k$, $N \geq 100,000$, sufficiently fills $\mathbf{Q}$ to produce a good approximate solution to the fuzzy optimization problem.

5.3.2 Search Space Ω for QBGFNs

Let $\mathbf{Q}_3$ = all $\overline{X} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$ for QBGFNs $\overline{X}_i \in [a_i, b_i]$, $i = 1, 2, 3$. Use our Sobol quasi-random number generator to produce a sequence $v_k = (h_{1k}, h_{2k}, h_{3k})$ in $[0,1]^{15}$ where $h_{1k} = (x_{1k}, ..., x_{5k})$, $h_{2k} = (x_{6k}, ..., x_{10,k})$, $h_{3k} = (x_{11,k}, ..., x_{15,k})$, $k = 1, 2, 3, ...$ Let V be all the v_k. V uniformly fills $[0,1]^{15}$. Define the mapping as above $\psi_i(h_{ik}) = \overline{X}_{ik}$, $i = 1, 2, 3$. Then $\psi = (\psi_1, \psi_2, \psi_3)$ maps V into $\mathbf{Q}_3$. We may argue as before that since V uniformly fills $[0,1]^{15}$, then $\psi(V)$ will uniformly fill $\mathbf{Q}_3$. Also, as before, we assume that a finite sequence with $N \geq 100,000$ will sufficiently fill the search space to give us a good approximate solution to a fuzzy optimization problem.

5.3.3 Search Space $[a, b]$ for TFNs

Let $\mathbf{Q}$ = all TFNs in $[a, b]$. We will randomly generate a sequence of TFNs $\overline{X}_k$, $k = 1, 2, 3, ...$, in $[a, b]$ to uniformly fill $\mathbf{Q}$. Notice at this point this is an infinite sequence and we will later discuss finite sequences to be used in applications. We first discuss how we produce the sequence $\overline{X}_k$.

We use our Sobol quasi-random number generator to produce the sequence $v_k = (x_{1k}, ..., x_{3k}) \in [0,1]^3$, $k = 1, 2, 3, ...$ Let $V = \{v_k | k = 1, 2, 3, ...\}$. For each v_k transform these three numbers into the interval $[a, b]$. Let $z_{1k} = a + (b-a)x_{1k}$, $z_{2k} = a + (b-a)x_{2k}$,$z_{3k} = a + (b-a)x_{3k}$ all in $[a, b]$. Next order the z_{ik} from smallest to largest giving $d_k = z_{2k} < e_k = z_{1k} < f_k = z_{3k}$. Then

$$u_k = (d_k, e_k, f_k) \tag{5.8}$$

defines a $\overline{X}_k$ in $\mathbf{Q}$. The construction of $\overline{X}_k$ is as follows: (1) the support (base) is $[d_k, f_k]$; and (2) the vertex is at $x = e_k$. This defines a mapping ψ of V into $\mathbf{Q}$. We know that V uniformly fills $[0,1]^3$. We now use this fact that V uniformly fills $[0,1]^3$ to argue that $\psi(V)$ will uniformly fill $\mathbf{Q}$.

Let $\mathbf{M} = \{(d, e, f) | d < e < f \in [a, b]\}$. Now each $m \in \mathbf{M}$ uniquely defines a $\overline{X} \in \mathbf{Q}$ and each $\overline{X} \in \mathbf{Q}$ uniquely specifies an $m \in \mathbf{M}$. In this sense $\mathbf{M} = \mathbf{Q}$.

Now equation (5.13) shows the mapping ψ of V into $\mathbf{M}$. We easily extend it to all of $[0,1]^3$. For any vector in $[0,1]^3$ we map the components into $[a, b]$, order them from smallest to largest, etc. and get a point in $\mathbf{M}$. So ψ maps $[0,1]^3$ onto $\mathbf{M}$. If V uniformly fills $[0,1]$ then surely $\psi(V)$ uniformly fills $\mathbf{M}$. If $\psi(V)$ uniformly fills $\mathbf{M}$ and $\mathbf{M} = \mathbf{Q}$ we have uniformly filled $\mathbf{Q}$.

Now assume we use a finite sequence v_k, $k = 1, 2, 3, ..., N$, in applications. We then plot these points in $[0,1]^3$. The sequence does not first uniformly fill a certain region in $[0,1]^3$, then move on the another region, etc. For this reason we will assume that the finite sequence of TFNs $\overline{X}_k$, $N \geq 100,000$, sufficiently fills $\mathbf{Q}$ to produce a good approximate solution to the fuzzy optimization problem.

5.3.4 Search Space Ω for TFNs

Let $\mathbf{Q}_3 = $ all $\overline{X} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$ for TFNs $\overline{X}_i \in [a_i, b_i]$, $i = 1, 2, 3$. Use our Sobol quasi-random number generator to produce a sequence $v_k = (h_{1k}, h_{2k}, h_{3k})$ in

$[0,1]^9$ where $h_{1k} = (x_{1k}, ..., x_{3k})$, $h_{2k} = (x_{4k}, ..., x_{6,k})$, $h_{3k} = (x_{7,k}, ..., x_{9,k})$, $k = 1, 2, 3, ...$ Let V be all the v_k. V uniformly fills $[0,1]^9$. Define the mapping as above $\psi_i(h_{ik}) = \overline{X}_{ik}$, $i = 1, 2, 3$. Then $\psi = (\psi_1, \psi_2, \psi_3)$ maps V into $\mathbf{Q}_3$. We may argue as before that since V uniformly fills $[0,1]^9$, then $\psi(V)$ will uniformly fill $\mathbf{Q}_3$. Also, as before, we assume that a finite sequence with $N \geq 100,000$ will sufficiently fill the search space to give us a good approximate solution to a fuzzy optimization problem.

5.3.5 Other Search Spaces

In Chapter 13 we need to uniformly fill $\mathbf{Q}_3$ = all $\overline{X} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$ for trapezoidal fuzzy numbers $\overline{X}_i \in [a_i, b_i]$, $i = 1, 2, 3$. Use our Sobol quasi-random number generator to produce a sequence $v_k = (h_{1k}, h_{2k}, h_{3k})$ in $[0,1]^{12}$ where $h_{1k} = (x_{1k}, ..., x_{4k})$, $h_{2k} = (x_{5k}, ..., x_{8,k})$, $h_{3k} = (x_{9,k}, ..., x_{12,k})$, $k = 1, 2, 3, ...$ To get $\overline{X}_{1k}$ we first map the points in h_{1k} into $[a_1, b_1]$, order them from smallest to largest, use the first and last for the base and the middle two for the core. Let V be all the v_k. V uniformly fills $[0,1]^{12}$. Define the mapping as above $\psi_i(h_{ik}) = \overline{X}_{ik}$, $i = 1, 2, 3$. Then $\psi = (\psi_1, \psi_2, \psi_3)$ maps V into $\mathbf{Q}_3$. We may argue as before that since V uniformly fills $[0,1]^{12}$, then $\psi(V)$ will uniformly fill $\mathbf{Q}_3$. Also, as before, we assume that a finite sequence with $N \geq 100,000$ will sufficiently fill the search space to give us a good approximate solution to a fuzzy optimization problem.

In Chapter 14 we just need random sequences of crisp vectors. We use our quasi-random number generator to get these sequences.

References

1. Bratley, P., Fox, B.L., Niederreiter, H.: Implementation and tests of low-discrepancy sequences. ACM Trans. Model. Comput. Simul. 2, 195–213 (1992)
2. Gentle, J.E.: Random Number Generation and Monte Carlo Methods, 2nd edn. Springer, Berlin (2003)
3. Gibbons, J.D., Chakraborti, S.: Nonparametric Statistical Inference, 3rd edn. Marcel Dekker, New York (1992)
4. Hogg, R.V., Tanis, E.A.: Probability and Statistical Inference, 6th edn. Prentice-Hall, Upper Saddle River, NJ (2001)
5. Knuth, D.E.: The Art of Computer Programming, 2nd edn., vol. 2. Addison-Wesley, Reading, Mass. (1981)
6. Kocis, L., Whiten, W.J.: Computational investigations of low-discrepancy sequences. ACM Trans. Math. Softw. 23, 266–294 (1997)
7. Lindgren, B.W.: Statistical Theory, 3rd edn. Macmillan, New York (1976)
8. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C: The Art of Scientific Computing, 2nd edn. Cambridge Univ. Press, Cambridge (2002)

Part II: Applications

6 Fuzzy Monte Carlo Method

6.1 Introduction

This chapter introduces our fuzzy Monte Carlo method. We will be working with a very simple linear programming problem. The crisp linear program is presented in the next section. Then we fuzzify the linear program in the third section. We make some of the parameters in the problem triangular fuzzy numbers and allow all the variables to be triangular shaped fuzzy numbers. We will need to decide on a definition of $\leq$ between fuzzy numbers and we will use Kerre's method (Section 2.6.2 of Chapter 2) first and then Chen's method (Section 2.6.3 of Chapter 2) second. This chapter, and Chapters 7 and 8, are based on ([5],[6]), see also ([3],[4]).

Fuzzy linear programming has become a very large area of research. Put "fuzzy linear programming" into your search engine and obtain over $17,000$ web sites to visit. Obviously we can not search all of these sites. A few recent references to this topic are the papers ([10]-[16],[18],[20],[22]-[25],[27]) and books (or articles in these books) ([1],[2],[7]-[9],[17],[19],[21],[26]).

6.2 Crisp Linear Program

Consider the optimization problem

$$\max Z = (2x_1 + 3x_2), \tag{6.1}$$

subject to

$$x_1 + 2x_2 \leq 6, \tag{6.2}$$

$$2x_1 + x_2 \leq 6, \tag{6.3}$$

$$0 \leq x_1, x_2 \leq M, \tag{6.4}$$

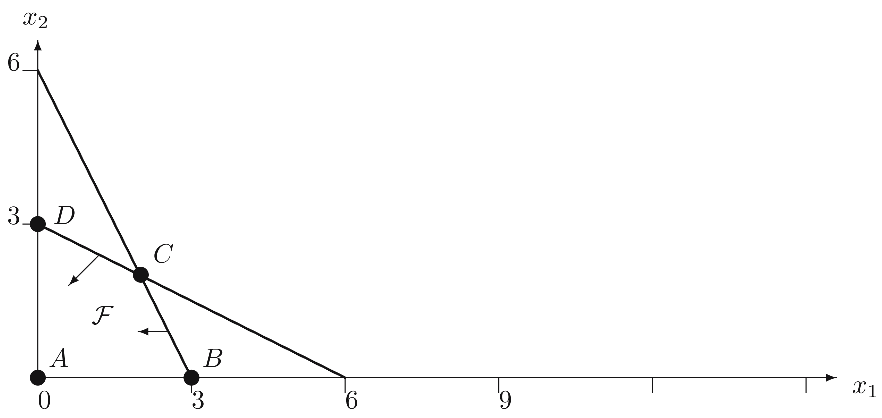


Fig. 6.1. Linear Programming Problem

Table 6.1. Solution to the Linear Program

Vertex	Coordinates	Objective Function
A	$(0, 0)$	$Z = 0.0$
B	$(3, 0)$	$Z = 6$
C	$(2, 2)$	$Z = 10$
D	$(0, 3)$	$Z = 9$

for some positive constant M. The positive constant M is for later on in the chapter and we will not get to use it in this section. Figure 6.1 shows the constraints and the feasible set $\mathcal{F}$. We know that the optimal solution will be at a vertex point of the feasible set. The values of the objective function at these vertex points is shown in Table 6.1 and we see that $\max Z = 10$ at $x_1 = x_2 = 2$.

6.3 Fuzzy Linear Program

Now we allow the parameters in the objective function and the constants on the right side of the inequalities to be fuzzy. The fuzzy linear program is

$$\max \overline{Z} = (\overline{C}_1 \overline{X}_1 + \overline{C}_2 \overline{X}_2), \tag{6.5}$$

subject to

$$\overline{X}_1 + 2\overline{X}_2 \leq \overline{B}_1, \tag{6.6}$$

$$2\overline{X}_1 + \overline{X}_2 \leq \overline{B}_2, \tag{6.7}$$

$$0 \leq \overline{X}_1, \overline{X}_2 \leq M, \tag{6.8}$$

where $\overline{C}_1 = (1/2/3)$, $\overline{C}_2 = (2/3/4)$, $\overline{B}_1 = (5/6/7)$, $\overline{B}_2 = (5/6/7)$ and $\overline{X}_1 \approx (x_{11}/x_{12}/x_{13})$, $\overline{X}_2 \approx (x_{21}/x_{22}/x_{23})$. The fuzzy parameters are all triangular fuzzy numbers but the variables will be Bézier (quadratic) fuzzy numbers (QBGFNs in Chapter 4).

Now we will look at two cases for evaluating $\leq$ between fuzzy numbers. Both of these methods are needed in the next two chapters, but they will not be used after Chapter 8. The first is Kerre's method from Section 2.6.2 in Chapter 2.

6.3.1 Kerre's Method

We will randomly generate, from Chapter 4, vectors $\overline{V} = (\overline{X}_1, \overline{X}_2)$ and first check to see if they satisfy equations (6.6) and (6.7) using Kerre's $\leq$. If these equations are satisfied, then $\overline{V}$ is feasible and we evaluate $\overline{Z} = \overline{C}_1\overline{X}_1 + \overline{C}_2\overline{X}_2$. Let the previous best (max) value of $\overline{Z}$ be $\overline{Z}^*$ and the current value of $\overline{Z} = \overline{Z}_0$ from the recent feasible $\overline{V}$. If $\overline{Z}^* < \overline{Z}_0$, then set $\overline{Z}^*$ to be $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and generate the next random $\overline{V}$. We are looking for an optimal solution and not all the $\overline{V}$ that produce the best $\overline{Z}$ value.

Next we need to determine intervals $I_i = [0, M_i]$, $M_i > 0$, $i = 1, 2$, for the $\overline{X}_i$, $i = 1, 2$, respectively. A good selection of these intervals will make the fuzzy Monte Carlo process more efficient. If an interval is too big, then too many $\overline{V}$ will be rejected as not being feasible. If an interval is too small we can miss the optimal solution. There is no natural upper bound on x_{13} (x_{23}) so that $\overline{V}$ is feasible. See Figure 6.2. Also see Figure 2.7 in Chapter 2. Here $\overline{E} \approx (e_1/e_2/e_3)$ represents $\overline{X}_1 + 2\overline{X}_2$ or $2\overline{X}_1 + \overline{X}_2$ and let $\overline{B} = (0.5/1.5/2.5)$. We changed $\overline{B}$ from $(5/6/7)$ to this value for this figure. Then $e_3 = x_{13} + 2x_{23}$ or $2x_{13} + x_{23}$. In Figure 6.2 $\overline{E} \approx (0/1/5)$. We see that $d(\overline{E}, \overline{max})$ is the area of regions A_1 and A_2 and $d(\overline{B}, \overline{max})$ is the area of A_3. Since $area(A_3) < area(A_1) + area(A_2)$ we get $\overline{E} < \overline{B}$ and $\overline{V}$ is feasible even as e_3 grows larger and larger. In practical problems there is going to be an upper bound for the variables which will produce an upper bound for e_3. Management will decide on practical upper bounds for the x_i giving the upper bounds for the M_i in the intervals I_i. Sometimes the optimization problem will dictate the upper bounds, but in this case we get them for experts familiar with the problem. Let us assume that $I_i = [0, 5]$, $i = 1, 2$, which implies that $\overline{E} < 15$.

So we now randomly generate a sequence $\overline{V}_k = (\overline{X}_{1k}, \overline{X}_{2k})$ with $\overline{X}_{ik} \in [0, 5]$ all i and all k. Using our Sobol quasi-random number generator we produce sequences of random vectors $v_{1k} = (x_{1k1}, ..., x_{1k5})$, $v_{2k} = (x_{2k1}, ..., x_{2k5})$, $k = 1, 2, 3, ...$ The sequence v_{1k} is used to get the sequence of quadratic fuzzy numbers $\overline{X}_{1k}$, recall that we only require vectors of length five for these fuzzy numbers (see Chapter 4), and the other sequence v_{2k} constructs the sequence of quadratic fuzzy numbers $\overline{X}_{2k}$, $k = 1, 2, 3, ...$ However, because we use vectors $\overline{V} = (\overline{X}_1, \overline{X}_2)$ we choose our stream of quasi-random numbers generated 10 at a time to get the two pairs of 5. If $\overline{V}_k$ is feasible we compute

$$\overline{Z}_k = \overline{C}_1\overline{X}_{1k} + \overline{C}_2\overline{X}_{2k}, \tag{6.9}$$

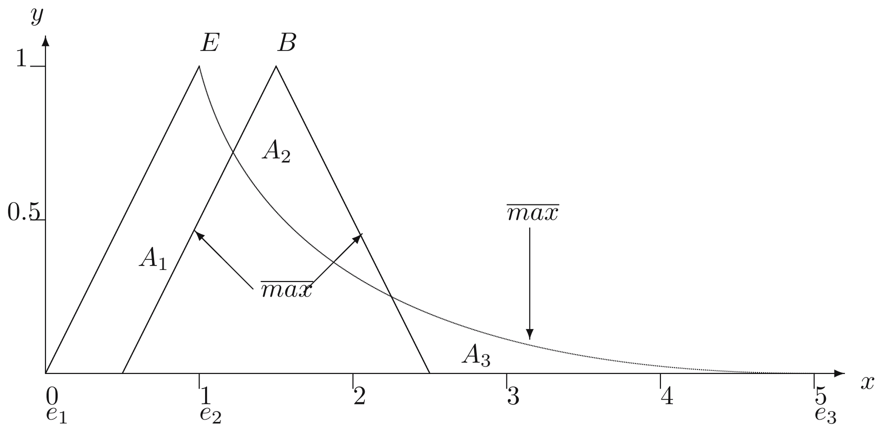


Fig. 6.2. Finding the Intervals for the $\overline{X}_i$ Using Kerre's Method

for $k = 1, 2, 3, ..., N$, where N is the predetermined total number of iterations. We want to find a k value, and hence a $\overline{V}_k$, to solve

$$max\{\overline{Z}_k | k = 1, 2, 3, ..., N\}. \tag{6.10}$$

With $N = 100,000$ pairs of QBGFNs (v_{1k},v_{2k}), the results of the fuzzy Monte Carlo method are shown in Table 6.2, and Figures 6.3 & 6.4. All the fuzzy numbers in Table 6.2 are triangular shaped fuzzy numbers. $\overline{X}_1$ and $\overline{X}_2$ are QBGFNs. The notation we use for these fuzzy numbers was explained in Section 4.3.2. We define a QBGFN as (a, b, c, d, e) where: (1) the support is the interval $[a, c]$; (2) the vertex is at $x = b$; (3) the three numbers a, d, b define the quadratic function for the left side of the fuzzy number; and (4) the three numbers b, e, c specify the quadratic function for the right side of the fuzzy number. Since $max\overline{Z}$ is not necessarily, or likely to be a QBGFN, we only give the support and core for $max\overline{Z}$. Our approximate solution to this fuzzy linear program are the fuzzy numbers determined by this Monte Carlo program. In the following three chapters we may have another solution, using an evolutionary algorithm, to compare to our fuzzy Monte Carlo solution.

Table 6.2. Monte Carlo Solution to the Fuzzy Linear Program, Kerre's Method, QBGFNs, N=100,000

type random	$max\overline{Z}$	$\overline{X}_i$
Sobol	$\approx (2.70/8.67/33.74)$	$\overline{X}_1 = (1.28, 1.49, 4.97, -0.42, 3.61)$
		$\overline{X}_2 = (0.71, 1.90, 4.70, -0.32, 2.69)$

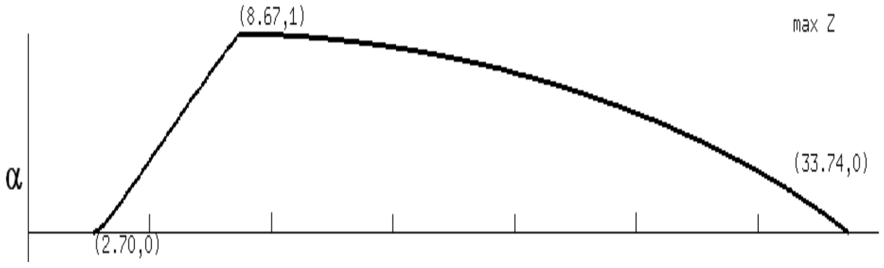


Fig. 6.3. Monte Carlo Solution to the Fuzzy Linear Program, $max\overline{Z}$, Kerre's Method, QBGFNs, N=100,000

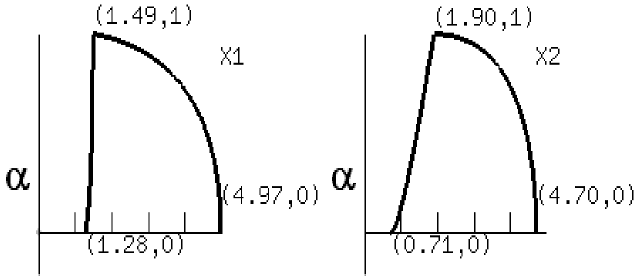


Fig. 6.4. Monte Carlo Solution to the Fuzzy Linear Program, $\overline{X}_1$ & $\overline{X}_2$ Kerre's Method, QBGFNs, N=100,000

6.3.2 Chen's Method

We randomly generate vectors $\overline{V} = (\overline{X}_1, \overline{X}_2)$ and check to see if they satisfy equations (6.6) and (6.7) using Chen's $\leq$. If these equations are satisfied, then $\overline{V}$ is feasible and we evaluate $\overline{Z}$. Let the previous best (max) value of $\overline{Z}$ be $\overline{Z}^*$ and the current value of $\overline{Z} = \overline{Z}_0$ from the recent feasible $\overline{V}$. If $\overline{Z}^* < \overline{Z}_0$ using Chen's $<$, then set $\overline{Z}^*$ to be $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and generate the next random $\overline{V}$.

Now we need to determine intervals $I_i = [0, M_i]$, $M_i > 0$, $i = 1, 2$, for the $\overline{X}_i$, $i = 1, 2$, respectively. There is no natural upper bound on x_{13} (x_{23}) so that $\overline{V}$ is feasible. See Figure 6.5. Also see Figure 2.8 in Chapter 2. Here $\overline{E} \approx (e_1/e_2/e_3)$ represents $\overline{X}_1 + 2\overline{X}_2$ or $2\overline{X}_1 + \overline{X}_2$ and let $\overline{B} = (0.5/1.5/2.5)$. We changed $\overline{B}$ from (5/6/7) to this value for this figure. Then $e_3 = x_{13} + 2x_{23}$ or $2x_{13} + x_{23}$. In Figure 6.5 $\overline{E} \approx (0/1/5)$. Consulting Figures 2.8 and 6.5 we see that the y coordinate at: (1) L_E is 0.8; (2) L_B is 0.7; (3) R_E is 0.3; and (4) R_B is 0.4. So, from equation (2.53) in Chapter 2 we compute

$$\mu_T(\overline{E}) = 0.5(0.3 + (1 - 0.8)) = 0.25, \tag{6.11}$$

and

$$\mu_T(\overline{B}) = 0.5(0.4 + (1 - 0.7)) = 0.35, \tag{6.12}$$

and

$$\mu_T(\overline{E}) < \mu_T(\overline{B}), \tag{6.13}$$

implying that $\overline{E} < \overline{B}$ and $\overline{V}$ is feasible even as e_3 grows larger and larger. In practical problems there is going to be an upper bound for the variables which will produce an upper bound for e_3. Management will decide on practical upper bounds for the x_i giving the upper bounds for the M_i in the intervals I_i. Sometimes the optimization problem will dictate the upper bounds, but in this case we get them for experts familiar with the problem. Let us assume that $I_i = [0, 5]$, $i = 1, 2$, so that $\overline{E} < 15$.

So we now randomly generate a sequence $\overline{V}_k = (\overline{X}_{1k}, \overline{X}_{2k})$ with $\overline{X}_{ik} \in [0, 5]$ all i and all k. Using our Sobol quasi-random number generator we produce sequences of random vectors $v_{1k} = (x_{1k1}, ..., x_{1k5})$, $v_{2k} = (x_{2k1}, ..., x_{2k5})$, $k = 1, 2, 3, ...$ The sequence v_{1k} is used to get the sequence of quadratic fuzzy numbers $\overline{X}_{1k}$ and the other sequence v_{2k} constructs the sequence of quadratic fuzzy numbers $\overline{X}_{2k}$, $k = 1, 2, 3, ...$ However, because we use vectors $\overline{V} = (\overline{X}_1, \overline{X}_2)$ we choose our stream of Sobol quasi-random numbers generated 10 at a time to get the two pairs of 5. If $\overline{V}_k$ is feasible we compute

$$\overline{Z}_k = \overline{C}_1\overline{X}_{1k} + \overline{C}_2\overline{X}_{2k}, \tag{6.14}$$

for $k = 1, 2, 3, ..., N$, where N is the predetermined total number of iterations. We want to find a k value, and hence a $\overline{V}_k$, to solve

$$max\{\overline{Z}_k | k = 1, 2, 3, ..., N\}. \tag{6.15}$$

With $N = 100,000$ the results of the fuzzy Monte Carlo method are shown in Table 6.3, and Figures 6.6 & 6.7. All the fuzzy numbers in Table 6.3 are triangular shaped fuzzy numbers. $\overline{X}_1$ and $\overline{X}_2$ are QBGFNs. The notation we use for these fuzzy numbers was explained above and in Section 4.3.2. Since $max\overline{Z}$ is not necessarily or likely to be a QBGFN, we only give the support

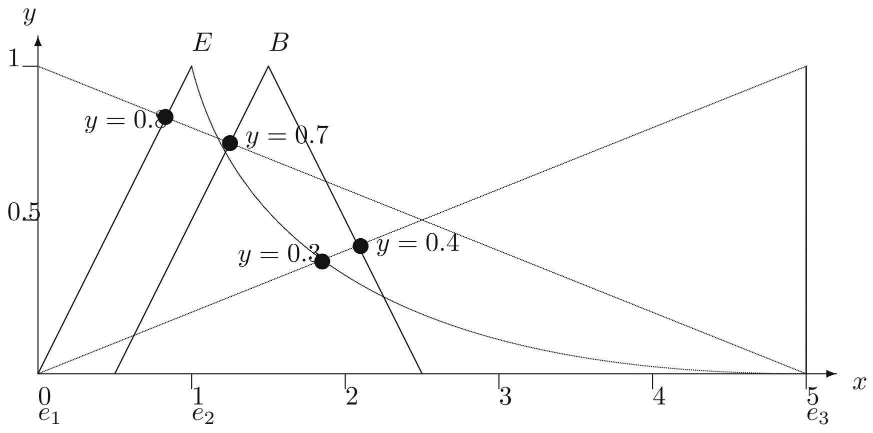


Fig. 6.5. Finding the Intervals for the $\overline{X}_i$ Using Chen's Method

Table 6.3. Monte Carlo Solution to the Fuzzy Linear Program, Chen's Method, QBGFNs, N=100,000

type random	$max\overline{Z}$	$\overline{X}_i$
Sobol	$\approx (1.37/5.07/34.84)$	$\overline{X}_1 = (0.08, 0.50, 4.99, -0.96, 4.69)$ $\overline{X}_2 = (0.64, 1.36, 4.97, 0.51, 2.72)$

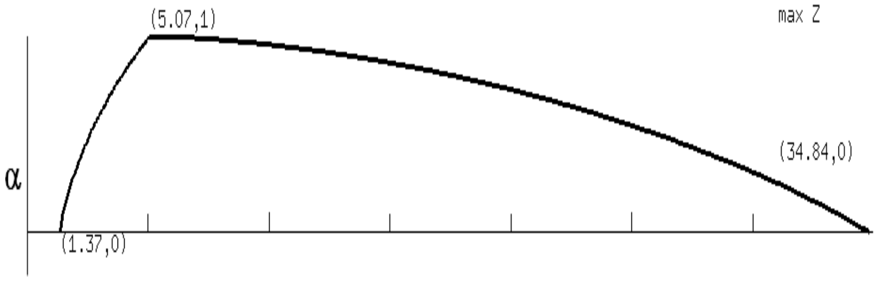


Fig. 6.6. Monte Carlo Solution to the Fuzzy Linear Program, $max\overline{Z}$, Chen's Method, QBGFNs, N=100,000

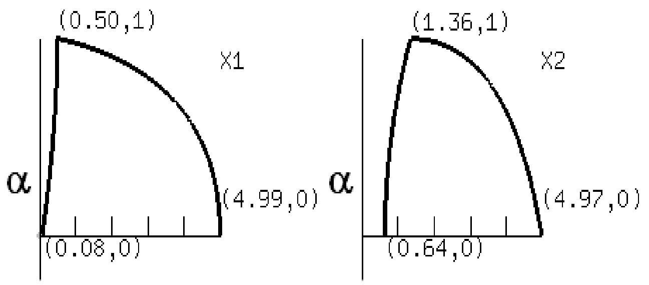


Fig. 6.7. Monte Carlo Solution to the Fuzzy Linear Program, $\overline{X}_1$ & $\overline{X}_2$ Chen's Method, QBGFNs, N=100,000

and core for $max\overline{Z}$. Our approximate solution to this fuzzy linear program are the fuzzy numbers determined by this Monte Carlo program. In the following three chapters we may have another solution, using an evolutionary algorithm, to compare to our fuzzy Monte Carlo solution.

6.3.3 Comparison of Solutions

All of these software efforts were performed on Windows-based PCs. For these fuzzy Monte Carlo optimizations, several computers were used, all Dell Optiplex

GX270's, 3.0GHz, 1GB RAM. The Kerre/Sobol running time was 2:27:50; the Chen/Sobol finished in 2:31:10.

The Kerre comparison method found $2,864$ feasible sets in a stream of $100,000$ $(\overline{X}_{1k},\overline{X}_{2k})$; 15 of them had triggered new maximums. The Chen comparison method found $17,047$ feasible sets in the same stream of $100,000$ $(\overline{X}_{1k},\overline{X}_{2k})$; 12 of them had triggered new maximums. The 341^{st} $(\overline{X}_{1k},\overline{X}_{2k})$ was a new maximum by both methods; the 4^{th} new maximum under Kerre, the 6^{th} new maximum under Chen. The last maximum by the Kerre method was found at the $15,251^{st}$ $(\overline{X}_{1k},\overline{X}_{2k})$ (it was the $2,864^{th}$, the last, feasible set). The last maximum by the Chen method was found at the $48,798^{th}$ $(\overline{X}_{1k},\overline{X}_{2k})$ (the $8,322^{nd}$ feasible set).

For our Monte Carlo solution using Sobol quasi-random numbers, we compare Kerre's method results and Chen's method results to find that the $max\overline{Z}$ from Kerre's method is greater than the $max\overline{Z}$ from Chen's method solution (regardless which of Buckley's method, Kerre's method, or Chen's method is used to compare the maximums). It appears that Kerre's method produced a solution closest to the crisp solution $x_1 = x_2 = 2$, $maxZ = 10$.

References

1. Aliev, R.A., Fazlollahi, B.: Soft Computing and its Applications to Business and Economics. Springer, Heidelberg (2004)
2. Bector, C.R., Chandra, S.: Fuzzy Mathematical Programming and Fuzzy Matrix Games. Springer, Heidelberg (2005)
3. Buckley, J.J.: Joint Solution to Fuzzy Linear Programming. Fuzzy Sets and Systems 72, 215–220 (1995)
4. Buckley, J.J., Feuring, T., Hayashi, Y.: Neural Net Solution to Fuzzy Linear Programming. Fuzzy Sets and Systems 106, 99–111 (1999)
5. Buckley, J.J., Feuring, T.: Evolutionary Algorithm Solution to Fuzzy Problems: Fuzzy Linear Programming. Fuzzy Sets and Systems 109, 35–53 (2000)
6. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
7. Carlsson, C., Fuller, R.: Fuzzy Reasoning in Decision Making and Optimization. Physica-Verlag, Heidelberg (2002)
8. Ehrgott, M., Gandibleux, X.: Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys. Kluwer Academic Press, Norwell, Mass. (2002)
9. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization. Wiley Interscience, N.Y. (1999)
10. Inuiguchi, M., Ramik, J., Tanino, T., Vlach, M.: Satisficing Solutions and Duality in Interval and Fuzzy Linear Programming. Fuzzy Sets and Systems 135, 151–177 (2003)
11. Jamison, K.D., Lodwick, W.A.: Fuzzy Linear Programming Using a Penalty Method. Fuzzy Sets and Systems 119, 97–110 (2001)
12. Li, D., Cheng, C.: Fuzzy Multiobjective Programming Methods for Fuzzy Constrained Matrix Games with Fuzzy Numbers. Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems 10, 385–400 (2002)
13. Li, D.-F., Yang, J.-B.: Fuzzy Linear Programming Technique for Multiattribute Group Decision Making in Fuzzy Environments. Information Sciences 158, 263–275 (2004)

14. Liu, X.: Measuring the Satisfaction of Constraints in Fuzzy Linear Programming. Fuzzy Sets and Systems 122, 263–275 (2001)
15. Maeda, T.: Fuzzy Linear Programming Problems as Bi-Criteria Optimization Problems. Applied Mathematics and Computation 120, 109–121 (2001)
16. Nishizaki, I., Sakawa, M.: Solutions Based on Fuzzy Goals in Fuzzy Linear Programming Games. Fuzzy Sets and Systems 115, 105–119 (2000)
17. Nishizaki, I., Sakawa, M.: Fuzzy and Multiobjective Games for Conflict Resolution. Physica-Verlag, Heidelberg (2001)
18. Rommelfanger, H.: Fuzzy Linear Programming and Applications. European J. Operational Research 92, 512–527 (1996)
19. Sakawa, M.: Genetic Algorithms and Fuzzy Multiobjective Optimization. Kluwer Academic Press, Norwell, Mass. (2002)
20. Tang, J., Wang, D., Fung, R.Y.K., Yung, K.-L.: Understanding of Fuzzy Optimization: Theories and Methods. J. System Science and Complexity 17, 117–136 (2004)
21. Trzaskalik, T., Michnik, J. (eds.): Multiple Objective and Goal Programming. Physica-Verlag, Heidelberg (2002)
22. Van Hop, N.: Solving Fuzzy (Stochastic) Linear Programming Problems Using Superiority and Inferiority Measures. Information Sciences 177, 1977–1991 (2007)
23. Vasant, P.M.: Application of Fuzzy Linear Programming in Production Planning. Fuzzy Optimization and Decision Making 2, 229–241 (2003)
24. Wang, R.-C., Fang, H.-H.: Aggregate Production Planning with Multiple Objectives in a Fuzzy Environment. European J. Operational Research 133, 521–536 (2001)
25. Wu, H.-C.: Duality Theory in Fuzzy Linear Programming Problems with Fuzzy Coefficients. Fuzzy Optimization and Decision Making 2, 61–73 (2003)
26. Yoshida, Y. (ed.): Dynamical Aspects in Fuzzy Decision Making. Physica-Verlag, Heidelberg (2001)
27. Zhang, G., Wu, Y.-H., Remias, M., Lu, J.: Formulation of Fuzzy Linear Programming Problems as Four-Objective Constrained Optimization Problems. Applied Mathematics and Computation 139, 383–399 (2003)

7 Fully Fuzzified Linear Programming I

7.1 Introduction

We first discuss the general fully fuzzified linear program in the next section. Then we study a max problem, the product mix problem, in Section 7.3. We have previously obtained an approximate fuzzy solution to this problem using an evolutionary algorithm ([1],[2]). In Section 7.4 we will apply our fuzzy Monte Carlo method to the problem to generate another approximate solution and then compare these new results to the evolutionary algorithm method.

7.2 Fully Fuzzified Linear Programming

Fuzzy linear programming has long been an area of application of fuzzy sets. Consider the classical linear program

$$\begin{aligned} & \max/\min Z = c_1x_1 + \cdots + c_nx_n \\ \text{subject to:} & \\ & a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i, \quad 1 \leq i \leq m \\ & x_i \geq 0, \text{for all } i. \end{aligned} \tag{7.1}$$

We need to have values for all the parameters c_i, a_{ij} and b_i to completely specify the optimization problem. Many of these must be estimated and are therefore uncertain. It is then natural to model these uncertain parameters using fuzzy numbers. The problem then becomes a fuzzy linear programming problem.

We are going to allow all the parameters to be fuzzy and we obtain what we have called the fully fuzzified linear programming problem. Many researchers (see the references to Chapter 6) have looked at parts of this problem: (1) the a_{ij} and b_i can be fuzzy; or (but not both) (2) the c_i can be fuzzy. The fully fuzzified (max) linear program is

$$\begin{aligned} max\overline{Z} = \overline{C}_1\overline{X}_1 + \cdots + \overline{C}_n\overline{X}_n \\ \text{subject to:} \\ \overline{A}_{i1}\overline{X}_1 + \cdots + \overline{A}_{in}\overline{X}_n \leq \overline{B}_i,\ 1 \leq i \leq m, \\ \overline{X}_i \geq 0, \text{for all i.} \end{aligned} \tag{7.2}$$

where the $\overline{C}_i$, $\overline{A}_{ij}$ and $\overline{B}_i$ can all be triangular fuzzy numbers. Not every single parameter need be fuzzy; but we shall assume that some of the $\overline{C}_i$ are fuzzy and some of the $\overline{A}_{ij}$ and $\overline{B}_i$ are fuzzy. Since the parameters are fuzzy, the variables $\overline{X}_i$ will be triangular shaped fuzzy numbers.

We are now concerned with solving the optimization problem in equation (7.2). But first we must do two things: (1) explain what we mean by max/min $\overline{Z}$ since $\overline{Z}$ will also be a triangular shaped fuzzy number; and (2) decide on how we will evaluate the inequality ($\leq$) between fuzzy numbers $\overline{E}_i \leq \overline{B}_i$, where $\overline{E}_i = \overline{A}_{i1}\ \overline{X}_1 + \cdots + \overline{A}_{in}\ \overline{X}_n$.

In our previous research on this topic we handled max/min $\overline{Z}$ as discussed in Section 2.5 of Chapter 2. Also, in those publications we used both Kerre's Method (Section 2.6.2) and Chen's Method (Section 2.6.3) to evaluate $\leq$ between fuzzy numbers. In our fuzzy Monte Carlo method we will use both Kerre's Method and Chen's Method to evaluate $\leq$ and $<$ between fuzzy numbers.

7.3 Product Mix Problem

A company produces three products P_1, P_2 and P_3 each of which must be processed through three departments D_1, D_2 and D_3. The approximate time, in hours, each P_i spends in each D_j is given in Table 7.1.

Table 7.1. Approximate Times Product P_i is in Department D_j

		Department		
		D_1	D_2	D_3
Product	P_1	6	12	2
	P_2	8	8	4
	P_3	3	6	1

Each department has only so much time available each week. These times can vary slightly from week to week so the following numbers are estimates of the maximum time available per week, in hours, for each department: (1) for D_1 288 hours; (2) 312 hours for D_2; and (3) D_3 has 124 hours. Finally, the selling price for each product can vary a little due to small discounts to certain customers but we have the following average selling prices: (1) \$6 per unit for P_1; (2) \$8 per unit for P_2, and (3) \$6 per unit for P_3. The company wants to determine the number of units to produce for each product per week to maximize its revenue.

Since all the numbers given are uncertain, we will model the problem as a fully fuzzified linear program. We substitute a triangular fuzzy number for each value where the peak of the fuzzy number is at the number given. So, we have the following fully fuzzified linear program

$$\max \overline{Z} = (5.8/6/6.2)\overline{X}_1 + (7.5/8/8.5)\overline{X}_2 + (5.6/6/6.4)\overline{X}_3 \tag{7.3}$$

$$\text{subject to:} \tag{7.4}$$

$$(5.6/6/6.4)\overline{X}_1 + (7.5/8/8.5)\overline{X}_2 + (2.8/3/3.2)\overline{X}_3 \leq (283/288/293),$$
$$(11.4/12/12.6)\overline{X}_1 + (7.6/8/8.4)\overline{X}_2 + (5.7/6/6.3)\overline{X}_3 \leq (306/312/318),$$
$$(1.8/2/2.2)\overline{X}_1 + (3.8/4/4.2)\overline{X}_2 + (0.9/1/1.1)\overline{X}_3 \leq (121/124/127),$$
$$\overline{X}_1, \overline{X}_2, \overline{X}_3 \geq 0,$$

where the $\overline{X}_i$ are triangular shaped fuzzy numbers for the amount to produce for P_i per week.

7.4 Fuzzy Monte Carlo Method

We need to find intervals $I_i = [0, M_i]$, $i = 1, 2, 3$, as explained in Section 6.3 in Chapter 6, for the $\overline{X}_i$. Since we form fuzzy vector $\overline{V} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$, and each $\overline{X}_i$ consumes 5 crisp numbers, we choose our stream of Sobol quasi-random numbers which had been generated 15 at a time to get three sets of 5. We randomly generate $\overline{X}_i \in [0, M_i]$, $i = 1, 2, 3$, and form the random fuzzy vector $\overline{V} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$. We test to see if $\overline{V}$ is feasible, or the $\overline{X}_i$ satisfy the constraints. Assuming that $\overline{V}$ is feasible we compute $\overline{Z}_0 = \overline{C}_1\overline{X}_1 + ... + \overline{C}_3\overline{X}_3$. If $\overline{Z}^*$ is the current best (max) value of $\overline{Z}$ then we replace $\overline{Z}^*$ with $\overline{Z}_0$ if $\overline{Z}^* < \overline{Z}_0$, otherwise we discard $\overline{Z}_0$.

We also need to solve the fuzzy max problem twice. First we use Kerre's Method to evaluate $\overline{E}_i \leq \overline{B}_i$ in the constraints and $\overline{Z}^* < \overline{Z}_0$ in the objective function. Then we use Chen's Method.

To get an idea for the intervals I_i for the $\overline{X}_i$, $i = 1, 2, 3$, we studied the solutions to this problem from the evolutionary algorithm, and we studied our constraint equations. Since our constraint equations are '$\leq$' inequalities, we may set any two of our $\overline{X}_i$ to be zero to determine the maximum possible value for the third fuzzy variable. Then we use the minimum of these maximums to determine a possible support interval for feasible sets. For example, set $\overline{X}_2 = \overline{X}_3 = 0$, and let $\overline{X}_1 \approx (x_{11}/x_{12}/x_{13})$. After multiplying by the fuzzy coefficients we solve each $6x_{12} \leq 293$, $12x_{12} \leq 318$, $2x_{12} \leq 127$ for x_{12} and take the minimum. The result is 26.5 and we take the interval $[0, 26.5]$ for $\overline{X}_1$. Similarly we obtain $[0, 31.8]$ for $\overline{X}_2$ and $[0, 53]$ for $\overline{X}_3$.

Now we follow the procedure outlined in Sections 6.3.1 and 6.3.2.

7.4.1 Kerre's Method

We randomly generate vectors $\overline{V} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$, where the $\overline{X}_{ik}$ are Bézier (quadratic) fuzzy numbers (QBGFNs in Chapter 4), and check to see if they

Table 7.2. Monte Carlo Solution to the Fuzzy Linear Program, Kerre's $\leq$, N=100,000

$max\overline{Z}$	$\overline{X}_i$ (QBGFNs from Sobol)
$\approx (168.02/260.40/591.80)$	$\overline{X}_1 = (0.10, 2.75, 12.41, 0.57, -10.55)$
	$\overline{X}_2 = (22.21, 24.30, 26.98, 0.92, 0.80)$
	$\overline{X}_3 = (0.15, 8.25, 44.61, -5.30, -31.23)$

satisfy constraint equations (7.4) using Kerre's $\leq$. We wish to solve the optimization problem given in equation (7.3). If these equations are satisfied, then $\overline{V}$ is feasible and we evaluate $\overline{Z}$, the fuzzy objective function in equation (7.3). Let the previous best (max) value of $\overline{Z}$ be $\overline{Z}^*$ and the current value of $\overline{Z} = \overline{Z}_0$ from the recent feasible $\overline{V}$. If $\overline{Z}^* < \overline{Z}_0$ using Kerre's $<$, then set $\overline{Z}^*$ to be $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and generate the next random $\overline{V}$.

With $N = 100,000$ the results of the fuzzy Monte Carlo method are shown in Table 7.2, and Figures 7.1 & 7.2. All the fuzzy numbers in Table 7.2 are triangular shaped fuzzy numbers. $\overline{X}_1$, $\overline{X}_2$ and $\overline{X}_3$ are QBGFNs. The notation we use for these fuzzy numbers was explained in Section 4.3.2. and also in Section 6.3.1. Since $max\overline{Z}$ is not necessarily or likely to be a QBGFN, we only give the support and core for $max\overline{Z}$. Our approximate solution to this fuzzy linear program are the fuzzy numbers determined by this Monte Carlo program.

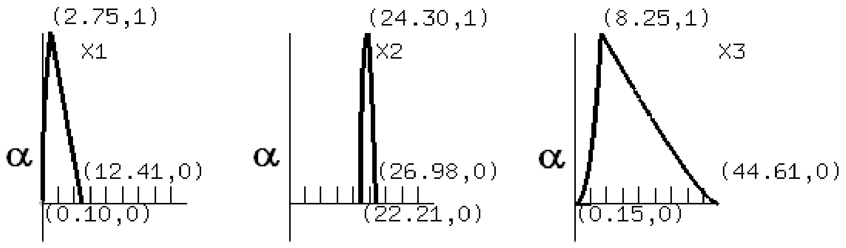


Fig. 7.1. $\overline{X}_1, \overline{X}_2, \overline{X}_3$ Solution using Kerre's $\leq$, Product Mix Problem

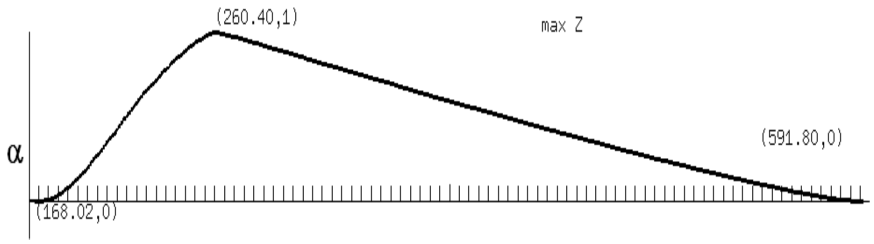


Fig. 7.2. $max\overline{Z}$ using Kerre's $\leq$, Product Mix Problem

Figure 7.1 show the "optimal" $\overline{X}_i$. Figure 7.2 shows the value of the objective function $\overline{Z}$, corresponding to the values of the $\overline{X}_i$ given in Figure 7.1.

7.4.2 Chen's Method

Now we randomly generate vectors $\overline{V} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$, where the $\overline{X}_{ik}$ are Bézier (quadratic) fuzzy numbers (QBGFNs in Chapter 4), and check to see if they satisfy constraint equations (7.4) using Chen's $\leq$. We wish to solve the optimization problem given in equation (7.3). If these equations are satisfied, then $\overline{V}$ is feasible and we evaluate $\overline{Z}$, the fuzzy objective function in equation (7.3). Let the previous best (max) value of $\overline{Z}$ be $\overline{Z}^*$ and the current value of $\overline{Z} = \overline{Z}_0$ from the recent feasible $\overline{V}$. If $\overline{Z}^* < \overline{Z}_0$ using Chen's $<$, then set $\overline{Z}^*$ to be $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and generate the next random $\overline{V}$.

With $N = 100,000$ the results of the fuzzy Monte Carlo method are shown in Table 7.3, and Figures 7.3 & 7.4. All the fuzzy numbers in Table 7.3 are triangular shaped fuzzy numbers. $\overline{X}_1$, $\overline{X}_2$ and $\overline{X}_3$ are QBGFNs. The notation we use for these fuzzy numbers was explained in Section 4.3.2. and also in Section 6.3.1. Since $max\overline{Z}$ is not necessarily or likely to be a QBGFN, we only give the support and core for $max\overline{Z}$. Our approximate solution to this fuzzy linear program are the fuzzy numbers determined by this Monte Carlo program.

Table 7.3. Monte Carlo Solution to the Fuzzy Linear Program, Chen's $\leq$, N=100,000

$max\overline{Z}$	$\overline{X}_i$ (QBGFN from Sobol)
$\approx (285.30/286.20/287.10)$	$\overline{X}_1 = (0.00, 0.00, 0.00, -1.00, -0.00)$
	$\overline{X}_2 = (15.90, 15.90, 15.90, -0.00, -0.00)$
	$\overline{X}_3 = (26.50, 26.50, 26.50, -0.00, -0.00)$

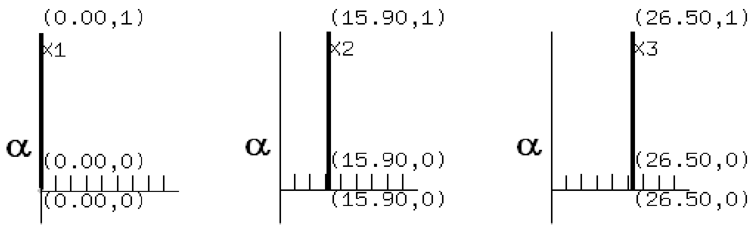


Fig. 7.3. $\overline{X}_1, \overline{X}_2, \overline{X}_3$ Solution using Chen's $\leq$, Product Mix Problem

Figure 7.3 show the "optimal" $\overline{X}_i$. Please note that this is not a crisp solution. They appear to be crisp when we round the results to two decimal places. Each left and right support is not coincident with its vertex. Figure 7.4 shows the value of the objective function $\overline{Z}$, corresponding to the values of the $\overline{X}_i$ given in Figure 7.3.

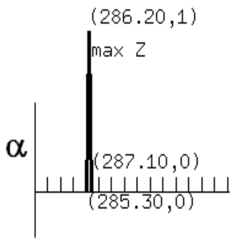


Fig. 7.4. $max\overline{Z}$ using Chen's $\leq$, Product Mix Problem

7.5 Comparison of Solutions

For another comparison the solution to the crisp problem is $x_1 = 0$, $x_2 = 27$, $x_3 = 16$ with max $z = 312$. The crisp linear program is the one obtained using the core values of all the fuzzy numbers.

All of these software efforts were performed on Windows-based PCs. These two fuzzy Monte Carlo optimizations were executed on a Lenovo T60 (T2400), 1.83GHz, 1.49GB RAM.

The Kerre comparison method found $8,749$ feasible sets in a stream of 100,000 $(\overline{X}_{1k},\overline{X}_{2k},\overline{X}_{3k})$; 38 of them had triggered new maximums. Elapsed time for the run was 14:38:08; the last maximum was found nearly at the end of the stream ($97,018^{th}$ vector triplet).

The Chen comparison method found $4,987$ feasible sets in the same stream of 100,000 $(\overline{X}_{1k},\overline{X}_{2k},\overline{X}_{3k})$; 7 of them had triggered new maximums. Elapsed time for the run was 14:14:35, but the last maximum was found after 04:44:17 at vector triplet $33,335$. The Chen solution was found with the $1,687^{th}$ feasible set. The solution using Chen's comparison method is not too fuzzy and is strangely not like the crisp solution reported above. We review the crisp solution with respect to the fuzzy constraint equations (7.4) and find that the second constraint equation evaluates to (311.3/312/312.7) which does not satisfy the constraint by either Kerre's or Chen's $\leq$. We note that a crisp solution to a crisp linear programming problem might not satisfy a fuzzy linear programming problem.

The two methods reported two identical new maximums. At the 492^{nd} vector triplet they reported a new maximum of $\approx$(155.06/275.57/346.00). Then again at vector triplet $33,335$ they found the same new maximum. Chen's method found no more new maximums.

For our Monte Carlo solution using Sobol quasi-random numbers, we compare Kerre's method results and Chen's method results to find that the $max\overline{Z}$ from Kerre's method is less than the $max\overline{Z}$ from Chen's method solution if Buckley's method or Kerre's method is used to compare the maximums. If Chen's method is used to compare the maximums, the $max\overline{Z}$ from Chen's method is less than the $max\overline{Z}$ from Kerre's method solution.

Yet another interesting solution is that given in [1], obtained by using an Evolutionary Algorithm to arrive at a solution. The results of that Evolutionary Algorithm method, using Kerre's method, are shown in Figures 7.5 and 7.6.

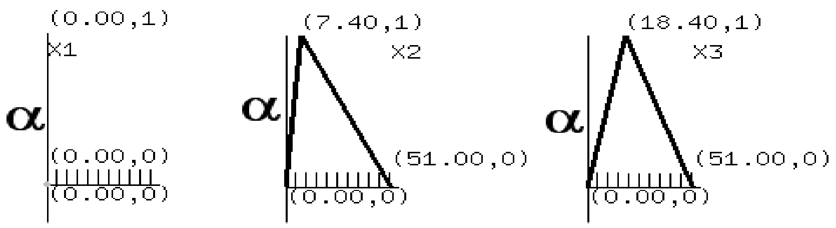


Fig. 7.5. Evolutionary Algorithm, $\overline{X}_1, \overline{X}_2, \overline{X}_3$ using Kerre's $\leq$ Product Mix Problem

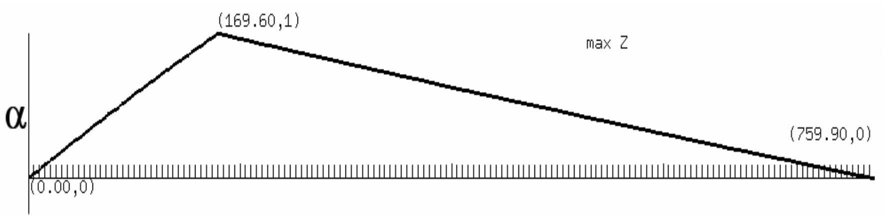


Fig. 7.6. Evolutionary Algorithm, $max\overline{Z}$ using Kerre's $\leq$, Product Mix Problem

All the fuzzy numbers in Figure 7.5 are triangular fuzzy numbers but $max\overline{Z}$ in Figure 7.6 is actually a triangular shaped fuzzy number. One will need to review [1] to understand their evolutionary algorithm method. We compare our Monte Carlo solution using Sobol quasi-random numbers and Kerre's method with the Evolutionary Algorithm solution from [1] and find that the $max\overline{Z}$ from our Monte Carlo solution is greater than the $max\overline{Z}$ from the Evolutionary Algorithm solution (regardless which of Buckley's method, Kerre's method, or Chen's method is used to compare the maximums).

Next we attempt to compare our Monte Carlo results using Chen's method with an Evolutionary Algorithm solution given in [1]. Unfortunately, we determined a discrepancy in those Evolutionary Algorithm results. That Evolutionary Algorithm solution satisfies neither the constraint equations nor the objective function. Thus we were not able to use them.

We see by comparing these fuzzy Monte Carlo solutions with a crisp solution that we have a solution not inconsistent with the crisp solution. Additionally, compared with an Evolutionary Algorithm solution, the fuzzy Monte Carlo Solution finds a greater fuzzy maximum.

References

1. Buckley, J.J., Feuring, T.: Evolutionary Algorithm Solution to Fuzzy Problems: Fuzzy Linear Programming. Fuzzy Sets and Systems 109, 35–53 (2000)
2. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)

8 Fully Fuzzified Linear Programming II

8.1 Introduction

This chapter follows from Chapter 7. However we now study a minimization problem. The diet problem is discussed in the next section. We have previously obtained an approximate fuzzy solution to this problem using an evolutionary algorithm ([1],[2]). In Section 7.3 we will apply our fuzzy Monte Carlo method to the problem to generate another approximate solution and then compare these new results to the evolutionary algorithm method.

8.2 Diet Problem

A farmer has three products P_1, P_2 and P_3 which he plans to mix together to feed his pigs. He knows the pigs require a certain amount of food F_1 and F_2 per day. Table 8.1 presents estimates of the units of F_1 and F_2 available, per gram of P_1, P_2 and P_3.

Table 8.1. Approximate Units of Food F_j in Product P_i

		Food	
		F_1	F_2
Product	P_1	2.5	5
	P_2	4.5	3
	P_3	5	10

Also, each pig should have approximately at least 54 units of F_1 and approximately at least 60 units of F_2, per day. The costs of P_1, P_2 and P_3 vary slightly from day to day but the average costs are: (1) 0.08\$ per gram of P_1; (2) 0.09\$ per gram of P_2; and (3) 0.10\$ per gram of P_3.

The farmer wants to know how many grams of P_1, P_2 and P_3 he should mix together each day, so his pigs will get the approximate minimums, to minimize his costs. Since all the numbers are uncertain we substitute triangular fuzzy numbers, whose peak is at the given value, for all the parameters to give a fully fuzzified linear program. The problem becomes

$$\min \overline{Z} = (7/8/9)\overline{X}_1 + (8/9/10)\overline{X}_2 + (9/10/11)\overline{X}_3 \tag{8.1}$$

$$\text{subject to:} \tag{8.2}$$

$$\begin{aligned}
&(2/2.5/3)\overline{X}_1 + (4/4.5/5)\overline{X}_2 + (4.5/5/5.5)\overline{X}_3 \geq (50/54/58),\\
&(4.5/5/5.5)\overline{X}_1 + (2.5/3/3.5)\overline{X}_2 + (9/10/11)\overline{X}_3 \geq (56/60/64),\\
&\overline{X}_1, \overline{X}_2, \overline{X}_3 \geq 0,
\end{aligned}$$

where $\overline{X}_i$ is the amount of P_i, in grams, to use per day, for each pig.

8.3 Fuzzy Monte Carlo Method

We first need to find intervals $I_i = [0, M_i]$, $i = 1, 2, 3$, as explained in Section 6.3 in Chapter 6, for the $\overline{X}_i$. Since we form fuzzy vector $\overline{V} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$, and each $\overline{X}_i$ consumes 5 crisp numbers, we choose our stream of Sobol quasi-random numbers which had been generated 15 at a time to get three sets of 5. We randomly generate $\overline{X}_i \in [0, M_i]$, $i = 1, 2, 3$, and form the random fuzzy vector $\overline{V} = (\overline{X}_1, \overline{X}_2, \overline{X}_3)$. We test to see if $\overline{V}$ is feasible, or the $\overline{X}_i$ satisfy the constraints. Assuming that $\overline{V}$ is feasible we compute $\overline{Z}_0 = \overline{C}_1\overline{X}_1 + ... + \overline{C}_3\overline{X}_3$. If $\overline{Z}^*$ is the current best (min) value of $\overline{Z}$ then we replace $\overline{Z}^*$ with $\overline{Z}_0$ if $\overline{Z}^* > \overline{Z}_0$, otherwise we discard $\overline{Z}_0$.

We also need to solve the fuzzy min problem twice. First we use Kerre's Method to evaluate $\overline{E}_i \geq \overline{B}_i$ in the constraints and $\overline{Z}^* > \overline{Z}_0$ in the objective function. Then we use Chen's Method.

To get an idea for the intervals I_i for the $\overline{X}_i$, $i = 1, 2, 3$, we studied the solutions to this problem from the evolutionary algorithm. Using Kerre's method the supports of all the optimal $\overline{X}_i$ were in $[0, 12]$. We also note that for low values of two $\overline{X}_i$, some $\overline{X}_i$ vertex must be near 20 to satisfy the constraint inequalities. So we chose all the intervals to be $[0, 20]$ for our Monte Carlo study when Kerre's method is used. From the evolutionary algorithm, using Chen's method the supports of all the optimal $\overline{X}_i$ were in $[0, 16]$. So we chose all the intervals to be $[0, 25]$ for a Monte Carlo study when Chen's method is used.

Now we follow the procedure outlined in Sections 6.3.1 and 6.3.2. We randomly generate sequences $\overline{V}_k = (\overline{X}_{1k}, \overline{X}_{2k}, \overline{X}_{3k})$ where the $\overline{X}_{ik}$ are Bézier (quadratic) fuzzy numbers (QBGFNs in Chapter 4). If $\overline{V}_k$ is feasible, using Kerre's or Chen's method, we compute the value of the fuzzy objective function in equation (8.1). We wish to solve the minimization problem given in equations (8.1) and (8.2).

With $N = 100,000$, and intervals $[0, 20]$, the results of the fuzzy Monte Carlo method for Kerre's comparison method are shown in Table 8.2. The results of the fuzzy Monte Carlo method for Chen's comparison method using a different stream of random fuzzy numbers based on $[0, 25]$, also with $N = 100,000$, are shown in Table 8.3. All the fuzzy numbers in Tables 8.2 and 8.3 are triangular shaped fuzzy numbers. $\overline{X}_1$, $\overline{X}_2$ and $\overline{X}_3$ are QBGFNs. The notation we use

Table 8.2. Results for the Diet Problem Using Fuzzy Monte Carlo, Kerre's Method, [0,20], N=100,000

$min\overline{Z}$	$\overline{X}_i$ (QBGFN from Sobol)
$\approx (23.70/122.20/224.99)$	$\overline{X}_1 = (0.41, 0.45, 3.11, 0.54, -0.59)$
	$\overline{X}_2 = (1.31, 6.78, 12.64, 5.32, 0.63)$
	$\overline{X}_3 = (1.15, 5.76, 6.43, 4.31, -0.16)$

Table 8.3. Results for the Diet Problem Using Fuzzy Monte Carlo, Chen's Method, [0,25], N=100,000

$min\overline{Z}$	$\overline{X}_i$ (QBGFN from Sobol)
$\approx (41.49/88.92/350.51)$	$\overline{X}_1 = (2.18, 3.71, 8.42, -0.00, 5.69)$
	$\overline{X}_2 = (2.39, 3.08, 23.64, -0.35, -4.23)$
	$\overline{X}_3 = (0.79, 3.16, 3.49, -2.19, -0.98)$

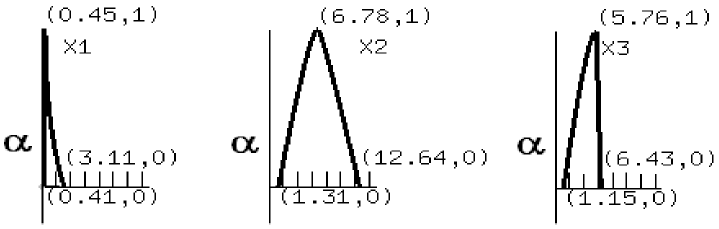


Fig. 8.1. $\overline{X}_1, \overline{X}_2, \overline{X}_3$ using Kerre's Inequality, Diet Problem

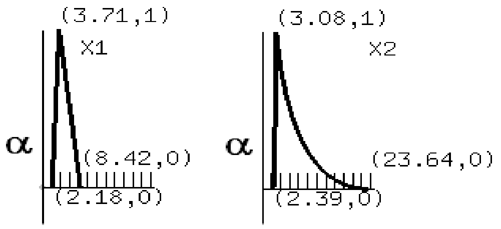


Fig. 8.2. $\overline{X}_1, \overline{X}_2, \overline{X}_3$ using Chen's Inequality, Diet Problem

for these fuzzy numbers was explained in Section 4.3.2. We define a QBGFN as (a, b, c, d, e) where: (1) the support is the interval $[a, c]$; (2) the vertex is at $x = b$; (3) the three numbers a, d, b define the quadratic function for the left side of the fuzzy number; and (4) the three numbers b, e, c specify the quadratic function for the right side of the fuzzy number. Since $min\overline{Z}$ is not necessarily, or likely to be a QBGFN, we only give the support and core for $min\overline{Z}$.

Figures 8.1 and 8.2 display the "optimal" $\overline{X}_i$, for both methods.

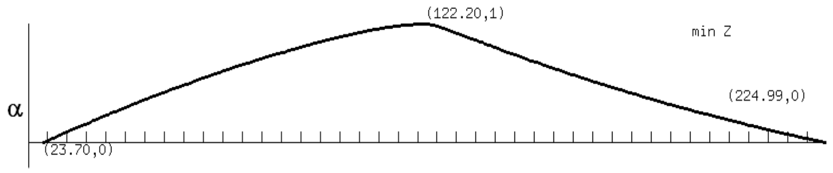


Fig. 8.3. $min\overline{Z}$ using Kerre's Inequality, Diet Problem

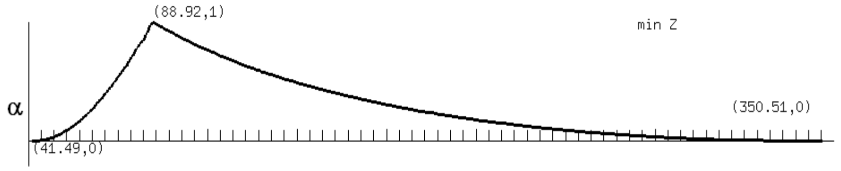


Fig. 8.4. $min\overline{Z}$ using Chen's Inequality, Diet Problem

Figures 8.3 and 8.4 present the value of the objective function $\overline{Z}$ for the values of the $\overline{X}_i$ shown in Figures 8.1 and 8.2, respectively.

