

# ファジィベクトルの二項演算用 プログラムパッケージの開発

福田 得 夫

## Abstract

本論文では、ファジィ数やファジィベクトルの二項演算と、その結果得られるファジィ集合の性質について考察する。さらに、ファジィ確率システム等のファジィ数およびファジィベクトルを使用するシミュレーション実験において活用できるように、得られた結果を用いてこれらの二項演算を実現するコンピュータプログラムパッケージを開発する。

## 1 緒 言

本論文では、ファジィ数の加算・減算・乗算・除算の四則計算およびファジィベクトルの加算・減算・内積計算と、その結果得られるファジィ集合（ファジィ数、ファジィベクトル）の性質について考察することと、得られた結果を活用することによって、これらの二項演算を効率良く実行できるプログラムパッケージを開発することを目的とする。

最初に、多値論理の立場よりファジィベクトルを定義し、ファジィ数は1次元ファジィベクトルとして取り扱うことにする。このようにすることによって、ファジィ数およびファジィベクトル間の二項演算が拡張原理

[1] により合理的に定義することが可能になる。さらに数値計算に便利な形式で演算を定義するために、ファジィ集合の集合表現 [2] を採用する。なお、ファジィ数については、例えば [3], [4], [5] など既に多くの研究がある。本論文でのファジィ数の定義は、これらの研究で用いられているものと本質的には同じものである。

第2節および第3節においてファジィ集合の集合表現と拡張原理を本論文の目的に適合した形式で概説する。またファジィベクトルを多値論理の立場より定義する。第4節においては、拡張原理を原いてファジィ数およびファジィベクトル間の演算を定義し、さらに演算結果として得られるファジィ集合の若干の性質について考察する。また、第5節では、第4節の結果を利用して、ファジィ数やファジィベクトルの二項演算のための数値計算用プログラムパッケージをプログラム言語 C++ を用いて作成する。

## 2 ファジィベクトルの集合表現

ファジィ集合  $\tilde{u}$  は  $\tilde{u} = (X, (\tilde{u}), s)$  で与えられる [6]。ここで  $X$  は基礎空間と呼ばれ、メンバーシップ関数  $(\tilde{u})$  は  $X$  から区間  $[0, 1]$  への写像である。さらに  $s$  は基礎空間  $X$  から全ての「陳述」(statement) の集合から構成されている「議論の対象領域」(universe of discourse)  $P$  への写像であり、 $X$  のすべての要素  $x$  に命題  $s(x)$  を与える。また、メンバーシップ関数  $(\tilde{u})(x)$  の値は命題  $s(x)$  の真理値として与えられるものとする。すなわち

$$(\tilde{u})(x) = t(s(x)) \quad (t(*): * \text{の真理値関数}).$$

本論文では基礎空間  $X$  は  $n$  次元ユークリッド空間  $R^n$  に限定する。またファジィ集合  $\tilde{u}$  のレベル  $\alpha$  での強カット  $L_\alpha \tilde{u}$  およびレベル集合  $L_\alpha \tilde{u}$  は、それぞれ

$$L_a \tilde{u} = \{x \in X \mid (\tilde{u})(x) > \alpha\} \quad \text{for } \forall \alpha \in [0, 1) \quad (2.1)$$

$$L_a \tilde{u} = \{x \in X \mid (\tilde{u})(x) \geq \alpha\} \quad \text{for } \forall \alpha \in (0, 1] \quad (2.2)$$

によって与えられる。さらにファジィ集合  $\tilde{u}$  の台 (support) は

$$\text{supp } \tilde{u} = \text{closure of } \{x \in X \mid (\tilde{u})(x) > 0\} \quad (2.3)$$

である。なお、

$$0 < \alpha \leq \beta < 1 \Rightarrow \tilde{u}_\beta \subseteq \tilde{u}_\alpha \subseteq X \quad (2.4)$$

と

$$(\tilde{u})(t) = \sup \{\alpha \mid \alpha \in (0, 1), t \in \tilde{u}_\alpha\} \quad (2.5)$$

を満たす集合族  $\{\tilde{u}_\alpha \mid \alpha \in (0, 1)\}$  をファジィ集合  $\tilde{u}$  の集合表現 (set representation of fuzzy set) と呼ぶことにする。強カット集合族  $\{L_a \tilde{u} \mid \alpha \in (0, 1)\}$  およびレベル集合族  $\{\tilde{u}_\alpha \mid \alpha \in (0, 1)\}$  は  $\tilde{u}$  の集合表現であり、また  $\alpha \in (0, 1)$  に対して

$$L_a \tilde{u} \subseteq \tilde{u}_\alpha \subseteq L_a \tilde{u} \quad (2.6)$$

を満たすことが知られている [2].

説明の都合上、以下の議論では次のような性質を満たすファジィ集合族を  $\mathcal{G}(R^n)$  で表しコンパクトファジィ集合族ということにする。すなわち  $\mathcal{G}(R^n)$  の任意の要素であるファジィ集合  $\tilde{u}$  において

- (i) そのメンバーシップ関数の値の最大値は 1.
- (ii) 全ての要素がコンパクト集合<sup>1)</sup>である集合表現が存在する.
- (iii)  $\text{supp } \tilde{u}$  はコンパクト.

---

1) 本論文では基礎空間は  $n$  次元ユークリッド空間に限定しているのでコンパクト集合は有界閉集合と同じである.

さらに  $\mathcal{G}(R^n)$  の要素であり、すべての要素がコンパクトでかつ凸である集合から構成される集合表現をもつファジィ集合の族を  $\mathcal{G}_c(R^n)$  と表すことにする。本論文では  $\mathcal{G}_c(R^n)$  に属するファジィ集合を ( $n$  次元) ファジィベクトルと呼ぶことにする。また 1 次元ファジィベクトルはファジィ数である。

### 3 ファジィベクトルの拡張原理

命題

$$\begin{aligned} s(u) &= \{u \text{ は } \tilde{u} \text{ の要素}\} & (\tilde{u} : \text{ファジィベクトル}) \\ &= \{u \in \tilde{u}\} \end{aligned} \quad (3.1)$$

の受容性 (acceptability) はその真理値関数を評価することで与えられるものとし、その値はメンバーシップ関数の値に一致するものとする。すなわち、

$$\text{acc } s(u) = t(s(u)) = (\tilde{u})(u) \quad (3.2)$$

である。ただし  $\text{acc}*$  は “\*” の受容性を表す。いま

$$\phi : R^m \times R^n \rightarrow R^l$$

を  $R^m \times R^n$  から  $R^l$  への写像とする。そうすると  $U \times V (U \subset R^m, V \subset R^n)$  の像  $\phi(U, V)$  は

$$\phi(U, V) = \{w \in R^l \mid u \in U; v \in V; \text{かつ } w = \phi(u, v)\} \quad (3.3)$$

で与えられる  $R^l$  の部分集合となる。ここで  $U$  と  $V$  を通常の集合ではなくファジィベクトル (ファジィ集合)  $\tilde{u}, \tilde{v}$  とすると、命題

$$\begin{aligned} s_\phi(w) &= \{w \text{ は } \phi(\tilde{u}, \tilde{v}) \text{ の要素}\} \\ &= \{w \in \phi(\tilde{u}, \tilde{v})\} \end{aligned} \quad (3.4)$$

の受容性は (3.2) と Bellmann and Giertz [7] の定理を用いることにより

$$\begin{aligned} \text{acc } s_\phi(w) &= t\left(\bigvee_{(u, v) \in X_1 \times X_2} \{u \in \tilde{u}; v \in \tilde{v}; \text{かつ } w = \phi(u, v)\}\right) \\ &= \sup \{t((u \in \tilde{u}) \wedge (v \in \tilde{v}) \wedge (w = \phi(u, v))) \mid (u, v) \in R^m \times R^n\} \\ &= \begin{cases} \sup \{\min \{(\tilde{u})(u), (\tilde{v})(v)\} \mid (u, v) \in \phi^{-1}(w) \\ \quad w = \phi(u, v) \text{ を満たす } u \text{ と } v \text{ が存在するとき} \\ 0 \quad \quad \quad \text{その他の場合} \end{cases} \end{aligned} \quad (3.5)$$

と評価することができる。したがって、ファジィベクトル  $\tilde{u}$  と  $\tilde{v}$  による写像  $\phi(\tilde{u}, \tilde{v})$  の値 (ファジィ集合) を次のように定義することができる。

