距離型ファジィ推論法に基づく
推論エンジンの開発

吉本　幸司
学籍番号 1020167
知能機械システム工学科
知能ロボティクス研究室

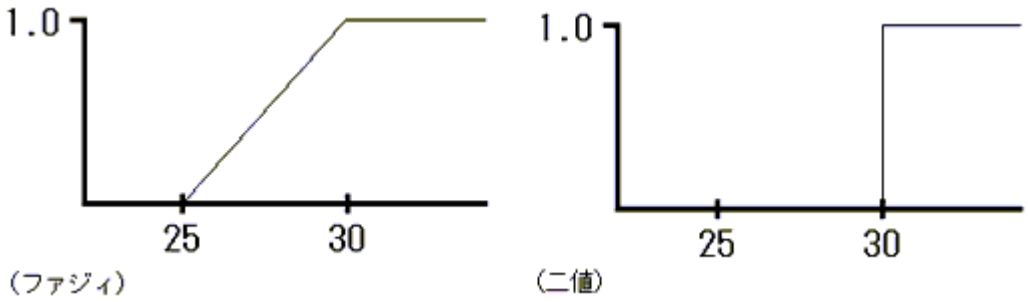
目次

第一章　序章

１－１　はじめに

ファジィは、「曖昧なもの」「はっきりしないもの」という風に理解されがちだが、実際には二値では表現できないものを精密に表現する事の出来るとても優れた表現法である。

ファジィを使った例として、「もし温度が 30℃以上ならば、クーラーを強めよ」というような制御があるとし、（図 1）に示すように縦軸に評価値、横軸に温度をとったグラフを考える。ここで、もし温度が 29.9℃だとする。29.9℃というのは限りなく 30℃に近いと言える。しかし二値の表現で判断する場合、グラフから分かるようにまだ 30℃に達していない為、この例は無視される。対してファジィで表現した場合、（図 1）から分かるように 30℃以上であるという事に対して、評価値 0.9 ぐらいの可能性が残される。これはどういう事かと言うと、人間に近い評価がファジィには可能なのである。ファジィは少し値が大きい、ほんの少し値が小さい、という風に漠然としたデータでも取り扱うことが出来る。



（図 1）

つまりファジィ推論というのは、変数に曖昧さを持たせる事により１つの規則の守備範囲に広がりを持たせる事ができる訳である。これにより、事実がどのように変化しても各規則の条件部がぼやけているので、部分的に適合する規則が必ず存在する。しかも、事実の変化に沿った適切な結論を導き出す事ができる。

これらのことから、ファジィ推論は限られた知識の中からあらゆる事実に対して結論を導き出せる良いモデルであり、その枠組みは人間の持つある種の表層的な推論機能をモデル化するのに良い近似手段を与えるものだと言う事ができる。

１－２　研究目的と意義

現在、ファジィという言葉は世の中に溢れているにも関わらず、ファジィというものがどういうものなのかという事は、多くの人が知らないというのが事実である。ファジィを使うとどういう事ができるのか、ファジィはどういう事を行うのに便利なのか、それを知ればファジィはもっと一般的に世の中に浸透していくはずである。

前節で述べたように、ファジィを使った推論を行うと YES もしくは NO の値しか返せない二値での推論に比べて、YES と NO に加えて YES と NO の中間、またはどちらかといえば YES より、どちらかといえば NO より、など様々な値を返すことができる。このファジィ推論を用いたファジィ制御は､現在あらゆる分野で活躍している｡例を挙げると、月桂冠はファジィ制御を使った清酒醸造をしている。

このように、ファジィ制御は世の中に存在しながらも、一部の工学者、企業などが使用しているのみである。一般の人々は、例えば家電製品などに搭載されたファジィ推論機能を外側から使用するのみであり、ファジィ制御の中身については知ることはない。

ファジィというものがもっと一般的に世の中に浸透していくには、ファジィというものが一般的に知られ、ファジィ推論というものがどういうものなのかを理解してもらう必要があり、それには実際にファジィ推論を行ってみるのが一番の方法である。

しかし、ファジィを使って推論を行うには様々な知識が必要である。まずファジィというものを理解する必要があり、その上計算方法についての知識も必要となる。特に、計算方法などは誰もが知識を持っている訳ではない。そこで、ファジィについての知識、計算についての知識が無くとも、視覚的に、また感覚的にファジィを使った推論が行える推論エンジンがあれば、ファジィというもの、ファジィ推論というものを簡単に知ることができ、様々な人達に、様々な用途で使ってもらえるであろう。

本論文は、ファジィについての知識が無い人にも簡単に使え、また知識のある人にも十分使える推論エンジンの開発を目的とした研究について述べる。本研究で開発する推論エンジンには、従来のファジィ推論法ではなく距離型ファジィ推論法を用いる。この距離型ファジィ推論法を用いることで、他の推論法には持っていない特徴が得られる。これらの詳細は後ほど、第二章で述べる。

ここで言う「推論エンジン」というのは、推論の手順及び流れが組み込まれたプログラムであり、この推論エンジンを基にファジィ推論の学習及びファジィ制御シミュレーション用、ファジィ制御用の 2 種類の推論アプリケーションを開発する。

１－３　研究概要

距離型ファジィ推論法を用い、２つの推論エンジンを開発する。まず、初心者の方にも簡単に使うことができるように考慮して開発した推論エンジン、そして実際の制御にも使えるようにしたファジィについて知識のある人向けの推論エンジンである。

前者の推論エンジンの開発環境には VisualBasic を用い、後者の推論エンジンの開発環境には VisualC++を用いる。このように、これら２つの推論エンジンは開発環境が互いに異なる訳だが、2 つの開発環境を用意した理由については第三章で述べる。

さらに、距離型ファジィ推論法が実際の制御に精密かつ強力に使用できることを証明する為、前者の推論エンジンにはシミュレーション機能を設け、このシミュレーション機能を使いファジィ制御シミュレーションを行う。このシミュレーションにはオイラー法を用いる。

今回の研究では、車の速度を時速６０ｋｍに維持するファジィ制御をこのシミュレーション機能を使い実験する。

第二章　距離型及びその他のファジィ推論法の比較

２－1　様々なファジィ推論法とその欠点

現在、モデル化できない非線形システムの制御によく使われているファジィ推論にはマムダニの推論法、関数型推論法、簡略型推論法などがあげられる。中でも、多くのファジィ制御に使用されている推論法は簡略型推論法である。

この簡略型推論法が多くのファジィ制御に使われている理由として、ルールの後件部をファジィ集合ではなく定数でおくことで、推論結果も実数値として出る為 min-max 重心法等の推論法に比べて重心計算をする必要が無い、という事が挙げられる。また、重心計算をしないことによって演算速度が格段に速くなるという事も理由に挙げられる。

簡略型推論法等の上記の推論法は、ルールの前件部と与えられた事実との共通集合の高さを、前件部と事実がどの程度一致しているか、ということから適合度を計算する。しかし、この推論方法にはいくつかの欠点がある。例として、前件部をシングルトンで置くと適切な推論結果を導くことができない。また、前件部集合が互いに空の共通部分を持つ場合は推論結果を求めることができない。分離規則を厳密に満たしていない、等が挙げられる。

ここで、それらの欠点を示す具体例として、視覚的に推論の流れが理解し易いと思われる min-max 重心法を紹介する。この min-max 重心法はマムダニの推論法とも呼ばれ、後件部のグレード値を求める際には前件部の解に対して AND 演算（ Min ）を用い、推論結果を求める際には OR 演算（ Max ）を用いる。AND 演算とは、データを比較した場合に小さい方のデータをとる事であり、対して OR 演算（ Max ）は、大きい方のデータをとる事である。