8.4 Comparison of Solutions

For another comparison the solution to the crisp problem is $x_1 = 0$, $x_2 = 8$, $x_3 = 3.6$ with min $z = 108$ (in cents). The crisp linear program is the one obtained using the core values of all the fuzzy numbers.

All of these software efforts were performed on Windows-based PCs. For the fuzzy Monte Carlo optimizations, several computers were used, all Dell Optiplex GX270's, 3.0GHz, 1GB RAM.

First we consider the Kerre simulations using the interval $[0, 20]$. The Kerre comparison method found $97,580$ feasible sets in a stream of 100,000 $(\overline{X}_{1k}, \overline{X}_{2k}, \overline{X}_{3k})$; 7 of them triggered new minimums. Elapsed time for the run was over 24 hours, but the last minimum was found after 640 feasible sets at 00:07:16 into the execution.

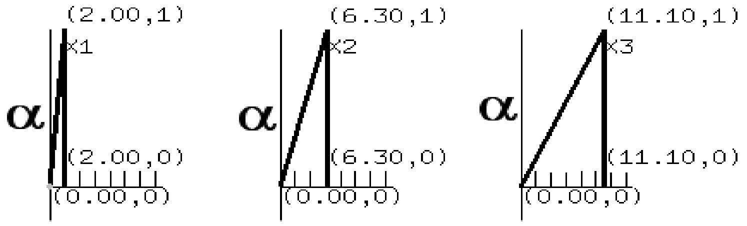


Fig. 8.5. Evolutionary Algorithm, $\overline{X}_1, \overline{X}_2, \overline{X}_3$ using Kerre's $\leq$, Diet Problem

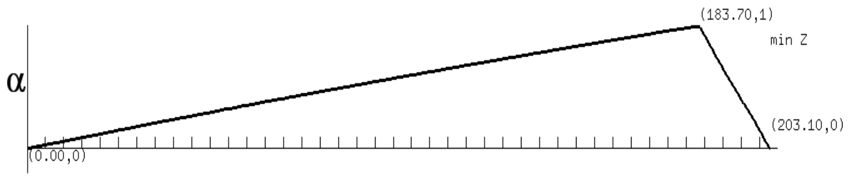


Fig. 8.6. Evolutionary Algorithm, $min\overline{Z}$ using Kerre's $\leq$, Diet Problem

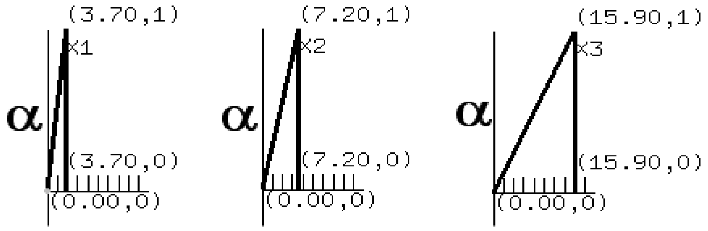


Fig. 8.7. Evolutionary Algorithm, $\overline{X}_1, \overline{X}_2, \overline{X}_3$ using Chen's $\leq$, Diet Problem

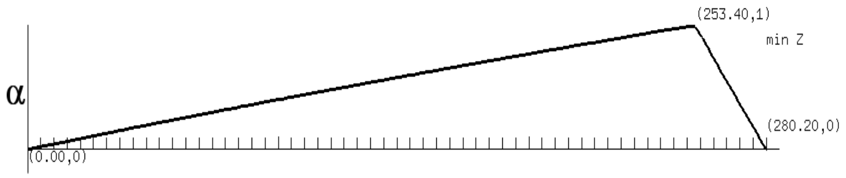


Fig. 8.8. Evolutionary Algorithm, $min\overline{Z}$ using Chen's $\leq$, Diet Problem

One may compare this solution is that given in [1], obtained by using an Evolutionary Algorithm to arrive at a solution. The results of that Evolutionary Algorithm method, using Kerre's method, are shown in Figures 8.5 and 8.6. The fuzzy numbers in Figure 8.5 are triangular fuzzy numbers. One will need to review [1] to understand their evolutionary algorithm method. We compare our Monte Carlo solution using Sobol quasi-random numbers and Kerre's method with the Evolutionary Algorithm solution from [1] and find that the $min\overline{Z}$ from our Monte Carlo solution is less than the $min\overline{Z}$ from the Evolutionary Algorithm solution (regardless which of Buckley's method, Kerre's method, or Chen's method is used to compare the minimums).

The Chen $[0, 25]$ comparison method found $69,498$ feasible sets in its stream of 100,000 ($\overline{X}_{1k}$,$\overline{X}_{2k}$,$\overline{X}_{3k}$); 15 of them triggered new minimums. Elapsed time for the run was about 15 hours, with the last minimum found at feasible set $30,460$ after 04:48:56. Having different input streams, the Kerre $[0, 20]$ and the Chen $[0, 25]$ methods' minimums did not track each other.

Another interesting solution is that given in [1], obtained by using an Evolutionary Algorithm. The results of that Evolutionary Algorithm method, using Chen's method, are shown in Figures 8.7 and 8.8. The fuzzy numbers in Figure 8.7 are triangular fuzzy numbers. One will need to review [1] to understand their evolutionary algorithm method. We compare our Monte Carlo solution using Sobol quasi-random numbers and Chen's method with the Evolutionary Algorithm solution using Chen's method from [1] and find that the $min\overline{Z}$ from our Monte Carlo solution is less than the $min\overline{Z}$ from the Evolutionary Algorithm solution (regardless which of Buckley's method, Kerre's method, or Chen's method is used to compare the minimums).

For our Monte Carlo solution using Sobol quasi-random numbers, we compare Kerre's method results and Chen's method results to find that the $min\overline{Z}$ from Chen's method is less than the $min\overline{Z}$ from Kerre's method solution (regardless which of Buckley's method, Kerre's method, or Chen's method is used to compare the maximums).

We see by comparing these fuzzy Monte Carlo solutions with a crisp solution that we have a solution consistent with the crisp solution. Additionally, compared with an Evolutionary Algorithm solution, the fuzzy Monte Carlo Solution finds a lesser fuzzy minimum.

References

1. Buckley, J.J., Feuring, T.: Evolutionary Algorithm Solution to Fuzzy Problems: Fuzzy Linear Programming. Fuzzy Sets and Systems 109, 35–53 (2000)
2. 2 Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)

9 Fuzzy Multiobjective LP

9.1 Introduction

This chapter continues Chapters 7 and 8. However now we have multiple objective functions we wish to maximize. We first discuss the general multiobjective fully fuzzified linear program in the next section. Then we study an example problem in Section 9.3. We have previously obtained an approximate fuzzy solution to this type of problem using an evolutionary algorithm [2]. In Section 9.4 we will apply our fuzzy Monte Carlo method to the problem to generate another approximate solution. Unfortunately, we will be unable to compare our Monte Carlo solution to our previous solution because we now are forced to use a different method of evaluating fuzzy inequalities.

Fuzzy multiobjective linear programming has also (along with fuzzy linear programming) become a large area of research. A few recent references to this topic are the papers ([1],[3],[6],[7],[9],[10],[16],[17]) and books (or articles in these books) ([4],[5],[8],[11]-[15]).

9.2 Multiobjective Fully Fuzzified Linear Programming

We are interested in the following problem

$$\max \left(\overline{Z} = (\overline{Z}_1, \ldots, \overline{Z}_K)\right) \tag{9.1}$$

where

$$\overline{Z}_k = \sum_{j=1}^{n} \overline{C}_{kj} \overline{X}_j, 1 \le k \le K, \tag{9.2}$$

subject to

$$\sum_{j=1}^{n} \overline{A}_{ij} \overline{X}_j \le \overline{B}_i, 1 \le i \le m, \tag{9.3}$$

$$\overline{X}_j \ge 0, \text{for all } j. \tag{9.4}$$

In the above problem $\overline{C}_{kj}$, $\overline{A}_{ij}$, and $\overline{B}_i$ are all triangular fuzzy numbers. The $\overline{X}_i$ are triangular shaped fuzzy numbers. Let us rewrite it using matrix notation

$$\max\left(\overline{Z} = (\overline{Z}_1, \ldots, \overline{Z}_K)\right) \tag{9.5}$$

$$\overline{Z}_k = \overline{C}_k\ \overline{X}, 1 \leq k \leq K \tag{9.6}$$

$$\overline{A}\ \overline{X} \leq \overline{B}, \overline{X} \geq 0, \tag{9.7}$$

for $\overline{C}_k = (\overline{C}_{k1}, \ldots, \overline{C}_{kn})$, $1 \leq k \leq K$, $\overline{X}^t = (\overline{X}_1, \ldots, \overline{X}_n)$, $\overline{B}^t = (\overline{B}_1 \ldots, \overline{B}_m)$ and $\overline{A} = [\overline{A}_{ij}]$ a $m \times n$ matrix of fuzzy numbers.

To completely define the problem in equations (9.5)-(9.7) we must do two things: (1) define what we mean by max $\overline{Z}$, or finding the maximum of a vector of triangular shaped fuzzy numbers; and (2) explain what is meant by $\overline{A}\ \overline{X} \leq \overline{B}$. In the previous publications (see [2]) we handled max/min $\overline{Z}$ basically as discussed in Section 2.5 of Chapter 2. Also, in those publications we used both Kerre's Method (Section 2.6.2) and Chen's Method (Section 2.6.3) to evaluate $\leq$ and $\geq$ between fuzzy numbers. In our fuzzy Monte Carlo method we want to use both Kerre's Method and Chen's Method to evaluate $\leq$, $\geq$, $>$ and $<$ between fuzzy numbers but we will explain that we need to use another method.

9.3 Example Problem

This example is adapted from an example in ([18], p. 217). The crisp problem is

$$\max\left(z_1 = 5x_1 + 3x_2, z_2 = 2x_1 + 8x_2\right) \tag{9.8}$$

$$\text{subject to: } x_1 + 4x_2 \leq 100 \tag{9.9}$$

$$3x_1 + 2x_2 \leq 150 \tag{9.10}$$

$$5x_1 + 3x_2 \geq 200 \tag{9.11}$$

$$2x_1 + 8x_2 \geq 75 \tag{9.12}$$

$$x_1, x_2 \geq 0. \tag{9.13}$$

The feasible set is shown $\mathcal{F}$ in Figure 9.1, which is the bounded region with vertices A,B,C and D, and the undominated set is the line segment from A to B. The last constraint is inactive and does not define any part of the boundary of the feasible set. Undominated crisp vectors are discussed in Section 2.7 in Chapter 2. The general solution to this type of optimization problem is the set of undominated vectors $z = (z_1, z_2)$.

We now fully fuzzify the crisp problem using triangular fuzzy numbers whose vertex (core) value is the corresponding number in the crisp problem. The variables $\overline{X}_i$ will be triangular shaped fuzzy numbers. The multiobjective fully fuzzified linear program is

$$\max\left(\overline{Z} = (\overline{Z}_1, \overline{Z}_2)\right) \tag{9.14}$$

$$\overline{Z}_1 = (4/5/6)\overline{X}_1 + (2/3/4)\overline{X}_2 \tag{9.15}$$

$$\overline{Z}_2 = (1/2/3)\overline{X}_1 + (6/8/10)\overline{X}_2 \tag{9.16}$$

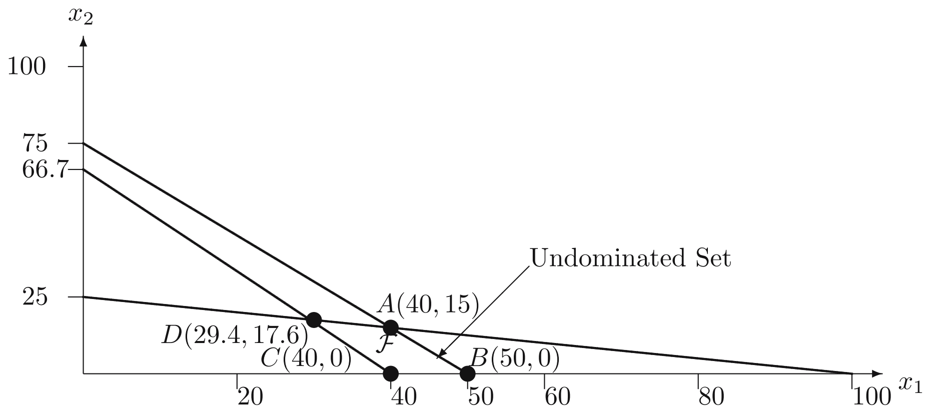


Fig. 9.1. Feasible Set in the Example problem

subject to:

$$(0/1/2)\overline{X}_1 + (3/4/5)\overline{X}_2 \leq (95/100/105) \tag{9.17}$$

$$(2/3/4)\overline{X}_1 + (1/2/3)\overline{X}_2 \leq (140/150/160) \tag{9.18}$$

$$(3/5/7)\overline{X}_1 + (2/3/4)\overline{X}_2 \geq (180/200/220) \tag{9.19}$$

$$(1/2/3)\overline{X}_1 + (6/8/10)\overline{X}_2 \geq (70/75/80) \tag{9.20}$$

$$\overline{X}_1, \overline{X}_2 \geq 0. \tag{9.21}$$

9.4 Fuzzy Monte Carlo Method

We will need to find intervals $I_i = [0, M_i]$, $i = 1, 2$, as explained in Section 6.3 in Chapter 6, for the $\overline{X}_i$. We randomly generate $\overline{X}_i \in [0, M_i]$, $i = 1, 2$, and form the random fuzzy vector $\overline{V} = (\overline{X}_1, \overline{X}_2)$. Since we form fuzzy vector $\overline{V} = (\overline{X}_1, \overline{X}_2)$, and each $\overline{X}_i$ consumes 5 crisp numbers, we choose our stream of Sobol quasi-random numbers which had been generated 10 at a time to get the two pairs of 5. We test to see if $\overline{V}$ is feasible, or the $\overline{X}_i$ satisfy the constraints. Assuming that $\overline{V}$ is feasible we compute $\overline{Z}_1 = \overline{C}_{11}\overline{X}_1 + \overline{C}_{12}\overline{X}_2$ and $\overline{Z}_2 = \overline{C}_{21}\overline{X}_1 + \overline{C}_{22}\overline{X}_2$. Now we combine $\overline{Z}_1$ and $\overline{Z}_2$ into one fuzzy number $\overline{Z}$ for ranking (finding the max). Let

$$\overline{Z} = \lambda\overline{Z}_1 + (1-\lambda)\overline{Z}_2, \tag{9.22}$$

for $0 < \lambda < 1$. The decision maker(s) will choose various values for λ and then solve the fuzzy optimization problem max $\overline{Z}$. Let the current value of $\overline{Z} = \overline{Z}_0$ from feasible fuzzy vector $\overline{V}$. If $\overline{Z}^*$ is the best (max) value of $\overline{Z}$ up to now, then we replace $\overline{Z}^*$ with $\overline{Z}_0$ if $\overline{Z}^* < \overline{Z}_0$, otherwise we discard $\overline{Z}_0$.

We discussed weak and strong domination between fuzzy vectors in Section 2.7. We concluded that if we use Buckley's Method (Section 2.6.1) of evaluating inequalities between fuzzy numbers, then solutions $\overline{V} = (\overline{X}_1, \overline{X}_2)$ to $max\overline{Z}$ in equation (9.22) are strongly undominated solutions to the multiobjective fully fuzzified linear program. Let us explain in more detail what this means.

Pick and fix a value for the $\lambda > 0$. Let $\overline{V} = (\overline{X}_1, \overline{X}_2)$ be feasible (satisfy the constraints in equations (9.17)-(9.21) using Buckley's Method) and maximize $\overline{Z}$ in equation (9.22). Then $\overline{V}$ is strongly undominated which means that no other feasible fuzzy vector $\overline{W}$ can strongly dominate $\overline{V}$. This result is not true using Kerre's Method or Chen's Method (Section 2.7). The general solution to the multiobjective fully fuzzified linear program will be the set of strongly undominated fuzzy vectors. This means that we do not use Kerre's Method or Chen's Method to evaluate fuzzy inequalities for this problem.

Looking at the undominated set in Figure 9.1 we chose the interval $[20, 70]$ for $\overline{X}_1$ and $[0, 50]$ for $\overline{X}_2$. The $\overline{X}_i$ will be Bézier (quadratic) fuzzy numbers (QBGFNs in Chapter 4). So we now randomly generate a sequence $\overline{V}_k = (\overline{X}_{1k}, \overline{X}_{2k})$ with $\overline{X}_{1k} \in [20, 70]$ and $\overline{X}_{2k} \in [0, 50]$ all k. Using our Sobol quasi-random number generator we produce sequences of random vectors $v_{1k} = (x_{1k1}, ..., x_{1k5})$, $v_{2k} = (x_{2k1}, ..., x_{2k5})$, $k = 1, 2, 3, ...$ The sequence v_{1k} is used to get the sequence of quadratic fuzzy numbers $\overline{X}_{1k}$, recall that we only require vectors of length five for these fuzzy numbers (see Chapter 4), and the other sequence v_{2k} constructs the sequence of quadratic fuzzy numbers $\overline{X}_{2k}$, $k = 1, 2, 3, ...$ If $\overline{V}_k$ is feasible we compute

$$\overline{Z}_{1k} = \overline{C}_{11}\overline{X}_{1k} + \overline{C}_{12}\overline{X}_{2k}, \tag{9.23}$$

and

$$\overline{Z}_{2k} = \overline{C}_{21}\overline{X}_{1k} + \overline{C}_{22}\overline{X}_{2k}, \tag{9.24}$$

for $k = 1, 2, 3, ..., N$, where N is the predetermined total number of iterations.

Assume that the decision maker(s) believe that the two goals are equally important so they picked $\lambda = 0.5$. We want to find a k value, and hence a $\overline{V}_k$, to solve

$$max\{\overline{Z}_k | k = 1, 2, 3, ..., N\}, \tag{9.25}$$

where

$$\overline{Z}_k = 0.5\overline{Z}_{1k} + 0.5\overline{Z}_{2k}, \tag{9.26}$$

$k = 1, 2, 3, ..., N$.

Table 9.1. Monte Carlo Solution to the Fuzzy Linear Program, Buckley's Method, N=100,000

$max\overline{Z}$	$\overline{X}_i$ (QBGFN from Sobol)
$\approx (76.25/188.57/337.92)$	$\overline{X}_1 = (39.75, 40.54, 45.21, -0.49, 0.55)$
	$\overline{X}_2 = (8.26, 8.49, 8.55, 0.69, 0.68)$

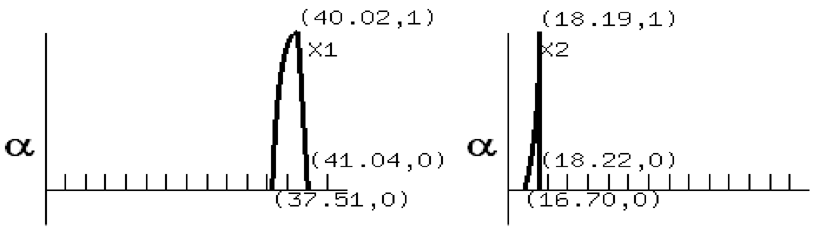


Fig. 9.2. $\overline{X}_1, \overline{X}_2$ using Buckley's Inequality, Example Problem

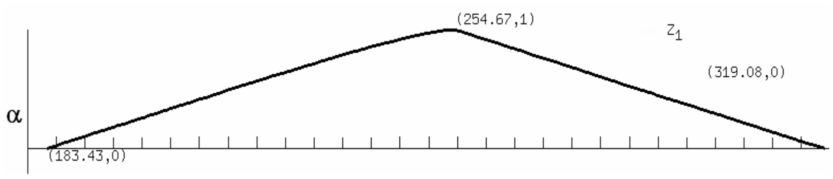
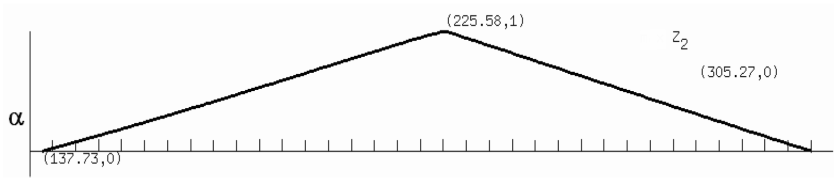


Fig. 9.3. $\overline{Z}_1$ using Buckley's Inequality, Example Problem



1.5

Fig. 9.4. $\overline{Z}_2$ using Buckley's Inequality, Example Problem

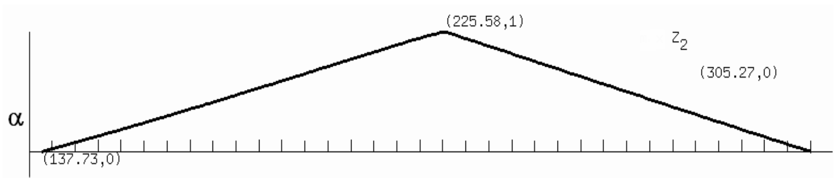


Fig. 9.5. $max\overline{Z}$ using Buckley's Inequality, Example Problem

With $N = 100,000$ the results of the fuzzy Monte Carlo method are shown in Table 9.1. All the fuzzy numbers in Table 9.1 are triangular shaped fuzzy numbers. $\overline{X}_1$ and $\overline{X}_2$ are QBGFNs. The notation we use for these fuzzy numbers was explained in Section 4.3.2. We define a QBGFN as (a, b, c, d, e) where: (1) the support is the interval $[a, c]$; (2) the vertex is at $x = b$; (3) the three numbers a, d, b define the quadratic function for the left side of the fuzzy number; and (4) the three numbers b, e, c specify the quadratic function for the right side of the fuzzy number. Since $max\overline{Z}$ is not necessarily, or likely to be a QBGFN, we only give the support and core for $max\overline{Z}$.

Figure 9.2 shows the "optimal" $\overline{X}_i$ and Figure 9.5 shows the value of the objective function $\overline{Z}$ using $\lambda = 0.5$, corresponding to the values of the $\overline{X}_i$ given in Figure 9.2.

9.5 Compare Solutions

For this multiobjective linear programming problem we have no crisp solution.

All of these software efforts were performed on Windows-based PCs. For the fuzzy Monte Carlo optimizations, several computers were used, mostly Dell Optiplex GX270's, 3.0GHz, 1GB RAM. This particular Monte Carlo simulation was performed on a Dell Inspiron 8200, 1.8GHz, 1GB RAM.

The fuzzy Monte Carlo optimization found 25,464 feasible sets in a stream of 100,000 ($\overline{X}_{1k}$,$\overline{X}_{2k}$); 36 of them triggered new maximums. Elapsed time for the run was 16:15:57, but the last minimum was found after 11,332 feasible sets at 07:26:06 (iteration 44,528 of 100,000) into the execution.

One may compare this solution is that given in [1], obtained by using an Evolutionary Algorithm to arrive at a solution. The results of that Evolutionary Algorithm method, using Kerre's method, are shown in Figures 9.6, 9.7, 9.8, and 9.9. The fuzzy numbers in Figure 9.6 are triangular fuzzy numbers. The fuzzy numbers in Figures 9.7, 9.8, and 9.9 are triangular shaped fuzzy numbers. One will need to review [1] to understand their evolutionary algorithm method. We compare our Monte Carlo solution using Sobol quasi-random numbers and Buckley's method with the Evolutionary Algorithm solution from [1] and find that the $max\overline{Z}$ from our Monte Carlo solution is greater than the $max\overline{Z}$ from the Evolutionary Algorithm solution (regardless which of Buckley's method, Kerre's method, or Chen's method is used to compare the maximums).

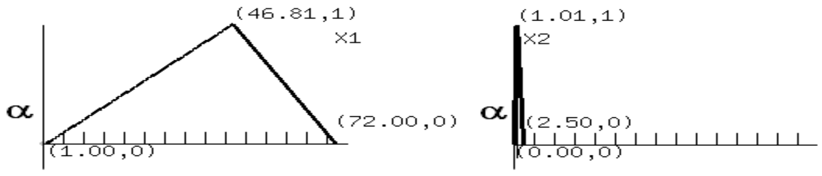


Fig. 9.6. Evolutionary Algorithm, $\overline{X}_1, \overline{X}_2$ using Kerre's $\leq$

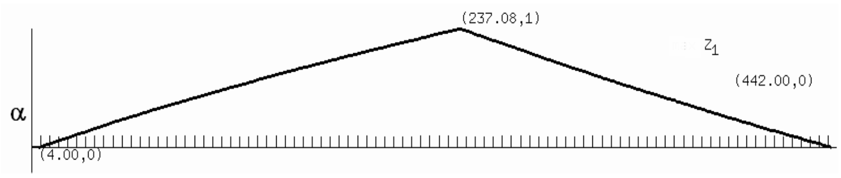


Fig. 9.7. Evolutionary Algorithm, $\overline{Z}_1$ using Kerre's $\leq$

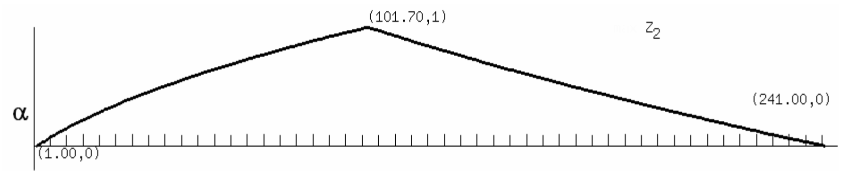


Fig. 9.8. Evolutionary Algorithm, $\overline{Z}_2$ using Kerre's $\leq$

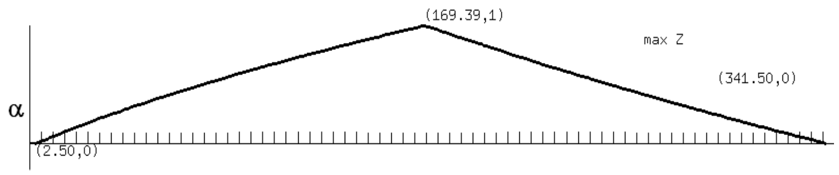


Fig. 9.9. Evolutionary Algorithm, $max\overline{Z}$ using Kerre's $\leq$

Next we attempt to compare our Monte Carlo results using Chen's method with an Evolutionary Algorithm solution given in [1]. Unfortunately, we determine a discrepancy in those Evolutionary Algorithm results. That Evolutionary Algorithm solution fails to satisfy the second and third constraint equations. Thus we were not able to use them.

We see by comparing these fuzzy Monte Carlo solutions with an Evolutionary Algorithm solution, the fuzzy Monte Carlo solution finds a greater fuzzy maximum.

References

1. Ammar, E.E., Kassem, M.A.E.-H.: On Stability Analysis of Multicriteria LP Problems with Fuzzy Parameters. Fuzzy Sets and Systems 82, 331–334 (1996)
2. Buckley, J.J., Feuring, T., Hayashi, Y.: Multiobjective Fully Fuzzified Linear Programming. Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems 9, 605–622 (2001)
3. Cadenas, J.M., Verdegay, J.L.: Using Ranking Functions in Multiobjective Fuzzy Linear Programming. Fuzzy Sets and Systems 111, 47–53 (2000)
4. Carlsson, C., Fuller, R.: Fuzzy Reasoning in Decision Making and Optimization. Physica-Verlag, Heidelberg (2002)
5. Ehrgott, M., Gandibleux, X.: Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys. Kluwer Academic Press, Norwell, Mass. (2002)
6. Fuller, R., Carlsson, C.: Fuzzy Multiple Criteria Decision Making: Recent Developments. Fuzzy Sets and Systems 78, 139–153 (1996)
7. Gen, M., Ida, K., Kobuchi, R.: Neural Network Techniques for Fuzzy Multiobjective Linear Programming. Computers and Industrial Engineering 35, 543–546 (1998)
8. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization. Wiley Interscience, N.Y. (1999)

9. Phruksaphanrat, B., Ohsato, A.: Linear Coordination Method for Fuzzy Multi-Objective Linear Programming Problems with Convex Polyhedral Membership Functions. Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems 12, 269–285 (2004)
10. Saad, O.M.: Stability on Multiobjective Linear Programming Problems with Fuzzy Parameters. Fuzzy Sets and Systems 74, 207–215 (1995)
11. Sakawa, M.: Fuzzy Sets and Interactive Multiobjective Optimization. Springer, Heidelberg (1993)
12. Sakawa, M.: Large Scale Interactive Fuzzy Multiobjective Programming. Springer, Heidelberg (2000)
13. Sakawa, M.: Genetic Algorithms and Fuzzy Multiobjective Optimization. Kluwer Academic Press, Norwell, Mass. (2001)
14. Tanino, T., Tanaka, T., Inuiguichi, M.: Multi-Objective Programming and Goal Programming: Theory and Applications. Springer, Heidelberg (2003)
15. Trzaskalik, T., Michnik, J. (eds.): Multiple Objective and Goal Programming. Physica-Verlag, Heidelberg (2002)
16. Wang, H.-F., Wang, M.-L.: A Fuzzy Multiobjective Linear Programming. Fuzzy Sets and Systems 86, 61–72 (1997)
17. Yu, C.S., Li, H.L.: Method for Solving Quasi-Concave and Non-Concave Fuzzy Multi-Objective Programming Problems. Fuzzy Sets and Systems 122, 205–227 (2001)
18. Zeleny, M.: Multiple Criteria Decision Making. McGraw Hill, N.Y. (1982)

10 Solving Fuzzy Equations

10.1 Introduction

We start in the next Section 10.2 with looking at possible solutions to the simple fuzzy linear equation $\overline{A} \cdot \overline{X} + \overline{B} = \overline{C}$. We discuss three different types of solution which we have studied before in solving fuzzy equations. Then we present a fourth type of solution, based on our fuzzy Monte Carlo method, in Section 10.2.2. This new solution is based on random fuzzy numbers. In Section 10.3 we look at only "classical" solutions to the fuzzy quadratic equation and apply our fuzzy Monte Carlo method to obtain new solutions. Then in Section 10.4 we consider the fuzzy matrix equation $\overline{A} \cdot \overline{X} = \overline{B}$ and a number of solution types for $\overline{X}$ and then another solution based on fuzzy Monte Carlo techniques. The last section contains a brief summary and our conclusions.

In this chapter $\overline{M} \leq \overline{N}$ will mean that $\overline{M}$ is a fuzzy subset of $\overline{N}$ (Section 2.2.3) and not that $\overline{M}$ is less than or equal to $\overline{N}$. Solving fuzzy equations has always been an active area of research. Some recent references on this topic are ([1]-[4],[16]-[18],[21],[22]).

10.2 $\overline{A}\,\overline{X} + \overline{B} = \overline{C}$

$\overline{A}$, $\overline{B}$ and $\overline{C}$ will be triangular fuzzy numbers so let $\overline{A} = (a_1/a_2/a_3)$, $\overline{B} = (b_1/b_2/b_3)$ and $\overline{C} = (c_1/c_2/c_3)$. $\overline{X}$, if it exists, will be a triangular shaped fuzzy number so let $\overline{X} \approx (x_1/x_2/x_3)$. In the crisp equation

$$ax + b = c, \tag{10.1}$$

we immediately obtain $X = (c-b)/a$, if $a \neq 0$. We used the important facts $b - b = 0$ and $(1/a)a = 1$ from real numbers to get the solution.

We try this same approach with the fuzzy equation

$$\overline{A}\,\overline{X} + \overline{B} = \overline{C}, \tag{10.2}$$

we get

$$(1/\overline{A})(\overline{A}\,\overline{X} + (\overline{B} - \overline{B})) = (1/\overline{A})(\overline{C} - \overline{B}). \tag{10.3}$$

But the left side of the equation (10.3) does not equal $\overline{X}$ since $\overline{B} - \overline{B} \neq 0$ and $(1/\overline{A})(\overline{A}) \neq 1$. For example, if $\overline{B} = (1/2/3)$, then $\overline{B} - \overline{B} = (-2/0/2)$ not zero. Also, if $\overline{A} = (1/2/3)$, $(1/\overline{A})(\overline{A}) \approx (\frac{1}{3}/1/3)$, a triangular shaped fuzzy number, not $(1/1/1)$.

This shows a major problem in solving fuzzy equations: some basic operations we used to solve crisp equations do not hold for fuzzy equations. Actually, this comes as no great surprise because this also happens in probability theory. If X is a random variable with positive variance, then $X - X \neq 0$ and $X/X \neq 1$ since both $X - X$ and X/X will have positive variance.

We now introduce our first solution method, called the classical method, producing solution $\overline{X}_c$ (when it exists). This procedure employs α-cuts and interval arithmetic (Section 2.4.2) to solve for $\overline{X}_c$. Let $\overline{A}[\alpha] = [a_1(\alpha), a_2(\alpha)]$, $\overline{B}[\alpha] = [b_1(\alpha), b_2(\alpha)]$, $\overline{C}[\alpha] = [c_1(\alpha), c_2(\alpha)]$ and $\overline{X}_c[\alpha] = [x_1(\alpha), x_2(\alpha)]$, $0 \leq \alpha \leq 1$. Substitute these into equation (10.2) producing

$$[a_1(\alpha), a_2(\alpha)][x_1(\alpha), x_2(\alpha)] + [b_1(\alpha), b_2(\alpha)] = [c_1(\alpha), c_2(\alpha)]. \tag{10.4}$$

We now use interval arithmetic (Section 2.3.2) to solve equation (10.4) for $x_1(\alpha)$ and $x_2(\alpha)$. We say that this method defines solution $\overline{X}_c$ when $[x_1(\alpha), x_2(\alpha)]$ defines the α-cuts of a fuzzy number. For the $x_1(\alpha)$, $x_2(\alpha)$ to specify a fuzzy number we need:

1. $x_1(\alpha)$ monotonically increasing, $0 \leq \alpha \leq 1$;
2. $x_2(\alpha)$ monotonically decreasing, $0 \leq \alpha \leq 1$; and
3. $x_1(1) \leq x_2(1)$.

We did not mention anything about the $x_i(\alpha)$ being continuous because throughout this chapter $x_1(\alpha)$, $x_2(\alpha)$ will be continuous.

Example 10.2.1

Let $\overline{A} = (1/2/3)$, $\overline{B} = (-3/-2/-1)$ and $\overline{C} = (3/4/5)$. Then $\overline{A}[\alpha] = [1+\alpha, 3-\alpha]$, $\overline{B}[\alpha] = [-3+\alpha, -1-\alpha]$, $\overline{C}[\alpha] = [3+\alpha, 5-\alpha]$. Since $\overline{A} > 0$ and $\overline{C} > 0$, we must have $\overline{X}_c > 0$, and equation (10.4) gives

$$[a_1(\alpha)x_1(\alpha) + b_1(\alpha), a_2(\alpha)x_2(\alpha) + b_2(\alpha)] = [c_1(\alpha), c_2(\alpha)], \tag{10.5}$$

or

$$x_1(\alpha) = \frac{6}{1+\alpha}, \tag{10.6}$$

$$x_2(\alpha) = \frac{6}{3-\alpha}, \tag{10.7}$$

after substituting for $a_1(\alpha), \ldots, c_2(\alpha)$ and solving for $x_i(\alpha)$. But $x_1(\alpha)$ is decreasing and $x_2(\alpha)$ is increasing. So, $\overline{X}_c$ does not exist.

Example 10.2.2

Now set $\overline{A} = (8/9/10)$, $\overline{B} = (-3/-2/-1)$ and $\overline{C} = (3/5/7)$. So $\overline{A}[\alpha] = [8+\alpha, 10-\alpha]$, $\overline{B}[\alpha] = [-3+\alpha, -1-\alpha]$, $\overline{C}[\alpha] = [3+2\alpha, 7-2\alpha]$. Again we must have $\overline{X}_c > 0$ so we obtain

$$x_1(\alpha) = \frac{6+\alpha}{8+\alpha}, \tag{10.8}$$

$$x_2(\alpha) = \frac{8-\alpha}{10-\alpha}. \tag{10.9}$$

We see that $x_1(\alpha)$ is increasing (its derivative is positive), $x_2(\alpha)$ is decreasing (derivative is negative) and $x_1(1) = 7/9 = x_2(1)$. The solution $\overline{X}_c$ exists, with α-cuts $[x_1(\alpha), x_2(\alpha)]$, shown in Figure 10.1.

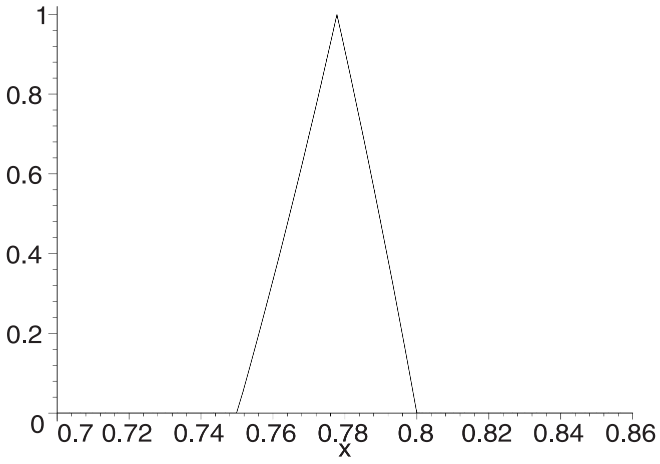


Fig. 10.1. Solution to Example 10.2.2, $\overline{X}_c \approx (0.75/0.77\overline{7}/0.8)$

Working more examples, like Examples 10.2.1 and 10.2.2 above, we conclude that too often fuzzy equations have no solution ($\overline{X}_c$). This motivated the authors in ([5]-[15]) to propose other solutions for fuzzy equations. These new solutions will be introduced in the next section. The classical solution, plus the new solutions, will be used throughout this chapter.

10.2.1 Other Solutions

We continue working with the fuzzy equation $\overline{A}\,\overline{X} + \overline{B} = \overline{C}$. The other solutions simply fuzzify the crisp solution $(c-b)/a$, $a \neq 0$. The fuzzified crisp solution is

$$(\overline{C} - \overline{B})/\overline{A}, \tag{10.10}$$

where we assume zero does not belong to the support of $\overline{A}$. There are two ways to evaluate equation (10.10). The first method is the extension principle. If $\overline{X}_e$ is the value of equation (10.10) using the extension principle, then

$$\overline{X}_e(x) = \max\{\pi(a,b,c) \mid (c-b)/a = x\}, \tag{10.11}$$

where

$$\pi(a,b,c) = \min\left\{\overline{A}(a), \overline{B}(b), \overline{C}(c)\right\}. \tag{10.12}$$

Since the expression $(c-b)/a$, $a \neq 0$, is continuous in a, b, c we know how to find α-cuts of $\overline{X}_e$ (Section 2.4.1)

$$x_{e1}(\alpha) = \min\left\{(c-b)/a \mid a \in \overline{A}[\alpha], b \in \overline{B}[\alpha], c \in \overline{C}[\alpha]\right\}, \tag{10.13}$$

$$x_{e2}(\alpha) = \max\left\{(c-b)/a \mid a \in \overline{A}[\alpha], b \in \overline{B}[\alpha], c \in \overline{C}[\alpha]\right\}, \tag{10.14}$$

where

$$\overline{X}_e[\alpha] = [x_{e1}(\alpha), x_{e2}(\alpha)], \tag{10.15}$$

$0 \leq \alpha \leq 1$. $\overline{X}_e$ will be a triangular shaped fuzzy number when $\overline{A}$, $\overline{B}$, $\overline{C}$ are triangular fuzzy numbers. It is not difficult to show that if $\overline{X}_c$ exists, $\overline{X}_c \leq \overline{X}_e$. In this chapter $\overline{X}_c \leq \overline{X}_e$ means that $\overline{X}_c$ is a fuzzy subset of $\overline{X}_e$ (Section 2.2.3).

An important fact about $\overline{X}_c$ is that it will satisfy the fuzzy equation. That is $\overline{A} \cdot \overline{X}_c + \overline{B} = \overline{C}$ holds using α-cuts and interval arithmetic. However, $\overline{X}_e$ may, or may not, satisfy the fuzzy equation. However, $\overline{X}_e$ will always exist but $\overline{X}_c$ may fail to exist.

The second way to evaluate equation (10.10) is to use α-cuts and interval arithmetic. If the result is $\overline{X}_I$, we have

$$\overline{X}_I[\alpha] = \frac{\overline{C}[\alpha] - \overline{B}[\alpha]}{\overline{A}[\alpha]}, \tag{10.16}$$

to be simplified by interval arithmetic, $0 \leq \alpha \leq 1$. It is also not too difficult to argue that $\overline{X}_e \leq \overline{X}_I$.

$\overline{X}_I$ may or may not satisfy the fuzzy equation. $\overline{X}_I$ will be a triangular shaped fuzzy number when $\overline{A}$, $\overline{B}$, $\overline{C}$ are all triangular fuzzy numbers. We summarize these results as:

1. If $\overline{X}_c$ exists, then $\overline{X}_c \leq \overline{X}_e \leq \overline{X}_I$;
2. $\overline{X}_c$ always satisfies the fuzzy equation;
3. $\overline{X}_e \leq \overline{X}_I$.

Up to now our general strategy for solving fuzzy equations will be:

1. the solution is $\overline{X}_c$ when it exists;
2. if $\overline{X}_c$ fails to exist, the solution is $\overline{X}_e$; and
3. if $\overline{X}_c$ fails to exist and $\overline{X}_e$ is difficult to construct, use $\overline{X}_I$ as the (approximate) solution.

For more complicated fuzzy equations $\overline{X}_e$ will be difficult to compute. However, $\overline{X}_I$ is usually easily constructed, since it uses only max, min and the arithmetic of real numbers. For this reason we suggest approximating $\overline{X}_e$ by $\overline{X}_I$ when we do not have $\overline{X}_e$.

Example 10.2.1.1

This continues Example 10.2.1 where $\overline{X}_c$ does not exist. To calculate $\overline{X}_e$ we need to evaluate equations (10.13) and (10.14). But this is easily done since $(c-b)/a$ is increasing in c and decreasing in both b and a. So

$$x_{e1}(\alpha) = \frac{c_1(\alpha) - b_2(\alpha)}{a_2(\alpha)} = \frac{4+2\alpha}{3-\alpha}, \tag{10.17}$$

$$x_{e2}(\alpha) = \frac{c_2(\alpha) - b_1(\alpha)}{a_1(\alpha)} = \frac{8-2\alpha}{1+\alpha}. \tag{10.18}$$

$\overline{X}_e$ is shown in Figure 10.2.

In calculating $\overline{X}_I[\alpha]$ we get

$$\overline{X}_I[\alpha] = [c_1(\alpha) - b_2(\alpha), c_2(\alpha) - b_1(\alpha)] \left[\frac{1}{a_2(\alpha)}, \frac{1}{a_1(\alpha)}\right], \tag{10.19}$$

which is the same as $\overline{X}_e[\alpha]$ because intervals in equation (10.19) are positive. In this example, we get $\overline{X}_e = \overline{X}_I$, whose support $\overline{X}_e[0] = [\frac{4}{3}, 8]$.

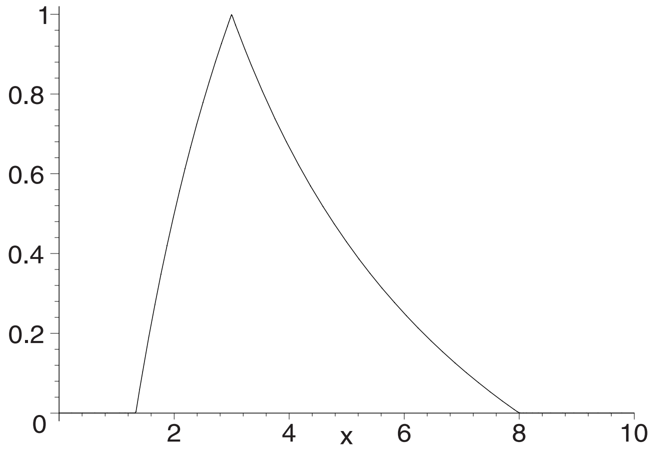


Fig. 10.2. Solution to Example 10.2.1.1, $\overline{X}_e = \overline{X}_I \approx (\frac{4}{3}/3/8)$

Example 10.2.1.2

This continues Example 10.2.2. We notice that, since $\overline{A} > 0$ and $\overline{C} - \overline{B} > 0$, $\frac{\partial}{\partial c}[\frac{c-b}{a}] = \frac{1}{a} > 0$, $\frac{\partial}{\partial b}[\frac{c-b}{a}] = -\frac{1}{a} < 0$ and $\frac{\partial}{\partial a}[\frac{c-b}{a}] = \frac{b-c}{a^2} < 0$. This means that the expression $\frac{c-b}{a}$ is increasing in c but decreasing in a and b. Then equations (10.13) and (10.14) become

$$x_{e1}(\alpha) = \frac{c_1(\alpha) - b_2(\alpha)}{a_2(\alpha)} = \frac{4+3\alpha}{10-\alpha}, \tag{10.20}$$

$$x_{e2}(\alpha) = \frac{c_2(\alpha) - b_1(\alpha)}{a_1(\alpha)} = \frac{10 - 3\alpha}{8 + \alpha}. \tag{10.21}$$

As in Example 10.2.1.1 we obtain $\overline{X}_I = \overline{X}_e$. $\overline{X}_c$ and $\overline{X}_e$ are shown in Figure 10.3. The support of $\overline{X}_e = \overline{X}_I$ is $\overline{X}_e[0] = [0.4, 1.25]$.

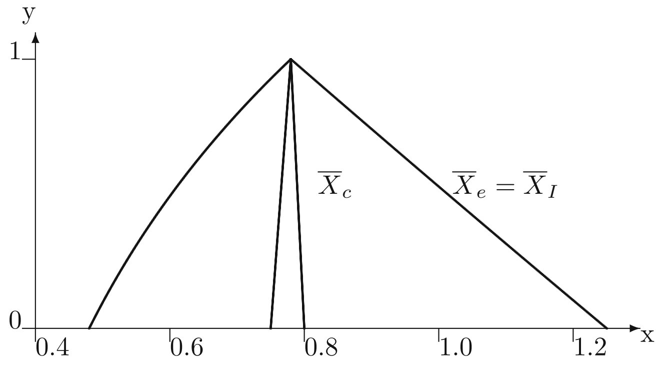


Fig. 10.3. Solutions to Example 10.2.1.2, $\overline{X}_c \approx (0.75/0.77\overline{7}/0.8)$ and $\overline{X}_e \approx (0.4/0.77\overline{7}/1.25)$

10.2.2 Fuzzy Monte Carlo Method

Let $\mathbf{Q}$ be the set of triangular fuzzy numbers and the set of quadratic fuzzy numbers (Section 4.3.2 called QBGFNs). Let D be some metric on $\mathbf{Q}$. Then D has the following properties: for $\overline{M}$, $\overline{N}$ and $\overline{P}$ in $\mathbf{Q}$

1. $D(\overline{M}, \overline{N}) \geq 0$;
2. $D(\overline{M}, \overline{N}) = 0$ implies that $\overline{M} = \overline{N}$;
3. $D(\overline{M}, \overline{N}) = D(\overline{N}, \overline{M})$; and
4. $D(\overline{M}, \overline{N}) \leq D(\overline{M}, \overline{P}) + D(\overline{P}, \overline{N})$.

Then our new solution will be $\overline{X}^*$ that solves the minimization problem

$$min\{D(\overline{A} \cdot \overline{X} + \overline{B}, \overline{C}) | \overline{X} \in \mathbf{Q}\}. \tag{10.22}$$

That is, $\overline{X}^*$ is a fuzzy number from $\mathbf{Q}$ that makes $\overline{A} \cdot \overline{X}^* + \overline{B}$ as close as possible to $\overline{C}$, where the distance is measured by the metric D.

Our fuzzy Monte Carlo method may be applied to approximate $\overline{X}^*$. Randomly generate a sequence $\overline{X}_1$, $\overline{X}_2$,... from $\mathbf{Q}$, compute the distance D between $\overline{A} \cdot \overline{X}_i + \overline{B}$ and $\overline{C}$, and keep the $\overline{X}_i$ that makes this distance the smallest. In this way we may compute better and better, for longer and longer sequences, approximations to $\overline{X}^*$.

Now we will rework the examples in the previous section using our fuzzy Monte Carlo method and compare the results. But first we need to select a metric D. Of course, the answers can vary depending on the choice of D. Metrics on fuzzy numbers were discussed in [8]. We first give an example of a "horizontal" metric and then an example of a "vertical" metric.

Let $\overline{M}[\alpha] = [m_1(\alpha), m_2(\alpha)]$, $\overline{N}[\alpha] = [n_1(\alpha), n_2(\alpha)]$, $L(\alpha) = |m_1(\alpha) - n_1(\alpha)|$ and $R(\alpha) = |m_2(\alpha) - n_2(\alpha)|$. Then

$$D(\overline{M}, \overline{N}) = max\{max(L(\alpha), R(\alpha))|0 \leq \alpha \leq 1\}, \tag{10.23}$$

is a metric. In [8] the authors show that $D(\overline{M}, \overline{N}) = 1$ for $\overline{M} = (1/2/4)$ and $\overline{N} = (1/3/4)$. For triangular/trapezoidal fuzzy numbers it is easy to compute the distance between them using equation (10.23). It is

$$D(\overline{M}, \overline{N}) = max\{|m_1(0) - n_1(0)|, |m_1(1) - n_1(1)|, |m_2(0) - n_2(0)|\}, \tag{10.24}$$

Also, the Hamming distance measure in equation (2.45) in Chapter 2 is a metric. Also in [8] the authors argue that the Hamming distance between $\overline{M} = (1/2/4)$ and $\overline{N} = (1/3/4)$ is 0.75. Let us use the first metric, the "horizontal metric" in our fuzzy Monte Carlo studies.

We will use the metric defined by equation (10.23) in equation (10.22). Let $D(\overline{X}) = D(\overline{A} \cdot \overline{X} + \overline{B}, \overline{C})$ and let $\epsilon \in (0, 1]$ be the "threshold". If the fuzzy Monte Carlo method produces a $\overline{X}$ so that $D(\overline{X}) < \epsilon$ we will say that we have found an acceptable approximate solution $\overline{X}_a^* = \overline{X}$ with $\overline{X}_a^* \approx \overline{X}^*$. We begin with $D(\overline{X})$ very large (100) and will accept a solution only if it minimizes to $D(\overline{X}) < \epsilon = 0.5$.

Example 10.2.2.1

This continues Examples 10.2.1 and 10.2.1.1. We wish to use our fuzzy Monte Carlo method to compute $\overline{X}_a^* \approx \overline{X}^*$ and then compare $\overline{X}_a^*$ to the other solution $\overline{X}_e = \overline{X}_I$. Recall that in this example the classical solution $\overline{X}_c$ does not exist and $\overline{X}_a^*$ is in $\mathbf{Q}$ and must satisfy $D(\overline{X}_a^*)$ small; e.g., $D(\overline{X}_a^*) < \epsilon = 0.5$.

We have already generated and studied 100,000 crisp random vectors $v = (x_1, ..., x_5)$ in $[0, 1]^5$ using a Sobol quasi-random number generator (Chapter 3). In Section 4.3.2 we relate how we create our vectors v. Next we determine an interval $[a, b]$, which will depend on the application, for the random quadratic fuzzy numbers. Then we map v into a QBGFN.

To map v into a QBGFN (Figure 4.4), first we sort, translate and transform $\{x_1, x_2, x_3\}$ to $\{z_1, z_2, z_3\}$ using $z_i = (b-a)x_i + a$, $i = 1, 2, 3$. Additionally we map x_4, x_5 into parameters for the left and right membership functions, respectively, using $z_4 = (2x_4 - 1)(z_2 - z_1 + 1)$ and $z_5 = (2x_5 - 1)(z_3 - z_2 + 1)$. In Section 5.3 we explain why we know these QBGFNs will cover our search space.

We modified the fuzzy Monte Carlo program used for Chapters 6-9 to optimize the minimization problem of equation (10.22). Since for this problem the classical solution does not exist, we choose $[a, b]$ as the support of $\overline{X}_e = \overline{X}_I$, which is $[\frac{4}{3}, 8]$. To compute $D(\overline{A} \cdot \overline{X} + \overline{B}, \overline{C})$, we compute 100 α-cuts of $\overline{A} \cdot \overline{X} + \overline{B}$ and $\overline{C}$ for given $\overline{A}$, $\overline{B}$ and $\overline{C}$, where $\overline{X}$ is one of 30,000 generated random quadratic fuzzy numbers.

Table 10.1. $(1/2/3) \cdot \overline{X} + (-3/-2/-1) = (3/4/5)$, Example 10.2.1, First Interval

Solution	$\overline{X}$	$\overline{A} \cdot \overline{X} + \overline{B}$
$\overline{X}_c$	does not exist	$(3/4/5) = \overline{C}$
$\overline{X}_I = \overline{X}_e$	$\approx (1.33\overline{3}/3/8)$	$\approx (-1.66\overline{6}/4/23)$
$\overline{X}^*_a$,	$(2.9655/2.9980/3.0176/0.1286/0.6457)$, $D(\overline{X}) = 3.0527$	$\approx (-0.03/4.00/8.05)$

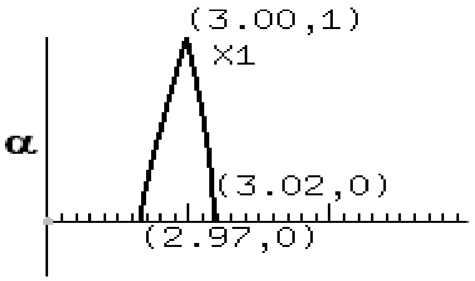


Fig. 10.4. $\overline{X}^*_a$ of Example 10.2.2.1

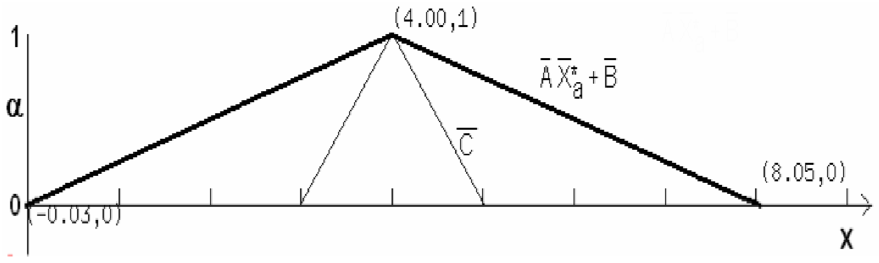


Fig. 10.5. $\overline{A} \cdot \overline{X}^*_a + \overline{B}$ of Example 10.2.2.1

We allow our program to execute for $30,000$ iterations and obtain the results shown in Table 10.1.

Searching for $\overline{X}$ in the interval $[1.33, 8.00]$, the smallest "horizontal" distance for equation (10.23) that we got was 3.0527. Increasing the numbers of fuzzy numbers to $50,000$ does not produce a change. In other words we obtained a quadratic fuzzy number $\overline{X}^*_a \approx (2.9655/2.9980/3.0176)$ (Figure 10.4) such that $D(\overline{A} \cdot \overline{X}^*_a + \overline{B}, \overline{C}) \approx 3.0527$. The graph of $\overline{X}^*_a$ is nearly crisp. $\overline{A} \cdot \overline{X}^*_a + \overline{B} \approx (-0.03/4.00/8.05)$. But this error is too large.

We changed the interval $[a, b]$ and saw that there is a good reduction in the distance between $\overline{A} \cdot \overline{X} + \overline{B}$ and $\overline{C}$. From an optimization over $[0, 5]$ we minimized at $D(\overline{A} \cdot \overline{X} + \overline{B}, \overline{C}) = 2.00$ for $\overline{X} = \overline{X}^*_a = (2.5000, 2.5000, 2.5000, -0.0000, 0.5000)$. This error is still too large with $\epsilon = 2.00$ and $\overline{A} \cdot \overline{X}^*_a + \overline{B} \approx (1.00/3.00/5.00)$ (Figure 10.5).

Our fuzzy Monte Carlo method was unable to get an acceptable approximate solution to this fuzzy equation. We believe that no quadratic fuzzy number $\overline{X}$ can make $D(\overline{X}) < \epsilon = 0.5$.

Example 10.2.2.2

This continues Examples 10.2.2 and 10.2.1.2. We apply our fuzzy Monte Carlo method to compute $\overline{X}_a^* \approx \overline{X}^*$ and then compare $\overline{X}_a^*$ to the other solutions $\overline{X}_c$ and $\overline{X}_e = \overline{X}_I$. Recall that in this example the classical solution exists and is shown in Figure 10.1. The constraints are still $\overline{X}_a^* \in \mathbf{Q}$ and it must satisfy $D(\overline{X}_a^*)$ small; e.g., $D(\overline{X}_a^*) < \epsilon = 0.5$. As in Example 10.2.2.1 we generate a solution using a Sobol quasi-random number generator to produce random quadratic

Table 10.2. $(8/9/10) \cdot \overline{X} + (-3/-2/-1) = (3/5/7)$, Example 10.2.2.2

Solution	$\overline{X}$	$\overline{A} \cdot \overline{X} + \overline{B}$
$\overline{X}_c$	$\approx (0.75/0.77\overline{7}/0.80)$	$\approx (3/5/7) = \overline{C}$
$\overline{X}_I = \overline{X}_e$	$\approx (0.4/0.77\overline{7}/1.25)$	$\approx (0.2/5/11.5)$
$\overline{X}_a^*$,	$(0.7523, 0.7786, 0.7999, 0.7788, 0.7816)$,	$\approx (3.02/5.01/7.00)$
	$D(\overline{X}) = 0.083976$	

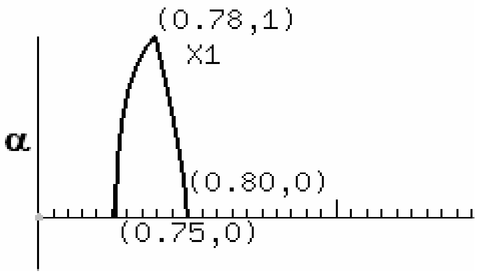


Fig. 10.6. $\overline{X}_a^*$ of Example 10.2.2.2

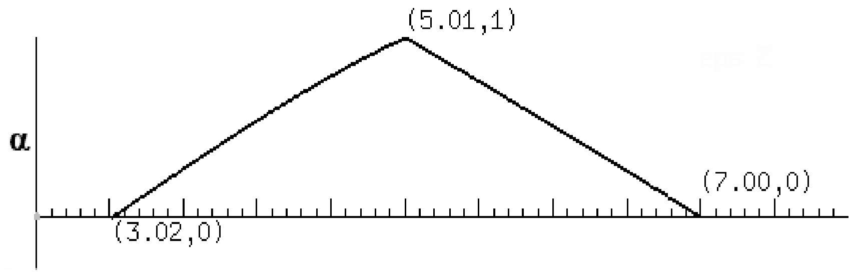


Fig. 10.7. $\overline{A} \cdot \overline{X}_a^* + \overline{B}$ of Example 10.2.2.2

fuzzy numbers determined by vectors of length 5. We consider fuzzy numbers in $[0.4, 1.25]$ which is the support of $\overline{X}_e = \overline{X}_I$.

We allow our program to execute for $50,000$ iterations and obtain the results shown in Table 10.2. $\overline{X}_a^*$ (Figure 10.6) is very close to $\overline{X}_c$ (Figure 10.1). Note too how $\overline{A} \cdot \overline{X}_a^* + \overline{B}$ (Figure 10.7) matches $\approx (3/5/7)$. Our fuzzy Monte Carlo method found an acceptable solution.

10.3 Fuzzy Quadratic Equation

In this section we wish to discuss solutions to

$$\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C} = \overline{D}, \tag{10.25}$$

for triangular fuzzy numbers $\overline{A}, \overline{B}, \overline{C}, \overline{D}$ and $\overline{X}$ a triangular shaped fuzzy number. We know a crisp quadratic equation can have real number solutions and complex number solutions. The same is true of the fuzzy quadratic. However, we will not consider fuzzy complex numbers in this book so we will only work with fuzzy quadratics that have no solution or the solutions are real triangular shaped fuzzy numbers. In Section 10.2 we looked at three possible solutions to the fuzzy linear equation: classical $(\overline{X}_c)$, extension principle $(\overline{X}_e)$ and the interval arithmetic $(\overline{X}_I)$. In this section we only consider the classical solution.

Let $\overline{A}[\alpha] = [a_1(\alpha), a_2(\alpha)]$, $\overline{B}[\alpha] = [b_1(\alpha), b_2(\alpha)]$, $\overline{C}[\alpha] = [c_1(\alpha), c_2(\alpha)]$, $\overline{D}[\alpha] = [d_1(\alpha), d_2(\alpha)]$, and $\overline{X}[\alpha] = [x_1(\alpha), x_2(\alpha)]$. We use $\alpha-$cuts and interval arithmetic to solve for $x_1(\alpha)$ and $x_2(\alpha)$. Equation (10.25) becomes

$$\begin{aligned}[a_1(\alpha), a_2(\alpha)][x_1(\alpha), x_2(\alpha)]^2 + [b_1(\alpha), b_2(\alpha)][x_1(\alpha), x_2(\alpha)] +& \\ +[c_1(\alpha), c_2(\alpha)] = [d_1(\alpha), d_2(\alpha)],&\end{aligned} \tag{10.26}$$

for all α. We do the interval arithmetic (Section 2.3.2), which depends on $\overline{A}, \overline{B}$ and $\overline{X}$ being positive or negative, and solve for $x_1(\alpha)$ and $x_2(\alpha)$. We have a solution if $x_1(0) < x_1(1) \leq x_2(1) < x_3(0)$ and $dx_1(\alpha)/d\alpha > 0$, $dx_2(\alpha)/d\alpha < 0$.

Now we look at two examples where the first has a solution and the second does not have a solution. More details on the fuzzy quadratic can be found in ([6]-[9],[11],[13]). Then we turn to our fuzzy Monte Carlo method to see what approximate answers it can give.

Example 10.3.1

Let $\overline{A} = (3/4/5)$, $\overline{B} = (1/2/3)$, $\overline{C} = (0/1/2)$ and $\overline{D} = (1/3/5)$. We will look for a solution where $\overline{X} \geq 0$. Then equation (10.26) becomes

$$\begin{aligned}[3+\alpha, 5-\alpha][x_1^2(\alpha), x_2^2(\alpha)] + [1+\alpha, 3-\alpha][x_1(\alpha), x_2(\alpha)]& \\ +[\alpha, 2-\alpha)] = [1+2\alpha, 5-2\alpha)],&\end{aligned} \tag{10.27}$$

for $\alpha \in [0,1]$. If $\overline{X} \approx (x_1/x_2/x_3)$ we first solve for the x_i getting $x_1 = 0.4343 < x_2 = 0.5000 < x_3 = 0.5307$. Looks like we will get a solution. Next we look at $x_1(\alpha)$ which is

$$x_1(\alpha) = \frac{-(1+\alpha) + \sqrt{5\alpha^2 + 18\alpha + 13}}{6 + 2\alpha}, \tag{10.28}$$

and then $x_2(\alpha)$

$$x_2(\alpha) = \frac{-(3-\alpha) + \sqrt{5\alpha^2 - 38\alpha + 69}}{10 - 2\alpha}, \tag{10.29}$$

for $0 \leq \alpha \leq 1$. We find that $dx_1(\alpha)/d\alpha > 0$ and $dx_2(\alpha)/d\alpha < 0$ and $\overline{X}$ is a solution. The graph of this solution is in Figure 10.8.

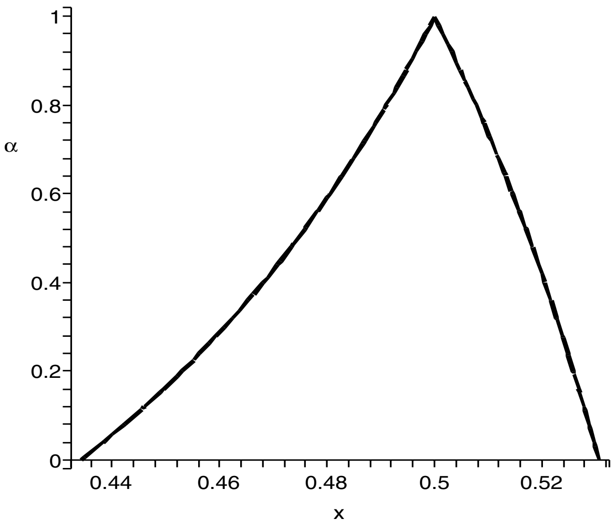


Fig. 10.8. Solution to Example 10.3.1, $\overline{X}_c \approx (0.4343/0.5/0.5307)$

Example 10.3.2

This example will have no (classical) solution for $\overline{X}$. Let $\overline{A} = (2/4/6)$, $\overline{B} = (0/2/4)$, $\overline{C} = 0$, $\overline{D} = (0.5/1/1.5)$ and $\overline{X}$ a non-negative triangular shaped fuzzy number. Let $\overline{X} \approx (x_1/x_2/x_3)$. So $x_1(0) = x_1$ and $x_2(0) = x_3$. Now we set up an equation, like equation (10.27), for $\alpha = 0$ and obtain two equations to solve

$$2x_1^2(0) = 0.5, \tag{10.30}$$

and

$$6x_2^2(0) + 4x_2(0) = 1.5. \tag{10.31}$$

If $\overline{X}$were a solution, its support would be $[x_1(0), x_2(0)]$. However $[x_1(0), x_2(0)] = [0.5, 0.2676]$; i.e., $x_2(0) < x_2(0)$. Thus there is no classical solution.