**定義 3.1**  $\tilde{u}$  と  $\tilde{v}$  をファジィベクトルとし、 $\phi$  を  $R^m \times R^n$  から  $R^l$  への写像とする。このときファジィ集合  $\phi(\tilde{u}, \tilde{v})$  は

$$\phi(\tilde{u}, \tilde{v}) = (R^l, (\phi(\tilde{u}, \tilde{v})), s_\phi) \quad (3.6)$$

となる。ここで  $\phi(\tilde{u}, \tilde{v})$  のメンバーシップ関数  $(\phi(\tilde{u}, \tilde{v}))(w)$  は全ての  $w \in R^l$  に対して

$$(\phi(\tilde{u}, \tilde{v}))(w) = \begin{cases} \sup \{\min \{(\tilde{u})(u), (\tilde{v})(v)\} \mid (u, v) \in \phi^{-1}(w) \\ \quad w = \phi(u, v) \text{ を満たす } u \text{ と } v \text{ が存在するとき} \\ 0 \quad \quad \quad \text{その他の場合} \end{cases} \quad (3.7)$$

で与えられる。また上記のように定義されるファジィ集合  $\phi(\tilde{u}, \tilde{v})$  に対して次の結果を得ることができる [8]。

**定理 3.1**  $\tilde{u}$  と  $\tilde{v}$  をファジィベクトルとし、 $\phi$  を  $R^m \times R^n$  から  $R^l$  への写像

とする. また,  $\{\tilde{u}_\alpha \mid \alpha \in (0, 1)\}$  と  $\{\tilde{v}_\alpha \mid \alpha \in (0, 1)\}$  をそれぞれ  $\tilde{u}$  と  $\tilde{v}$  の集合表現とする. そうすると

$$\{\phi(\tilde{u}_\alpha, \tilde{v}_\alpha) \mid \alpha \in (0, 1)\} \quad (3.8)$$

は  $\phi(\tilde{u}, \tilde{v})$  の集合表現となる. さらに, すべての  $\alpha \in [0, 1)$  に対して

$$L_\alpha \phi(\tilde{u}, \tilde{v}) = \phi(L_\alpha \tilde{u}, L_\alpha \tilde{v}) \quad (3.9)$$

が成立する. また  $\phi$  が連続関数である場合は  $\phi(\tilde{u}, \tilde{v}) \in \mathcal{D}(R^l)$  であり, すべての  $\alpha \in (0, 1]$  に対して

$$L_\alpha \phi(\tilde{u}, \tilde{v}) = \phi(L_\alpha \tilde{u}, L_\alpha \tilde{v}). \quad (3.10)$$

#### 4 ファジィベクトルの二項演算

定義 3.1 を用いることにより, 次のようなファジィベクトル間の演算を定義することができる.

**ファジィ線形結合**  $n$  次元ファジィベクトル  $\tilde{u}$  と  $\tilde{v}$  の線形結合は

$$x\tilde{u} + y\tilde{v} = (R^m, (x\tilde{u} + y\tilde{v}), s_{lc})$$

で与えられる. ここで  $x$  と  $y$  は任意の実数であり, メンバーシップ関数  $(x\tilde{u} + y\tilde{v})$  は

$$(x\tilde{u} + y\tilde{v})(w) = \sup \{ \min \{ (\tilde{u})(u), (\tilde{v})(v) \} \mid w = xu + yv \} \} \quad (4.1)$$

で, また  $s_{lc}$  は

$$s_{lc}(w) = \{w \in x\tilde{u} + y\tilde{v}\}$$

という命題である.

ファジィ内積  $n$  次元ファジィベクトル  $\tilde{u}$  と  $\tilde{v}$  の内積  $\tilde{u}'\tilde{v}$  は

$$\tilde{u}'\tilde{v} = (R, (\tilde{u}'\tilde{v}), s_{ip})$$

で与えられる。ここで

$$(\tilde{u}'\tilde{v})(w) = \sup \{ \min \{ (\tilde{u})(u), (\tilde{v})(v) \} \mid w = u'v \} \quad (4.2)$$

であり,  $s_{ip}$  は

$$s_{ip}(w) = \{w \in \tilde{u}'\tilde{v}\}$$

という命題である。

上記のように定義されるファジィベクトルの線形結合と内積に対して、次のような結果を得ることができる。

**定理 4.1**  $n$  次元ファジィベクトル  $\tilde{u}$  と  $\tilde{v}$  に対して、その線形結合および内積は、各々ファジィベクトルおよびファジィ数となる。すなわちそれぞれ  $\mathcal{G}_c(R^n)$  および  $\mathcal{G}_c(R)$  の要素となる。また

$$L_a(x\tilde{u} + y\tilde{v}) = xL_a\tilde{u} + yL_a\tilde{v}, L_a(x\tilde{u}' + y\tilde{v}') = xL_a\tilde{u}' + yL_a\tilde{v}' \quad (4.3)$$

と

$$L_a(\tilde{u}'\tilde{v}) = (L_a\tilde{u})'(L_a\tilde{v}), L_a(\tilde{u}'\tilde{v}') = (L_a\tilde{u})'(L_a\tilde{v}') \quad (4.4)$$

が成立する。さらにファジィベクトルの内積に関しては

$$\partial(L_a(\tilde{u}'\tilde{v})) = \{w \mid w = u'v; u \in \partial(L_a\tilde{u}); v \in \partial(L_a\tilde{v})\} \quad (4.5)$$

が成立する。ただしここで  $\partial(*)$  は  $*$  の境界である。

[証明] (i)  $n$  次元ベクトルの線形結合に関する結果については、既に

[8] で証明されているので、ここではファジィ内積  $\tilde{u}'\tilde{v}$  に関する結果のみの証明の概略を述べることにする。  $n$  次元ベクトルの内積は  $R^n \times R^n$  から  $R$  への連続写像であるから、定理 3.1 より明らかに  $\tilde{u}'\tilde{v} \in \mathcal{G}(R)$  であり、また (4.4) が成立する。いま  $\{\tilde{u}_\alpha \mid \alpha \in (0, 1)\}$  と  $\{\tilde{v}_\alpha \mid \alpha \in (0, 1)\}$  を、それぞれ  $\tilde{u}$  と  $\tilde{v}$  のコンパクトで凸な集合表現とする。そうすると直積集合  $\tilde{u}_\alpha \times \tilde{v}_\alpha$  は弧状連結となるから

$$\tilde{u}'_\alpha \tilde{v}_\alpha = \{u'v \mid u \in \tilde{u}_\alpha, v \in \tilde{v}_\alpha\} \quad (4.6)$$

もまたコンパクトで弧状連結な集合となる。このことは  $\tilde{u}'_\alpha \tilde{v}_\alpha$  がコンパクトな凸集合（有界な閉区間）であることを示している。

(ii)  $\tilde{u}, \tilde{v} \in \mathcal{G}_c(R^n)$  であるとき  $L_\alpha \tilde{u}, L_\alpha \tilde{v}$  はそれぞれ  $\tilde{u}$  と  $\tilde{v}$  のコンパクトで凸な集合表現であることを示すことができる [8]。したがって (i) より  $L_\alpha(\tilde{u}'\tilde{v})$  は有界閉区間である。ここで  $u$  は  $L_\alpha \tilde{u}$  の任意の内点、 $v$  は  $L_\alpha \tilde{v}$  の点とし、

$$z = u'v \in \partial\{L_\alpha(\tilde{u}'\tilde{v})\} \quad (4.7)$$

であるものとする。そうすると  $L_\alpha \tilde{u}$  が凸であるので

$$u = \lambda u_1 + (1-\lambda)u_2 \quad 0 < \lambda < 1 \quad (4.8)$$

を満たす  $L_\alpha \tilde{u}$  の点  $u_1$  と  $u_2$  が存在し

$$z = u'v = \lambda u'_1 v + (1-\lambda)u'_2 v \quad (4.9)$$

となる。したがって、

$$\begin{cases} z_{\min} = \min\{u'_1 v, u'_2 v\} \\ z_{\max} = \max\{u'_1 v, u'_2 v\} \end{cases} \quad (4.10)$$

とすると

$$\min\{L_\alpha(\tilde{u}'\tilde{v})\} \leq z_{\min} < z < z_{\max} \leq \max\{L_\alpha(\tilde{u}'\tilde{v})\} \quad (4.11)$$