例えば、min-max 重心法を用いてトマトの赤さから熟度を推論する場合、以下のようなルールを設定し、与える。

ルール１　　IF　(赤さ is BG)　THEN　(熟度 is BG)

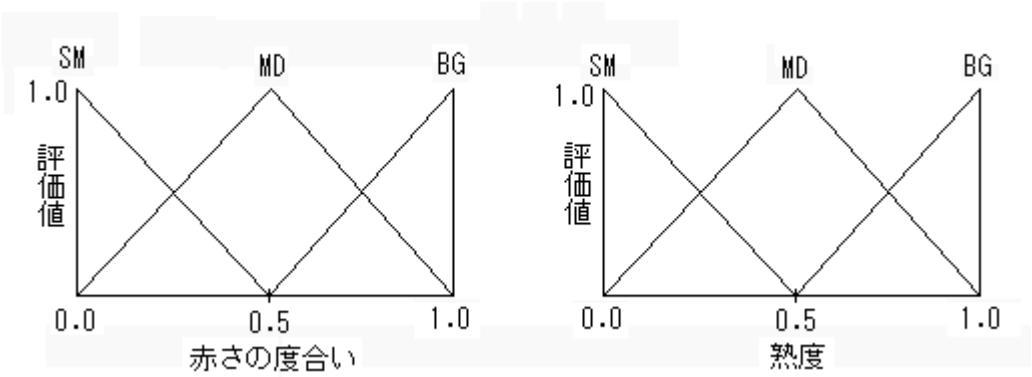
ルール２　　IF　(赤さ is MD)　THEN　(熟度 is MD)

ルール３　　IF　(赤さ is SM)　THEN　(熟度 is SM)

BG…BIG　MD…MEDIUM　SM…SMALL

この場合「赤さ」が前件部、「熟度」が後件部となる。

これら３つのルールをそれぞれ三角型ファジィ集合で図示すると（図 2）のようになる。



（図２）

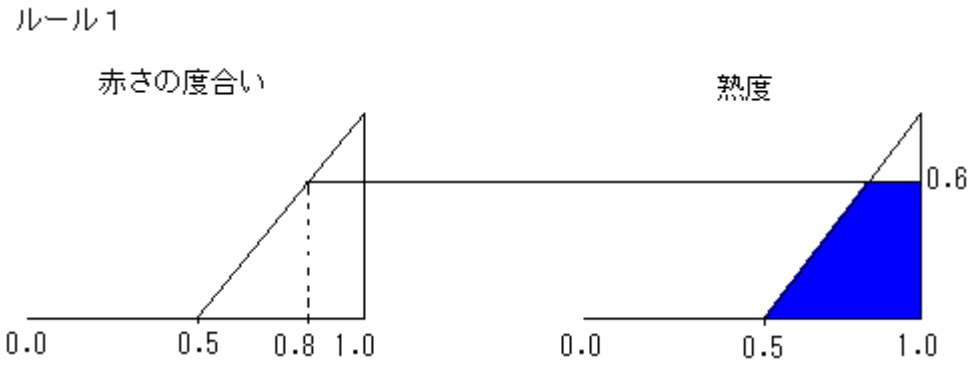
これらのルールに対して赤さの度合いが 0.8 であったとする（この 0.8 を「事実」という）。この事実を３つのルールにそれぞれ当てはめてみる。すると、以下のような結果が得られる。

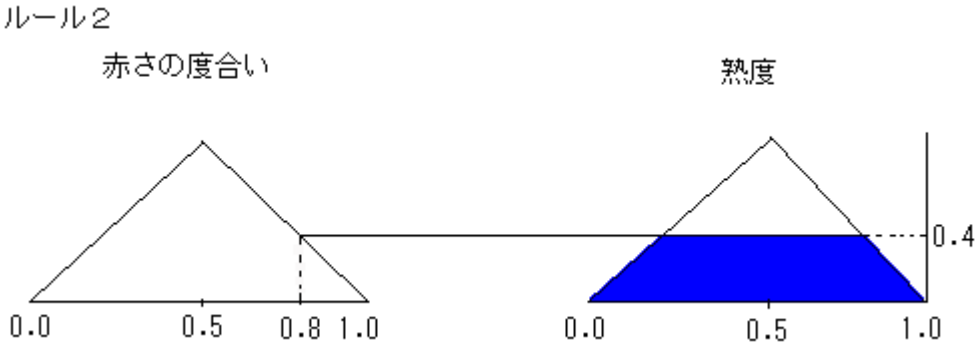
BG の適合度　　0.6

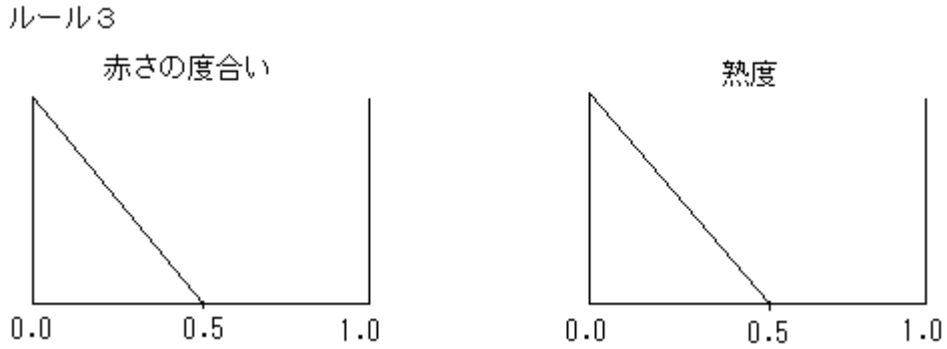
ＭＤの適合度　　0.4

SM の適合度　　0

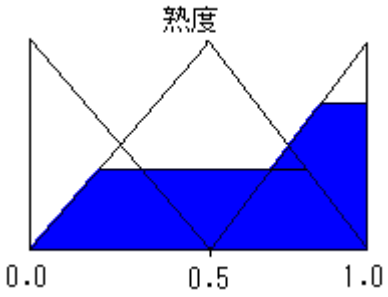
これで、前件部のデータは全て求まった。これらの得られたデータから AND 演算、OR 演算を行い、推論結果であるファジィ集合が得られる。演算の様子は（図３）に示す。







↓　OR 演算



（図３）

最後にこのファジィ集合の重心をとって、これを推論結果（ 解 ）とする。重心を求める公式は、重心を y0、横軸を y、合成したファジィ集合をμ(y)として、(1）式で与えられる。

$$y_0 = \frac{\int y * \mu(y)\,dy}{\int \mu(y)}$$

（1）

この場合、重心は 0.681 となる。

以上が min-max 重心法を使った推論法である。推論の流れから、分離規則を満たしていない、事実が前件部のどの集合にも属さない場合には推論が不可能である、等の弱点が理解できたと思う。

推論を精度良く確実に実行する為には、これらの欠点を克服する必要がある。次節で、これらの欠点を克服した距離型ファジィ推論法の特徴について述べる。

２－３　距離型ファジィ推論法の特徴

前に述べたように、min-max 重心法等の直接法が、推論方法をルールの前件部と与えられた事実との共通集合の高さを、前件部と事実がどの程度一致しているかを示す適合度としているのに対し、距離型ファジィ推論法はファジィ集合間の距離に基づいた推論を行う。