10.3.1 Fuzzy Monte Carlo Method

We follow the same strategy as outlined in Section 10.2.2 for generating approximate solutions to fuzzy linear equations using fuzzy Monte Carlo methods. **Q** will be the same set of fuzzy numbers and we use the same metric between fuzzy numbers given in equation (10.23). We call our new solution $\overline{X}^*$, the fuzzy number in **Q** that minimizes the distance $D(\overline{X})$ between $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$ and $\overline{D}$. Using our fuzzy Monte Carlo method we obtain $\overline{X}_a^*$ an approximation to $\overline{X}^*$. We use the same "threshold" ϵ discussed in Section 10.2.2. If the fuzzy Monte Carlo method produces a $\overline{X}$ so that $D(\overline{X}) < \epsilon$ we will say that we have found an acceptable approximate solution $\overline{X}_a^* = \overline{X}$ with $\overline{X}_a^* \approx \overline{X}^*$. We begin with $D(\overline{X})$ very large (100) and will accept a solution only if it minimizes to $D(\overline{X}) < \epsilon = 0.5$.

Now we will rework Examples 10.3.1 and 10.3.2 using our fuzzy Monte Carlo method.

Example 10.3.1.1

This continues Example 10.3.1. We wish to use our fuzzy Monte Carlo method to compute $\overline{X}_a^* \approx \overline{X}^*$ and then compare $\overline{X}_a^*$ to the classical solution $\overline{X}_c$ in Figure 10.8. Recall that in this example the classical solution $\overline{X}_c$ exists and $\overline{X}_a^*$ is in **Q** and must satisfy $D(\overline{X}_a^*) < \epsilon = 0.5$. We follow the same procedure discussed in Example 10.2.2.1.

We have already generated and studied 100,000 crisp random vectors $v = (x_1, ..., x_5)$ in $[0, 1]^5$ using a Sobol quasi-random number generator (Chapter 3). In Section 4.3.2 we relate how we create our vectors v. Next we determine an interval $[a, b]$, which will depend on the application, for the random quadratic fuzzy numbers. Then we map v into a QBGFN.

Since for this problem the classical solution does exist, we choose $[a, b]$ as the support of $\overline{X}_c$, which is $[0.4343, 0.5307]$. To compute $D(\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}, \overline{D})$, we compute 100 α-cuts of $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$ and $\overline{D}$ for given $\overline{A}$, $\overline{B}$, $\overline{C}$ and $\overline{D}$, where $\overline{X}$ is one of 50,000 generated random quadratic fuzzy numbers.

Searching for $\overline{X}$ in the interval $[0.4343, 0.5307]$, the smallest "horizontal" distance for equation (10.23) that we got in $50,000$ iterations was 0.014840 (Table 10.3), in the 19931^{th} iteration. However, we found $D(\overline{X}) < 0.20$ on the 7^{th} iteration. Figure 10.9 shows $\overline{X}_a^*$; Figure 10.10 shows $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$ for $\overline{X} = \overline{X}_a^*$.

Wanting to investigate how a change in the interval $[a, b]$ might affect our result, we executed a test using $[a, b] = [0, 1]$. and also saw a very good reduction in the distance between $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$ and $\overline{D}$. The graphs of these two solutions are so similar that we do not show graphs for this follow-up experiment. As early as the 637^{th} iteration, an acceptable solution with $D(\overline{X}) = 0.193726$ was found. Although we performed $50,000$ iterations, our acceptable solution was found on the $20,726^{th}$ iteration. From an optimization over $[0, 1]$ we minimized at $D(\overline{X}) = 0.036417$ (Table 10.3).

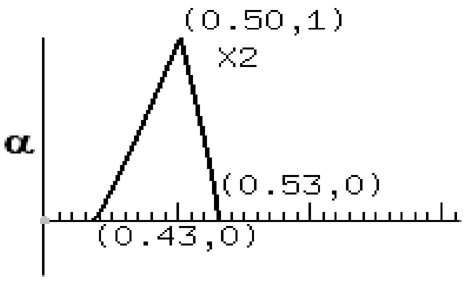


Fig. 10.9. $\overline{X}_a^*$ of Example 10.3.1.1

Table 10.3. $(3/4/5) \cdot \overline{X}^2 + (1/2/3) \cdot \overline{X} + (0/1/2) = (1/3/5)$, Example 10.3.1.1

Solution	$\overline{X}$	$\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$
$\overline{X}_c$	$\approx (0.4343/0.5/0.5307)$	$\approx (1/3/5) = \overline{D}$
$\overline{X}_a^*$,	$[a,b] = [0.4343, 0.5307]$ $(0.4348, 0.5016, 0.5306, -0.0074, 0.8749)$, $D(\overline{X}) = 0.014840$	$\approx (1.00/3.01/5.00)$
$\overline{X}_a^*$,	$[a,b] = [0.0, 1.0]$ $(0.4419, 0.5021, 0.5341, 0.0980, 0.9120)$, $D(\overline{X}) = 0.036417$	$\approx (1.03/3.01/5.03)$

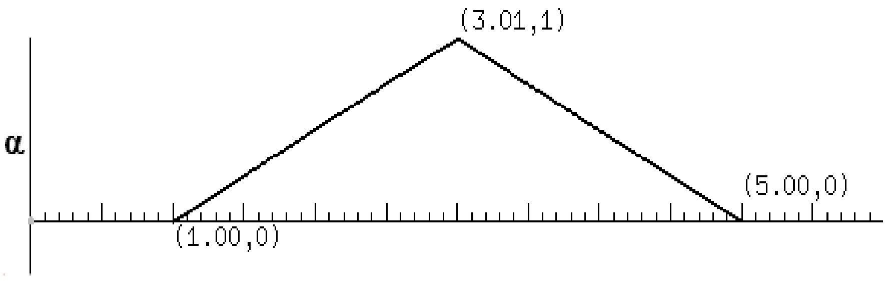


Fig. 10.10. $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$ for $\overline{X} = \overline{X}_a^*$ of Example 10.3.1.1

Example 10.3.1.2

This continues Examples 10.3.2. We apply our fuzzy Monte Carlo method to compute $\overline{X}_a^* \approx \overline{X}^*$. Recall that in this example the classical solution does not exist. The constraints are still $\overline{X}_a^* \in \mathbf{Q}$ and it must satisfy $D(\overline{X}_a^*) < \epsilon = 0.5$. We follow the same procedure discussed in Example 10.3.1.1.

Table 10.4. $(2/4/6) \cdot \overline{X}^2 + (0/2/4) \cdot \overline{X} + (0/0/0) = (0.5/1/1.5)$, Example 10.3.2

Solution	$\overline{X}$	$\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$
$\overline{X}_c$	does not exist	$(0.5/1/1.5) = \overline{D}$
$\overline{X}^*_a$,	$(0.3058, 0.3088, 0.3107, -0.3767, -0.8683)$, $D(\overline{X}) = 0.321659$	$\approx (0.19/1.00/1.82)$

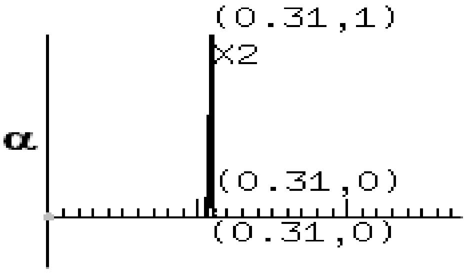


Fig. 10.11. $\overline{X}^*_a$ of Example 10.3.1.2

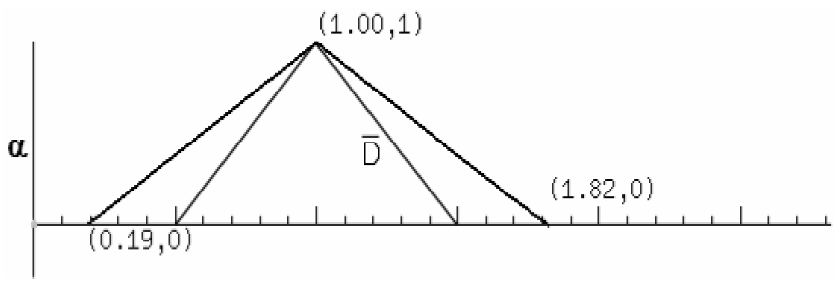


Fig. 10.12. $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C}$ for $\overline{X} = \overline{X}^*_a$ of Example 10.3.1.2

Now we need to find an interval $[a, b]$ for our random quadratic fuzzy numbers. The classical solution does not exist. Choose the vertex values for the $\overline{A}$,...,$\overline{D}$ and consider $4x^2 + 2x + 0 = 1$ which has positive solution $x = 0.3090$. We begin with intervals centered at this value so we begin with the interval $I = [0, 0.6180]$.

To compute $D(\overline{A}{\cdot}\overline{X}^2{+}\overline{B}{\cdot}\overline{X}{+}\overline{C}, \overline{D})$, we compute 100 α-cuts of $\overline{A}{\cdot}\overline{X}^2{+}\overline{B}{\cdot}\overline{X}{+}\overline{C}$ and $\overline{D}$ for given $\overline{A}$, $\overline{B}$, $\overline{C}$ and $\overline{D}$, where $\overline{X}$ is one of 50,000 generated random quadratic fuzzy numbers.

From an optimization over $[0, 0.6180]$ we minimized at $D(\overline{X}) = 0.3217$ (Table 10.4 and Figure 10.12). Our fuzzy Monte Carlo result generated an "acceptable" solution. Perhaps we could reduce the error measure with more iterations.

10.4 Fuzzy Matrix Equation

This section is based on ([6],[8],[12],[13]). Let $\overline{A} = [\overline{a}_{ij}]$ be a $n \times n$ matrix of triangular fuzzy numbers $\overline{a}_{ij}$, $\overline{B}^t = (\overline{b}_1, \ldots, \overline{b}_n)$ a $n \times 1$ vector of triangular fuzzy numbers $\overline{b}_i$ and $\overline{X}^t = (\overline{x}_1, \ldots, \overline{x}_n)$ a $n \times 1$ vector of unknown triangular shaped fuzzy numbers $\overline{x}_j$. Set $\overline{a}_{ij} = (a_{ij1}/a_{ij2}/a_{ij3})$, $\overline{b}_i = (b_{i1}/b_{i2}/b_{i3})$, and $\overline{x}_j \approx (x_{j1}/x_{j2}/x_{j3})$. We wish to solve

$$\overline{A}\,\overline{X} = \overline{B}, \tag{10.32}$$

for $\overline{X}$.

We need to introduce some more notation. Define

$$a[\alpha] = \prod_{i,j=1}^{n} a_{ij}[\alpha], \tag{10.33}$$

$$b[\alpha] = \prod_{i=1}^{n} b_i[\alpha], \tag{10.34}$$

for $0 \le \alpha \le 1$. Let $v = (a_{11}, a_{12}, \ldots, a_{nn}) \in \mathbb{R}^k$, $k = n^2$, be a vector in $a[0]$. Each $v \in a[0]$ determines a crisp $n \times n$ matrix $A = [a_{ij}]$. Also, $b^t = (b_1, \ldots, b_n) \in \mathbb{R}^n$ is a vector in $b[0]$. As in our previous research on this topic we assume A^{-1} exists for all v in $a[0]$. The existence of A^{-1} over $a[0]$ simplifies the discussion of the joint solution to be introduced below.

The joint solution $\overline{X}_J$, a fuzzy subset of $\mathbb{R}^n$, is based on the extension principle

$$\overline{X}_J(x) = \max\left\{\, \pi(v,b) \mid x = A^{-1}b \,\right\}, \tag{10.35}$$

where

$$\pi(v,b) = \min\left\{\, \overline{a}_{ij}(a_{ij}), \overline{b}_i(b_i) \mid \text{all } i,j \,\right\}. \tag{10.36}$$

The vertex of $\overline{X}_J(x)$, where the membership value is equal to one, is at $x = A^{-1}b$ for $v = (a_{112}, a_{122}, \ldots, a_{nn2})$, $b^t = (b_{12}, \ldots, b_{n2})$. In the crisp case the solution to $Ax = b$ is a vector $x = A^{-1}b$ in $\mathbb{R}^n$, so for the fuzzy case $\overline{A}\ \overline{X} = \overline{B}$, the (joint) solution is a fuzzy vector about the crisp solution $A^{-1}b$, for v and b at the vertex values of all the triangular fuzzy numbers.

In the crisp case the marginals, the x_i, are just the components of the vector $x = A^{-1}b$. In the fuzzy case we obtain the marginals $\overline{X}_{Ji}$ by projecting $\overline{X}_J$ onto the coordinate axes. Then

$$\overline{X}_{Ji}(w) = \max\left\{\overline{X}_J(x) \mid x \in \mathbb{R}^n, x_i = w \,\right\}, \tag{10.37}$$

for $1 \le i \le n$. Obviously, it will be difficult to compute $\overline{X}_J$ and $\overline{X}_{Ji}$, $1 \le i \le n$, for $n \ge 4$. We will determine the joint solution, and its marginals, in two examples at the end of this section for $n = 2$.

Since $\overline{X}_J$ is difficult to determine we now turn to methods of finding the marginals directly without first computing the joint solution. As in the

Section 10.2 there will be three solutions: (1) the classical solution $\overline{X}_{ci}$; (2) the extension principle solution $\overline{X}_{ei}$; and (3) the interval arithmetic solution $\overline{X}_{Ii}$.

The classical solution is determined by substituting the intervals $\overline{a}_{ij}[\alpha]$, $\overline{b}_i[\alpha]$ and $\overline{X}_i[\alpha] = [x_{i1}(\alpha), x_{i2}(\alpha)]$ into $\overline{A}\ \overline{X} = \overline{B}$ and solving for the $x_{i1}(\alpha)$, $x_{i2}(\alpha)$, $1 \leq i \leq n$. The resulting equations are evaluated using interval arithmetic. If the intervals $[x_{i1}(\alpha), x_{i2}(\alpha)]$ define a triangular shaped fuzzy number $\overline{X}_i$ for $0 \leq \alpha \leq 1$, $1 \leq i \leq n$, then this solution is called the classical solution and we write $\overline{X}_{ci} = \overline{X}_i$, $1 \leq i \leq n$. The conditions for $[x_{i1}(\alpha), x_{i2}(\alpha)]$ to define $\overline{X}_{ci}$ were discussed in Section 10.2. The equations to solve for $x_{i1}(\alpha)$ and $x_{i2}(\alpha)$ are

$$\sum_{j=1}^{n}[a_{ij1}(\alpha), a_{ij2}(\alpha)][x_{j1}(\alpha), x_{j2}(\alpha)] = [b_{i1}(\alpha), b_{i2}(\alpha)], \tag{10.38}$$

for $1 \leq i \leq n$, where $\overline{a}_{ij}[\alpha] = [a_{ij1}(\alpha), a_{ij2}(\alpha)]$, $\overline{b}_i[\alpha] = [b_{i1}(\alpha), b_{i2}(\alpha)]$. After using interval arithmetic we obtain a $(2n) \times (2n)$ system to solve for $x_{i1}(\alpha)$, $x_{i2}(\alpha)$, $0 \leq \alpha \leq 1$.

Too often the $\overline{X}_{ci}$ fail to exist. We only need $\overline{X}_{ci}$, for one value of i, to fail to exist for the classical solution to not exist. When the classical solution does not exist we turn to $\overline{X}_{ei}$, $1 \leq i \leq n$.

We will use Cramer's rule on $Ax = b$ to solve for each x_i. A comes from $v \in a[0]$ and let $b \in b[0]$. Let A_j be A with its j–th column replaced by b. Then

$$x_j = \frac{|A_j|}{|A|}, \tag{10.39}$$

$1 \leq j \leq n$, where $|\cdot|$ denotes the determinant. We fuzzify equation (10.39), using the extension principle, to get

$$\overline{X}_{ej}(x_j) = \max\{\pi(v,b) \mid x_j = |A_j|/|A|\}, \tag{10.40}$$

$1 \leq j \leq n$. If $\overline{X}_{ej}[\alpha] = [x_{ej1}(\alpha), x_{ej2}(\alpha)]$, we may find the α-cuts of $\overline{X}_{ej}$ as (Section 2.4.1)

$$x_{ej1}[\alpha] = \min\left\{\frac{|A_j|}{|A|} \mid v \in a[\alpha], b \in b[\alpha]\right\}, \tag{10.41}$$

$$x_{ej2}[\alpha] = \max\left\{\frac{|A_j|}{|A|} \mid v \in a[\alpha], b \in b[\alpha]\right\}, \tag{10.42}$$

To get the $\overline{X}_{Ij}$ we evaluate equation (10.39) using α-cuts and interval arithmetic. Substitute intervals $\overline{a}_{ij}[\alpha]$ and $\overline{b}_i[\alpha]$ for a_{ij} and b_i in $|A_j|/|A|$, evaluate using interval arithmetic, and the result is $\overline{X}_{Ij}[\alpha]$, $0 \leq \alpha \leq 1$, $1 \leq j \leq n$.

We have the following result: If the $\overline{X}_{ci}$ exist, $1 \leq i \leq n$, then $\overline{X}_{ci} \leq \overline{X}_{Ji} \leq \overline{X}_{ei} \leq \overline{X}_{Ii}$, $1 \leq i \leq n$.

Our solution strategy is: (1) use $\overline{X}_{ci}$, $1 \leq i \leq n$, if it exists; (2) if the classical solution does not exist use $\overline{X}_{Ji}$, $1 \leq i \leq n$. However, if the joint solution is too difficult to compute use $\overline{X}_{ei}$, $1 \leq i \leq n$. Equations (10.41) and (10.42) may be hard to evaluate to get the $\overline{X}_{ei}$. One can always use the $\overline{X}_{Ii}$ because they are the easiest to calculate. Notice how the fuzziness grows (the supports do not decrease) as we go from $\overline{X}_{ci}$ to $\overline{X}_{Ii}$. The only solution guaranteed to satisfy the fuzzy equations is the classical solution.

In the following two examples we only consider 2×2 fuzzy matrices since then we can easily see pictures of α-cuts of the joint solution. It is known that, in general, α-cuts of the joint solution need not be convex ([19],[20]). For example, in two dimensions $\overline{X}_J[\alpha]$ need not be a rectangle.

Example 10.4.1

Let

$$\overline{A} = \begin{pmatrix} \overline{a}_{11} & 0 \\ 0 & \overline{a}_{22} \end{pmatrix}, \tag{10.43}$$

and $\overline{B}^t = (\overline{b}_1, \overline{b}_2)$, where $\overline{a}_{11} = (4/5/7)$, $\overline{a}_{22} = (6/8/12)$, $\overline{b}_1 = (1/2/3)$ and $\overline{b}_2 = (2/5/8)$.

Then $\overline{a}_{11}[\alpha] = [4+\alpha, 7-2\alpha]$, $\overline{a}_{22}[\alpha] = [6+2\alpha, 12-4\alpha]$, $\overline{b}_1[\alpha] = [1+\alpha, 3-\alpha]$ and $\overline{b}_2[\alpha] = [2+3\alpha, 8-3\alpha]$. Since all the fuzzy numbers are positive we will solve for $\overline{X}_{ci} > 0$, $i = 1, 2$. The equations are

$$[4+\alpha, 7-2\alpha] \cdot [x_{c11}(\alpha), x_{c12}(\alpha)] = [1+\alpha, 3-\alpha], \tag{10.44}$$

$$[6+2\alpha, 12-4\alpha] \cdot [x_{c21}(\alpha), x_{c22}(\alpha)] = [2+3\alpha, 8-3\alpha], \tag{10.45}$$

which define triangular shaped fuzzy numbers,

$$\overline{X}_{c1}[\alpha] = \left[\frac{1+\alpha}{4+\alpha}, \frac{3-\alpha}{7-2\alpha}\right], \tag{10.46}$$

$$\overline{X}_{c2}[\alpha] = \left[\frac{2+3\alpha}{6+2\alpha}, \frac{8-3\alpha}{12-4\alpha}\right], \tag{10.47}$$

$0 \leq \alpha \leq 1$.

We have shown before that a way to find α-cuts of $\overline{X}_J$ is

$$\overline{X}_J[\alpha] = \{A^{-1}b \mid v \in a[\alpha], b \in b[\alpha] \}. \tag{10.48}$$

Now $A^{-1}b = (b_1/a_{11}, b_2/a_{22})^t$ so that

$$\overline{X}_J[\alpha] = \frac{\overline{b}_1[\alpha]}{\overline{a}_{11}[\alpha]} \times \frac{\overline{b}_2[\alpha]}{\overline{a}_{22}[\alpha]}, \tag{10.49}$$

which is a rectangle in $\mathbb{R}^2$ for all $0 \leq \alpha \leq 1$.

$\overline{X}_J[\alpha]$ is expressed in equation (10.49) as the product of two factors. The first factor of $\overline{X}_J[\alpha]$ is $\overline{X}_{J1}[\alpha]$ and $\overline{X}_{J2}[\alpha]$ is the second. Then

$$\overline{X}_{J1}[\alpha] = \left[\frac{1+\alpha}{7-2\alpha}, \frac{3-\alpha}{4+\alpha}\right], \tag{10.50}$$

$$\overline{X}_{J2}[\alpha] = \left[\frac{2+3\alpha}{12-4\alpha}, \frac{8-3\alpha}{6+2\alpha}\right], \tag{10.51}$$

$0 \le \alpha \le 1$.

Next we find that $|A_1|/|A| = b_1/a_{11}$ and $|A_2|/|A| = b_2/a_{22}$. From equations (10.41) and (10.42) we obtain $\overline{X}_{ei} = \overline{X}_{Ji}$, $i = 1, 2$.

Finally, we substitute the intervals $\overline{a}_{11}[\alpha]$, $\overline{a}_{22}[\alpha]$, $\overline{b}_1[\alpha]$ and $\overline{b}_2[\alpha]$ into $x_1 = b_1/a_{11}$ and $x_2 = b_2/a_{22}$ and we see that $\overline{X}_{Ii} = \overline{X}_{ei}$, $i = 1, 2$.

For this 2×2 fuzzy diagonal matrix $\overline{A}$ we get

$$\overline{X}_{ci} \subsetneq \overline{X}_{Ji} = \overline{X}_{ei} = \overline{X}_{Ii}, \tag{10.52}$$

$i = 1, 2$. The graphs of $\overline{X}_{ci}$ and $\overline{X}_{Ji}$, $i = 1, 2$ are in Figures 10.13 and 10.14.

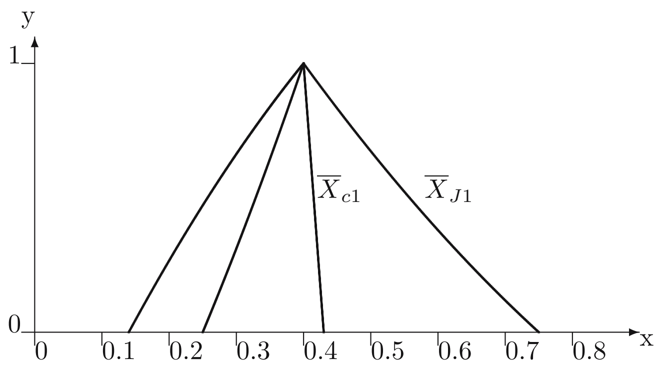


Fig. 10.13. $\overline{X}_{c1} \approx (\frac{1}{4}/\frac{2}{5}/\frac{3}{7})$ and $\overline{X}_{J1} \approx (\frac{1}{7}/\frac{2}{5}/\frac{3}{4})$ in Example 10.4.1

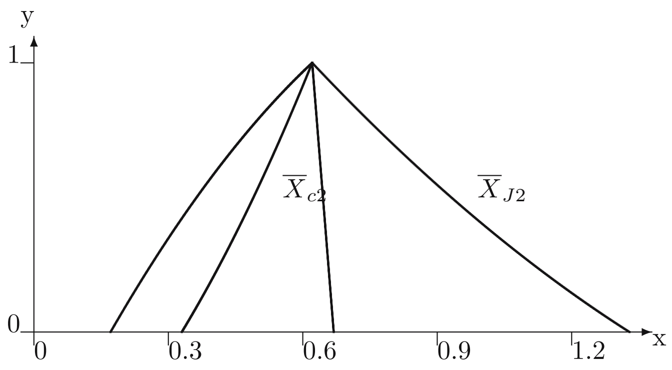


Fig. 10.14. $\overline{X}_{c2} \approx (\frac{1}{3}/\frac{5}{8}/\frac{3}{4})$ and $\overline{X}_{J2} \approx (\frac{1}{6}/\frac{5}{8}/\frac{4}{3})$ in Example 10.4.1

Example 10.4.2

Let

$$\overline{A} = \begin{pmatrix} \overline{a}_{11} & 0 \\ 1 & \overline{a}_{22} \end{pmatrix}, \tag{10.53}$$

and $\overline{B}^t = (\overline{b}_1, \overline{b}_2)$, where $\overline{a}_{11} = (1/2/3)$, $\overline{a}_{22} = (2/5/8)$, $\overline{b}_1 = (4/5/7)$ and $\overline{b}_2 = (6/8/12)$. Then $\overline{a}_{11}[\alpha] = [1+\alpha, 3-\alpha]$, $\overline{a}_{22}[\alpha] = [2+3\alpha, 8-3\alpha]$, $\overline{b}_1[\alpha] = [4+\alpha, 7-2\alpha]$ and $\overline{b}_2[\alpha] = [6+2\alpha, 12-4\alpha]$.

As in Example 10.4.1 we solve for $\overline{X}_{c1} > 0$ and obtain

$$\left[\frac{4+\alpha}{1+\alpha}, \frac{7-2\alpha}{3-\alpha}\right], \tag{10.54}$$

which does not define a fuzzy number since $\partial/\partial\alpha[(4+\alpha)/(1+\alpha)] < 0$, or $(4+\alpha)/(1+\alpha)$ is a decreasing function of α in $[0,1]$. The classical solution does not exist.

We find α-cuts of $\overline{X}_J$ using equation (10.48). We only go through the details for $\alpha = 0$ and $\alpha = 1$. An equivalent expression to equation (10.48) is

$$\overline{X}_J[\alpha] = \{\, x \in \mathbb{R}^n \mid Ax = b, v \in a[\alpha], b \in b[\alpha]\}\,. \tag{10.55}$$

For $\alpha = 1$ we get $x = (2.5, 1.1)$. For $\alpha = 0$ first assume $x_1 \geq 0$, $x_2 \geq 0$. Then we want all solutions for x_1 and x_2 so that

$$([1,3]x_1 + [0,0]x_2) \cap [4,7] \neq \emptyset, \tag{10.56}$$
$$([1,1]x_1 + [2,8]x_2) \cap [6,12] \neq \emptyset. \tag{10.57}$$

We have used the $\alpha = 0$ cuts of $\overline{a}_{11}$, $\overline{a}_{22}$, $\overline{b}_1$ and $\overline{b}_2$. This means

$$x_1 \leq 7, \tag{10.58}$$
$$3x_1 \geq 4, \tag{10.59}$$
$$x_1 + 2x_2 \leq 12, \tag{10.60}$$
$$x_1 + 8x_2 \geq 6, \tag{10.61}$$

for $x_1 \geq 0$, $x_2 \geq 0$ in the first quadrant. Now x_1 must be non–negative so we can now only consider the fourth quadrant.

Assume $x_1 \geq 0$ and $x_2 \leq 0$. Then the equations become

$$x_1 \leq 7, \tag{10.62}$$
$$3x_1 \geq 4, \tag{10.63}$$
$$x_1 + 8x_2 \leq 12, \tag{10.64}$$
$$x_1 + 2x_2 \geq 6, \tag{10.65}$$

for $x_1 \geq 0$, $x_2 \leq 0$. The solution $\overline{X}_J[0]$ is shown in Figure 10.15. It is not convex since the line joining $(4/3, 7/12)$ and $(7, -1/2)$ is not entirely in $\overline{X}_J[0]$.

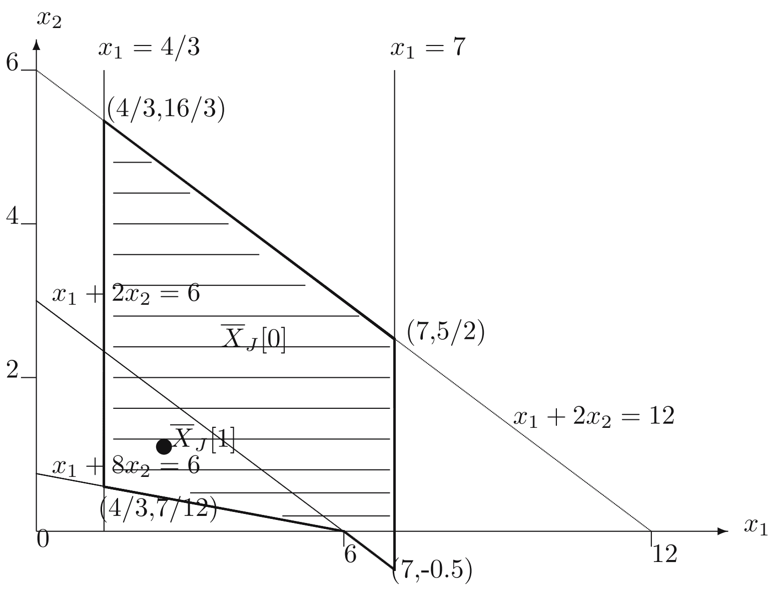


Fig. 10.15. Support of the Joint Solution in Example 10.4.2

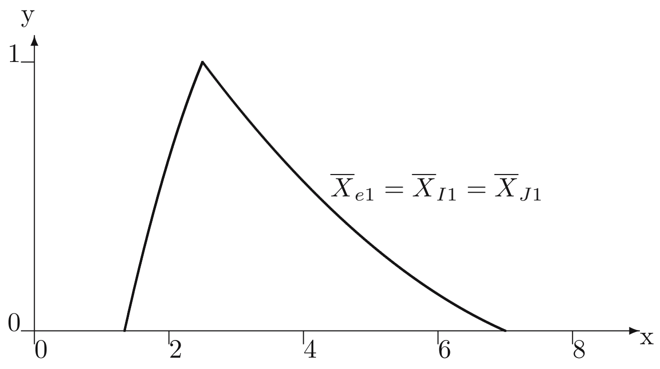


Fig. 10.16. $\overline{X}_{e1} = \overline{X}_{I1} = \overline{X}_{J1} \approx (\frac{4}{3}/\frac{5}{2}/7)$ in Example 10.4.2

Projecting $\overline{X}_J$ onto the x_i–axes produces $\overline{X}_{Ji}$, $i = 1, 2$. These marginals are in Figures 10.16 and 10.17.

We find the α-cuts of the $\overline{X}_{Ji}$ as follows: (1) we first construct a diagram like Figure 10.15 for $\overline{X}_J[\alpha]$ for each $0 \leq \alpha \leq 1$; (2) project the diagram onto the x_1-axis to get $\overline{X}_{J1}[\alpha]$; and (3) project the picture onto the x_2-axis to obtain $\overline{X}_{J2}[\alpha]$. It turns out, for this example, that $\overline{X}_{J1} = \overline{X}_{e1}$ and $\overline{X}_{J2} = \overline{X}_{e2}$.

Using equations (10.41) and (10.42) we find α-cuts of $\overline{X}_{ej}$, $j = 1, 2$. It is easy to see that

$$\overline{X}_{e1}[\alpha] = \left[\frac{4+\alpha}{3-\alpha}, \frac{7-2\alpha}{1+\alpha}\right], \tag{10.66}$$

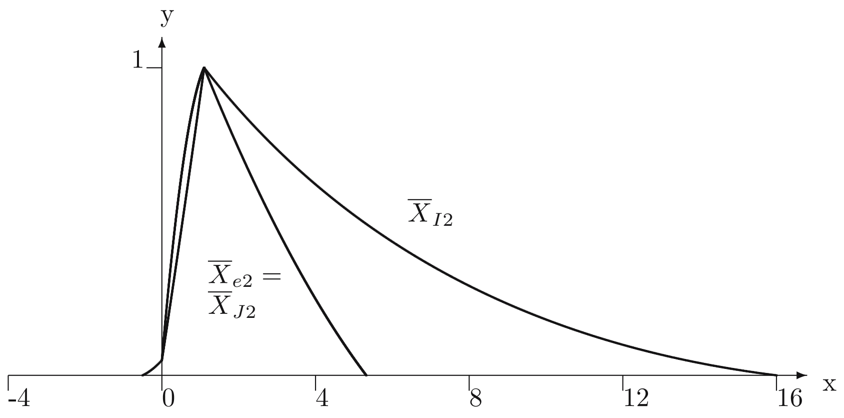


Fig. 10.17. $\overline{X}_{e2} = \overline{X}_{J2}, \overline{X}_{I2} \approx (-\frac{1}{2}/\frac{11}{10}/16)$ in Example 10.4.2

$0 \leq \alpha \leq 1$. However, $\overline{X}_{e2}[\alpha]$ is a little more difficult since we need to find the max and min of

$$\frac{a_{11}b_2 - b_1}{a_{11}a_{22}}, \tag{10.67}$$

for $a_{11} \in \overline{a}_{11}[\alpha]$, $a_{22} \in \overline{a}_{22}[\alpha]$, $b_1 \in \overline{b}_1[\alpha]$ and $b_2 \in \overline{b}_2[\alpha]$. We did this and $\overline{X}_{e2}$ is shown in Figure 10.17.

Lastly, we see that $\overline{X}_{I1} = \overline{X}_{e1}$ and

$$\overline{X}_{I2}[\alpha] = [x_{I21}(\alpha), x_{I22}(\alpha)], \tag{10.68}$$

with $x_{I21}(\alpha) = N_1(\alpha)/D_2(\alpha)$, $0 \leq \alpha \leq 0.0981 = (\sqrt{108} - 10)/4$, $x_{I21}(\alpha) = N_1(\alpha)/D_1(\alpha)$ for $0.0981 \leq \alpha \leq 1$ and $x_{I22}(\alpha) = N_2(\alpha)/D_2(\alpha)$ for all α and

$$N_1(\alpha) = (1+\alpha)(6+2\alpha) - (7-2\alpha), \tag{10.69}$$

$$N_2(\alpha) = (3-\alpha)(12-4\alpha) - (4+\alpha), \tag{10.70}$$

$$D_1(\alpha) = (3-\alpha)(8-3\alpha), \tag{10.71}$$

$$D_2(\alpha) = (1+\alpha)(2+3\alpha). \tag{10.72}$$

The reason for the change in the denominator for $x_{I21}(\alpha)$ is that $N_1(\alpha)$ is negative for $0 \leq \alpha \leq 0.0981$. We used the fact that $[a,b][c,d] = [ad, bd]$ if $a < 0 < b$ and $0 < c < d$ but $[a,b][c,d] = [ac, bd]$ when $0 < a$ and $0 < c$. $\overline{X}_{ei}$ and $\overline{X}_{Ii}$ are in Figures 10.16 and 10.17 for $i = 1, 2$.

10.4.1 Fuzzy Monte Carlo Method

We follow the same strategy as outlined in Section 10.2.2 for generating approximate solutions to fuzzy linear equations using fuzzy Monte Carlo methods. **Q** will be the same set of fuzzy numbers and we use the same metric between fuzzy numbers given in equation (10.23). But now we need to extend that metric to a distance measure between fuzzy vectors. Let $\overline{X}^t = (\overline{x}_1, ..., \overline{x}_n)$ and

$$\overline{W}_i = \overline{a}_{i1}\overline{x}_1 + ... + \overline{a}_{in}\overline{x}_n, \tag{10.73}$$

for $i = 1, 2, ..., n$. Then $\overline{A}\ \overline{X} = \overline{B}$ is the same as $\overline{W} = \overline{B}$ where $\overline{W}^t = (\overline{W}_1, ..., \overline{W}_n)$. Given a $\overline{X}$ the distance between $\overline{W} = \overline{A}\ \overline{X}$ and $\overline{B}$ will be

$$D(\overline{W}, \overline{B}) = max\{D(\overline{W}_i, \overline{b}_i) | i = 1, 2, ..., n\}, \tag{10.74}$$

where $D(\overline{W}_i, \overline{b}_i)$ is from equation (10.23). Our new solution $\overline{X}^*$, whose elements $\overline{x}_i^*$ are fuzzy numbers in **Q**, solves

$$min\{D(\overline{W}, \overline{B}) | \overline{x}_i \in \mathbf{Q}\ all\ i\}. \tag{10.75}$$

Let $(\overline{X}^*)^t = (\overline{x}_1^*, ..., \overline{x}_n^*)$. Let $D(\overline{X}) = D(\overline{W}, \overline{B})$. Using our fuzzy Monte Carlo method we obtain $\overline{X}_a^*$ an approximation to $\overline{X}^*$. Let $\overline{X}_a^* = (\overline{x}_{a1}^*, ..., \overline{x}_{an}^*)$. We use the same "threshold" ϵ discussed in Section 10.2.2. If the fuzzy Monte Carlo method produces a $\overline{X}$ so that $D(\overline{X}) < \epsilon$ we will say that we have found an acceptable approximate solution $\overline{X}_a^* = \overline{X}$ with $\overline{X}_a^* \approx \overline{X}^*$. We begin with $D(\overline{X})$ very large (100) and will accept a solution only if it minimizes to $D(\overline{X}) < \epsilon = 0.5$.

Now we will rework Examples 10.4.1 and 10.4.2 using our fuzzy Monte Carlo method.

Example 10.4.1.1

This continues Example 10.4.1. We wish to use our fuzzy Monte Carlo method to compute $\overline{X}_a^* \approx \overline{X}^*$ and then compare $\overline{X}_a^*$ to $\overline{X}_c = (\overline{X}_{c1}, \overline{X}_{c2})$ shown in Figures 10.16 and 10.17. Recall that in this example the classical solution $\overline{X}_c$ exists and the components of $\overline{X}_a^* = (\overline{x}_{a1}^*, \overline{x}_{a2}^*)$ are in **Q** and we must have $D(\overline{X}_a^*) < \epsilon = 0.5$.

The fuzzy matrix equation in this example may be written $\overline{a}_{11}\ \overline{x}_1 = \overline{b}_1$ and $\overline{a}_{22}\ \overline{x}_2 = \overline{b}_2$; hence, we can solve for the $\overline{x}_i$, $i = 1, 2$, separately. However, we choose to solve for them simultaneously. Although we may determine individually lower $D(\overline{x}_{ai}^*)$ (having only to satisfy them one at a time), it is more algorithmically convenient to solve them simultaneously as we will do for this Example and for Example 10.4.1.2 discussed below. The difference between the methods will be whether we take crisp quasi-random numbers as 5-tuples for each $\overline{x}_i$, or as 10-tuples for $(\overline{x}_1, \overline{x}_2)$.

We solve for the "best" $\overline{x}_1$ and $\overline{x}_2$ simultaneously. We generate a Sobol quasi-random number 10-tuple with which we generate $\overline{x}_1$ with the first five and $\overline{x}_2$ with the last five. Using $\overline{X} = (\overline{x}_1, \overline{x}_2)$ we compute $\overline{W}_1 = \overline{a}_{11}\overline{x}_1 + \overline{a}_{12}\overline{x}_2$, and

$\overline{W}_2 = \overline{a}_{21}\overline{x}_1 + \overline{a}_{22}\overline{x}_2$ (even though for Example 10.4.1 $\overline{a}_{12} = 0$ and $\overline{a}_{21} = 0$). Then $D(\overline{W}, \overline{B}) = max\{D(\overline{W}_1, \overline{b}_1), D(\overline{W}_2, \overline{b}_2)\}$ is determined for this $\overline{X}$. As we evaluate these $\overline{X}$'s we find the least $D(\overline{W}, \overline{B})$, capturing its $\overline{X}$ as $\overline{X}^*_a$.

Next we determine intervals, which will depend on the application, for the random quadratic fuzzy numbers (QBGFNs). We modified the fuzzy Monte Carlo program to optimize the minimization problem of equations (10.74) and (10.75) for fuzzy matrix equations. For this problem the classical solution does exist. For $\overline{x}_1$ we choose $[a, b] = [0.1429, 0.7500]$ which is approximately the support of $\overline{X}_{J1}$. For $\overline{x}_2$ we choose $[a, b] = [0.1666, 1.3333]$ which is approximately the support of $\overline{X}_{J2}$.

To compute $D(\overline{A} \cdot \overline{X}, \overline{B})$, we used 100 α-cuts of $\overline{A} \cdot \overline{X}$ and $\overline{B}$ for given $\overline{A}$ and $\overline{B}$, where $\overline{X}$ is one of 50,000 pairs of random quadratic fuzzy numbers.

The smallest $D(\overline{W}, \overline{B})$ we obtained was 0.302329. The $\overline{X}$ that produced this value we had saved as $\overline{X}^*_a = (\overline{x}^*_{a2}, \overline{x}^*_{a2})$. $\overline{x}^*_{a1}$ and $\overline{x}^*_{a2}$ are displayed in Figure 10.18. As one can see in Table 10.5, solutions of $\left[\begin{smallmatrix} (4/5/7) & 0 \\ 0 & (6/8/12) \end{smallmatrix}\right] \cdot \overline{X} = \left[\begin{smallmatrix} (1/2/3) \\ (2/5/8) \end{smallmatrix}\right]$, we have acceptable solutions for $\overline{x}^*_{a1}$ and $\overline{x}^*_{a2}$. Figure 10.19 and Figure 10.20 are graphs for $\overline{A} \cdot \overline{X}$.

Table 10.5. Solutions for Example 10.4.1, $\overline{A} \cdot \overline{X} = \overline{B}$

Solution	$\overline{X}$	$\overline{A} \cdot \overline{X}$
$\overline{X}_{c1}$	$\approx (0.25/0.4/0.4286)$	$\approx (1/2/3) = \overline{b}_1$
$\overline{X}_{c2}$	$\approx (0.333\overline{3}/0.625/0.75)$	$\approx (2/5/8) = \overline{b}_2$
$\overline{X}_{J1}$	$\approx (0.1429/0.4/0.75)$	$\approx (0.5716/2/5.25)$
$\overline{X}_{J2}$	$\approx (0.166\overline{6}/0.625/1.333\overline{3})$	$\approx (1/5/20)$
$\overline{x}^*_{a1}$,	$(0.2473, 0.3973, 0.4436, -0.7088, 0.4686)$,	$\approx (0.99/1.99/3.11)$
	$D(\overline{W}_1, \overline{b}_1) = 0.119918$	
$\overline{x}^*_{a2}$,	$(0.3646, 0.6203, 0.6908, -0.0618, 0.0208)$,	$\approx (2.19/4.96/8.29)$
	$D(\overline{W}_2, \overline{b}_2) = 0.302329$ (min)	

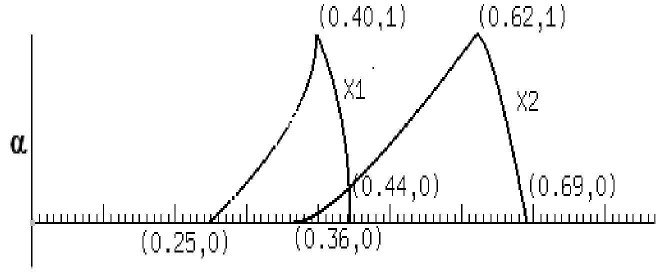


Fig. 10.18. $\overline{x}^*_{a1}$ and $\overline{x}^*_{a2}$ of Example 10.4.1.1

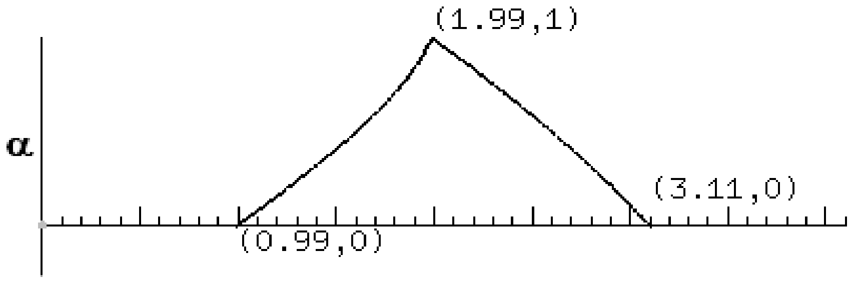


Fig. 10.19. $(4/5/7) \cdot \overline{x}^*_{a1}$ of Example 10.4.1.1

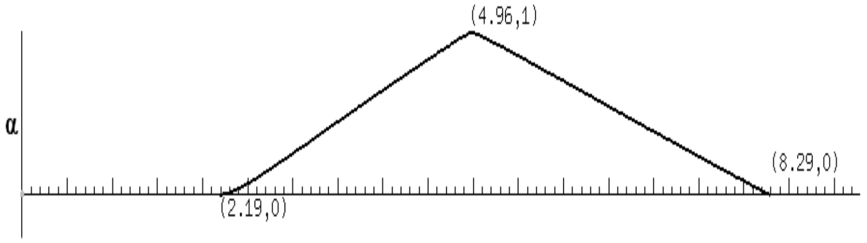


Fig. 10.20. $(6/8/12) \cdot \overline{x}^*_{a2}$ of Example 10.4.1.1

Example 10.4.1.2

This continues Examples 10.4.2. We apply our fuzzy Monte Carlo method to compute $\overline{X}^*_a \approx \overline{X}^*$. Recall that in this example the classical solution does not exist but we do have $\overline{X}_e$, $\overline{X}_J$ and $\overline{X}_I$ shown in Figures 10.16 and 10.17. The constraints are still that the components of $\overline{X}^*_a$ are in $\mathbf{Q}$ and it must satisfy $D(\overline{X}^*_a) < \epsilon = 0.5$.

Table 10.6. Solutions for Example 10.4.2, $\overline{A} \cdot \overline{X} = \overline{B}$

	$\overline{X}$	$\overline{A} \cdot \overline{X}$
$\overline{X}_c$	does not exist	$\approx [(4/5/7)\ (6/8/12)]^t$
$\overline{X}_{I1}$	$\approx (1.333\overline{3}/2.5/7)$	$\approx (1.333\overline{3}/5/21)$
$\overline{X}_{I2}$	$\approx (-0.5/1.1/16)$	$\approx (1.333\overline{3}/8/135)$
$\overline{x}^*_{a1}$,	$(1.5847, 1.6205, 3.3639, 0.6678, -2.6900)$,	$\approx (1.58/3.24/10.09)$
	$D(\overline{W}_1, \bar{b}_1) = 3.091642$	
$\overline{x}^*_{a2}$,	$(0.4937, 0.7202, 1.3369, -0.8838, -1.1810)$,	$\approx (2.57/5.22/14/06)$
	$D(\overline{W}_1, \bar{b}_1) = 3.427969$ (min)	

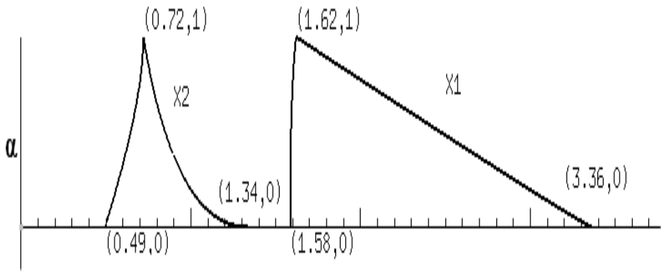


Fig. 10.21. $\overline{x}_{a1}^*$ and $\overline{x}_{a2}^*$ of Example 10.4.1.2

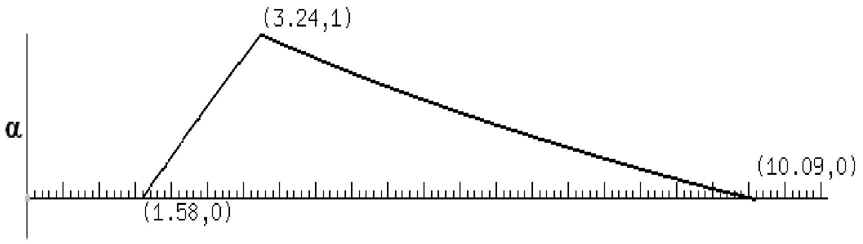


Fig. 10.22. $(1/2/3) \cdot \overline{x}_{a1}^*$ of Example 10.4.1.2

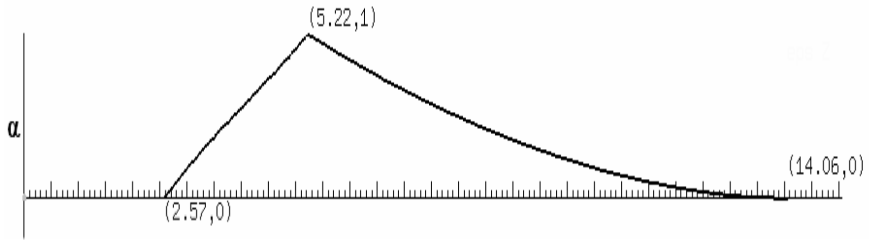


Fig. 10.23. $\overline{x}_{a1}^* + (2/5/8) \cdot \overline{x}_{a2}^*$ of Example 10.4.1.2

For this problem the classical solution does not exist. For $\overline{x}_1$ we choose $[a, b] = [1.3333, 7]$ which is approximately the support of $\overline{X}_{I1}$. For $\overline{x}_2$ we choose $[a, b] = [0, 16]$ which covers the positive portion of the support of $\overline{X}_{I2}$.

To compute $D(\overline{A} \cdot \overline{X}, \overline{B})$, we used 100 α-cuts of $\overline{A} \cdot \overline{X}$ and $\overline{B}$ for given $\overline{A}$ and $\overline{B}$, where $\overline{X}$ is one of 50,000 pairs of random quadratic fuzzy numbers.

The distance measure D is given in equations (10.74) and (10.75). The smallest $D(\overline{W}, \overline{B})$ we obtained was 3.427969. The $\overline{X}$ that produced this value we had saved as $\overline{X}_a^* = (\overline{x}_{a1}^*, \overline{x}_{a2}^*)$. $\overline{X}_a^*$ is shown in Figure 10.21. $(1/2/3) \cdot \overline{x}_{a1}^*$ is shown in Figure 10.22. The corresponding $\overline{x}_{a1}^* + (2/5/8) \cdot \overline{x}_{a2}^*$ is shown in Figure 10.23.

Our fuzzy Monte Carlo method was unable to get an acceptable approximate solution to this fuzzy matrix equation. Table 10.6 lists solutions of $\left[\begin{smallmatrix} (1/2/3) & 0 \\ 1 & (2/5/8) \end{smallmatrix}\right] \cdot \overline{X} = \left[\begin{smallmatrix} (4/5/7) \\ (6/8/12) \end{smallmatrix}\right]$ for Example 10.4.2. As our value of $D(\overline{X})$ indicates, we do not have good correspondence with the right had side of our fuzzy matrix equation. We believe that no fuzzy quadratic fuzzy vector $\overline{X}$ can make this $D(\overline{X}) < \epsilon = 0.5$.

10.5 Summary and Conclusions

Through several examples of three types of fuzzy equations (fuzzy linear equations $\overline{A} \cdot \overline{X} + \overline{B} = \overline{C}$, fuzzy quadratic equations $\overline{A} \cdot \overline{X}^2 + \overline{B} \cdot \overline{X} + \overline{C} = \overline{D}$, and fuzzy matrix equations $\overline{A} \cdot \overline{X} = \overline{B}$) we have demonstrated a use of fuzzy Monte Carlo optimization to obtain solutions. Where possible we demonstrated computation of the "classical" solution. We showed how one may compute an "extension principle" solution; and we showed how one may determine a "interval arithmetic" solution.

Also, for every example we used fuzzy Monte Carlo optimization to produce some solution. In those examples where a "classical" solution could be computed, fuzzy Monte Carlo found an acceptable solution. In one case, Example 10.3.1.2, were no classical solution existed, fuzzy Monte Carlo determined an acceptable approximate solution.

This study supports the viability of this method for solving fuzzy equations. Our choice of a value ϵ, and performance of $D(\overline{X})$ during simulations, indicates we may in the future choose a smaller $\epsilon = 0.2$ to differentiate "tight" solutions from "loose" ones.

These optimizations were performed on various Windows XP machines running in the 2Ghz range with over 1GB RAM.

$\overline{A} \cdot \overline{X} + \overline{B} = \overline{C}$ completed $50,000$ iterations in about 1.5 hours. Fuzzy quadratic exercises and fuzzy matrix equations completed $50,000$ iterations in $2 - 3$ hours. Convergence to an acceptable $D(\overline{X})$ was evident within a few hundred iterations.

References

1. Abbasbandy, S., Jafarian, A.: Steepest Descent Method for System of Fuzzy Linear Equations. Applied Math. and Computation 175, 823–833 (2006)
2. Allahviranloo, T., Kermani, M.A.: Solution of a Fuzzy System of Linear Equations. Applied Math. and Computation 175, 519–531 (2006)
3. Amirfakhrian, M.: Numerical Solution of a Fuzzy System of Linear Equations with Polynomial Parametric Form. Int. J. Computer Mathematics 84, 1089–1097 (2007)
4. Brudaru, O., Leon, F., Buzatu, O.: Genetic Algorithm for Solving Fuzzy Equations. In: Proc. 8^{th} Int. Symposium Automatic Control and Computer Science, October 22-23, Iasi, Romania (2004)
5. Buckley, J.J.: Solving Fuzzy Equations in Economics and Finance. Fuzzy Sets and Systems 48, 289–296 (1992)
6. Buckley, J.J.: Solving Fuzzy Equations. Fuzzy Sets and Systems 50, 1–14 (1992)

7. Buckley, J.J., Eslami, E.: Neural Net Solutions to Fuzzy Problems: The Quadratic Equation. Fuzzy Sets and Systems 86, 289–298 (1997)
8. Buckley, J.J., Eslami, E.: Introduction to Fuzzy Logic and Fuzzy Sets. Physica-Verlag, Heidelberg (2002)
9. Buckley, J.J., Qu, Y.: Solving Linear and Quadratic Fuzzy Equations. Fuzzy Sets and Systems 18, 43–59 (1990)
10. Buckley, J.J., Qu, Y.: On Using Alpha-Cuts to Evaluate Fuzzy Equations. Fuzzy Sets and Systems 38, 309–312 (1990)
11. Buckley, J.J., Qu, Y.: Solving Fuzzy Equations: A New Solution Concept. Fuzzy Sets and Systems 39, 291–301 (1991)
12. Buckley, J.J., Qu, Y.: Solving Systems of Fuzzy Linear Equations. Fuzzy Sets and Systems 43, 33–43 (1991)
13. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
14. Buckley, J.J., Eslami, E., Hayashi, Y.: Solving Fuzzy Equations Using Neural Nets. Fuzzy Sets and Systems 86, 271–278 (1997)
15. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Equations Using Evolutionary Algorithms and Neural Nets. Soft Computing 6, 116–123 (2002)
16. Dehghan, M., Hashemi, B.: Solution of the Fully Fuzzy Linear Systems Using the Decomposition Procedure. Applied Math. and Computation 182, 1568–1580 (2006)
17. Dehghan, M., Hashemi, B., Ghatee, M.: Computational Methods for Solving Fully Fuzzy Linear Systems. Applied Math. and Computation 179, 328–343 (2006)
18. Muzzioli, S., Reynaerts, H.: The Solution of Fuzzy Linear Systems by Nonlinear Programming: a Financial Application. European J. Operational Research 177, 1218–1231 (2007)
19. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge, UK (1990)
20. Hansen, E.: On the Solution of Linear Equations with Interval Coefficients. Linear Algebra and its Applications 2, 153–165 (1969)
21. Vroman, A., Deschrijver, G., Kerre, E.E.: Solving Systems of Fuzzy Equations by Parametric Functions–An Improved Algorithm. Fuzzy Sets and Systems 158, 1515–1534 (2007)
22. Wang, K., Zheng, B.: Inconsistent Fuzzy Linear Systems. Applied Math. and Computation 181, 973–981 (2006)

11 Fuzzy Linear Regression I

11.1 Introduction

This is the first of four chapters on fuzzy regression. This chapter and Chapter 14 are about fuzzy linear regression and Chapters 12 and 13 consider fuzzy nonlinear regression. In this chapter the independent (predictor, explanatory) variables are crisp but the dependent (response) variable is fuzzy. In Chapter 14 both the independent variables and the dependent variable are fuzzy. This chapter is based on [1].

Fuzzy linear regression has become a very large area of research. Put "fuzzy regression" into your search engine and you can get too many web sites to visit. "Fuzzy linear regression" will eliminate a lot of web sites but the list is still quite long. We have selected a few recent and key references on fuzzy linear regression which are: (1) books (or articles in these books) ([4],[8],[14]); and (2) papers ([2],[3],[5],[9],[10],[12],[13],[15],[19]-[24],[27]-[32]). As far as the authors know our research is the only research on using Monte Carlo techniques in fuzzy linear regression. However, there have been other approaches employing random search (genetic algorithms) and others using neural nets. If we put "genetic algorithms" and "fuzzy linear regression" into the search engine there are less than 200 references. A recent reference is [11]. We feel that one problem with using a GA is that it can converge to a local minimum and to avoid this you need to start it with many different randomly generated initial populations. Also, we believe that our Monte Carlo method is easier to apply than a genetic algorithm, once you have a quasi-random number generator in your computer. Next we searched for "neural nets" and "fuzzy linear regression" getting less than 100 references. A key reference on this topic is [6].

Consider a fuzzy linear regression model

$$\overline{Y} = \overline{A}_0 + \overline{A}_1 x_1 + ... + \overline{A}_m x_m, \tag{11.1}$$

where the $x_1, ..., x_m$ are crisp real numbers and the $\overline{A}_0, ..., \overline{A}_m$ and $\overline{Y}$ are all triangular fuzzy numbers. In this model the independent (predictor, explanatory) variables are crisp but the dependent (response) variable is fuzzy. The data will be $((x_{1l},...,x_{ml}),\overline{Y}_l)$, $1 \leq l \leq n$, for the x_{il} given real numbers and $\overline{Y}_l$ are given triangular fuzzy numbers. The best way to fit the model to the data is to have the $\overline{A}_j$, $0 \leq j \leq m$, also triangular fuzzy numbers. Given the data the objective is to find the "best" $\overline{A}_j$, $0 \leq j \leq m$. We propose to employ Monte Carlo methods to approximate the "best" values for the $\overline{A}_j$, $j = 0,1,...,m$.

In Monte Carlo we randomly generate a possible solution, evaluate how "good" it is, discard inferior solutions, and continue N times. N is usually large like $10,000$ or $100,000$. In the next section we discuss how to randomly produce vectors $\overline{V}_k = (\overline{A}_{0k},...,\overline{A}_{mk})$, $k = 1,2,3,...,N$. Using the $\overline{V}_k$ we determine the predicted values

$$\overline{Y}^*_{lk} = \overline{A}_{0k} + \overline{A}_{1k}x_{1l} + ... + \overline{A}_{mk}x_{ml}, \tag{11.2}$$

for $k = 1,2,3,..,N$ and $l = 1,2,...,m$. To see how good this $\overline{V}_k$ is we find the error between the given values $\overline{Y}_l$ and the predicted values $\overline{Y}^*_{lk}$. We will have two error measures in this chapter. The first error measure is

$$E_{1k}(E_1) = \sum_{l=1}^{n}[\int_{-\infty}^{\infty} |\overline{Y}_l(x) - \overline{Y}^*_{lk}(x)|dx]/[\int_{-\infty}^{\infty} \overline{Y}_l(x)dx], \tag{11.3}$$

where the integrals are really only over interval(s) containing the support of the fuzzy numbers. Let $\overline{Y}_l = (y_{l1}/y_{l2}/y_{l3})$ and $\overline{Y}^*_{lk} = (y_{lk1}/y_{lk2}/y_{lk3})$ all triangular fuzzy numbers. Then our second error measure is

$$E_{2k}(E_2) = \sum_{l=1}^{n}[|y_{l1} - y_{lk1}| + |y_{l2} - y_{lk2}| + |y_{l3} - y_{lk3}|]. \tag{11.4}$$

So we calculate $\overline{V}_k$, E_{1k} and E_{2k} for $k = 1,2,...,N$. A "best" solution is a value of $\overline{V}_k$ that minimizes E_{1k} (E_{2k}) for all k. An approximate "best" solution is a $\overline{V} \in \{\overline{V}_1,...,\overline{V}_N\}$ that minimizes an error measure. So we can have two approximate "best" solutions one with respect to E_1 and an other for E_2. Next we see how we will produce sequences of random vectors $\overline{V}_k$, $k = 1,2,3,..,N$.

11.2 Random Fuzzy Vectors

To obtain random sequences $\overline{V}_k = (\overline{X}_{0k},...,\overline{X}_{mk})$, where the $\overline{X}_{ik}$ are all triangular fuzzy numbers, we first randomly generate crisp vectors $v_k = (x_{1k},...,x_{3m+3,k})$ using our Sobol quasi-random number generator (Chapter 3) with all the x_{ik} in $[0,1]$, $k = 1,2,..,N$. We choose the first three numbers in v_k and order them from smallest to largest. Assume that $x_{3k} < x_{1k} < x_{2k}$. Then the first triangular fuzzy number $\overline{X}_0 = (x_{3k}/x_{1k}/x_{2k})$ which becomes $\overline{A}_0$. Continue with the next three numbers in v_k, etc. making $\overline{X}_i = \overline{A}_i$, $i = 1,2,...,m$.

However the $\overline{A}_i$ will need to be in certain intervals. Suppose $\overline{A}_i$ is to be in interval $I_i = [a_i,b_i]$, $i = 0,1,2,...,m$. These intervals are very important to the Monte Carlo process because: (1) if they are wrong and/or too small we can miss a "good" solution; and (2) if they are too big the simulation can spend too much time looking at situations that will not produce a "good" solution. Since each

Table 11.1. Data for the Application

Fuzzy Output $\overline{Y}_l$	x_{1l}	x_{2l}	x_{3l}
(2.27/5.83/9.39)	2.00	0.00	15.25
(0.33/0.85/1.37)	0.00	5.00	14.13
(5.43/13.93/22.43)	1.13	1.50	14.13
(1.56/4.00/6.44)	2.00	1.25	13.63
(0.64/1.65/2.66)	2.19	3.75	14.75
(0.62/1.58/2.54)	0.25	3.50	13.75
(3.19/8.18/13.17)	0.75	5.25	15.25
(0.72/1.85/2.98)	4.25	2.00	13.50

$\overline{X}_i$ starts out in $[0, 1]$ we may easily put it into $[a_i, b_i]$ by $\overline{A}_i = a_i + (b_i - a_i)\overline{X}_i$, $i = 0, 1, ..., m$.

11.3 Application

The data for this application was taken from [16] and is shown in Table 11.1. There are three ($m = 3$) independent variables x_1, x_2 and x_3. Also, there are only eight ($n = 8$) items in the data set. We will need to find intervals I_i, $i = 0, 1, 2, 3$, as explained above for the $\overline{A}_i$. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the $\overline{A}_i$, $i = 0, 1, 2, 3$, from [7]; and (2) secondly, in the second subsection, we use two optimization procedures to determine these intervals.

The authors in [16] compared their method, applied to the data in Table 11.1, to that in [25] and [26] applied to the same data set, in their Table 5. They give the predicted values for the dependent variable using the three methods. We have also studied this data set in [7] and obtained predicted values for the dependent variable using a least absolute values estimator. In this chapter we apply our Monte Carlo method to compute predicted values and compare our new results to the other four methods using error measures E_1 and E_2. All programs were written in MATLAB [18]. A copy of the MATLAB program may be obtained from the authors. For all our calculations, we used a Pentium III, Processor: 933 MHz.

11.3.1 First Choice of Intervals

After studying the solutions for the $\overline{A}_i$ using the methods in [7] we first decided on the following intervals for our fuzzy Monte Carlo method: (1) $[-1, 0]$ for $\overline{A}_0$; (2) $[-1, 0]$ for $\overline{A}_1$; (3) $[-1.5, -0.5]$ for $\overline{A}_2$; and (4) $[0, 1]$ for $\overline{A}_3$. The exact solutions in [7] for the $\overline{A}_i$ are given below.

Using our Sobol quasi-random number generator we produced $70,000$ vectors $v_k = (x_{1k}, ..., x_{12,k})$ which defined the $\overline{A}_i$, $i = 0, 1, 2, 3$, as described in Section 11.2. Results for the $\overline{A}_i$ are shown in Table 11.2 with minimum error values in Table 11.8. Since we will have four Monte Carlo studies on this data we call this one MCI.