が得られることになる。ところが前述したように  $L_a(\bar{u}'\bar{v})$  は有界閉区間であり、 $z$  は  $L_a(\bar{u}'\bar{v})$  の境界上の点、すなわち閉区間  $L_a(\bar{u}'\bar{v})$  の最大値であるかまたは最小値のいずれかであるが、これは (4.11) と矛盾する。また  $u$  が  $L_a\bar{u}$  の境界上の点、 $v$  が  $L_a\bar{v}$  の内点のときも同様の手順により矛盾が生じることが示せるから、(4.7) の  $z$  を構成する  $u$  と  $v$  は

$$u \in \partial L_a\bar{u} \quad \text{かつ} \quad v \in \partial L_a\bar{v}. \quad (4.12)$$

でなければならないことがわかる。(証明終)

$\bar{u}$  と  $\bar{v}$  が 1 次元ファジィベクトルすなわちファジィ数である場合には、除算を定義することができる。

ファジィ除算  $\bar{u}$  をファジィ数、 $\bar{v}$  を正または負のファジィ数<sup>2)</sup>とする。このときファジィ除算は

$$\bar{u}/\bar{v} = (R, (\bar{u}/\bar{v}), s_{\text{div}})$$

によって定義される。ここで

$$(\bar{u}/\bar{v})(w) = \sup \{ \min \{ (\bar{u})(u), (\bar{v})(v) \} \mid w = u/v (v \neq 0) \} \} \quad (4.13)$$

であり、また命題  $s_{\text{div}}$  は

$$s_{\text{div}}(w) = \{ w \in \bar{u}/\bar{v} \}$$

である。ファジィ除算に関しては次の結果が得られる。

**定理 4.2**  $\bar{u}$  をファジィ数、 $\bar{v}$  を正または負のファジィ数とする。このとき  $\bar{u}/\bar{v}$  もファジィ数であり

$$L_a(\bar{u}/\bar{v}) = (L_a\bar{u})/(L_a\bar{v}), \quad L_a(\bar{u}/\bar{v}) = (L_a\bar{u})/(L_a\bar{v}) \quad (4.14)$$

2)  $\inf \{ \text{supp } \bar{v} \} > 0$  のとき  $\bar{v}$  を正のファジィ数、逆に  $\sup \{ \text{supp } \bar{v} \} < 0$  のとき負のファジィ数という。

を満たす.

[証明]  $1/v$  は  $v \neq 0$  で  $v$  の連続関数であるから, 定理 3.1 より明らかに

$$\frac{1}{\bar{v}} \in \mathcal{G}(R) \quad (4.15)$$

であることがわかる. また  $\{\bar{v}_\alpha \mid \alpha \in (0, 1)\}$  を各  $\bar{v}_\alpha$  がコンパクトで凸な  $\bar{v}$  の集合表現であるとする,  $1/\bar{v}_\alpha$  もコンパクト凸集合であることを示せる. つまり

$$\frac{1}{\bar{v}_\alpha} = \left\{ \frac{1}{v} \mid v \in \bar{v}_\alpha \right\} \quad (4.16)$$

は有界閉区間である. 以上の結果より  $1/\bar{v}$  はファジィ数であること, すなわち  $1/\bar{v} \in \mathcal{G}_c(R)$  であり

$$L_\alpha\left(\frac{1}{\bar{v}}\right) = \frac{1}{L_\alpha \bar{v}}, \quad L_\alpha\left(\frac{1}{\bar{v}}\right) = \frac{1}{L_\alpha \bar{v}}. \quad (4.17)$$

となる. さらに定理 4.1 より

$$\bar{u} \cdot \frac{1}{\bar{v}} = \frac{\bar{u}}{\bar{v}} \in \mathcal{G}_c(R)$$

であり (4.14) をみたすことがわかる. (証明終)

## 5 プログラムパッケージの開発

ファジィベクトルの二項演算用のプログラムパッケージを作成するにあたって, 簡単のため 2 次元のものまで, すなわち, ファジィ数および 2 次元ファジィベクトルに限定した. また定理 4.1 で得られた結果を活用するため, ファジィベクトル (ファジィ数) を各要素がレベル集合で構成される集合表現を用いて近似的に表現することとした. すなわち, 0 から 1 までのレベルを等間隔で数十に分割し, その各レベルでのレベル集合を集め

たものを近似的なファジィベクトル（ファジィ数）の集合表現と考えることにした。したがって、1次元ファジィベクトル、すなわちファジィ数では各レベルでのレベル集合である閉区間の集まりが近似的集合表現となる。また2次元ファジィベクトルでは、各レベル集合は2次元閉凸集合となるが、このような一般的な形状のままでは、簡単な2項演算においても、その（近似）数値解を求めることは莫大な計算量を必要とし非常に困難である場合が多い。そこで、ここでは各レベル集合の形状が矩形であるか、または円形である場合のみに限定した。このような形状である場合、同じ形状どうしの加算については図 5.1 と図 5.2 に示したように容易に得られることが示せるから [4]、それらに対応するファジィベクトルの和の近似的集合表現も容易に得られることになる。減算についても、レベル集合の形状が矩形である場合にはやはり容易に演算結果も矩形であることがわかるが、他の場合については現状では不明である。また内積についてはレベ

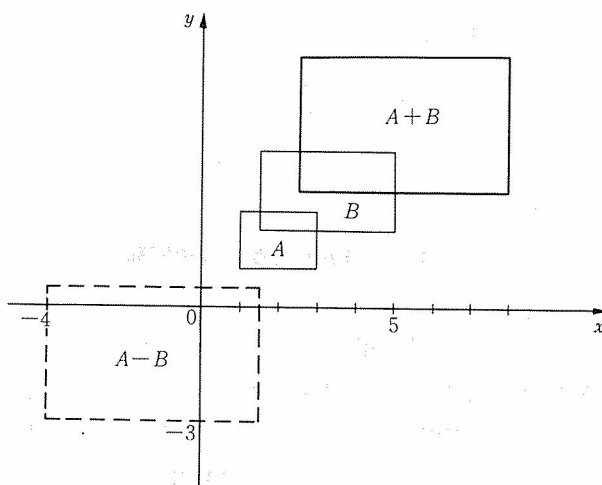


Fig. 5.1 矩形 A と B の加算と減算

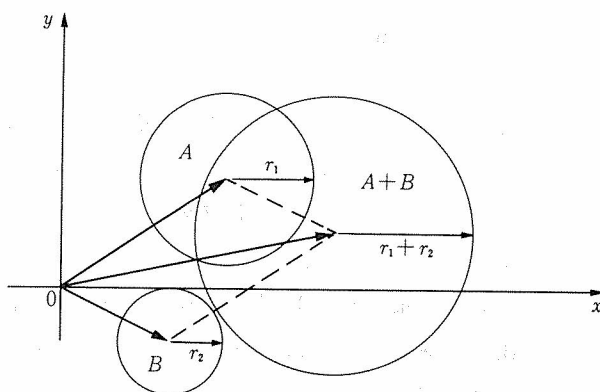


Fig. 5.2 円 A と B の加算

ル集合の形状には関係なく、各レベル集合の境界値のみに依存していることが定理 4.1 の (4.5) によって示されているので、各境界を数十から数百に分割し、その各分割点どうしの内積をすべて計算することによって近似的に求めることにした。

以上の設定のもとで、ファジィ数に関しては四則演算すなわち加算・減算・乗算・除算を、ファジィベクトルに関しては、そのレベル集合が矩形形状であるときには加算・減算・スカラー倍および内積を、円形の場合には加算と内積を数値計算するためのコンピュータプログラムパッケージを開発した。実際の開発にあたっては、Borland 社製の BC++ for Windows を用いてパーソナルコンピュータ上でおこなった。特に C++ のオーバーローディング機能を活用することにより、通常の実数に対する演算子記号 '+' や '-' などがファジィ数、ファジィベクトルに対しても同様に使用できるように試みた。また完全とは言えないが、オブジェクト指向プログラミングの考え方を取り入れ、見通しの良いプログラムとなるように努力した。付録にプログラムリストを示す。

## 6 結 言

本論文では、多値論理の立場よりファジィベクトルを定義し、拡張原理を用いてファジィベクトル間の二項演算を考察した。また二項演算の結果得られるファジィ集合が、ファジィ数あるいはファジィベクトルとなるのはどのような場合であるかについて考察した。さらにこれらの結果を利用し、ファジィ数の加算・減算・乗算・除算の四則計算およびファジィベクトルの加算・減算・内積計算を効率良く実行するプログラムパッケージをプログラム言語 C++ を用いて開発をおこなった。