ファジィ集合間の距離計算法について述べる。まず、色々な計算法を定義する事ができるが、以下の３つの公理を満たさなければならない。

$$d(A,B) \geqq 0 \; ; \; d(A,B) = 0 \quad \Leftrightarrow \quad A = B \tag{2}$$

$$d(A,B) = d(B,A) \tag{3}$$

$$d(A,B) \leqq d(A,C) + d(C,B) \tag{4}$$

ここで、凸ファジィ集合を対象として以下のような一つの簡単な距離計算法を提案する。この計算法は、連続メンバーシップ関数を持つファジィ集合を対象としているが、シングルトン値をメンバーシップ関数の幅を零に収束させた時の極限状態として考えられるので、シングルトンにも適用できる。

$$d(A.B) = \left[\int_0^1 \left| \inf A_{\alpha} - \inf B_{\alpha} \right|^p d\alpha \right]^{1/p} + \left[\int_0^1 \left| \sup A_{\alpha} - \sup B_{\alpha} \right|^p d\alpha \right]^{1/p} + \left[\int_{-\infty}^{\infty} \left| \left(\frac{1}{M_A} - 1 \right) \mu_A(x) - \left(\frac{1}{M_B} - 1 \right) \mu_B(x) \right|^p dx \right]^{1/p} \tag{5}$$

inf はファジィ集合の下限値を表し、sup は上限値を表す。MA,MB はファジィ集合の高さを表す。

本論文では、上式のｐを２として、一般的なファジィ集合間の距離計算式を与えている。本研究の推論エンジンの推論過程としては、具体的に次の３ステップからなる。

Step1) 上記の公理を満たす距離関数に基づき、事実と前件部ファジィ集合との距離を計算する。

Step2) 以下の（6）式を使い、定数 Kq を計算する。

$$Kq = \frac{\prod_{j=1, j\neq q}^{n} dj}{\sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} dj} \qquad (q = 1, 2, 3, \cdots, n)$$

ｎ・・・ルール数

dj・・・距離

（6）

Step3) 最後に、（７）式よりメンバーシップ関数が導かれる。

$$\mu_B(y)=\begin{cases}\dfrac{1}{\sum_{q=1}^{n} Kq\left(a_2^q-a_1^q\right)}\left(y-\sum_{q=1}^{n} Kq\,a_1^q\right) & \sum_{q=1}^{n} Kq\,a_1^q \le y \le \sum_{q=1}^{n} Kq\,a_2^q \\ \dfrac{1}{\sum_{q=1}^{n} Kq\left(a_2^q-a_3^q\right)}\left(y-\sum_{q=1}^{n} Kq\,a_3^q\right) & \sum_{q=1}^{n} Kq\,a_2^q \le y \le \sum_{q=1}^{n} Kq\,a_3^q \\ 0 & y < \sum_{q=1}^{n} Kq\,a_1^q \text{ or } y > \sum_{q=1}^{n} Kq\,a_3^q \end{cases} \quad (7)$$

＊ Step4) ファジィ制御などに重心の値を求める必要がある場合、重心 y0 は（８）式から求めることができる。

$$y0 = \frac{1}{3} \sum_{q=1}^{n} Kq \left(a_1^q + a_2^q + a_3^q \right)$$

ｎ・・・ルール数
ｑ・・・ルール番号
ａ・・・後件部ファジィ集合データ

（８）

距離型ファジィ推論法には、以下のような特徴が挙げられる。

１、分離規則を満たす

２、後件部が凸のファジィ集合であれば、推論結果も凸なファジィ集合となる

３、漸近特性を持つ

４、前件部集合が互いに疎な場合でも推論可能

分離規則とは、A=A1 なら B=B1 となる事である。つまり、前件部集合 A１に事実 A が重なった場合、後件部 B１に推論結果 B が重なる事を指す。

漸近特性とは、前件部集合 A１と事実 A の距離が０に収束すれば、後件部 B１と推論結果 B の距離も０に収束することを指す。

前件部集合が互いに疎な場合でも推論可能というのは、事実がどの前件部集合にも属すことができない場合でも推論可能であるという事を示す。

第三章　推論エンジンの開発

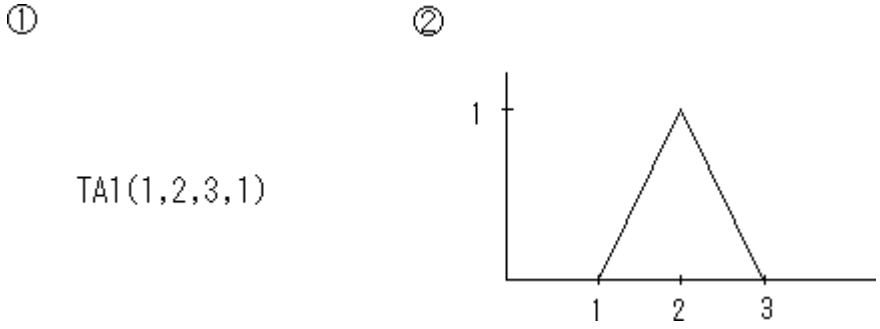
３－１　それぞれの推論エンジンの特徴

この節では、VisualBasic 及び VisualC++を用いて開発した推論エンジンのそれぞれの特徴について述べる。

今回、「初心者向けの推論エンジン」、「実際の制御用の推論エンジン」という風に２つの推論エンジンを開発した。推論法には第二章で述べた距離型ファジィ推論法を用いることで、従来の推論法では対応しきれなかった状況に対しても推論可能となっている。

まず、『初心者の方にも簡単に使える』という目標を持って開発した推論エンジンについてだが、開発環境は VisualBasic である。この推論エンジンには様々な機能が設けられている。ファジィについて知識の無い人が、ファジィ集合のデータをコンピューターに対して入力し、ファジィ集合を自分の頭の中で想像することは極めて困難である。そこで、ファジィ集合の自動入力システムを採用した。これは、コンピューターが三角型ファジィ集合やシングルトン等、ごく一般的なファジィ集合を設定ルール数自動でグラフ上に描いてくれるものである。ユーザーは、この自動入力されたファジィ集合のデータポイントをクリック・ドラッグ・ダブルクリック等することでファジィ集合を視覚的・感覚的に任意の場所まで動かすことができ、形を自由に変えることもできる。なお、ファジィ集合の移動等、変化に伴い事実と前件部の距離、推論結果であるメンバーシップ関数、その重心などリアルタイムで計算結果として表示し、推論結果であるファジィ集合もその都度表示されるようになっている為、どのようにメンバーシップ関数が動いているか等、自由に見て、動かして、体験することができる。