Table 11.2. Results of the Monte Carlo (MCI) Method, First Choice of Intervals, to Minimize the Error

Coefficient	*MCI* E_1	*MCI* E_2
$\overline{A}_0$	$(-0.4953/-0.4306/-0.3393)$	$(-0.8902/-0.6815/-0.3355)$
$\overline{A}_1$	$(-0.5005/-0.4656/-0.0059)$	$(-0.6808/-0.5436/-0.5194)$
$\overline{A}_2$	$(-0.7965/-0.7864/-0.7165)$	$(-1.0640/-1.0476/-0.8578)$
$\overline{A}_3$	$(0.3335/0.3540/0.3920)$	$(0.4756/0.5379/0.6112)$

Table 11.3. Results of the Monte Carlo (MCII) Method, First Choice of Intervals, to Minimize the Error

Coefficient	*MCII* E_1	*MCII* E_2
$\overline{A}_0$	$(0.2464/0.4892/0.7266)$	$(0.0285/0.3569/0.8847)$
$\overline{A}_1$	$(-0.4815/-0.2852/-0.1398)$	$(-0.5654/-0.5329/-0.3708)$
$\overline{A}_2$	$(-0.8760/-0.8303/-0.7575)$	$(-1.0999/-1.0600/-0.9360)$
$\overline{A}_3$	$(0.3174/0.3361/0.3398)$	$(0.4052/0.4381/0.5280)$

Next we experimented with other intervals. We started with larger intervals, shifted them and shortened them, until we arrived at: (1) $[0,1]$ for $\overline{A}_0$; (2) $[-1,0]$ for $\overline{A}_1$; (3) $[-1.5,-0.5]$ for $\overline{A}_2$; and (4) $[0,1]$ for $\overline{A}_3$. The only difference is for $\overline{A}_0$. After another run of $70,000$ quasi-random vectors the results for the $\overline{A}_i$ are in Table 11.3 with minimum error values in Table 11.8. This Monte Carlo study is called MCII. For these choices of intervals, in MCI and MCII, the computational time was between 25 and 28 minutes.

11.3.2 Second Choice of Intervals

The first thing to do is to determine the intervals $I_i = [a_i, b_i]$ for the $\overline{A}_i$, $0 \le i \le 3$. We first describe an optimization method used to determine these intervals. This procedure will be called *MCIII*. A second optimization method will be used and it will be described below. Let

$$[L_l, R_l] = I_0 + I_1 x_{1l} + I_2 x_{2l} + I_3 x_{3l}, \tag{11.5}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 8$. Recall the data $\overline{Y}_l = (y_{l1}/y_{l2}/y_{l3})$. Define

$$W = \sum_{l=1}^{8} (L_l - y_{l1})^2 + \sum_{l=1}^{8} (R_l - y_{l3})^2. \tag{11.6}$$

The optimization problem is to minimize W subject to $a_i \le b_i$ all i. We want to find the intervals that make L_l and R_l closest, in the sense of minimizing W, to the end points of the bases of the dependent fuzzy numbers $\overline{Y}_l$ in the

Table 11.4. First Method of Determining the Intervals (MCIII) for Monte Carlo

Interval	Value
I_0	$[-18.174, -18.174]$
I_1	$[-1.083, -1.083]$
I_2	$[-1.150, -1.150]$
I_3	$[1.733, 2.149]$

Table 11.5. Results of the Monte Carlo Method (MCIII), Second Choice of Intervals, to Minimize the Error

Coefficient	$MCIII\ E_1$	$MCIII\ E_2$
A_0	-18.174	-18.174
A_1	-1.083	-1.083
A_2	-1.150	-1.150
$\overline{A}_3$	$(1.736/1.752/1.792)$	$(1.733/1.799/1.958)$

data. We solved this using Maple [17]. The results, rounded to three decimal places, are in Table 11.4. It is very interesting that the first three "intervals" are degenerate and are just real numbers. Using these intervals $\overline{A}_0 = A_0 = -18.174$, $\overline{A}_1 = A_1 = -1.083$, $\overline{A}_2 = A_2 = -1.150$ and the support of $\overline{A}_3$ is a subset of $[1.733, 2.149]$ with only one fuzzy number.

We now produce a sequence of random crisp vectors $v_k = (x_{1k}, ..., x_{3k})$, $k = 1, 2, ..., N$, using our Sobol quasi-random number generator as described in Section 11.2, to get a sequence of triangular fuzzy numbers $\overline{A}_{3k}$ and computed E_{1k} and E_{2k}. After a run of $N = 70,000$ the smallest E_1 value and the minimum E_2 value found are shown in Table 11.8 with corresponding $\overline{A}_3$ shown in Table 11.5.

We also investigated a second optimization procedure for finding the intervals. L_l and R_l are defined as above. This method is called MCIV. Let

$$W = \sum_{l=1}^{8}(y_{l1} - L_l) + \sum_{l=1}^{8}(R_l - y_{l3}). \tag{11.7}$$

The linear programming problem is to minimize W subject to the constraints: (1) $L_l \leq y_{l1}$ all l; (2) $R_l \geq y_{l3}$ all l; and (3) $a_i \leq b_i$ all i. We solved this problem using Maple [17] and the results are in Table 11.6. It is again interesting that the last three "intervals" are degenerate and are just real numbers. Using these intervals with the support of $\overline{A}_0$ a subset of $[28.000, 47.916]$, $\overline{A}_1 = A_1 = -2.542$, $\overline{A}_2 = A_2 = -2.323$ and $\overline{A}_3 = A_3 = -1.354$ with only one fuzzy number.

We now produce a sequence of random crisp vectors $v_k = (x_{1k}, ..., x_{3k})$, $k = 1, 2, ..., N$, using a quasi-random number generator as described in Section 11.2, to get a sequence of triangular fuzzy numbers $\overline{A}_{0k}$ and computed E_{1k} and E_{2k}. After a run of $N = 70,000$ the smallest E_1 value and the minimum E_2 value

Table 11.6. Second Method of Determining the Intervals (MCIV) for Monte Carlo

Interval	Value
I_0	$[28.000, 47.916]$
I_1	$[-2.542, -2.542]$
I_2	$[-2.323, -2.323]$
I_3	$[-1.354, -1.354]$

Table 11.7. Results of the Monte Carlo Method (MCIV), Second Choice of Intervals, to Minimize the Error

Coefficient	$MCIV\ E_1$	$MCIV\ E_2$
$\overline{A}_0$	(35.842/36.030/36.030)	(31.062/33.336/36.228)
A_1	-2.542	-2.542
A_2	-2.323	-2.323
A_3	-1.354	-1.354

found are shown in Table 11.8 with corresponding $\overline{A}_0$ shown in Table 11.7. The total computational time for the second choice of intervals, MCIII and MCIV, was between 15 and 18 minutes.

11.3.3 Comparison of Solutions

Table 5 in [16] gives the predicted values for the dependent variable for the techniques used in [16],[25] and [26]. The predicted values were not given in [7] but we know the optimal solution for the $\overline{A}_i$: $\overline{A}_0 = (-0.71/-0.539/-0.524)$, $\overline{A}_1 = (-0.61/-0.473/-0.472)$, $\overline{A}_2 = (-1.09/-1.089/-1.088)$ and $\overline{A}_3 = (0.459/0.487/0.68)$. From this we may compute the predicted values. From the Monte Carlo methods discussed above we take the solutions for the $\overline{A}_i$ and determine the predicted values. From the predicted values we can find E_1 (equation (11.3)) and E_2 (equation (11.4)). The results are shown in Table 11.8.

The dependent variable represents "response time" and can not be negative. If a predictive value is $(-1.15/2.33/3.04)$ the authors in [16] round up the negative to zero and present $(0/2.33/3.04)$ as the predicted response time in their Table 5. Since we did not have access to the original predicted values we used those in their Table 5 with the zero value in our calculations for our Table 11.8. By rounding the negative left end point up to zero E_2 will decrease and E_1 may decrease or increase.

We see from Table 11.8 that our Monte Carlo method obtained the smallest values for error measure E_1. However, our Monte Carlo procedure did not get the smallest values for E_2. The smallest value for E_2 was gotten by [7]. Let us explain why we did not expect Monte Carlo to do better than [7] on E_2.

Table 11.8. Error Measures in the Application

Error	[25]	[26]	[16]	[7]	MCI	$MCII$	$MCIII$	$MCIV$
E_1	53.82	48.79	16.98	9.26	6.169	5.812	7.125	8.201
E_2	143.45	131.83	70.99	61.86	64.878	63.590	66.463	94.092

Let $\overline{A}_i = (a_{i1}/a_{i2}/a_{i3})$, $i = 0, 1, 2, 3$. Also let $LL_l = a_{01} + \sum_{i=1}^{3} a_{i1}x_{il}$,$C_l = a_{02} + \sum_{i=1}^{3} a_{i2}x_{il}$ and $RR_l = a_{03} + \sum_{i=1}^{3} a_{i3}x_{il}$. In [7] they first solve

$$min \sum_{l=1}^{8} |C_l - y_{l2}|, \tag{11.8}$$

for the a_{i2}, $i = 0, 1, 2, 3$. Let the solution be a_{i2}^*, $i = 0, 1, 2, 3$. Then they solve for the a_{i1} from

$$min \sum_{l=1}^{8} |LL_l - y_{l1}|, \tag{11.9}$$

subject to $a_{i1} \leq a_{i2}^*$ all i, and solve for the a_{i3} from

$$min \sum_{l=1}^{8} |RR_l - y_{l3}|, \tag{11.10}$$

subject to $a_{i3} \geq a_{i2}^*$ all i. This is like finding the $\overline{A}_i$ to minimize E_2. Hence, we expected [7] to have a minimum value for E_2.

11.4 Summary and Conclusions

In this chapter we studied the fuzzy linear regression problem given in equation (11.1). We employed our fuzzy Monte Carlo method to approximate the "best" solutions for the coefficients $\overline{A}_i$, $0 \leq i \leq m$. Best will be measured by two error measures E_1 (equation (11.3)) and E_2 (equation (11.4)). We showed in an example problem that our Monte Carlo method was best according to E_1 with respect to the results on the same data set in four other publications. Monte Carlo did not get the smallest E_2 value, but MCI and MCII came close, and we explained above why [7] was expected to show the smallest E_2 value. Given any error measure E^* we conjecture that our Monte Carlo method, allowing the number of iterations N to be sufficiently large, will be best (minimizing E^*), or approximately best. If this conjecture is true, then the estimation technique in fuzzy linear regression may become Monte Carlo. We can easily extend out method to trapezoidal fuzzy numbers, quadratic fuzzy numbers and other more general fuzzy numbers.

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Linear Regression. Soft Computing 11, 991–996 (2007)
2. Bargiela, A., Pedrycz, W., Nakashima, T.: Multiple Regression with Fuzzy Data. Fuzzy Sets and Systems 158, 2169–2188 (2007)
3. Buckley, J.J., Feuring, T.: Linear and Non-Linear Fuzzy Regression: Evolutionary Algorithm Solutions. Fuzzy Sets and Systems 112, 381–394 (2000)
4. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
5. Chang, Y.-H.O., Ayyub, B.M.: Fuzzy Regression Methods-a Comparative Assessment. Fuzzy Sets and Systems 119, 187–203 (2001)
6. Cheng, C.-B.: Fuzzy Regression Analysis by a Fuzzy Neural Network and Its Application to Dual Response Optimization. In: Proceedings Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf., Vancouver, Canada, vol. 5, pp. 2681–2686 (2001)
7. Choi, S.H., Buckley, J.J.: Fuzzy Regression Using Least Absolute Deviation Estimators. Soft Computing (to appear)
8. Diamond, P., Tanaka, H.: Fuzzy Regression Analysis. In: Slowinski, R. (ed.) Fuzzy Sets in Decision Analysis, Operations Research and Statistics, pp. 349–390. Kluwer Academic Publishers, Boston (1998)
9. D'Urso, P.: Linear Regression Analysis for Fuzzy/Crisp Input and Fuzzy/Crisp Output Data. Computational Statistics and Data Analysis 42, 47–72 (2003)
10. D'Urso, P., Gastaldi, T.: A Least-Squares Approach to Fuzzy Linear Regression Analysis. Computational Statistics and Data Analysis 34, 427–440 (2000)
11. Feuring, T., Golubski, W., Gassmann, M.: Fuzzy Regression: A Genetic Programming Approach. In: Fourth Int. Conf. Knowledge-Based Intelligent Engineering Systems and Allied Technologies, August 30-September 1, Brighton, UK (2000)
12. Hong, D.M., Lee, S., Do, H.Y.: Fuzzy Linear Regression Analysis for Fuzzy Input-Output Data Using Shape-Preserving Operations. Fuzzy Sets and Systems 122, 513–526 (2001)
13. Ishibuchi, H., Nii, M.: Fuzzy Regression using Asymmetric Fuzzy Coefficients and Fuzzified Neural Networks. Fuzzy Sets and Systems 119, 273–290 (2001)
14. Kacprzyk, J., Fedrizzi, M.: Fuzzy Regression Analysis. Omnitech Press, Warsaw and Physica–Verlag, Heidelberg (1992)
15. Kao, C., Chyu, C.-L.: A Fuzzy Linear Regression Model with Better Explanatory Power. Fuzzy Sets and Systems 126, 401–409 (2002)
16. Kim, B., Bishu, R.R.: Evaluation of Fuzzy Linear Regression Models by Comparing Membership Functions. Fuzzy Sets and Systems 100, 343–352 (1998)
17. Maple 9.5, Waterloo Maple Inc., Waterloo, Canada
18. MATLAB 7.0, The MathWorks, www.mathworks.com
19. Ming, M., Friedman, M., Kandel, A.: General Fuzzy Least Squares. Fuzzy Sets and Systems 88, 107–118 (1997)
20. Modarres, M., Nasrabadi, E., Nasrabadi, M.M.: Fuzzy Linear Regression Analysis from the Point of View Risk. Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems 12, 635–649 (2005)
21. Peters, G.: Fuzzy Linear Regression with Fuzzy Intervals. Fuzzy Sets and Systems 63, 45–55 (1994)
22. Redden, D.T., Woodall, W.H.: Properties of Certain Fuzzy Linear Regression Methods. Fuzzy Sets and Systems 64, 361–375 (1994)

23. Redden, D.T., Woodall, W.H.: Further Examination of Fuzzy Linear Regression. Fuzzy Sets and Systems 79, 203–211 (1996)
24. Sakawa, M., Yano, H.: Fuzzy Linear Regression Analysis for Fuzzy Input-Output Data. Information Sciences 63, 191–206 (1992)
25. Savic, D.A., Pedryzc, W.: Evaluation of Fuzzy Linear Regression Models. Fuzzy Sets and Systems 39, 51–63 (1991)
26. Tanaka, H.: Fuzzy Data Analysis by Possibilistic Linear Regression Models. Fuzzy Sets and Systems 24, 363–375 (1987)
27. Wang, Z.-Y., Li, S.-M.: Fuzzy Linear Regression Analysis of Fuzzy Valued Variables. Fuzzy Sets and Systems 36, 125–136 (1990)
28. Wang, H.-F., Tsaur, R.-C.: Insight of a Fuzzy Regression Model. Fuzzy Sets and Systems 112, 355–369 (2000)
29. Wu, H.-C.: Fuzzy Estimates of Regression Parameters in Linear Regression Models for Imprecise Input and Output Data. Computational Statistics and Data Analysis 42, 203–217 (2003)
30. Yang, M.-S., Lin, T.-S.: Fuzzy Least-Squares Linear Regression Analysis for Fuzzy Input-Output Data. Fuzzy Sets and Systems 126, 389–399 (2002)
31. Yang, M.-S., Liu, H.H.: Fuzzy Least-Squares Algorithms for Interactive Fuzzy Linear Regression Models. Fuzzy Sets and Systems 135, 305–316 (2003)
32. Yen, K.K., Ghoshray, S., Roig, G.: A Linear Regression Model Using Triangular Fuzzy Number Coefficients. Fuzzy Sets and Systems 106, 167–177 (1999)

12 Univariate Fuzzy Nonlinear Regression

12.1 Introduction

This chapter is concerned with fuzzy nonlinear regression. One of the problems in fuzzy linear regression is to find the "best" values of triangular (trapezoidal) (shaped) fuzzy numbers $\overline{A}$ and $\overline{B}$ so that the fuzzy linear function $\overline{Y} = \overline{A}\,\overline{X} + \overline{B}$ "explains" the fuzzy data $(\overline{X}_l, \overline{Z}_l)$, $1 \leq l \leq n$. In fuzzy nonlinear regression we are looking for a fuzzy polynomial, or a fuzzy exponential, or a fuzzy logarithmic, ..., function that "explains" the data. In the next section we discuss univariate (one independent variable) fuzzy nonlinear regression. The next chapter continues this discussion where there is more than one independent (explanatory) variable.

We will use non-negative fuzzy numbers in this chapter. We restrict ourselves to non-negative fuzzy numbers when we want our algebraic expressions to have the same value using the extension principle or using α-cuts and interval arithmetic. For example, $\overline{Y} = \overline{A}\ \overline{X}^2 + \overline{B}\ \overline{X} + \overline{C}$ produces the same $\overline{Y}$ using the extension principle, or α-cuts and interval arithmetic, when $\overline{A}$, $\overline{X}$, $\overline{B}$ and $\overline{C}$ are all non-negative.

What is normally done in regular (crisp) linear regression is to derive formulas for the estimates of the parameters in the regression model so that you can construct confidence intervals, do hypothesis testing, etc. Some of this can also be done in fuzzy linear regression. However, in our fuzzy nonlinear regression we have no formulas for the parameters so we are unable to do any "statistical" analysis of the regression models.

There have been relatively few papers on fuzzy nonlinear regression. See ([2],[6]-[8],[11]-[14]), and the references in these papers, for a survey of fuzzy nonlinear regression. We have used an evolutionary algorithm (EA) in fuzzy nonlinear regression ([3]-[5]). We used an evolutionary algorithm to choose the "best" fuzzy function, from a library of fuzzy functions, to explain the fuzzy data.

We use triangular (shaped) fuzzy numbers in this chapter but trapezoidal (shaped) fuzzy numbers are used in the following chapter. This chapter is based on [1].

12.2 Univariate Fuzzy Nonlinear Regression

Let $\mathcal{T}_0$ denote all triangular fuzzy numbers and $\mathcal{T}$ is all triangular shaped fuzzy numbers. Also, let $\mathcal{F}$ be all fuzzy numbers. We shall not be working a lot with general fuzzy numbers and all we need to know about $\overline{N}$ in $\mathcal{F}$ is that its membership function $y = \overline{N}(x)$ need not be continuous but $\overline{N}[\alpha]$ is always a closed, bounded, interval and there is some x so that $\overline{N}(x) = 1$. A function $F : \mathcal{T}_0 \to \mathcal{F}$ will be written $\overline{Y} = F(\overline{X}; \overline{K}_1, \ldots, \overline{K}_m)$ where $\overline{X}$ is the variable in $\mathcal{T}_0$ and the $\overline{K}_i$, $1 \leq i \leq m$, are constants also in $\mathcal{T}_0$. For example, the fuzzy quadratic $\overline{Y} = \overline{K}_2\, \overline{X}^2 + \overline{K}_1\, \overline{X} + \overline{K}_0$ is such a function. $\overline{Y}$ will be in $\mathcal{F}$ and possibly also in $\mathcal{T}$. Ω will be some fixed collection of $F : \mathcal{T}_0 \to \mathcal{F}$ called our library. We will discuss the composition of Ω later in this section.

The data is $(\overline{X}_l, \overline{Z}_l)$, $1 \leq l \leq n$, with $\overline{X}_l \in \mathcal{T}_0$ and $\overline{Z}_l \in \mathcal{T}$. The generation of the data, using specific members F of the library, is discussed in the next section. The restricted fuzzy nonlinear regression problem is to find the "best" $F \in \Omega$ that models (explains) the data. To define "best" we need to define a distance measure on $\mathcal{F}$. If $\overline{N}$, $\overline{M}$ are in $\mathcal{F}$, the distance between $\overline{M}$ and $\overline{N}$ will be taken to be

$$D(\overline{M}, \overline{N}) = \sup_{0 \leq \alpha \leq 1} \left\{ \max\{|m_1(\alpha) - n_1(\alpha)|, |m_2(\alpha) - n_2(\alpha)|\} \right\}, \tag{12.1}$$

where $\overline{M}[\alpha] = [m_1(\alpha), m_2(\alpha)]$, $\overline{N}[\alpha] = [n_1(\alpha), n_2(\alpha)]$. We measure the "best" through the error function

$$E = E(F) = \frac{1}{n} \sum_{l=1}^{n} D^2(\overline{Y}_l, \overline{Z}_l), \tag{12.2}$$

for $F \in \Omega$, $\overline{Y}_l = F(\overline{X}_l; \overline{K}_1, \ldots, \overline{K}_m)$, $1 \leq l \leq n$. This F is any member of the library. The restricted (to Ω) fuzzy nonlinear regression problem is to find $F^* \in \Omega$ so that

$$\inf_{F \in \Omega} (E(F)) = E(F^*). \tag{12.3}$$

Now we describe the basic library. First, set Ω_1 to be all $\overline{K}_2\, \overline{X} + \overline{K}_1$ for $\overline{K}_1$, $\overline{K}_2 \in \mathcal{T}_0$. Ω_1 contains all linear fuzzy functions. Ω_2 will be all

$$\overline{K}_m\, \overline{X}^m + \overline{K}_{m-1}\, \overline{X}^{m-1} + \cdots + \overline{K}_1\, \overline{X} + \overline{K}_0, \tag{12.4}$$

for $\overline{K}_i \in \mathcal{T}_0$, $0 \leq i \leq m$, and $m \leq \Delta$, Δ some positive integer. Ω_2 contains all fuzzy polynomials of degree less than, or equal to, Δ. Ω_3 is all

$$\overline{K}_3 \exp(\overline{K}_2\, \overline{X}) + \overline{K}_1, \tag{12.5}$$

for $\overline{K}_i \in \mathcal{T}_0$, $1 \leq i \leq 3$. Fuzzy exponentials make up Ω_3. Lastly, Ω_4 is all

$$\overline{K}_3 \ln(\overline{X} + \overline{K}_2) + \overline{K}_1, \tag{12.6}$$

for $\overline{K}_i \in \mathcal{T}_0$, $1 \leq i \leq 3$, with $\overline{X} + \overline{K}_2 \geq 1$. If $\overline{X} + \overline{K}_2 = (a/b/c)$, we require $a \geq 1$. Then

$$\Omega = \bigcup_{i=1}^{4} \Omega_i. \tag{12.7}$$

12.2.1 Evolutionary Algorithm (EA)

To test our EA (evolutionary algorithm) we picked a $G(\overline{X};\overline{K}_1,\dots,\overline{K}_m) \in \Omega$ to generate the data and then asked the EA to find G. We did this two ways: (1) first the data was $(\overline{X}_l,\overline{Z}_l)$, $\overline{Z}_l = G(\overline{X}_l;\overline{K}_1,\dots,\overline{K}_m)$, $1 \le l \le n$; and (2) secondly, the data $(\overline{X}_l,\overline{Z}_l)$, $\overline{Z}_l = G(\overline{X}_l;\overline{K}_1,\dots,\overline{K}_m)+\varepsilon_l$, $1 \le l \le n$, had "noise" in it with ε_l distributed $N(0,\sigma^2)$. We will now only discuss the first case.

For the no noise case there were four tests: (1) $\overline{Z} = \overline{K}_2\ \overline{X} + \overline{K}_1$; (2) $\overline{Z} = \overline{K}_2\ \overline{X}^2 + \overline{K}_1\ \overline{X} + \overline{K}_0$; (3) $\overline{Z} = \overline{K}_3 \exp(\overline{K}_2\ \overline{X}) + \overline{K}_1$; and (4) $\overline{Z} = \overline{K}_3 \ln(\overline{X} + \overline{K}_2) + \overline{K}_1$ for given values of the $\overline{K}_i$. Consider case (2) above where the "unknown" function was $G(\overline{X};\overline{K}_0,\overline{K}_1,\overline{K}_2) = \overline{K}_2\ \overline{X}^2 + \overline{K}_1\ \overline{X} + \overline{K}_0$. The EA found $F(\overline{X};\overline{C},\overline{B},\overline{A}) = \overline{A}\ \overline{X}^2 + \overline{B}\ \overline{X} + \overline{C}$ where $\overline{A} \approx \overline{K}_2$, $\overline{B} \approx \overline{K}_1$ and $\overline{C} \approx \overline{K}_0$. In all tests the EA essentially found the unknown G.

The EA used to discover the correct function was in two parts. In the data let $\overline{X}_l = (x_{l1}/x_{l2}/x_{l3})$ in $\mathcal{T}_0$ and $\overline{Z}_l \approx (z_{l1}/z_{l2}/z_{l3}) \in \mathcal{T}$, $1 \le l \le n$. The first EA, EA$_1$, attempts to fit a crisp linear, polynomial of degree $\le \Delta$, exponential and logarithmic function to the crisp data (x_{l2}, z_{l2}), $1 \le l \le n$. The model with the smallest error was chosen for the next phase. For example, if $\overline{Z} = \overline{K}_2\ \overline{X}^2 + \overline{K}_1\ \overline{X}+\overline{K}_0$, then EA$_1$ would pick $f(x) = ax^2+bx+c$ for some a, b, c. The second EA, EA$_2$, would look for $\overline{A}$, $\overline{B}$, and $\overline{C}$ so that $F(\overline{X};\overline{C},\overline{B},\overline{A}) = \overline{A}\ \overline{X}^2 + \overline{B}\ \overline{X}+\overline{C}$ would fit the data, or $E(F)$ would be acceptably small. But, in order to make EA$_2$ run faster, we first cut down on the search space. Based on the results of EA$_1$ intervals J_i where computed for the unknown parameters $\overline{K}_i$. That is, for $\overline{Z} = \overline{K}_2\ \overline{X}^2 + \overline{K}_1\ \overline{X} + \overline{K}_0$, EA$_2$ only looked in J_1 for $\overline{A}$, J_2 for $\overline{B}$ and J_3 for $\overline{C}$. In all cases (with or without noise) the EA essentially found the unknown G in Ω.

12.3 Fuzzy Monte Carlo Method

Now we plan to achieve similar results described above for the EA using our fuzzy Monte Carlo method. We will not do the part completed by EA$_1$ discussed above because that can be done by any crisp statistical software. Assume first we choose a fuzzy quadratic in the library to generate the data. The case of choosing a fuzzy logarithmic will be discussed later in the chapter. Let the fuzzy quadratic be (as in [3]) $\overline{Z} = \overline{A}\ \overline{X}^2 + \overline{B}\ \overline{X} + \overline{C}$ for $\overline{A} = (2.2/2.5/2.7)$, $\overline{B} = (4.6/5.1/5.3)$ and $\overline{C} = (2.9/3.2/3.6)$. We randomly generate $n = 30$ values of $\overline{X}$ in $[0, 40]$ to produce the data $(\overline{X}_l,\overline{Z}_l)$ using this fuzzy quadratic. Then the $\overline{Z}_l$ will be triangular shaped fuzzy numbers. We will not consider the case of "noise" in this chapter. Before we can go on we need to discuss how we will produce sequences of random fuzzy vectors.

To obtain random sequences $\overline{V}_k = (\overline{X}_{k1},...,\overline{X}_{kp})$, $k = 1,2,...,N$, where the $\overline{X}_{ki}$ are all triangular fuzzy numbers, we first randomly generate crisp vectors $v_k = (x_{k1},...,x_{k,3p})$ with all the x_{ki} in $[0,1]$, $k = 1,2,..,N$. We choose the first three numbers in v_k and order them from smallest to largest. Assume that

$x_{k3} < x_{k1} < x_{k2}$. Then the first triangular fuzzy number $\overline{X}_{k1} = (x_{k3}/x_{k1}/x_{k2})$. Continue with the next three numbers in v_k, etc. making $\overline{X}_{ki}$, $i = 2, ..., p$. This was all discussed in Chapter 4.

However the $\overline{X}_{ki}$ we want will need to be in certain intervals. Suppose $\overline{X}_{ki}$ is to be in interval $I_i = [a_i, b_i]$, $i = 1, 2, ..., p$. Since each $\overline{X}_{ki}$ starts out in $[0, 1]$ we may easily put it into $[a_i, b_i]$ by computing $a_i + (b_i - a_i)\overline{X}_{ki}$, $i = 1, ..., p$.

To randomly generate $\overline{X}_l \in [0, 40]$, $1 \leq l \leq 30$, we use our Sobol quasi-random number generator to obtain random vector $v_l = (v_{l1}, v_{l2}, v_{l3}) \in [0, 1]^3$, $1 \leq l \leq 30$. Order the components in v_l from smallest to largest, say $v_{l2} < v_{l3} < v_{l1}$, and then set $x_{l1} = v_{l2}$, $x_{l2} = v_{l3}$ and $x_{l3} = v_{l1}$ and define triangular fuzzy number $\overline{X}_l = 40(x_{l1}/x_{l2}/x_{l3})$, $1 \leq l \leq 30$. Now compute

$$\overline{Z}_l = \overline{A}\,\overline{X}_l^2 + \overline{B}\,\overline{X}_l + \overline{C}, \tag{12.8}$$

for $l = 1, 2, ..., 30$. The $\overline{A}$, $\overline{B}$ and $\overline{C}$ in the above equation were given above. The data is $(\overline{X}_l, \overline{Z}_l)$, $1 \leq l \leq 30$. All fuzzy arithmetic is to be done using $\alpha-$cuts and interval arithmetic with α going from zero to one in increments of 0.1. So the data is really $(\overline{X}_l[\alpha], \overline{Z}_l[\alpha])$ for $l = 1, 2, ..., 30$ and $\alpha = 0.0, 0.1, 0.2, ..., 0.9, 1.0$.

Randomly generate, using a quasi-random number generator, random vectors $v_k = (x_{k1}, ..., x_{k9})$, $k = 1, 2, ..., N$, using the first three numbers (as discussed above) to define triangular fuzzy number $\overline{A}_k$, the next three numbers for $\overline{B}_k$ and the last three numbers for $\overline{C}_k$. Next we need to determine intervals $I_i = [a_i, b_i]$, $i = 1, 2, 3$, for these fuzzy numbers. A good selection of these intervals will make the fuzzy Monte Carlo process more efficient because: (1) if they are wrong and/or too small we can miss a "good" solution; and (2) if they are too big the simulation can spend too much time looking at situations that will not produce a "good" solution. Defining the intervals I_i is application dependent so we now proceed to the applications. We first consider the fuzzy quadratic function and then the fuzzy logarithmic function.

12.4 Fuzzy Quadratic

This application is to use our new Monte Carlo method to find the values of $\overline{A}$, $\overline{B}$ and $\overline{C}$ that produced the data $(\overline{X}_l, \overline{Z}_l)$, $1 \leq l \leq 30$, which is generated as described above. We will need to find intervals I_i, $i = 1, 2, 3$, as explained above. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the intervals from [3]; and (2) secondly, in the second subsection, we use two optimization procedures to determine these intervals.

All programs were written in MATLAB [10]. For all our Monte Carlo calculations, we used a Pentium III, Processor: 933 MHz.

12.4.1 First Choice of Intervals

The selection of these intervals was discussed in detail in [3]. However, those intervals turned out to be much too large, having large negative components, and the fuzzy coefficients are known to be positive. So we take only the positive

Table 12.1. Results of the Monte Carlo (QMCI) Method, First Choice of Intervals, to Minimize the Error

Coefficient	$QMCI$
$\overline{A}$	(2.2060/2.5085/2.6942)
$\overline{B}$	(4.6798/4.8534/5.4691)
$\overline{C}$	(2.7931/3.0474/3.4434)

part of the intervals calculated in [3] to be: (1) $[0, 4.2]$ for $\overline{A}$; (2) $[0, 11.2]$ for $\overline{B}$; and (3) $[0, 171.8]$ for $\overline{C}$. Assume that we have already ordered the first three components in $v_k = (x_{k1}, ..., x_{k9})$ from smallest to largest, same for the next three and the last three, then $\overline{A}_k = 4.2(x_{k1}/x_{k2}/x_{k3})$, $\overline{B}_k = 11.2(x_{k4}/x_{k5}/x_{k6})$ and $\overline{C}_k = 171.8(x_{k7}/x_{k8}/x_{k9})$. Let

$$\overline{Y}_{kl} = \overline{A}_k \ \overline{X}_l^2 + \overline{B}_k \ \overline{X}_l + \overline{C}_k, \tag{12.9}$$

for $k = 1, 2, ..., N$ and $l = 1, ..., 30$. Now compute $D_{kl} = D(\overline{Z}_l, \overline{Y}_{kl})$, $l = 1, ..., 30$, from equation (12.1), for $\alpha = 0, 0.1, ..., 0.9, 1.0$, and then the error is

$$E_k = (1/30) \sum_{l=1}^{30} D_{kl}^2, \tag{12.10}$$

as in equation (12.2), for $k = 1, ..., N$.

Using our Sobol quasi-random number generator we produced $N = 200,000$ vectors $v_k = (x_{k1}, ..., x_{k9})$ which defined the $\overline{A}$, $\overline{B}$ and $\overline{C}$ as described above. Results are shown in Table 12.1 with minimum error value in Table 12.5. Since we will have four Monte Carlo studies for the fuzzy quadratic we call this one QMCI (for quadratic Monte Carlo).

Next we experimented with other intervals. We started with larger intervals, shifted them and shortened them, until we arrived at: (1) $[2.101, 2.701]$ for $\overline{A}$; (2) $[4.601, 5.401]$ for $\overline{B}$; and (3) $[2.801, 3.801]$ for $\overline{C}$. After another run of $N = 200,000$ quasi-random vectors the results for $\overline{A}$, $\overline{B}$ and $\overline{C}$ are in Table 12.2 with minimum error value in Table 12.5. This Monte Carlo study is called QMCII.

Table 12.2. Results of the Monte Carlo (QMCII) Method, First Choice of Intervals, to Minimize the Error

Coefficient	$QMCII$
$\overline{A}$	(2.2001/2.5002/2.6999)
$\overline{B}$	(4.6011/5.0950/5.3041)
$\overline{C}$	(2.8589/3.2241/3.5759)

12.4.2 Second Choice of Intervals

The first thing to do is to determine the intervals $I_i = [a_i, b_i]$, $i = 1, 2, 3$, for $\overline{A}(i = 1)$, $\overline{B}(i = 2)$ and $\overline{C}(i = 3)$. We first describe an optimization method used to determine these intervals. This procedure will be called *QMCIII*. A second optimization method will be used and it will be described below. Let

$$[L_l, R_l] = I_1[x_{1l}^2, x_{l3}^2] + I_2[x_{l1}, x_{l3}] + I_3, \tag{12.11}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$. Recall the data $\overline{Z}_l \approx (z_{l1}/z_{l2}/z_{l3})$. Define

$$W = \sum_{l=1}^{30}(L_l - z_{l1})^2 + \sum_{l=1}^{30}(R_l - z_{l3})^2. \tag{12.12}$$

The optimization problem is to minimize W subject to $a_i \leq b_i$ all i. We want to find the intervals that make L_l and R_l closest, in the sense of minimizing W, to the end points of the bases of the $\overline{Z}_l$ in the data. We solved this using Maple [9]. The results, rounded to three decimal places, are in Table 12.3.

Table 12.3. First Method of Determining the Intervals (QMCIII) for Monte Carlo

Interval	Value
I_1 for $\overline{A}$	$[2.200, 2.701]$
I_2 for $\overline{B}$	$[4.600, 5.310]$
I_3 for $\overline{C}$	$[2.900, 3.601]$

We now produce a sequence of random crisp vectors $v_k = (x_{k1}, ..., x_{k9})$, $k = 1, 2, ..., N$, using our Sobol quasi-random number generator as described above, to get a sequence of triangular fuzzy numbers $\overline{A}_k$, $\overline{B}_k$ and $\overline{C}_k$ in their intervals and computed E in equation (12.2). After a run of $N = 200,000$ the smallest E value found is shown in Table 12.5 with corresponding $\overline{A}$, $\overline{B}$, $\overline{C}$ shown in Table 12.4.

We also investigated a second optimization procedure for finding the intervals. L_l and R_l are defined as above. This method is called QMCIV. Let

$$W = \sum_{l=1}^{30}(z_{l1} - L_l) + \sum_{l=1}^{30}(R_l - z_{l3}). \tag{12.13}$$

Table 12.4. Results of the Monte Carlo Method (QMCIII), Second Choice of Intervals, to Minimize the Error

Coefficient	*QMCIII*
$\overline{A}$	$(2.2002/2.5001/2.6999)$
$\overline{B}$	$(4.6001/5.0952/5.3042)$
$\overline{C}$	$(2.9003/3.2359/3.5741)$

The linear programming problem is to minimize W subject to the constraints: (1) $L_l \leq z_{l1}$ all l; (2) $R_l \geq z_{l3}$ all l; and (3) $a_i \leq b_i$ all i. We solved this problem of determining intervals using Maple [9] and the results are the same as shown in Table 12.3. Hence, the results for QMCIV will be the same as for QMCIII.

12.4.3 Comparison of Solutions

We see (Table 12.5) that the error measure for QMCI is much too large compared to the error in EA, QMCII and QMCIII. This was because the intervals for $\overline{A}$, $\overline{B}$ and $\overline{C}$ were too large for a Monte Carlo run of $N = 200,000$. Possibly $N = 1,000,000$ could result in a much smaller error for QMCI. The error measures for QMCII and QMCIII were comparable to that for the EA method. The computing time for: (1) QMCI and QMCII was between 22 and 25 minutes; and (2) for QMCIII was also between 22 and 25 minutes.

Table 12.5. Comparing Fuzzy Monte Carlo Solutions and EA Solution to the Fuzzy Quadratic Data

Method	Error(E)
EA	0.0078
QMCI	8.5658
QMCII	0.0025
QMCIII	0.0034

12.5 Fuzzy Logarithmic

The procedure here will be very similar to that for the fuzzy quadratic discussed above so we will not present as many details. To produce the data $(\overline{Z}_l, \overline{X}_l)$, $1 \leq l \leq 30$ let $\overline{Z}_l = \overline{C} + \overline{A}\ln(\overline{X}_l + \overline{B})$ for $\overline{A} = (2.1/2.7/2.9)$, $\overline{B} = (3.6/4.1/4.4)$, $\overline{C} = (0.8/1.1/1.7)$ for 30 random values for triangular fuzzy number $\overline{X}$ in $[0, 40]$. We evaluate the fuzzy logarithmic function using α-cuts and interval arithmetic (all intervals are positive).

This application is to use our new Monte Carlo method to find the values of $\overline{A}$, $\overline{B}$ and $\overline{C}$ that produced the data $(\overline{X}_l, \overline{Z}_l)$, $1 \leq l \leq 30$. We will need to find intervals I_i, $i = 1, 2, 3$, for the "unknown" fuzzy parameters. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the intervals from [3]; and (2) secondly, in the second subsection, we use two optimization procedures to determine these intervals.

12.5.1 First Choice of Intervals

The selection of these intervals was discussed in detail in [3]. The intervals calculated in [3] were: (1) $I_1 = [1.501, 3.601]$ for $\overline{A}$; (2) $I_2 = [2.105, 24.905]$ for $\overline{B}$; and (3) $I_3 = [0.897, 1.797]$ for $\overline{C}$. Using our Sobol quasi-random number generator we produced $N = 200,000$ vectors $v_k = (x_{k1}, ..., x_{k9})$ which defined the $\overline{A}$, $\overline{B}$ and

Table 12.6. Results of the Monte Carlo (LMCI) Method, First Choice of Intervals, to Minimize the Error

Coefficient	*LMCI*
$\overline{A}$	(1.9651/2.5743/2.9242)
$\overline{B}$	(3.9632/4.8364/5.0394)
$\overline{C}$	(0.9473/1.4353/1.7133)

Table 12.7. Results of the Monte Carlo (LMCII) Method, First Choice of Intervals, to Minimize the Error

Coefficient	*LMCII*
$\overline{A}$	(2.1067/2.6212/2.9526)
$\overline{B}$	(3.6011/3.7972/4.1890)
$\overline{C}$	(0.8101/1.3787/1.5461)

$\overline{C}$ in their respective intervals. Results are shown in Table 12.6 with minimum error value (equation (12.2)) in Table 12.10. Since we will have four Monte Carlo studies for the fuzzy logarithmic we call this one LMCI (for logarithmic Monte Carlo).

Next we experimented with other intervals. We started with larger intervals, shifted them and shortened them, until we arrived at: (1) $[2.101, 3.001]$ for $\overline{A}$; (2) $[3.601, 4.201]$ for $\overline{B}$; and (3) $[0.801, 1.601]$ for $\overline{C}$. After another run of $N = 200,000$ quasi-random vectors the results for $\overline{A}$, $\overline{B}$ and $\overline{C}$ are in Table 12.7 with minimum error value in Table 12.10. This Monte Carlo study is called LMCII.

12.5.2 Second Choice of Intervals

The first thing to do is to determine the intervals $I_i = [a_i, b_i]$ for $\overline{A}$ $(i = 1)$, $\overline{B}$ $(i = 2)$ and $\overline{C}$ $(i = 3)$. We first describe an optimization method used to determine these intervals. This procedure will be called *LMCIII*. A second optimization method will be used and it will be described below. Let

$$[L_l, R_l] = I_1[\ln(x_{l1} + a_2), \ln(x_{l3} + b_2)] + I_3, \tag{12.14}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$. Define

$$W = \sum_{l=1}^{30}(L_l - z_{l1})^2 + \sum_{l=1}^{30}(R_l - z_{l3})^2. \tag{12.15}$$

The optimization problem is to minimize W subject to $a_i \leq b_i$ all i. We solved this using Maple [9]. The results, rounded to three decimal places, are in Table 12.8.

We now produce a sequence of random crisp vectors $v_k = (x_{k1}, ..., x_{k9})$, $k = 1, 2, ..., N$, using our Sobol quasi-random number generator as described above,

Table 12.8. First Method of Determining the Intervals (LMCIII) for Monte Carlo

Interval	Value
I_1 for $\overline{A}$	$[2.100, 2.990]$
I_2 for $\overline{B}$	$[3.600, 4.400]$
I_3 for $\overline{C}$	$[0.800, 1.700]$

Table 12.9. Results of the Monte Carlo Method (LMCIII), Second Choice of Intervals, to Minimize the Error

Coefficient	$LMCIII$
$\overline{A}$	(2.1003/2.6217/2.9519)
$\overline{B}$	(3.6041/3.8629/4.1861)
$\overline{C}$	(0.8109/1.3440/1.5821)

to get a sequence of triangular fuzzy numbers $\overline{A}_k$, $\overline{B}_k$ and $\overline{C}_k$ in their respective intervals and computed E in equation (12.2). After a run of $N = 200,000$ the smallest E value found is shown in Table 12.10 with corresponding $\overline{A}$, $\overline{B}$, $\overline{C}$ shown in Table 12.9.

We also investigated a second optimization procedure for finding the intervals. L_l and R_l are defined as above. This method is called QMCIV. Let

$$W = \sum_{l=1}^{30}(z_{l1} - L_l) + \sum_{l=1}^{30}(R_l - z_{l3}). \tag{12.16}$$

The optimization problem is to minimize W subject to the constraints: (1) $L_l \leq z_{l1}$ all l; (2) $R_l \geq z_{l3}$ all l; and (3) $a_i \leq b_i$ all i. We solved this problem of determining intervals using Maple [9] and the results are same as in Table 12.8. So the results for LMCIV will be the same as for LMCIII.

12.5.3 Comparison of Solutions

We see (Table 12.10) that the error measure for LMCI is slightly larger than the error in EA, LMCII and LMCIII. This was because the intervals for $\overline{A}$, $\overline{B}$ and

Table 12.10. Comparing Fuzzy Monte Carlo Solutions and EA Solution to the Fuzzy Logarithmic Data

Method	Error(E)
EA	0.0030
LMCI	0.0473
LMCII	0.0009
LMCIII	0.0025

$\overline{C}$ were larger for a Monte Carlo run of $N = 200,000$. Possibly $N = 1,000,000$ could result in a much smaller error for LMCI. The error measures for LMCII and LMCIII were comparable to that for the EA method. The computing time for LMCI, LMCII and LMCIII was between 36 and 40 minutes.

12.6 Summary and Conclusions

We have shown that our new fuzzy Monte Carlo method can do better than the evolutionary algorithm method on two fuzzy nonlinear regression problems: a fuzzy quadratic equation and a fuzzy logarithmic equation. An interested reader may obtain our data and MATLAB programs by contacting the authors. Of course more research is needed to conclude that our fuzzy Monte Carlo method can do as well and better than an evolutionary algorithm in fuzzy regression. However we believe that fuzzy Monte Carlo will become very important in solving fuzzy optimization problems.

Evolutionary algorithms can be difficult to construct; often a special algorithm for each fuzzy optimization problem is needed. But, just as every high-level computer language has a pseudo-random number generator, every language can have a quasi-random number generator and then a random fuzzy vector generator. Everyone can then use their random fuzzy vector generator to solve fuzzy optimization problems.

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Nonlinear Regression (under review)
2. Acton, S.T.: On Fuzzy Nonlinear Regression for Image Enhancement. J. Mathematical Imaging and Vision 8, 239–253 (1998)
3. Buckley, J.J., Feuring, T.: Linear and Nonlinear Fuzzy Regression: Evolutionary Algorithm Solutions. Fuzzy Sets and Systems 112, 381–394 (2000)
4. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
5. Buckley, J.J., Feuring, T., Hayashi, Y.: Multivariate Nonlinear Fuzzy Regression: An Evolutionary Algorithm Approach. Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems 7, 83–98 (1999)
6. Chang, Y.-H.O.: Hybrid Fuzzy Least-Squares Regression Analysis and Its Reliability Measures. Fuzzy Sets and Systems 119, 225–246 (2001)
7. Hong, D.H., Hwang, C.: Support Vector Fuzzy Regression Machines. Fuzzy Sets and Systems 138, 271–281 (2003)
8. Ishibuchi, H., Nii, M.: Fuzzy Regression using Asymmetric Fuzzy Coefficients and Fuzzified Neural Networks. Fuzzy Sets and Systems 119, 273–290 (2001)
9. Maple 9.5, Waterloo Maple Inc., Waterloo, Canada
10. MATLAB, The MathWorks, http://www.mathworks.com
11. Ruoning, X.: S-Curve Regression Model in Fuzzy Environment. Fuzzy Sets and Systems 90, 317–326 (1997)

12. Yabuuchi, Y., Watada, J., Tatsumi, T.: Fuzzy Regression Analysis of Data with Error. Japanese J. of Fuzzy Theory and Systems 6, 673–685 (1994)
13. Zhang, X., Omachi, S., Aso, H.: Fuzzy Regression Analysis Using RFLN and its Application. In: Proc. FUZZ-IEEE 1997, Barcelona, Spain, July 1-5,1997, vol. 1, pp. 51–56 (1997)
14. Zhang, D., Deng, L.-F., Cai, K.-Y., So, A.: Fuzzy Nonlinear Regression with Fuzzified Radial Basis Function Network. IEEE Trans. Fuzzy Systems 13, 742–760 (2005)

13 Multivariate Nonlinear Regression

13.1 Introduction

This chapter continues the previous chapter but now we can have two or more independent (predictor, explanatory) variables, the $\overline{X}_i$. Another change from Chapter 12 is now we will work with fuzzy trapezoidal (shaped) fuzzy numbers instead of fuzzy triangular (shaped) fuzzy numbers. As in Chapter 12 we usually assume our fuzzy numbers are non-negative so that in the evaluation of fuzzy expressions the extension principle and the α-cut and interval arithmetic method produce the same results. However, in Section 13.5.4 we allow fuzzy numbers in an interval $I = [-a, a]$, $a > 0$. This chapter is based on [1].

We will need a metric (distance measure) on the collection of fuzzy numbers. Since α-cuts of fuzzy numbers are always closed, bounded, intervals, we will use

$$D(\overline{N}, \overline{M}) = sup_\alpha \{max[|m_1(\alpha) - n_1(\alpha)|, |m_2(\alpha) - n_2(\alpha)|\ ]\}, \tag{13.1}$$

where $\overline{M}[\alpha] = [m_1(\alpha), m_2(\alpha)]$ and $\overline{N}[\alpha] = [n_1(\alpha), n_2(\alpha)]$, all α. This is the same distance measure used in Chapter 12.

In the next section we present a universal approximation result for fuzzy functions. The third section discusses evolutionary algorithm results followed by, in the fourth section, a presentation of our fuzzy Monte Carlo method to be used in this chapter. The fifth section first looks at three applications given in [2] and then a new application for our fuzzy Monte Carlo method. The last section has a brief summary and our conclusions.

13.2 Universal Approximator

The result in this section was proven in [2] and we include it here to explain the nonlinear regression library defined in the next section and for Section 13.5.4.

Let $\mathcal{T}_0$ be all trapezoidal fuzzy numbers, $\mathcal{T}$ is all trapezoidal shaped fuzzy numbers and $\mathcal{F}$ denotes all fuzzy numbers. To simplify the discussion for the rest of this chapter we will only be working with two independent variables, so we will be interested in functions like $\overline{Z} = F(\overline{X}_1, \overline{X}_2; \overline{K}_1, \ldots, \overline{K}_m)$ where $\overline{X}_1$,

$\overline{X}_2$, $\overline{K}_1, \ldots, \overline{K}_m$ are all in $\mathcal{T}_0$, $\overline{X}_1$ in interval $[a,b]$, $\overline{X}_2$ in interval $[c,d]$, the parameters $\overline{K}_i$ are in intervals I_i, $1 \leq i \leq m$, and $\overline{Z}$ is in $\mathcal{F}$ and possibly also in $\mathcal{T}$.

We obtain such an F via the extension principle. Let $f(x_1, x_2; k_1, \ldots, k_m)$: $[a,b] \times [c,d] \to \mathbf{R}$ where the parameters k_i belong to closed, bounded, intervals I_i, $1 \leq i \leq m$. Although the k_i are constants we will consider f having $m+2$ variables so it will be a continuous mapping from $[a,b] \times [c,d] \times \prod_{i=1}^{m} I_i$ into $\mathbf{R}$. Now we extend f, using the extension principle, to $F(\overline{X}_1, \overline{X}_2; \overline{K}_1, \ldots, \overline{K}_m)$ for $\overline{X}_i$ in $\mathcal{T}_0$, $\overline{X}_1$ in $[a,b]$, $\overline{X}_2$ in $[c,d]$ and all the $\overline{K}_i$ in $\mathcal{T}_0$ with $\overline{K}_j$ in I_j, $1 \leq j \leq m$. Let $\overline{Z} = F(\overline{X}_1, \overline{X}_2; \overline{K}_1, \ldots, \overline{K}_m)$ with $\overline{Z}$ in $\mathcal{F}$.

We will use the notation $p_\theta(x_1, x_2; k_1, \ldots, k_m)$ for a polynomial in variables $x_1, x_2, k_1, \ldots, k_m$ of degree d_1 in x_1, d_2 in x_2, d_3 in k_1, $\ldots$, d_{m+2} in k_m with $\theta = (d_1, d_2; d_3, \ldots, d_{m+2})$. Given $\epsilon > 0$, there is a p_θ so that [5]

$$|f(x_1, x_2; k_1, \ldots, k_m) - p_\theta(x_1, x_2; k_1, \ldots, k_m)| < \epsilon, \tag{13.2}$$

for all $x_1 \in [a,b]$, $x_2 \in [c,d]$ and $k_j \in I_j$, $1 \leq j \leq m$.

Now use the extension principle to extend p_θ to $P_\theta(\overline{X}_1, \overline{X}_2; \overline{K}_1, \ldots, \overline{K}_m) = \overline{Y}$. Then [2] using the same ϵ as in equation (13.2)

$$D(F(\overline{X}_1, \overline{X}_2; \overline{K}_1, \ldots, \overline{K}_m) - P_\theta(\overline{X}_1, \overline{X}_2; \overline{K}_1, \ldots, \overline{K}_m)) < \epsilon, \tag{13.3}$$

for all $\overline{X}_1 \in [a,b]$, $\overline{X}_2 \in [c,d]$ and all $\overline{K}_j \in I_j$, $1 \leq j \leq m$.

Let $\mathcal{C}$ be all F which are extension principle extensions of continuous f defined above. Let $\mathcal{C}_0$ be all P_θ which are extension principle extensions of multivariate polynomials p_θ also defined above. Then $\mathcal{C}_0$ is an universal approximator for $\mathcal{C}$. However, we will use (and also in [2]) α-cuts and interval arithmetic to evaluate fuzzy functions like F and P_θ. Let P_θ^* be the extension of p_θ using α-cuts and interval arithmetic. Then it is well known (Chapter 2) that $P_\theta \leq P_\theta^*$ and they may not be equal. But if all the fuzzy numbers are non-negative we get $P_\theta = P_\theta^*$. Hence, all our fuzzy numbers in the rest of the chapter will be non-negative except in Section 13.5.4.

Suppose we have some fuzzy data $((\overline{X}_{1l}, \overline{X}_{2l}), \overline{Z}_l)$, $1 \leq l \leq p$, generated by some $F \in \mathcal{C}$. Then, from the result above, there is a $P_\theta \in \mathcal{C}_0$ that can model this data. Hence our multivariate fuzzy nonlinear regression library, discussed in the next section, will be composed of P_θ in $\mathcal{C}_0$.

13.3 Evolutionary Algorithm

Let the library be $\mathcal{L}$. This library will contain multivariate, only two independent variables, fuzzy polynomials P_θ which are sums of terms like

$$\overline{C}\ \overline{X}_1^{\gamma} \overline{X}_2^{\mu}, \tag{13.4}$$

with $\lambda = \gamma + \mu \leq \delta$ for some positive integer δ. The largest value of λ is called the degree of P_θ and we will assume that (as in [2]) $\delta \leq 4$. We assume that all the fuzzy numbers are non-negative and in $\mathcal{T}_0$ and $\overline{X}_1 \in [a,b]$, $\overline{X}_2 \in [c,d]$.

To test our EA in [2] we picked an $F \in \mathcal{L}$ to generate the fuzzy data and then asked the EA to find this F. Our EA was in two parts EA$_1$ and EA$_2$. EA$_1$ first decided if λ was $1, 2, 3$ or 4. This can be performed by crisp regression software and will be omitted in this chapter. Suppose EA$_1$ decided $\lambda = 2$ so

$$F = \overline{K}_1\overline{X}_1^2 + \overline{K}_2\overline{X}_1\overline{X}_2 + \overline{K}_3\overline{X}_2^2 + \overline{K}_4\overline{X}_1 + \overline{K}_5\overline{X}_2 + \overline{K}_6. \tag{13.5}$$

Then EA$_2$ found

$$P_\theta = \overline{C}_1\overline{X}_1^2 + \overline{C}_2\overline{X}_1\overline{X}_2 + \overline{C}_3\overline{X}_2^2 + \overline{C}_4\overline{X}_1 + \overline{C}_5\overline{X}_2 + \overline{C}_6, \tag{13.6}$$

with $\overline{C}_j \approx \overline{K}_j$, $1 \le j \le 6$. We will discuss these tests in more detail in Section 13.5.

We now define the error measure used in [2] and in this chapter. Let the data be $((\overline{X}_{1l}, \overline{X}_{2l}), \overline{Z}_l)$, $1 \le l \le p$, generated by some $F \in \mathcal{L}$. Also let $\overline{Y}_l = P_\theta(\overline{X}_{1l}, \overline{X}_{2l}; \overline{C}_1, \ldots, \overline{C}_6)$, $1 \le l \le p$. Then

$$Error = E = (1/p)\sum_{l=1}^{p} D^2(\overline{Z}_l, \overline{Y}_l). \tag{13.7}$$

The EA and our fuzzy Monte Carlo method want to find the $\overline{C}_i$ to minimize E. A major problem in the EA and fuzzy Monte Carlo method is to find intervals for the $\overline{C}_i$. That is, determine $I_i = [a_i, b_i]$ so that $\overline{C}_i \in [a_i, b_i]$ all i. A good choice for these intervals will make both methods more efficient.

13.4 Fuzzy Monte Carlo Method

Now we plan to achieve similar results described above for the EA using our fuzzy Monte Carlo method. Assume first we choose a fuzzy linear function in the library to generate the data. Let the fuzzy linear function be (as in [2]) $\overline{Z} = \overline{K}_1\ \overline{X}_1 + \overline{K}_2\overline{X}_2 + \overline{K}_3$ for $\overline{K}_1 = (3.0/3.2, 3.7/4.2)$, $\overline{K}_2 = (2.0/3.0, 4.0/5.0)$ and $\overline{K}_3 = (4.5/5.0, 5.5/6.0)$. We randomly generate $p = 30$ values of $\overline{X}_i$ in $[0, 30]$, $i = 1, 2$, to produce the data $((\overline{X}_{1l}, \overline{X}_{2l}; \overline{Z}_l)$, $1 \le l \le 30$, using this fuzzy linear function. Then the $\overline{Z}_l$ will be trapezoidal shaped fuzzy numbers. We will not consider the case of "noise" in this chapter which was discussed in [2].

To obtain random sequences $\overline{V}_k = (\overline{X}_{k1}, ..., \overline{X}_{kp})$, $k = 1, 2, ..., N$, where the $\overline{X}_{ki}$ are all trapezoidal fuzzy numbers, we first randomly generate crisp vectors $v_k = (x_{k1}, ..., x_{k,4p})$ with all the x_{ki} in $[0, 1]$, $k = 1, 2, .., N$. We choose the first four numbers in v_k and order them from smallest to largest. Assume that $x_{k3} < x_{k1} < x_{k2} < x_{k4}$. Then the first trapezoidal fuzzy number $\overline{X}_{k1} = (x_{k3}/x_{k1}, x_{k2}/x_{k4})$. Continue with the next four numbers in v_k, etc. making $\overline{X}_{ki}$, $i = 2, ..., p$.

However the $\overline{X}_{ki}$ we want will need to be in certain intervals. Suppose $\overline{X}_{ki}$ is to be in interval $I_i = [a_i, b_i]$, $i = 1, 2, ..., p$. Since each $\overline{X}_{ki}$ starts out in $[0, 1]$ we may easily put it into $[a_i, b_i]$ by computing $a_i + (b_i - a_i)\overline{X}_{ki}$, $i = 1, ..., p$.

How shall we make the random crisp vectors v_k? We use our Sobol quasi-random number generator discussed in Chapter 3.

Now we return to the example we were discussing at the beginning of this section. To randomly generate $\overline{X}_{il} \in [0,30]$, $i = 1,2$ and $1 \leq l \leq 30$, we use our Sobol quasi-random number generator to obtain random vectors $v_l = (v_{l1}, \ldots, v_{l8}) \in [0,1]^8$, $1 \leq l \leq 30$. Order the first four components in v_l from smallest to largest to construct trapezoidal fuzzy number $\overline{X}_{1l}$ and then do the same for the next four components to get $\overline{X}_{2l}$ and compute

$$\overline{Z}_l = \overline{K}_1 \ \overline{X}_{1l} + \overline{K}_2 \ \overline{X}_{2l} + \overline{K}_3, \tag{13.8}$$

for $l = 1,2,...,30$. The $\overline{K}_1$, $\overline{K}_2$ and $\overline{K}_3$ in the above equation were given above. The data is $((\overline{X}_{1l}, \overline{X}_{2l}); \overline{Z}_l)$, $1 \leq l \leq 30$. All fuzzy arithmetic is to be done using $\alpha-$cuts and interval arithmetic with α going from zero to one in increments of 0.1. So the data is really $((\overline{X}_{1l}[\alpha], \overline{X}_{2l}[\alpha]); \overline{Z}_l[\alpha])$ for $l = 1,2,...,30$ and $\alpha = 0.0, 0.1, 0.2, ..., 0.9, 1.0$.

Now we are ready for our fuzzy Monte Carlo method. Randomly generate, using our Sobol quasi-random number generator, random vectors $v_k = (x_{k1}, ..., x_{k,12})$, $k = 1,2,...,N$, using the first four numbers (as discussed above) to define trapezoidal fuzzy number $\overline{C}_{1k}$, the next four numbers for $\overline{C}_{2k}$ and the last four numbers for $\overline{C}_{3k}$. Now compute

$$\overline{Y}_{lk} = \overline{C}_{1k}\overline{X}_{1l} + \overline{C}_{2k}\overline{X}_{2l} + \overline{C}_{3k}, \tag{13.9}$$

for $l = 1,2,...,30$, $k = 1,2,...,N$. Determine the error measure

$$Error = E_k = (1/30) \sum_{l=1}^{30} D^2(\overline{Y}_{lk}, \overline{Z}_l). \tag{13.10}$$

Find the values for the $\overline{C}_i$ to minimize the error measure.

Next we need to determine intervals $I_i = [a_i, b_i]$, $i = 1,2,3$, for these fuzzy numbers. A good selection of these intervals will make the fuzzy Monte Carlo process more efficient because: (1) if they are wrong and/or too small we can miss a "good" solution; and (2) if they are too big the simulation can spend too much time looking at situations that will not produce a "good" solution. Defining the intervals I_i is application dependent so we now proceed to the applications.

13.5 Applications

The first three applications are from [2] and the last application is a new example.

13.5.1 First Application

This application is to use our new Monte Carlo method to find the values of $\overline{K}_1$, $\overline{K}_2$ and $\overline{K}_3$ that produced the data $((\overline{X}_{1l}, \overline{X}_{2l}); \overline{Z}_l)$, $1 \leq l \leq 30$, which is generated as described above. We will need to find intervals I_i, $i = 1,2,3$, as explained above. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the intervals from [2]; and (2) secondly, in the second subsection, we use an optimization procedure to determine these intervals.

Monte Carlo programs were written in MATLAB [4]. For all our Monte Carlo calculations, we used a Pentium III, Processor: 933 MHz.

First Choice of Intervals

The selection of these intervals was discussed in detail in [2]. However, those intervals turned out to be much too large, having large negative components, and the fuzzy coefficients are known to be positive. So we take only the positive part of the intervals calculated in [2] to be: (1) $[0, 10.604]$ for $\overline{C}_1$; (2) $[0, 7.803]$ for $\overline{C}_2$; and (3) $[0, 47.846]$ for $\overline{C}_3$. Assume that we have already ordered the first four components in $v_k = (x_{k1}, ..., x_{k,12})$ from smallest to largest, same for the next four and the last four, then $\overline{C}_{1k} = 10.604(x_{k1}/x_{k2}, x_{k3}/x_{k4})$, $\overline{C}_{2k} = 7.803(x_{k5}/x_{k6}, x_{k7}/x_{k8})$ and $\overline{C}_{3k} = 47.846(x_{k9}/x_{k,10}, x_{k,11}/x_{k,12})$. Now find $\overline{Y}_{lk}$ as in equation (13.9) and E_k in equation (13.10).

Table 13.1. Results of the Monte Carlo (LMCI) Method, First Choice of Intervals, to Minimize the Error, First Application

Coefficient	*LMCI*
$\overline{C}_1$	$(2.989/3.181, 3.674/4.218)$
$\overline{C}_2$	$(2.021/2.979, 3.961/4.956)$
$\overline{C}_3$	$(4.451/4.893, 5.517/6.139)$

Using our Sobol quasi-random number generator we produced $N = 400,000$ vectors $v_k = (x_{k1}, ..., x_{k,12})$ which defined the $\overline{C}_i$ as described above. Results are shown in Table 13.1 with minimum error value in Table 13.5. Since we will have three Monte Carlo studies for the fuzzy linear function we call this one LMCI (for linear Monte Carlo).

Second Choice of Intervals

We need to determine the intervals $I_i = [a_i, b_i]$ for the supports of the $\overline{C}_i$, $i = 1, 2, 3$. We first describe an optimization method used to determine these intervals. This procedure will be called *LMCII*. Let

$$[L_l, R_l] = I_1[x_{11l}, x_{1l4}] + I_2[x_{2l1}, x_{2l4}] + I_3, \tag{13.11}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$, where $\overline{X}_{1l} = (x_{1l1}/x_{1l2}, x_{1l3}/x_{1l4})$ and $\overline{X}_{2l} = (x_{2l1}/x_{2l2}, x_{2l3}/x_{2l4})$. Recall the data $\overline{Z}_l \approx (z_{l1}/z_{l2}, z_{l3}/z_{l4})$. Define

$$W = \sum_{l=1}^{30}(L_l - z_{l1})^2 + \sum_{l=1}^{30}(R_l - z_{l4})^2. \tag{13.12}$$

Table 13.2. Optimization Method of Determining the Supports (QMCII) for Monte Carlo

Coefficient	Support
$\overline{C}_1$	$[3.000, 4.200]$
$\overline{C}_2$	$[2.000, 5.000]$
$\overline{C}_3$	$[4.500, 6.000]$

Table 13.3. Results of the Monte Carlo Method (LMCII), Second Choice of Intervals, to Minimize the Error, First Application

Coefficient	*LMCII*
$\overline{C}_1$	$(3.021/3.219, 3.699/4.193)$
$\overline{C}_2$	$(1.998/2.982, 3.988/4.989)$
$\overline{C}_3$	$(4.504/5.102, 5.521/5.996)$

Table 13.4. Results of the Monte Carlo Method (LMCIII), Second Choice of Intervals, to Minimize the Error, First Application

Coefficient	*LMCIII*
$\overline{C}_1$	$(3.019/3.196, 3.697/4.203)$
$\overline{C}_2$	$(2.003/2.999, 3.998/5.001)$
$\overline{C}_3$	$(4.453/5.032, 5.502/5.984)$

The optimization problem is to minimize W subject to $a_i \leq b_i$ all i. We want to find the intervals that make L_l and R_l closest, in the sense of minimizing W, to the end points of the supports of the $\overline{Z}_l$ in the data. We solved this using Maple [4]. The results, rounded to three decimal places, are in Table 13.2.