## 参 考 文 献

- [1] L. A. Zadeh. The concept of linguistic variables and its application in approximate reasoning, part 1, 2 and 3. *Information Sciences*, 8: 199-249, 301-357, and 9: 43-80, respectively, 1975, 76.
- [2] R. Kruse and K. D. Meyer. *Statistics with Vague Data*. D. Reidel Publishing Company, Dordrecht, 1987.
- [3] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, New York, 1980.
- [4] A. Kaufmann and M. M. Gupta. *Introduction to Fuzzy Arithmetic: Theory and Applications*. Van Nostrand Reinhold, New York, 1985.
- [5] A. Kaufmann and M. M. Gupta. *Fuzzy Mathematical Models in Engineering and Management Science*. Elsevier Science, Netherlands, 1988.
- [6] H. Kwakernaak. Fuzzy random variables-I: definitions and theorems. *Information Sciences*, 15: 1-29, 1978.
- [7] R. E. Bellman and M. Giertz. On the analytic formalism of the theory of fuzzy sets. *Information Sciences*, 5: 149-156, 1973.
- [8] T. Fukuda. Basic properties of fuzzy sets. *The Kyoto Gakuen University Review*, 18: 125-164, 1990.

(平成6年4月14日受理)

## 付録: プログラムリスト

### テスト用プログラム

```

1: #include <math.h>
2: #include <stdio.h>
3: #include <conio.h>
4:
5: #ifndef FUZZY_SET
6: #include "f_num.h"
7: #endif
8: // extern unsigned _stklen=30000U;
9:
10: // ++++++
11: void main(){
12: // ++++++
13: FILE *file;
14: Point P1(1.0,1.0), P2(2.0,2.0);
15:
16: file= fopen("sample.dta","w+");
17:
18: // ++++++
19: // Operations of Fuzzy Numbers
20: // ++++++
21: fprintf(file, "Operations of Fuzzy Numbers \n");
22: Intervals I1(0.5,1.5), I2(1.5,2.5);
23: FuzzyNumbers F1(1.0,I1), F2(2.0,I2), F;
24:
25: // Addition of Two Fuzzy Numbers
26: fprintf(file, "\n Addition of Fuzzy Numbers, F1 and F2 \n\n");
27: fprintf(file, " F1= \n");
28: F1.WriteToFile(file);
29: fprintf(file, "\n F2= \n");
30: F2.WriteToFile(file);
31: F= F1+F2;
32: fprintf(file, "\n F= F1+F2 \n");
33: F.WriteToFile(file);
34:
35: // Subtraction of Two Fuzzy Numbers
36: fprintf(file, "\n\n Subtraction of Fuzzy Numbers, F1 and F2 \n");
37: F= F1-F2;
38: fprintf(file, " F= F1-F2 \n");
39: F.WriteToFile(file);
40:
41: // Multiplication of Two Fuzzy Numbers
42: fprintf(file, "\n\n Multiplication of Fuzzy Numbers, F1 and F2 \n");
43: F= F1*F2;
44: fprintf(file, " F= F1*F2 \n");
45: F.WriteToFile(file);
46:
47: // ++++++
48: // Operations of Fuzzy Vectors

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

49: // ++++++
50:
51: fprintf(file, "\n\n Operations of Fuzzy Vectors \n");
52:
53: // ++++++
54: // Pyramid Shape
55: // ++++++
56: fprintf(file, " Pyramid Shape \n");
57: Rectangles R1(P1,1.0,1.0), R2(P2,1.0,1.0), R;
58: Pyramids Py1(P1,R1), Py2(P2,R2), Py;
59:
60: // Addition of Two Fuzzy Vectors
61: fprintf(file, "\n Addition of Fuzzy Vectors, Py1 and Py2 \n");
62: fprintf(file, "\n Py1= ");
63: Py1.WriteToFile(file);
64: fprintf(file, "\n\n Py2= ");
65: Py2.WriteToFile(file);
66: Py= Py1+Py2;
67: fprintf(file, "\n\n Py= Py1+Py2 ");
68: Py.WriteToFile(file);
69:
70: // Subtraction of Two Fuzzy Vectors
71: fprintf(file, "\n\n Subtraction of Fuzzy Vectors, Py1 and Py2 \n");
72: Py= Py1-Py2;
73: fprintf(file, " Py= Py1-Py2 \n");
74: Py.WriteToFile(file);
75:
76: // InnerProduct of Two Fuzzy Vectors
77: fprintf(file, "\n\n InnerProduct of Fuzzy Vectors, Py1 and Py2 \n");
78: F= Py1.InnerProduct(Py2);
79: fprintf(file, " F(Fuzzy Number)= (Py1'Py2) \n");
80: F.WriteToFile(file);
81:
82:
83: // ++++++
84: // Bell Shape
85: // ++++++
86:
87: fprintf(file, "\n\nfs Bell Shape \n");
88: Circles C1(P1,2.0), C2(P2,1.0), C;
89: Bells B1(P1,C1), B2(P2,C2), B;
90:
91: // Addition of Two Fuzzy Vectors
92: fprintf(file, "\n Addition of Fuzzy Vectors, B1 and B2 \n");
93: fprintf(file, "\n B1= \n");
94: B1.WriteToFile(file);
95: fprintf(file, "\n B2= \n");
96: B2.WriteToFile(file);
97: B= B1+B2;
98: fprintf(file, "\n B= B1+B2 \n");
99: B.WriteToFile(file);

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

100:
101: // InnerProduct of Two Fuzzy Vectors
102: fprintf(file, "\n InnerProduct of Fuzzy Vectors, B1 and B2 \n");
103: F= B1.InnerProduct(B2);
104: fprintf(file, " F(Fuzzy Number)= (B1'B2) \n");
105: F.WriteToFile(file);
106:
107: fclose(file);
108: }
109:

```

ヘッダーファイル

F\_NUM.H

```

1: /*
2:   ARITHMETIC OPERATIONS FOR FUZZY NUMBERS
3: */
4: #include <stdio.h>
5: #include <fstream.h>
6: #define max(a,b)      ((a) > (b)) ? (a) : (b)
7: #define min(a,b)      ((a) < (b)) ? (a) : (b)
8:
9: #define FUZZY_SETS      1
10: #define NUMBER_OF_LEVELS 20
11: #define WIDTH          5
12: #define EPS             1.0e-4
13:
14: /*          ***** Relations among Classes *****
15:
16:   Point
17:       -- Intervals
18:       |
19:   CrispSets --|-- Rectangles
20:       |
21:       -- Circles
22:
23:
24:       -- FuzzyNumbers
25:       |
26:   FuzzySets --|-- Pyramids
27:       |
28:       -- FuzzyVectors --|
29:       |
30:       -- Bells
31:
32:
33: */
34: // *****
35: class Point{
36: // *****
37: protected:

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

38:  double x, y;
39:
40: public:
41:  // Constructor #1
42:  Point(){x=0.0; y=0.0;}
43:  // Constructor #2
44:  Point(double p, double q){x=p; y=q;}
45:
46:  // Destructor
47:  ~Point(){}
48:
49:  double  GetPointX(){return x;}
50:  double  GetPointY(){return y;}
51:  void    PutPointX(double a){x= a;}
52:  void    PutPointY(double b){y= b;}
53:  void    PutPoint(double a, double b){x= a; y= b;}
54:  double  InnerProduct(Point& v){return x*v.x+y*v.y;}
55:  friend Point operator +(Point&, Point&);
56:  friend Point operator -(Point&, Point&);
57:  friend Point operator *(double, Point&);
58:  friend Point operator -(Point&);
59:
60: };
61:
62:
63: // *****
64: class CrispSets{
65: // *****
66:
67: public:
68:
69:  // Constructor #1
70:  CrispSets(){}
71:
72:  // Destructor
73:  ~CrispSets(){}
74: };
75:
76:
77: // *****
78: class Intervals:public CrispSets{
79: // *****
80: protected:
81:
82:  double Minimum, Maximum;
83:
84: public:
85:
86:  // Constructor #1
87:  Intervals(double, double);
88:  // Constructor #2

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```
89:     Intervals(Intervals&);
90:
91: // Destructor
92: ~Intervals(){}
93:
94:     double GetMinimum(){return Minimum;}
95:     double GetMaximum(){return Maximum;}
96:     void PutMinimum(double m){Minimum= m;}
97:     void PutMaximum(double m){Maximum= m;}
98:     void PutInterval(double, double);
99:     void Show();
100:    void WriteToFile(FILE*);
101:    void ReadFromFile(FILE*);
102:
103:
104: // Operator Overloading
105:
106:     friend Intervals operator +(Intervals&, Intervals&);
107:     friend Intervals operator -(Intervals&, Intervals&);
108:     friend Intervals operator -(Intervals&);
109:     friend Intervals operator *(Intervals&, Intervals&);
110:     friend Intervals operator *(double, Intervals&);
111:     friend Intervals operator /(Intervals&, Intervals&);
112:     friend Intervals operator /(double, Intervals&);
113: };
114:
115:
116: // *****
117: class Rectangles:public CrispSets{
118: // *****
119: protected:
120:     Intervals X, Y;
121:
122: public:
123:
124: // Constructor #1
125:     Rectangles();
126: // Constructor #2
127:     Rectangles(Point, double, double);
128: // Constructor #3
129:     Rectangles(Intervals&, Intervals&);
130: // Constructor #4
131:     Rectangles(Rectangles& OtherRectangle);
132:
133:
134:
135: // Destructor
136:     ~Rectangles(){}
137:
138:     Intervals GetIntervalX(){return X;}
139:     Intervals GetIntervalY(){return Y;}
140:     void PutIntervalX(Intervals& IntervalX){X= IntervalX;}
```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