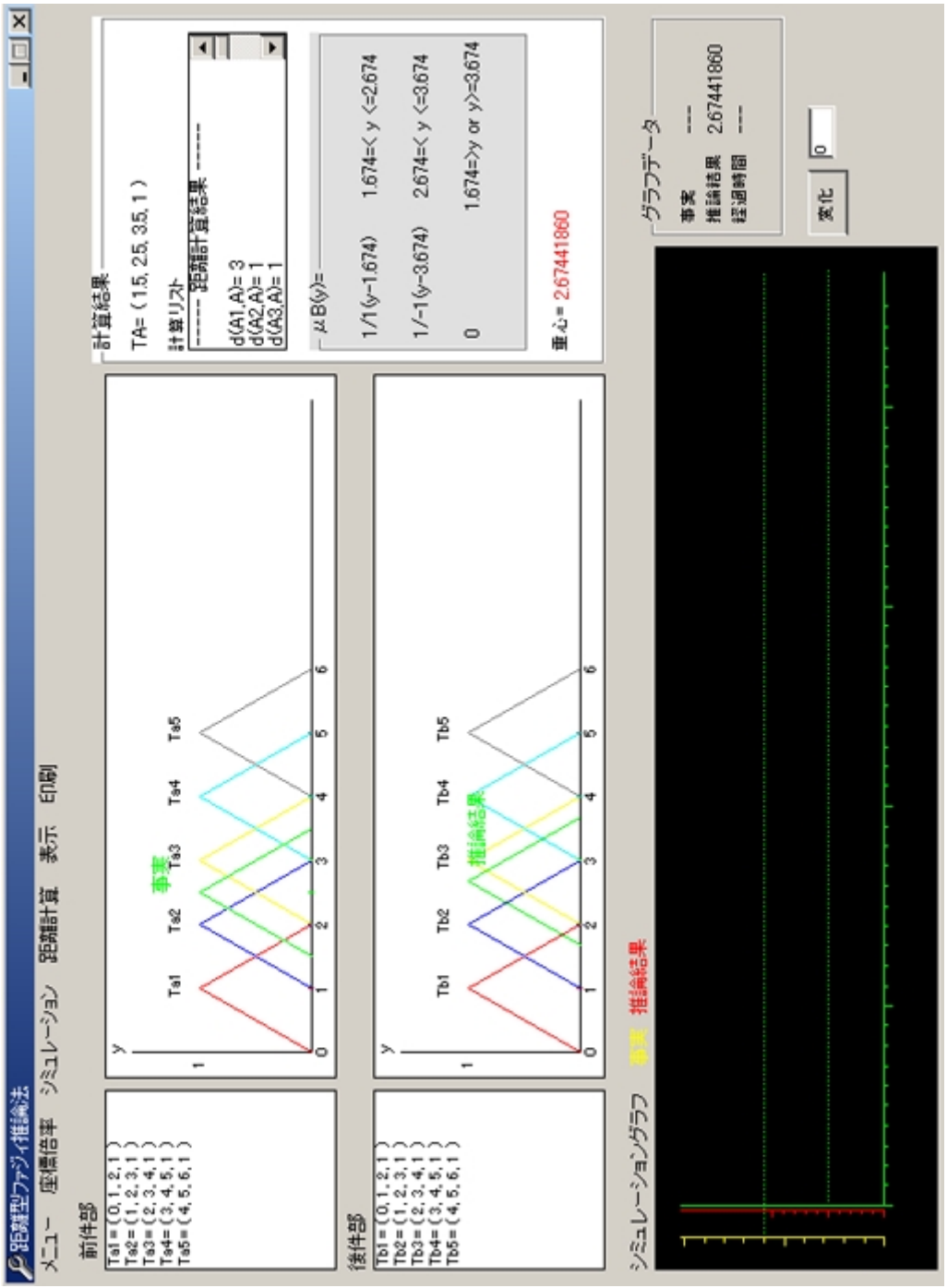
ファジィ集合等、初心者に理解し易くする為には、『グラフィック表示』は極めて重要な要素である。例として、（図 4）の①と②は同じ意味であるが、ファジィについて知識の無い人にとっては、②のグラフィック表示されたファジィ集合を視覚的に理解する方が、①のデータから理解するよりも格段理解し易い。



（図 4）

よって、グラフィック表示を特に重要視した。例として、ルールが数種類有る場合には、各ルールの前件部・後件部に色づけをし、ルールごとに一目で分かるようにしている。（図５）に実行中の画面を示す。

（実行画面）


距離型ファジィ推論法
メニュー 座標倍率 シミュレーション 距離計算 表示 印刷
前件部
Ta1 = (0, 1, 2, 1)
Ta2 = (1, 2, 3, 1)
Ta3 = (2, 3, 4, 1)
Ta4 = (3, 4, 5, 1)
Ta5 = (4, 5, 6, 1)
後件部
Tb1 = (0, 1, 2, 1)
Tb2 = (1, 2, 3, 1)
Tb3 = (2, 3, 4, 1)
Tb4 = (3, 4, 5, 1)
Tb5 = (4, 5, 6, 1)
事実
推論結果
計算結果
TA= (1.5, 2.5, 3.5, 1)
計算リスト
----- 距離計算結果 -----
d(A1,A)= 3
d(A2,A)= 1
d(A3,A)= 1
μB(y)=
1/1(y-1.674) 1.674=< y <=2.674
1/-1(y-3.674) 2.674=< y <=3.674
0 1.674=>y or y>=3.674
重心= 2.67441860
グラフデータ
事実 ---
推論結果 2.67441860
経過時間 ---
変化
シミュレーショングラフ 事実 推論結果

（図５）

（図５）の状態は、例として以下のようにルールを５つ与えて実行している状態の画像である。

ルール１

IF　x= Ta1（0，1，2，1）　THEN　y= Tb1（0，1，2，1）

ルール 2

IF　x= Ta2（1，2，3，1）　THEN　y= Tb2（1，2，3，1）

ルール 3

IF　x= Ta3（2，3，4，1）　THEN　y= Tb2（2，3，4，1）

ルール 4

IF　x= Ta4（3，4，5，1）　THEN　y= Tb2（3，4，5，1）

ルール 5

IF　x= Ta5（4，5，6，1）　THEN　y= Tb2（4，5，6，1）

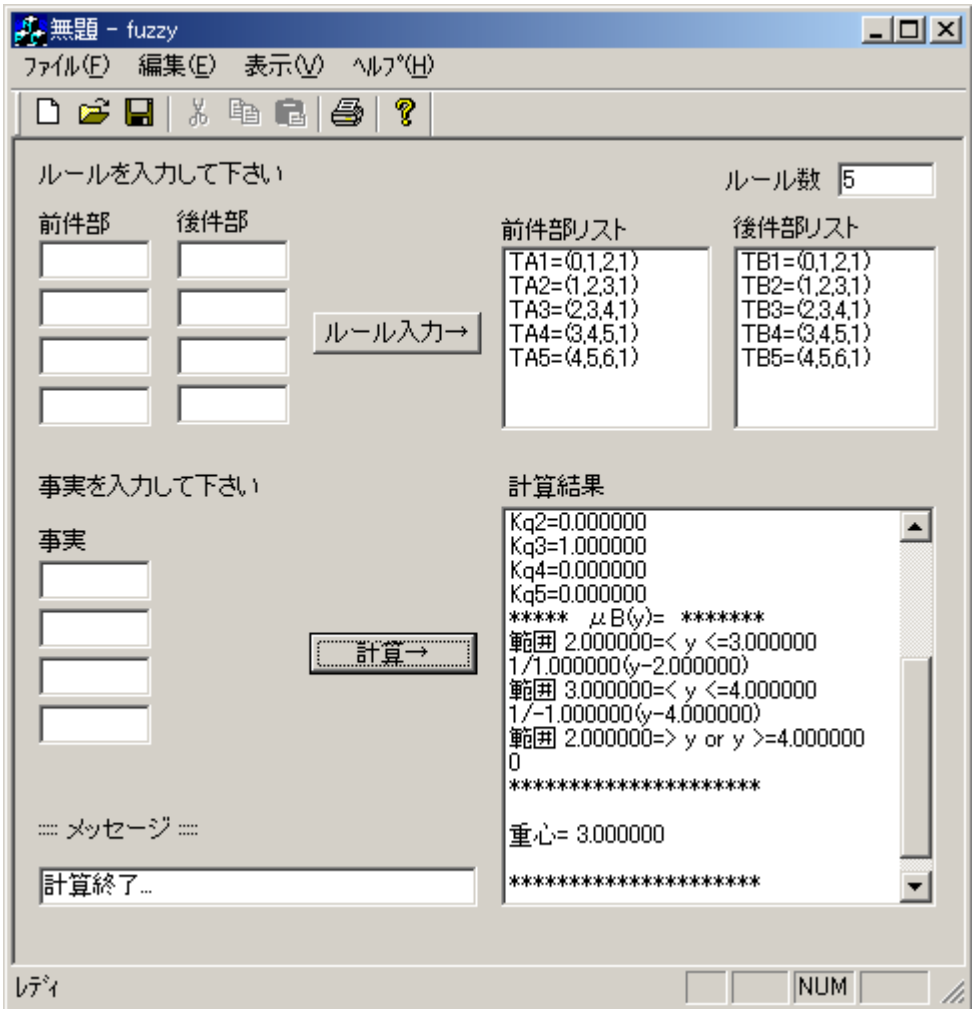
図を見てもらうと理解できるが、事実をファジィ集合 Ta（1.5，2.5，3.5，1）のように与えており、前件部・後件部にそれぞれ事実・推論結果が緑線で表示されている。また画面左側にはこれら前件部及び後件部のデータが表示され、右側には事実、距離、メンバーシップ関数、重心などのデータが表示されている。