We now produce a sequence of random crisp vectors $v_k = (x_{k1}, ..., x_{k,12})$, $k = 1, 2, ..., N$, using our Sobol quasi-random number generator as described above, to get a sequence of trapezoidal fuzzy numbers $\overline{C}_{ik}$, $i = 1, 2, 3$, in their intervals and compute E_k in equation (13.10). After a run of $N = 400,000$ the smallest error value found is shown in Table 13.5 with corresponding $\overline{C}_i$ values shown in Table 13.3.

As a result of LMCI and LMCII we experimented with other choices for these intervals. In a Monte Carlo study it is rare to obtain the end points of an interval I_i to be the exact support of a $\overline{C}_i$, $i = 1, 2, 3$. For example, $I_1 = [3.000, 4.200]$ and we would not expect to get in our Monte Carlo study support$(\overline{C}_1) = [3.000, 4.200]$. So we slightly enlarged the intervals in Table 13.2 so that our Monte Carlo program could obtain the intervals in Table 13.2 for the support of $\overline{C}_i$, $i = 1, 2, 3$. The intervals we used were $I_1 = [2.900, 4.300]$, $I_2 = [1.900, 5.100]$ and $I_3 = [4.400, 6.100]$. We called this method LMCIII. After a run of $N = 400,000$ the smallest error is in Table 13.5 with the corresponding $\overline{C}_i$ in Table 13.4.

Table 13.5. Comparing Fuzzy Monte Carlo Solutions and EA Solution to the Fuzzy Linear Function Data

Method	Error(E)
EA	0.00217
LMCI	0.334181
LMCII	0.058135
LMCIII	0.002031

Comparison of Solutions

We see that the error measure for LMCI is much too large compared to the error in EA, LMCII and LMCIII. The error for EA was taken from [2]. This was because the intervals for the $\overline{C}_i$ were too large for a Monte Carlo run of $N = 400,000$. Possibly $N = 1,000,000$ could result in a much smaller error for LMCI. The error measure for LMCIII was comparable to that for the EA method. The computing time was between 14 and 16 minutes for $N = 100,000$.

13.5.2 Second Application

This application is to use our Monte Carlo method to find the values of $\overline{K}_i$, $1 \leq i \leq 6$, that produced the data $((\overline{X}_{1l}, \overline{X}_{2l}); \overline{Z}_l)$, $1 \leq l \leq 30$, which was generated by the fuzzy function

$$\overline{Z} = \overline{K}_1\overline{X}_1^2 + \overline{K}_2\overline{X}_1\overline{X}_2 + \overline{K}_3\overline{X}_2^2 + \overline{K}_4\overline{X}_1 + \overline{K}_5\overline{X}_2 + \overline{K}_6. \qquad (13.13)$$

The values used for the $\overline{K}_i$ are $\overline{K}_1 = (1.0/1.5, 2.5/3.0)$, $\overline{K}_2 = (3.2/3.4, 4.0/4.3)$, $\overline{K}_3 = (3.0/3.2, 3.7/4.2)$, $\overline{K}_4 = (2.4/2.5, 2.9/3.0)$, $\overline{K}_5 = (0.8/1.1, 1.5/2.1)$, and $\overline{K}_6 = (4.5/4.6, 5.1/5.4)$. The value for $\overline{K}_6$ was changed slightly from [2] because there was an error in reporting its value in [2]. We randomly, using our Sobol quasi-random number generator as explained above, generated 30 values of the $\overline{X}_i \in [0, 30]$, $i = 1, 2$, and then used equation (13.13) to compute the $\overline{Z}_l$ and obtain the data. In our fuzzy Monte Carlo method we will randomly generate fuzzy vectors $\overline{V}_k = (\overline{C}_{1k}, \dots, \overline{C}_{6k})$, $k = 1, \dots, N$, and compute for $l = 1, \dots, 30$;

$$\overline{Y}_{lk} = \overline{C}_{1k}\overline{X}_{1l}^2 + \overline{C}_{2k}\overline{X}_{1l}\overline{X}_{2l} + \overline{C}_{3k}\overline{X}_{2l}^2 + \overline{C}_{4k}\overline{X}_{1l} + \overline{C}_{5k}\overline{X}_{2l} + \overline{C}_{6k}. \qquad (13.14)$$

Then evaluate the error measure in equation (13.10) and search for the vector $\overline{V}_k$ that minimizes E_k.

We will need to find intervals I_i, as explained above, for the $\overline{C}_i$, $i = 1, \dots, 6$. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the intervals from [2]; and (2) secondly, in the second subsection, we use an optimization procedure to determine these intervals.

First Choice of Intervals

The selection of these intervals was discussed in detail in [2]. However, those intervals turned out to be much too large, having large negative components, and the fuzzy coefficients are known to be positive. So we take only the positive part of the intervals calculated in [2] to be: (1) $[0, 5.369]$ for $\overline{C}_1$; (2) $[0, 11.129]$ for $\overline{C}_2$; (3) $[0, 10.710]$ for $\overline{C}_3$; (4) $[0, 10.371]$ for $\overline{C}_4$; (5) $[0, 13.954]$ for $\overline{C}_5$; and (6) $[0, 13.954]$ for $\overline{C}_6$. Assume that we have already ordered the first four components in $v_k = (x_{k1}, ..., x_{k,24})$ from smallest to largest, same for the next four, . . . and the last four, then $\overline{C}_{1k} = 5.369(x_{k1}/x_{k2}, x_{k3}/x_{k4})$, $\overline{C}_{2k} = 11.129(x_{k5}/x_{k6}, x_{k7}/x_{k8})$, $\overline{C}_{3k} = 10.710(x_{k9}/x_{k,10}, x_{k,11}/x_{k,12})$, $\overline{C}_{4k} = 10.371(x_{k,13}/x_{k,14}, x_{k,15}/x_{k,16})$, $\overline{C}_{5k} = 13.954(x_{k,17}/x_{k,18}, x_{k,19}/x_{k,20})$, and $\overline{C}_{6k} = 13.954(x_{k,21}/x_{k,22}, x_{k,23}/x_{k,24})$. Now find the $\overline{Y}_{lk}$ for equation (13.14) and E_k from equation (13.10).

Using our Sobol quasi-random number generator we produced N vectors $v_k = (x_{k1}, ..., x_{k,24})$ which defined the $\overline{C}_i$ as described above. We were unable to drive the error measure sufficiently small for $N = 600,000$. We would probably need $N = 2,000,000$ or $N = 3,000,000$. The reason was that the intervals for the $\overline{C}_i$ were too large. Also, if a $\overline{C}_i$ was just slightly different from a $\overline{K}_i$, $i = 1, 2, 3$, the error could be large because of the squared (product) terms with a $\overline{X}_i \approx (20/25, 26/30)$, $i = 1, 2$.

Second Choice of Intervals

We need to determine the intervals $I_i = [a_i, b_i]$ for the supports of the $\overline{C}_i$, $i = 1, \ldots, 6$. We first describe an optimization method used to determine these intervals. Then we find intervals for the cores of the $\overline{C}_i$, $1 \le i \le 6$. Let $Q_{1l} = [x_{1l1}, x_{1l4}]$ and $Q_{2l} = [x_{2l1}, x_{2l4}]$. Define

$$[L_l, R_l] = I_1 Q_{1l}^2 + I_2 Q_{1l} Q_{2l} + I_3 Q_{2l}^2 + I_4 Q_{1l} + I_5 Q_{2l} + I_6, \tag{13.15}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$. Recall the data $\overline{Z}_l \approx (z_{l1}/z_{l2}, z_{l3}/z_{l4})$. Define

$$W = \sum_{l=1}^{30} (L_l - z_{l1})^2 + \sum_{l=1}^{30} (R_l - z_{l4})^2. \tag{13.16}$$

The optimization problem is to minimize W subject to $a_i \le b_i$ all i. We want to find the intervals that make L_l and R_l closest, in the sense of minimizing W, to the end points of the supports of the $\overline{Z}_l$ in the data. We solved this using Maple [4]. The results, rounded to three decimal places, are in Table 13.6.

Now we do the same for the cores of the fuzzy parameters. Let $V_{1l} = [x_{1l2}, x_{1l3}]$ and $V_{2l} = [x_{2l2}, x_{2l3}]$. Define

$$[L_{1l}, R_{1l}] = I_1 V_{1l}^2 + I_2 V_{1l} V_{2l} + I_3 V_{2l}^2 + I_4 V_{1l} + I_5 V_{2l} + I_6, \tag{13.17}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$. Define

$$W = \sum_{l=1}^{30} (L_{1l} - z_{l2})^2 + \sum_{l=1}^{30} (R_{1l} - z_{l3})^2. \tag{13.18}$$

The optimization problem is to minimize W subject to $a_i \le b_i$ all i. We want to find the intervals that make L_{1l} and R_{1l} closest, in the sense of minimizing W,

Table 13.6. Determining the Intervals for the Support and Core for Monte Carlo, Second Application

Coefficient	Support	Core
$\overline{C}_1$	$[1.000, 3.000]$	$[1.500, 2.500]$
$\overline{C}_2$	$[3.200, 4.300]$	$[3.400, 4.000]$
$\overline{C}_3$	$[3.000, 4.200]$	$[3.200, 3.700]$
$\overline{C}_4$	$[2.400, 3.000]$	$[2.500, 2.900]$
$\overline{C}_5$	$[0.800, 2.100]$	$[1.100, 1.500]$
$\overline{C}_6$	$[4.500, 5.400]$	$[4.600, 5.100]$

to the end points of the cores of the $\overline{Z}_l$ in the data. We solved this using Maple [4]. The results, rounded to three decimal places, are also in Table 13.6.

Now we tried the intervals in Table 13.6 for the $\overline{C}_i$, for example $\overline{C}_1 = (1.000/1.500, 2.500/3.000), \ldots, \overline{C}_6 = (4.500/4.600, 5.100/5.400)$, and got zero error. Problem finished. This analytical method found the unknown fuzzy parameters $\overline{K}_i$, $1 \leq i \leq 6$, and no EA or Monte Carlo procedure is required. However, for this analytical procedure to work we needed to know the structure of the fuzzy function that produced the fuzzy data.

13.5.3 Third Application

This application is to use our Monte Carlo method to find the values of $\overline{K}_i$, $1 \leq i \leq 10$, that produced the data $((\overline{X}_{1l}, \overline{X}_{2l}); \overline{Z}_l)$, $1 \leq l \leq 30$, which was generated by the fuzzy cubic polynomial

$$\begin{aligned}\overline{Z} = \overline{K}_1\overline{X}_1^3 + \overline{K}_2\overline{X}_1^2\overline{X}_2 + \overline{K}_3\overline{X}_1\overline{X}_2^2 +\\ \overline{K}_4\overline{X}_2^3 + \overline{K}_5\overline{X}_1^2 + \overline{K}_6\overline{X}_1\overline{X}_2 +\\ \overline{K}_7\overline{X}_2^2 + \overline{K}_8\overline{X}_1 + \overline{K}_9\overline{X}_2 + \overline{K}_{10}.\end{aligned} \tag{13.19}$$

The values used for the $\overline{K}_i$ are $\overline{K}_1 = (2.8/3.5, 4.1/4.8)$, $\overline{K}_2 = (0.8/1.1, 1.5/2.1)$, $\overline{K}_3 = (2.0/2.5, 2.9/4.0)$, $\overline{K}_4 = (3.0/3.2, 3.7/4.0)$, $\overline{K}_5 = (3.2/4.1, 5.1/6.0)$, $\overline{K}_6 = (1.0/1.3, 1.7/2.3)$, $\overline{K}_7 = (2.6/3.1, 3.7/4.3)$, $\overline{K}_8 = (3.0/3.2, 3.7/4.2)$, $\overline{K}_9 = (1.2/1.4, 2.0/2.6)$, and $\overline{K}_{10} = (0.4/0.6, 1.2/1.5)$.

We randomly, using our Sobol quasi-random number generator as explained above, generate 30 values of the $\overline{X}_i \in [0, 30]$, $i = 1, 2$, and then use equation (13.19) to compute the $\overline{Z}_l$ and obtain the data. In our fuzzy Monte Carlo method we will randomly generate fuzzy vectors $\overline{V}_k = (\overline{C}_{1k}, \ldots, \overline{C}_{10,k})$, $k = 1, \ldots, N$, and compute for $l = 1, \ldots, 30$

$$\begin{aligned}\overline{Y}_{lk} = \overline{C}_{1k}\overline{X}_{1l}^3 + \overline{C}_{2k}\overline{X}_{1l}^2\overline{X}_{2l} + \overline{C}_{3k}\overline{X}_{1l}\overline{X}_{2l}^2 +\\ \overline{C}_{4k}\overline{X}_{2l}^3 + \overline{C}_{5k}\overline{X}_{1l}^2 + \overline{C}_{6k}\overline{X}_{1l}\overline{X}_{2l} +\\ \overline{C}_{7k}\overline{X}_{2l}^2 + \overline{C}_{8k}\overline{X}_{1l} + \overline{C}_{9k}\overline{X}_{2l} + \overline{C}_{10,k}.\end{aligned} \tag{13.20}$$

Then evaluate the error measure in equation (13.10) and search for the vector $\overline{V}_k$ that minimizes E_k.

We will need to find intervals I_i, as explained above, for the $\overline{C}_i$, $i = 1, \ldots, 10$. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the intervals from [2]; and (2) secondly, in the second subsection, we use an optimization procedure to determine these intervals.

First Choice of Intervals

The selection of these intervals was discussed in detail in [2]. However, those intervals turned out to be much too large, some having large negative components, and the fuzzy coefficients are known to be positive. So we take only the positive part of the intervals calculated in [2] to be: (1) $[0.021, 5.249]$ for $\overline{C}_1$; (2) $[0, 4.791]$ for $\overline{C}_2$; (3) $[0, 6.064]$ for $\overline{C}_3$; (4) $[1.002, 5.194]$ for $\overline{C}_4$; (5) $[0, 11.209]$ for $\overline{C}_5$; (6) $[0, 7.481]$ for $\overline{C}_6$; (7) $[0, 10.201]$ for $\overline{C}_7$; (8) $[0, 12.955]$ for $\overline{C}_8$; (9) $[0, 10.445]$ for $\overline{C}_9$; and (10) $[0, 286.450]$ for $\overline{C}_{10}$. Assume that we have already ordered the first four components in $v_k = (x_{k1}, ..., x_{k,40})$ from smallest to largest, same for the next four, ... and the last four, then $\overline{C}_{1k} = 0.021 + 5.228(x_{k1}/x_{k2}, x_{k3}/x_{k4})$, $\overline{C}_{2k} = 4.791(x_{k5}/x_{k6}, x_{k7}/x_{k8})$, $\overline{C}_{3k} = 6.064(x_{k9}/x_{k,10}, x_{k,11}/x_{k,12})$, $\overline{C}_{4k} = 1.002 + 4.192(x_{k,13}/x_{k,14}, x_{k,15}/x_{k,16})$, $\overline{C}_{5k} = 11.209(x_{k,17}/x_{k,18}, x_{k,19}/x_{k,20})$, $\overline{C}_{6k} = 7.481(x_{k,21}/x_{k,22}, x_{k,23}/x_{k,24})$, $\overline{C}_{7k} = 10.201(x_{k,25}/x_{k,26}, x_{k,27}/x_{k,28})$, $\overline{C}_{8k} = 12.955(x_{k,29}/x_{k,30}, x_{k,31}/x_{k,32})$, $\overline{C}_{9k} = 10.445(x_{k,33}/x_{k,34}, x_{k,35}/x_{k,36})$, and $\overline{C}_{10,k} = 286.450(x_{k,37}/x_{k,38}, x_{k,39}/x_{k,40})$. Now find the $\overline{Y}_{lk}$ for equation (13.20) and E_k from equation (13.10).

Using our Sobol quasi-random number generator we produced $N = 200,000$ vectors $v_k = (x_{k1}, ..., x_{k,24})$ which defined the $\overline{C}_i$ as described above. We were unable to drive the error measure sufficiently close to zero even with $N = 600,000$ for the same reasons discussed in the previous application.

Second Choice of Intervals

We need to determine the intervals $I_i = [a_i, b_i]$ for the supports of the $\overline{C}_i$, $i = 1, \ldots, 10$. We first describe an optimization method used to determine these intervals. Then we do the same for the cores. Let $Q_{1l} = [x_{1l1}, x_{1l4}]$ and $Q_{2l} = [x_{2l1}, x_{2l4}]$. Define

$$\begin{aligned}[L_l, R_l] = I_1 Q_{1l}^3 + I_2 Q_{1l}^2 Q_{2l} + I_3 Q_{1l} Q_{2l}^2 + I_4 Q_{2l}^3 + I_5 Q_{1l}^2 + \\ I_6 Q_{1l} Q_{2l} + I_7 Q_{2l}^2 + I_8 Q_{1l} + I_9 Q_{2l} + I_{10},\end{aligned} \tag{13.21}$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$. Recall the data $\overline{Z}_l \approx (z_{l1}/z_{l2}, z_{l3}/z_{l4})$. Define

$$W = \sum_{l=1}^{30} (L_l - z_{l1})^2 + \sum_{l=1}^{30} (R_l - z_{l4})^2. \tag{13.22}$$

The optimization problem is to minimize W subject to $a_i \leq b_i$ all i. We want to find the intervals that make L_l and R_l closest, in the sense of minimizing W, to the end points of the supports of the $\overline{Z}_l$ in the data. We solved this using Maple

Table 13.7. Determining the Intervals for the Support and Core for Monte Carlo, Third Application

Coefficient	Support	Core
$\overline{C}_1$	$[2.800, 4.800]$	$[3.500, 4.100]$
$\overline{C}_2$	$[0.800, 2.100]$	$[1.100, 1.500]$
$\overline{C}_3$	$[2.000, 4.000]$	$[2.500, 2.900]$
$\overline{C}_4$	$[3.000, 4.000]$	$[3.200, 3.700]$
$\overline{C}_5$	$[3.200, 6.000]$	$[4.100, 5.100]$
$\overline{C}_6$	$[1.000, 2.300]$	$[1.300, 1.700]$
$\overline{C}_7$	$[2.600, 4.300]$	$[3.100, 3.700]$
$\overline{C}_8$	$[3.000, 4.200]$	$[3.200, 3.700]$
$\overline{C}_9$	$[1.200, 2.600]$	$[1.400, 2.000]$
$\overline{C}_{10}$	$[0.400, 1.501]$	$[0.600, 1.200]$

[4]. This time we gave the program initial values for the variables $a_i = 1.0$ and $b_i = 2.0$ all i. The results, rounded to three decimal places, are in Table 13.7.

Let $V_{1l} = [x_{1l2}, x_{1l3}]$ and $V_{2l} = [x_{2l2}, x_{2l3}]$. Define

$$[L_{1l}, R_{1l}] = I_1 V_{1l}^3 + I_2 V_{1l}^2 V_{2l} + I_3 V_{1l} V_{2l}^2 + I_4 V_{2l}^3 + I_5 V_{1l}^2 + I_6 V_{1l} V_{2l} + I_7 V_{2l}^2 + I_8 V_{1l} + I_9 V_{2l} + I_{10}, \qquad (13.23)$$

evaluated using interval arithmetic, for $l = 1, 2, ..., 30$. Define

$$W = \sum_{l=1}^{30} (L_{1l} - z_{l2})^2 + \sum_{l=1}^{30} (R_{1l} - z_{l3})^2. \qquad (13.24)$$

The optimization problem is to minimize W subject to $a_i \leq b_i$ all i. We want to find the intervals that make L_{1l} and R_{1l} closest, in the sense of minimizing W, to the end points of the cores of the $\overline{Z}_l$ in the data. We solved this using Maple [4]. We gave the program initial values for the variables $a_i = 2.0$ and $b_i = 3.0$ all i. The results, rounded to three decimal places, are in Table 13.7.

Now we tried the intervals in Table 13.7 for the $\overline{C}_i$, for example $\overline{C}_1 = (2.800/3.500, 4.100/4.800), \ldots, \overline{C}_{10} = (0.400/0.600, 1.200/1.501)$, and got an error of approximately zero. Problem finished. This analytical method found the unknown fuzzy parameters $\overline{K}_i$, $1 \leq i \leq 10$, and no EA or Monte Carlo procedure is required. However, for this analytical procedure to work we needed to know the structure of the fuzzy function that produced the fuzzy data which will not happen in the next application.

13.5.4 Fourth Application

In this example we act like we do not know the structure of the function f that generates the fuzzy data and it does not belong to our library $\mathcal{L}$. That is, f is not a fuzzy multivariate polynomial. Let $z = f(x_1, x_2) = \exp(x_1 + x_2)$ for

$x_i \in I = [-a, a]$, $a > 0$, $i = 1, 2$. By the extension principle $\overline{Z} = f(\overline{X}_1, \overline{X}_2)$ for $\overline{X}_i$ trapezoidal fuzzy numbers in I, $i = 1, 2$. The data will be $((\overline{X}_{1l}, \overline{X}_{2l}), \overline{Z}_l)$, $1 \leq l \leq 30$, generated by f. We use our Sobol quasi-random number generator to obtain $\overline{X}_{1l}$ and $\overline{X}_{2l}$ in I and then $\overline{Z}_l = f(\overline{X}_{1l}, \overline{X}_{2l})$, $l = 1, \ldots, 30$.

We will now use the results in Section 13.2. We could write the function f as $z = k_1 \exp(k_2 x_1 + k_3 x_2 + k_4) + k_5$ for constants $k_1 = k_2 = k_3 = 1$ and $k_4 = k_5 = 0$. These constants will be fixed at their crisp values given above and not fuzzified when we go to $\overline{Z} = f(\overline{X}_1, \overline{X}_2)$ which means we do not have any $\overline{K}_j$ used in Section 13.2. Let $g(x_1, x_2)$ be any function having continuous partial derivatives of all orders on $I \times I$. Then there are constants c_i, $i = 1, \cdots, 10$ so that the Taylor series expansion of g about $(0, 0)$ is

$$g(x_1, x_2) = c_1 + (c_2 x_1 + c_3 x_2) + (c_4 x_1^2 + c_5 x_1 x_2 + c_6 x_2^2) + (c_7 x_1^3 + c_8 x_1^2 x_2 + c_9 x_1 x_2^2 + c_{10} x_2^3) + \mathbf{R}, \tag{13.25}$$

where $\mathbf{R}$ is the remainder and the c_i depend on the partial derivatives of g evaluated at $(0, 0)$ [5]. Let $p(x_1, x_2; c_1, \ldots, c_{10})$ denote the cubic polynomial in x_1 and x_2 in the above equation. We will assume that

$$|g(x_1, x_2) - p(x_1, x_2; c_1, \ldots, c_{10})| < \epsilon, \tag{13.26}$$

for some ϵ for all $x_i \in I$, $i = 1, 2$. The ϵ in the above equation will depend on g and the interval I. Using the extension principle get $\overline{Y} = p(\overline{X}_1, \overline{X}_2; c_1, \ldots, c_{10})$ for some constants c_i, $1 \leq i \leq 10$.

Now f is such a function as g so equations (13.25) and (13.26) will hold for the (unknown) f. So there are constants c_i so that

$$|f(x_1, x_2) - p(x_1, x_2; c_1, \ldots, c_{10})| < \epsilon, \tag{13.27}$$

for some ϵ and all $x_i \in I$. This equation is similar to equation (13.2). Then by a proof similar to that in [2] we may attain

$$D(f(\overline{X}_1, \overline{X}_2), p(\overline{X}_1, \overline{X}_2; c_1, \ldots, c_{10})) < \epsilon, \tag{13.28}$$

for all $\overline{X}_i \in I$. We point out that the fuzzy functions in equation (13.28) are both evaluated using the extension principle.

If we knew f we could find the c_i, $i = 1, \ldots, 10$, choose an interval I and estimate/compute ϵ. We will do this at the end of the section but for now we continue with unknown f. Let us pick an interval I and apply our Monte Carlo method to estimate the c_i, $i = 1, \ldots, 10$, and find the minimum error.

Monte Carlo Method

Let $I = [-0.6, 0.6]$. Using this interval we may generate the data. We will need to find intervals $I_i = [a_i, b_i]$ for the c_i, $1 \leq i \leq 10$. Given these intervals we use our Sobol quasi-random number generator to produce vectors $v_k = (c_{1k}, \ldots, c_{10,k})$, $c_{ik} \in I_i$, $1 \leq i \leq 10$ and $k = 1, \ldots, N$. Our first choice is $I_i = [0, 1]$ all i. Compute

$$\overline{Y}_{lk} = p(\overline{X}_{1l}, \overline{X}_{2l}; c_{1k}, \ldots, c_{10,k}), \tag{13.29}$$

Table 13.8. Results of the Monte Carlo (MC) Method, to Minimize the Error, Fourth Application

Coefficient	Value
c_1	0.9577
c_2	0.8812
c_3	0.8737
c_4	0.6337
c_5	0.4918
c_6	0.0006
c_7	0.0011
c_8	0.0004
c_9	0.0010
c_{10}	0.0003

for $l = 1, \ldots, 30$, $k = 1, \ldots, N$. We evaluate equation (13.29) using α-cuts and interval arithmetic. We used $\alpha-$cuts and interval arithmetic to evaluate the fuzzy polynomial because we found it difficult to evaluate using the extension principle in a MATLAB program. Determining $\overline{Y}_{lk}$ is now more difficult because the intervals given by $\alpha-$cuts of $\overline{X}_{1l}$ and $\overline{X}_{2l}$ can have negative end points. This only complicates the multiplication of intervals and we use the general formula $[a, b] * [c, d] = [s, t]$ where $s(t) = min(max)\{ac, ad, bc, bd\}$. Next we compute E_k in equation (13.10) and find the minimum error value.

After a Monte Carlo run of $N = 100,000$ the results for the c_i are shown in Table 13.8 with minimum error in Table 13.10. We call this method MC (for Monte Carlo).

Optimization Method

Now we find the values for the c_i, $i = 1, \ldots, 10$, using an optimization procedure similar to the one used in the previous two sections. Let $\overline{X}_{1l} = (x_{1l1}/x_{1l2,}, x_{1l3}/x_{1l4})$, $\overline{X}_{2l} = (x_{2l1}/x_{2l2,}, x_{2l3}/x_{2l4})$, $Q_{1l} = [x_{1l1}, x_{1l4}]$, $V_{1l} = [x_{1l2}, x_{1l3}]$, $V_{2l} = [x_{2l2}, x_{2l3}]$ and $Q_{2l} = [x_{2l1}, x_{2l4}]$. Define

$$[L_l, R_l] = p(Q_{1l}, Q_{2l}; c_1, \ldots, c_{10}), \tag{13.30}$$

and

$$[L_{1l}, R_{1l}] = p(V_{1l}, V_{2l}; c_1, \cdots, c_{10}), \tag{13.31}$$

both evaluated using interval arithmetic, $l = 1, \ldots, 30$. Define W_1 as in equation (13.12) and then define

$$W_2 = \sum_{l=1}^{30} (L_{1l} - z_{l2})^2 + \sum_{l=1}^{30} (R_{1l} - z_{l3})^2, \tag{13.32}$$

and set $W = W_1 + W_2$. The optimization problem is to minimize W subject to $0 \leq c_i \leq 1$, $1 \leq i \leq 10$. The objective is to find the constants to make the support

Table 13.9. Determining the Constants, Optimization Method, Fourth Application

Coefficient	Value
c_1	1.0000
c_2	0.7731
c_3	0.8736
c_4	0.4726
c_5	0.0000
c_6	0.8130
c_7	0.0000
c_8	0.0000
c_9	0.0000
c_{10}	0.0000

Table 13.10. Comparing Solutions in the Fourth Application

Method	Error(E)
MC	0.0084
OPT	0.0554
TAY	0.0493

and core of the $\overline{Y}_{lk}$ close to the support and core of the data $\overline{Z}_l$, $1 \leq l \leq 30$. We solved this problem using Maple. The results are in Table 13.9.

We now computed $\overline{Y}_{lk} = p(\overline{X}_{1l}, \overline{X}_{2l}; c_{1k}, \ldots, c_{10,k})$ using the c_i values in Table 13.9 and α-cuts with interval arithmetic. We will call this method OPT (optimization method) and the error measure for this procedure is shown in Table 13.10.

Now let us assume we know f. Then $c_1 = c_2 = c_3 = c_5 = 1$, $c_4 = c_6 = c_8 = c_9 = 0.5$ and $c_7 = c_{10} = 1/6$. Using these values for the c_i we evaluated equation (13.29) using α-cuts with interval arithmetic and computed the error measure in equation (13.10). We call this procedure TAY (for Taylor series) and the error value is in Table 13.10. We may also find ϵ in equation (13.27). We used Maple to find the maximum of $|f(x_1, x_2) - p(x_1, x_2; c_1, \cdots, c_{10})|$, using the Taylor series values for the c_i given above, for $x_1, x_2 \in [-0.6, 0.6]$. The maximum was $\epsilon = 0.1121$. If we use the extension principle to evaluate the fuzzy functions in equation (13.28), then that equation holds for $\epsilon = 0.1121$. Hence, from equation (13.10)

$$Error \leq 30(\epsilon)^2/30 = 0.012. \tag{13.33}$$

Comparison of Solutions

We see that our Monte Carlo method (MC) obtained the smallest error. The computer time for the fuzzy Monte Carlo was 7 to 8 hours for a run of $N = 100,000$.

The optimization method (OPT), that worked so well in the second and third application, did not get the smallest error. In this application we did not know the structure of the fuzzy function that produced the fuzzy data and this may be the reason this method did not perform that well.

Now suppose we know the fuzzy function that produced the fuzzy data and we compute the first four terms in the Taylor series, equation (13.25), with the c_i values previously given. We then use this fuzzy polynomial to get the $\overline{Y}_{lk}$ as in equation (13.29) using α-cuts and interval arithmetic to evaluate the fuzzy function. This method (TAY) had error 0.0493. But if we used the extension principle to evaluate the fuzzy polynomial, equation (13.3) holds, and the error will be less than 0.012. Since, in practice, we will not know the fuzzy function that produced the fuzzy data, these results were included only for comparison.

Our Monte Carlo method was clearly the best way to model the fuzzy data.

13.6 Summary and Conclusions

In this chapter we introduced our new fuzzy Monte Carlo procedure for multivariate fuzzy nonlinear regression. The basic requirement of any fuzzy Monte Carlo method is to be able to randomly produce fuzzy/crisp vectors to uniformly fill the search space. We suggested using a quasi-random number generator to make these random fuzzy/crisp vectors. Theoretically, given enough iterations of the fuzzy Monte Carlo technique, it will produce a sufficiently small error. A major problem with our fuzzy Monte Carlo method is to find a sufficiently small (to reduce computer time) search space.

We applied our fuzzy Monte Carlo method in four applications. The first three were taken from [2] so that we could compare the fuzzy Monte Carlo procedure to the evolutionary algorithm (EA) that was used in that paper. The first application was a fuzzy linear function with two variables that produced the fuzzy data. The objective was to find the fuzzy constants in the fuzzy function and our fuzzy Monte Carlo method did as well as the EA.

The second two applications involved fuzzy polynomials and we discovered an optimization procedure to find the "unknown" fuzzy constants in the fuzzy polynomials and an EA or fuzzy Monte Carlo method was not needed. The optimization procedure was originally designed to find a "good" search space. This optimization method required knowing the structure of the fuzzy function that produced the fuzzy data, usually not the case in applications.

The fourth application was about a "unknown" fuzzy function producing the fuzzy data and this fuzzy function was not a fuzzy linear/polynomial fuzzy function. Our universal approximation result in Section 13.2 says that there is a fuzzy polynomial that can model this data, but all fuzzy functions must use the extension principle for evaluation. We showed that a cubic polynomial, with crisp coefficients, can be used and our Monte Carlo method found these coefficients that will give small error. However, we employed $\alpha-$cuts and interval arithmetic to evaluate the fuzzy polynomial. We did not find the unknown fuzzy function that made the fuzzy data; we found a polynomial that can model the data. The optimization procedure that worked so well in the second and third application did not perform well in this application since we could not use the structure of the unknown fuzzy function.

All the fuzzy data used and all the MATLAB programs are available from the authors.

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Multivariate Regression (under review)
2. Buckley, J.J., Feuring, T., Hayashi, Y.: Multivariate Nonlinear Fuzzy Regression: An Evolutionary Algorithm Approach. Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems 7, 83–98 (1999)
3. Maple 9.5, Waterloo Maple Inc., Waterloo, Canada
4. MATLAB, The MathWorks, `http://www.mathworks.com`
5. Taylor, A.E.: General Theory of Functions and Integration. Blaisdell, Waltham, Mass. (1965)

14 Fuzzy Linear Regression II

14.1 Introduction

This chapter is a continuation of Chapter 11 and is based on [1]. We wish to use our Monte Carlo method to get approximate solutions for crisp numbers a_i, $0 \leq i \leq m$, to the fuzzy linear regression model

$$\overline{Y} = a_0 + a_1\overline{X}_1 + ... + a_m\overline{X}_m, \tag{14.1}$$

for $\overline{X}_i$, $1 \leq i \leq m$, triangular fuzzy numbers and $\overline{Y}$ a triangular fuzzy number. The fuzzy linear regression model in equation (14.1) has been previously studied in [2]-[6]. In this model the independent variables $\overline{X}_i$ will be given triangular fuzzy numbers, the dependent variable $\overline{Y}$ will be a given triangular fuzzy number, so the best way to fit the model to the data is to use real numbers for the a_i. If a_i is also a triangular fuzzy number, then $a_i\overline{X}_i$ will be a triangular shaped fuzzy number and the right side of equation (14.1) is a triangular shaped fuzzy number which is used to approximate $\overline{Y}$ a triangular fuzzy number. If the a_i are real numbers the right side of equation (14.1) will be a triangular fuzzy number which is better to use to approximate a triangular fuzzy number $\overline{Y}$. The data will be $((\overline{X}_{1l}, ..., \overline{X}_{ml}), \overline{Y}_l)$, $1 \leq l \leq n$, for the $\overline{X}_{il} = (x_{il1}/x_{il2}/x_{il3})$ triangular fuzzy numbers and $\overline{Y}_l = (y_{l1}/y_{l2}/y_{l3})$ triangular fuzzy numbers. Given the data the objective is to find the "best" a_j, $0 \leq j \leq m$. We propose to employ our Monte Carlo methods to approximate the "best" values for the a_j, $j = 0, 1, ..., m$.

In Monte Carlo we randomly generate a possible solution, evaluate how "good" it is, discard inferior solutions, and continue N times. N is usually large like $10,000$ or $100,000$. In the next section we discuss how to randomly produce vectors $v_k = (a_{0k}, ..., a_{mk})$, $k = 1, 2, 3, ..., N$. Using the v_k we determine the predicted values

$$\overline{Y}^*_{lk} = a_{0k} + a_{1k}\overline{X}_{1l} + ... + a_{mk}\overline{X}_{ml}, \tag{14.2}$$

for $k = 1, 2, 3, .., N$ and $l = 1, 2, ..., m$. To see how good this v_k is we find the error between the given values $\overline{Y}_l$ and the predicted values $\overline{Y}^*_{lk}$. We will have two error measures in this chapter. The first error measure is

$$E_{1k}(E_1) = \sum_{l=1}^{n} [\int_{-\infty}^{\infty} |\overline{Y}_l(x) - \overline{Y}^*_{lk}(x)|dx]/[\int_{-\infty}^{\infty} \overline{Y}_l(x)dx], \tag{14.3}$$

where the integrals are really only over interval(s) containing the support of the fuzzy numbers. E_1 is a "vertical" error measure. Let $\overline{Y}_l = (y_{l1}/y_{l2}/y_{l3})$ and $\overline{Y}^*_{lk} = (y_{lk1}/y_{lk2}/y_{lk3})$ all triangular fuzzy numbers. Then our second error measure is

$$E_{2k}(E_2) = \sum_{l=1}^{n} [|y_{l1} - y_{lk1}| + |y_{l2} - y_{lk2}| + |y_{l3} - y_{lk3}|]. \tag{14.4}$$

E_2 is a "horizontal" error measure. So we calculate v_k, E_{1k} and E_{2k} for $k = 1, 2, ..., N$. A "best" solution is a value of v_k that minimizes E_{1k} (E_{2k}) for all k. An approximate "best" solution is a $v \in \{v_1, ..., v_N\}$ that minimizes an error measure. So we can have two approximate "best" solutions one with respect to E_1 and an other for E_2. Next we see how we will produce sequences of crisp random vectors v_k, $k = 1, 2, 3, .., N$, and then apply our Monte Carlo method to an example problem. In this chapter we will be generating sequences of crisp random vectors and not sequences of fuzzy random vectors. But first let us discuss the choice of error measures in more detail.

14.2 Error Measures

Consider the model in equation (14.1) and the given data and let $\mathcal{P}$ be some procedure for estimating the parameters a_i, $0 \leq i \leq m$. For example: (1) in [2] they used least absolute deviation; (2) in [4] they used least squares; and (3) in [5] they used a two stage method the first was least squares and the second another minimization procedure. For any $\mathcal{P}$ let the estimates be $\widehat{a}_i$, $0 \leq i \leq m$. The predicted values will then be

$$\widehat{\overline{Y}}_l = \widehat{a}_0 + \widehat{a}_1 \overline{X}_{1l} + ... + \widehat{a}_m \overline{X}_{ml}, \tag{14.5}$$

for $l = 1, ..., n$. Then we compute the value of some error measure $E = E(\overline{Y}, \widehat{\overline{Y}})$ which aggregates the "error" between $\overline{Y}_l$ and $\widehat{\overline{Y}}_l$ all l. The objective is to use a $\mathcal{P}$ that will make E small.

Now E should not be directly related to $\mathcal{P}$. Suppose $\mathcal{P}_1$ is least squares and $\mathcal{P}_2$ is least absolute deviation and E is also least squares. Then it is not fair to compare $\mathcal{P}_1$ and $\mathcal{P}_2$ using E. Actually it does not matter what $\mathcal{P}$ we use (discussed below). So, if we are comparing $\mathcal{P}_i$, $1 \leq i \leq I$, we may compare using E as long as E is not directly related to any $\mathcal{P}_i$.

Now consider E_1 and E_2 defined above. We will have four procedures $\mathcal{P}_1$ goes with [2], $\mathcal{P}_2$ with [4], $\mathcal{P}_3$ for [5] and $\mathcal{P}_4$ will be associated with our fuzzy Monte Carlo. But there are two relationships: (1) E_2 and $\mathcal{P}_1$ from [2]; and (2) E_1 with the second stage in $\mathcal{P}_3$. There is a strong relationship between $\mathcal{P}_1$ and E_2, discussed further in section 14.3.3, and we do not expect $\mathcal{P}_4$ to do better than $\mathcal{P}_1$ for E_2. The relationship between $\mathcal{P}_3$ and E_1 is weak, only the second stage in [5], and should not effect the results.

Why does it not matter what $\mathcal{P}$ we use? We know that in crisp linear regression, under certain normality assumptions, such as when $\mathcal{P}$ is least squares, we have a statistical theory developed for confidence intervals for the a_i and Y, and hypothesis tests for the a_i. But this does not yet exist for fuzzy linear regression. So, until such a theory exists for fuzzy linear regression and some $\mathcal{P}$, it does not matter what procedure you use. Given the model and the data fuzzy Monte Carlo can estimate the absolute minimum of any error measure. Even when we have a statistical theory for fuzzy linear regression it would be nice to have an estimate of the minimum of your error measure.

14.3 Random Vectors

To obtain random sequences $v_k = (a_{0k}, ..., a_{mk})$, where the a_{ik} are all real numbers in intervals I_i, $i = 0, ..., m$, we first randomly generate crisp vectors $v_k = (x_{1k}, ..., x_{mk})$ with all the x_{ik} in $[0, 1]$, $k = 1, 2, .., N$. Suppose a_i is to be in interval $I_i = [c_i, d_i]$, $i = 0, 1, 2, ..., m$. These intervals are very important to the Monte Carlo process because: (1) if they are wrong and/or too small we can miss a "good" solution; and (2) if they are too big the simulation can spend too much time looking at situations that will not produce a "good" solution. Since each x_{ik} starts out in $[0, 1]$ we may easily put them into $I_i = [c_i, d_i]$ by $a_{ik} = c_i + (d_i - c_i)x_{ik}$, $i = 0, 1, ..., m$.

How shall we make the random crisp vectors v_k? Now v_k is initially in $[0, 1]^{m+1}$. We will use our Sobol quasi-random number generator discussed in Chapter 3 to make the initial v_k and then transform to intervals I_i, $0 \leq i \leq m$. Our Sobol quasi-random number generator will then produce the transformed v_k uniformly filling the search space $I_0 \times ... \times I_m$.

14.4 Example Problem

The data for this application was taken from [2] and [4], and is shown in Table 14.1. We converted the "LR" fuzzy numbers in an example in [4] to triangular fuzzy numbers. There are two ($m = 2$) independent variables $\overline{X}_1$ and $\overline{X}_2$. Also, there are ten ($n = 10$) items in the data set. We will need to find intervals I_i, $i = 0, 1, 2$, as explained above for the a_i. We will solve for these intervals two ways: (1) first, in the next subsection, using the solutions for the a_i, $i = 0, 1, 2$, from [2]; and (2) secondly, in the second subsection, we use two optimization procedures to determine these intervals.

The authors in [2] compared their method, applied to the data in Table 14.1, to that in [5] applied to the same data set, in their Table 3.2. They give the error values E_1 for both methods. In this chapter we apply our Monte Carlo method to compute predicted values and compare our new results to the other two methods using error measures E_1 and E_2. All programs were written in MATLAB [8]. An

Table 14.1. Data for the Application

Fuzzy Output $\overline{Y}_l$	$\overline{X}_{1l}$	$\overline{X}_{2l}$
(55.4/61.6/64.7)	(5.7/6.0/6.9)	(5.4/6.3/7.1)
(50.5/53.2/58.5)	(4.0/4.4/5.1)	(4.7/5.5/5.8)
(55.7/65.5/75.3)	(8.6/9.1/9.8)	(3.4/3.6/4.0)
(61.7/64.9/74.7)	(6.9/8.1/9.3)	(5.0/5.8/6.7)
(69.1/72.7/80.0)	(8.7/9.4/11.2)	(6.5/6.8/7.1)
(49.6/52.2/57.4)	(4.6/4.8/5.5)	(6.7/7.9/8.7)
(47.7/50.2/55.2)	(7.2/7.6/8.7)	(4.0/4.2/4.8)
(41.8/44.0/48.4)	(4.2/4.4/4.8)	(5.4/6.0/6.3)
(45.7/53.8/61.9)	(8.2/9.1/10.0)	(2.7/2.8/3.2)
(45.4/53.5/58.9)	(6.0/6.7/7.4)	(5.7/6.7/7.7)

example MATLAB program is given at the end of this chapter. For all our calculations, we used a Pentium III, Processor: 933 MHz.

14.4.1 First Choice of Intervals

Now we need to find intervals I_i, $i = 0, 1, 2$, as explained above, for the a_i. After studying the solutions for the a_i ($a_0 = 4.19$, $a_1 = 4.97$, $a_2 = 3.11$) in [2] we decided on the following intervals for our fuzzy Monte Carlo method: (1) $I_0 = [0, 5]$ for a_0; (2) $[0, 6] = I_1$ for a_1; and (3) $I_2 = [0, 4]$ for a_2.

Table 14.2. Results of the Monte Carlo (MCI), Method First Choice of Intervals, to Minimize the Error

Coefficient	$MCI\ E_1$	$MCI\ E_2$
a_0	3.9855	1.462
a_1	0.0060	4.837
a_2	0.0096	3.7712

Using our Sobol quasi-random number generator we produced $100,000$ vectors $v_k = (a_{0k}, a_{1k}, a_{2k})$ which defined the $a_i \in I_i$, $i = 0, 1, 2$, as described in Section 14.3. Results for the a_i, that minimize the error, are shown in Table 14.2, with minimum error values in Table 14.8. Since we will have four Monte Carlo studies on this data we call this one MCI.

Next we experimented with other intervals. We started with larger intervals, shifted them and shortened them, until we arrived at: (1) $I_0 = [0, 37]$ for a_0; (2) $I_1 = [0, 6]$ for a_1; and (3) $[0, 6] = I_2$ for a_2. After another run of $100,000$ quasi-random vectors the results for the a_i are in Table 14.3 with minimum error values in Table 14.8. This Monte Carlo study is called MCII. For these choices of intervals, in MCI and MCII, the computing time was between 40 and 45 minutes.

Table 14.3. Results of the Monte Carlo (MCII) Method, First Choice of Intervals, to Minimize the Error

Coefficient	$MCII\ E_1$	$MCII\ E_2$
a_0	35.3251	1.4652
a_1	3.9498	4.8378
a_2	0.0063	3.7692

14.4.2 Second Choice of Intervals

The first thing to do is to determine the intervals $I_i = [c_i, d_i]$ for the a_i, $0 \leq i \leq 2$. We first describe an optimization method used to determine these intervals. This procedure will be called $MCIII$. A second optimization method will be used and it will be described below. Let

$$[L_l, R_l] = I_0 + I_1\overline{X}_{1l}[0] + I_2\overline{X}_{2l}[0], \tag{14.6}$$

evaluated using interval arithmetic (all intervals are positive), for $l = 1, 2, ..., 10$. Recall the data $\overline{Y}_l = (y_{l1}/y_{l2}/y_{l3})$. Define

$$W = \sum_{l=1}^{10}(L_l - y_{l1})^2 + \sum_{l=1}^{10}(R_l - y_{l3})^2. \tag{14.7}$$

The optimization problem is to minimize W subject to $c_i \leq d_i$ all i. We want to find the intervals that make L_l and R_l closest, in the sense of minimizing W, to the end points of the bases of the dependent fuzzy numbers $\overline{Y}_l$ in the data. We solved this using Maple [7]. The results, rounded to three decimal places, are in Table 14.4. It is very interesting that the two of the "intervals" are degenerate and are just real numbers. Using these intervals $a_0 = 16.528$, $a_1 \in [3.558, 3.982]$ and $a_2 = 2.575$ and we will need to randomly generate only one value for a_1.

We now produce a sequence of random crisp vectors (only one is random) $v_k = (a_{0k}, a_{1k}, a_{2k})$, $k = 1, 2, ..., N$, using our Sobol quasi-random number generator as described in Section 4.2, with $a_{ik} \in I_i$, $i = 0, 1, 2$ and computed minimum E_{1k} and minimum E_{2k}. After a run of $N = 100,000$ the smallest E_1 value and the minimum E_2 value found are shown in Table 14.8 with corresponding a_i shown in Table 14.5.

Table 14.4. First Method of Determining the Intervals (MCIII) for Monte Carlo

Interval	Value
I_0	$[16.528, 16.528]$
I_1	$[3.558, 3.982]$
I_2	$[2.575, 2.575]$

Table 14.5. Results of the Monte Carlo Method (MCIII), Second Choice of Intervals, to Minimize the Error

Coefficient	*MCIII* E_1	*MCIII* E_2
a_0	16.528	16.528
a_1	3.9820	3.7132
a_2	2.575	2.575

We also investigated a second optimization procedure for finding the intervals. L_l and R_l are defined as above. This method is called MCIV. Let

$$W = \sum_{l=1}^{10}(y_{l1} - L_l) + \sum_{l=1}^{10}(R_l - y_{l3}). \tag{14.8}$$

The linear programming problem is to minimize W subject to the constraints: (1) $L_l \le y_{l1}$ all l; (2) $R_l \ge y_{l3}$ all l; and (3) $c_i \le d_i$ all i. We solved this problem using Maple [7] and the results are in Table 14.6. It is again interesting that one of the "intervals" is degenerate and is just a real number. Using these intervals $a_0 \in [33.808, 36.601]$, $a_1 \in [1.294, 3.756]$ and $a_2 = 0.473$ and we will need to randomly generate only a value for a_0 and a value for a_1.

We now produce a sequence of random crisp vectors (only two are random) $v_k = (a_{0k}, a_{1k}, a_{2k})$, $k = 1, 2, ..., N$, using our Sobol quasi-random number generator as described in Section 4.2 and computed E_{1k} and E_{2k}. After a run of $N = 100,000$ the smallest E_1 value and the minimum E_2 value found are shown in Table 14.8 with corresponding a_i shown in Table 14.7. The total computing time for the second choice of intervals, MCIII and MCIV, was between 27 and 35 minutes.

14.4.3 Comparison of Solutions

In Example 2 in [4], we changed all LR fuzzy numbers to triangular fuzzy numbers; the authors give the model they fit to the data (Table 14.1) from which we can determine their predicted values for the dependent variable and hence E_1 and E_2. This is what we report in Table 14.8. The authors in [2] reported only $E_1 = 12.031$ for [5] and no E_2 value.

We see from Table 14.8 that our Monte Carlo method obtained the smallest values for error measure E_1 for MCII and MCIV. MCI and MCII gave the smallest error values for E_2. Let us explain why we did not expect Monte Carlo to do much better than [2] on E_2.

Let y_{lc} be the centroid of $\overline{Y}_l$, $1 \le l \le n$, and let x_{ilc} be the centroid of $\overline{X}_{il}$, $1 \le i \le m$, $1 \le l \le n$. In [2] the authors first estimate the a_i, $0 \le i \le m$ from

$$min \sum_{l=1}^{n} |y_{lc} - [a_0 + \sum_{j=1}^{m} a_j x_{jlc}]|, \tag{14.9}$$

Table 14.6. Second Method of Determining the Intervals (MCIV) for Monte Carlo

Interval	Value
I_0	$[33.808, 36.601]$
I_1	$[1.294, 3.756]$
I_2	$[0.473, 0.473]$

Table 14.7. Results of the Monte Carlo Method (MCIV), Second Choice of Intervals, to Minimize the Error

Coefficient	$MCIV\ E_1$	$MCIV\ E_2$
a_0	33.8196	33.7081
a_1	3.7559	3.3208
a_2	0.473	0.473

Table 14.8. Error Measures for the Application (na=not available)

Error	[4]	[2]	[5]	MCI	MCII	MCIII	MCIV
E_1	13.5748	11.1060	12.031	10.0170	9.3888	12.7267	9.5933
E_2	141.6260	137.8470	na	133.1181	133.1239	146.5307	170.1175

giving $\hat{a}_i$, $0 \leq i \leq m$. Next let

$$\overline{Z}_l = \hat{a}_0 + \hat{a}_1 \overline{X}_{1l} + ... + \hat{a}_m \overline{X}_{ml}, \tag{14.10}$$

for $l = 1, 2, ..., n$. The model now is the predicted value for the dependent variable is

$$\overline{Y}_l^* = \overline{Z}_l + \overline{E}, \tag{14.11}$$

for "error" term $E = (-L/0/R)$. The authors then estimated L and R by minimizing new sums of absolute values of differences. This is somewhat like finding the a_i to minimize E_2. Hence, we expected [2] to have a minimum value for E_2.

14.5 Summary and Conclusions

In this chapter we studied the fuzzy linear regression problem given in equation (14.1). We employed our fuzzy Monte Carlo method to approximate the "best" solutions for the crisp coefficients a_i, $0 \leq i \leq m$. Best will be measured by two error measures E_1 (equation (14.3)) and E_2 (equation (14.4)). We showed in an example problem that our Monte Carlo method was best according to E_1 with respect to the results on the same data set in three other publications. Monte Carlo also obtained the smallest E_2 value. Given any error measure E^* we conjecture that our Monte Carlo method, allowing the number of iterations N to be sufficiently large, will be best (minimizing E^*). If this conjecture is true, then the estimation technique in fuzzy linear regression may become Monte Carlo.

14.6 MATLAB Program

This program is to apply Mont Carlo Methods to a Fuzzy Linear Regression Problem to estimate the best solution (vector of crisp numbers). It runs on a data file, data.dat, that contains N quasi-random Vectors each of length 3, generated by the Sobol subroutine.

MC.m

```
%
% Initializing The minimum values for the Error measures and their indices
%
min_E1 = 10000; index = 0; min_E2 = 10000; index_E2 = 0;
N = 100000 ;               % The number of runs of the algorithm.
% The data from table 1
    Y = [55.4,61.6,64.7,50.5,53.2,58.5,55.7,65.5,75.3,61.7,64.9,74.7,69.1,
         72.7,80.0,49.6,52.2,57.4,47.7,50.2,55.2,41.8,44.0,48.4,45.7,53.8,
         61.9,45.4,53.5,58.9];
    X1 = [5.7,6.0,6.9,4,4.4,5.1,8.6,9.1,9.8,6.9,8.1,9.3,8.7,9.4,11.2,4.6,
          4.8,5.5,7.2,7.6,8.7,4.2,4.4,4.8,8.2,9.1,10.0,6.0,6.7,7.4];
    X1 = reshape(X1,3,10)';
    X2 = [5.4,6.3,7.1,4.7,5.5,5.8,3.4,3.6,4.0,5.0,5.8,6.7,6.5,6.8,7.1,6.7,
          7.9,8.7,4.0,4.2,4.8,5.4,6,6.3,2.7,2.8,3.2,5.7,6.7,7.7];
    X2 = reshape(X2,3,10)';
%
% Reading N vectors of length 3, generated by the Quasi-Random Vector
% Generator, Sobol
%
fid_2 = fopen('data.dat','r');
D = fscanf(fid_2,'%f');
fclose(fid_2); A = reshape(D,3,N)';

fid_1 = fopen('minE.dat','w');  % Opening a file for the output
for k =1:N
    %
    % Initializing E1 and E2 for the kth iteration
    %
    E1 = 0; E2 = 0;
    %
    % Calculating E1 (equation 3) and E2 (equation 4).
    %
    for i=0:9
        a=Y(i*3+1);b=Y(i*3+2);c=Y(i*3+3);
        %
        % Mapping the three real numbers into the intervals
        % I0= [r0,s0], I1= [r1,s1],I2= [r2,s2]
        %
        A0 =(s0-r0)*A(k,1) + r0;
        A1 =(s1-r1)*A(k,2) + r1;
        A2 =(s2-r2)*A(k,3) + r2;
        %
        % Calculating Y* using equation (2)
        %
        YY =A0 + A1*[X1(i+1,1),X1(i+1,2),X1(i+1,3)] +
             A2*[X2(i+1,1),X2(i+1,2),X2(i+1,3)] ;
        %
        % diff_area is a function that evaluates the integral in the
        % numerator in equation 3.
        %
        f  = diff_area(a,b,c,YY(1),YY(2),YY(3));
        E1 = E1 + f/(.5*(c-a));
        E2 = E2 + sum(abs([a,b,c]-YY));
    end         % end of for loop for E1, and E2
    %
    % Storing the minimum values for E1 and E2
    %
    if(E1 < min_E1)
        min_E1 = E1;
        index_E1 = k;
    end
    if(E2 < min_E2)
        min_E2 = E2;
        index_E2 = k;
    end
    %
    % Recording the minimum values for E1 and E2 and their corresponding
    % best solution to the output file
    %
end             % end of for loop, the number of runs
%
% Writing the output to file
%
fclose(fid_1);                              % closing the file
```

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Linear Regression II. Soft Computing (to appear)
2. Choi, S.H, Buckley, J.J.: Fuzzy Regression Using Least Absolute Deviation Estimators. Soft Computing (to appear)
3. Diamond, P.: Fuzzy Least Squares. Information Sciences 46, 141–157 (1988)
4. Diamond, P., Korner, R.: Extended Fuzzy Linear Models and Least Squares Estimates. Computers and Mathematics with Applications 33, 15–32 (1997)
5. Kao, C., Chyu, C.: A Fuzzy Linear Regression Model with Better Explanatory Power. Fuzzy Sets and Systems 126, 401–409 (2002)

6. Kao, C., Chyu, C.: Least-Squares Estimates in Fuzzy Linear Regression Analysis. European J. Operational Research 148, 426–435 (2003)
7. Maple 9.5, Waterloo Maple Inc., Waterloo, Canada
8. MATLAB, The MathWorks, http://www.mathworks.com

15 Fuzzy Two-Person Zero-Sum Games

15.1 Introduction

In this chapter we use fuzzy Monte Carlo methods to get approximate optimal fuzzy mixed strategies for fuzzy two-person zero-sum games. In the next section we briefly review the results for crisp two-person zero-sum games. Then in Section 15.3 we fuzzify the games and define optimal fuzzy values for the players and optimal fuzzy mixed strategies. In the fourth section we introduce our fuzzy Monte Carlo method and use it on an example problem to generate approximate solutions. The last section contains our conclusions and suggestions for future research. Our fuzzy Monte Carlo method will be programmed in MATLAB [6]. This chapter is based on [1].

15.2 Two-Person Zero-Sum Games

There are two players named Player I and Player II. A is a $m \times n$ matrix of real numbers a_{ij}. Player I has pure strategies $i = 1, 2, 3, ..., m$, the row labels, and Player II has pure strategies $j = 1, 2, 3, ..., n$, the labels for the columns. If Player I chooses pure strategy i and Player II chooses pure strategy j, then the payoff from Player II to Player I is a_{ij} when $a_{ij} > 0$. If $a_{ij} < 0$, then the payoff is $-a_{ij}$ from Player I to Player II.

Sometimes the games have optimal strategies for both players in pure strategies. This is when the game has a saddle point. Suppose a_{ij} is both the maximum entry in its column and the minimum entry in its row. We have a saddle point and the pure strategy i for Player I and pure strategy j for Player II are optimal strategies for both players. So assume that the game has no saddle points. We now consider mixed strategies.

A mixed strategy for Player I is a probability vector $x = (x_1, ..., x_m)$, $x_i \in [0, 1]$ all i, and $\sum_{i=1}^{m} x_i = 1$. A mixed strategy for Player II is also a probability vector $y = (y_1, ..., y_n)$, $y_j \in [0, 1]$ all j, $\sum_{j=1}^{n} y_j = 1$. Player I chooses pure strategy i with probability x_i and Player II will choose pure strategy j with probability y_j. The expected payoff to Player I will be

$$E = xAy^t, \tag{15.1}$$

where y^t is the transpose of row vector y.

Let X (Y) be the set of all mixed strategies for Player I (II). For a fixed $x \in X$ let

$$v(x) = min\{xAy^t | y \in Y\}, \tag{15.2}$$

and

$$v_I = max\{v(x) | x \in X\}. \tag{15.3}$$

The value of the game for Player I is v_I and a mixed strategy x^* in X solving equation (15.3) is an optimal mixed strategy for Player I. For a fixed $y \in Y$ define

$$v(y) = max\{xAy^t | x \in X\}, \tag{15.4}$$

and

$$v_{II} = min\{v(y) | y \in Y\}. \tag{15.5}$$

The value of the game for Player II is v_{II} and a mixed strategy y^* in Y solving equation (15.5) is an optimal mixed strategy for Player II. The minimax theorem says that $v_I = v_{II}$.

The details on two-person zero-sum games are in many books and two references are ([8],[10]). We now consider the probability vectors and the payoff matrix becoming fuzzy.

15.3 Fuzzy Two-Person Zero-Sum Games

There have been some papers/chapters in books, about fuzzy two-person zero-sum games which consider fuzzy payoffs, and sometimes fuzzy goals for the fuzzy payoffs ([2],[4],[5],[7],[9],[11],[12]), but not with fuzzy mixed strategies. We will allow both fuzzy payoffs and fuzzy mixed strategies.

We first fuzzify the payoff matrix $\overline{A} = (\overline{a}_{ij})$ where the $\overline{a}_{ij}$ are trapezoidal fuzzy numbers, or real numbers. Some of the payoffs can be real numbers but we still write all of them as fuzzy numbers. For example, if $a_{23} = 20$ we write $\overline{a}_{23} = 20$. The fact that a $\overline{a}_{ij}$ is fuzzy represents any uncertainty in the exact value of the payment.

How do we get these trapezoidal fuzzy numbers. We could employ expert opinion if we do not have any historical/statistical data to estimate these parameters. Suppose a_{34} is an uncertain payoff value in the fuzzy matrix $\overline{A}$. First assume we have only one expert and he/she is to estimate the value of some a_{ij}. We can solicit this estimate from the expert as is done in estimating job times in project scheduling ([10], Chapter 13). Let $a =$ the "pessimistic" value of a_{ij}, or the smallest possible value, let $d =$ the "optimistic" value of a_{ij}, or the highest possible value, and let $[b, c]$ be the interval of the most likely values of a_{ij}. We then ask the expert to give values for a, b, c, d and we construct the trapezoidal fuzzy number $\overline{a}_{34} = (a/b, c/d)$ for a_{34}. If we have a group of experts all to estimate the value of some a_{ij} we would average their response.

We define a saddle point for the fuzzy games the same way it was defined for the crisp game. Adopt the definitions of $<$, $\approx$ and $>$ between fuzzy numbers discussed in Section 15.4.2. We say the fuzzy game has a saddle point at $\overline{a}_{34}$ if $\overline{a}_{34} \geq \overline{a}_{i4}$, $i = 1, 2, \ldots, n$ and $\overline{a}_{34} \leq \overline{a}_{3j}$, $j = 1, 2, \ldots, m$. A $\leq$ ($\geq$) between fuzzy numbers means $<$ or $\approx$ ($>$ or $\approx$). If $\overline{a}_{34}$ is a saddle point the players have optimal strategies in pure strategies: Player I chooses the third row and Player II picks the fourth column. So assume that the fuzzy game does not have a saddle point.

Next the probability vectors for mixed strategies become fuzzy probabilities [3]. A fuzzy mixed strategy for Player I is $\overline{x} = (\overline{x}_1, ..., \overline{x}_m)$ where $\overline{x}_i \in [0, 1]$ is a triangular fuzzy number or a real number, all i, and there are $x_i \in \overline{x}_i[1]$ so that $\sum_{i=1}^{m} x_i = 1$. The last constraint says that for any $\alpha \in [0, 1]$ we can find $x_i \in \overline{x}_i[\alpha]$, $1 \leq i \leq m$, so that $x_1 + ... + x_m = 1$. Similarly we define a fuzzy mixed strategy $\overline{y}$ for Player II. For example if $m = 4$ we could have $\overline{x} = (0.2, \overline{x}_2, \overline{x}_3, 0.3)$ where $\overline{x}_1 = 0.2$, $\overline{x}_2 = (0/0.1/0.2)$, $\overline{x}_3 = (0.3/0.4/0.5)$ and $\overline{x}_4 = 0.3$. In this example we could use $(0.2, 0.2, 0.3, 0.3)$ when $\alpha = 0$ for a crisp mixed strategy. Let $\mathcal{X}$ ($\mathcal{Y}$) be all fuzzy mixed strategies for Player I (Player II). We wish to define, and find, optimal $\overline{x}^* \in \mathcal{X}$ ($\overline{y}^* \in \mathcal{Y}$) for Player I (Player II). We could consider trapezoidal fuzzy numbers for the $\overline{x}_i$ in $\overline{x}$ but we will use triangular fuzzy numbers in this chapter.

The fuzzy expected payoff $\overline{E}$ from the fuzzy game is determined by its α-cuts [3]

$$\overline{E}(\overline{x}, \overline{y})[\alpha] = \{\sum_{i=1}^{m} \sum_{j=1}^{n} x_i a_{ij} y_j | \ \mathcal{S} \ \}, \tag{15.6}$$

where $\mathcal{S}$ denotes the statement " $\overline{x} \in \mathcal{X}$, $x_i \in \overline{x}_i[\alpha]$ all i, $x_1 + ... + x_m = 1$, $\overline{y} \in \mathcal{Y}$, $y_j \in \overline{y}_j[\alpha]$ all j, $y_1 + ... + y_n = 1$ and $a_{ij} \in \overline{a}_{ij}[\alpha]$ all i, j ". This is how we will compute with fuzzy probabilities: for any α-cut we always choose only crisp probability distributions [3]. We may find these $\alpha-$cuts as follows

$$e(\overline{x}, \overline{y})_1(\alpha) = min\{\sum_{i=1}^{m} \sum_{j=1}^{n} x_i a_{ij} y_j | \ \mathcal{S} \ \}, \tag{15.7}$$

and

$$e(\overline{x}, \overline{y})_2(\alpha) = max\{\sum_{i=1}^{m} \sum_{j=1}^{n} x_i a_{ij} y_j | \ \mathcal{S} \ \}, \tag{15.8}$$

where $\overline{E}(\overline{x}, \overline{y})[\alpha] = [e(\overline{x}, \overline{y})_1(\alpha), e(\overline{x}, \overline{y})_2(\alpha)]$. We will need to solve this optimization problem using the Optimization Toolbox in MATLAB [6]. Also, this is the way we compute with fuzzy probabilities. We use "complete" crisp probabilities selected from the fuzzy numbers $\overline{x}_i$, $1 \leq i \leq m$ ($\overline{y}_j$, $1 \leq j \leq n$). Equation (15.6) defines the $\alpha-$cuts of trapezoidal shaped fuzzy number $\overline{E}(\overline{x}, \overline{y})$. Notice that $\overline{E}(\overline{x}, \overline{y})$ was not evaluated using the extension principle nor by α-cuts and interval arithmetic.

