

2015

2018 年改訂

初めてのファジィ 理論入門

学部生のためのファジィ制御の基礎

目次

はじめに
ファジィ集合とは
従来の集合の問題点
ファジィ集合の考え方

ファジィ推論とは
ルールベース推論

ファジィ推論
・ 1 入力の場合
・ 2 入力への拡張

ファジィ制御
まとめ

演習課題



1.はじめに

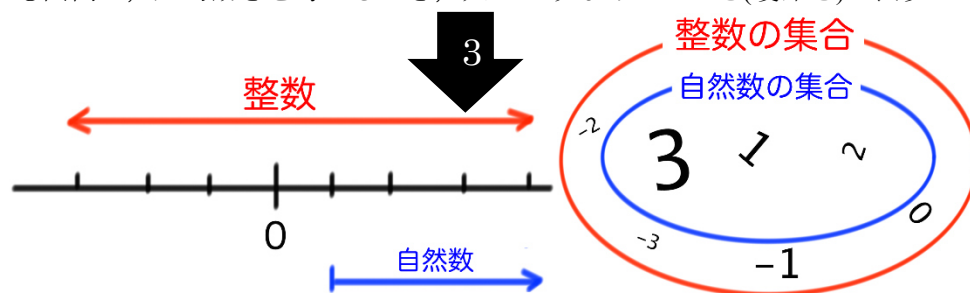
ファジィ理論はファジィ集合、ファジィ推論などからなり、私たちの身の回りのあいまいな情報を取り扱うための理論である。特徴として、設計者の持つ知識をコンピュータに取り込む場合に、人の感覚に似た自然な表現のままに、知識を記述できる手法として、広く産業界に定着している。ファジィ集合を基にファジィ推論法が考案され、現在までにファジィ制御に関する研究開発が多方面で行なわれている。心理学、経済学などの文系の分野から、プラント、自動車、家電製品の制御など、工学の分野まで幅広い応用が進められている。

本章では、ファジィ理論の概要について簡単に触れ、ファジィ理論の基礎であるファジィ集合と、ファジィ推論の基礎的な考え方について解説を行い、その応用例として、ファジィ制御について述べる。

2.ファジィ理論概要

現代制御理論は数学モデルに基づいて大いに発展してきたが、制御理論を現実のシステムに適用しようとすると、実際の制御対象となるシステムにあいまい性が存在するため、数式で厳密に表現することが難しいことが多い。また制御理論は線形系の理論を基本としているために、対象とするシステムに非線形性が多い場合は線形理論により良好な制御システムを構成することが難しい。1965年に **Lotfi A. Zadeh [2]**は、厳密に境界を定義する集合論では私たちの持っている感覚などを表現することが困難であることを指摘し、境界があいまいな集合の概念であるファジィ集合の理論を提唱した。ファジィ理論を学ぶ前に簡単に数と集合の世界を見てみる。数字を粒として考えてみよう。そうすると、数の取り扱いにおいて、はっきりとした数値で扱うことが少々難しくなってくる。例えば、米粒を扱うときは1粒2粒で一般的には扱わない。グラム g で扱ったり、合で扱ったりする。つまり、集まりや範囲(値域)で扱うことになる。実数値のデータを扱う場合も、同様に細かい数字を直接扱わないで、集まり=集合や、範囲(地域)や、集合の関係や写像を使って考えることになる。

では、図を見てみよう。数直線上において数値3は自然数にも含まれており、整数にも含まれていることが分かる。そこで、右図のベン図を見てみよう。数値3は整数の集合にも自然数の集合にも所属していることが分かる。所属している or していない。つまり、属する or 属さない のようにハッキリした分類になっている。ここにはあいまいさ(曖昧さ)は存在せず、極めてハッキリした分類がされている。さて、人が何かを制御する事を考えてみよう。はっきりとした分類で柔軟な制御が可能であろうか？きめ細かく分類しなければならない。それは数値が細かな(連続な)粒であるからである。そうになると計算はとても面倒で、知的動きを考えると、人のようなあいまいさ(曖昧さ)の表現が必要だろう。



3.ファジィ集合

3.1 従来の集合(クリスプ集合)

従来、集合の境界は厳密に定義に定義されなければならなかった。ある要素が集合に属するか、属さないか、そのどちらでもないか等は、本来議論を始めることのできないことであった。しかし、現実の世界には境界のあやふやな事柄が数多く存在する。特に私たちが日常生活で使っている言葉には、厳密であるものの方が少ない。以下には、従来の集合(クリスプ集合)とファジィ集合の考え方を述べる。

お湯の「熱い」という言葉を定義してくださいと言われたらどうするか。集合論によると、集合“熱い”は温度Tを要素として、

$$\text{“熱い”} : \{ T \mid T \geq 43^{\circ}\text{C} \} \quad (1)$$

と定義することを考える。これは、特性関数を用いると次のように表現できる。

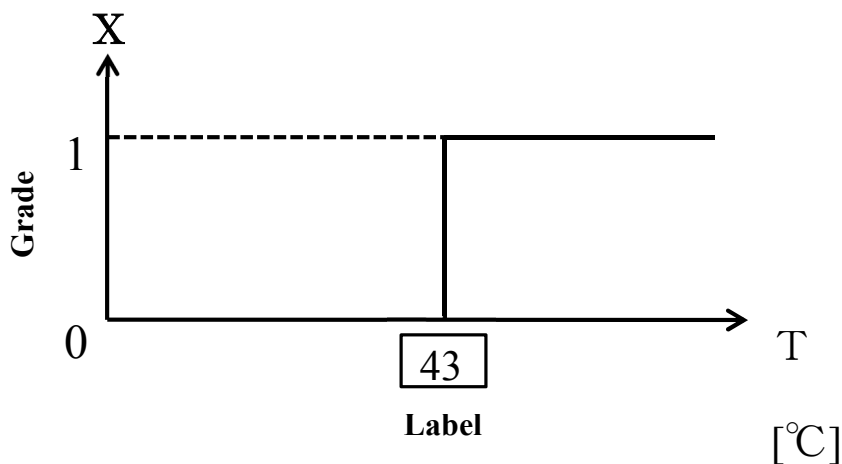


Fig.1 クリスプなメンバーシップ関数 “熱い”

Fig.1 は式(1)の特性関数を示す。温度 T が集合“熱い”に属すれば特性関数の値は 1、そうでなければ 0 である。この定義は「熱い」という言葉を誰にでも誤解を与えることなく伝えることができる。しかし、この定義は私たちの持つ「熱い」という言葉の概念を的確に表現できていない。この定義に対して、次の 2 通りの回答がよく聞かれる。

(I) : 「私は暑がりだから、もっと低い温度でも熱いと思う。」

(II) : 「42.99 度は熱くなくて、43 度は熱いというのはおかしい。」

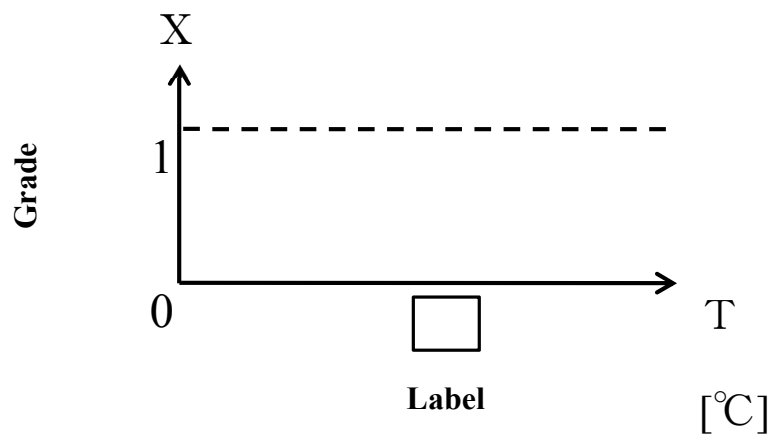
(I)に対しては、仮に

$$\text{“A君にとっての熱い”} : \{T \mid T \geq 41^{\circ}\text{C}\} \quad (2)$$

とすればよい．しかし，(Ⅱ)の回答は，従来の集合表現の本質的な問題を含んでおり，他の言葉でも同様に起きる．

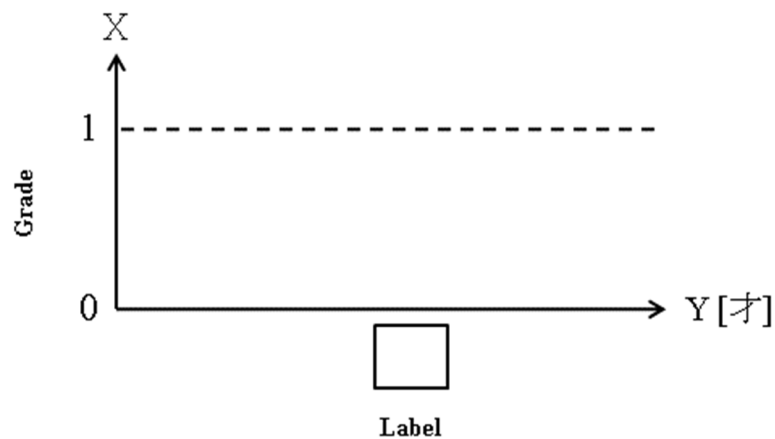
演習 1 上記では，お湯を例に従来の集合論について示した．これが以下の集合の場合，特性関数がどうなるか？ “寒い” は温度 T を要素として，Fig.1 のように下の図を完成させよ．

$$\text{“寒い”} : \{T \mid T \leq 10^{\circ}\text{C}\}$$



演習 2 あなたにとっての “若い” という集合はどのようなになるか示せ． “若い” は年齢 Y を要素として，以下の集合の定義と図を完成させよ．

$$\text{“若い”} : \{ \quad \quad \quad \}$$



個人の尺度によって個別に異なっていて，個別に決まっていることが分かる．
次に “高い” を身長 H の要素として，次の集合を定義した場合を考える．

$$\text{“高い”} : \{H \mid H > 180\text{cm}\} \quad (3)$$

身体測定の結果、「身長が 179 cm だったので、あなたの身長は低い」と言われると理不尽である。この他にも、“少し” “もうちょっと” “かなり” など、私たちの使っている多くの言葉は従来の集合では表現しきれない。では、どうしたらよいか？それらを表現するには次節のファジィ集合が適している。