このように、視覚的にルールの中での事実の位置、推論結果の位置が簡単に理解でき、またデータとしても理解することができる。

図の下側にはシミュレーショングラフがあるが、この推論エンジンにはさらにシミュレーション機能を設けることで、時間の概念を取り入れた様々な実験をユーザーが行うことも可能としている。これにより、ファジィ推論・ファジィ制御というものを実際に体験することができ、よりファジィというものを知ることができる。

次に、実際のファジィ制御に使用する事を想定した推論エンジンだが、開発環境は VisualC++である。この推論エンジンの特徴は、まず実際の制御を想定しているということで、処理速度を最優先する為グラフィック表示、シミュレーション等一切省いており、ファジィ推論のみに機能をしぼっている。構成は、ユーザーがファジィ集合のデータをルール数、前件部データ、後件部データ、という風にコンピューターに入力し、それを基に推論するといういたってシンプルなものになっている。（図６）に実行画面を示す。

（実行画面）



（図６）

この実行画面では、VisualBasic を用いた推論エンジンの実行画面と同じく例として５つのルールが与えられている。（図６）から分かるように画面右上部には前件部・後件部データを表示し、画面右下部には先ほどと同じような計算結果が表示される。VisualBasic を用いた推論エンジンとは表示方法に若干の違いがある。こちらは実際の制御を想定している為、後で距離、メンバーシップ関数、重心などのデータを参照できるようにリストとして表示する方法を採用している。しかし、もしユーザーが個々のデータとして表示したい場合でも、簡単に変更す
る事が可能である。

推論部分のプログラミングだが、VisualBasic,VisualC++ともに同じような構造となっている。前件部ファジィ集合のデータ格納用変数を配列として宣言し、同じく後件部ファジィ集合のデータ格納用変数も、配列として宣言している。配列にすることにより、計算時にこれらの変数に格納されたデータを簡単かつ確実にそれぞれ呼び出すことが可能となる。また、仮にルール数が変化した場合でも正確に対処することができる。推論部分のソースについての詳しい説明は、３節で述べる。

３－２　開発環境の違いの理由

今回、２つの推論エンジンの開発にそれぞれ VisualBasic、VisualC++、と 2 種類のプログラミングソフトウェアを使い分けた。これらのプログラミングソフトウェアは、Windows 上でのプログラム開発言語として Microsoft 社から提供されている。

VisualBasic と VisualC++、両者は言語仕様、操作性、開発効率、実行性能などにおいて一長一短があり、一概にどちらが良いとはいえず、一般的には目的に応じてこれらを使い分けているのが現状である。今回、これらの開発環境を使い分けたそれぞれの理由について述べたいと思う。

まず VisualBasic を使った推論エンジンの開発についてだが、こちらは「初心者にも簡単に使える推論エンジン」、「視覚的、感覚的にファジィを扱える」、「シミュレーションも可能」ということで、前節で述べたようにグラフィック表示など視覚的なものが重要視される。今回使用した VisualBasic は、グラフィック表示など、視覚的な部分を簡単かつ強力にプログラミングすることができ、さらにプログラムの開発効率は非常に高い。VisualC++の場合、画面の初期化、描画を繰り返すにあたって、ポインタやデバイスコンテキストの取得など、画面にグラフィック操作をするのに様々な儀式が必要となる。例えばピクチャーボックス（グラフィックが表示されるボックス）を初期化する場合、VisualBasic と VisualC++では例えば以下のような記述をする。

VisualBasic ：　Picture1.Cls

```
VisualC++  ：  CWnd* h = GetDlgItem( IDC_PICT1 ) ;
               CDC*   pDC = h->GetDC( ) ;
               Crect  r ;
               h->GetClient( &r ) ;
               pDC->FillSolidRect( r, RGB( 255, 255, 255 ) ) ;
```

以上のプログラムは、それぞれ同じ動作をさせるにも関わらず、それぞれのプログラムを見比べると分かるように、VisualC++ではピクチャーボックスを初期化するだけでVisualBasic よりもかなり長い記述が必要になる。VisualC++のプログラムソースを上から順に説明すると、まず、コード中からコントロール（ここではピクチャーボックス）を参照する為に、GetDlgItem 関数によって指定したコントロールへのポインタを取得している。次に、ピクチャーボックスに対してグラフィック描画を行うにはピクチャーボックスに対するデバイスコンテキストを取得する必要があり、GetDC 関数を使用してデバイスコンテキストを取得している。CDC はデバイスコンテキストクラスというもので、ｐDC は CDC のクラスへのポインタとして宣言されている。そして、画面をクリアするに当たってその範囲を得る必要があるわけだが、そこで、Crect というクラスを使用する。これは、短形の座標を扱うクラスである。Crect 型としてｒを宣言し、そして GetClientRect 関数を実行することで、ピクチャーボックスの領域サイズをｒに取得する。最後に FillSolidRect 関数を使用し、ｒの領域を塗りつぶしている。このような処理が、VisualC++のプログラムソースでは行われている。開発効率の面以外にも、両者のプログラムソースを比較すると分かるように VisualBasic の方が初心者がプログラムソースを参照する時に流れをつかみやすいと思われる。

またシミュレーション機能を設けるということで、時間の概念も必要になる。VisualBasic には、タイマーコントロールが標準で搭載されており、これを使うことで様々な時間域でのシミュレーションが可能となり、タイマーのオンオフ、リセットなど、素早く確実に実行することができる。

次に VisualC++を使った推論エンジン開発についてだが、こちらは「実際のファジィ制御を想定」ということで、処理速度が重要視される。VisualC++は様々な開発環境の中でも処理速度の速いアプリケーションを開発することができ、実行性能は VisualBasic よりも遥かに高いプログラミングソフトウェアである。そこで、この開発環境を選択した。また実際の制御でも使用されていることをふまえた上で、十分応用が可能と言える。