Now we define optimal fuzzy mixed strategies and consider a fuzzy minimax theorem. $\mathcal{X}$ ($\mathcal{Y}$) is the set of fuzzy mixed strategies to be used by Player I (Player II). These could be finite, or some other infinite restricted set of fuzzy probabilities, and are not necessarily all possible fuzzy probabilities. Let us first assume that $\mathcal{X}$ ($\mathcal{Y}$) is all fuzzy mixed strategies for Player I (Player II). We will use some method of defining $\leq$, $<$ and $\approx$ between fuzzy numbers, and in choosing the maximum/minimum of a (finite) set of fuzzy numbers, which will be defined and discussed in the next section. For each $\overline{x} \in \mathcal{X}$ define the fuzzy function

$$\overline{V}(\overline{x}) = min\{\overline{E}(\overline{x}, \overline{y}) \mid \overline{y} \in \mathcal{Y}\ \}. \tag{15.9}$$

Then the fuzzy value for the game for Player I is

$$\overline{V}_I = max\{\overline{V}(\overline{x}) \mid \overline{x} \in \mathcal{X}\ \}. \tag{15.10}$$

But we are unable to determine the fuzzy function $\overline{V}(\overline{x}) = \overline{z}$, $\overline{z}$ a trapezoidal shaped fuzzy number. For this reason we will employ our fuzzy Monte Carlo method to approximate $\overline{V}_I$ ($\overline{V}_{II}$) for Player I (Player II). This means that we will restrict $\mathcal{X}$ and $\mathcal{Y}$ to be finite sets of fuzzy mixed strategies.

In order for the notation here to match that in the next section on our Monte Carlo method we will now assume that $\mathcal{X}$ and $\mathcal{Y}$ are finite. Let $\mathcal{X} = \{\overline{x}_i \mid i = 1, \ldots, N\ \}$ and $\mathcal{Y} = \{\overline{y}_j \mid j = 1, \ldots, N\ \}$. We have changed our notation where now $\overline{x}_i$ ($\overline{y}_j$) is the whole fuzzy mixed strategy for Player I (II) and not a component of a fuzzy mixed strategy. That is, now $\overline{x}_i = (\overline{x}_{i1}, \ldots, \overline{x}_{im})$, $i = 1, \ldots, N$, and a similar expression for $\overline{y}_j$. Pick and fix $\overline{x}_i \in \mathcal{X}$. Let

$$\overline{E}(\overline{x}_i, \overline{y}_j) = \overline{U}_{ij}, \tag{15.11}$$

for $j = 1, \ldots, N$. Next compute

$$\overline{U}_i = min\{\overline{U}_{ij} \mid j = 1, \ldots, N\ \}. \tag{15.12}$$

We do this for each $i = 1, 2, \ldots, N$.

Next we determine

$$\overline{U}^* = max\{\overline{U}_i \mid i = 1, \ldots, N\ \}. \tag{15.13}$$

Now $\overline{U}^*$ will equal $\overline{U}_i$ for some i. If $\overline{U}^* = \overline{U}_{143}$, then set $\overline{x}^* = \overline{x}_{143}$. The fuzzy value for the game for Player I is $\overline{V}_I = \overline{U}^*$ and his/her optimal fuzzy mixed strategy is $\overline{x}^*$. Now we do similar calculations for Player II.

Pick and fix $\overline{y}_j$ in $\mathcal{Y}$. Let

$$\overline{E}(\overline{x}_i, \overline{y}_j) = \overline{V}_{ij}, \tag{15.14}$$

for $i = 1, \ldots, N$. Next compute

$$\overline{V}_j^* = max\{\overline{V}_{ij} \mid i = 1, \ldots, N\ \}. \tag{15.15}$$

We do this for each $j = 1, 2, \ldots, N$.

Next we determine

$$\overline{V}^* = min\{\overline{V}_j \mid j = 1, \ldots, N\ \}. \tag{15.16}$$

Now $\overline{V}^*$ will equal $\overline{V}_j$ for some j. If $\overline{V}^* = \overline{V}_{643}$, then set $\overline{y}^* = \overline{y}_{643}$. The fuzzy value for the game for Player II is $\overline{V}_{II} = \overline{V}^*$ and his/her optimal fuzzy mixed strategy is $\overline{y}^*$.

Now compare $\overline{U}^*$ and $\overline{V}^*$ both are trapezoidal shaped fuzzy numbers. Can they be equal? This would be the fuzzy minimax theorem. We will investigate this possibility using our fuzzy Monte Carlo method discussed in detail in the next section. However we do have the following result. We assume that we are using definitions of $<$, $\leq$ and $\approx$ between fuzzy numbers (see Section 15.4.2) so that: (1) we may find a unique solution to $max\{\overline{A}_\tau \mid \tau \in \Upsilon\}$ and to $min\{\overline{A}_\tau \mid \tau \in \Upsilon\}$ for the $\overline{A}_\tau$ fuzzy numbers and Υ an index set; and (2) given two fuzzy numbers $\overline{M}$ and $\overline{N}$ one and only one of the following is true $\overline{M} < \overline{N}$, $\overline{M} \approx \overline{N}$, $\overline{M} > \overline{N}$.

Theorem 15.1. $\overline{V}_I \leq \overline{V}_{II}$.

Proof
For $\overline{y} \in \mathcal{Y}$ consider

$$max\{\overline{E}(\overline{x}, \overline{y}) \mid \overline{x} \in \mathcal{X}\} = \overline{V}(\overline{y}). \tag{15.17}$$

This defines a mapping from each $\overline{y} \in \mathcal{Y}$ to an $\overline{x}^*$ in $\mathcal{X}$ so that $\overline{E}(\overline{x}^*, \overline{y}) = \overline{V}(\overline{y})$. We write $\overline{x}^* = f(\overline{y})$ so that $\overline{E}(f(\overline{y}), \overline{y}) = \overline{V}(\overline{y})$. Then

$$\overline{E}(\overline{x}, \overline{y}) \leq \overline{E}(f(\overline{y}), \overline{y}), \tag{15.18}$$

for all $\overline{x}$ and all $\overline{y}$. It follows that

$$min\{\overline{E}(\overline{x}, \overline{y}) \mid \overline{y} \in \mathcal{Y}\} \leq min\{\overline{E}(f(\overline{y}), \overline{y}) \mid \overline{y} \in \mathcal{Y}\}, \tag{15.19}$$

for all $\overline{x}$. But the right side of equation (15.19) is $\overline{V}_{II}$. Hence

$$min\{\overline{E}(\overline{x}, \overline{y}) \mid \overline{y} \in \mathcal{Y}\} \leq \overline{V}_{II}, \tag{15.20}$$

for all $\overline{x}$. Now take the max on $\overline{x} \in \mathcal{X}$ of the left side of equation (15.20) and the result follows. ■

A complete fuzzy Monte Carlo study would generate N fuzzy mixed strategies $\overline{x} \in \mathcal{X}$ and N fuzzy mixed strategies $\overline{y} \in \mathcal{Y}$, for $l = 1, 2, 3, \ldots, L$. Each study would produce fuzzy values $\overline{V}_I^{(l)}$, $\overline{V}_{II}^{(l)}$, and optimal fuzzy mixed strategies $\overline{x}^*_{(l)}$, $\overline{y}^*_{(l)}$, $l = 1, 2, \ldots, L$. Then we would compare these fuzzy values to choose our final approximations to the fuzzy value of the game for the players and their optimal fuzzy mixed strategies. However, because of the long computer time to accomplish each fuzzy Monte Carlo study, we will be able to do only one of them in the example in Section 15.4.3.

15.4 Fuzzy Monte Carlo

Assume that the fuzzy payoff matrix $\overline{A}$ is given. We first need to do two things: (1) describe how to get random sequences of fuzzy mixed strategies $\overline{x}_k$ and $\overline{y}_k$,

$k = 1, \ldots, N$; and (2) how we will determine the maximum/minimum of a finite set of fuzzy numbers. We first consider random sequences of fuzzy mixed strategies and then the max/min of a finite set of fuzzy numbers (equations (15.12),(15.13),(15.15),(15.16)). Then we can outline our fuzzy Monte Carlo method for producing an approximate solution to the problem discussed above and consider an example problem.

15.4.1 Random Sequences of Fuzzy Mixed Strategies

To obtain a random sequence $\overline{x}_k = (\overline{x}_{k1}, \ldots, \overline{x}_{km})$, $k = 1, 2, ..., N$, where each $\overline{x}_{kj}$ is a triangular fuzzy number in $[0,1]$ and $\sum_{j=1}^{m} \overline{x}_{kj}[1] = 1$, we first randomly generate crisp vectors $v_k = (a_{k1}, ..., a_{k,3m})$ with all the a_{ki} in $[0,1]$, $k = 1, 2, .., N$. We obtain the sequence v_k using our Sobol quasi-random number generator discussed in Chapter 3. We choose the first three numbers in v_k and order them from smallest to largest. Assume that $a_{k3} < a_{k1} < a_{k2}$. Then the first triangular fuzzy number $\overline{z}_{k1} = (a_{k3}/a_{k1}/a_{k2})$. Continue with the next three numbers in v_k, making $\overline{z}_{k2}$, etc. Assume $\overline{z}_{kj} = (z_{kj1}/z_{kj2}/z_{kj3})$, all k and j. Let $L_k = \sum_{j=1}^{m} z_{kj2}$. Then the final $\overline{x}_k$ is $\overline{x}_{kj} = (1/L_k) z_{kj}$ all k and j. If $\overline{x}_{kj} = (x_{kj1}/x_{kj2}/x_{kj3})$ we now have $\sum_{j=1}^{m} x_{kj2} = 1$. We construct the random sequence of fuzzy mixed strategies $\overline{y}_k$ for Player II the same way.

15.4.2 Max/Min of Fuzzy Numbers

Given a finite set of fuzzy numbers $\overline{U}_1, ..., \overline{U}_N$ we want to find the maximum and the minimum. For a finite set of real numbers there is no problem in ordering them from smallest to largest. However, in the fuzzy case there is no universally accepted way to do this. There are probably more than 50 methods proposed in the literature of defining $\overline{M} \leq \overline{N}$, for two fuzzy numbers $\overline{M}$ and $\overline{N}$.

Here we will use only Buckley's Method presented in Section 2.6.1. We will now use $\eta = 0.9$ in Buckley's Method to help reduce the number of fuzzy numbers that could be considered approximately equal for the maximum/minimum of a set of fuzzy numbers. But note that different definitions of $\leq$ between fuzzy numbers can give different orderings and therefore different final answers to the fuzzy game theory problem.

Now apply this to $\overline{U}_{ij}$, $j = 1, \ldots, N$, in equation (15.12). We will find the minimum sequentially. Suppose we are at stage $j = T - 1$ and the current minimum of $\overline{U}_{ij}$, $1 \leq j \leq T - 1$, is $\overline{S}$. The next step computes $\overline{U}_{iT} = \overline{R}$. There are three possibilities: (1) if $\overline{S} < \overline{R}$, then min remains $\overline{S}$ go on to the next step; (2) if $\overline{S} > \overline{R}$, then the new min is $\overline{R}$; and (3) if $\overline{S} \approx \overline{R}$, there are three more cases. Let $\overline{S} \approx (s_1/s_2, s_3/s_4)$ and $\overline{R} \approx (r_1/r_2, r_3/r_4)$ since they will be trapezoidal shaped fuzzy numbers. The next three cases are: (1) if $s_2 < r_2$, then min remains $\overline{S}$; (2) if $s_2 > r_2$, the min is now $\overline{R}$; and (3) if $s_2 = r_2$, there are three more cases. At this point $\overline{S} \approx \overline{R}$ and $s_2 = r_2$. The three new cases are: (1) if $s_3 < r_3$, the min remains $\overline{S}$; (2) if $s_3 > r_3$, then the min is $\overline{R}$; and (3) if $s_3 = r_3$, there are three more cases. We are at $\overline{S} \approx \overline{R}$, $s_2 = r_2$ and $s_3 = r_3$. The next three cases are: (1) if $s_1 < r_1$, then the min remains $\overline{S}$; (2) if $s_1 > r_1$, then the min is $\overline{R}$; and (3) if $s_1 = r_1$, then we have a final three cases to consider. We are at $\overline{S} \approx \overline{R}$, $s_2 = r_2$, $s_3 = r_3$ and $s_1 = r_1$. The final three cases are: (1) if $s_4 < r_4$, then the minimum remains $\overline{S}$; (2) if $s_4 > r_4$, then the minimum is $\overline{R}$; and (3) if $s_4 = r_4$, then randomly delete one of them and the other is the min. We do this so that there will be one and only one minimum. It is clear what changes are needed for a maximum.

15.4.3 Fuzzy Monte Carlo Solution Method

The basic program was written in MATLAB [6]. We must decide on N, the number of random fuzzy mixed strategies for each player and the number of α-cuts we need to determine for all the fuzzy numbers. For the α-cuts we need $\alpha = 0, 1$ and $\alpha = 0.9$ for the comparison of two fuzzy numbers (Section 15.4.2). We will use one more α-cut between zero and 0.9 so the α-cuts will be $\alpha = 0, 0.4, 0.9, 1$. After generating the random fuzzy mixed strategies $\overline{x}_k$ and $\overline{y}_k$, $k = 1, \ldots, N$, we need to evaluate equation (15.11) N^2 times. Also, each one is done four times for the α-cuts. So to get to equation (15.13) we compute equation (15.6) $4N^2$ times. For $\overline{y}^*$ also $4N^2$ times. A total of $8N^2$. With $N = 1000$ that equals $8,000,000$. That seems like the absolute max for our computer. So for the example below we will pick the smallest $\overline{A}$, a 2×2 fuzzy payoff matrix.

Before we consider the example let us look more closely at our MATLAB program. Let $\overline{E}(\overline{x}_i, \overline{y}_j) = \overline{W}_{ij}$. We previously called this $\overline{U}_{ij}$ in equation (15.11) and then $\overline{V}_{ij}$ in equation (15.14). Now we call it just $\overline{W}_{ij}$, a trapezoidal shaped fuzzy number evaluated at α-cuts, $\alpha = 0, 0.4, 0.9, 1$. We compute these α-cuts as in equations (15.7) and (15.8) using the Optimization Toolbox in MATLAB. Now form a $N \times N$ matrix rows labeled $\overline{x}_i$, columns labeled $\overline{y}_j$, whose ij^{th} element is the α-cuts of $\overline{W}_{ij}$. For each row $\overline{x}_i$ scan the row $j = 1, \ldots, N$ for the minimum $\overline{U}_i$ producing column vector $(\overline{U}_1, \ldots, \overline{U}_N)^t$. Now scan this column vector for the maximum $\overline{V}_I$. Next for each column $\overline{y}_j$ scan the row $i = 1, \ldots, N$ for the maximum $\overline{V}_j$ producing the row vector $(\overline{V}_1, \ldots, \overline{V}_N)$. Scan this row vector for its minimum $\overline{V}_{II}$.

Example 15.4.3.1

The fuzzy payoff matrix $\overline{A}$ will be 2×2 with trapezoidal fuzzy numbers $\overline{a}_{11} = (0/1, 2/3)$, $\overline{a}_{12} = 0$, $\overline{a}_{21} = (-2/-1, 0/1)$ and $\overline{a}_{22} = (1/2, 3/4)$. Now we want to estimate the fuzzy values of the game using $\overline{A}$ and random fuzzy mixed strategies for both players. The results are in Table 15.1 and Figure 15.1. In Figure 15.1 we approximated the trapezoidal shaped fuzzy numbers $\overline{V}_I$ and $\overline{V}_{II}$ by trapezoidal fuzzy numbers using only the support and core.

We see that the intersection of the core of $\overline{V}_I$ with the core of $\overline{V}_{II}$ is non-empty. Then from our definition of $<$ and $\approx$ between fuzzy numbers at the beginning of Section 15.4.2 we get $\overline{V}_I \approx \overline{V}_{II}$. However, if we also use the rest of Section 15.4.2 where we "fine tuned" $<$ between fuzzy numbers to obtain a unique max/min of a set of fuzzy numbers, we have $\overline{V}_I < \overline{V}_{II}$. Our Monte Carlo study showed $\overline{V}_I \leq \overline{V}_{II}$ which is the theorem in Section 15.3.

Table 15.1. Optimal Results from the Fuzzy Monte Carlo Method

Player I	Player II
$\overline{x}_1^* = (0.0398/0.7491/1.3658)$	$\overline{y}_1^* = (0.5247/0.5842/0.6245)$
$\overline{x}_2^* = (0.1432/0.2509/2.5463)$	$\overline{y}_2^* = (0.3454/0.4158/0.5953)$
$\overline{V}_I \approx (-1.8958/0.4983, 1.4671/2.6918)$	$\overline{V}_{II} \approx (-0.4976/0.7383, 1.7183/2.9469)$

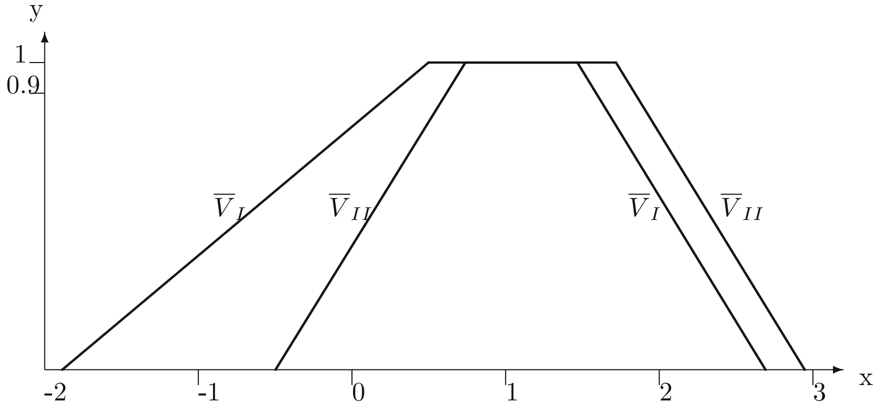


Fig. 15.1. Fuzzy Values $\overline{V}_I$ and $\overline{V}_{II}$ for the Players from the Monte Carlo Method

15.5 Conclusions and Future Research

In this chapter we considered a two-person zero-sum game with fuzzy payoffs and fuzzy mixed strategies for both players. We defined the fuzzy value of the game for both players ($\overline{V}_I$,$\overline{V}_{II}$) and also defined an optimal fuzzy mixed strategy for both players. We showed that $\overline{V}_I \leq \overline{V}_{II}$. We then employed our fuzzy Monte Carlo method to produce approximate solutions, to an example fuzzy game with no (fuzzy) saddle point, for the fuzzy values $\overline{V}_I$ for Player I and $\overline{V}_{II}$ for Player II; and also approximate solutions for the optimal fuzzy mixed strategies for both players. We then looked at $\overline{V}_I$ and $\overline{V}_{II}$ to see if there could be a Minimax theorem ($\overline{V}_I = \overline{V}_{II}$) for this fuzzy game. All our Monte Carlo study showed was $\overline{V}_I \leq \overline{V}_{II}$ which was the theorem in Section 15.3. So, it remains an open question will $\overline{V}_I = \overline{V}_{II}$ for these fuzzy games?

For all our Monte Carlo calculations, we used a Dell Optiplex GX 250 with a dual core and a 64-bit pentium D 2.8 GHz processor running on Windows XP. The computer time for $N = 100,000$ random fuzzy mixed strategies for both players was approximately 68 hours.

It would be nice to try $1,000,000$ random fuzzy mixed strategies for both players, but the computing time would be too excessive. However, if we could run the fuzzy Monte Carlo program with $N = 100,000$ random fuzzy mixed strategies for each player simultaneously on ten separate machines and then combine the results, we could go to $1,000,000$. The MATLAB program is available from the authors.

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Game Theory. New Mathematics and Natural Computation 3, 259–269 (2007)
2. Bector, C.R., Chandra, S.: Fuzzy Mathematical Programming and Fuzzy Matrix Games. Springer, Heidelberg (2005)
3. Buckley, J.J.: Fuzzy Probabilities: New Approach and Applications. Springer, Heidelberg (2003)
4. Liu, S.-T., Kao, C.: Solution of Fuzzy Matrix Games: An Application of the Extension Principle. Int. J. Intelligent Systems 22, 891–903 (2007)
5. Maeda, T.: On Characterization of Equilibrium Strategy of Two-Person Zero-Sum Games with Fuzzy Payoffs. Fuzzy Sets and Systems 139, 283–296 (2003)
6. MATLAB, The MathWorks, http://www.mathworks.com
7. Nishizaki, I., Sakawa, M.: Fuzzy and Multiobjective Games for Conflict Resolution. Physica-Verlag, Heidelberg (2001)
8. Owen, G.: Game Theory. W.B.Saunders Company, Philadelphia, PA (1968)
9. Sakawa, M., Nishizaki, I.: Max-Min Solutions for Fuzzy Multiobjective Matrix Games. Fuzzy Sets and Systems 67, 53–69 (1994)
10. Taha, H.A.: Operations Research. Macmillan, New York (1992)
11. Vijay, V., Chandra, S., Bector, C.R.: Matrix Games with Fuzzy Goals and Fuzzy Payoffs. Omega 33, 425–429 (2005)
12. Xu, L., Zhao, R., Shu, T.: Three Equilibrium Strategies for Two-Person Zero-Sum Games with Fuzzy payoffs. In: Wang, L., Jin, Y. (eds.) FSKD 2005. LNCS (LNAI), vol. 3613, Springer, Heidelberg (2005)

16 Fuzzy Queuing Models

16.1 Introduction

This chapter is based on, and expanded from, Chapters 11, 12 and 14 of [4] which is about using fuzzy probabilities and fuzzy sets in web site planning. So the queuing network considered in this chapter is within a web site. For other papers/chapters in books, on this topic of fuzzy queuing theory, we refer the reader to ([2],[3],[6],[7],[10]) and the references in these papers/books. In the next section we discuss the crisp queuing optimization problem and then we fuzzify the optimization problem in the third section. In the fourth section we present our fuzzy Monte Carlo method and how we will generate sequences of random fuzzy vectors. Our fuzzy Monte Carlo solution to the fuzzy queuing optimization problem is the fifth section and the last section has a summary and our conclusions. All the fuzzy numbers used in this chapter, except fuzzy profit starting in Section 16.3, will be non-negative. We will program our fuzzy Monte Carlo method in MATLAB [8]. This chapter is also based on [1].

16.2 Queuing Model

We will model the queuing system using the arrival rate λ and the service rate μ for any server. This is a common method used in queuing theory ([9],[11]). The system has c parallel and identical servers, system capacity M (in the servers and in the queue, $c \leq M$) and an infinite calling source. If the system is full, new arrivals are turned away and lost from the system. The λ rate will be state independent, which means that λ does not depend on how many customers are in the system. But if there are n customers in the system, then the rate of departure from the whole system is $\mu_n = n\mu$, for $0 \leq n < c$ and $\mu_n = c\mu$ for $c \leq n \leq M$.

A basic assumption is that we are in steady-state, all transient behavior has died down and can be neglected, and the time interval δ is sufficiently small so that the probability of two or more events occurring during δ is zero. If we are in state n, or there are n customers in the system with $0 < n < M$, we can have only two events occurring: (1) one customer arrives and we have $n+1$ in the system; or (2) one customer finishes service and leaves and we have $n-1$ left in the system. Usually for steady-state we assume that $\lambda \leq \mu$ when we have infinite capacity. However, since we have finite system capacity we do not need to assume that $\lambda \leq \mu$. If we are in state zero ($n=0$), we can only go to $n=1$ and we can get to state $n=0$ from $n=1$ when a customer leaves service. We can get to state $n=M$ only from $n=M-1$ with an arrival and we can leave state $n=M$ to state $M-1$ when a customer leaves service.

The first objective is to compute the steady-state probabilities w_i, $0 \leq i \leq M$, from which we may determine various measures of system performance.

Using a transition rate diagram, the expected rate of flow into state n is

$$\lambda w_{n-1} + \mu_{n+1} w_{n+1}, \tag{16.1}$$

and the expected rate of flow out of state n is

$$\lambda w_n + \mu_n w_n, \tag{16.2}$$

for $0 < n < M$. We set these two equal to get the balance equation

$$\lambda w_{n-1} + \mu_{n+1} w_{n+1} = \lambda w_n + \mu_n w_n, \tag{16.3}$$

for $0 < n < M$. The balance equation for state $n=0$ is simply $\lambda w_0 = \mu_1 w_1$ and for $n=M$ it is $\lambda w_{M-1} = \mu_M w_M$. We solve these balance equations for w_i, $1 \leq i \leq M$, functions of w_0 and then use the fact that the sum of all the w_i must equal one to obtain a formula for w_0. The final result is that $w_i = F_i(\lambda, \mu, c, M)$, $0 \leq i \leq M$. That is, the steady-state probabilities are function of λ, μ, c and M ([9],[11]).

Now we can determine measures of system performance such as U=server utilization and N=expected number of customers in the system. All we will need in this chapter is

$$N = \sum_{k=0}^{M} k w_k. \tag{16.4}$$

We next wish to consider the optimal queuing network maximizing profit with variables λ, μ, c and M.

There are many types of servers each with associated service rate $\mu \in [0, 10]$. Let $C = K_1 \mu$ be the cost, in \$ per unit time, of operating a server having corresponding service rate μ for some constant $K_1 > 0$. Determining the cost of a server per unit time is a difficult number to estimate so we later assign a fuzzy number to its value.

There is a cost involved in maintaining the queue, or those in the system but not yet in a server. Let Q be the cost, in \$ per unit time, of having one space available in the queue. So the queue cost is $Q(M-c)$. Q is also difficult to estimate exactly so we will later model it as fuzzy.

There will be certain fixed costs associated with maintaining the web site which do not depend on the decision variables. Since they do not depend on M,c,λ and μ they can be omitted for the model.

We will assume that we can affect the arrival rate λ through advertising. Let $A = K_2\lambda$ be the advertising level in \$ per unit time that is expected to produce arrival rate λ for constant $K_2 > 0$. This cost A will be hard to know precisely so it too will become fuzzy. The web site pays for these advertisements but other advertisers will pay the web site, to place their ads, depending on the number of customers in the queue who can see their (pop-up) adds.

Revenue from advertisers is assumed to be proportional to the average number of customers in the system N. If T is the revenue, in \$ per unit time, per customer in the system, then total revenue per unit time is TN.

Profit per unit time, to be maximized, is

$$Profit = TN - [K_2\lambda + Q(M-c) + K_1\mu\ c], \tag{16.5}$$

We will next model all the cost/income parameters as fuzzy numbers, and if any are known exactly, then we would use their exact values. The variables are $c = 1, 2, .., 10$ and $M = c, c+1, ..., 30$, $\lambda \in [0, 10]$ and $\mu \in [0, 10]$.

We could consider a budget constraint, only so much money available per unit time, but we will not do this here. The above profit equation gives us only one goal, maximize profit. We could add other goals [4] such as maximize server utilization, minimize number of lost customers due to system capacity M, etc.

There are many other costs associated with the system, such as startup costs and operating costs ([9], Chapter 5), which we have not incorporated into the model. Many of these costs are independent from our variables, so can be classified as fixed costs in our model, and hence omitted.

16.3 Fuzzy Queuing Model

The arrival rate would need to be estimated and we will use a fuzzy estimator $\overline{\lambda}$ ([4], Chapter 3). Assume that we gather data to estimate the arrival rate. Then we can construct $(1-\gamma)100\%$ confidence intervals for λ. If we place these confidence intervals one on top of another, $0.001 \leq \gamma \leq 1$, we obtain a fuzzy number $\overline{\lambda}$. We can easily change the MATLAB program to use a crisp value for λ but in this chapter we will use a fuzzy $\overline{\lambda}$. Also, the service rate has to be estimated so we have a fuzzy estimator $\overline{\mu}$ ([4], Chapter 3). We get $\overline{\mu}$ as described above from the $(1-\gamma)100\%$ confidence intervals, $0.001 \leq \gamma \leq 1$. The fuzzy numbers obtained from the confidence interval method will be triangular shaped fuzzy numbers but in this chapter we will use triangular fuzzy numbers for $\overline{\lambda}$ and $\overline{\mu}$. The server cost automatically becomes fuzzy $\overline{C} = K_1\overline{\mu}$ for crisp constant $K_1 > 0$. Also, the advertising cost is fuzzy $\overline{A} = K_2\overline{\lambda}$ for crisp $K_2 > 0$. The fuzzy queue cost is $\overline{Q} > 0$ and the fuzzy revenue from advertisers is $\overline{T} > 0$. All these crisp and fuzzy constants are given in Table 16.1. The fuzzy constants are all triangular fuzzy numbers. The constants K_1, K_2, $\overline{Q}$ and $\overline{T}$ might be obtained from expert opinion (Section 3.4 in [4]).

The fuzzy optimization problem is to find integer $c \in [1, 10]$, integer $M \in [1, 30]$ with $c \leq M$ and $\overline{\lambda}$, $\overline{\mu} \in [0, 10]$ to maximize fuzzy profit

$$\overline{\Pi} = \overline{T}\ \overline{N} - [K_2\overline{\lambda} + \overline{Q}(M-c) + K_1\overline{\mu}c]. \tag{16.6}$$

Table 16.1. Crisp/Fuzzy Parameters in the Fuzzy Optimization Problem

Constant	$\alpha = 0$ Cut
K_1	0.04
K_2	0.03
$\overline{Q}$	(0.04/0.07/0.10)
$\overline{T}$	(3.15/3.45/3.80)

Given the values of the variables the next thing to do is to get the fuzzy steady-state probabilities $\overline{w}_k$, $0 \leq k \leq M$. We will first discuss computing the crisp steady-state probabilities and then fuzzify them using the extension principle. Let $\rho = \lambda/\mu$ and [11]

$$w_k = F_k(\lambda, \mu, c, M) = \frac{\rho^k}{k!} w_0, \ 1 \leq k \leq c, \tag{16.7}$$

and

$$w_k = F_k(\lambda, \mu, c, M) = \frac{\rho^k}{c! c^{k-c}} w_0, \ c \leq k \leq M. \tag{16.8}$$

Now $w_0 = F_0(\lambda, \mu, c, M)$ where

$$w_0 = [\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c(1-(\rho/c)^{M-c+1})}{c!(1-\rho/c)}]^{-1}, \ \rho/c \neq 1, \tag{16.9}$$

and

$$w_0 = [\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c}{c!}(M-c+1)]^{-1}, \ \rho/c = 1. \tag{16.10}$$

The test $\rho/c \neq 1$ and $\rho/c = 1$ in equations (16.9) and (16.10) will be difficult to do when λ and μ are fuzzy numbers. So we combine both of these equations into one equation eliminating the two tests on ρ/c. Let $s = M - c + 1$ and do the division in equation (16.9) producing

$$w_0 = [\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c}{c! c^{s-1}} P(c, s, \rho)]^{-1}, \tag{16.11}$$

where

$$P(c, s, \rho) = c^{s-1} + c^{s-2}\rho + c^{s-3}\rho^2 + \ldots + \rho^{s-1}. \tag{16.12}$$

We will use equations (16.11) and (16.12) to determine the steady-state probability w_0. Then

$$\overline{w}_k = F_k(\overline{\lambda}, \overline{\mu}, c, M), \tag{16.13}$$

for $0 \leq k \leq M$, evaluated using the extension principle. Let $\overline{w}_k[\alpha] = [w_{k1}(\alpha), w_{k2}(\alpha)]$, $k = 0, \ldots, M$, $0 \leq \alpha \leq 1$. Then we know how to get the α-cuts of the $\overline{w}_k$ as [5]

$$w_{k1}(\alpha) = min\{F_k(\lambda, \mu, c, M) \mid \lambda \in \overline{\lambda}[\alpha], \mu \in \overline{\mu}[\alpha]\}, \tag{16.14}$$

and

$$w_{k2}(\alpha) = max\{F_k(\lambda, \mu, c, M) \mid \lambda \in \overline{\lambda}[\alpha], \mu \in \overline{\mu}[\alpha]\}, \tag{16.15}$$

for all k and α. Let us change equations (16.14) and (16.15) to F_k a function of ρ, c and M. Let $\overline{\lambda}[\alpha] = [\lambda_1(\alpha), \lambda_2(\alpha)]$ and $\overline{\mu}[\alpha] = [\mu_1(\alpha), \mu_2(\alpha)]$. Then $\overline{\rho} = \overline{\lambda}/\overline{\mu}$ so that $\overline{\rho}[\alpha] = [\lambda_1(\alpha)/\mu_2(\alpha), \lambda_2(\alpha)/\mu_1(\alpha)]$. Then

$$w_{k1}(\alpha) = min\{F_k(\rho, c, M) \mid \rho \in \overline{\rho}[\alpha]\}, \tag{16.16}$$

and

$$w_{k2}(\alpha) = max\{F_k(\rho, c, M) \mid \rho \in \overline{\rho}[\alpha]\}, \tag{16.17}$$

for all k and α. We solve these optimization problems, equations (16.16) and (16.17), using the Optimization Toolbox in MATLAB [8].

Next we find $\overline{N}$ as

$$\overline{N}[\alpha] = \{\sum_{k=0}^{M} kw_k \mid \mathbf{S}\}, \tag{16.18}$$

all $\alpha \in [0, 1]$, where $\mathbf{S}$ is the statement "$w_k \in \overline{w}_k[\alpha], 0 \leq k \leq M, w_0 + ... + w_M = 1$". If $\overline{N}[\alpha] = [n_1(\alpha), n_2(\alpha)]$ then

$$n_1^*(\alpha) = min\{\sum_{k=0}^{M} kw_k \mid \mathbf{S}\}, \tag{16.19}$$

and

$$n_2^*(\alpha) = max\{\sum_{k=0}^{M} kw_k \mid \mathbf{S}\}. \tag{16.20}$$

Then since $n_i^*(\alpha)$, $i = 1, 2$, could exceed M we set

$$n_1(\alpha) = min\{n_1^*(\alpha), M\}, \tag{16.21}$$

and

$$n_2(\alpha) = min\{n_2^*(\alpha), M\}, \tag{16.22}$$

Equations (16.19) and (16.20) are linear programming problems which can be solved using the Optimization Toolbox in MATLAB.

Now we can compute the fuzzy profit $\overline{\Pi}$. We will use α-cuts and interval arithmetic because in this case it produces the same result as the extension principle. Let $\overline{\Pi}[\alpha] = [\pi_1(\alpha), \pi_2(\alpha)]$. Also let $\overline{T}[\alpha] = [t_1(\alpha), t_2(\alpha)]$, $\overline{N}[\alpha] = [n_1(\alpha), n_2(\alpha)]$, $\overline{\lambda}[\alpha] = [\lambda_1(\alpha), \lambda_2(\alpha)]$, $\overline{Q}[\alpha] = [q_1(\alpha), q_2(\alpha)]$, $\overline{\mu}[\alpha] = [\mu_1(\alpha), \mu_2(\alpha)]$. Then

$$\overline{\Pi}[\alpha] = [t_1(\alpha)n_1(\alpha), t_2(\alpha)n_2(\alpha)] - [s, t], \tag{16.23}$$

where

$$[s, t] = [K_2\lambda_1(\alpha) + q_1(\alpha)(M - c) + K_1\mu_1(\alpha)c, K_2\lambda_2(\alpha) + q_2(\alpha)(M - c) + K_1\mu_2(\alpha)c], \tag{16.24}$$

and

$$\overline{\Pi}[\alpha] = [t_1(\alpha)n_1(\alpha) - t, t_2(\alpha)n_2(\alpha) - s]. \tag{16.25}$$

16.4 Fuzzy Monte Carlo Method

We plan to produce (approximate) solutions to the fuzzy optimization problem in equation (16.6) using our fuzzy Monte Carlo method. We will randomly generate fuzzy vectors

$$\overline{V_k} = (\overline{\lambda}_k, \overline{\mu}_k, c_k, M_k), \tag{16.26}$$

with $\overline{\lambda}_k, \overline{\mu}_k \in [0, 10]$, integer c_k in $[1, 10]$ and integer M_k in $[c_k, 100]$, for $k = 1, 2, \ldots, P$. We evaluate fuzzy profit $\overline{\Pi}_k$ for each $\overline{V}_k$ and find the $\overline{V}_k$ to maximize $\overline{\Pi}_k$. With $P = 100,000$ we should get a good estimate of maximum fuzzy profit. So we need to do two things: (1) describe how to get random sequences of the vectors $\overline{V}_k$; and (2) how we will determine the maximum of the set $\Lambda = \{\overline{\Pi}_k \mid k = 1, 2, \ldots, P\}$. We first consider $\overline{V}_k$ and how to produce the fuzzy values for λ and μ and then separately generate c and M. Then we discuss finding the maximum of Λ.

16.4.1 Random Sequence $\overline{V}_k$

To obtain random sequences $\overline{V}_{k1} = (\overline{\lambda}_k, \overline{\mu}_k)$, $k = 1, 2, ..., P$, where the $\overline{\lambda}_k$ and $\overline{\mu}_k$ are triangular fuzzy numbers, we first randomly generate crisp vectors $v_k = (x_{k1}, ..., x_{k6})$, using our Sobol quasi-random number generator (Chapter 3), with all the x_{ki} in $[0, 1]$, $k = 1, 2, .., P$. We choose the first three numbers in v_k and order them from smallest to largest. Assume that $x_{k3} < x_{k1} < x_{k2}$. Then the first triangular fuzzy number $\overline{\lambda}_k = (x_{k3}/x_{k1}/x_{k2})$. Continue with the next three numbers in v_k making $\overline{\mu}_k$. However the $\overline{\lambda}_k$ and $\overline{\mu}_k$ we want need to be in $[0, 10]$. Since $\overline{\lambda}_k$ and $\overline{\mu}_k$ start out in $[0, 1]$ we may easily map them into $[0, 10]$ by using $10\overline{\lambda}_k$ ($10\overline{\mu}_k$) for fuzzy λ (μ).

Next we need to randomly get the sequence of integers $c_k \in [1, 10]$. Randomly generate $\nu \in [0, 1]$ and then define $\pi = 0.5 + 10\nu$ making $\pi \in [0.5, 10.5]$. Then round π off to the nearest integer producing c. We round 0.5 to one and 10.5 to 10. Use a pseudo-random number generator for ν. Finally randomly produce $\xi \in [0, 1]$ and then define $\sigma = (c - 0.5) + (31 - c)\xi$ putting $\sigma \in [c - 0.5, 30.5]$. Round σ off to the nearest integer giving M. Also use a pseudo-random number generator for ξ.

16.4.2 Maximum of Fuzzy Profit

Given a finite set of fuzzy numbers $\overline{\Pi}_1, ..., \overline{\Pi}_P$ we want to order them from smallest to largest. For a finite set of real numbers there is no problem in ordering them from smallest to largest. However, in the fuzzy case there is no universally accepted way to do this. There are probably more than 50 methods proposed in the literature of defining $\overline{U} \leq \overline{V}$, for two fuzzy numbers $\overline{U}$ and $\overline{V}$.

Here we will use only one procedure for ordering fuzzy numbers which is Buckley's Method in Section 2.6.1. We will now use $\eta = 0.9$ in Buckley's Method to help reduce the number of fuzzy profits that could be considered approximately equal for the maximum fuzzy profit. But note that different definitions of $\leq$

between fuzzy numbers can give different orderings and therefore different final answers to the fuzzy optimization problem.

Now apply this to $\overline{\Pi}_1, \ldots, \overline{\Pi}_P$. These are all triangular shaped fuzzy numbers. We want H_K the set of undominated fuzzy profits. We then present H_K, together with the corresponding values for $\overline{\lambda}$,$\overline{\mu}$, c and M, to management for their decision. It is usually better to present multiple optimal solutions then one unique optimal solution. Managers are decision makers and if you give then one optimal solution they essentially have almost no decision: accept it or reject it. However, with multiple solutions the manager can study them, bringing in new information etc., to make their final decision. However, with $P = 100,000$ H_K could be too large, like $100 - 200$ fuzzy sets. We need to restrict the size of H_K and in this chapter we decide that the maximum size of H_K will be three fuzzy sets. Whenever we have more than three fuzzy sets to be in H_K we pick the three with largest vertex points. Therefore, the maximum of Λ will be H_K.

Now we need to incorporate this into the iterations in our fuzzy Monte Carlo method. Suppose at some point in the iterations $H_K = \{\overline{\Pi}_a, \overline{\Pi}_b, \overline{\Pi}_c\}$. The next iteration produces fuzzy profit $\overline{\Pi}_0$. We then compare $\overline{\Pi}_0$ to $\overline{\Pi}_i$, for $i = a, b, c$. There are nine possible outcomes. For example $\overline{\Pi}_0 \approx \overline{\Pi}_c$, $\overline{\Pi}_0 > \overline{\Pi}_b$ and $\overline{\Pi}_0 \approx \overline{\Pi}_a$ is one possible result. Then $H_K = \{\overline{\Pi}_0, \overline{\Pi}_a, \overline{\Pi}_c\}$. Any fuzzy set which is dominated can not be in H_K and all the fuzzy sets in H_K must be equivalent ($\approx$).

16.5 Fuzzy Monte Carlo Solution

We will describe our fuzzy Monte Carlo program. The program is written in MATLAB. We first generate $\overline{V}_k$, $k = 1, \ldots, P$. We compute the fuzzy numbers using the α-cuts $\alpha = 0.00, 0.30, 0.60, 0.90, 1.00$. We have our first file

$$\mathcal{F}_1 = \{(\overline{\lambda}[\alpha], \overline{\mu}[\alpha], c_k, M_k) \mid \alpha = 0, 0.30, 0.60, 0.90, 1;\ k = 1, \ldots, P\}. \quad (16.27)$$

Using this file we determine the fuzzy steady-state probabilities $\overline{w}_{jk}$, $j = 0, \ldots, M$, $k = 1, \ldots, P$, from equations (16.16) and (16.17), using the Optimization Toolbox. This makes our second file

$$\mathcal{F}_2 = \{(\overline{w}_{0k}[\alpha], \ldots, \overline{w}_{M,k}[\alpha]) \mid \alpha = 0, 0.30, 0.60, 0.90, 1;\ k = 1, \ldots, P\ \}. \quad (16.28)$$

Using file $\mathcal{F}_2$ we determine $\overline{N}_k[\alpha]$ from equations (16.19)-(16.22) using the Optimization Toolbox. This produces file

$$\mathcal{F}_3 = \{\overline{N}_k[\alpha] \mid \alpha = 0, 0.30, 0.60, 0.90, 1;\ k = 1, \ldots, P\ \}. \quad (16.29)$$

Now we are ready to find fuzzy profit $\overline{\Pi}_k[\alpha]$, $\alpha = 0, 0.30, 0.60, 0.90, 1$, $k = 1, \ldots, P$, from equations (16.23)-(16.25).

The results of the Monte Carlo method, after $P = 100,000$ iterations, gave us the three (approximate) optimal solutions shown in Table 16.2. The graph of the three maximum fuzzy profits in the final H_K are shown in Figure 16.1. For

Table 16.2. Optimal Numerical Results from the Fuzzy Monte Carlo Method

Solution	$\overline{\lambda}$	$\overline{\mu}$	c	M	$\overline{\Pi}$
1	(4.81/5.85/6.59)	(2.40/3.28/4.00)	1	30	$\approx$ (16.83/96.75/112.54)
2	(4.52/7.16/7.46)	(3.25/4.18/4.18)	1	30	$\approx$ (−0.62/96.25/108.77)
3	(7.96/9.72/9.78)	(1.91/2.33/2.36)	2	29	$\approx$ (83.57/94.51/103.18)

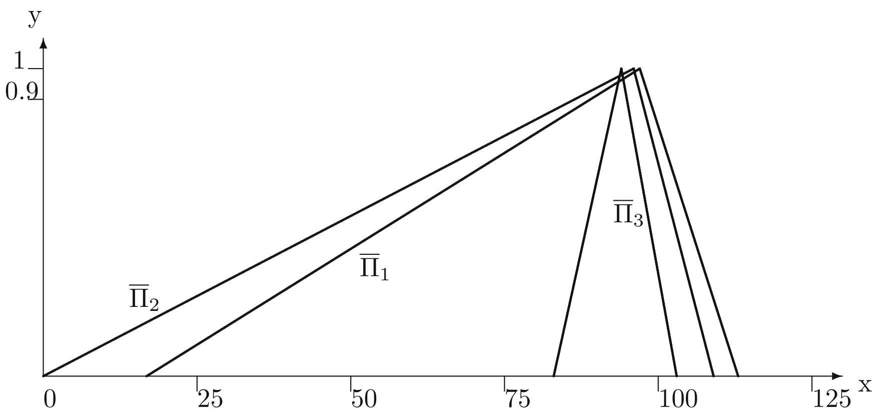


Fig. 16.1. Optimal Fuzzy Profits from the Fuzzy Monte Carlo Method

simplicity the graphs in Figure 16.1 are triangular fuzzy numbers, using only their base and vertex, where they are really triangular shaped fuzzy numbers.

Let us now compare the results above for our fuzzy Monte Carlo method to those in Chapter 14 of [4]. Both basically consider the same problem of maximizing (almost the same) fuzzy profit with variables fuzzy arrival rate $\overline{\lambda}$, fuzzy service rate $\overline{\mu}$, number of servers c and system capacity M. Because of the computational burden of calculating fuzzy profit in [4] the author only considered 16 cases. For example, $\overline{\lambda} = (4/5/6)$, $\overline{\mu} = (5/6/7)$, $c = 2$, $M = 10$ was one of the cases. The author of [4], at that time, could not look at $P = 100,000$ random cases. In [4] $H_K = \{\overline{\Pi}_a, \overline{\Pi}_b, \overline{\Pi}_c\}$ where $\overline{\Pi}_a \approx (-0.8/0.7/4.6)$, $\overline{\Pi}_b \approx (-0.5/1.9/6.3)$ and $\overline{\Pi}_c \approx (-0.8/0.9/4.3)$. The graphs of these fuzzy profits, as triangular fuzzy numbers, are shown in Figure 16.2 for comparison to Figure 16.1. Now let us briefly summarize the model in Chapter 14 of [4] to compare to the model in this chapter.

The fuzzy profit function in [4] was

$$\overline{\Pi} = \overline{TN} - [\overline{A}_u + \overline{Q}(M-c) + \overline{C}_v c]. \tag{16.30}$$

In [4] the author considered only two fuzzy arrival rates $\overline{\lambda}_i$ and only two fuzzy service rates $\overline{\mu}_i$, $i = 1, 2$, in their 16 cases. Then $\overline{A}_u = \overline{A}_i$ when $\overline{\lambda} = \overline{\lambda}_i$, $i = 1, 2$. In this chapter we used $K_2\overline{\lambda}$ since we could not use $P = 100,000$ different constants for that many different fuzzy arrival rates. Also, $\overline{C}_v = \overline{C}_i$ when $\overline{\mu} = \overline{\mu}_i$, $i = 1, 2$.

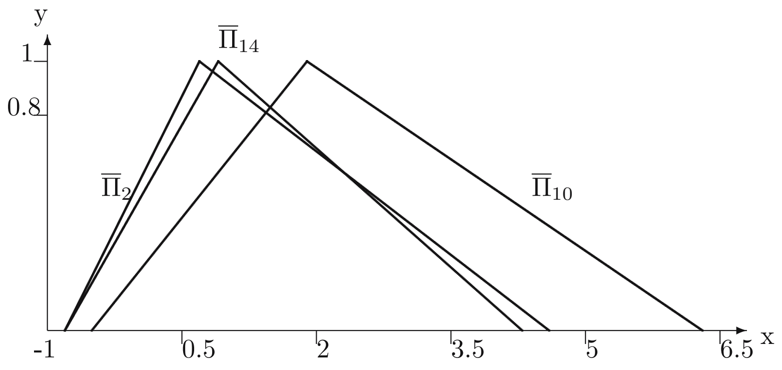


Fig. 16.2. Optimal Fuzzy Profits from Chapter 14 of [4]

In this chapter we used $K_1\overline{\mu}c$ since we could not use $P = 100,000$ different constants for that many different fuzzy service rates. The $\overline{Q}(M-c)$ is the same in both models. In this chapter our $\overline{T}$ is greater then the fuzzy T value used in [4] because the $\overline{\Pi} \approx (\pi_1/\pi_2/\pi_3)$ in [4] had too many results with negative π_2 values. Finally, in [4] they used $\eta = 0.8$ in the comparison of fuzzy numbers and in this chapter we used $\eta = 0.9$ to help in reducing the size of the set H_K.

We see from Figures 16.1 and 16.2 that the results are similar: three fuzzy sets clustered together approximating maximum fuzzy profit. However, those in Figure 16.1 would present a better approximation since they are the result of 100,000 random choices for the variables. The fuzzy profits in Figure 16.1 will all lie to the right of those in Figure 16.2 because $\overline{T}$ used in this chapter is approximately three plus the $\overline{T}$ used in Chapter 14 of [4].

16.6 Summary and Conclusions

In this chapter we introduced our new fuzzy Monte Carlo procedure. The basic requirement of any fuzzy Monte Carlo method is to be able to randomly produce fuzzy/crisp vectors to uniformly fill the search space. We suggested using a quasi-random number generator to make these random fuzzy/crisp vectors. Theoretically, given enough iterations of the fuzzy Monte Carlo technique, it will produce a very good approximate solutions to the fuzzy optimization problem.

We applied our fuzzy Monte Carlo method to a fuzzy optimization problem from fuzzy queuing theory. For all our Monte Carlo calculations, we used a Dell Optiplex GX 250 with a dual core and a 64-bit pentium D 2.8 GHz processor running on Windows XP. The computer time for 100,000 iterations was approximately 52.5 hours. To construct file $\mathcal{F}_2$ (equation (16.29)) using the Optimization Toolbox the computer time was approximately 47 hours and then to finish the program it was approximately 5.5 hours.

It would be nice to try 1,000,000 iterations, but the computing time on one office PC would be too excessive. However, if we could run the fuzzy Monte Carlo program 100, 000 iterations simultaneously on ten separate machines, computing over the weekend, and then combine the results, we could go to 1, 000, 000 iterations. The MATLAB program is available from the authors.

References

1. Abdalla, A., Buckley, J.J.: Monte Carlo Methods in Fuzzy Queuing Theory (under review)
2. Buckley, J.J.: Elementary Queuing Theory Based on Possibility Theory. Fuzzy Sets and Systems 37, 43–52 (1990)
3. Buckley, J.J.: Fuzzy Probabilities: New Approach and Applications. Springer, Heidelberg (2003)
4. Buckley, J.J.: Fuzzy Probabilities and Fuzzy Sets for Web Planning. Springer, Heidelberg (2004)
5. Buckley, J.J., Qu, Y.: On Using α-cuts to Evaluate Fuzzy Equations. Fuzzy Sets and Systems 38, 309–312 (1990)
6. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
7. Buckley, J.J., Feuring, T., Hayashi, Y.: Fuzzy Queuing Theory Revisited. Int. J. Uncertainty, Fuzziness and Knowledge Based Systems 9, 527–538 (2001)
8. MATLAB, The MathWorks, `http://www.mathworks.com`
9. Menasce, D.A., Almeida, V.A.F.: Capacity Planning for Web Performance. Prentice Hall, Upper Saddle River, N.J. (1998)
10. Pardo, M.J., de la Fuente, D.: Optimizing a Priority-Discipline Queueing Model Using Fuzzy Set Theory. Computers & Math. with Applications 54, 267–281 (2007)
11. Taha, H.A.: Operations Research, 5th edn. Macmillan, N.Y. (1992)

Part III: Unfinished Business

17 Fuzzy Min-Cost Capacitated Network

17.1 Introduction

The min-cost capacitated flow network problem generalizes the transportation, transshipment and max-flow problems [4]. This chapter is based on ([1]and Section 6.5.2 of [2]). A recent reference in this area is [3]. In the next section we first discuss the crisp problem and then fuzzify it. We have previously found an approximate fuzzy solution to an example problem using an evolutionary algorithm. Then in Section 17.3 we plan to apply our fuzzy Monte Carlo method to obtain another approximate solution to this example problem and then compare both solutions.

17.2 Min-Cost Capacitated Network

There are n nodes, some of which are "sources" which accept input, and some others are "sinks" which produce output. Let I_i be the input to node i and set $I_i = 0$ if node i is not a source node. Let O_i be the output from node i and set $O_i = 0$ if node i is not a sink node.

We assume that all nodes are connected by an arc which are now all one-way streets. All arcs have a unit cost c_{ij} associated with them. We set $c_{ii} = 0$ all i and if there is no arc from node i to node j we make $c_{ij} = M$, M a large positive number, so that the optimal solution to the minimization problem will not choose an arc with $c_{ij} = M$.

Let x_{ij} be the flow from node i to node j. For example, x_{ij} may be gallons of oil. The x_{ij} flow along the arc from i to j at unit cost c_{ij}. Also, all arcs have a minimum capacity L_{ij} and a maximum capacity U_{ij}. Now an L_{ij} can be zero (no minimum capacity) and a U_{ij} can be infinite (no maximum capacity). If there is no maximum capacity we let $U_{ij} = M$.

So, the optimization problem is to minimize the total cost

$$minZ = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}, \tag{17.1}$$

subject to capacity constraints

$$L_{ij} \leq x_{ij} \leq U_{ij}, \tag{17.2}$$

all i, j, plus the conservation of flow constraints

$$I_i + \sum_{k=1}^{n} x_{ki} = \sum_{k=1}^{n} x_{ik} + O_i, \tag{17.3}$$

for all nodes $i = 1, 2, \cdots, n$. The conservation of flow constraint says that the total flow into a node must equal to the total flow out of that node. For the problem to be feasible we need $\sum_{i=1}^{n} I_i = \sum_{i=1}^{n} O_i$.

The complete fuzzification has all the $\overline{C}_{ij}$, $\overline{L}_{ij}$, $\overline{U}_{ij}$, $\overline{I}_i$ and $\overline{O}_i$ triangular fuzzy numbers. The fuzzy flows $\overline{X}_{ij}$ will also be triangular fuzzy numbers. The fuzzy optimization problem is

$$min \overline{Z} = \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{C}_{ij} \overline{X}_{ij}, \tag{17.4}$$

subject to

$$\overline{L}_{ij} \leq \overline{X}_{ij} \leq \overline{U}_{ij}, \tag{17.5}$$

for all i, j and

$$\overline{I}_i + \sum_{k=1}^{n} \overline{X}_{ki} \approx \sum_{k=1}^{n} \overline{X}_{ik} + \overline{O}_i, \tag{17.6}$$

for $1 \leq i \leq n$. In equation (17.5) we have some special cases: (1) if $\overline{L}_{ij} = 0$, then it is $0 \leq \overline{X}_{ij} \leq \overline{U}_{ij}$; (2) if $\overline{U}_{ij} = M$, then it is $\overline{L}_{ij} \leq \overline{X}_{ij}$; and (3) if $\overline{L}_{ij} = 0$ and $\overline{U}_{ij} = M$, then it is $0 \leq \overline{X}_{ij}$. In equation (17.6) we cannot, because of fuzzy arithmetic, demand exact equality as in the crisp case, so we use $\approx$. We also assume that $\sum_{i=1}^{n} \overline{I}_i \approx \sum_{i=1}^{n} \overline{O}_i$. We will be using Buckley's Method (Section 2.6.1 in Chapter 2) to evaluate $\leq$, $<$ and $\approx$ between fuzzy numbers because that is what we used in our original research on this topic. We will not allow negative flows through the network, so we add the extra constraint that $\overline{X}_{ij} \geq 0$.

$\overline{Z}$ will be a triangular shaped fuzzy number and in our previous discussions of this topic we handled min $\overline{Z}$ as discussed in Section 2.5 of Chapter 2, subject to the constraints given in equation (17.5) and (17.6).

Example 17.2.1

Our example is shown in Figure 17.1. For all arcs not shown we set $\overline{C}_{ij} = M$. If no pair $(\overline{L}_{ij}, \overline{U}_{ij})$ is specified for an arc it means that it is equal to $(0, M)$, or no restrictions. Nodes 1 and 2 are source nodes and the sink nodes are 4 and 5. The values of the fuzzy numbers are: (1) $\overline{I}_1 = (35/40/45)$ and $\overline{I}_2 = (45/50/55)$; (2) $\overline{O}_4 = (25/30/35)$ and $\overline{O}_5 = (55/60/65)$: (3) $\overline{C}_{12} = (1/3/5)$, $\overline{C}_{13} = (6/7/8)$, $\overline{C}_{14} = (4/5/6)$, $\overline{C}_{23} = (1/2/3)$, $\overline{C}_{25} = (0.5/1/2)$, $\overline{C}_{35} = (6/7/8)$, $\overline{C}_{45} = (2/4/6)$;

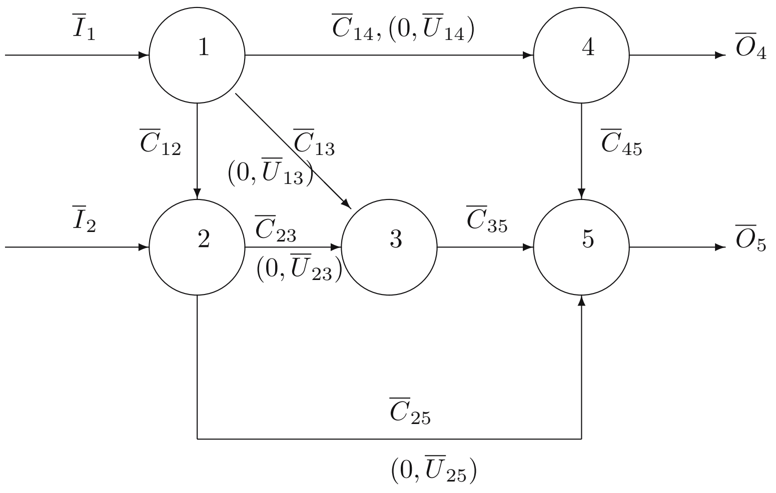


Fig. 17.1. Fuzzy Min-Cost Capacitated Flow Problem in Example 17.2.1

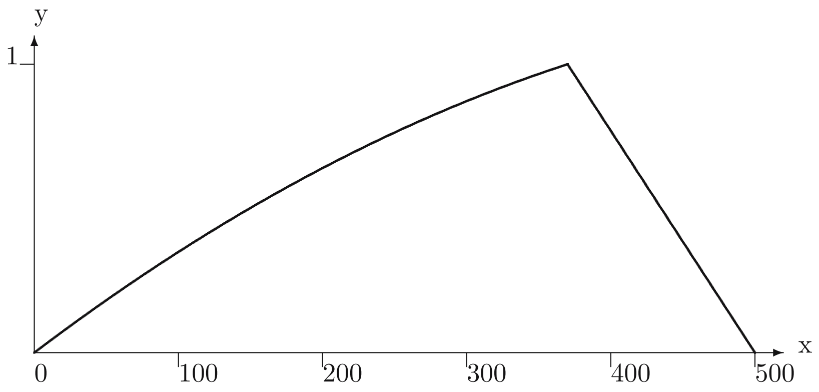


Fig. 17.2. Evolutionary Algorithm Min Fuzzy Cost in Example 17.2.1

and (4) $\overline{U}_{13} = (8/10/11)$, $\overline{U}_{14} = (30/35/40)$, $\overline{U}_{23} = (50/60/70)$, and $\overline{U}_{25} = (28/30/34)$. Using the vertex values for all the fuzzy numbers the crisp optimal solution is $x_{12} = 5$, $x_{13} = 0$, $x_{14} = 35$, $x_{23} = 25$, $x_{25} = 30$, $x_{35} = 25$ and $x_{45} = 5$ with minimum cost of 490. In equation (2.42) of Chapter 2 we used $\lambda_1 = 0.4 = \lambda_2$ and $\lambda_3 = 0.2$ since the decision maker was more interested in maximizing A_1 and minimizing u_2 than in minimizing A_2. The evolutionary algorithm solution for $\overline{Z}$ is shown in Figure 17.2 with the corresponding values for the $\overline{X}_{ij}$ in Table 17.1.

The evolutionary algorithm is not fast because it spends a lot of time checking the constraints. For much larger n we would need to construct special algorithms just for checking the constraints.

Table 17.1. Best Fuzzy Flows in Example 17.2.1

	Evolutionary Algorithm	Fuzzy Monte Carlo
$\overline{X}_{12}$	(0.0041/4.8012/4.8168)	TBC
$\overline{X}_{13}$	(0.0035/0.0052/0.0112)	TBC
$\overline{X}_{14}$	(0.0020/34.1925/34.1955)	TBC
$\overline{X}_{23}$	(0.0000/13.4586/13.4641)	TBC
$\overline{X}_{25}$	(0.0034/38.4981/42.9179)	TBC
$\overline{X}_{35}$	(0.0003/13.1353/13.1410)	TBC
$\overline{X}_{45}$	(0.0000/6.4814/6.4830)	TBC

17.3 Fuzzy Monte Carlo Method

Now obtain another approximate solution to Example 17.2.1 using the fuzzy Monte Carlo method. Generate sequences of random fuzzy vectors $\overline{V}_i = (\overline{X}_{i1}, ..., \overline{X}_{i7})$, $i = 1, 2, 3, ...$ and set $\overline{X}_{12} = \overline{X}_{i1}$,...,$\overline{X}_{45} = \overline{X}_{i7}$ all triangular fuzzy numbers. We get these random fuzzy vectors using our Sobol quasi-random number generator as discussed in Section 4.5.

Now we need to find intervals $T_j = [0, M_j]$, $j = 1, 2, ..., 7$, as explained in previous chapters, for the $\overline{X}_{ij}$. We randomly generate $\overline{X}_{ij} \in [0, M_j]$, $j = 1, ..., 7$, and form the random fuzzy vector $\overline{V}_i = (\overline{X}_{i1}, ..., \overline{X}_{i7})$. We test to see if $\overline{V}_i$ is feasible, or the $\overline{X}_{ij}$ satisfy the constraints. Assuming that $\overline{V}_i$ is feasible we compute the value of the objective function $\overline{Z} = \overline{Z}_0$. If $\overline{Z}^*$ is the current best (min) value of $\overline{Z}$ then we replace $\overline{Z}^*$ with $\overline{Z}_0$ if $\overline{Z}^* > \overline{Z}_0$, otherwise we discard $\overline{Z}_0$. The determination of these intervals is very important. If they are too big too many $\overline{V}_i$ can be rejected as not feasible, and if they are too small we could miss a "good" solution. After studying the crisp optimal solution, the evolutionary algorithm results in Table 17.1 and the constraints $\overline{X}_{ij} \le \overline{U}_{ij}$ we came up with $T_1 = [0, 10]$, $T_2 = [0, 11]$, $T_3 = [0, 40]$, $T_4 = [0, 70]$, $T_5 = [0, 45]$, $T_6 = [0, 30]$ and $T_7 = [0, 10]$. Changing these intervals can effect the solution. Also we are using Buckley's Method (Section 2.6.1 of Chapter 2) to evaluate $\le$, $>$ and $\approx$ between fuzzy numbers with tie breakers as discussed in Section 2.6.4.

The results of the fuzzy Monte Carlo method, after N iterations, would also be presented in Table 17.1 where TBC represents "to be completed". The graph of the best (minimum) $\overline{Z}$ would be shown in another figure. Then we compare the results.

References

1. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems: Operations Research. J. Advanced Computational Intelligence 3, 171–176 (1999)
2. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
3. Ghatee, M., Hashemi, S.M.: Ranking Function-Based Solutions of Fully Fuzzified Minimal Cost Flow Problems. Information Sciences 177, 4271–4294 (2007)
4. Taha, H.A.: Operations Research, 5th edn. MacMillan, N.Y. (1992)

18 Fuzzy Shortest Path Problem

18.1 Introduction

In this chapter we consider the shortest route problem where distances/costs are not known precisely and are modeled using fuzzy numbers. The fuzzy shortest route problem is outlined in the next section. We have previously used an evolutionary algorithm to solve an example problem (Section 6.5.1 of [2] and [3]). In Section 18.3 we plan to apply our fuzzy Monte Carlo method to obtain a solution to this example problem and then compare both solution methods.