141: void          PutIntervalX(double Min, double Max)
                                {X.PutInterval(Min,Max);}
142: void          PutIntervalY(Interval& Interval){Y= IntervalY;}
143: void          PutIntervalY(double Min, double Max)
                                {Y.PutInterval(Min,Max);}
144: void          Boundary(Point*, const int);
145: void          Show();
146: void          WriteToFile(FILE*);
147: void          ReadFromFile(FILE*);
148: Interval&     InnerProduct(Rectangles&);
149:
150: // Operator Overloading
151:
152:     friend Rectangles operator +(Rectangles&, Rectangles&);
153:     friend Rectangles operator -(Rectangles&, Rectangles&);
154:     friend Rectangles operator -(Rectangles&);
155:     friend Rectangles operator *(double, Rectangles&);
156:
157: };
158:
159:
160: // *****
161: class Circles:public CrispSets{
162: // *****
163: protected:
164:
165:     Point Center;
166:     double Radius;
167:
168: public:
169:
170: // Constructor #1
171:     Circles();
172: // Constructor #2
173:     Circles(Point, double);
174: // Constructor #3
175:     Circles(Circles&);
176:
177: // Destructor
178:     ~Circles(){}
179:
180:     Point      GetCenter(){return Center;}
181:     double     GetRadius(){return Radius;}
182:     void       PutCenter(Point c){Center= c;}
183:     void       PutRadius(double r){Radius= r;}
184:     void       Boundary(Point*, const int);
185:     Interval&  InnerProduct(Circles&);
186:     void       Show();
187:     void       WriteToFile(FILE*);
188:     void       ReadFromFile(FILE*);
189:

```

## フuzzyベクトルの二項演算用プログラムパッケージの開発

```
190: // Operator Overloading
191: friend Circles operator +(Circles&, Circles&);
192:
193: };
194:
195:
196: // *****
197: class FuzzySets{
198: // *****
199:
200: public:
201:
202: // Constructor #1
203:     FuzzySets(){}
204:
205: // Destructor
206:     ~FuzzySets(){}
207: };
208:
209: // *****
210: class FuzzyNumbers:public FuzzySets{
211: // *****
212: protected:
213:
214:     Intervals LevelSets[NUMBER_OF_LEVELS+1];
215:
216: public:
217:
218: // Constructor #1
219:     FuzzyNumbers();
220:
221: // Constructor #2
222:     FuzzyNumbers(double);
223:
224: // Constructor #3
225:     FuzzyNumbers(double, Intervals);
226:
227:
228:
229: // Constructor #4
230:     FuzzyNumbers(FuzzyNumbers&);
231:
232: // Destructor
233:     ~FuzzyNumbers(){}
234:
235:     Intervals GetLevelSets(int);
236:     void PutLevelSets(int, Intervals);
237:     void PutCrispValue(double);
238:     void Show();
239:     void WriteToFile(FILE*);
240:     void ReadFromFile(FILE*);
241:
242: // Operator Overloading
243:
```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```
244:     friend FuzzyNumbers operator +(FuzzyNumbers&, FuzzyNumbers&);
245:     friend FuzzyNumbers operator -(FuzzyNumbers&, FuzzyNumbers&);
246:     friend FuzzyNumbers operator -(FuzzyNumbers&);
247:     friend FuzzyNumbers operator *(FuzzyNumbers&, FuzzyNumbers&);
248:     friend FuzzyNumbers operator *(double, FuzzyNumbers&);
249:     friend FuzzyNumbers operator /(FuzzyNumbers&, FuzzyNumbers&);
250:     friend FuzzyNumbers operator /(double, FuzzyNumbers&);
251: };
252:
253:
254: // *****
255: class FuzzyVectors:public FuzzySets{
256: // *****
257:
258: public:
259:
260: // Constructor
261:     FuzzyVectors(){}
262:
263: // Destructor
264:     ~FuzzyVectors(){}
265: };
266:
267:
268: // *****
269: class Pyramids:FuzzyVectors{
270: // *****
271: protected:
272:     Rectangles LevelSets[NUMBER_OF_LEVELS+1];
273:
274: public:
275:
276: // Constructor #1
277:     Pyramids();
278: // Constructor #2
279:     Pyramids(Point, Rectangles&);
280: // Constructor #3
281:     Pyramids(Pyramids&);
282:
283: // Destructor
284:     ~Pyramids(){}
285:
286:     FuzzyNumbers InnerProduct(Pyramids&);
287:     void          Show();
288:     void          PutLevelSet(int, Rectangles);
289:     Rectangles    GetLevelSet(int);
290:     void          WriteToFile(FILE*);
291:     void          ReadFromFile(FILE*);
292:
293: // Operator Overloading
294:
```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```
295:     friend Pyramids operator +(Pyramids&, Pyramids&);
296:     friend Pyramids operator -(Pyramids&, Pyramids&);
297:     friend Pyramids operator -(Pyramids&);
298: };
299:
300:
301: // *****
302: class Bells:FuzzyVectors{
303: // *****
304: protected:
305:
306:     Circles LevelSets[NUMBER_OF_LEVELS+1];
307:
308: public:
309:
310: // Constructor #1
311:     Bells();
312: // Constructor #2
313:     Bells(Point, Circles);
314: // Constructor #3
315:     Bells(Bells&);
316:
317: // Destructor
318:     ~Bells(){}
319:
320:     FuzzyNumbers InnerProduct(Bells&);
321:     void Show();
322:     void WriteToFile(FILE*);
323:     void ReadFromFile(FILE*);
324:
325: // Operator Overloading
326:
327:     friend Bells operator +(Bells&, Bells&);
328:     friend Bells operator -(Bells&, Bells&);
329:     friend Bells operator -(Bells&);
330:     friend Bells operator *(double, Bells&);
331: };
332:
333:
```

### プログラム本体

F\_NUM.CPP