3.2 ファジィ集合の考え方

従来の集合では特性関数が集合の定義に用いられる。これに対しファジィ集合ではメンバーシップ関数により集合を定義できる。Fig.2 は集合“熱い”を定義した例である。

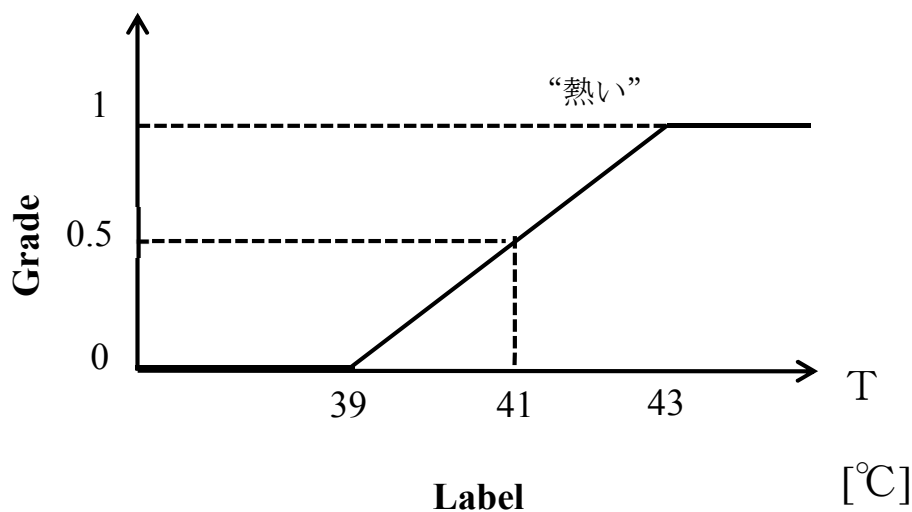


Fig.2 メンバーシップ関数“熱い”

横軸が温度で、縦軸はこの温度が集合“熱い”に属する度合(Grade)を表している。Fig.1 では、43℃は集合“熱い”の境界であった。43℃を境に“熱い”に属する or 属さないが決定していた。Fig.2 の例では 41℃は 0.5 の度合いで“熱い”に属のすることが分かる。43℃を越えてしまうと確かに熱く、つまり所属度 1.0 で属している。また、39℃を下回ると熱いとは言い難く、人によっては“ちょうどよい”や“すこしぬるい”など別のラベルを持つファジィ集合に属することになる。

この表現によれば、40.99℃は所属度 0.499 で 41℃の時との差はわずかである。温度 T に関する“熱い”のメンバーシップ関数は、その関数値が“熱い”の所属度を表し式(4)のように表記される。

$$\mu_{\text{“熱い”}}(T) \quad (4)$$

暑がりや寒がりの人なども、それぞれ Fig.3 に示すように柔軟に表現できる。さらに Fig.4 のように“少し熱い” “熱い” “とても熱い” に対しては、私たちの感覚に近い自然な表現になる。このように境

界のあいまいな集合がファジィ集合と名づけられている． Fig.1 のような従来の境界が明確な集合は，ファジィ集合と区別するために，クリスプ集合と呼ばれる．

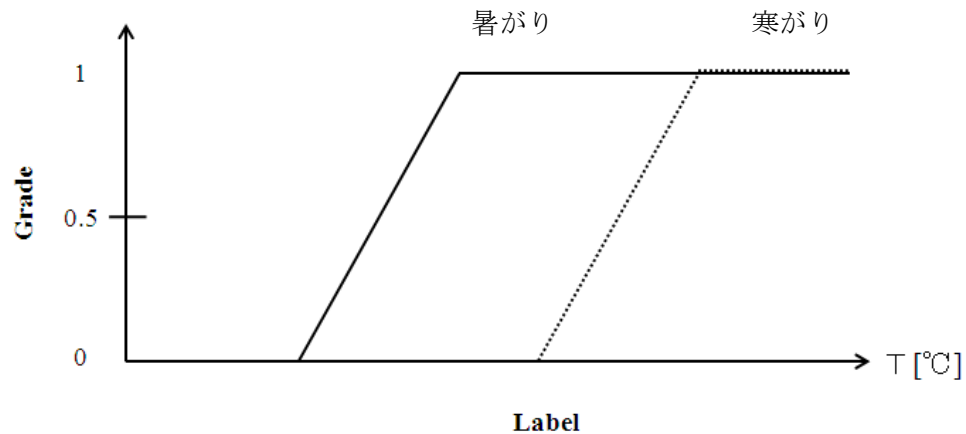


Fig.3 メンバーシップ関数 “暑がりの人” “寒がりの人” の熱い

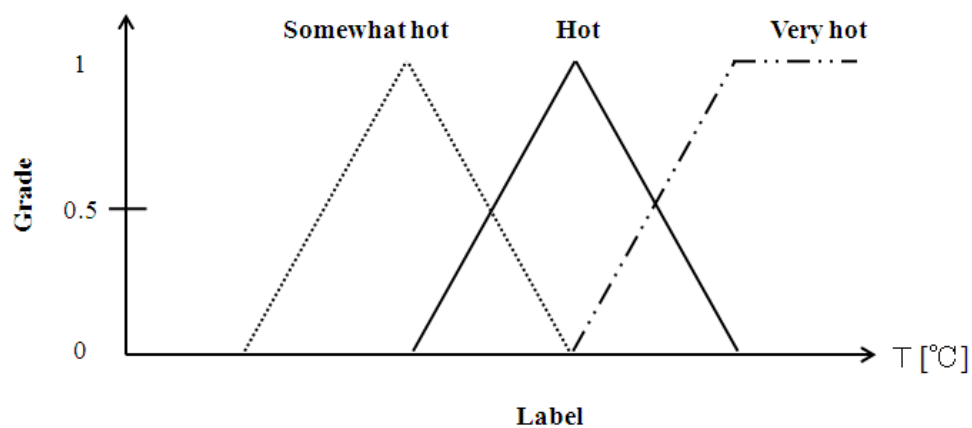


Fig.4 メンバーシップ関数 “少し熱い” “熱い” “とても熱い” (英語版)

演習 3 Fig.4 では，3 つのメンバーシップで”熱い”を表現している．そこで，”普通”，”冷たい”のメンバーシップを考えて，新たなメンバーシップ関数を設計せよ．

演習 4 演習 3 で設計したメンバーシップを C 言語で実現し，エクセルの折れ線グラフで表現せよ．完成したプログラムは `fuzzy4.c` で保存し，グラフは `ensyu4.xlsx` で保存せよ．この後の演習ではこれらをアレンジして使用するため，丁寧にコーディングして流用できるようにコメント等をつけてわかりやすくしておくこと．(付録 1 参照)

4.ファジィ推論とは

4.1 ルールベース推論

前節のファジィ集合を基に、ファジィ推論が考案され、心理学、経済学などの文系の分野から、プラント、自動車、家電製品の制御など、工学の分野まで幅広い応用が進められている。本節では、ファジィ推論について具体例を用いて説明する。

自動車の運転をする時、例えば、加速するとき、減速するとき、カーブを曲がるとき、車庫に入れるときに私たちはアクセルを“少しずつ”ふかし、ブレーキは“控え目に”かけ、ハンドルを“ちょっとずつ”切って、“気持の良い”運転を心掛けたりする。このとき私たちが用いる制御ルールは、いちいち言葉にしなくても、体が覚えていてくれる。このベテランドライバーのテクニックをコンピュータに移植し、制御の自動化を行おうとしたとき、有効な手段にルールベース制御がある。

ルールベース制御は、(5)式で表現されるルールで構成される。

$$\text{もし } x_1 \text{ が } A \text{ で } x_2 \text{ が } B \text{ であれば } y \text{ は } C \text{ である。} \quad (5)$$

これらのルールをコンピュータの中にたくさん用意し、この場合はこの操作、別の場合はこっちの操作と、ルールベースに蓄えられたルールにしたがってコンピュータが自動的に制御を行うものである。ここで、ルールベースにはベテランの経験が記述される。

x_1 , x_2 は車の速度であったり、加速度であったりする。 y は例えばアクセルの踏み具合である。ベテランの経験には前述のような“少し”“きつく”などの言葉がふんだんに入ってくるであろう。 A , B , C をファジィ集合とすることが有効となる。

そこで、もう少し簡単でしかも身近な冷房を具体例に、ファジィ集合を用いたファジィ推論についてみていく。まず、ルールベースによる部屋の温度制御を考える。この制御では部屋の温度を温度計により測り、この測定値を目標値に近づけるようにエアコンのつまみを調整する。ベテランから(6)式のような5個のルールが聴取できたとする。

R_1	:	If “暑い”	Then “とても強く冷す”
R_2	:	If “少し暑い”	Then “強く冷す”
R_3	:	If “ちょうど良い”	Then “中位に冷す”
R_4	:	If “少し涼しい”	Then “弱く冷す”
R_5	:	If “涼しい”	Then “とても弱く冷す”

(6)

もし、従来の境界が明確な集合により、これらのルールの中の言葉を定義したとすると、例えば、Fig.5のように暑さに関する各集合の特性関数を定義できる。ここではグレード値と所属度 X としている。“ち

ちょうど良い”は 26.5℃以上 27.5℃未満とした．同じ様にして，“とても強く冷す”などの言葉に対応したエアコンのつまみの位置を決める．Fig.6 はつまみの模式図である．

“とても強く冷す”	=	1.00
“強く冷す”	=	0.75
“中位に冷す”	=	0.50
“弱く冷す”	=	0.25
“とても弱く冷す”	=	0.00

(7)