18.2 Fuzzy Shortest Path Problem

Consider a network having n nodes linked by arcs. We allow all nodes to be connected by an arc. All arcs can be two-way streets (can travel in both directions). Define $\overline{a}_{ij}$ to be the fuzzy distance (or cost) of traveling from node i to node j, $1 \leq i, j \leq n$. Set $\overline{a}_{ii} = 0$ for $1 \leq i \leq n$. If there is no arc/street connecting node i to node j we set $\overline{a}_{ij} = M$, M is a large positive number. Since this is a minimization problem the optimal solution will not select a path from i to j if $\overline{a}_{ij} = M$. We wish to find paths from node 1 to node n with smallest total fuzzy distance/cost.

For a survey of the literature on the fuzzy shortest path problem see the papers/books ([1],[4]-[14]) and the references in these papers/books. Our solution concept using an evolutionary algorithm, or fuzzy Monte Carlo, is different from those used by other researchers.

Define a path of length $K+1$, $1 \leq K \leq n-1$, to be $(1, i_1, \cdots, i_K, n)$ where i_1, i_2, $\cdots$, i_K are distinct numbers in the set $\{2, \cdots, n-1\}$. A path of length 1 is $(1, n)$ and a path of length 2 is $(1, i_1, n)$ for $2 \leq i_1 \leq n-1$. Let Ω be all paths from 1 to n. If $\Gamma \in \Omega$, then define $\overline{D}(\Gamma) = \overline{a}_{1,i_1} + \overline{a}_{i_1,i_2} + \cdots + \overline{a}_{i_K,n}$. If all the $\overline{a}_{ij}$ are triangular fuzzy numbers, or real numbers in $[0, M]$, then $\overline{D}(\Gamma)$ is also a triangular fuzzy number. We will be using Buckley's Method (Section 2.6.1 of Chapter 2), with $\eta = 0.8$, to evaluate $\leq$, $<$ and $\approx$ between fuzzy numbers since that is what we used in our previous research on this topic. We wish to find paths Γ with minimum $\overline{D}(\Gamma)$. We will employ the "tie" breaking strategy in Section 2.6.4 so we expect a unique solution.

18.3 Monte Carlo Method

Now we want to apply our fuzzy Monte Carlo method to obtain solutions to this problem. First randomly choose $K \in \{1, 2, ..., n-2\}$ and using this K randomly choose, without replacement, K numbers in $\{2, 3, ..., n-1\}$. Let the first number chosen be i_1, the second i_2,..., and the last i_K. Then the path Γ is $1 \rightarrow i_1 \rightarrow i_2 \rightarrow ... \rightarrow i_K \rightarrow n$. In this application we used a pseudo-random number generator. Compute $\overline{D}(\Gamma)$. At some point in the simulation let $\overline{D}^*(\Gamma)$ be the smallest fuzzy distance from node 1 to node n. Let the next random path produce $\overline{D}_0(\Gamma)$. If $\overline{D}_0(\Gamma) < \overline{D}^*(\Gamma)$, then replace $\overline{D}^*(\Gamma)$ with $\overline{D}_0(\Gamma)$, otherwise discard $\overline{D}_0(\Gamma)$. Since we are using a method of breaking "ties", we do not expect to get $\overline{D}_0(\Gamma) \approx \overline{D}^*(\Gamma)$.

Apply this algorithm to a network with $n = 6$ nodes and fuzzy distances/costs given in Table 18.1. All the fuzzy distances/costs are triangular fuzzy numbers and we will use $M = 10,000$. So $\overline{D}(\Gamma)$ will also be a triangular fuzzy number. If we use all the vertices of these triangular fuzzy numbers we have a crisp network and its solution for the shortest distance from node 1 to node 6 is 18 using path $1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$.

In this application we do not need to generate random fuzzy numbers, only random paths. The results of our Monte Carlo study, after N iterations, would

Table 18.1. Fuzzy Distances Between the Nodes in Example Problem

	1	2	3	4	5	6
1	0	(1/3/4)	(4/6/8)	(11/13/16)	(13/15/18)	M
2	(2/4/5)	0	(3/5/7)	(6/8/9)	(10/12/15)	(14/19/23)
3	(1/3/4)	(3/4/6)	0	(3/4/6)	M	(11/14/18)
4	(9/11/14)	(5/8/10)	(5/7/10)	0	(1/2/3)	(7/9/12)
5	(11/15/18)	(8/10/13)	M	(5/6/7)	(0	(5/6/8)
6	M	(16/20/26)	(8/10/13)	(7/9/11)	(2/3/4)	0

Table 18.2. Fuzzy Shortest Routes from Our Evolutionary Algorithm

Path	Fuzzy Distance
1–3–4–5–6	(13/18/25)
1–2–4–5–6	(13/19/24)
1–3–4–6	(14/19/26)
1–2–4–6	(14/20/25)
1–3–6	(15/20/26)
1–2–3–4–5–6	(13/20/28)
1–5–6	(18/21/26)

be: (1) shortest fuzzy distance from node 1 to node 6 is $\overline{D}(\Gamma) = TBC$; and (2) the path is $\Gamma = 1 \rightarrow TBC \rightarrow n$ where TBC means "to be completed".

The results of the evolutionary algorithm applied to the same problem are given in Table 18.2. We obtained seven "solutions" because we did not use any "tie breakers".

References

1. Boulmakoul, A.: Generalized Path-Finding Algorithms on Semirings and the Fuzzy Shortest Path Problem. J. Computational and Applied Mathematics 162, 263–272 (2004)
2. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
3. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems: Operations Research. J. Advanced Computational Intelligence 3, 171–176 (1999)
4. Chuang, T.-N., Kung, J.-Y.: The Fuzzy Shortest Path Length and the Corresponding Shortest Path in a Network. Computers and Operations Research 32, 1409–1428 (2005)
5. Cornelis, C., DeKesel, P., Kerre, E.E.: Shortest Paths in Fuzzy Weighted Graphs. Int. J. Intelligent Systems 19, 1051–1068 (2004)
6. Hernandes, F., Lamata, M.T., Verdegay, J.L., Yamakami, A.: The Shortest Path Problem on Networks with Fuzzy Parameters. Fuzzy Sets and Systems 158, 1561–1570 (2007)
7. Itoh, T., Ishii, H.: A Model of Fuzzy Shortest Path by Possibility Measure. Japanese J. of Fuzzy Theory and Systems 8, 977–990 (1996)
8. Ji, X., Iwamura, K., Shao, Z.: New Models for Shortest Path Problem with Fuzzy Arc Lengths. Applied Math. Modeling 31, 259–269 (2007)
9. Lin, K.-C., Chern, M.-S.: The Fuzzy Shortest Path Problem and Its Most Vital Arcs. Fuzzy Sets and Systems 58, 343–353 (1993)
10. Malik, D.S., Mordeson, J.N.: Fuzzy Discrete Structures. Springer, Heidelberg (2000)
11. Okada, S.: Fuzzy Shortest Path Problems Incorporating Interactivity Among Paths. Fuzzy Sets and Systems 142, 335–357 (2004)
12. Okada, S., Soper, T.: A Shortest Path Problem on a Network with Fuzzy Arc Lengths. Fuzzy Sets and Systems 109, 129–140 (2000)
13. Shih, H.-S., Lee, E.S.: Fuzzy Multi-Level Minimum Cost Flow Problem. Fuzzy Sets and Systems 107, 159–176 (1999)
14. Yao, J.-S., Lin, F.-T.: Fuzzy Shortest-Path Network Problems with Uncertain Edge Weights. J. Information Science and Engineering 19, 329–351 (2003)

19 Fuzzy Max-Flow Problem

19.1 Introduction

If we take the "Fuzzy Min-Cost Capacitated Network" problem in Chapter 17 and delete the costs we get the "Fuzzy Max-Flow Problem" in this chapter. This is exactly what we will do. The crisp problem is outlined in the next section. Then in Section 19.3 we fuzzify the problem and plan to apply our fuzzy Monte Carlo method to get approximate solutions.

There have been only a few papers/articles in books, on the fuzzy max-flow problem ([1]-[6]).

19.2 Max-Flow Problem

This model is discussed in most operations research/management science books [7]. There are n nodes, some of which are "sources" which accept input, and some others are "sinks" which produce output. Let I_i be the input to node i and set $I_i = 0$ if node i is not a source node. Let O_i be the output from node i and set $O_i = 0$ if node i is not a sink node. The $I_i \neq 0$ represent the supplies and the $O_i \neq 0$ are the demands.

Some of the nodes are connected by an arc which are all one-way streets. If nodes i and j are connected by an arc then let x_{ij} be the flow from node i to node j. For example, x_{ij} may be gallons of oil per day. If there is no arc going from i to j then set $x_{ij} = 0$. Also, all arcs have a maximum capacity U_{ij}. Now a U_{ij} can be infinite (no real maximum capacity) and we may set this $U_{ij} = M$ a very large positive number.

So, the optimization problem is to maximize the total flow through the network satisfying (if possible) all demands

$$max\ Z = \sum_{i=1}^{n} O_i, \tag{19.1}$$

subject to capacity constraints

$$0 \leq x_{ij} \leq U_{ij}, \tag{19.2}$$

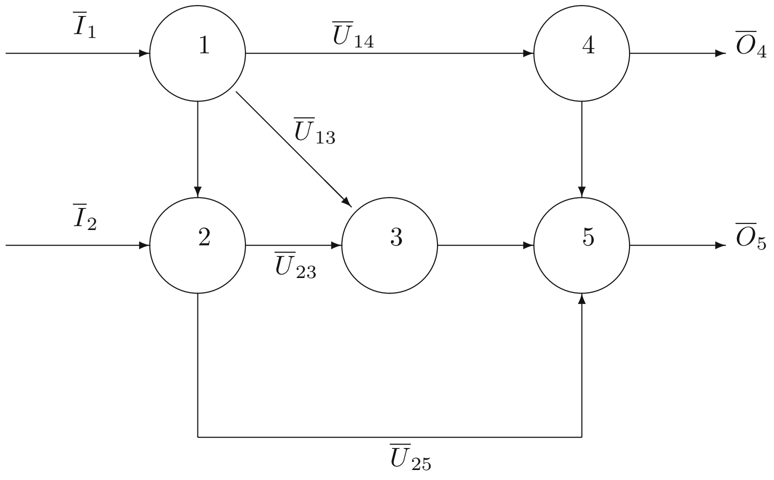


Fig. 19.1. Fuzzy Max-Flow Problem

all i, j, plus the conservation of flow constraints

$$I_i + \sum_{k=1}^{n} x_{ki} = \sum_{k=1}^{n} x_{ik} + O_i, \tag{19.3}$$

for all nodes $i = 1, 2, \cdots, n$. The conservation of flow constraint says that the total flow into a node must equal to the total flow out of that node. For the problem to be feasible we need $\sum_{i=1}^{n} I_i = \sum_{i=1}^{n} O_i$.

An example is shown in Figure 19.1 which we shall use below. Notice that: (1) nodes #1 and #2 are input/source nodes; (2) nodes #4 and #5 are output/sink nodes; (3) arcs $1 \to 4$, $1 \to 3$, $2 \to 3$ and $2 \to 5$ have finite capacities; and (4) arcs $1 \to 2$, $3 \to 5$ and $4 \to 5$ have infinite capacities.

19.3 Fuzzy Max-Flow Problem

The complete fuzzification has all the $\overline{U}_{ij}$, $\overline{I}_i$ and $\overline{O}_i$ triangular fuzzy numbers. The fuzzy flows $\overline{X}_{ij}$ will be triangular fuzzy numbers. The fuzzy optimization problem is

$$max\ \overline{Z} = \sum_{i=1}^{n} \overline{O}_i, \tag{19.4}$$

subject to

$$0 \leq \overline{X}_{ij} \leq \overline{U}_{ij}, \tag{19.5}$$

for all i, j and

$$\overline{I}_i + \sum_{k=1}^{n} \overline{X}_{ki} \approx \sum_{k=1}^{n} \overline{X}_{ik} + \overline{O}_i, \tag{19.6}$$

for $1 \leq i \leq n$. In equation (19.5) we have a special case: if $\overline{U}_{ij} = M$, then it is $0 \leq \overline{X}_{ij}$. In equation (19.6) we cannot, because of fuzzy arithmetic, demand exact equality as in the crisp case, so we use $\approx$. We also assume that $\sum_{i=1}^{n} \overline{I}_i \approx \sum_{i=1}^{n} \overline{O}_i$. We will be using Buckley's Method (Section 2.6.1 in Chapter 2) to evaluate $\leq$, $<$ and $\approx$ between fuzzy numbers because that is what we used in Chapter 17.

19.3.1 Fuzzy Monte Carlo Solution

The example we will investigate is shown in Figure 19.1. The fuzzy constants, all triangular fuzzy numbers, are given in Table 19.1. The demands are $\overline{O}_4 = (25/30/35)$ and $\overline{O}_5 = (55/60/65)$. So $\overline{I}_1 + \overline{I}_2 = (80/90/100) = (80/90/100) = \overline{O}_4 + \overline{O}_5$ and the problem is feasible.

Table 19.1. Fuzzy Parameters in the Monte Carlo Study

Constant	Fuzzy Value
$\overline{U}_{13}$	(8/10/11)
$\overline{U}_{14}$	(30/35/40)
$\overline{U}_{23}$	(50/60/70)
$\overline{U}_{25}$	(28/30/34)
$\overline{I}_1$	(35/40/45)
$\overline{I}_2$	(45/50/55)

The fuzzy optimization problem is

$$max\ \overline{Z} = \overline{O}_4 + \overline{O}_5, \tag{19.7}$$

subject to: (1) $\overline{X}_{13} \leq \overline{U}_{13}$, $\overline{X}_{14} \leq \overline{U}_{14}$, $\overline{X}_{23} \leq \overline{U}_{23}$, $\overline{X}_{25} \leq \overline{U}_{25}$; (2) $\overline{I}_1 \approx \overline{X}_{14} + \overline{X}_{13} + \overline{X}_{12}$, $\overline{I}_2 + \overline{X}_{12} \approx \overline{X}_{23} + \overline{X}_{25}$, $\overline{X}_{13} + \overline{X}_{23} \approx \overline{X}_{35}$, $\overline{X}_{14} \approx \overline{X}_{45} + \overline{O}_4$ and $\overline{X}_{25} + \overline{X}_{35} + \overline{X}_{45} \approx \overline{O}_5$; and (3) $\overline{X}_{ij} \geq 0$ all i, j.

Using the vertex values of all the fuzzy parameters the crisp optimal solution is $x_{12} = 5$, $x_{13} = 0$, $x_{14} = 35$, $x_{23} = 25$, $x_{25} = 30$, $x_{35} = 25$, $x_{45} = 5$ and max flow equal to 90 units with $O_4 = 30$, $O_5 = 60$ and all demands are met.

We randomly generate a fuzzy vector of triangular fuzzy numbers (Section 4.4, Chapter 4) $\overline{V} = (\overline{X}_{12}, ..., \overline{X}_{45})$ of length 7 and if it satisfies all the constraints we compute $\overline{Z} = \overline{O}_4 + \overline{O}_5$. Now we get $\overline{O}_5$ from $\overline{X}_{25} + \overline{X}_{35} + \overline{X}_{45}$ and we have to compute $\overline{O}_4$ from the equation

$$\overline{X}_{14} \approx \overline{X}_{45} + \overline{O}_4. \tag{19.8}$$

But equation (19.8) may not have any exact solution (using an equal sign) for fuzzy number $\overline{O}_4$. So, in general, we set $\overline{O}_4 = \overline{X}_{14} - \overline{X}_{45}$ and we will get $\overline{X}_{14}$ approximately equal to $\overline{X}_{45} + \overline{O}_4$ because both fuzzy numbers have the same vertex.

The problem is that too many of the random vectors $\overline{V}$ will not be feasible. We need to find intervals $T_{ij} = [0, M_{ij}]$, as explained in previous chapters, for the $\overline{X}_{ij}$ so that $\overline{V}$ has a better chance of being feasible. Looking at the crisp solution and the constraints we pick $T_{12} = [0, 10]$, $T_{13} = [0, 12]$, $T_{14} = [0, 45]$,$T_{23} = [0, 75]$, $T_{25} = [0, 40]$, $T_{35} = [0, 30]$ and $T_{45} = [0, 10]$. Changing these intervals can effect the solution. The determination of these intervals is very important. If they are too big too many $\overline{V}_i$ can be rejected as not feasible, and if they are too small we could miss a "good" solution.

Now assume we are at some point in the algorithm and $\overline{V}$ is feasible so we compute the value of the objective function $\overline{Z} = \overline{Z}_0$. If $\overline{Z}^*$ is the current best (max) value of $\overline{Z}$ then we replace $\overline{Z}^*$ with $\overline{Z}_0$ if $\overline{Z}^* < \overline{Z}_0$, otherwise we discard $\overline{Z}_0$. We are using Buckley's Method (Section 2.6.1 of Chapter 2) to evaluate $\leq$, $>$ and $\approx$ between fuzzy numbers and the method of breaking "ties" discussed in Section 2.6.4. We therefore do not expect to get $\overline{Z}^* \approx \overline{Z}_0$.

The results of the fuzzy Monte Carlo method, after N iterations, would be presented in a table and the graph of the maximum $\overline{Z}$ to be shown in a figure.

References

1. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems: Operations Research. J. Advanced Computational Intelligence 3, 171–176 (1999)
2. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
3. Chanas, S., Zielinski, P.: Criticality in the Network with Imprecise Activity Times. In: Proc. IPMU 2000, Madrid, Spain, July 3-7, 2000, pp. 1233–1238 (2000)
4. Diamond, P.: A Fuzzy Max-Flow Min-Cut Theorem. Fuzzy Sets and Systems 119, 139–148 (2001)
5. Malik, D.S., Mordeson, J.N.: Fuzzy Discrete Structures. Springer, Heidelberg (2000)
6. Shih, H.-S., Lee, E.S.: Fuzzy Multi–Level Minimum Cost Flow Problems. Fuzzy Sets and Systems 107, 159–176 (1999)
7. Taha, H.A.: Operations Research, 5th edn. MacMillan, N.Y. (1992)

20 Inventory Control: Known Demand

20.1 Introduction

This is the first of three chapters on fuzzy inventory control. We start with crisp known demand in this chapter followed by fuzzy demand in the next two chapters ending with allowing backorders in Chapter 22.

Most of the applications of fuzzy sets to inventory control have been in the area of the EOQ, or the economic order quantity problem. See ([4]-[12], [14]-[17]), and the references in these papers, for a survey of this literature on fuzzy inventory control. In this chapter we will apply fuzzy sets to model uncertainty in the single item, N-period, inventory control problem. This problem will be outlined below in Section 20.2. We will assume that demand is known (not fuzzy) but the ordering cost and holding cost can be fuzzy. In Chapter 21 demand will be fuzzy, shortages are allowed and the penalty cost for shortages is also fuzzy. Finally, we add to the model in Chapter 22 backorders, maximize profit instead of minimizing cost since we now sell the product, and discount all future monies to their present value. This chapter, and Chapters 21 and 22, are based on (Section 6.3 of [1], Chapter 9 of [2] and [3]).

20.2 Inventory Problem

Let us now describe the inventory problem we shall be studying [13]. The flow is shown in Figure 20.1. The incoming inventory x_1 will always be a non-negative integer. The variables are the z_1, $z_2, \ldots, z_N$, which are the amounts we are to order each period. The z_i, $1 \leq i \leq N$, will always be non–negative integers. If we allow the z_i to be fuzzy we will have to defuzzify them in the end, so we will have them non–negative integers. The D_i represents the demand in the i-th period and the x_i, $2 \leq i \leq N$, stands for the outgoing inventory, which will be the starting inventory for the next period, in the i-th period. So we must have

$$x_{i+1} = x_i + z_i - D_i, \tag{20.1}$$

$1 \leq i \leq N$.

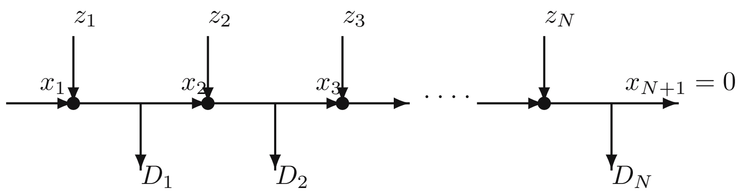


Fig. 20.1. Inventory Problem

There are only N periods and at the end of the planning horizon we assume that the final inventory will be zero. That is, we want $x_{N+1} = 0$. When D_i is fuzzy, the x_i, $i \geq 2$, will also be fuzzy from equation (20.1), so we will adjust this crisp constraint of zero final inventory in the next two chapters.

Another basic assumption throughout these three chapters is that we have zero delivery lag with instant replenishment at the start of each period.

We wish to minimize the total inventory cost over the N periods. This cost is made up of four components: (1) purchase cost; (2) ordering cost; (3) holding cost; and (4) shortage cost. We now discuss the first three in detail. The fourth one will be discussed later in Chapter 21.

We assume that we are to buy the item and we do not produce it ourselves. There may, or may not, be price breaks. A simple model, which we shall use is

$$c_i(z_i) = \begin{cases} 10z_i & : \quad 0 \leq z_i \leq L_i \\ 10L_i + 5(z_i - L_i) & : \quad z_i > L_i. \end{cases} \tag{20.2}$$

This means in the i-th period we pay \$ 10 per unit for the first L_i units and then we pay \$ 5/unit for each additional unit. We will assume these are all known numbers and they will not be fuzzy.

The ordering cost K_i in the i-th period is the cost of placing the order, checking up on the order, and putting the items into inventory when they arrive. This number is always difficult to estimate so we will model it using a fuzzy number. Then the total cost of obtaining z_i units at the start of the i-th period is

$$\overline{C}_i(z_i) = \begin{cases} 0 & : \quad z_i = 0 \\ \overline{K}_i + c_i(z_i) & : \quad z_i > 0, \end{cases} \tag{20.3}$$

for fuzzy $\overline{K}_i$.

The holding cost is assumed to be proportional to the ending inventory $x_{i+1} = x_i + z_i - D_i$. The model may be readily extended to cover any holding cost function $H_i(x_{i+1})$ by replacing x_{i+1} by $H_i(x_{i+1})$. For example, holding cost may be modeled as proportional to $(x_i + x_{i+1})/2$. In our first model in this chapter no shortages are allowed and D_i is crisp so all the x_i will be non-negative real numbers. In our other models in later chapters shortages are allowed and D_i can be fuzzy, so certain adjustments in the development must be made. Let h_i be the holding cost per unit for the i-th period. This number, depending on interest on invested capital, depreciation, etc., is very difficult to determine exactly so it will be fuzzy. For non-negative x_i and crisp D_i the holding cost for the i-th period is

$$\overline{h}_i(x_i + z_i - D_i), \tag{20.4}$$

for fuzzy $\overline{h}_i$, since $x_{i+1} = x_i + z_i - D_i$.

When no shortages are allowed let

$$\overline{TC}_i = \overline{C}_i(z_i) + \overline{h}_i(x_i + z_i - D_i), \tag{20.5}$$

and

$$\overline{Z} = \sum_{i=1}^{N} \overline{TC}_i. \tag{20.6}$$

We wish to find z_i, $1 \leq i \leq N$, to minimize $\overline{Z}$. The constraint is $x_{N+1} = 0$.

Now $\overline{Z}$ is a fuzzy set so we can not minimize it. What we did in our previous research on this topic was to first change $\min(\overline{Z})$ into a multiobjective optimization problem, and then into a single objective problem as in Section 2.5 of Chapter 2.

The model discussed in the next section, together with a numerical example, is to find the z_i to minimize the expression in equation (20.6) subject to $x_{N+1} = 0$ given: (1) x_1, $c_i(z_i)$, N, D_i, all crisp; and (2) $\overline{h}_i$ and $\overline{K}_i$ fuzzy. Our first approximate solution to this problem was to employ an evolutionary algorithm and now we plan to use a Monte Carlo method. Notice that the variables are the z_i which are all non-negative integers. So, in this and the next two chapters, we will be using sequences of random non-negative integers in Monte Carlo and not sequences of random fuzzy numbers.

The N-period inventory problem studied in this chapter is usually solved, with no fuzzy sets, using dynamic programming. So a solution method, when some of the parameters are fuzzy, might be fuzzy dynamic programming. Some work has been done, employing (fuzzy) dynamic programming, for N–period fuzzy inventory control problem ([7],[11],[12]).

The book [11] gives an excellent review of multi–stage inventory control. Both fuzzy sets and stochastic models are used. The solution method is dynamic programming. They seek to maximize the weighted sum of the membership functions which assures us that the system states, over their respective stages, are satisfactory. This is not the standard cost minimization, or profit maximization, model in inventory control. The set up cost, demand, storage cost, lost sales cost are all crisp and not fuzzy.

A recent survey of fuzzy sets applied to production management research (including inventory control, job shop scheduling, project scheduling, and forecasting) is [6].

Dynamic programming can be very computationally intensive, and even more so, for fuzzy dynamic programming. Therefore, we wished to avoid a fuzzy dynamic programming approach. Instead we employed an evolutionary algorithm and now Monte Carlo.

20.3 Monte Carlo Method

The objective is to find non-negative integers z_i, $1 \leq i \leq N$, to minimize $\overline{Z}$. $\overline{K}_i$ and $\overline{h}_i$ will be trapezoidal fuzzy numbers which implies $\overline{Z}_i$ will also be a trapezoidal fuzzy number. Let $\overline{Z} = (z_1/z_2, z_3/z_4)$. The only other thing we must do is to expand constraint $x_{N+1} = 0$ to involve the z_i (decision variables).

Since no shortages are allowed the constraints are:

$$\sum_{i=1}^{k} D_i - x_1 \leq \sum_{i=1}^{k} z_i \leq \sum_{i=1}^{N} D_i - x_1, \tag{20.7}$$

for $k = 1, 2, \ldots, N-1$, and since ending inventory is zero

$$\sum_{i-1}^{N} z_i = \sum_{i=1}^{N} D_i - x_1. \tag{20.8}$$

We summarize this fuzzy inventory control problem: (1)find non-negative integers z_i to minimize $\overline{Z}$; (2) subject to the constraints in equations (20.7)-(20.8); and (3) for given values of x_1, $\overline{K}_i$, $c_i(z_i)$, N, D_i and $\overline{h}_i$.

Example 20.3.1

We use $N = 4$ to be able to easily compute the corresponding crisp solution to compare to the result of the evolutionary algorithm and the Monte Carlo technique.

Initial inventory (x_1) is 15 units, and we use

$$c_i(z_i) = \begin{cases} 6z_i & : \quad 0 \leq z_i \leq 50 \\ 300 + 4(z_i - 50) & : \quad z_i > 50. \end{cases} \tag{20.9}$$

for $i = 1, 2, 3, 4$ and $\overline{K}_i$, D_i, $\overline{h}_i$ are given in Table 20.1.

The evolutionary algorithm used $\lambda_1 = \lambda_2 = 0.4$, $\lambda_3 = 0.2$ and $M = 1000$ (see equation (2.42)). The evolutionary algorithm solution for min$\overline{Z}$ produced $z_1 = 91$, $z_2 = 0$, $z_3 = 153$ and $z_4 = 0$.

Table 20.1. Data for Example 20.3.1

Period	D_i	$\overline{K}_i$	$\overline{h}_i$
1	76	$(92/96, 100/104)$	$(0.8/1, 1/1.2)$
2	26	$(108/112, 116/120)$	$(1.4/1.8, 2.2/2.6)$
3	90	$(179/183, 187/191)$	$(1.4/1.8, 2.2/2.6)$
4	67	$(64/68, 72/76)$	$(0.8/1, 1/1.2)$

To obtain a crisp solution we defuzzify $\overline{K}_i$ and $\overline{h}_i$ as follows: (1) if $\overline{K}_i = (k_{i1}/k_{i2}, k_{i3}/k_{i4})$, the defuzzified value is $(k_{i2} + k_{i3})/2 = k_i^*$; (2) if $\overline{h}_i = (h_{i1}/h_{i2}, h_{i3}/h_{i4})$, use $h_i^* = (h_{i2} + h_{i3})/2$. These values are $k_1^* = 98$, $k_2^* = 114$, $k_3^* = 185$, $k_4^* = 70$ and $h_1^* = h_4^* = 1.0$, $h_2^* = h_3^* = 2$ for all i. The crisp optimal solution is $z_1 = 88$, $z_2 = 0$, $z_3 = 156$ and $z_4 = 0$ with minimum cost \$1622.

20.3.1 Monte Carlo Solution

Randomly generate vectors $v = (z_1, z_2, z_3, z_4)$, where the z_i are non-negative integers as discussed in Chapter 3, and they satisfy equations (20.7)-(20.8). The resulting constraints are: (1) $61 \leq z_1 \leq 244$; (2) $87 \leq z_1 + z_2 \leq 244$; (3) $177 \leq z_1 + z_2 + z_3 \leq 244$; and (4) $244 = z_1 + z_2 + z_3 + z_4$. Then evaluate equation (20.6) to obtain fuzzy cost $\overline{Z}$. Let the current value of fuzzy cost be $\overline{Z}_0$ and, up to now, the minimum fuzzy cost $\overline{Z}^*$. If $\overline{Z}_0 < \overline{Z}^*$, then replace $\overline{Z}^*$ with $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and produce the next random vector v. Now we need to decide on how we will evaluate $<$ between fuzzy numbers. We will modify Buckley's Method, Section 2.6.1 of Chapter 2, in order to reduce the possibility of "ties" (Section 2.6.4). We do not want a large collection of fuzzy numbers for $\overline{Z}$ in equation (20.6), all approximately equal to each other, and all considered the minimum.

Let $\overline{M} = (m_1/m_2, m_3/m_4)$ and $\overline{N} = (n_1/n_2, n_3/n_4)$. Then $\overline{N} < \overline{M}$ if $n_2 < m_2$ and $\overline{N} > \overline{M}$ if $n_2 > m_2$. Assume that $n_2 = m_2$. Then $\overline{N} < \overline{M}$ if $n_3 < m_3$ and $\overline{N} > \overline{M}$ if $n_3 > m_3$. Assume that $n_2 = m_2$ and $n_3 = m_3$. Then $\overline{N} < \overline{M}$ if $n_1 < m_1$ and $\overline{N} > \overline{M}$ if $n_1 > m_1$. Assume that $n_2 = m_2$, $n_3 = m_3$ and $n_1 = m_1$. Then $\overline{N} < \overline{M}$ if $n_4 < m_4$ and $\overline{N} > \overline{M}$ if $n_4 > m_4$. If $n_2 = m_2$, $n_3 = m_3$, $n_1 = m_1$ and $n_4 = m_4$, then $\overline{M} = \overline{N}$ and discard one of them, with their corresponding z_i values.

After many iterations (number of random vectors v generated) the Monte Carlo solution will also be shown in Table 20.2. In Table 20.2 "EA" denotes the evolutionary algorithm solution and TBC denotes "to be completed".

The minimum fuzzy cost $\overline{Z}$ found by the evolutionary algorithm, corresponding to the optimal values of the z_i given in Table 20.2, is given in Table 20.3. The results of the Monte Carlo simulation would also appear in these tables. The fuzzy numbers in Table 20.3 are trapezoidal fuzzy numbers. The crisp minimum cost is \$1622.

Table 20.2. Solutions to the Inventory Problem

Variable	Crisp	EA	Monte Carlo
z_1	88	91	TBC
z_2	0	0	TBC
z_3	156	153	TBC
z_4	0	0	TBC

Table 20.3. Minimum Fuzzy Cost from EA and Monte Carlo Methods

Method	Fuzzy Cost
EA	$(1570/1613, 1649/1693)$
Monte Carlo	TBC

References

1. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
2. Buckley, J.J.: Fuzzy probabilities: New Approach and Applications. Physica-Verlag, Heidelberg (2003)
3. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems in Operations Research: Fuzzy Inventory Control. Soft Computing 7, 121–129 (2002)
4. Chang, S.-C.: Fuzzy Production Inventory for Fuzzy Product Quantity with Triangular Fuzzy Numbers. Fuzzy Sets and Systems 107, 37–57 (1999)
5. Gen, M., Tsujiomura, Y., Zheng, D.: An Application of Fuzzy Set Theory to Inventory Control Problems. Computers in Industrial Engineering 33, 553–556 (1997)
6. Guiffrida, A.L., Nagi, R.: Fuzzy Set Theory Applications in Production Management Research: A Literature Survey. J. Intelligent Manufacturing 9, 39–56 (1998)
7. Kacprzyk, J.: Multistage Decision-Making under Fuzziness. Verlag TÜV Reinland, Köln, Germany (1983)
8. Katagiri, H., Ishii, H.: Some Inventory Problems with Fuzzy Shortage Cost. Fuzzy Sets and Systems 111, 87–97 (2000)
9. Lee, H.-M., Yao, J.-S.: Economic Order Quantity in Fuzzy Sense for Inventory without Backorder Model. Fuzzy Sets and Systems 105, 13–31 (1999)
10. Lin, D.-C., Tao, J.-S.: Fuzzy Economic Production for Production Inventory. Fuzzy Sets and Systems 111, 465–495 (2000)
11. Liu, B.L., Esogbue, A.O.: Esogbue: Decision Criteria and Optimal Inventory Processes. Kluwer Academic Publishers, Norwell, Mass. (1999)
12. Sommer, G.: Fuzzy Inventory Scheduling in Applied Systems. In: Lasker, G.E. (ed.) Applied Systems and Cybernetics, vol. VI, pp. 3052–3062. Pergamon Press, N.Y. (1981)
13. Taha, H.A.: Operations Research. Macmillan, New York (1992)
14. Xiong, G., Koivisto, H.: Research on Fuzzy Inventory Control under Supply Chain Management Environment. In: Sloot, P.M.A., Abramson, D., Bogdanov, A.V., Gorbachev, Y.E., Dongarra, J.J., Zomaya, A.Y. (eds.) ICCS 2003. LNCS, vol. 2658, pp. 907–916. Springer, Heidelberg (2003)
15. Yao, J.-S., Lee, H.-M.: Fuzzy Inventory with or without Backorder Quantity with Trapezoidal Fuzzy Numbers. Fuzzy Sets and Systems 105, 311–337 (1999)
16. Yao, J.-S., Su, J.-S.: Fuzzy Inventory with Backorder for Fuzzy Total Demand Based on Interval–Valued Fuzzy Set. European J. Operational Research 124, 390–408 (2000)
17. Yao, J.-S., Chang, S.-C., Su, J.-S.: Fuzzy Inventory without Backorder for Fuzzy Order Quantity and Fuzzy Total Demand Quantity. Computers and Operations Research 27, 935–962 (2000)

21 Inventory Control: Fuzzy Demand

21.1 Introduction

This chapter continues Chapter 20 on fuzzy inventory control. The first changes made in this chapter are to allow shortages and have demand $\overline{D}_i$ fuzzy. There will be no backlogging (back orders) so shortages result in lost sales and loss of customer's goodwill. The penalty cost, due to shortages, is usually very difficult to estimate so it will be modeled by a fuzzy number $\overline{p}_i$ for the period i. We will still minimize the total fuzzy cost, with $\overline{D}_i$, $\overline{K}_i$, $\overline{h}_i$, $\overline{p}_i$ and also the $\overline{x}_i$ $(i \geq 2)$ all fuzzy, subject to fuzzy $\overline{x}_{N+1}$ approximately zero. We now discuss in more detail the changes from Chapter 20 to this chapter in order to handle fuzzy demand and shortages. Then we plan to apply our Monte Carlo method to a numerical example and compare its solution to a crisp solution and an evolutionary algorithm solution.

21.2 Inventory Model

There will be a slight change in the fuzzy mathematics in this and the following chapter that does not occur in the rest of the book. We will be working with fuzzy sets that are parts of fuzzy numbers. We will call these fuzzy sets "non-standard" fuzzy numbers. We will discuss these fuzzy sets in more detail later in this section.

The $\overline{D}_i$, $\overline{K}_i$, $\overline{h}_i$ and $\overline{p}_i$ will all be non-negative trapezoidal fuzzy numbers, the $\overline{x}_i$ $(i \geq 2)$ incoming inventory is fuzzy, and fuzzy cost $\overline{Z}$ will be a non–negative fuzzy set. Also, we need a new definition of holding cost and a new definition of $\overline{x}_{N+1} \approx 0$ which specifies the constraints on the z_i. What has not changed from Chapter 20 is $\overline{C}_i(z_i)$, $\overline{h}_i$ and crisp starting inventory x_1.

We now show how we will construct non-standard fuzzy numbers. Let

$$\overline{\text{Pos}}(x) = \begin{cases} 1 & : \quad x \geq 0 \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{21.1}$$

and let $\overline{N} = (a/b, c/d)$, with $a < 0 < d$. Then fuzzy sets like

$$\overline{N} \cap \overline{\text{Pos}} \tag{21.2}$$

will be used.

$\overline{D}_i$ will be a non-negative trapezoidal fuzzy number for all i. The $\overline{x}_i$, $i \geq 2$, as we shall see, will also be fuzzy but the equations $\overline{x}_{i+1} = \overline{x}_i + z_i - \overline{D}_i$, $i \geq 2$, will no longer hold.

Shortages are allowed but there are no backorders. When demand exceeds supply sales are lost. Let $\overline{H}_i(z_i)$ measure how much supply exceeds demand and $\overline{P}_i(z_i)$ measure how much demand exceeds supply, for all periods i. Define

$$\overline{H}_i(z_i) = (\overline{x}_i + z_i - \overline{D}_i) \cap \overline{\text{Pos}}, \tag{21.3}$$

$$\overline{P}_i(z_i) = (\overline{D}_i - [\overline{x}_i + z_i]) \cap \overline{\text{Pos}}, \tag{21.4}$$

$1 \leq i \leq N$, for $\overline{\text{Pos}}$ defined by equation (21.1). See Figures 21.1-21.6. Of course, for $i = 1$ we have $x_1 = \overline{x}_1$. Then

$$\overline{TC}_i = \overline{C}_i(z_i) + \overline{h}_i \cdot \overline{H}_i(z_i) + \overline{p}_i \cdot \overline{P}_i(z_i), \tag{21.5}$$

and total fuzzy cost is

$$\overline{Z} = \sum_{i=1}^{N} \overline{TC}_i. \tag{21.6}$$

In equation (21.5) $\overline{p}_i$ is the fuzzy penalty cost due to shortages. $\overline{Z}$ will be a trapezoidal shaped fuzzy number. Since $\overline{H}_i(z_i)$ measures how much supply exceeds demand, then

$$\overline{x}_{i+1} = \overline{H}_i(z_i), \tag{21.7}$$

$i = 1, 2, \ldots, N-1$, which is the outgoing inventory for the i-th period.

We now need to discuss the fuzzy arithmetic involving non-standard fuzzy numbers in more detail. Examples of non-standard fuzzy numbers can be seen as $\overline{H}_i(z_i)$ ($\overline{P}_i(z_i)$) in Figure 21.2 (21.3) and as $\overline{H}_i(z_i)$ ($\overline{P}_i(z_i)$) in Figure 21.5 (21.6). Notice that all of these non-standard fuzzy numbers are normalized (have maximum membership equal to one) but the $\overline{H}_i(z_i)$ in Figure 21.5 has maximum membership 0.4 (this fuzzy set will also be called a subnormal fuzzy set). Suppose $\overline{M} \approx (m_1/m_2, m_3/m_4)$, $m_1 \geq 0$, is a non-negative trapezoidal shaped fuzzy number and $\overline{N}$ is a non-standard fuzzy number. We will need to add and multiply $\overline{M}$ and $\overline{N}$ using α-cuts and interval arithmetic (Section 2.3.3 of Chapter 2). There is no problem if $\overline{N}$ is normalized as in Figures 21.2, 21.3 and 21.6. So let us consider $\overline{N} = \overline{H}_i(z_i)$ in Figure 21.5 and $\overline{M} + \overline{N} = \overline{Q}$. We can compute the $\alpha-$cuts of $\overline{M}$ and $\overline{N}$ for $0 \leq \alpha \leq 0.4$, add them, and get the corresponding α-cuts of $\overline{Q}$. However, for $\alpha > 0.4$ the α-cuts of $\overline{N}$ are empty and we get nothing for Q. So $\overline{Q}$ will be a non-standard fuzzy number with maximum membership 0.4. The same thing will happen for multiplication. In any operation involving a series of additions and multiplications between fuzzy numbers and non-standard fuzzy numbers, where at least one of the items is a non-standard fuzzy number with maximum membership $s < 1$, then the final result will be a non-standard fuzzy number with maximum membership less than or equal to s. Non-standard fuzzy numbers with maximum membership less than one create a lot of problems in the model: (1)fuzzy cost $\overline{Z}$ in equation (21.6) could be subnormal (maximum membership less than one); (2) outgoing inventory $\overline{x}_{i+1}$ could be subnormal; (3) final inventory $\overline{x}_{N+1}$ may be subnormal; and (4) "Penalty" can be subnormal. So as to avoid all these problems we will not allow subnormal (maximum membership less than one) fuzzy sets. We will treat the $\overline{H}_i(z_i)$ in Figure 21.5 as a number $0.xxxx$ with $xxxx < 0.5$ and round it down to zero. So if zero falls between c and d as in Figure 21.4 $\overline{H}_i(z_i) = 0$ and $\overline{P}_i(z_i)$ is the non-standard fuzzy number in Figure 21.6 using the vertical straight line from $(-c, 0)$ to $(-c, 1)$.

We now introduce notation for these normalized non-standard fuzzy numbers. The trapezoidal fuzzy number in Figure 21.1 is represented as $(a/b, c/d)$, for $a < b < 0 < c < d$, and if the sides are curved $\approx (a/b, c/d)$. The non-standard fuzzy number in Figure 21.2 (21.3) has representation $(0/0, c/d)$ $((0/0, -b/-a))$. The non-standard fuzzy number in Figure 21.6, after rounding the $\overline{H}_i(z_i)$ in Figure 21.5 down to zero, and rounding up to the vertical line in Figure 21.6, has representation $(-c/-c, -b/-a)$. The membership function for $(x_1/x_1, x_2/x_3)$, with $0 \le x_1 < x_2 < x_3$, is: (1) zero on $(-\infty, x_1)$; (2) one on $[x_1, x_2]$; 3) continuously decreasing from $(x_2, 1)$ to $(x_3, 0)$ on $[x_2, x_3]$; and (4) zero on (x_3, ∞).

We now have the following cases depending where zero falls in relation to a, b, c, d in Figure 21.1.

1. If $0 \le a$, then $\overline{H}_i(z_i) = \overline{x}_i + z_i - \overline{D}_i$, $\overline{P}_i(z_i) = 0$.
2. If $a < 0 < b$, then we get a subnormal fuzzy set on the interval $[a, 0]$ which we round down to zero giving $\overline{P}_i(z_i) = 0$. The normalized non-standard fuzzy number on $[0, d]$ we round up to be on the interval $[b, d]$ and set $\overline{H}_i(z_i) = (b/b, c/d)$.
3. If $b \le 0 \le c$, then we obtain two normalized non-standard fuzzy numbers one on the interval $[0, d]$ and the other on the interval $[a, 0]$. Set $\overline{H}_i(z_i) = (0/0, c/d)$ and define $\overline{P}_i(z_i) = (0/0, -b/-a)$.
4. If $c < 0 < d$, then we obtain a subnormal fuzzy set on the interval $[0, d]$ which we round down to zero and $\overline{H}_i(z_i) = 0$. What is left is the normalized non-standard fuzzy number on the interval $[a, 0]$ which we round down to be on the interval $[a, c]$ and, after making it positive, set $\overline{P}_i(z_i) = (-c/-c, -b/-a)$.
5. If $d \le 0$, then $\overline{H}_i(z_i) = 0$, $\overline{P}_i(z_i) = -\overline{x}_i - z_i + \overline{D}_i$.

In all cases $\overline{x}_i + z_i - \overline{D}_i$ will be a trapezoidal fuzzy number like the one in Figure 21.1, since $\overline{D}_i$ is always a trapezoidal fuzzy number, but zero may be placed anywhere from $-\infty$ to ∞. In the computations to get $\overline{TC}_i$ in equation (21.5) $\overline{H}_i(z_i)$ and /or $\overline{P}_i(z_i)$ may be normalized non-standard fuzzy numbers but, since $\overline{K}_i$, $\overline{h}_i$ and $\overline{p}_i$ are always trapezoidal fuzzy numbers, $\overline{TC}_i$ will be a trapezoidal (shaped) fuzzy number. The sides of $\overline{TC}_i$ may be straight lines or curves. In general we will call $\overline{TC}_i$ a trapezoidal shaped fuzzy numbers so $\overline{Z}$ in equation (21.6) will also be a trapezoidal shaped fuzzy number.

The last change, from Chapter 20, is to discuss is the constraint $\overline{x}_{N+1} \approx 0$. Now $\overline{x}_{N+1}$ could be zero and this surely satisfies $\overline{x}_{N+1} \approx 0$. So assume that final

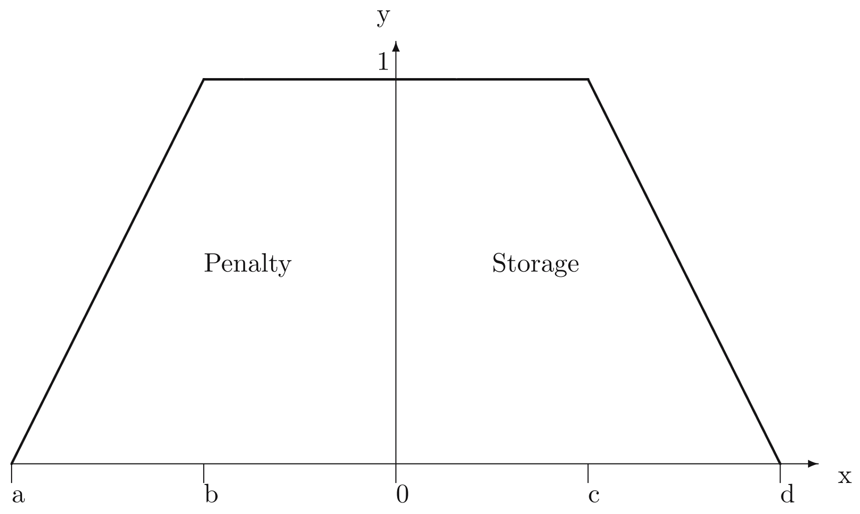


Fig. 21.1. The Fuzzy Set $\overline{x}_i + z_i - \overline{D}_i$

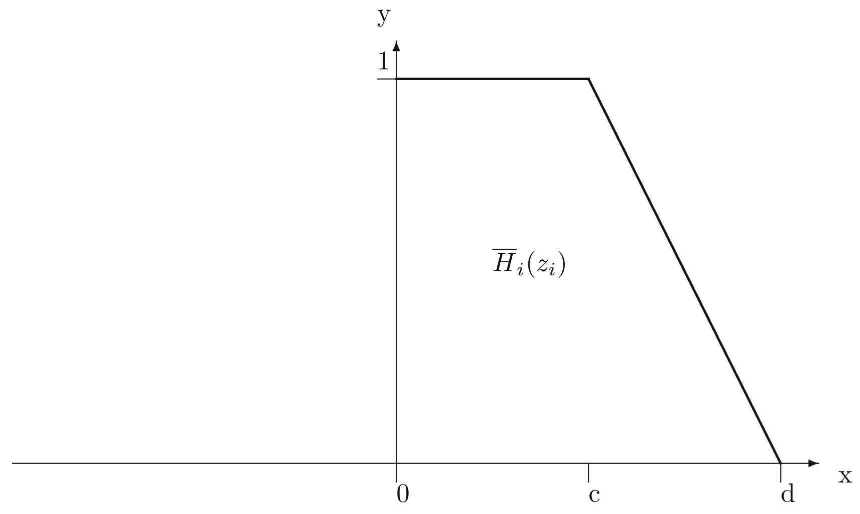


Fig. 21.2. Storage from Figure 21.1

inventory is not zero and let $\overline{x}_{N+1}[1] = [u, v]$. If $u \leq 0 \leq v$ we will say that the constraint $\overline{x}_{N+1} \approx 0$ is satisfied, otherwise the constraint is not satisfied.

We summarize this fuzzy N-period inventory control problem: (1) find nonnegative integers z_i to minimize the expression in equation (21.6); (2) subject to the constraint $\overline{x}_{N+1} \approx 0$; (3) for $\overline{Z}$, computed using α-cuts and interval arithmetic, given by equations (21.5) and (21.6); and (4) for given values of x_1, $c_i(z_i)$, $\overline{K}_i$, N, $\overline{D}_i$, $\overline{h}_i$, $\overline{p}_i$.

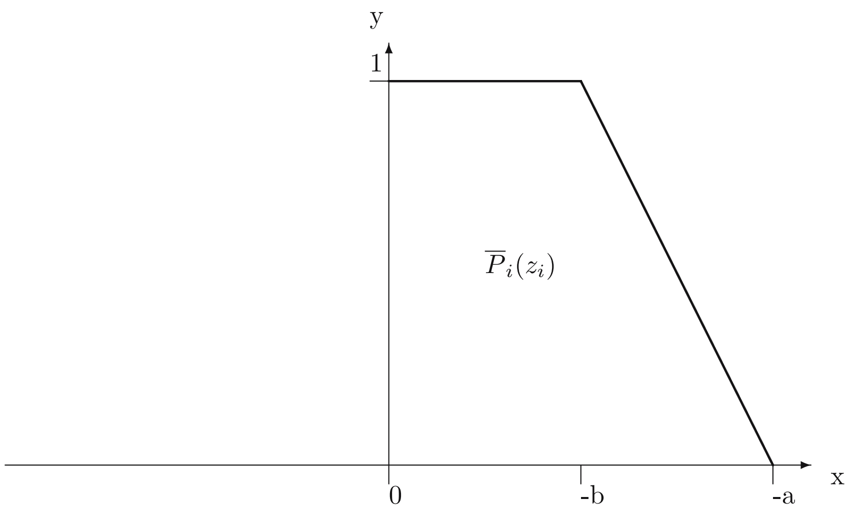


Fig. 21.3. Penalty from Figure 21.1

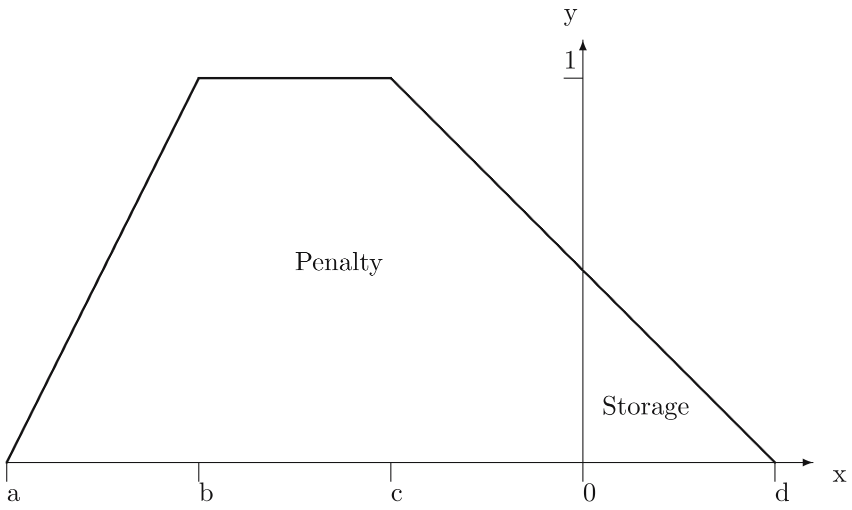


Fig. 21.4. The Fuzzy Set $\overline{x}_i + z_i - \overline{D}_i$

Example 21.2.1

This will continue Example 20.3.1. Fuzzy demand $\overline{D}_i$ and fuzzy penalty cost $\overline{p}_i$ are shown in Table 21.1. The defuzzified values of the $\overline{D}_i$ in Table 21.1 will equal the crisp D_i values in Table 20.1. The defuzzified values of the $\overline{p}_i$ are all equal to 3.00. We also change $c_i(z_i)$ to equal $2z_i$ with no price breaks. All the other parameter values are the same as in Example 20.3.1 (see Table 20.1).

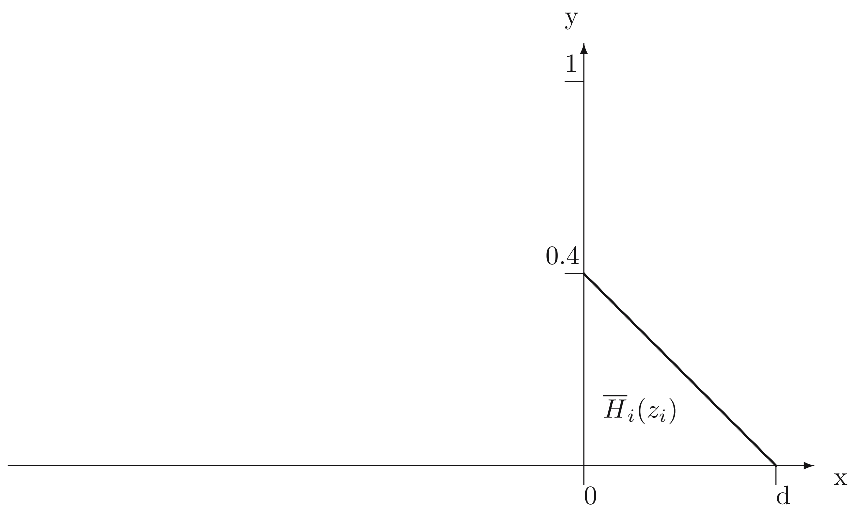


Fig. 21.5. Storage from Figure 21.4

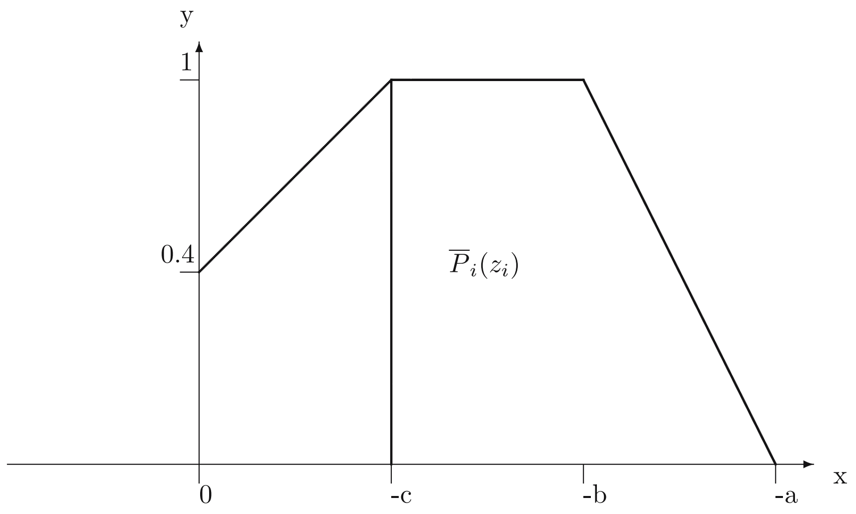


Fig. 21.6. Penalty from Figure 21.4

A main problem now is for the evolutionary algorithm and/or the Monte Carlo method to check the constraints. A population member is still $z = (z_1, z_2, z_3, z_4)$ for $z_i \geq 0$ integer.

To check feasibility we first compute $\overline{x}_2 = \overline{H}_1(z_1)$ from z_1. Then $\overline{x}_3 = \overline{H}_2(z_2)$ from $\overline{x}_2$, and z_2 and $\overline{x}_4 = \overline{H}_3(z_3)$ from $\overline{x}_3$ and z_3. Finally, compute $\overline{x}_5$ and check $\overline{x}_5 \approx 0$. If $\overline{x}_5$ is approximately zero (defined above), then we keep this $z = (z_1, \ldots, z_4)$. Otherwise, discard this population member and generate another one. Keep generating new population members until we obtain one that is

Table 21.1. Data for Example 21.2.1

Period	$\overline{D}_i$	$\overline{p}_i$
1	$(70/74, 78/82)$	$(2.0/3.0, 3.0/4.0)$
2	$(24/26, 26/28)$	$(2.5/2.8, 3.2/3.5)$
3	$(86/89, 91/94)$	$(2.0/2.7, 3.3/4.0)$
4	$(64/65, 69/70)$	$(2.5/2.9, 3.1/3.5)$

feasible, which is kept. For the evolutionary algorithm solution we minimized $\overline{Z}$, a trapezoidal shaped fuzzy number, in Equation (21.6) as discussed in Section 2.5 in Chapter 2. The values for the λ_i in equation (2.42) were $\lambda_1 = \lambda_2 = 0.4$ and $\lambda_3 = 0.2$.

The evolutionary algorithm's solution for $\overline{Z}$ produced fuzzy cost $\overline{Z}$ whose core, the membership values were all equal to one, was the interval $[859.0, 907.2]$. The best values for the decision variables were $z_1 = 89$, $z_2 = 0$, $z_3 = 89$ and $z_4 = 65$.

Next, we defuzzified the fuzzy inventory problem and obtained its solution $z_1 = 61$, $z_2 = 116$, $z_3 = 0$ and $z_4 = 67$. We used the same defuzzified values of $\overline{K}_i$ and $\overline{h}_i$ as in Example 20.3.1, defuzzified $\overline{D}_i$ gives D_i values in Table 20.1, and defuzzified $\overline{p}_i$ is always 3.00. The minimum cost in the crisp problem was \$860, which was in the core of $\overline{Z}$.

21.2.1 Monte Carlo Solution

Randomly generate vectors $v = (z_1, z_2, z_3, z_4)$ where the z_i are non-negative integers as discussed in Chapter 4, and they satisfy the constraint $\overline{x}_5 \approx 0$. Looking at the "EA" solution in Table 21.2 we initially chose the interval $[0, 100]$ for all the z_i. Then evaluate equations (21.5) and (21.6) to obtain fuzzy cost $\overline{Z}$. Let the current value of fuzzy cost be $\overline{Z}_0$ and, up to now, the minimum fuzzy cost $\overline{Z}^*$. If $\overline{Z}_0 < \overline{Z}^*$, then replace $\overline{Z}^*$ with $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and produce the next random vector v. Now we need to decide on how we will evaluate $<$ between fuzzy numbers. We will use the modified Buckley's Method discussed in Section 20.3.1 in Chapter 20.

In comparing two trapezoidal (shaped) fuzzy numbers $\overline{M} \approx (m_1/m_2, m_3/m_4)$ and $\overline{N} \approx (n_1/n_2, n_3/n_4)$ the modified Buckley's Method depends only on the numbers m_i and n_i, $i = 1, 2, 3, 4$. So we only need to find these numbers for the $\overline{TC}_i$, $i = 1, 2, 3, 4$. Let $\overline{K}_i = (k_{i1}/k_{i2}, k_{i3}/k_{i4})$, $\overline{h}_i = (h_{i1}/h_{i2}, h_{i3}/h_{i4})$, $\overline{p}_i = (p_{i1}/p_{i2}, p_{i3}/p_{i4})$, $\overline{H}_i(z_i) = (H_{i1}/H_{i2}, H_{i3}/H_{i4})$ and $\overline{P}_i(z_i) = (P_{i1}/P_{i2}, P_{i3}/P_{i4})$, $i = 1, 2, 3, 4$. We may get $H_{i1} = H_{i2}$ and $P_{i1} = P_{i2}$. If $\overline{TC}_i \approx (TC_{i1}/TC_{i2}, TC_{i3}/TC_{i4})$ then

$$TC_{i1} = k_{i1} + 2z_i + h_{i1}H_{i1} + p_{i1}P_{i1}, \tag{21.8}$$

and continuing we end with

$$TC_{i4} = k_{i4} + 2z_i + h_{i4}H_{i4} + p_{i4}P_{i4}. \tag{21.9}$$

If $\overline{Z} \approx (w_1/w_2, w_3/w_4)$, then $w_1 = \sum_{j=1}^{4} TC_{j1}$, etc.

Table 21.2. Solutions to the Inventory Problem

Variable	Crisp	EA	Monte Carlo
z_1	61	89	TBC
z_2	116	0	TBC
z_3	0	89	TBC
z_4	67	65	TBC

Table 21.3. Core of Minimum Fuzzy Cost from EA and Monte Carlo Methods

Method	Core of Fuzzy Cost
EA	$[859, 907])$
Monte Carlo	TBC

After N iterations (number of random vectors v generated) the Monte Carlo solution will also be shown in Table 21.2. In Table 21.2 "EA" denotes the evolutionary algorithm solution and TBC means "to be completed". We also want to see if the minimum cost to the crisp solution falls in the core of the minimum fuzzy cost for the evolutionary algorithm and Monte Carlo solutions which is to be shown in Table 21.3.

22 Inventory Control: Backordering

22.1 Introduction

The major changes in this chapter, from the previous chapter, are: (1) we now allow for backorders; (2) we maximize profit instead of minimizing cost; and (3) we discount future monies back to their present value.

We still assume we buy the item, and we do not produce it ourselves, so we can still use $\overline{C}_i(z_i)$ in equation (20.3) in Chapter 20. However, we will need to redefine $\overline{H}_i(z_i)$ and $\overline{P}_i(z_i)$ in equations (21.3) and (21.4) in Chapter 21, respectively. Let us now look at these changes in more detail and develop our new inventory model.

22.2 Inventory Model

We are going to resell the item at \$$r$ per unit where r is known (not fuzzy). We cannot use $\overline{x}_i + z_i - \overline{D}_i$ directly, as shown in Figures 21.1 - 21.6 of Chapter 21, because they do not show what we sell each period. Let $\overline{S}_i(z_i)$ be the (fuzzy) amount we sell during period i, $1 \leq i \leq N$. First we need to decide when $\overline{x}_i + z_i > \overline{D}_i$, or $\overline{D}_i > \overline{x}_i + z_i$ or $\overline{x}_i + z_i \approx \overline{D}_i$, $i = 1, 2, ..., N$. We will employ the following rules: (1) $\overline{x}_i + z_i > \overline{D}_i$ if the core of $(\overline{x}_i + z_i - \overline{D}_i)$ lies to the right of zero; (2) $\overline{D}_i > \overline{x}_i + z_i$ when the core of $(\overline{x}_i + z_i - \overline{D}_i)$ lies to the left of zero; and (3) $\overline{x}_i + z_i \approx \overline{D}_i$ if zero belongs to the core of $(\overline{x}_i + z_i - \overline{D}_i)$. Notice that $\overline{x}_i + z_i - \overline{D}_i$ will be a trapezoidal fuzzy number as shown in Figure 21.1. We use this method, instead of one of those from Section 2.6 of Chapter 2, because that was what we used in our previous research on this topic which used an evolutionary algorithm and we wish to be able to compare our Monte Carlo results to the evolutionary algorithm results.

Now we may describe our procedure to determine $\overline{S}_i(z_i)$. Assume that $\overline{x}_i + z_i - \overline{D}_i$ is a trapezoidal fuzzy number like the one shown in Figure 21.1 but zero may be anywhere from $-\infty$ to ∞. We have the following cases.

1. If $0 < b$, or $\overline{x}_i + z_i > \overline{D}_i$, then $\overline{S}_i(z_i) = \overline{D}_i$, $\overline{H}_i(z_i) = \overline{x}_i + z_i - \overline{D}_i = \overline{x}_{i+1}$, $\overline{P}_i(z_i) = 0$.

2. If $c < 0$, or $\overline{D}_i > \overline{x}_i + z_i$, then $\overline{S}_i(z_i) = \overline{x}_i + z_i$, $\overline{H}_i(z_i) = 0 = \overline{x}_{i+1}$, $\overline{P}_i(z_i) = \overline{D}_i - [\overline{x}_i + z_i]$, and next periods demand $\overline{D}_{i+1}$ will be $\overline{D}_{i+1} + \overline{P}_i(z_i)$.
3. If $b \leq 0 \leq c$, or $\overline{x}_i + z_i \approx \overline{D}_i$, then $\overline{H}_i(z_i) = 0 = \overline{x}_{i+1} = \overline{P}_i(z_i)$ and $\overline{S}_i(z_i) = \overline{D}_i$.

Notice that $\overline{P}_i(z_i)$, if it is not zero, is what is backordered and is added to next periods demand. The value $\overline{p}_i$ will now be the backorder cost per unit.

Next we may set up the profit function to be maximized. Let

$$\overline{TP}_i = r\overline{S}(z_i) - \overline{C}_i(z_i) - \overline{h}_i\,\overline{H}_i(z_i) - \overline{p}_i\,\overline{P}_i(z_i), \tag{22.1}$$

and

$$\overline{Z} = \sum_{i=1}^{N} (\overline{\alpha})^{i-1}\overline{TP}_i. \tag{22.2}$$

The $\overline{\alpha}$ are fuzzy discount factors which bring future fuzzy monies back to their present value. We must be careful in evaluating this last equation using α-cuts and interval arithmetic because $\overline{TP}_i$ may not be a non-negative trapezoidal fuzzy number. In ([1]-[3]) the evolutionary algorithm looks for the optimal values of the z_i to $max\overline{Z}$, subject to the constraint $\overline{x}_{N+1} \approx 0$ from Chapter 21, and it handles the max problem as in Section 2.5 of Chapter 2. The values for the λ_i in equation (2.42) were $\lambda_1 = \lambda_2 = 0.4$ and $\lambda_3 = 0.2$.