これらの理由から、互いの推論エンジンを開発するのにそれぞれの特徴に合った開発環境を採用した。

３－３　推論エンジン部分のプログラミング手法

両プログラム共通である推論エンジン部分の手法について説明する。内容としては、二章の推論手順に基づく。距離型ファジィ推論法の推論の流れとして、距離計算、メンバーシップ関数の計算があげられるが、本研究で開発した推論エンジンのプログラミングソースではこれらをサブルーチン化することにより、作業を格段にしやすくしている。ここではVisualC++のプログラムソースを例として使用する。

まず、距離計算のサブルーチンを以下に示す。

```
//距離計算サブルーチン
void KyoriKeisann()
{
	int		i,j;
	double	r,s,t;

	t=0;

	for (i=0; i<2; i++)
	{
		r = 0;

		s = 1;

		for (j=i; j<i+2; j++)
		{
			r += (A[j]-J[j])*(A[j]-J[j]);
			s *= (A[j]-J[j]);
		}
	t += sqrt(r+s);

	}
distance=(t+(1-AA)*sqrt(A[2]-A[0])+(1-JJ)*sqrt(J[2]-J[0]))/sqrt(3);
 }
```

J[0～２]、JJ は事実集合のパラメータを表し、A[0～2]、AA は前件部集合のパラメータを格納する変数である。このサブルーチンにより、ファジィ集合の距離計算を行うことができる。

以下の（９）式が、三角型及びシングルトンの距離関数である。この距離関数を、距離計算サブルーチンの中で実行している。

$$d(A.B) = \frac{1}{\sqrt{3}} \sum_{i=1}^{2} \left[\sum_{j=i}^{i+1} (a_j - b_j)^2 + \prod_{j=i}^{i+1} (a_j - b_j) \right]^{1/2}$$

$$+ \frac{1}{\sqrt{3}} \left[(1-a)(a_3 - a_1)^{1/2} + (1-b)(b_3 - b_1)^{1/2} \right]$$

（９）

for 文を使うことでΣ及びΠを実現していることがプログラムから読み取れるであろう。最後に distance でまとめの距離計算を実行している。

次に、メンバーシップ関数計算サブルーチンを以下に示す。

```
//メンバーシップ関数計算サブルーチン
void Membership()
{
        int     i,j,k;
        double     r=0,s1=1,s2=1;


        for(i=0;i<RuleN;i++)
        {
                ju[i]=0;
        }

        for(i=0;i<RuleN;i++)                                    ①
```

```
{
        for(j=0;j<RuleN;j++)
        {
            if(j!=i)

            {
                s1=s1*DA[j];
            }
        }

        for(j=0;j<RuleN;j++)
        {
            for(k=0;k<RuleN;k++)
            {
                if(k!=j)
                {
                    s2=s2*DA[k];
                }
            }
            r+=s2;
            s2=1;
        }
        Kq[i]=s1/r;
        s1=1;

        r=0;
}
for(i=0;i<RuleN;i++)                                    ②
{
        r1 +=(Kq[i]*KoukenB[ i * 4 ]);
        r2 +=(Kq[i]*KoukenB[ i * 4 + 1 ]);
        r3 +=(Kq[i]*KoukenB[ i * 4 + 2 ]);
        r4 +=(Kq[i]*( KoukenB[[i*4 + 1]-KoukenB[i*4] ]);
        r5 +=(Kq[i]*( KoukenB[i*4+1]-KoukenB[i*4+2]));
}
```

```
        for(i=0;i<RuleN;i++)                                    ③
        {
                ju[i]+=KoukenB[i*4];
                ju[i]+=KoukenB[i*4+1];
                ju[i]+=KoukenB[i*4+2];
        }
        for(i=0;i<RuleN;i++)
        {
                Juushin +=(Kq[i]*ju[i]);

        }
        Juushin=Juushin/3;
}
```

以上のソースにより、メンバーシップ関数及びメンバーシップ関数の重心が求まる。

このサブルーチンでは、①の for 文でまず距離計算のサブルーチンで計算された距離の値を基に、定数 Kq を計算している。

さらに②の for 文でこの定数 Kq を用い、推論結果であるメンバーシップ関数の計算に使用するパラメータを計算する。上記のプログラムで言うと、ｒ１、ｒ２などがそれに当たる。

これらのパラメータを基に推論結果μB(y)のメンバーシップ関数を組み立てる。パラメータの組み立てはサブルーチンの外で表示する直前に実行している。

最後に③以降のプログラムで推論結果のファジィ集合から重心を計算している。以上が、メンバーシップ関数計算サブルーチンで行っている内容及び手順である。実際のプログラムの流れでは、入力された事実と、配列に格納されている各ファジィ集合データを使用し、距離計算及びメンバーシップ関数計算サブルーチンで計算し、計算結果を出力する、という風になっている。

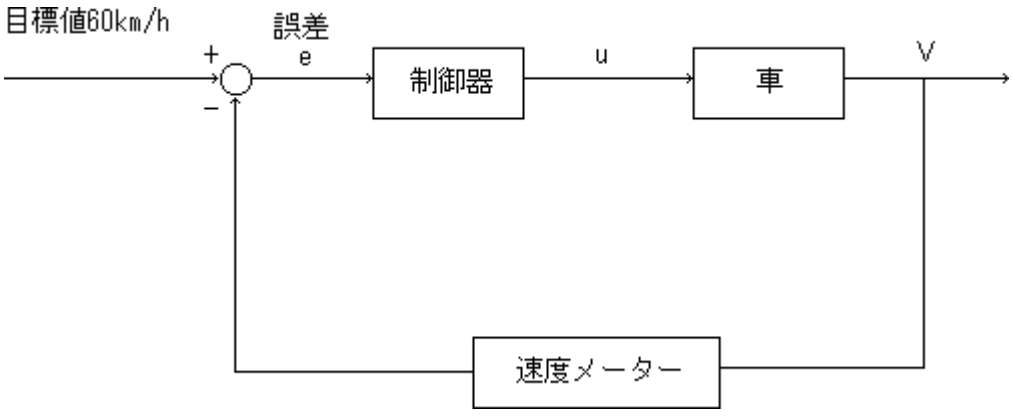
第四章　シミュレーション（実験）

４－１　シミュレーション内容

これまで、距離型ファジィ推論法を用いてファジィ推論が行えることは分かった。しかし、実際にはこのファジィ推論を用いて、ファジィ制御を行える必要がある。

この章では、距離型ファジィ推論法が、求められた推論に対して精密かつ確実な推論結果を返し、実際の制御で使用できるという事を示す為に、距離型ファジィ推論法に基づいたファジィ制御シミュレーションを、VisualBasic で開発した推論エンジンのシミュレーション機能を使い行う。シミュレーションでは運転知識から抽出したルールにより車の速度を制御しており、車の動力学モデルを必要としていない。

制御対象は車の速度であり、目標値時速６０ｋｍに速度を維持するシミュレーションとする。そこで車の速度維持制御をするにあたって、（図７）のような制御系のブロック図を作成した。



（図７）

制御の流れとしては、まず現在の速度と目標値 60ｋm/h との誤差を計算し、その値を誤差 e として制御器へ渡す。この制御器が、ファジィ推論エンジンに当たる部分である。次に、ファジィ推論エンジンである制御器から推論結果として計算されたアクセル開度の値 u を車に与える。ここでのアクセル開度はエンジンから車に与えられる力と仮定する。このアクセル開度が変化することにより車の速度が変化する。最後に出力された車の速度を速度メーターで感知し、ブロック図の始めに戻る。これらの動作を繰り返すことで、速度制御を実現する。