(7)式のように決めれば，これでエアコンの制御は可能である．部屋の温度とつまみの関係は Fig.7 に示すような階段状の変化として得られる．

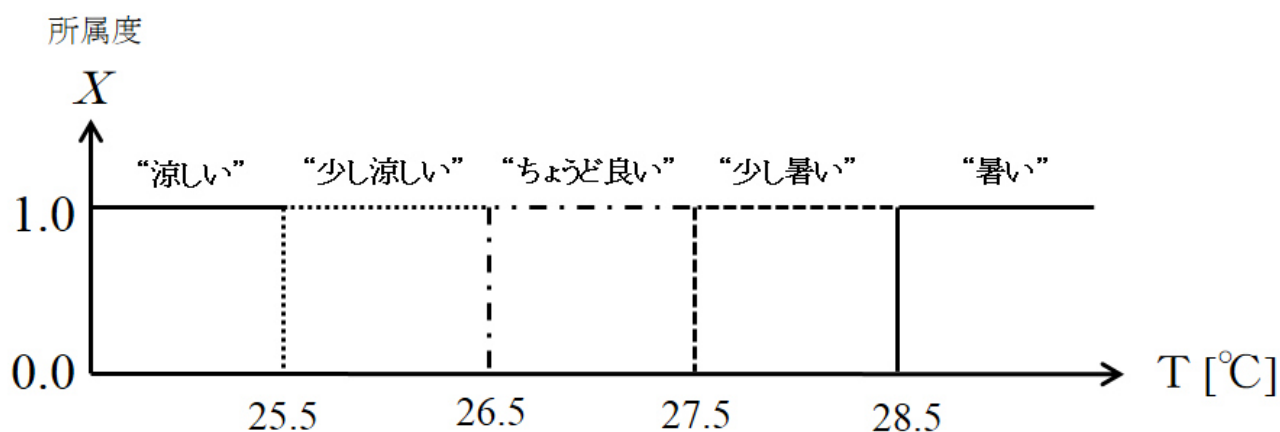


Fig.5 暑さに関する特性関数

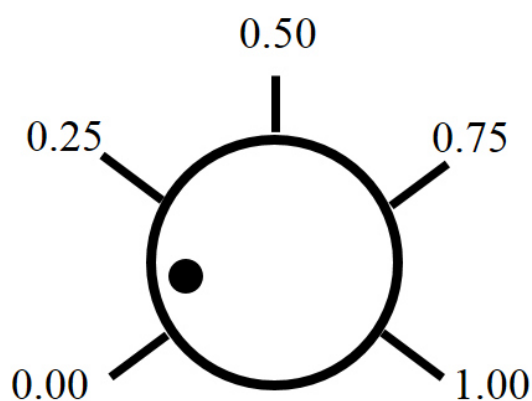


Fig.6 エアコンのつまみ

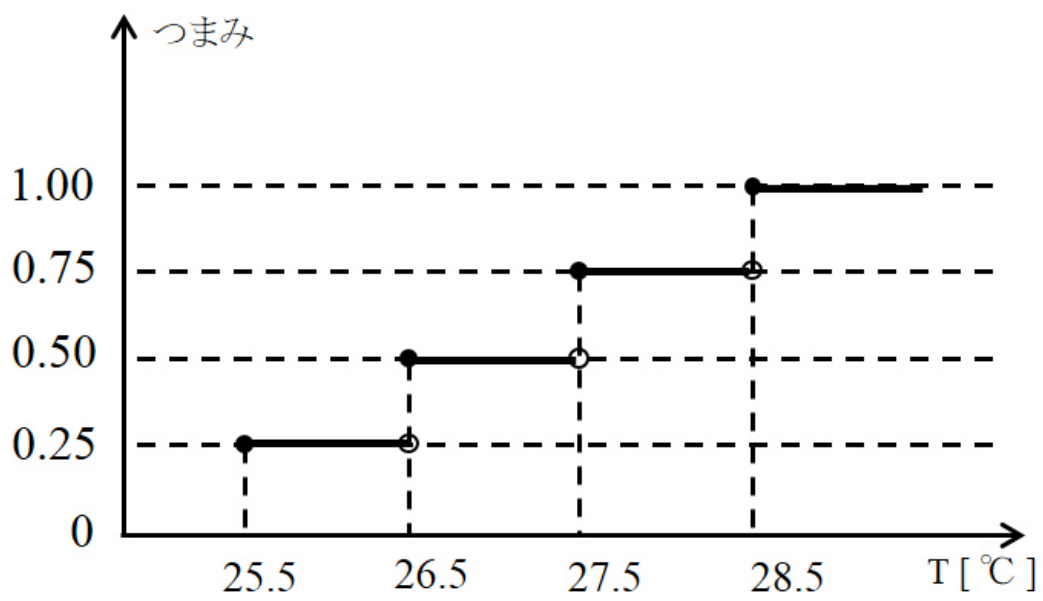


Fig.7 部屋の温度とつまみの関係
(ルールベース制御)

5. ファジィ推論

ファジィ制御で用いられるファジィ推論は、ファジィ論理に基づくファジィ推論法よりも簡略化された推論法を用いる。ファジィ推論は入力情報とファジィ集合をつきあわせて必要な制御操作量を決定することである。主にプロセス制御、エキスパート・システムなどに応用され、家電製品にもごく当たり前に使われている。

ファジィ推論は通常 if (前件部)–then(後件部)型のファジィ制御規則を用いる。一般的な制御方法との違いは、ファジィルールの前件部、後件部にファジィ集合を記述できる点にある。“大きい”、“弱い”など、定量的に規定できない内容を表せる。「もし、自動車の速度が速ければ、減速せよ!」のような人間の経験的知識をルールとして表現できるようになる。次節より代表的なファジィ推論法である、Min-Max 重心法と簡略型推論法の特徴について述べる。

5.1 Min-Max 重心法

Mamdani の推論法は 1974 年、ロンドン大学の Mamdani が最初にファジィ制御で用いた推論法である。この推論法は、推論結果を求めるまでに Minimum 演算、Maximum 演算、重心法を用いているために、Min-Max 重心法とも呼ばれる。彼の成功がファジィ集合の概念の工学的有用性を多くの人に認めさせる結果となった。この推論法は推論手順が理解しやすく、今日までのファジィ制御で最も広く使われている。次節に Min-Max 重心法について述べる。

Min-Max 重心法の計算例

(6)式の中の暑さおよび冷え方に関する言葉のそれぞれにファジィ集合を適用する。Fig.8, 9 はその例である。なお、各メンバーシップ関数は三角形としている。それぞれの集合の境界をあいまいにすることで、ベテランの表現した言葉の感覚に近づけることができている。メンバーシップ関数が決定できたところで、これらの関数とベテランから聴取したルールを用いて部屋の温度制御を行うことになるが、実際の部屋の温度からつまみの位置を求める推論の仕方を決めなければならない。

前節のルールベース制御では、ある室温に対して該当する（発火する）ルールは一つであるので、つまみの位置は一意に決めることができた。集合の境界をあいまいにしたことで、温度の一つの値は複数の集合に属することになる。従って、複数のルールが発火するので、これらのルール間の調整をする必要がある。

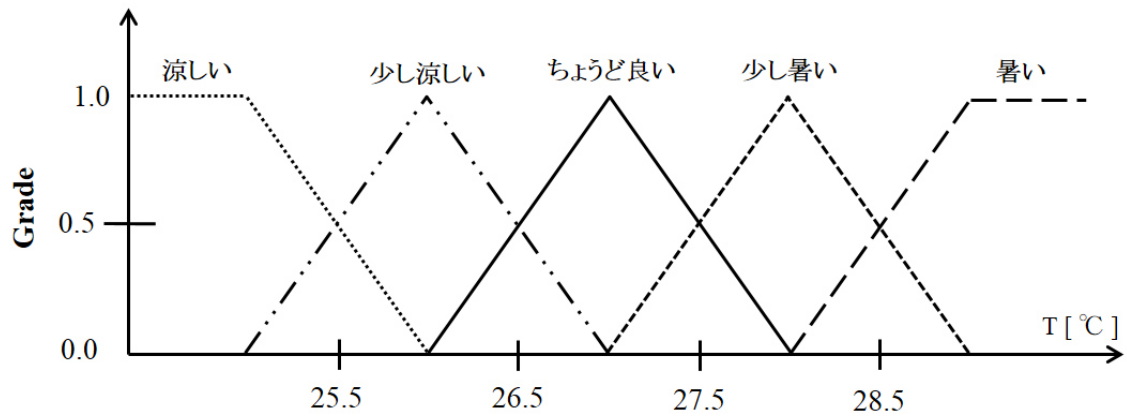


Fig.8 部屋の温度制御のためのメンバーシップ関数“部屋の温度”

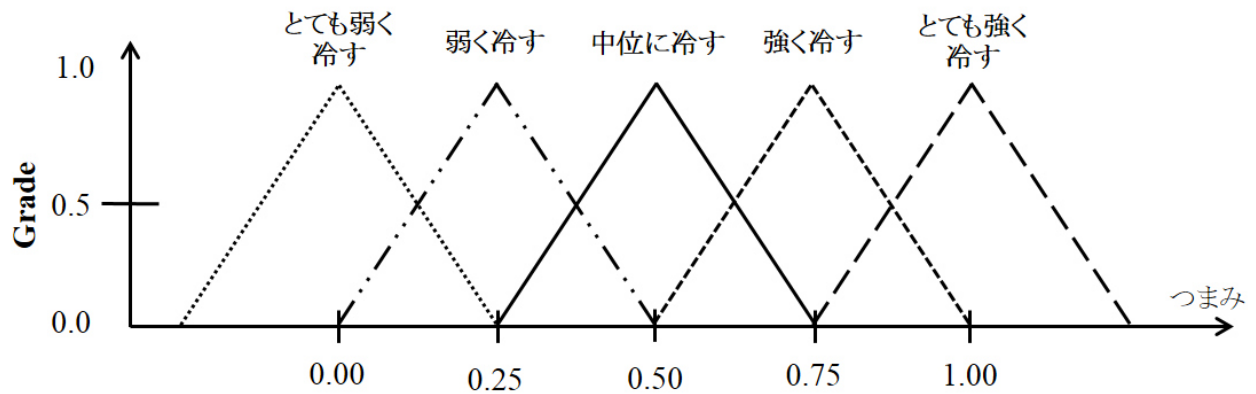


Fig.9 部屋の温度制御のためのメンバーシップ関数“つまみの位置”

演習 5 Fig.8 のメンバーシップのグラフを描く C 言語プログラムを作成せよ．完成したプログラムを fuzzy5.c として保存し，実行結果のグラフは，ensyu5.xlsx として保存せよ．

演習 6 Fig.9 のメンバーシップのグラフを描く C 言語プログラムを作成せよ．完成したプログラムを fuzzy6.c として保存し，実行結果のグラフは，ensyu6.xlsx として保存せよ．

今, Fig.10 に示すように実際の室温が 27.8℃であったとする. このとき, “少し暑い” “ちょうど良い” のメンバーシップのグレードは, (7)式のようにそれぞれ 0.8, 0.2 となる.