Now: (1) $\overline{D}_i$, $\overline{K}_i$, $\overline{h}_i$ and $\overline{p}_i$ are all non-negative trapezoidal fuzzy numbers; (2) $\overline{x}_i + z_i - \overline{D}_i$ are all trapezoidal fuzzy numbers; (3) $\overline{H}_i(z_i) = \overline{x}_{i+1}$ are non-negative trapezoidal fuzzy numbers or zero; (4) $\overline{P}_i(z_i)$ is a non-negative trapezoidal fuzzy number or zero; (5) $\overline{S}_i(z_i)$ is a non-negative trapezoidal fuzzy number or a crisp number (Case 2 above when $\overline{x}_i = 0$); (6) $\overline{TP}_i$ in equation (22.1) is a trapezoidal (shaped) fuzzy number because the sides may be curved; and (7) $\overline{Z}$ in equation (22.2) is a trapezoidal shaped fuzzy number.

22.3 Monte Carlo Method

Randomly generate vectors $v = (z_1, z_2, z_3, z_4)$ where the z_i are non-negative integers as discussed in Chapter 4, and they satisfy feasibility condition $\overline{x}_5 \approx 0$ detailed in Section 21.2 of Chapter 21. Feasibility is easy to determine because x_5 is zero (v is feasible) or in Case 1 above it is a trapezoidal fuzzy number with core greater than zero and then v is not feasible. Looking at the solutions in Table 20.2 we initially picked the interval $[0, 120]$ for the z_i. Then evaluate equations (22.1) and (22.2) to obtain fuzzy profit $\overline{Z}$. Let the current value of fuzzy profit be $\overline{Z}_0$ and, up to now, the maximum fuzzy profit $\overline{Z}^*$. If $\overline{Z}_0 > \overline{Z}^*$, then replace $\overline{Z}^*$ with $\overline{Z}_0$, otherwise discard $\overline{Z}_0$ and produce the next random vector v. Now we need to decide on how we will evaluate $<$ between fuzzy numbers. We will use the modified Buckley's Method discussed in Section 20.3.1 in Chapter 20 with obvious changes for a max problem.

In comparing two trapezoidal (shaped) fuzzy numbers $\overline{M} \approx (m_1/m_2, m_3/m_4)$ and $\overline{N} \approx (n_1/n_2, n_3/n_4)$ the modified Buckley's Method depends only on the numbers m_i and n_i, $i = 1,2,3,4$. So we only need to find these numbers for the $\overline{TP}_i$, $i = 1,2,3,4$, and then $\overline{Z}$. Let $\overline{K}_i = (k_{i1}/k_{i2}, k_{i3}/k_{i4})$, $\overline{h}_i = (h_{i1}/h_{i2}, h_{i3}/h_{i4})$, $\overline{p}_i = (p_{i1}/p_{i2}, p_{i3}/p_{i4})$, $\overline{H}_i(z_i) = (H_{i1}/H_{i2}, H_{i3}/H_{i4})$, $\overline{S}_i(z_i) = (S_{i1}/S_{i2}, S_{i3}/S_{i4})$, $\overline{P}_i(z_i) = (P_{i1}/P_{i2}, P_{i3}/P_{i4})$, and $\overline{\alpha} = (\alpha_1/\alpha_2, \alpha_3/\alpha_4)$, $i = 1,2,3,4$. If $\overline{H}_i(z_i)$ and/or $\overline{P}_i(z_i)$ are zero we use $(0/0,0/0)$. If $\overline{S}_i(z_i) = \theta$ (crisp) we use $(\theta/\theta, \theta/\theta)$. If $\overline{TP}_i = (or \approx)(TP_{i1}/TP_{i2}, TP_{i3}/TP_{i4})$ then

$$TP_{i1} = rS_{i1} - k_{i4} - c_i(z_i) - h_{i4}H_{i4} - p_{i4}P_{i4}, \tag{22.3}$$

and continuing we end with

$$TP_{i4} = rS_{i4} - k_{i1} - c_i(z_i) - h_{i1}H_{i1} - p_{i1}P_{i1}. \tag{22.4}$$

If $\overline{Z} \approx (w_1/w_2, w_3/w_4)$, then $w_1 = \sum_{j=1}^{4}(\alpha_1)^{j-1}TP_{j1}$, etc.

Example 22.3.1

Let $r_i = 5$ for all i and let the discount figure $\overline{\alpha} = (1+\overline{\beta})^{-1}$ for $\overline{\beta} \approx 6\%$; that is, $\overline{\beta} = (0.055/0.058, 0.062/0.065)$. We will use $\overline{K}_i$ and $\overline{h}_i$ from Table 22.1 because we have changed the $\overline{h}_i$ values from Table 20.1 in Chapter 20. Also, we use the $\overline{D}_i$ and $\overline{p}_i$ values from Table 21.1 in Chapter 21. We set $c_i(z_i) = 2z_i$ for all i.

Table 22.1. Data for Example 22.3.1

Period	$\overline{K}_i$	$\overline{h}_i$
1	$(92/96, 100/104)$	$(0.8/1, 1/1.2)$
2	$(108/112, 116/120)$	$(0.8/0.9, 1.1/1.2)$
3	$(179/183, 187/191)$	$(0.6/0.8, 1.2/1.4)$
4	$(64/68, 72/76)$	$(0.7/1, 1/1.3)$

Table 22.2. Solutions to the Inventory Problem in Example 22.3.1

Variable	Crisp	EA	Monte Carlo
z_1	87	59	TBC
z_2	0	116	TBC
z_3	90	0	TBC
z_4	67	65	TBC

Table 22.3. Maximum Fuzzy Profit from EA and Monte Carlo Methods

Method	Core of Max Fuzzy Profit
EA	$[351, 428]$
Monte Carlo	TBC

The initial inventory remains $x_1 = 15$. We defuzzify $\overline{K}_i$, $\overline{h}_i$, $\overline{D}_i$ and $\overline{p}_i$ to obtain a crisp optimal solution as in the previous two chapters. The optimal defuzzified solution, using $\alpha = (1.06)^{-1}$, is shown in Table 22.2. The evolutionary algorithm solution is also given in Table 22.2.

After N iterations (number of random vectors v generated) the Monte Carlo solution will also be shown in Table 22.2. In Table 22.2 "EA" denotes the evolutionary algorithm solution and TBC stands for "to be completed". We also want to see if the maximum profit to the crisp solution (\$390) falls in the core of the maximum fuzzy profit for the evolutionary algorithm and Monte Carlo solutions (Table 21.3).

References

1. Buckley, J.J.: Fuzzy probabilities: New Approach and Applications. Physica-Verlag, Heidelberg (2003)
2. Buckley, J.J., Eslami, E., Feuring, T.: Fuzzy Mathematics in Economics and Engineering. Physica-Verlag, Heidelberg (2002)
3. Buckley, J.J., Feuring, T., Hayashi, Y.: Solving Fuzzy Problems in Operations Research: Fuzzy Inventory Control. Soft Computing 7, 121–129 (2002)

23 Fuzzy Transportation Problem

23.1 Introduction

In this chapter we apply our fuzzy Monte Carlo method to determine approximate solutions to a fuzzy transportation problem. The next section presents the standard definition of the transportation model. Then in Section 23.3 we fuzzify it and apply our fuzzy Monte Carlo method to generate approximate solutions. An example is given which is to show the results of our fuzzy Monte Carlo method.

23.2 Transportation Problem

The standard transportation model seeks to find a transportation plan for a single commodity from a number of sources to a number of destinations [8]. The data in the model includes: (1) the amount of supply at each source and the demand at each destination; and (2) the unit transportation cost of the commodity from each source to each destination. A destination may receive its demand from many sources. The objective is to determine the shipping plan, to meet all demands but not exceed any supply, to minimize the total transportation cost.

Assume there are m sources and n destinations. Let x_{ij} be the amount to be shipped from source i to destination j, $1 \leq i \leq m$, $1 \leq j \leq n$. x_{ij} will be an integer greater than, or equal to, zero. Also let a_i be the amount of supply at source $i = 1, ..., m$ and b_j the demand at destination $j = 1, ..., n$. The unit transportation cost from source i to destination j is c_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$.

The linear programming model is

$$minimize\ Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}, \tag{23.1}$$

subject to

$$\sum_{i=1}^{m} x_{ij} \geq b_j,\ j = 1, ..., n, \tag{23.2}$$

and

$$\sum_{j=1}^{n} x_{ij} \leq a_i,\ i = 1, ..., m, \tag{23.3}$$

and $x_{ij} \geq 0$ and x_{ij} integer. For the model to be feasible we must have the total supply is at least equal to the total demand

$$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j. \tag{23.4}$$

This problem has a special solution algorithm and is not usually solved as a linear program [8].

23.3 Fuzzy Transportation Problem

Now we allow some, or all, of the parameters c_{ij}, a_i and b_j to be fuzzy showing any uncertainty in their values. However, the x_{ij} will be crisp because if we allowed them to be fuzzy we would have to defuzzify them at the end to obtain a feasible shipping plan. Let $\overline{C}_{ij}$ be a triangular fuzzy number representing the cost of sending one unit from source i to destination j. $\overline{A}_i$ is a triangular fuzzy number for the amount at source i and $\overline{B}_j$ is another triangular fuzzy number for the demand at destination j. Some of these parameters may be crisp. Then the fuzzy optimization problem to solve is

$$minimize\ \overline{Z} = \sum_{i=1}^{m} \sum_{j=1}^{n} \overline{C}_{ij} x_{ij}, \tag{23.5}$$

subject to

$$\sum_{i=1}^{m} x_{ij} \geq \overline{B}_j,\ j = 1, ..., n, \tag{23.6}$$

and

$$\sum_{j=1}^{n} x_{ij} \leq \overline{A}_i,\ i = 1, ..., m, \tag{23.7}$$

and $x_{ij} \geq 0$ and x_{ij} integer. For the model to be feasible we must have the total supply is at least equal to the total demand

$$\sum_{i=1}^{m} \overline{A}_i \geq \sum_{j=1}^{n} \overline{B}_j. \tag{23.8}$$

There have been numerous papers on the fuzzy transportation problem (see [1]-[7]). These authors allow for fuzzy demand and fuzzy supply and/or fuzzy cost coefficients. However, they keep the flow from source i to destination j (the x_{ij}) non-negative integers. We will do the same but apply our fuzzy Monte Carlo method to generate (approximate) solutions.

Now we must adopt a method of deciding $\leq$, $<$ and $\approx$ between fuzzy numbers to understand equations (23.5) - (23.8). We will employ Buckley's Method discussed in Section 2.6.1 using $\eta = 0.8$. First consider equation (23.6) and let crisp number $x_j = \sum_{i=1}^{m} x_{ij}$. Let $\overline{B}_j(x)$ be the membership function for triangular $\overline{B}_j = (b_{j1}/b_{j2}/b_{j3})$. We say equation (23.6) is true if $x_j \geq b_{j2}$ or if $x_j < b_{j2}$ then $\overline{B}_j(x_j) \geq 0.8$. Next look at equation (23.7) and let crisp $x_i = \sum_{j=1}^{n} x_{ij}$. Let $\overline{A}_i(x)$ be the membership function for triangular $\overline{A}_i = (a_{i1}/a_{i2}/a_{i3})$. We say equation (23.7) is true if $x_i < a_{i2}$ or if $x_i \geq a_{i2}$ then $\overline{A}_i(x_i) \geq 0.8$. Now equation (23.8) is just comparing two triangular fuzzy numbers and can be done directly using Buckley's Method. Lastly, we need to be able to decide on min $\overline{Z}$ in equation (23.5). This can be done sequentially during the simulation. Suppose at some point in the Monte Carlo process the current minimum of $\overline{Z}$ is $\overline{Z}^*$. The next feasible values for the x_{ij} produce $\overline{Z} = \overline{Z}_0$. If $\overline{Z}_0 < \overline{Z}^*$ replace $\overline{Z}^*$ with $\overline{Z}_0$, otherwise discard $\overline{Z}_0$. We will be using the "tie" breakers discussed in Section 2.6.4 so we do not expect to get $\overline{Z}^* \approx \overline{Z}_0$.

In this way we may run the fuzzy Monte Carlo method. The main problem is that when we randomly generate non-negative integers x_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$ too often they are not feasible, or equations (23.6) and (23.7) are not true. So we must pay attention to how we produce the x_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$, so that many of these sets are feasible.

Example 23.3.1

We start with the following crisp transportation problem shown in Table 23.1. There are three plants denoted as $P1$, $P2$ and $P3$ and four distribution centers shown as $D1$,..,$D4$. This is a balanced problem because the total supply equals the total demand. The values inside the table are the unit transportation costs in dollars. We easily compute the optimal solution to be $x_{13} = 2$, $x_{14} = 10$, $x_{22} = 9$, $x_{23} = 8$, $x_{31} = 10$, $x_{32} = 1$ with the rest of the x_{ij} zero and minimum total cost \$20,200.

Now some of the parameters become triangular fuzzy numbers given in Table 23.2. We used the data in Table 23.1 for the vertex values of the triangular fuzzy numbers in Table 23.2.

We randomly generate integers in $[0, 10]$, since each $x_{ij} \leq 10$, to form the vector

$$v = (x_{11}, ..., x_{14}, x_{21}, ..., x_{34}), \tag{23.9}$$

of length 12 as a possible feasible shipping schedule. If it is feasible, satisfies equations (23.6) and (23.7), we compute $\overline{Z}$ in equation (23.5). We use our Sobol quasi-random number generator (Section 3.5, Chapter 3) to get v in equation (23.9) so that these vectors will be uniformly spread around $I \cap [0, 10]^{12}$, where the "I" denotes integers.

We may simplify the constraints. Let $s_i = \sum_{j=1}^{4} x_{ij}$, $i = 1, 2, 3$ and $c_j = \sum_{i=1}^{3} x_{ij}$, $j = 1, 2, 3, 4$. The constraints can be rewritten as: (1) $s_1 < 13$, $s_2 < 18$, $s_3 < 12$; and (2) $c_j > 9$ all j.

Table 23.1. Crisp Transportation Problem in Example 23.3.1

	$D1$	$D2$	$D3$	$D4$	Supply
$P1$	500	750	300	450	12
$P2$	650	800	400	600	17
$P3$	400	700	500	550	11
Demand	10	10	10	10	40

Table 23.2. Fuzzy Transportation Problem in Example 23.3.1

	$D1$	$D2$	$D3$	$D4$	Supply
$P1$	(450/500/550)	(725/750/775)	(280/300/320)	(410/450/460)	(10/12/14)
$P2$	(600/650/700)	(800/800/800)	(390/400/420)	(570/600/650)	(15/17/20)
$P3$	(400/400/400)	(680/700/710)	(450/500/550)	(510/550/560)	(8/11/12)
Demand	(8/10/12)	(9/10/11)	(6/10/13)	(7/10/14)	

Table 23.3. Solutions in Example 23.3.1

x	Crisp Problem	Fuzzy Problem
x_{11}	0	TBC
x_{12}	0	TBC
x_{13}	2	TBC
x_{14}	10	TBC
x_{21}	0	TBC
x_{22}	9	TBC
x_{23}	8	TBC
x_{24}	0	TBC
x_{31}	10	TBC
x_{32}	1	TBC
x_{33}	0	TBC
x_{34}	0	TBC

The results of our fuzzy Monte Carlo method, after N iterations, will be displayed in Table 23.3 where TBC is for "to be completed".

References

1. Chanas, S., Kuchta, D.: A Concept of the Optimal Solution of the Transportation Problem with Fuzzy Cost Coefficients. Fuzzy Sets and Systems 82, 299–305 (1996)
2. Chanas, S., Kuchta, D.: Fuzzy Integer Transportation Problem. Fuzzy Sets and Systems 98, 291–298 (1998)
3. Chiang, J.: The Optimal Solution of the Transportation Problem with Fuzzy Demand and Fuzzy Product. J. Info. Science and Engineering 21, 439–451 (2005)
4. Dymowa, L., Dolata, M.: The Transportation Problem Under Probabilistic and Fuzzy Uncertainties. Operation Research and Decisions 4, 23–31 (2003)

5. Liu, S.-T.: Fuzzy Total Transportation Cost Measures for Fuzzy Solid Transportation Problem. Applied Math. and Computation 174, 927–941 (2006)
6. Liu, S.-T., Kao, C.: Solving Fuzzy Transportation Problems Based on Extension Principle. European J. Operational Research 153, 661–674 (2004)
7. Tada, M., Ishii, H.: An Integer Fuzzy Transportation Problem. Computers and Mathematics with Applications 31, 71–87 (1996)
8. Taha, H.A.: Operations Research. Macmillan, New York (1992)

24 Fuzzy Integer Programming

24.1 Introduction

We did not discuss fuzzy integers in Chapter 2 so we will do this in the next section. Then in Section 24.3 we look at a standard integer programming problem and then fuzzify it. We plan to use our fuzzy Monte Carlo method in Section 24.4 to obtain (approximate) solutions to this fuzzy integer programming problem. There have been a few papers on fuzzy integers and optimization problems and we refer the reader to ([1]-[5]).

24.2 Fuzzy Integers

A fuzzy integer $\overline{I}$ will have discrete membership function (equation (2.3) of Chapter 2)

$$\overline{I} = \{\frac{\mu_1}{n-r}, ..., \frac{\mu_r}{n-1}, \frac{1}{n}, \frac{\mu_{r+2}}{n+1}, ..., \frac{\mu_{2r+1}}{n+r}\}, \tag{24.1}$$

where n is an integer, r can be $1, 2, 3, ...$ and $0 < \mu_1 < ... < \mu_r < 1$, $1 > \mu_{r+2} > ... > \mu_{2r+1} > 0$. These are "triangular shaped" fuzzy integers since their membership functions monotonically increase (decrease) from zero to one (from one to zero). $\overline{I}$ would be approximately n. An example of approximately ten could be

$$\overline{I} = \{\frac{0.4}{9}, \frac{1}{10}, \frac{0.6}{11}\} \tag{24.2}$$

and approximately four might be

$$\overline{I} = \{\frac{0.2}{2}, \frac{0.7}{3}, \frac{1}{4}, \frac{0.6}{5}, \frac{0.3}{6}\}. \tag{24.3}$$

The arithmetic of fuzzy integers is done using the extension principle (Section 2.3.1) and using equations (2.4)-(2.7). For example consider

$$\overline{I}_1 = \{\frac{0.5}{0}, \frac{1}{1}, \frac{0.6}{2}\} \times \{\frac{0.7}{1}, \frac{1}{2}, \frac{0.3}{3}\} = \overline{I}_2. \tag{24.4}$$

Let $\overline{I}_3 = \overline{I}_1 \times \overline{I}_2$. Then

$$\overline{I}_3 = \{\frac{0.5}{0}, \frac{0.7}{1}, \frac{1}{2}, \frac{0.3}{3}, \frac{0.6}{4}, \frac{0.3}{6}\}, \tag{24.5}$$

because: (1) the value of zero can occur three ways; and (2) the value of two can happen two ways. So the membership value at zero is

$$max\{min\{\overline{I}_1(0), \overline{I}_2(1)\}, min\{\overline{I}_1(0), \overline{I}_2(2)\}, min\{\overline{I}_1(0), \overline{I}_2(3)\}\}. \tag{24.6}$$

This is no longer a "triangular shaped" fuzzy integer. A similar calculation is done for $\overline{I}_3(2)$.

Next we need to determine a way of deciding on $<$, $\leq$ and $\approx$ between fuzzy integers. We will use Buckley's Method in Section 2.6.1. Consider

$$\overline{I}_1 = \{\frac{0.3}{6}, \frac{1}{7}, \frac{0.6}{8}\} \tag{24.7}$$

and

$$\overline{I}_2 = \{\frac{0.4}{6}, \frac{0.8}{7}, \frac{1}{8}, \frac{0.5}{9}, \frac{0.2}{10}\}. \tag{24.8}$$

We use equation (2.43) for $v(\overline{I}_1, \overline{I}_2) = 1$ and $v(\overline{I}_2, \overline{I}_1) = 0.8$. This second value is 0.8 because for $x = 7$ we have the maximum of $[min(0.8, 1)$ for $y = 7$ and $min(0.8, 0.6)$ for $y = 8]$ is 0.8 which is the largest value for $x \leq y$. Having $\eta = 0.8$ means that $\overline{I}_1 \approx \overline{I}_2$ but $\eta = 0.9$ gives $\overline{I}_1 < \overline{I}_2$.

This is all that we will need about fuzzy integers for the rest of this chapter.

24.3 An Integer Programming Problem

Let us consider the following simple integer programming problem from ([6], p.309)

$$max\ Z = 6x_1 + 4x_2, \tag{24.9}$$

subject to

$$x_1 + x_2 \leq 5, \tag{24.10}$$

and

$$10x_1 + 6x_2 \leq 45, \tag{24.11}$$

for $x_1, x_2 \geq 0$ and integer. The optimal solution is $x_1 = 3$ and $x_2 = 2$ with max $Z = 26$.

24.4 A Fuzzy Integer Programming Problem

We now have the following fuzzy integer programming problem

$$max\ \overline{Z} = \overline{6}\overline{x}_1 + \overline{4}\overline{x}_2, \tag{24.12}$$

subject to

$$\overline{x}_1 + \overline{x}_2 \leq \overline{5}, \tag{24.13}$$

and

$$10\overline{x}_1 + 6\overline{x}_2 \leq \overline{45}, \tag{24.14}$$

for $\overline{x}_1$ and $\overline{x}_2$ non-negative fuzzy integers. Recall that a "bar" over a symbol means it is fuzzy. So the variables will be non-negative fuzzy integers, the parameters in the objective function are considered uncertain and are fuzzy integers and the right sides of the constraints are also uncertain and modeled as fuzzy integers. The parameters on the left side of the constraints are all considered known and crisp.

Values for the fuzzy parameters in the fuzzy integer programming problem are given in Table 24.1.

Table 24.1. Fuzzy Parameters in the Fuzzy Integer Programming Problem

Parameter	Fuzzy Integer
$\overline{6}$	$\{\frac{0.4}{5}, \frac{1}{6}, \frac{0.6}{7}\}$
$\overline{4}$	$\{\frac{0.5}{3}, \frac{1}{4}, \frac{0.4}{5}\}$
$\overline{5}$	$\{\frac{0.3}{4}, \frac{1}{5}, \frac{0.5}{6}\}$
$\overline{45}$	$\{\frac{0.2}{44}, \frac{1}{45}, \frac{0.7}{46}\}$

24.4.1 Fuzzy Monte Carlo Solution

We plan to apply our fuzzy Monte Carlo method to the fuzzy integer programming problem outlined above. We first need to decide on what type of fuzzy integers we will allow for $\overline{x}_1$ and $\overline{x}_2$. We will use two types of triangular shaped fuzzy integers. Because of the crisp optimal solution we keep the $\overline{x}_i$ in $[0, 10]$. The first type of fuzzy integer we can use is

$$\overline{x} = \{\frac{\mu_1}{n-1}, \frac{1}{n}, \frac{\mu_2}{n+1}\}, \tag{24.15}$$

for $0 < \mu_i < 1$, $i = 1, 2$ and $n \in \{1, 2, ..., 9\}$. The second type is

$$\overline{x} = \{\frac{\mu_1}{n-2}, \frac{\mu_2}{n-1}, \frac{1}{n}, \frac{\mu_3}{n+1}, \frac{\mu_4}{n+2}\}, \tag{24.16}$$

for $0 < \mu_1 < \mu_2 < 1$, $1 > \mu_3 > \mu_4 > 0$ and $n \in \{2, ..., 8\}$.

Now we randomly generate a fuzzy vector $\overline{v} = (\overline{x}_1, \overline{x}_2)$, with the fuzzy integers of either type in equations (24.15)-(24.16) and check if it is feasible, or satisfies equations (24.13)-(24.14). Let the vertex point (where the membership value is one) of $\overline{x}_1$ ($\overline{x}_2$) be at n_1 (n_2). We do not want $n_1 + n_2$ ($10n_1 + 6n_2$) to be much greater than 5 (45) or else it will not be feasible. We will leave it up to the reader on how they would randomly produce a sequence of fuzzy vectors $\overline{v}$ uniformly filling the search space. Assuming it is feasible we compute the objective function

$\overline{Z}$. Now we have to adjust the breaking of "ties" in Section 2.6.4 for discrete fuzzy sets. Let us first change the η value to 0.9.

Assume at some point in the simulation the maximum value of $\overline{Z}$ is $\overline{Z}^*$. The next generation of a feasible $\overline{v}$ gives $\overline{Z}_0$ for the objective function and assume that $\overline{Z}^* \approx \overline{Z}_0$. Let the smallest value in $\overline{Z}^*$ ($\overline{Z}_0$) which has membership one be z_1^* (z_{10}) and the largest value with membership one z_2^* (z_{20}). If $z_1^* < z_{10}$ we have $\overline{Z}^* < \overline{Z}_0$ and if $z_1^* > z_{10}$ then $\overline{Z}^* > \overline{Z}_0$. If $z_1^* = z_{10}$ we go on and do the same thing with z_2^* and z_{20}. In this way we can reduce the number of $\overline{Z}$ values considered the maximum.

After N iterations of the algorithm the results of the fuzzy Monte Carlo method will give (approximate) solutions for $\overline{x}_1$, $\overline{x}_2$ and $\overline{Z}$.

References

1. Herrera, F., Verdegay, J.L.: Fuzzy Almost Integer Variables in Integer Programming Problems. J. Fuzzy Mathematics 2, 259–270 (1994)
2. Herrera, F., Verdegay, J.L.: Three Models of Fuzzy Integer Linear Programming. European J. Operational Research 83, 581–593 (1995)
3. Herrera, F., Verdegay, J.L.: Approaching Fuzzy Integer Linear Programming. In: Fedrizzi, M., Kacprzyk, J., Roubens, M. (eds.) Interactive Fuzzy Optimization, pp. 79–91. Springer, Heidelberg (1991)
4. Herrera, F.: Models and Methods in Fuzzy Discrete Programming. In: Delgado, M., Kacprzyk, J., Verdegay, J.L., Vila, M.A. (eds.) Fuzzy Optimization: Recent Advances, pp. 255–271. Physica-Verlag, Heidelberg (1994)
5. Herrera, F., Verdegay, J.L.: Making Decisions on Fuzzy Integer Linear Programming Problems. In: Ruan, D. (ed.) Fuzzy Logic Foundations and Industrial Applications, pp. 147–164. Kluwer Academic, Dordrecht (1996)
6. Taha, H.A.: Operations Research. Macmillan, New York (1992)

25 Fuzzy Dynamic Programming

25.1 Introduction

Dynamic programming has become a very large field of applications. Any management science/operations research book will discuss this topic and have list of references on the topic. Fuzzy dynamic programming is dynamic programming with some of the parameters/variables fuzzy. Put "fuzzy dynamic programming" into your search engine and get around 500 hits. A few key references on fuzzy dynamic programming are ([1]-[5],[7]-[9]. The inventory control problems discussed in Chapters 20-22 are usually solved by dynamic programming. Sequential decision problems usually fall into the dynamic programming area. So, fuzzy dynamic programming might be used on the fuzzy inventory control problems in those chapters. However, we used Monte Carlo methods there and will do the same in this chapter.

We will look at a typical crisp dynamic programming problem in the next section. We will just set the problem up, but not go through the details of the dynamic programming solution. In Section 25.3 we fuzzify this problem and then plan to apply our fuzzy Monte Carlo method to obtain a fuzzy solution to this example problem.

25.2 A Dynamic Programming Problem

This problem is adapted from an example in ([6],p.361). Consider the design of an electronic device consisting of K main components. The K main components are arranged in series so that if one component fails the entire device fails. The reliability (probability of no failure) of the device can be improved by installing backup units for each component. We may add one, two, three,..., or N-1 backup units for each component. These backup units could be added in parallel to all components. We collect the data for the construction of the device and we compute R_{ik} = the reliability of the k^{th} component if it has $i-1$ backups in parallel, $1 \leq k \leq K$, $i = 1, 2, 3, ..., N$. The R_{ik} increase, nonlinearly, as i increases. There are costs involved in making this devise and we have a budget constraint. Let C_{ik} = be the total cost of the k^{th} component having $i-1$ backups. The budget constraint is that we have at most \$$B$ to spend. A design is $d = ((i_1, 1), ..., (i_K, K))$ where the first component has $i_1 - 1$ backups with i_1 in $\{1, 2, 3, ..., N\}$,...,the K^{th} component has $i_K - 1$ backups with i_K in $\{1, 2, 3, ..., N\}$. Let $\mathbf{D}$ be the set of all designs. The optimization problem is

$$max \prod_{s \in d} R_s, \tag{25.1}$$

subject to

$$\sum_{s \in d} C_s \leq B, \tag{25.2}$$

and

$$d \in \mathbf{D}, \tag{25.3}$$

where $s \in d$ means $s \in \{(i_1, 1), ..., (i_K, K)\}$ for $d = ((i_1, 1), ..., (i_K, K))$.

25.3 A Fuzzy Dynamic Programming Problem

We just fuzzify the crisp design problem in the previous section. We will use $N = K = 3$ and the fuzzy data is in Table 25.1. The cost values are all in multiples of \$1000. All the fuzzy numbers in this table are triangular fuzzy numbers. The budget constraint is approximately \$10,000 and we take $\overline{B} = (8,500/10,000/11,500)$. We will use Buckley's Method (Section 2.6.1) with $\eta = 0.9$ to decide on $\leq$ and $<$ between fuzzy numbers, and the rules for breaking ties in Section 2.6.4 to change $\approx$ between fuzzy numbers to $<$ or $>$.

The fuzzy optimization problem is

$$max \prod_{s \in d} \overline{R}_s, \tag{25.4}$$

subject to

$$\sum_{s \in d} \overline{C}_s \leq \overline{B}, \tag{25.5}$$

and

$$d \in \mathbf{D}. \tag{25.6}$$

Table 25.1. Fuzzy Reliability and Cost in the Monte Carlo Study

i	$\overline{R}_{i1}$	$\overline{C}_{i1}$	$\overline{R}_{i2}$	$\overline{C}_{i2}$	$\overline{R}_{i3}$	$\overline{C}_{i3}$
1	(0.4/0.6/0.8)	(0.5/1/1.5)	(0.5/0.7/0.9)	(2/3/4)	(0.4/0.5/0.6)	(1/2/3)
2	(0.7/0.8/0.9)	(1.7/2/2.3)	(0.7/0.8/0.9)	(4.5/5/5.5)	(0.5/0.7/0.9)	(2/4/6)
3	(0.85/0.9/0.95)	(2.8/3/3.2)	(0.85/0.9/0.95)	(5/6/7)	(0.85/0.9/0.95)	(3/5/7)

25.3.1 Fuzzy Monte Carlo Solution

We next want to apply our fuzzy Monte Carlo method to produce a design solution to this fuzzy optimization problem. Randomly generate a design d by randomly picking i_k in $\{1, 2, 3\}$ for $k = 1, 2, 3$. This design will be feasible if $\sum_{s \in d} \overline{C}_s \leq \overline{B}$. This constraint is easy to check since all the fuzzy numbers are triangular fuzzy numbers. Let $\sum_{s \in d} \overline{C}_s = (c_{d1}/c_{d2}/c_{d3})$. Then this constraint is true if $c_{d2} \leq 10,000$. So if $c_{d2} > 10,000$ we need to check and see if $\sum_{s \in d} \overline{C}_s \approx \overline{B}$ or $\sum_{s \in d} \overline{C}_s > \overline{B}$. If $\sum_{s \in d} \overline{C}_s \approx \overline{B}$ then $\sum_{s \in d} \overline{C}_s \leq \overline{B}$. Once we have a feasible design we compute the value of the fuzzy objective function. Now $\prod_{s \in d} \overline{R}_s$ will be a triangular shaped fuzzy number. Assume we are at some point in the simulation and the current maximum fuzzy reliability is $\overline{R}^* = \prod_{s \in d} \overline{R}_s$. The next feasible design produces fuzzy reliability $\overline{R}_0$. If $\overline{R}^* < \overline{R}_0$, then replace $\overline{R}^*$ with $\overline{R}_0$, otherwise delete $\overline{R}_0$. Since we are using the tie breaking rules we do not expect to get $\overline{R}^* \approx \overline{R}_0$.

The Monte Carlo method may be inefficient in having too many infeasible designs. But in this example there are only $3^3 = 27$ possible designs and the simulation should run quickly. For larger problems we would need to determine how to make mostly feasible designs. If we use the vertex values of all the triangular fuzzy numbers the resulting crisp problem has optimal design $d^* = ((2, 1), (1, 2), (3, 3))$ with maximum reliability $R^* = 0.504$ and total cost \$10,000.

The results of the fuzzy Monte Carlo method, after N iterations, will produce an optimal design of $d^* = ((x, 1), (x, 2), (x, 3))$ with maximum fuzzy reliability $\overline{R}^* \approx TBC$ and fuzzy cost $\overline{C} = TBC$ where TBC means "to be completed".

References

1. Esogbue, A.O., Kacprzyk, J.: Fuzzy Dynamic Programming: A Survey of Main Developments and Applications. Archives of Control Sciences 5, 39–59 (1996)
2. Kacprzyk, J.: Multistage Decision-Making under Fuzziness. Verlag TÜV, Reinland, Köln, Germany (1983)
3. Kacprzyk, J.: Fuzzy Dynamic Programming: Basic Issues. In: Delgado, M., Kacprzyk, J., Verdegay, J.L., Vila, M.A. (eds.) Fuzzy Optimization: Recent Advances, pp. 321–331. Physica-Verlag, Heidelberg (1994)
4. Kacprzyk, J., Esogbue, A.O.: Fuzzy Dynamic Programming: Main Developments and Applications. Fuzzy Sets and Systems 81, 31–45 (1996)
5. Li, L., Lai, K.K.: Fuzzy Dynamic Programming Approach to Hybrid Multiobjective Multistage Decision-Making Problems. Fuzzy Sets and Systems 117, 13–25 (2001)
6. Taha, H.A.: Operations Research. Macmillan, New York (1992)
7. Yoshida, Y. (ed.): Dynamical Aspects in Fuzzy Decision Making. Springer, Heidelberg (2001)

8. Weber, K., Sun, Z.: Fuzzy Stochastic Dynamic Programming for Marketing Decision Support. Int. J. Intelligent Systems 15, 763–783 (2000)
9. Xiong, Y., Rao, S.S.: A Fuzzy Dynamic Programming Approach for the Mixed-Discrete Optimization of Mechanical Systems. J. Mechanical Design 127, 1088–1099 (2005)

26 Fuzzy Project Scheduling/PERT

26.1 Introduction

PERT stands for “Project Evaluation and Review Technique”. A project defines a combination of interrelated activities (jobs) that must be completed in a certain order before the entire project can be completed. The project that we will concentrate on in this chapter is shown in Figure 26.1.

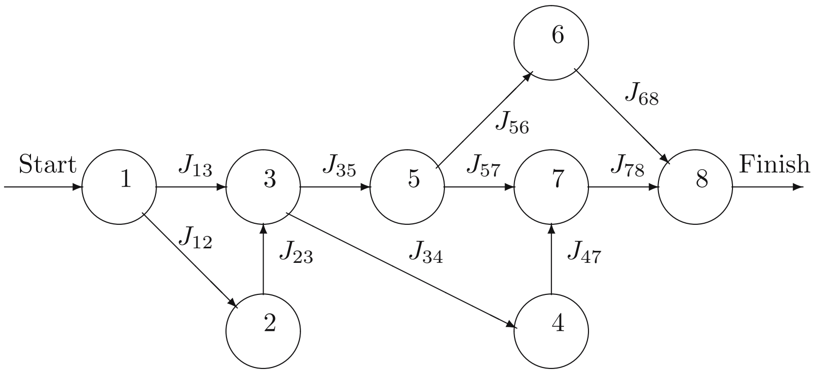


Fig. 26.1. Project Network

The jobs are interrelated in a sequence such that some jobs cannot start until others are finished. In Figure 26.1 we see that jobs J_{35} and J_{34} coming from node number 3, cannot start until all jobs leading into node 3, which are J_{13} and J_{23}, have been completed. In general, jobs require time and resources for completion. We will not discuss the allocation of resources to jobs in this chapter so the concept of “crashing” ([7],[13]) will not be used. So, attached to each job J_{ij} is only its time to completion $t_{ij} \geq 0$. Let T_{ij} denote a variable whose values t_{ij} are possible job times for job J_{ij}.

In general, job times are uncertain and in PERT T_{ij} is a random variable, while in fuzzy PERT, T_{ij} is a fuzzy variable. Let T_i be the time all jobs entering node number i are finished and is therefore the earliest time all jobs leaving this node may start. We wish to find T_8 giving the time of project duration. We see from Figure 26.1 that $T_8 = \max\{T_{68}, T_{78}\}$, $T_7 = \max\{T_{47}, T_{57}\}$, etc. In stochastic PERT, T_8 will be a random variable, so one would want to find its probability density function so that we can compute Prob$[T_8 \leq \lambda]$ for various possible due date times λ. We may have a constraint like Prob$[T_8 > \lambda] \leq 0.05$. However, in probabilistic PERT it is very difficult, in general, to find the probability density of project duration and one usually uses simulation to construct a histogram approximating the unknown density. In fuzzy PERT T_8 is a fuzzy variable and we wish to find the fuzzy set $\overline{M}_8$ describing project duration. We now argue that there is an algorithm to find $\overline{M}_8$ for both job times fuzzy numbers and for job times discrete fuzzy sets. For fuzzy PERT we will consider two cases: (1) job times are trapezoidal fuzzy numbers; and (2) job times are discrete fuzzy sets. We will only discuss our fuzzy Monte Carlo method for the case where job times are discrete fuzzy sets.

There are a number of other items usually computed in PERT such as floats, earliest start time for a job, latest start time for a job, etc. We will not be concerned with any of these computations and the reader may consult ([7],[13]) for probabilistic PERT, and ([1],[3]-[12],[14]-[15]), and the references in these papers, for fuzzy PERT/fuzzy CPM/fuzzy project scheduling. This chapter is based on [2].

26.2 Job Times Fuzzy Numbers

Let the fuzzy job time for T_{ij} be the trapezoidal fuzzy number $\overline{M}_{ij} = (a_{ij}/m^1_{ij}, m^2_{ij}/b_{ij})$. In the definition of $\overline{M}_{ij}$: (1) a_{ij} is the optimistic job time for J_{ij}, or the shortest time possible if everything goes well; (2) m^1_{ij}, m^2_{ij} are the lower, and upper, bounds respectively, of the most likely, or expected, job time; and (3) b_{ij} is the pessimistic job time for J_{ij}, or the longest time possible when everything goes wrong. We assume that $0 \leq a_{ij} < m^1_{ij} \leq m^2_{ij} < b_{ij}$ for all jobs. So job times may be triangular when $m^1_{ij} = m^2_{ij}$. These fuzzy numbers $\overline{M}_{ij}$ can be obtained from expert opinion and if so, one might want to only solicit the numbers a_{ij}, m^1_{ij}, m^2_{ij} and b_{ij} from experts.

To determine the fuzzy set for project duration we employ the extension principle (Section 2.4.1 of Chapter 2). Let F be a function that computes project duration given the job times for all jobs. So

$$F(t_{12}, t_{13}, ..., t_{78}) = T_8, \tag{26.1}$$

where $a_{ij} \leq t_{ij} \leq b_{ij}$ for all jobs. By the extension principle

$$\overline{M}_8 = F(\overline{M}_{12}, ..., \overline{M}_{78}), \tag{26.2}$$

and then

$$\overline{M}_8(x) = sup\{min(\overline{M}_{12}(t_{12}), .., \overline{M}_{78}(t_{78})) | F(t_{12}, .., t_{78}) = x\}, \tag{26.3}$$

where $\overline{M}_8$ is the fuzzy set for project duration.

We now present a computational method of computing $\overline{M}_8$. Let

$$L_{ij}(y) = (m^1_{ij} - a_{ij})y + a_{ij}, \tag{26.4}$$

$$R_{ij}(y) = (m^2_{ij} - b_{ij})y + b_{ij}, \tag{26.5}$$

for $0 \le y \le 1$. Let us assume we have a software package, which we will call $\mathcal{L}$, that computes project duration T_8 given a vector of job times $v = (t_{12}, ..., t_{78})$. The project in Figure 26.1 is fixed, all we input into $\mathcal{L}$ is a vector of job times v. Let $\mathcal{L}(v) = T_8$. For each fixed y in $[0,1]$ let $\mathcal{L}(v_l(y)) = t^1_8(y)$ and $\mathcal{L}(v_r(y)) = t^2_8(y)$, where $v_l(y) = (L_{12}(y), ..., L_{78}(y))$ and $v_r(y) = (R_{12}(y), ..., R_{78}(y))$. The $t^1_8(y)$ and $t^2_8(y)$ are project duration times. Then it was shown in [2] that

$$\overline{M}_8(x) = \begin{cases} y & : \text{ if } x = t^1_8(y),\ 0 \le y < 1, \\ 1 & : \text{ if } t^1_8(1) \le x \le t^2_8(1), \\ y & : \text{ if } x = t^2_8(y), 0 \le y < 1, \\ 0 & : \text{ otherwise.} \end{cases} \tag{26.6}$$

Example 26.2.1

The project network in Figure 26.1 with fuzzy job times is presented in Table 26.1.

We ran $\mathcal{L}$ on $L_{ij}(y)$ and $R_{ij}(y)$ for $y = 0, 0.1, \dots, 0.9, 1.0$. Using these results and equation (26.6) we constructed $\overline{M}_8$ shown in Figure 26.2.

Notice that the slope of $\overline{M}_8$ in Figure 26.2 changes at $t_8 = 20$ and at $t_8 = 27.5$ both corresponding to $\alpha = 0.5$. In both cases the slope changes due to different critical paths being created in the network. A critical path is a path through the network from start to finish so that the delay of the completion of any job on this path will cause a delay in the completion of the project.

Table 26.1. Fuzzy Job Times in Example 26.2.1

Job	a_{ij}	m^1_{ij}	m^2_{ij}	b_{ij}
J_{12}	4	5	5	6
J_{13}	7	8	8	10
J_{23}	2	4	4	5
J_{35}	3	4	5	6
J_{34}	3	5	5	6
J_{47}	3	5	5	6
J_{56}	2	5	6	7
J_{57}	4	5	5	6
J_{78}	4	5	5	8
J_{68}	3	5	5	6

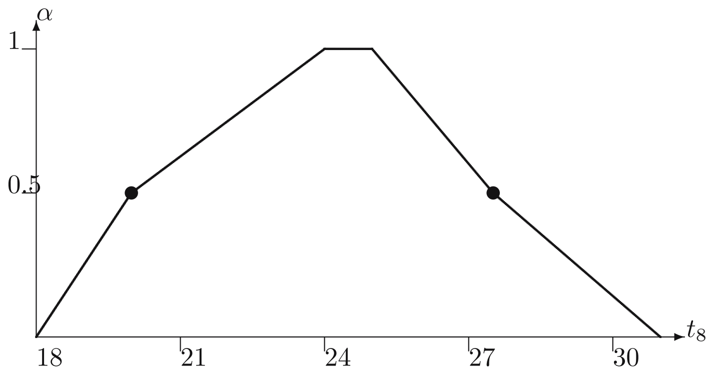


Fig. 26.2. Fuzzy Project Duration in Example 26.2.1

26.3 Job Times Discrete Fuzzy Sets

Now job times t_{ij} will be non–negative integers. A discrete fuzzy set for a job time might look like

$$\left\{\frac{0.2}{2}, \frac{0.5}{3}, \frac{1.0}{4}, \frac{0.6}{5}, \frac{0.1}{6}\right\}. \tag{26.7}$$

The computational algorithm to find the discrete fuzzy set for project duration is more complicated than the one for trapezoidal number job times and we will omit it. The interested reader needs to consult [2] for the complete details. However, the discrete fuzzy set for project duration is still computed from equation (26.3). Now we will construct the discrete fuzzy set for project time using our fuzzy Monte Carlo method.

26.3.1 Fuzzy Monte Carlo Method

We first need the discrete fuzzy sets for job times. They are:

1. $\overline{M}_{12} = \{\frac{0.5}{4}, \frac{1}{5}, \frac{0.5}{6}\}$;
2. $\overline{M}_{13} = \{\frac{0.5}{7}, \frac{1}{8}, \frac{0.67}{9}, \frac{0.33}{10}\}$;
3. $\overline{M}_{23} = \{\frac{0.33}{2}, \frac{0.67}{3}, \frac{1}{4}, \frac{0.5}{5}\}$;
4. $\overline{M}_{35} = \{\frac{0.5}{3}, \frac{1}{4}, \frac{1}{5}, \frac{0.5}{6}\}$;
5. $\overline{M}_{34} = \{\frac{0.33}{3}, \frac{0.67}{4}, \frac{1}{5}, \frac{0.5}{6}\}$;
6. $\overline{M}_{47} = \{\frac{0.33}{3}, \frac{0.67}{4}, \frac{1}{5}, \frac{0.5}{6}\}$;
7. $\overline{M}_{56} = \{\frac{0.25}{2}, \frac{0.5}{3}, \frac{0.75}{4}, \frac{1}{5}, \frac{1}{6}, \frac{0.5}{7}\}$;
8. $\overline{M}_{57} = \{\frac{0.5}{4}, \frac{1}{5}, \frac{0.5}{6}\}$;
9. $\overline{M}_{78} = \{\frac{0.5}{4}, \frac{1}{5}. \frac{0.75}{6}, \frac{0.5}{7}, \frac{0.25}{8}\}$;
10. $\overline{M}_{68} = \{\frac{0.33}{3}, \frac{0.67}{4}, \frac{1}{5}, \frac{0.5}{6}\}$.

We randomly generate vectors $v^{(k)} = (t_{12}^{(k)}, ..., t_{78}^{(k)})$ where $t_{12}^{(k)} \in \{4, 5, 6\}$,...,$t_{78}^{(k)} \in \{4, 5, 6, 7, 8\}$, for $k = 1, 2, 3, ...$ We want random vectors of non-negative integers here and not random vectors of fuzzy numbers. To obtain $v^{(k)}$ we may modify the procedure given in Section 3.5. Use our Sobol quasi-random number generator for random vector $v = (x_1, ..., x_{10})$ and then transform x_1 to be an integer in $\{4, 5, 6\}$ and set the result to $t_{12}^{(k)}$, transform x_2 to be an integer in $\{7, 8, 9, 10\}$ and set the result to $t_{13}^{(k)}$, etc.

Let $\mathcal{L}(v^{(k)}) = t_8^{(k)}$ project duration given job times in $v^{(k)}$. Next set

$$\mu_8^{(k)} = min\{\overline{M}(t_{ij}^{(k)})|v^{(k)} = (t_{12}^{(k)}, ..., t_{78}^{(k)})\}. \tag{26.8}$$

We may get multiple $v^{(k)}$ which produce the same project duration. Suppose we get $t_8^{(a)} = t_8^{(b)} = t_8^{(c)} = t_8$. Then $\mu_8 = \overline{M}(t_8) = max\{\mu_8^{(a)}, \mu_8^{(b)}, \mu_8^{(c)}\}$. Then we will use μ_8/t_8 in the discrete fuzzy set for project duration.

In this way we may construct a discrete fuzzy set for project duration

$$\overline{M}_8 = \{\frac{\mu_1}{18}, ..., \frac{\mu_{14}}{31}\}, \tag{26.9}$$

with the membership values μ_i determined by our fuzzy Monte Carlo method.

References

1. Bonnal, P., Gourc, D., Lacoste, G.: Where Do We Stand with Fuzzy Project Scheduling? J. Construction Engineering and Management 130, 114–123 (2004)
2. Buckley, J.J.: Fuzzy PERT. In: Evans, G.W., Karwowski, W., Wilhelm, M.R. (eds.) Applications of Fuzzy Set Methodologies in Industrial Engineering, pp. 103–114. Elsevier Science, Amsterdam (1989)
3. Chanas, S., Kamburowski, J.: The Use of Fuzzy Variables in PERT. Fuzzy Sets and Systems 5, 11–19 (1981)
4. Chanas, S., Zielinski, P.: Critical Path Analysis in a Network with Fuzzy Activity Times. Fuzzy Sets and Systems 122, 195–204 (2001)
5. Chanas, S., Zielinski, P.: The Computational Complexity of the Interval Critical Path Method. European J. Operational Research 136, 541–550 (2002)
6. Chen, S.-P.: Analysis of Critical Paths in a Project Network with Fuzzy Activity Times. European J. Operational Research 183, 259–442 (2007)
7. Davis, E.W., Moder, J.J., Phillps, C.R.: Project Management with CPM, PERT and Precedence Diagramming, 3rd edn. Van Nostrand Reinhold, N.Y. (1983)
8. Dubois, D., Fargier, H., Fortemps, P.: Fuzzy Scheduling: Modelling Flexible Constraints vs. Coping with Incomplete Knowledge. European J. Operational Research 147, 231–252 (2003)
9. Fargier, H., Galvagnon, V., Dubois, D.: Fuzzy PERT in Series-Parallel Graphs. In: Proc. FUZZ-IEEE 2000, San Antonio, Texas, May 7-10, 2000, pp. 717–722 (2000)
10. Guiffrida, A.L., Nagi, R.: Fuzzy Set Theory Applications in Production Management Research: A Literature Survey. J. Intelligent Manufacturing 9, 39–56 (1998)
11. Lin, F.-T., Yao, J.-S.: Fuzzy Critical Path Method Based on Signed-Difference Ranking and Statistical Confidence-Interval Estimates. J. Supercomputing 24, 305–325 (2003)
12. Nasution, S.H.: Techniques and Applications of Fuzzy Theory to Critical Path Methods. In: Leondes, C. (ed.) Fuzzy Theory Systems: Techniques and Applications, vol. 4, pp. 1561–1597. Academic Press, San Diego, Cal. (1999)

13. Taha, H.A.: Operations Research, 5th edn. MacMillan, N.Y. (1992)
14. Yao, J.-S., Lin, F.T.: Fuzzy Critical Path Method Based on Signed Distance Ranking of Fuzzy Numbers. IEEE Trans. Systems, Man and Cybernetics, Series A 30, 76–82 (2000)
15. Yao, J.-S., Lin, F.T.: Corrections to "Fuzzy Critical Path Method Based on Signed Distance Ranking of Fuzzy Numbers". IEEE Trans. Systems, Man and Cybernetics, Series A 30, 230 (2000)

27 Max/Min Fuzzy Function

27.1 Introduction

In this chapter we first look at finding solutions to

$$max/min \overline{Y} = f(\overline{X}), \tag{27.1}$$

where $\overline{X}$ is a triangular (shaped) fuzzy number in some interval I, f is a fuzzy function mapping fuzzy numbers into fuzzy numbers and $\overline{Y}$ is also a triangular shaped fuzzy number. As an example of this problem we consider a fuzzy EOQ (economic order quantity) problem to solve. To generate approximate solutions we plan to employ our fuzzy Monte Carlo method.

Then we consider finding solutions to

$$max/min \overline{Z} = f(\overline{X}, \overline{Y}), \tag{27.2}$$

where $\overline{X}$, $\overline{Y}$ and $\overline{Z}$ are all triangular (shaped) fuzzy numbers with $\overline{X}$ ($\overline{Y}$) in some interval I_x (I_y) and f is a fuzzy function mapping pairs of fuzzy numbers into fuzzy numbers. As an example of this problem we plan to use our fuzzy Monte Carlo method to obtain approximate solutions to a fuzzy profit function of two fuzzy variables.

27.2 Max/Min $f(\overline{X})$

We first briefly discuss the crisp EOQ model. We have adapted this description from ([2],p.246). See also [4]. The average weekly cost of ordering, paying for and storing merchandize is

$$C(x) = \frac{kD}{x} + cx + \frac{hx}{2}, \tag{27.3}$$

where x is the order quantity, k is the cost of placing the order (the same no matter how large x is), D is the weekly demand, c is the purchase cost in \$ per unit and h is the storage cost (including space, utilities, insurance, security,...). The object is to find x to minimize $C(x)$. The solution is, after setting the derivative to zero,

$$x^* = \sqrt{\frac{2kD}{2c+h}}, \tag{27.4}$$

from which we may compute the minimum weekly cost.

These parameters k, D and h are difficult to estimate. So we will use fuzzy estimators [1] and then $\overline{k}$, $\overline{D}$ and $\overline{h}$ are all fuzzy numbers expressing the uncertainty in their values. We assume that c is a known crisp number. To complete the fuzzification of the EOQ model we will also let the order quantity $\overline{x}$ be a fuzzy number. We wish to find $\overline{x}$ in some interval I to minimize

$$\overline{C}(\overline{x}) = \frac{\overline{k}\,\overline{D}}{\overline{x}} + c\overline{x} + \frac{\overline{h}\overline{x}}{2}. \tag{27.5}$$

We may have to defuzzify our solution $\overline{x}^*$ to get a usable order quantity.

For literature on the fuzzy EOQ model see Chapter 20.

Example 27.2.1

Let $\overline{k} = (8/10/12)$, $\overline{D} = (40/50/60)$, $\overline{h} = (0.06/0.10/0.14)$ and $c = \$5.00/unit$. These fuzzy numbers are all triangular fuzzy numbers. Defuzzify the fuzzy numbers to $k = 10$, $D = 50$ and $h = 0.10$ and then the crisp EOQ has solution $x^* = 100$ with minimum cost \$260 per week. So we pick the interval for the fuzzy order $\overline{x}$ to be $I = [0, 150]$.

We randomly generate a QBGFNs (Chapter 4) $\overline{x}$ and evaluate equation (27.5) producing the current value of $\overline{C}_0$ for the fuzzy cost. Let $\overline{C}^*$ be the previous minimum for fuzzy cost. If $\overline{C}_0 < \overline{C}^*$, then discard $\overline{C}^*$ and set minimum fuzzy cost to $\overline{C}_0$. If $\overline{C}_0 > \overline{C}^*$, then discard $\overline{C}_0$. If $\overline{C}_0 \approx \overline{C}^*$, let $\overline{C}_0 \approx (c_{01}/c_{02}/c_{03})$ and $\overline{C}^* \approx (c_1^*/c_2^*/c_3^*)$. If $c_{02} < c_2^*$ we keep $\overline{C}_0$, if $c_{02} > c_2^*$ we discard $\overline{C}_0$. If $c_{02} = c_2^*$, then use c_{03} and c_3^*. If $c_{02} = c_2^*$ and $c_{03} = c_3^*$, then use c_{01} and c_1^*. If the supports and cores are equal randomly discard one of them. Continue this process until it appears that we are close to the minimum fuzzy cost. Let the minimum fuzzy cost be $\overline{C}^*$ with corresponding fuzzy order $\overline{x} = \overline{x}^*$. For $<$ and $\approx$ between fuzzy numbers we will use Buckley's Method (Section 2.6.1 in Chapter 2) using $\eta = 0.9$.

27.3 Max/Min $f(\overline{X}, \overline{Y})$

We first briefly discuss the crisp profit model. We have adapted this description from ([3],p.861). The total profit from one acre of a certain crop depends on the amount spent (x) on fertilizer and the amount spent (y) on seed according to the model

$$P(x, y) = c(a_1x^2 + a_2xy + a_3x + a_4y^2 + a_5y + a_6), \tag{27.6}$$

for parameters c and a_i, $i = 1, 2, ..., 6$. Given values for all the parameters we wish to find x and y to maximize $P(x, y)$. Crisp values for these parameters have been estimated. They are $c = 0.01$, $a_1 = -1$, $a_2 = 3$, $a_3 = 160$, $a_4 = -5$, $a_5 = 200$ and $a_6 = 2600$. Then after setting the partial derivatives to zero we obtain the optimal solution of $x^* = 200$, $y^* = 80$ with maximum profit \$266.

All these parameters were estimated from data or from experts. We therefore use fuzzy estimators and obtain $\overline{c}$ and $\overline{a}_i$, $i = 1, ..., 6$, all triangular fuzzy numbers. We will also let x and y be fuzzy numbers. We wish to find $\overline{x}$ ($\overline{y}$) in some interval I_x (I_y) to maximize

$$\overline{P}(\overline{x}, \overline{y}) = \overline{c}(\overline{a}_1 \overline{x}^2 + ... + \overline{a}_6). \tag{27.7}$$

We may have to defuzzify our solutions $\overline{x}^*$ and $\overline{y}^*$ to get usable values. We now can use our fuzzy Monte Carlo method to obtain approximate solutions.

Example 27.3.1

Let $\overline{c} = (0.005/0.01/0.05)$, $\overline{a}_1 = (-1.5/-1/-0.5)$, $\overline{a}_2 = (2/3/4)$, $\overline{a}_3 = (150/160/170)$, $\overline{a}_4 = (-5.5/-5/-4.5)$, $\overline{a}_5 = (170/200/230)$ and $\overline{a}_6 = (2500/2600/2700)$. We choose the interval $I_x = [0, 300]$ for $\overline{x}$ and $I_y = [0, 120]$ for $\overline{y}$.

We randomly generate QBGFNs (Chapter 4) $\overline{x}, \overline{y}$ and evaluate equation (27.7) producing the current value of $\overline{P}_0$ for the fuzzy profit. Let $\overline{P}^*$ be the previous maximum for fuzzy profit. If $\overline{P}_0 > \overline{P}^*$, then discard $\overline{P}^*$ and set maximum fuzzy profit to $\overline{P}_0$. If $\overline{P}_0 < \overline{P}^*$, then discard $\overline{P}_0$. If $\overline{P}_0 \approx \overline{P}^*$, let $\overline{P}_0 \approx (p_{01}/p_{02}/p_{03})$ and $\overline{P}^* \approx (p_1^*/p_2^*/p_3^*)$. If $p_{02} > p_2^*$ we keep $\overline{P}_0$, if $p_{02} < p_2^*$ we discard $\overline{P}_0$. If $p_{02} = p_2^*$, then use p_{03} and p_3^*. If $p_{02} = p_2^*$ and $p_{03} = p_3^*$, then use p_{01} and p_1^*. If the supports and cores are equal randomly discard one of them. Continue this process until it appears that we are close to the maximum fuzzy profit. Let the maximum fuzzy profit be $\overline{P}^*$ with corresponding fuzzy variables $\overline{x} = \overline{x}^*$ and $\overline{y}^*$. For $>$ and $\approx$ between fuzzy numbers we will use Buckley's Method (Section 2.6.1 in Chapter 2) using $\eta = 0.9$.

References

1. Buckley, J.J.: Fuzzy Probability and Statistics. Springer, Heidelberg (2006)
2. Finney, R.L., Thomas Jr., G.B.: Calculus, 2nd edn. Addison-Wesley, Reading, Mass. (1994)
3. Lial, M.L., Hungerford, T.W., Miller, C.M.: Mathematics with Applications, 6th edn. Harper Collins, New York (1995)
4. Taha, H.A.: Operations Research, 5th edn. Macmillan, New York (1992)

Part IV: Summary, Conclusions, Future Research

28 Summary, Conclusions, Future Research

28.1 Summary

In this book we hoped to convince the reader that Monte Carlo methods can be useful in generating approximate solutions to fuzzy optimization problems. In a Monte Carlo procedure we randomly produce N possible (feasible) solutions to an optimization problem, subject to some criteria we keep only the "best" feasible solutions, and as N grows larger and larger we converge to an optimal solution. Having a feasible solution usually means it satisfies the constraints to the problem and these constraints usually involve equalities and inequalities. Monte Carlo methods are known to be very inefficient and are seldom used in crisp optimization problems since these problems usually have their own efficient solution algorithms. However, fuzzy optimization problems usually do not have their own efficient solution algorithms so Monte Carlo methods become more important in fuzzy optimization.

Monte Carlo methods in fuzzy optimization require us to randomly generate N feasible solutions. These feasible solutions will be fuzzy numbers or fuzzy vectors. The fuzzy numbers we use are triangular fuzzy numbers, trapezoidal fuzzy numbers or quadratic fuzzy numbers (coded QBGFNs in the book). A quadratic fuzzy number is a triangular shaped fuzzy number whose sides are described by quadratic functions. Fuzzy vectors are vectors composed of fuzzy numbers. We need to produce random sequences of fuzzy numbers and random sequences of fuzzy vectors. How we do this is explained in detail in Chapter 4. We use streams of crisp quasi-random numbers discussed in Chapter 3 to produce sequences of random fuzzy numbers/vectors. Randomness tests on our sequences of random fuzzy numbers/vectors is presented in Chapter 5.

Next we need to check to see if our random fuzzy number/vector is feasible. When the constraints involve $<$, or $\leq$, between fuzzy numbers we use three methods to evaluate inequalities between fuzzy numbers: (1) Buckley' Method in Section 2.6.1; (2) Kerre's Method in Section 2.6.2; and (3) Chen's Method in Section 2.6.3. Of course, our computer programs may be altered to employ any method of evaluating inequalities between fuzzy numbers. Once we have a sequence of random feasible solutions we need to pick the “best” ones. Most of the fuzzy optimization problems we study in the applications chapters (Chapters 6 - 16) involve finding the maximum/minimum of a fuzzy function subject to fuzzy constraints. We use the same methods (Buckley, Kerre, Chen) to find the maximum/minimum of a collection of fuzzy numbers. A key property that a Monte Carlo method must have, to be able to converge on an optimal solution, is that it uniformly fill the search space. We argue that our procedure has this property in Section 5.3.

A major problem in fuzzy Monte Carlo is to decide on intervals $[a_i, b_i]$, $1 \leq i \leq m$, for our random fuzzy numbers $\overline{X}_i$, $i = 1, 2, 3, ..., m$. That is, randomly generate $\overline{X}_i \in [a_i, b_i]$ all i. If an interval is too small we can miss a good solution. If the intervals are too big we can produce many infeasible candidates. This problem is discussed starting in Chapter 6 and you can use the key phrase “intervals for Monte Carlo” in the Index.

We applied our fuzzy Monte Carlo method to fully fuzzified linear programming (Chapters 7-9), solving fuzzy equations (Chapter 10), to fuzzy regression (Chapters 11-14), fuzzy game theory (Chapter 15) and fuzzy queuing theory (Chapter 16). In some cases these fuzzy optimization problems had approximate optimal solutions from previous publications. In all cases, except one situation as explained in Chapter 11, our Monte Carlo method obtained a better approximate solution.

The computer time can be quite long. In one case it was 68 hours for $N = 100,000$ in Chapter 15. So we suggested doing it in parallel. Use 10 computers each for $N = 100,000$ to get a run of $1,000,000$.

There are many fuzzy optimization problems we have not yet applied our fuzzy Monte Carlo method to calculate an approximate solution and some of these are outlined in Chapters 17-27.

28.2 Future Research

Future research could be involved with continuing to use our fuzzy Monte Carlo method on fuzzy linear programming problems and on fuzzy regression problems. But more importantly it would involve attacking those fuzzy optimization problems discussed in Chapters 17-27. Of course, there are fuzzy optimization problems not presented in this book that may also be (approximately) solved by Monte Carlo.

We encourage others to review and extend the work which we have begun. In preparation for Chapters 3-10 of this book we have developed seven programs (Table 28.1). Each is written in Visual C++; some use the OpenGL graphics library. We offer the source and data files freely to those wanting to do further research. Contact Leonard Jowers (`LJJowers@uab.edu`) for more information. A number of MATLAB programs were created for Chapters 11-16. One is contained in Chapter 14. Others may be obtained by contacting James J.Buckley (`buckley@math.uab.edu`).

Table 28.1. Programs Available from Book Preparation

Program Name	Description	Chapter
`RNGenerator`	generates homogeneous files of crisp random numbers	3
`ComparisonOfBuckleys`	pair-wise Buckley comparison for various thresholds	4
`ComparisonOfCompares`	pair-wise Buckley, Kerre, Chen FN comparisons	4
`FuzzyRunsTest`	performs fuzzy runs test	5
`FuzzyFrequencyTest`	performs fuzzy frequency test	5
`FuzzyMonteCarlo`	performs fuzzy Monte Carlo	6-10
`Plotter`	creates plots of results file	6-10

28.3 Conclusions

We believe that we have convinced the reader that fuzzy Monte Carlo methods are worthy of research and application to fuzzy optimization. We also believe that processing capabilities are mature enough to support fuzzy Monte Carlo optimizations. Fast processors and new computer architectures (e.g., technologies behind simultaneous multithreading, dual-core and cell processors) make these types of research more feasible for researchers not having access to supercomputers. There are surely hidden opportunities in this work.

Index

List of Figures

List of Tables