```
1: // =====
2: // Implementation of F_NUM.CPP
3: // =====
4:
5: #include <stdio.h>
6: #include <stdlib.h>
7: #include <math.h>
8: #include <iostream.h>
```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

9: #include <iomanip.h>
10: #include <conio.h>
11: #ifndef FUZZY_NUMBER
12:     #include "f_num.h"
13: #endif
14:
15:
16: // #####
17: // Class Point
18: // #####
19:
20: // Operator Overloading
21: // *****
22: Point operator +(Point& U, Point& V){
23: // *****
24:     Point Result(U.GetPointX() + V.GetPointX(),
                   U.GetPointY() + V.GetPointY());
25:     return Result;
26: }
27:
28: // *****
29: Point operator -(Point& U, Point& V){
30: // *****
31:     Point Result(U.GetPointX() - V.GetPointX(),
                   U.GetPointY() - V.GetPointY());
32:     return Result;
33: }
34:
35: // *****
36: Point operator *(double a, Point& U){
37: // *****
38:     Point Result(a*U.GetPointX(), a*U.GetPointY());
39:     return Result;
40: }
41:
42: // *****
43: Point operator -(Point& U){
44: // *****
45:     Point Result(-U.GetPointX(), -U.GetPointY());
46:     return Result;
47: }
48:
49:
50: // #####
51: // Class Intervals
52: // #####
53:
54:
55: // Constructor #1
56: // *****
57: Intervals::Intervals(double m=0.0, double n=0.0){

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

58: // *****
59:   if(m < n)
60:     {Minimum= m; Maximum= n;}
61:   else
62:     {Minimum= n; Maximum= m;}
63: }
64:
65: // Constructor #2
66: // *****
67: Intervals::Intervals(Intervals& OtherInterval){
68: // *****
69:   Minimum= OtherInterval.Minimum;
70:   Maximum= OtherInterval.Maximum;
71: }
72:
73: // *****
74: void Intervals::PutInterval(double m, double n){
75: // *****
76:   if(m < n)
77:     {Minimum= m; Maximum= n;}
78:   else
79:     {Minimum= n; Maximum= m;}
80: }
81:
82: // *****
83: void Intervals::Show(){
84: // *****
85:   printf("%lf, %lf", Minimum, Maximum);
86: }
87:
88: // *****
89: void Intervals::WriteToFile(FILE* stream){
90: // *****
91:   fprintf(stream,"%lf, %lf", Minimum, Maximum);
92: }
93:
94: // *****
95: void Intervals::ReadFromFile(FILE* stream){
96: // *****
97:   fscanf(stream,"%lf, %lf", &Minimum, &Maximum);
98: }
99:
100:
101: // Operator Overloading
102:
103: // *****
104: Intervals operator +(Intervals& U, Intervals& V){
105: // *****
106: Intervals Result;
107:   Result.Minimum= U.Minimum + V.Minimum;
108:   Result.Maximum= U.Maximum + V.Maximum;

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

109: return Result;
110: }
111:
112: // *****
113: Intervals operator -(Intervals& U, Intervals& V){
114: // *****
115: Intervals Result;
116: Result.Minimum= U.Minimum - V.Maximum;
117: Result.Maximum= U.Maximum - V.Minimum;
118: return Result;
119: }
120:
121: // *****
122: Intervals operator -(Intervals& U){
123: // *****
124: Intervals Result;
125: Result.Minimum= -U.Maximum;
126: Result.Maximum= -U.Minimum;
127: return Result;
128: }
129:
130: // *****
131: Intervals operator *(Intervals& U, Intervals& V){
132: // *****
133: double t1, t2, t3, t4, t5, t6;
134: Intervals Result;
135: t1= U.Minimum * V.Minimum;
136: t2= U.Minimum * V.Maximum;
137: t3= U.Maximum * V.Minimum;
138: t4= U.Maximum * V.Maximum;
139: t5= min(t1,t2); t6= min(t3,t4);
140: Result.Minimum= min(t5,t6);
141: t5= max(t1,t2); t6= max(t3,t4);
142: Result.Maximum= max(t5,t6);
143: return Result;
144: }
145:
146: // *****
147: Intervals operator *(double u, Intervals& V){
148: // *****
149: double t1, t2;
150: Intervals Result;
151: t1= u*V.Minimum;
152: t2= u*V.Maximum;
153: Result.Minimum= min(t1,t2);
154: Result.Maximum= max(t1,t2);
155: return Result;
156: }
157:
158: // *****
159: Intervals operator /(Intervals& U, Intervals& V){

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```
164: // *****
165: Intervals T;
166: T.PutInterval(1.0/V.GetMaximum(),1.0/V.GetMinimum());
167: return U*T;
168: }
169:
170: // *****
171: Intervals operator /(double u, Intervals& V){
172: // *****
173: Intervals T;
174: T.PutInterval(1.0/V.GetMaximum(),1.0/V.GetMinimum());
175: return u*T;
176: }
177:
178: // *****
179: // Class Rectangles
180: // *****
181: // *****
182:
183: // Constructor #1
184: // *****
185: Rectangles::Rectangles(){
186: // *****
187: Point Center;
188: X.PutMinimum(0.0);
189: X.PutMaximum(0.0);
190: Y.PutMinimum(0.0);
191: Y.PutMaximum(0.0);
192: }
193: // Constructor #2
194: // *****
195: Rectangles::Rectangles(Point Center, double Hwidth, double Vwidth){
196: // *****
197: double t1= Hwidth/2.0;
198: double t2= Vwidth/2.0;
199: X.PutMinimum(Center.GetPointX() - t1);
200: X.PutMaximum(Center.GetPointX() + t1);
201: Y.PutMinimum(Center.GetPointY() - t2);
202: Y.PutMaximum(Center.GetPointY() + t2);
203: }
204:
205: // Constructor #3
206: // *****
207: Rectangles::Rectangles(Intervals& IntervalX, Intervals& IntervalY){
208: // *****
209: X= IntervalX;
210: Y= IntervalY;
211: }
212:
213: // Constructor #4
214: // *****
215: Rectangles::Rectangles(Rectangles& OtherRectangle){
216: // *****
```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

217:   X= OtherRectangle.X;
218:   Y= OtherRectangle.Y;
219: }
220:
221: // *****
222: void Rectangles::Show(){
223: // *****
224:   printf("%lf,%lf)  ",X.GetMinimum(),Y.GetMinimum());
225:   printf("%lf,%lf)  ",X.GetMaximum(),Y.GetMinimum());
226:   printf("%lf,%lf)  ",X.GetMaximum(),Y.GetMaximum());
227:   printf("%lf,%lf)  ",X.GetMinimum(),Y.GetMinimum());
228:   printf("\n");
229: }
230:
231: // *****
232: void Rectangles::Boundary(Point* u, const int N=25){
233: // *****
234:   double Min, Max, x, y;
235:   for(int i=1;i<=4;i++){
236:     switch(i) {
237:       case 1: Min= X.GetMinimum(); Max= X.GetMaximum();
238:               y= Y.GetMinimum();
239:               for(int j=0;j<=N-1;j++){
240:                 x= Min + float(j)/float(N-1.0)*(Max-Min);
241:                 u[(i-1)*N+j].PutPoint(x,y);
242:               }
243:               break;
244:       case 2: Min= Y.GetMinimum(); Max= Y.GetMaximum();
245:               x= X.GetMaximum();
246:               for(j=0;j<=N-1;j++){
247:                 x= Min + float(j)/float(N-1.0)*(Max-Min);
248:                 u[(i-1)*N+j].PutPoint(x,y);
249:               }
250:               break;
251:       case 3: Min= X.GetMinimum(); Max= X.GetMaximum();
252:               y= Y.GetMaximum();
253:               for(j=0;j<=N-1;j++){
254:                 x= Max + float(j)/float(N-1.0)*(Min-Max);
255:                 u[(i-1)*N+j].PutPoint(x,y);
256:               }
257:               break;
258:       case 4: Min= Y.GetMinimum(); Max= Y.GetMaximum();
259:               x= X.GetMinimum();
260:               for(j=0;j<=N-1;j++){
261:                 y= Max + float(j)/float(N-1.0)*(Min-Max);
262:                 u[(i-1)*N+j].PutPoint(x,y);
263:               }
264:     }
265:   }
266: }
267:

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

268: // *****
269: Intervals Rectangles::InnerProduct(Rectangles& V){
270: // *****
271: double Min, Max, t;
272: Point u[100], v[100];
273: Intervals W;
274:   this->Boundary(u,25);
275:   V.Boundary(v,25);
276:   Min= Max= u[0].InnerProduct(v[0]);
277:   for(int i=1;i<=99;i++){
278:     t= u[i].InnerProduct(v[i]);
279:     if(Min > t) Min=t;
280:     if(Max < t) Max=t;
281:   }
282:   W.PutInterval(Min,Max);
283:   return W;
284: }
285:
286: // *****
287: void Rectangles::WriteToFile(FILE* stream){
288: // *****
289:
290:   fprintf(stream, "(%lf, %lf) \t", X.GetMinimum(), Y.GetMinimum());
291:   fprintf(stream, "(%lf, %lf) \t", X.GetMaximum(), Y.GetMinimum());
292:   fprintf(stream, "(%lf, %lf) \t", X.GetMaximum(), Y.GetMaximum());
293:   fprintf(stream, "(%lf, %lf)", X.GetMinimum(), Y.GetMaximum());
294: }
295:
296:
297: // *****
298: void Rectangles::ReadFromFile(FILE* stream){
299: // *****
300:   double tx1,tx2,tx3,tx4,ty1,ty2,ty3,ty4;
301:   fscanf(stream, "(%lf, %lf) \t (%lf, %lf) \t (%lf, %lf) \t\n\n",
302:           (%lf, %lf)", &tx1, &ty1, &tx2, &ty2, &tx3, &ty3, &tx4, &ty4);
303:   X.PutInterval(tx1,tx2);
304:   Y.PutInterval(ty1,ty4);
305: }
306:
307: // Operator Overloading
308: // *****
309: Rectangles operator +(Rectangles& U, Rectangles& V){
310: // *****
311: Rectangles Result(U.X + V.X, U.Y + V.Y);
312:   return Result;
313: }
314:
315: // *****
316: Rectangles operator -(Rectangles& U
317: , Rectangles& V){