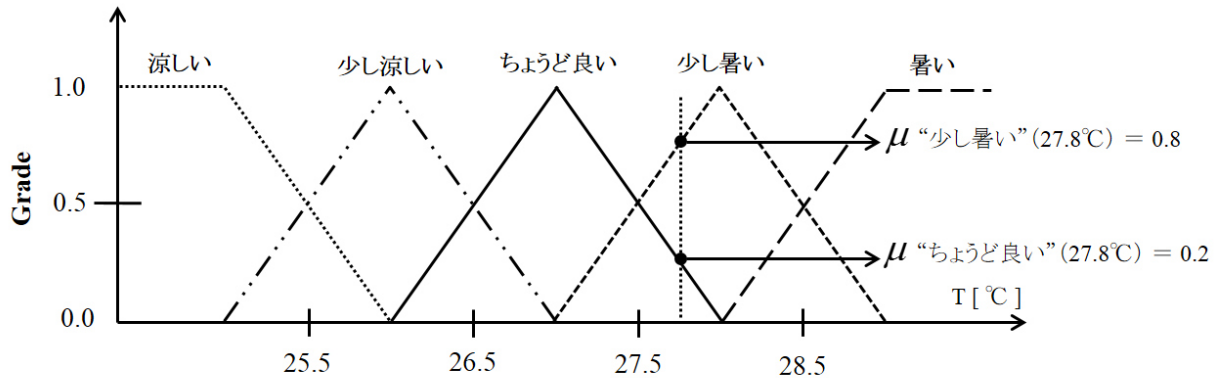


Fig.10 メンバーシップ関数のグレード

$$\begin{aligned}\mu_{\text{“少し暑い”}}(27.8^{\circ}\text{C}) &= 0.8 \\ \mu_{\text{“ちょうど良い”}}(27.8^{\circ}\text{C}) &= 0.2\end{aligned}\tag{7}$$

とも表現される. 関係するルールは(8)式のように定義している.

$$\begin{aligned}R_2 : & \text{ If “少し暑い” Then “強く冷す”} \\ R_3 : & \text{ If “ちょうど良い” Then “中位に冷す”}\end{aligned}\tag{8}$$

これらのルールが今の室温の状況に適合する度合いを適合度と呼び, ルール R_2 , R_3 の適合度 ω_2 , ω_3 を (9)式のように定めることができる.

$$\begin{aligned}\omega_2 &= \mu_{\text{“少し暑い”}}(27.8^{\circ}\text{C}) = 0.8 \\ \omega_3 &= \mu_{\text{“ちょうど良い”}}(27.8^{\circ}\text{C}) = 0.2\end{aligned}\tag{9}$$

ここでは, ルールの適合度と室温に関するメンバーシップのグレードは一致している. 次に, Fig.11 に示す演算を行う. ルール R_2 の適合度が 0.8 であるので “強く冷す” のメンバーシップ関数の頭を 0.8 でアルファカットする. そして “強く冷す” のメンバーシップ関数についても同様に頭をアルファカットする. 残った台形部分が各ルールの推論値となる. 2つの推論値が得られたので, ここで各ルール間の調整をする. この調整は各メンバーシップ関数の最大値を取る合成演算により行う.

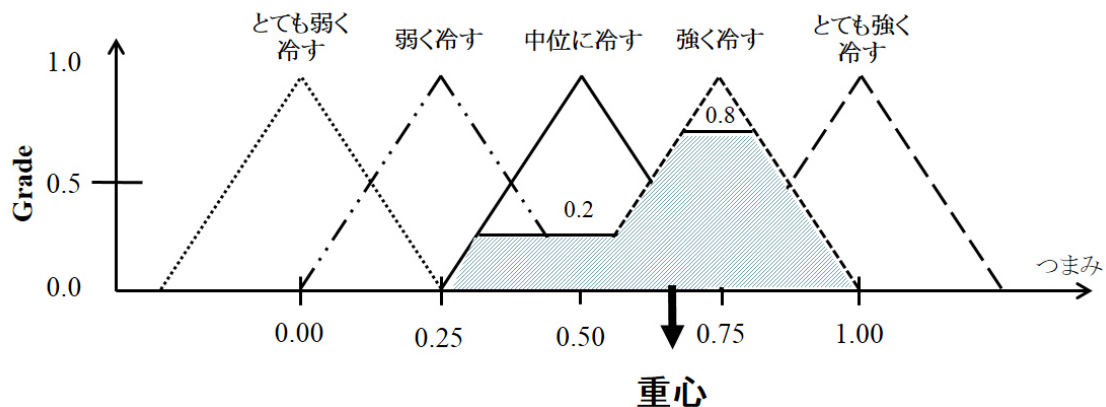


Fig.11 つまみの位置出力

式で書くと(10)式のようなになる.

$$\begin{aligned} \text{つまみの位置} = & \left(\mu_{\text{“強く冷す”}}(\text{つまみ}) \wedge \omega_2 \right) \\ & \vee \left(\mu_{\text{“中位に冷す”}}(\text{つまみ}) \wedge \omega_3 \right) \end{aligned} \quad (10)$$

\wedge は最小値を求める演算であり，前述の頭切り演算である． \vee は最大値を求める演算である．以上により，つまみの位置がファジィ集合として得られた．このままでは，コンピュータにはつまみの位置が分からないので，Fig.11の斜線部の重心位置をつまみの位置とする．上記が，ファジィ推論法の代表的な例とされる，Mamdaniの推論法である．

演習 7 演習 5 および 6 で作成した C 言語プログラムと付録 2 のプログラム(Min-Max 演算)を参考にして，Fig.11 のように合成演算を折れ線グラフで表現せよ．完成したプログラムを fuzzy7.c として保存し，実行結果のグラフは，ensyu7.xlsx として保存せよ．

演習 8 演習 7 で作成した C 言語プログラムを改造して，付録 3 のプログラムを参考にして Min-Max 重心法によるファジィ推論を計算せよ．完成したプログラムを fuzzy8.c として保存せよ．

5.2 簡略型推論法

簡略型推論法は、菅野らによって提案されたもので、基本的には Min-Max 重心法と同じ推論法を行う。異なる点は、重心を求める際にある。前件部には Min 演算を行い、後件部にはファジィ集合ではなく、定数(シングルトン)で与えられる。こうして、ファジィ推論の高速化と簡略化を主眼とした推論法である。前節と同様、次節に簡略型推論法について述べる。

簡略型推論法の計算例

簡略型推論法の前件部には Mamdani の推論法同様 Fig.8 のメンバーシップ関数で表す。つまみの位置の出力としては Fig.9 のメンバーシップ関数ではなく、Fig.12 のシングルトンを用いる。シングルトンはメンバーシップの広がりがないクリスプラベルである。Fig.12 の例では Grade 値 1.0 のシングルトンを用いる。

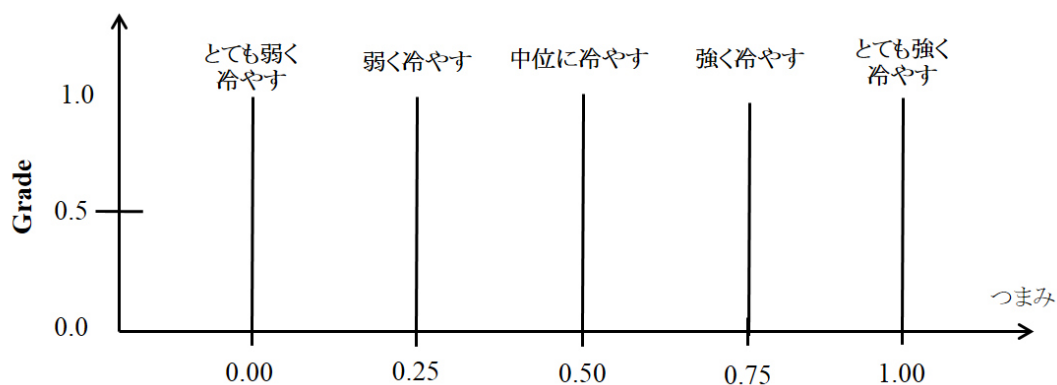


Fig.12 部屋の温度制御のためのシングルトン “つまみの位置”

Fig.10 に示すように実際の室温が 27.8℃であったとする。このとき、“少し暑い” “ちょうど良い” のメンバーシップのグレードはそれぞれ 0.8, 0.2 となり、関係するルールは式(8)となる。同様に適合度 ω_2 , ω_3 は式(9)のようになる。つまみの位置は Fig.13 のように表すことができる。