４－２　シミュレーション手法

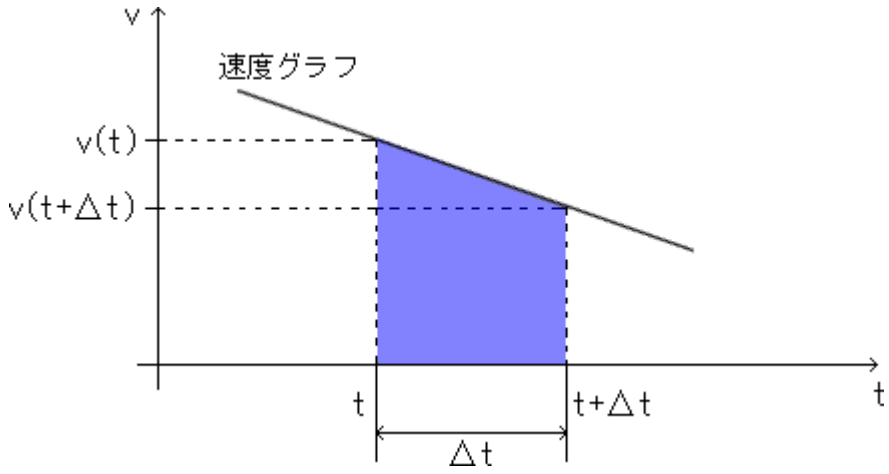
シミュレーションには、オイラー法という最も簡単な数値計算の方法を用いる。ここで、オイラー法について簡単に説明する。

時刻ｔの時の物体の位置、速度を考えた時、これらはそれぞれ時刻ｔの関数であり、y(t), v(t)として表される。さらに、加速度 a は常に一定であるとすると、時刻 t+Δt における物体の速度 v(t + Δt) は、

$$v(t + \Delta t) = v(t) + a \cdot \Delta t \qquad (10)$$

と表される。加速度 a というのは速度の増加率であるから、a・Δt は Δt という短い時間内での速度の増加量を表す。

さて、物体が動いた距離は速度を積分すると表せることから、（図８）の斜線部分の面積は、時刻ｔから t+Δt の間に物体が動いた距離を表していると言える。



（図８）

時刻 t+Δt における出発点からの距離は以下のように表せる。

$$\left(\begin{array}{c}\text{時刻 t＋△ t における}\\ \text{出発点からの距離}\end{array}\right) = \left(\begin{array}{c}\text{時刻 t における}\\ \text{出発点からの距離}\end{array}\right) + \text{（斜線部分の面積）}$$

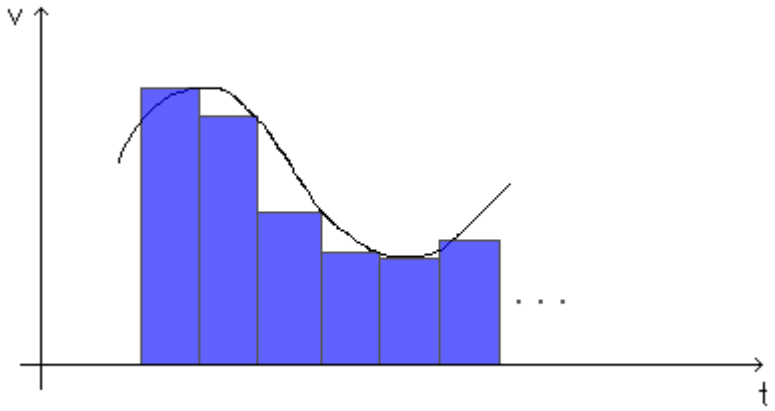
（11）

これを式で表すと、

$$y(t+\Delta t) = y(t) + (\text{斜線部分の面積})$$

（12）

となる。この斜線部分の面積は、本来は台形の面積を計算するのだが（図９）のように、



（図９）

（斜線部分の面積）　（棒グラフの面積）＝ $v(t+\Delta t)\cdot\Delta t$

（13）

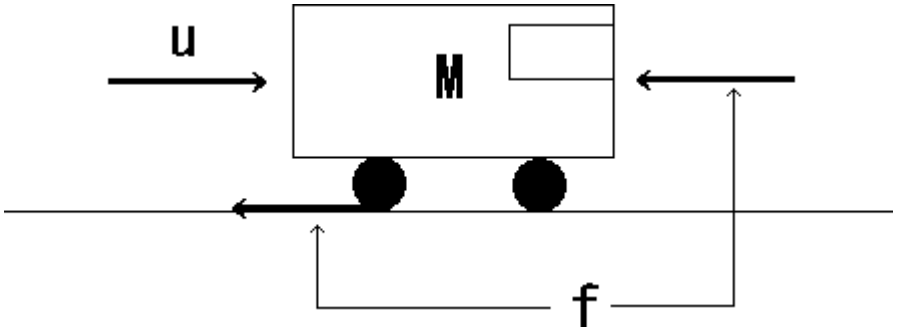
で近似をする。これがオイラー法である。

このオイラー法を用いて速度の式を組み立てると、次式のようになる。

$$\dot{x}(t+\Delta t) = \dot{x}(t) + \Delta t \cdot \ddot{x}(t)$$

（14）

この式に基づいてシミュレーションを行うが、この式だけではシミュレーションは実行できず、加速度の式が必要である。そこで、シミュレーションで想定する車には、（図 10）に示すように u , f の力がかかるものとする。



（図 10）

u はエンジンから車へのアクセル開度に伴う力であり、ｆは車に外部からかかる摩擦力及び空気抵抗である。車にかかる力の関係は、ニュートンの運動法則 F=mα より、

$$(u-f) = M\frac{d^2x}{dt^2}$$

（１５）

とおくことができる。さらに摩擦力及び空気抵抗は以下のように与える。

$$\text{摩擦力} = \mu M$$
$$\text{空気抵抗} = k\frac{dx}{dt}$$

（16）

これらの式を用いると、

$$u = M\frac{d^2x}{dt^2} + \mu M + k\frac{dx}{dt}$$

（17）

となり、車にかかる力の関係式ができる。さらにこの関係式を置き換えると（18）式のように加速度の式が導き出せる。

$$\ddot{x}(t) = \frac{u(t) - \mu M - k\dot{x}(t)}{M}$$

（18）

これで、速度の計算に必要な式は全て揃った。これらの式をオイラー法に適用し、シミュレーションを行う。

４－３　実験及び実験結果

VisualBasic で開発した推論エンジンのシミュレーション機能を用いて、車の速度を６０km/h に維持するシミュレーションを行った。今回、シミュレーション条件は以下のように設定した。

車の質量M	５００ｋｇ
摩擦係数μ	０．１
空気抵抗係数ｋ	０．１

この条件に基づき、シミュレーションを行う。

まず、推論ルールをおおよその予想として設定する。ルールを５つとし、目標速度と実際速度との速度誤差を推論の前件部、アクセルの開度を推論の後件部とする。

e=目標値ー現在の速度

ルール１　　IF (e が大) THEN (u を大)

ルール２　　IF (e が少し大) THEN (u を少し大)

ルール３　　IF (e が０) THEN (u をそのまま)

ルール４　　IF (e が少し小) THEN (u を少し小)

ルール５　　IF (e が小) THEN (u を小)

e：速度誤差
u：アクセル開度

これらのルールをファジィ集合で表す。

R１:　IF　A = A1(30, 60, 90, 1)　THEN　　B = B1(8, 9, 10, 1)