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

318: // *****
319: Rectangles Result(U.X - V.X, U.Y - V.Y);
320: return Result;
321: }
322:
323: // *****
324: Rectangles operator -(Rectangles& U){
325: // *****
326: return (-1)*U;
327: }
328:
329: // *****
330: Rectangles operator *(double u, Rectangles& V){
331: // *****
332: Rectangles Result(u*V.X, u*V.Y);
333: return Result;
334: }
335:
336:
337: // #####
338: // Class Circles
339: // #####
340:
341:
342: // Constructor #1
343: // *****
344: Circles::Circles(){
345: // *****
346: Center.PutPoint(0.0,0.0);;
347: Radius= 0.0;
348: }
349:
350: // Constructor #2
351: // *****
352: Circles::Circles(Point c, double r){
353: // *****
354: Center= c;
355: Radius= r;
356: }
357:
358: // Constructor #3
359: // *****
360: Circles::Circles(Circles& OtherCircle){
361: // *****
362: Center= OtherCircle.Center;
363: Radius= OtherCircle.Radius;
364: }
365:
366: // *****
367: void Circles::Boundary(Point* u, const int N=100){
368: // *****

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

369: const Pi= 3.14159;
370:   for(int i=1;i<=N;i++){u[i-1].PutPoint(Center.GetPointX()
371:       +Radius*cos(2*Pi/float(N)*(i-1)),Center.GetPointY()
372:       +Radius*sin(2*Pi/float(N)*(i-1)));
373:   }
374: }
375:
376: // *****
377: Intervals Circles::InnerProduct(Circles& V){
378: // *****
379: double Min, Max, t;
380: Point u[100], v[100];
381: Intervals W;
382:   this->Boundary(u,100);
383:   V.Boundary(v,100);
384:   Min= Max= u[0].InnerProduct(v[0]);
385:   for(int i=1;i<=99;i++){
386:     t= u[i].InnerProduct(v[i]);
387:     if(Min > t) Min=t;
388:     if(Max < t) Max=t;
389:   }
390:   W.PutInterval(Min,Max);
391:   return W;
392: }
393:
394: // *****
395: void Circles::Show(){
396: // *****
397:   printf("Center= (%lf, %lf) \t Radius= %lf",
398:       Center.GetPointX(), Center.GetPointY(), Radius);
399: }

400: // *****
401: void Circles::WriteToFile(FILE* stream){
402: // *****
403:   fprintf(stream,"Center= (%lf, %lf) \t Radius= %lf \n",
404:       Center.GetPointX(), Center.GetPointY(), Radius);
405: }

406: // *****
407: void Circles::ReadFromFile(FILE* stream){
408: // *****
409:   double CenterX, CenterY, r;
410:   fscanf(stream,"Center= (%lf, %lf) \t Radius= %lf \n", &CenterX,
411:       &CenterY, &r);
412:   Center.PutPoint(CenterX, CenterY);
413:   Radius= r;
414: }

415: // Operator Overloading
416: // *****

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

417: Circles operator +(Circles& U, Circles& V){
418: // *****
419: Circles Result(U.GetCenter() + V.GetCenter(),
420:               U.GetRadius() + V.GetRadius());
421: return Result;
422: }
423: }
424: // #####
425: // Class Fuzzy_Numbers
426: // #####
427:
428: // *****
429: FuzzyNumbers::FuzzyNumbers(){
430: // *****
431: for(int i=0;i<= NUMBER_OF_LEVELS;i++){
432:     LevelSets[i].PutMinimum(0.0);
433:     LevelSets[i].PutMaximum(0.0);
434: }
435:
436: }
437: // *****
438: FuzzyNumbers::FuzzyNumbers(double Number){
439: // *****
440: for(int i=0;i<= NUMBER_OF_LEVELS;i++){
441:     LevelSets[i].PutMinimum(Number);
442:     LevelSets[i].PutMaximum(Number);
443: }
444: }
445:
446: // *****
447: FuzzyNumbers::FuzzyNumbers(double apex, Intervals Base){
448: // *****
449: double alpha;
450: double minimum= Base.GetMinimum();
451: double maximum= Base.GetMaximum();
452: if((apex<minimum) || (apex>maximum)){
453:     printf("apex should locate in the interval");
454:     exit(EXIT_FAILURE);
455: }
456: for(int i=0;i<= NUMBER_OF_LEVELS;i++){
457:     alpha= double(i)/double(NUMBER_OF_LEVELS);
458:     minimum= Base.GetMinimum();
459:     maximum= Base.GetMaximum();
460:     LevelSets[i].PutMinimum(alpha*(apex-minimum)+minimum);
461:     LevelSets[i].PutMaximum(alpha*(apex-maximum)+maximum);
462: }
463: }
464: // *****
465: FuzzyNumbers::FuzzyNumbers(FuzzyNumbers& OtherFuzzyNumber){
466: // *****
467: for(int i=0;i<= NUMBER_OF_LEVELS;i++)
468:     LevelSets[i]= OtherFuzzyNumber.LevelSets[i];

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

469: }
470:
471: // *****
472: Intervals FuzzyNumbers::GetLevelSets(int i){
473: // *****
474:     return LevelSets[i];
475: }
476:
477: // *****
478: void FuzzyNumbers::PutLevelSets(int i, Intervals U){
479: // *****
480:     LevelSets[i]= U;
481: }
482:
483: // *****
484: void FuzzyNumbers::PutCrispValue(double Number){
485: // *****
486:     for(int i=0;i<= NUMBER_OF_LEVELS;i++){
487:         LevelSets[i].PutMinimum(Number);
488:         LevelSets[i].PutMaximum(Number);
489:     }
490: }
491:
492: // *****
493: void FuzzyNumbers::Show(){
494: // *****
495:     printf("\n");
496:     for(int i=0;i<= NUMBER_OF_LEVELS;i++){
497:         printf("level= %5.3f \t", (float)i/float(NUMBER_OF_LEVELS));
498:         LevelSets[i].Show();
499:         printf("\n");
500:     }
501: }
502:
503: // *****
504: void FuzzyNumbers::WriteToFile(FILE* stream){
505: // *****
506:     for(int i=0;i<=NUMBER_OF_LEVELS;i++){
507:         fprintf(stream, "%5.31f \t", (double)i/double(NUMBER_OF_LEVELS));
508:         LevelSets[i].WriteToFile(stream);
509:         fprintf(stream, "\n");
510:     }
511: }
512:
513: // *****
514: void FuzzyNumbers::ReadFromFile(FILE* stream){
515: // *****
516: double level;
517:     for(int i=0;i<=NUMBER_OF_LEVELS;i++){
518:         fscanf(stream, "%5.31f", &level);
519:         LevelSets[i].ReadFromFile(stream);