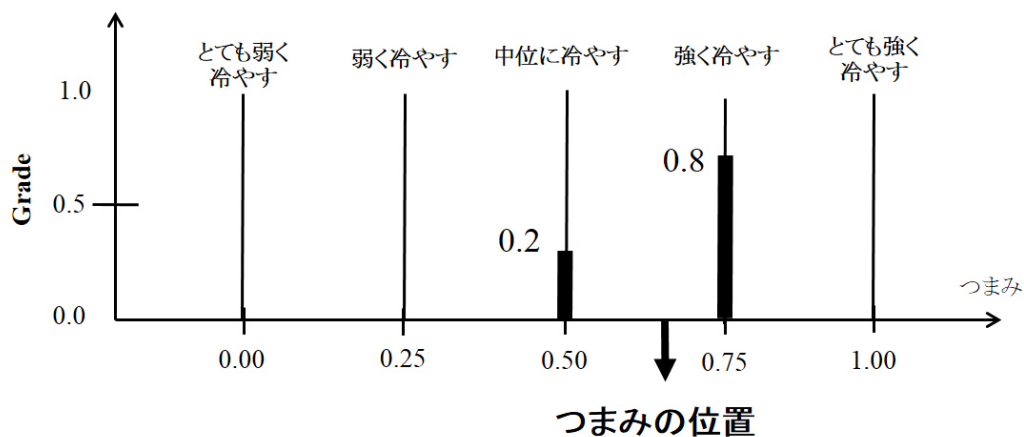


Fig.13 つまみの位置

計算は次のようになり，単純な数値比較(\wedge : min)と四則演算(+, −, \times , \div)だけになる．

$$\begin{aligned}
 \text{つまみの位置} &= ((\mu_{\text{“強く冷す”}}(0.75) \wedge \omega_2) \times 0.75 \\
 &\quad + (\mu_{\text{“中位に冷す”}}(0.5) \wedge \omega_3) \times 0.50) \div (\omega_2 + \omega_3) \\
 &= ((1.0 \wedge 0.8) \times 0.75 \\
 &\quad + (1.0 \wedge 0.2) \times 0.50) \div (0.2 + 0.8) \\
 &= (0.8 \times 0.75 + 0.2 \times 0.50) \div 1 = 0.8 \times 0.75 + 0.2 \times 0.50 \\
 &= 0.7
 \end{aligned}$$

この簡略型推論法は，上記で解説した Mamdani の推論法と比べて，非ファジィ化の演算操作が必要ないため，演算速度が格段に速くなる．またこの推論法は単純な四則演算を使った式となるため，解析が容易になるといった特徴を持つ．

演習 9 これまでの演習で作成したプログラムを参考にして，簡略型推論法を用いた C 言語プログラムを作成せよ．完成したプログラムを fuzzy8.c として保存せよ．

実現されたつまみと温度の関係は Fig.14 のようになる．この図から明らかになったことは，集合の境界をあいまいにし，先ほど解説した推論法を用いることで，ルール間の滑らかな補間を実現できたことである．

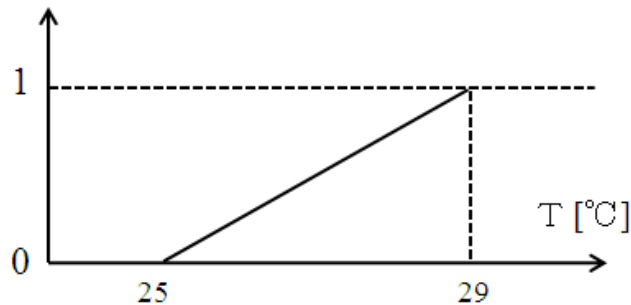


Fig.14 部屋の温度とつまみの関係

演習 10 これまでの演習で作成したプログラムを参考にして，Fig.14 のような関係を示すグラフを作成せよ．完成したプログラムを fuzzy10.c として保存し，実行結果のグラフは，ensyu10.xlsx として保存せよ．

この滑らかな補間はファジィ推論の重要な特徴である．この特徴により，ファジィ集合はベテランのさじ加減をコンピュータに取り込む有力なツールとなる．例えば，ベテランは，“ちょうど良い”あたりをきめ細かく制御し，“暑い”“寒い”ところはおおざっぱに制御していたとする．この加減は，Fig.15 に示す様なメンバーシップ関数で表現されたとする．26.5°Cから 27.5°Cの間のメンバーシップ関数は細かく記述し，その外側は概略の表現になっている．Fig.16 は得られた室温とつまみの関係を示す．

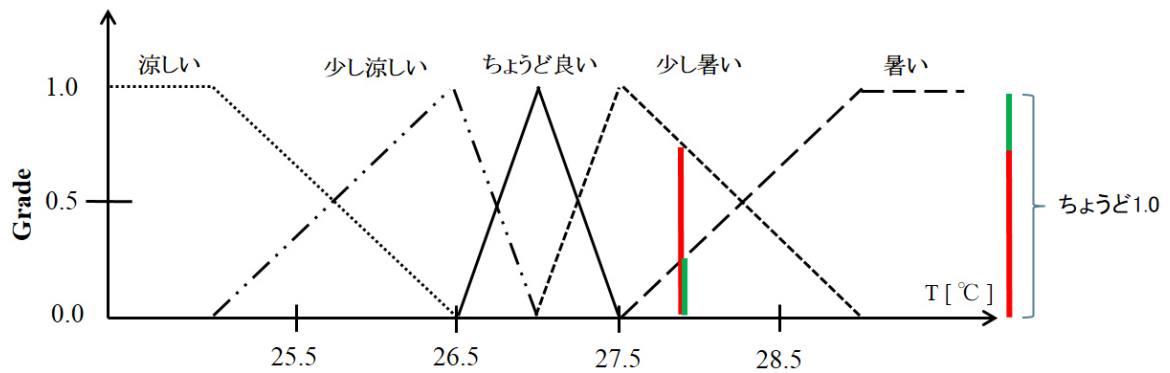


Fig.15 ベテランによる加減の表現

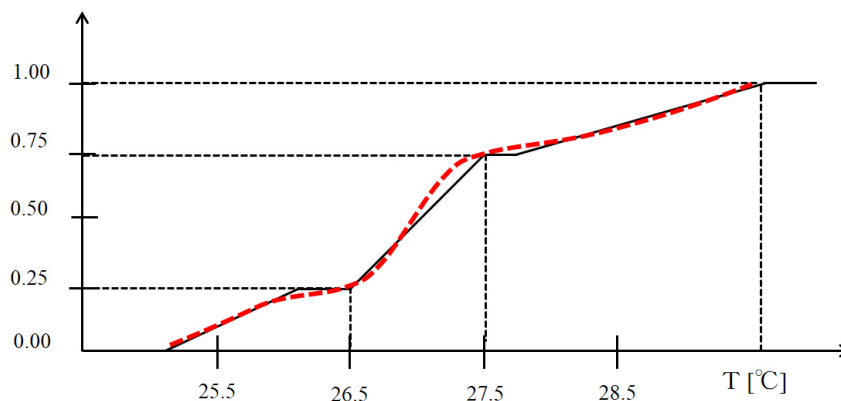


Fig.16 部屋の温度とつまみの関係

図からは、注目している区間ではつまみの変化を大きくし、それ以外のところではだいたい“強く”もしくは“弱く”する制御がたしかに実現できていることが分かる。

【コーヒーブレイク】

多くの Fuzzy 演算の例では、グレード値を[0:1]とし、ファジィ数同士の重なりをちょうど 1.0 にしているが、厳密な理論としてはそうする必要は無い。この場合において、割算が常に 1.0 になるので、計算が省略できる。そうした方が演算コスト上の都合が良い。グレード値も[0:1]である必要もなく、1.0 以上の値を設定しても良いが計算が面倒になる。実は、論理回路でシステムを設計する場合、[0:1]でない方が都合の良い場合がある。これらの厳密な理論については T-ノルムの理論にゆずることにする。興味のある人が測度論、距離論、超平面の理論 等を勉強してみてほしい。ソフトコンピューティングの世界が広がるだろう。

5.4 2 入力への拡張

さて、5.3 節で紹介したベテランのルールは少し単純すぎると言える。

$$R_3 : \text{ If “ちょうど良い” Then “中位に冷す” } \quad (11)$$

(11)式のルールでは、今ちょうど良くても実は暑くなりつつあるかもしれない。(12)式のように少し暑くなってからでは、効率の良い制御方法とは言えない。

$$R_2 : \text{ If “少し暑い” Then “強く冷す” } \quad (12)$$

そのため、温度の変化分まで考慮して、今ちょうど良くても“少し上昇中”であれば強く冷すことにすれば、制御はきめ細やかになり、より効率の良い制御に近づく。Table 1 は室温の変化分 ΔT [°C /分] を新たに考慮して作成したルールを示す。

Table 1 ファジィルール表

$T \Delta T$	降下	少下	零	少昇	上昇
涼	弱弱	弱弱	弱弱	弱	中冷
少涼	弱弱	弱弱	弱	中冷	強
ちょうど	弱	弱	中冷 ω_{13}	強 ω_{14}	強強
少暑	弱	中冷	強 ω_{18}	強強 ω_{19}	強強
暑	中冷	強	強強	強強	強強

温度変化は“降下” “少し降下” “零” “少上” “上昇” の5つの言葉で表現している。表の左上から右へと順番に番号を付けていくと、表の中央のルール R_{13} および、その右隣のルール R_{14} は(13)式のようになる。

$$R_{13} : \text{ If “ちょうど良い” and “温度変化が零” Then “中位に冷す” }$$

$$R_{14} : \text{ If “ちょうど良い” and “温度が少し上昇” Then “強く冷す” }$$

(13)

今回は説明では Mamdani の推論法を使用する．室温とつまみに関しては前節 5.3 と同じメンバーシップ関数を使う．新たに温度変化に関するメンバーシップ関数を定義する必要がある．Fig.16, 17 はこれらのメンバーシップ関数と，実際に室温 27.8℃，温度の変化 0.35℃ / 分が与えられた時のファジィ推論の方法を示している．

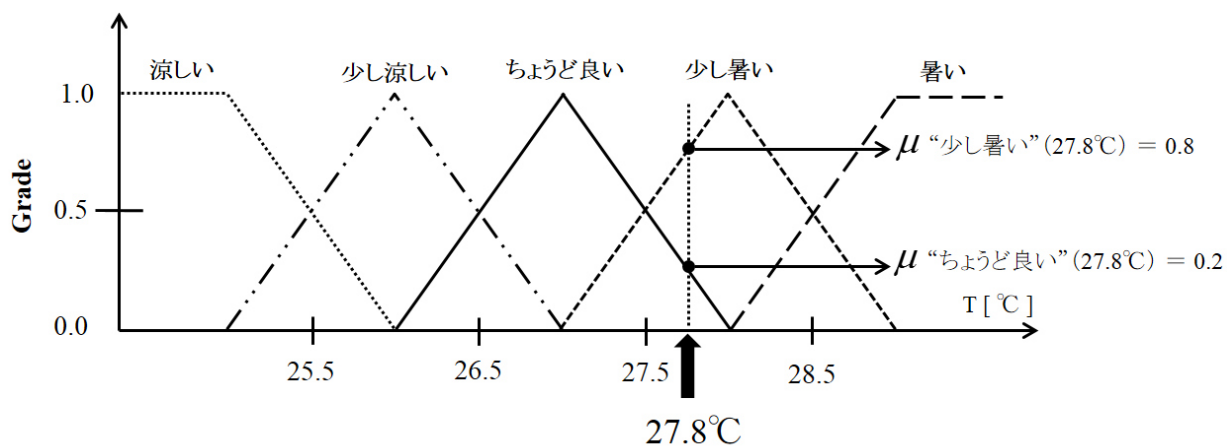


Fig.16 室温のメンバーシップ関数

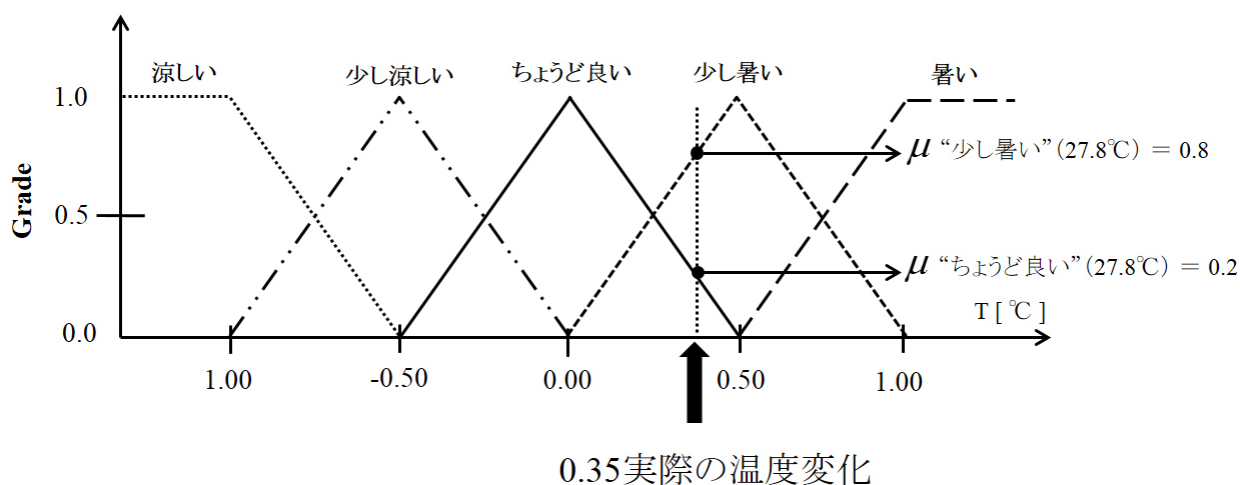


Fig.17 室温変化のメンバーシップ関数

各メンバーシップのグレードは

$$\begin{aligned}\mu \text{ “少し暑い” } (27.8^{\circ}\text{C}) &= 0.8 \\ \mu \text{ “ちょうど良い” } (27.8^{\circ}\text{C}) &= 0.2 \\ \mu \text{ “零” } (0.35^{\circ}\text{C} / \text{分}) &= 0.3 \\ \mu \text{ “少し上昇” } (0.35^{\circ}\text{C} / \text{分}) &= 0.7\end{aligned}$$

(14)

である。関係するルールは、 R_{13} , R_{14} と(15)式の R_{18} , R_{19} となる。

R_{18} : If “少し暑い” and “温度変化が零” Then “強く冷す”
 R_{19} : If “少し暑い” and “温度が少し上昇” Then “とても強く冷す”

(15)

この4ルールの適合度は(16)式のようになる。

$$\begin{aligned}\omega_{13} &= \mu_{\text{“ちょうど良い”}}(27.8^{\circ}\text{C}) \wedge \mu_{\text{“零”}}(0.35^{\circ}\text{C / 分}) \\ &= 0.2 \wedge 0.3 = 0.2\end{aligned}$$

$$\begin{aligned}\omega_{14} &= \mu_{\text{“ちょうど良い”}}(27.8^{\circ}\text{C}) \wedge \mu_{\text{“少し上昇”}}(0.35^{\circ}\text{C / 分}) \\ &= 0.2 \wedge 0.7 = 0.2\end{aligned}$$

$$\begin{aligned}\omega_{18} &= \mu_{\text{“少し暑い”}}(27.8^{\circ}\text{C}) \wedge \mu_{\text{“零”}}(0.35^{\circ}\text{C / 分}) \\ &= 0.8 \wedge 0.3 = 0.3\end{aligned}$$

$$\begin{aligned}\omega_{19} &= \mu_{\text{“ちょうど良い”}}(27.8^{\circ}\text{C}) \wedge \mu_{\text{“少し上昇”}}(0.35^{\circ}\text{C / 分}) \\ &= 0.8 \wedge 0.7 = 0.7\end{aligned}$$

(16)

したがって、ここで求められる、つまみの位置は(17)式を用いた推論となる。

$$\begin{aligned}\text{つまみの位置} &= (\mu_{\text{“中位に冷す”}}(\text{つまみ}) \wedge \omega_{13}) \\ &\vee (\mu_{\text{“強く冷す”}}(\text{つまみ}) \wedge \omega_{14}) \\ &\vee (\mu_{\text{“強く冷す”}}(\text{つまみ}) \wedge \omega_{18}) \\ &\vee (\mu_{\text{“とても強く冷す”}}(\text{つまみ}) \wedge \omega_{19})\end{aligned}$$

(17)

したがって Fig.18 に示すようにこの重心位置が最終的なつまみの位置となる。

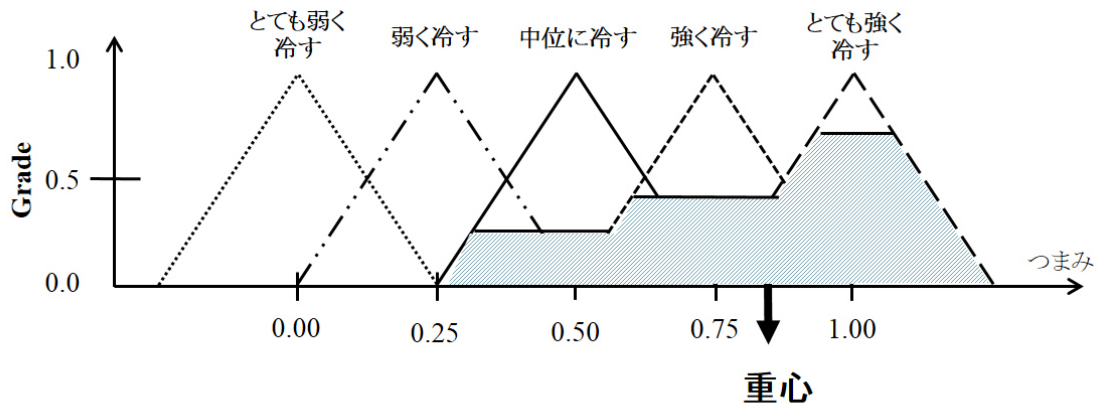


Fig.18 つまみのメンバーシップ

以上の Mamdani の推論法は、適合度を Minmum 演算で求め、つまみの位置（推論値）を Maximum 演算と重心により求めていることから、Min-Max 重心法とも呼ぶ。

Mamdani の推論法は、それを用いた制御器設計が容易であり、しかも制御結果がとても良かったことから、良いとされている。しかし、なぜこの推論法が良いのか？について理論的根拠がない。基本的に、それまで出されなかった性能を出せたのか、もしくはそれまでより簡単に作れるようになった等のメリットがあれば、工学的には意味がある。ファジィ推論法は、現在様々なバリエーションが提供されている。

終了課題 これまでの演習で作成したプログラムを参考にして、Fig.18 に示すような 2 入力の場合のプログラムを完成し、関係を示すグラフを作成せよ。2 入力であるため、グラフ表現に工夫が必要である。グループで考えて、グラフを掲載したプレゼンテーションを作成して発表準備をせよ。完成したプログラムを `fuzzy_final.c` として保存し、実行結果のグラフは、`final.xlsx` として保存せよ。

付録 1 C 言語(gcc)で作るメンバーシップ関数(fuzzy_member_ship.c)

ターミナル(Windows ならコマンドプロンプト)上で動作するの表示ソフトでメンバーシップ関数について学んでみよう.

- コンパイル : gcc fuzzy_member_ship.c -lm
- 実行 : a.out > membership.csv (Windows なら a.exe)