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

520:     fscanf(stream, "\n");
521: }
522: }
523:
524: // *****
525: FuzzyNumbers operator +(FuzzyNumbers& U, FuzzyNumbers& V){
526: // *****
527: FuzzyNumbers W;
528: for(int i=0; i<=NUMBER_OF_LEVELS; i++){
529:     W.PutLevelSets(i, U.GetLevelSets(i)+V.GetLevelSets(i));
530: }
531: return W;
532: }
533: // *****
534: FuzzyNumbers operator -(FuzzyNumbers& U, FuzzyNumbers& V){
535: // *****
536: FuzzyNumbers W;
537: for(int i=0; i<=NUMBER_OF_LEVELS; i++){
538:     W.PutLevelSets(i, U.GetLevelSets(i)-V.GetLevelSets(i));
539: }
540: return W;
541: }
542: // *****
543: FuzzyNumbers operator ~(FuzzyNumbers& U){
544: // *****
545: FuzzyNumbers W;
546: for(int i=0; i<=NUMBER_OF_LEVELS; i++){
547:     W.PutLevelSets(i, ~U.LevelSets[i]);
548: }
549: return W;
550: }
551: // *****
552: FuzzyNumbers operator *(FuzzyNumbers& U, FuzzyNumbers& V){
553: // *****
554: FuzzyNumbers W;
555: for(int i=0; i<=NUMBER_OF_LEVELS; i++){
556:     W.PutLevelSets(i, U.GetLevelSets(i)*V.GetLevelSets(i));
557: }
558: return W;
559: }
560: // *****
561: FuzzyNumbers operator *(double u, FuzzyNumbers& V){
562: // *****
563: FuzzyNumbers W;
564: for(int i=0; i<=NUMBER_OF_LEVELS; i++){
565:     W.LevelSets[i] = u*V.LevelSets[i];
566: }
567: return W;
568: }
569: // *****
570: FuzzyNumbers operator /(FuzzyNumbers& U, FuzzyNumbers& V){

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

571: // *****
572: FuzzyNumbers W;
573:   for(int i=0;i<=NUMBER_OF_LEVELS;i++)
574:     W.PutLevelSets(i, U.GetLevelSets(i)/V.GetLevelSets(i));
575:   return W;
576: }
577:
578: // *****
579: FuzzyNumbers operator /(double u, FuzzyNumbers& V){
580: // *****
581: FuzzyNumbers W;
582:   for(int i=0;i<=NUMBER_OF_LEVELS;i++)
583:     W.LevelSets[i]= u/V.LevelSets[i];
584:   return W;
585: }
586:
587: // #####
588: // Class Pyramid
589: // #####
590:
591:
592: // Constructor #1
593: // *****
594: Pyramids::Pyramids(){
595: // *****
596:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
597:     LevelSets[i].PutIntervalX(0.0, 0.0);
598:     LevelSets[i].PutIntervalY(0.0, 0.0);
599:   }
600: }
601:
602: // Constructor #2
603: // *****
604: Pyramids::Pyramids(Point apex, Rectangles& Base){
605: // *****
606:   double alpha, Min, Max;
607:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
608:
609:     alpha= float(i)/float(NUMBER_OF_LEVELS);
610:
611:     Min= (Base.GetIntervalX()).GetMinimum();
612:     Max= (Base.GetIntervalX()).GetMaximum();
613:     LevelSets[i].PutIntervalX(Min+alpha*(apex.GetPointX()- Min),
614:                               Max+alpha*(apex.GetPointX()- Max));
615:
616:
617:     Min= (Base.GetIntervalY()).GetMinimum();
618:     Max= (Base.GetIntervalY()).GetMaximum();
619:     LevelSets[i].PutIntervalY(Min+alpha*(apex.GetPointY()- Min),
620:                               Max+alpha*(apex.GetPointY()- Max));
621:   }

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

622: }
623:
624: // Constructor #3
625: // *****
626: Pyramids::Pyramids(Pyramids& U){
627: // *****
628:   for(int i=0;i<= NUMBER_OF_LEVELS;i++)
629:     LevelSets[i]= U.LevelSets[i];
630: }
631:
632: // *****
633: FuzzyNumbers Pyramids::InnerProduct(Pyramids& U){
634: // *****
635:   FuzzyNumbers W;
636:   for(int i=0;i<= NUMBER_OF_LEVELS;i++)
637:     W.PutLevelSets(i,LevelSets[i].InnerProduct(U.LevelSets[i]));
638:   return W;
639: }
640:
641:
642: // *****
643: void Pyramids::PutLevelSet(int i, Rectangles U){
644: // *****
645:   LevelSets[i]= U;
646: }
647:
648: // *****
649: Rectangles Pyramids::GetLevelSet(int i){
650: // *****
651:   return LevelSets[i];
652: }
653:
654: // *****
655: void Pyramids::Show(){
656: // *****
657:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
658:     printf("\n Levels= %10.7f \t", float(i)/float(NUMBER_OF_LEVELS));
659:     LevelSets[i].Show();
660:   }
661: }
662:
663: // *****
664: void Pyramids::WriteToFile(FILE* stream){
665: // *****
666:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
667:     fprintf(stream, "\n %5.3lf \t", double(i)/double(NUMBER_OF_LEVELS));
668:     LevelSets[i].WriteToFile(stream);
669:   }
670: }
671:
672:

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```

673: // *****
674: void Pyramids::ReadFromFile(FILE* stream){
675: // *****
676: double level;
677:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
678:     fprintf(stream, "%5.3lf", &level);
679:     LevelSets[i].ReadFromFile(stream);
680:   }
681: }
682:
683: // Operator Overloading
684: // *****
685: Pyramids operator +(Pyramids& U, Pyramids& V){
686: // *****
687: int i;
688: Pyramids W;
689:   for(i=0;i<= NUMBER_OF_LEVELS;i++){
690:     W.LevelSets[i] = (U.LevelSets[i]) + (V.LevelSets[i]);
691:   }
692:   return W;
693: }
694:
695:
696: // *****
697: Pyramids operator -(Pyramids& U, Pyramids& V){
698: // *****
699: Pyramids W;
700:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
701:     W.LevelSets[i] = U.LevelSets[i] - V.LevelSets[i];
702:   }
703:   return W;
704: }
705: // *****
706: Pyramids operator -(Pyramids& U){
707: // *****
708: Pyramids W;
709:   for(int i=1;i<= NUMBER_OF_LEVELS;i++){
710:     W.PutLevelSet(i, U.GetLevelSet(i));
711:   }
712:   return W;
713: }
714:
715: // #####
716: // Class Bells
717: // #####
718:
719:
720: // Constructor #1
721: // *****
722: Bells::Bells(){
723: // *****

```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```
724: Point apex(0.0,0.0);
725:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
726:     LevelSets[i].PutCenter(apex);
727:     LevelSets[i].PutRadius(0.0);
728:   }
729: }
730:
731: // Constructor #2
732: // *****
733: Bells::Bells(Point apex, Circles Base){
734: // *****
735: double alpha;
736:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
737:     alpha= float(i)/float(NUMBER_OF_LEVELS);
738:     LevelSets[i].PutCenter(apex);
739:     LevelSets[i].PutRadius((1.0-alpha)*(Base.GetRadius()));
740:   }
741: }
742:
743: // Constructor #3
744: // *****
745: Bells::Bells(Bells& U){
746: // *****
747:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
748:     LevelSets[i]= U.LevelSets[i];
749:   }
750: }
751:
752: // *****
753: FuzzyNumbers Bells::InnerProduct(Bells& U){
754: // *****
755: FuzzyNumbers W;
756:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
757:     W.PutLevelSets(i,LevelSets[i].InnerProduct(U.LevelSets[i]));
758:   }
759:   return W;
760: }
761:
762: // *****
763: void Bells::Show(){
764: // *****
765:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
766:     printf("Levels= %5.3f \t", (float)i/float(NUMBER_OF_LEVELS));
767:     LevelSets[i].Show();
768:   }
769: }
770:
771: // *****
772: void Bells::WriteToFile(FILE* stream){
773: // *****
774:   for(int i=0;i<= NUMBER_OF_LEVELS;i++){
```

## ファジィベクトルの二項演算用プログラムパッケージの開発

```
775:     fprintf(stream, "%.3lf \t", double(i)/double(NUMBER_OF_LEVELS));
776:     LevelSets[i].WriteToFile(stream);
777: }
778: }
779:
780: // *****
781: void Bells::ReadFromFile(FILE* stream){
782: // *****
783: double level;
784: for(int i=0;i<= NUMBER_OF_LEVELS;i++){
785:     fprintf(stream, "%.3lf", &level);
786:     LevelSets[i].ReadFromFile(stream);
787: }
788: }
789:
790:
791: // Operator Overloading
792: // *****
793: Bells operator +(Bells& U, Bells& V){
794: // *****
795: Bells W;
796: for(int i=0;i<= NUMBER_OF_LEVELS;i++)
797:     W.LevelSets[i]= U.LevelSets[i] + V.LevelSets[i];
798: return W;
799: }
```