画面にメンバーシップ関数の Grade 値が表示される. これらの数値データを gnuplot やエクセルでグラフを描いてみよう. 傾斜型, 三角型, 逆傾斜型, 台形メンバーシップのメンバーシップの計算結果が表示される. プログラムの最後では Grade 値を入力 x に対しての出力値を表示している. カンマ, 区切りで表現しているので, リダクション機能等を使えば csv ファイルにしてエクセルに読み込ませることができる. ターミナルの出力をマウスでコピーペーストしてエクセルに張り付けてグラフを作成しても良い. メンバーシップ関数の特性を可視化して理解する事はとても重要なので, PC 等で確かめよう.

```
#include <stdio.h>
// (furoku.2:top)
// (furoku.3:top)

// 傾斜型メンバーシップのグレード値計算関数
// 入力: value 左下: x0 右上: x1
double FuzzyGrade(double value, double x0, double x1){
    double result = 0;
    double x;

    x = value;

    if(x <= x0)
        result = 0;
    else if(x >= x1)
        result = 1;
    else
        result = (x - x0) / (x1 - x0);
    return result;
}

// 逆傾斜型メンバーシップのグレード値計算関数
// 入力: value 左上: x0 右下: x1
double FuzzyReverseGrade(double value, double x0, double x1){
    double result = 0;
    double x;

    x = value;

    if(x <= x0)
        result = 1;
    else if(x >= x1)
        result = 0;
    else
        result = (x1 - x) / (x1 - x0);
    return result;
}
```

```

// 三角型メンバーシップのグレード値計算関数
// 入力: value 左下: x0 中央上: x1 右下: x2
double FuzzyTriangleGrade(double value, double x0, double x1,
double x2){
    double result = 0;
    double x;

    x = value;

    if(x <= x0 || x >= x2)
        result = 0;
    else if(x == x1)
        result = 1;
    else if((x > x0) && (x < x1))
        result = (x - x0) / (x1 - x0);
    else
        result = (x2 - x) / (x2 - x1);
    return result;
}

// 台形メンバーシップのグレード値計算関数
// 入力: value 左下: x0 左上: x1 右上: x2 右下: x3
double FuzzyTrapezoidGrade(double value, double x0, double x1,
double x2, double x3){

    double result = 0;
    double x;

    x = value;

    if(x <= x0 || x >= x3)
        result = 0;
    else if(x >= x1 && x <= x2)
        result = 1;
    else if((x > x0) && (x < x1))
        result = (x - x0) / (x1 - x0);
    else if((x > x2) && (x < x3 ))
        result = (x3 - x) / (x3 - x2);
    return result;
}

int main(){
    double x,x0,x1,x2,x3;

    x0 = 3.0; x1 = 5.0; x2 = 9.0; x3 = 13.0;

    printf("Member ship function output sample¥n");
    printf("x , FuzzyGrade, FuzzyReverseGrade,
        FuzzyTriangleGrade, FuzzyTrapezoidGrade ¥n");
    for (x=0.0; x< 20.1 ; x += 0.1){
        printf ("%2.2f , %2.2f , %2.2f , %2.2f , %2.2f ¥n",
            x,FuzzyGrade(x,x0,x1),FuzzyReverseGrade(x,x0,x1),
            FuzzyTriangleGrade(x,x0,x1,x2),
            FuzzyTrapezoidGrade(x,x0,x1,x2,x3) );
    }
    return 0;
}

```

エクセルの散布図で表示してみる． $x_0 = 3.0$; $x_1 = 5.0$; $x_2 = 9.0$; $x_3 = 13.0$; と main 関数で宣言されている． グラフを描いてみると， Fig.19 に示すようになる． この場合傾斜型 (x_0, x_1) は実線， 逆傾斜 (x_0, x_1) は破線のようになる． Fig.20 を見てみよう． 三角型 (x_0, x_1, x_2) は実線， 台形 (x_0, x_1, x_2, x_3) は破線でそれぞれ描かれている． それぞれの設定を変えて試してみると良いだろう．

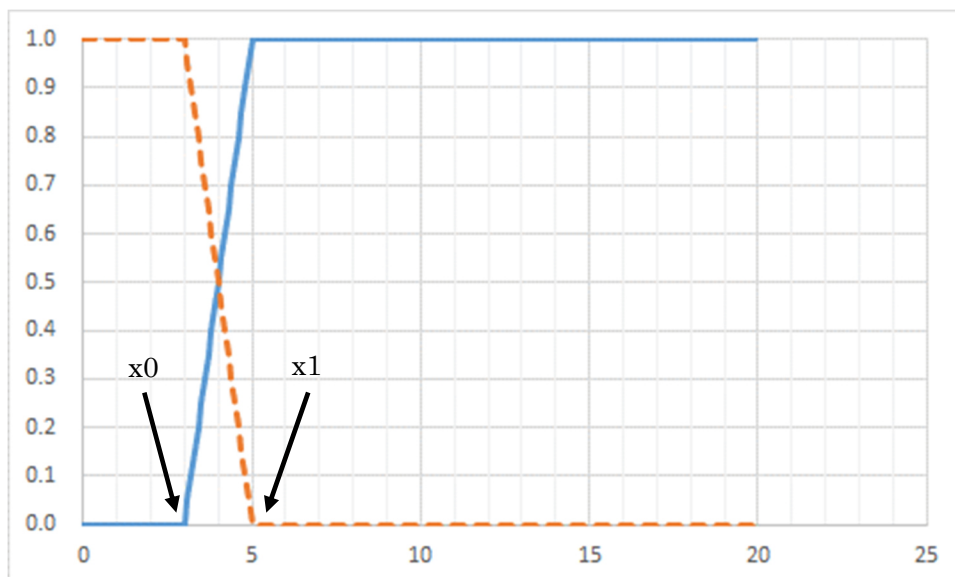


Fig.19 傾斜と逆傾斜のメンバーシップ関数

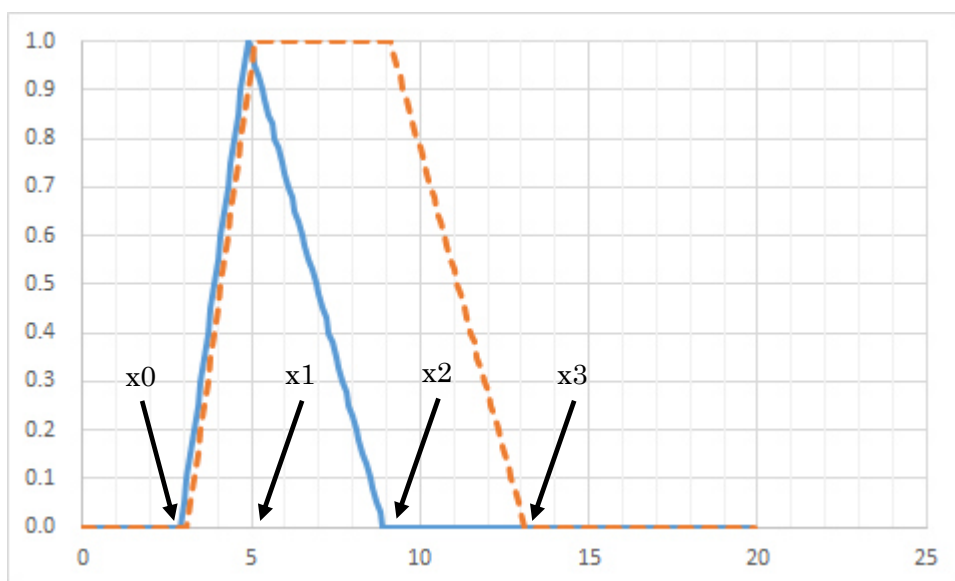


Fig.20 三角形と台形のメンバーシップ関数

付録 2 C 言語(gcc)で作る min max 演算(fuzzy_member_ship.c) 一部抜粋

付録 1 のプログラムに上側の部分を付け加えて、main 関数を変更するとメンバーシップ関数を OR, AND, NOT, アルファカット演算した結果は表示される 付録 1 と同じく、ターミナル(Windows なら コマンドプロンプト)上で動作する表示ソフトでグラフ化して確認してみよう。簡単な演算なので、付録 1 と同様にグラフ線が重なってしまうことに注意してグラフを見てみよう。

- コンパイル : gcc fuzzy_min_max_cut.c -lm
- 実行 : a.out > min_max_cut.csv (Windows なら a.exe)

《上側の部分はここから》 ※付録.1 のプログラムの(furoku.2:top)以下に加筆すると良い。

```
#define MAX(a,b) ((a)<(b)?(b):(a))
#define MIN(a,b) ((b)<(a)?(b):(a))

// Max 演算の計算処理
// グレード値入力 : A,B
double FuzzyAND(double A, double B){
    return MIN(A,B);
}

// Min 演算の計算処理
// グレード値入力 : A,B
double FuzzyOR(double A, double B){
    return MAX(A,B);
}

// Max 演算の計算処理
// グレード値入力 : A,B
double FuzzyMIN(double A, double B){
    return MIN(A,B);
}

// Min 演算の計算処理
// グレード値入力 : A,B
double FuzzyMAX(double A, double B){
    return MAX(A,B);
}

// Not 演算の計算処理
double FuzzyNOT(double A){
    return (1.0 - A);
}

//アルファカット演算
double FuzzyAlphaCut(double A, double alpha){
    return MIN(A,alpha);
}
```


《main 関数の変更はここから》

```
int main(){
    double x;
        double A,Ax0,Ax1;
        double B,Bx0,Bx1,Bx2;
        double C,Cx0,Cx1;

    Ax0 = 5.0; Ax1 = 10.0;
    Bx0 = 5.0; Bx1 = 10.0; Bx2 = 15.0;
    Cx0 = 10.0; Cx1 = 15.0;

    printf("Member ship function output sample¥n");
    printf("x , A, B, C, FuzzyAND(A:B), FuzzyOR(B:C),
        FuzzyNOT(B), FuzzyAlphaCut(B:0.3) ¥n");
    for (x=0.0; x< 20.1 ; x += 0.1){
        A = FuzzyReverseGrade(x,Ax0,Ax1);
        B = FuzzyTriangleGrade(x,Bx0,Bx1,Bx2);
        C = FuzzyGrade(x,Cx0,Cx1);
        printf ("%2.2f , %2.2f , %2.2f , %2.2f ,
            %2.2f , %2.2f , %2.2f , %2.2f¥n",x, A ,B, C,
            FuzzyAND(A,B), FuzzyOR(B,C), FuzzyNOT(B),
            FuzzyAlphaCut(B,0.3));
    }
    return 0;
}
```

単純に散布図を描くと Fig.21 のようになる。詳細を見てみよう。エクセルに張り付けたデータのうち、メンバーシップ A, B, C はそれぞれ左の 3 行にある。main 関数の printf 関数の引数変数の順番をみたら理解できるだろう。その他は AND, OR, NOT 演算が並んでいる。エクセルの表示を変更して、ファジィ演算の特性を視覚的に理解してほしい。もちろん gnuplot でも描くことは可能である。また、いろいろなメンバーシップを設計して、いろんな演算をやってみても良いだろう。この後のファジィ推論に向けたファジィルールをデザインする上でもこれらを視覚的にイメージしておくことはとても重要である。

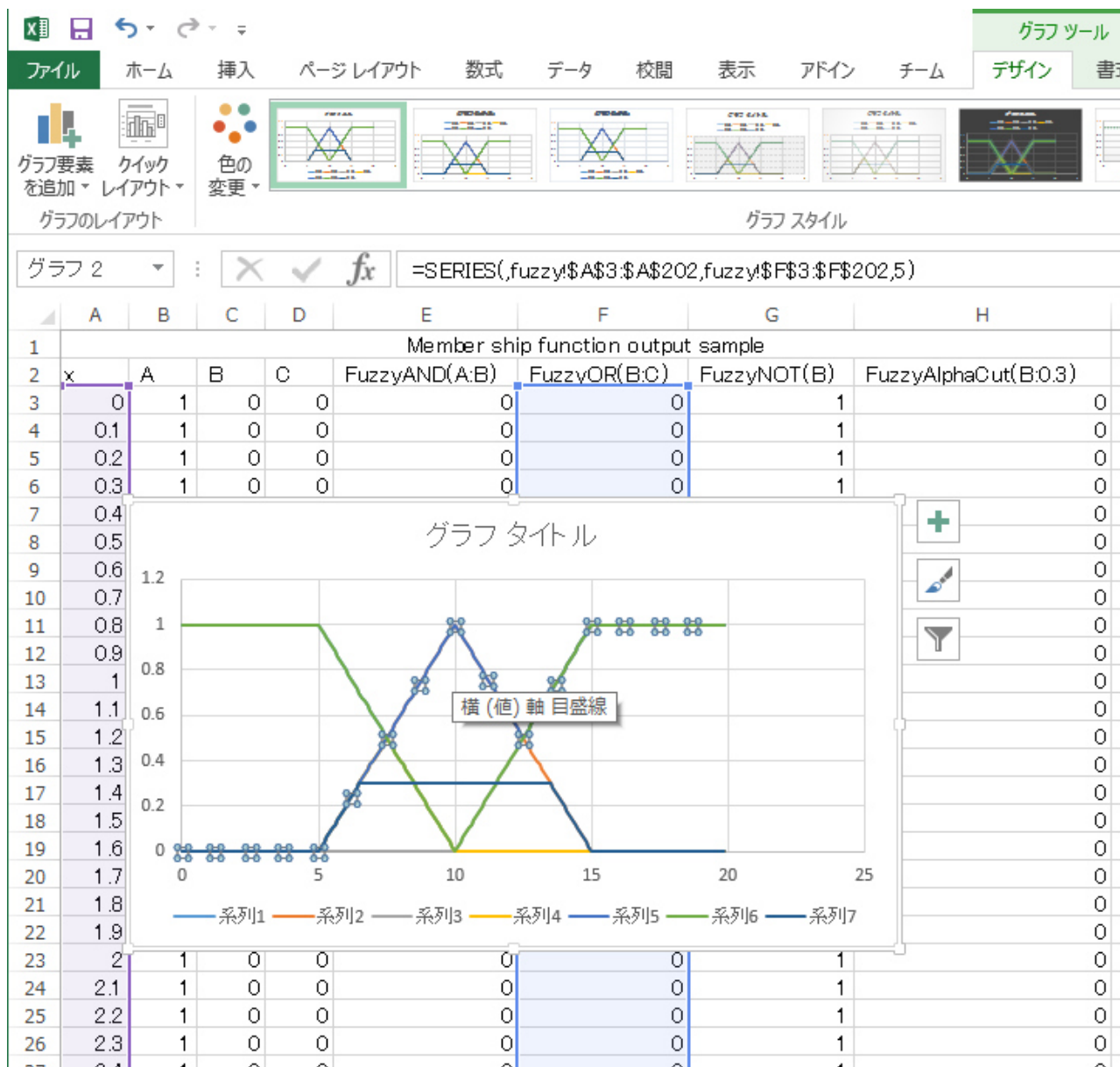


Fig.21 エクセルで描画した MIN 演算・MA 演算・アルファカット (α カット) 演算

付録 3 C 言語(gcc)で作るファジィ推論(fuzzy_defuzzification.c) 一部抜粋

付録 2 のプログラムに上側の部分を付け加えて、main 関数を変更するとメンバーシップ関数を OR, AND, NOT, アルファカット演算した結果は表示される。付録 2 と同じく、最もシンプルなターミナルソフト(Windows ならコマンドプロンプト)の表示ソフトになっている。グラフ化して確認してみよう。

- コンパイル: `gcc fuzzy_min_max_cut.c -lm`
- 実行: `a.out > min_max_cut.csv` (Windows なら `a.exe`)

《上側の部分はここから》※付録.1 のプログラムの (furoku.3:top) 以下に加筆すると良い。

```
double FuzzyInfluence(double x[], double grade[], int xsize){
    int i;
    double defuzz, total_gx, total_g;

    defuzz = 0; total_gx = 0; total_g = 0;
    for(i=0;i<xsize;i++){
        total_gx = total_gx + grade[i]*x[i];
        total_g = total_g + grade[i];
    }
    if(total_g != 0.0) defuzz = total_gx / total_g;
    else                defuzz = 0.0;

    return defuzz;
}
```

《main 関数の変更はここから》

```

int main() {
    double x, set_x[256];
    double A, Ax0, Ax1;
    double B, Bx0, Bx1, Bx2;
    double C, Cx0, Cx1;
    double A02andB03, B03orC;
    double Fuzzy_A02andB03[256], Fuzzy_B03orC[256];
    double FInf_x_A02andB03, FInf_x_B03orC;
    int i, count_size;

    Ax0 = 5.0; Ax1 = 10.0;
    Bx0 = 5.0; Bx1 = 10.0; Bx2 = 15.0;
    Cx0 = 10.0; Cx1 = 15.0;
    i = 0;

    printf("# Member ship function output sample 3rd\n");
    printf("# x , A, B, C, FuzzyAND(A02:B03),  

                                                    FuzzyOR(B03:C)  \n");
    for (x=0.0; x< 20.1 ; x += 0.1){
        A = FuzzyReverseGrade(x, Ax0, Ax1);
        B = FuzzyTriangleGrade(x, Bx0, Bx1, Bx2);
        C = FuzzyGrade(x, Cx0, Cx1);
        A02andB03 = FuzzyAND( FuzzyAlphaCut(A, 0.2),  

                                FuzzyAlphaCut(B, 0.3) );
        B03orC      = FuzzyOR( FuzzyAlphaCut(B, 0.3), C );
    }
}

```

```

printf ("%2.2f , %2.2f , %2.2f , %2.2f , %2.2f ,
        %2.2f ¥n",x, A ,B, C, A02andB03, B03orC);

/* Memory the result Fuzzy Op*/
set_x[i] = x;
Fuzzy_A02andB03[i] = A02andB03;
Fuzzy_B03orC[i]    = B03orC;
i++;
}

count_size = i;

// Defuzzification
printf("# Defuzzification ¥n");
printf("# FInf_x_A02andB03 , FInf_x_B03orC ¥n");
FInf_x_A02andB03 = FuzzyInfluence(set_x,
                                   Fuzzy_A02andB03, count_size);
FInf_x_B03orC = FuzzyInfluence(set_x,
                                Fuzzy_B03orC, count_size);
printf("# %2.2f , %2.2f ¥n", FInf_x_A02andB03,
        FInf_x_B03orC);
return 0;
}

```

出力結果は次のようになる.

Defuzzification

FInf_x_A02andB03 , FInf_x_B03orC

7.50 , 14.83

単純に散布図を描くと Fig.22 のようになる。詳細を見てみよう。エクセルに張り付けたデータのうち、メンバーシップ A, B, C はそれぞれ左の 3 行にある。main 関数の printf 関数の引数変数の順番をみたら理解できるだろう。FuzzyAND(A02:B03)のデータは A を 0.3 でアルファカットし、B を 0.3 でアルファカットした後、AND した計算結果であり、また出力結果の FInf_x_A02andB03 はその横軸重心を計算した結果である。FuzzyOR(B03:C)のデータは B を 0.3 でアルファカットし、C と OR した計算結果であり、また出力結果の FInf_x_B03orC はその横軸重心を計算した結果である。エクセルの表示を変更して、ファジィ演算とファジィ推論の特性を視覚的に理解してほしい。もちろん gnuplot でも描くことは可能である。また、いろいろなメンバーシップを設計して、いろんな演算をやってみても良いだろう。この後のファジィ推論に向けたファジィルールをデザインする上でもこれらを視覚的にイメージしておくことはとても重要である。

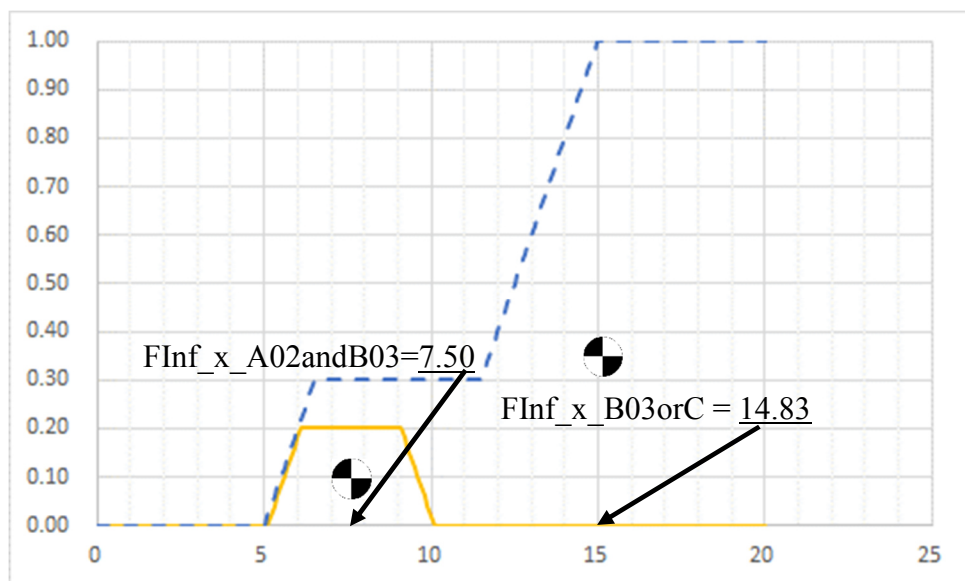


Fig.22 複雑な形状のメンバーシップ関数を重心法で推論