R2:　　IF　A = A2(0, 30, 60, 1)　THEN　　B = B2(5, 7, 9, 1)

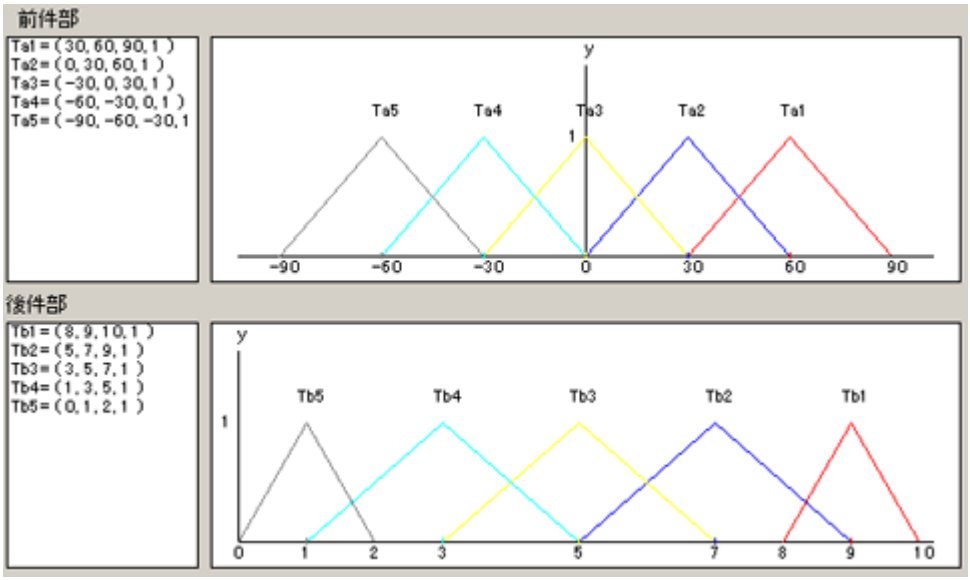
R3:　　IF　A = A3(-30, 0, 30, 1)　THEN　　B = B3(3, 5, 7, 1)

R4:　　IF　A = A4(-60, -30, 0, 1)　THEN　　B = B4(1, 3, 5, 1)

R5:　　IF　A = A5(-90, -60, -30, 1)　THEN　B = B5(0, 1, 2, 1)

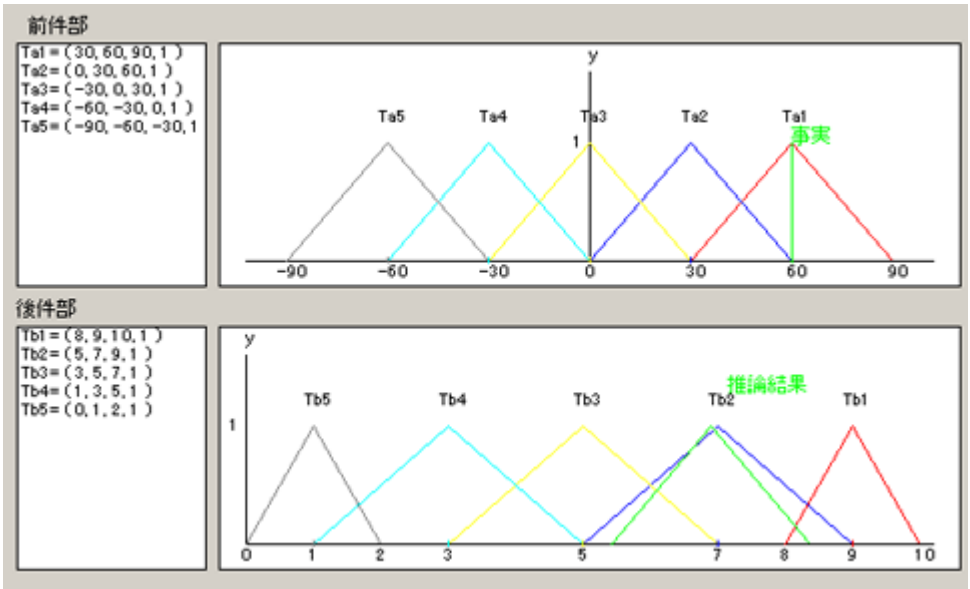
A:　事実
B:　推論結果

そしてファジィ集合で与えられたルールを入力する。ルール入力後の画面を（図１１）に示す。

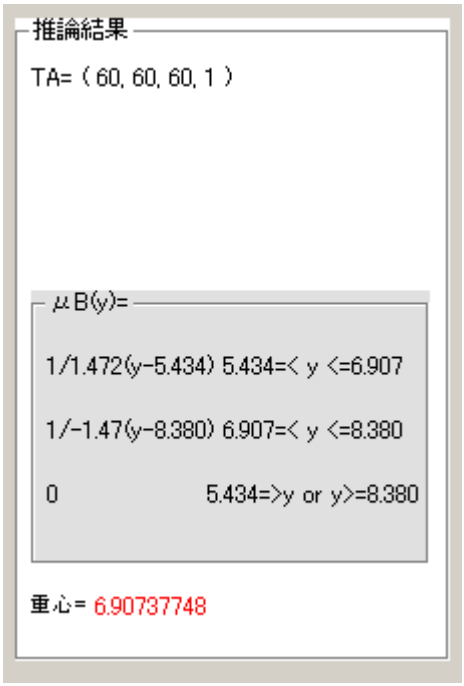


（図１１）

この状態で誤差 60（速度０km/h）を入力し、推論してみる。



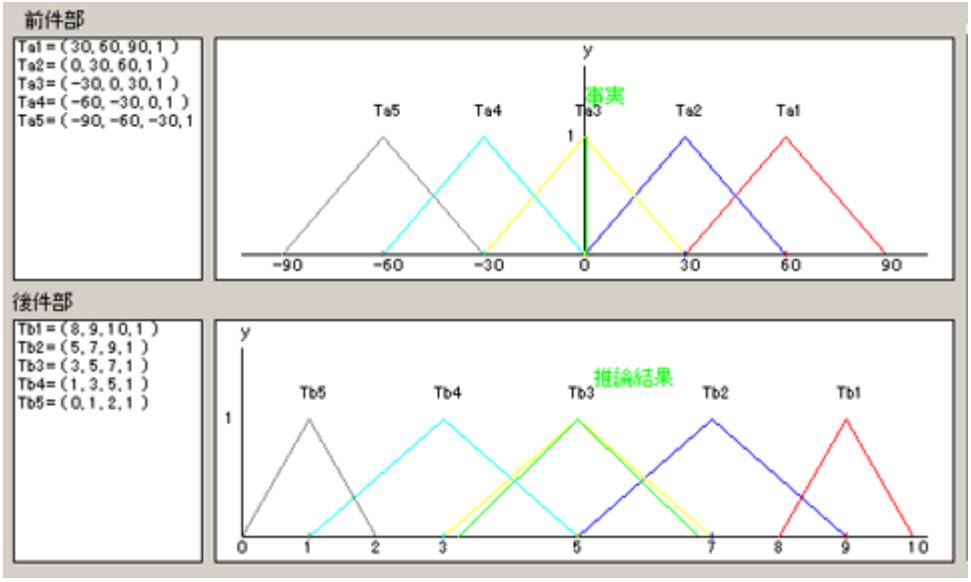
（図１２）



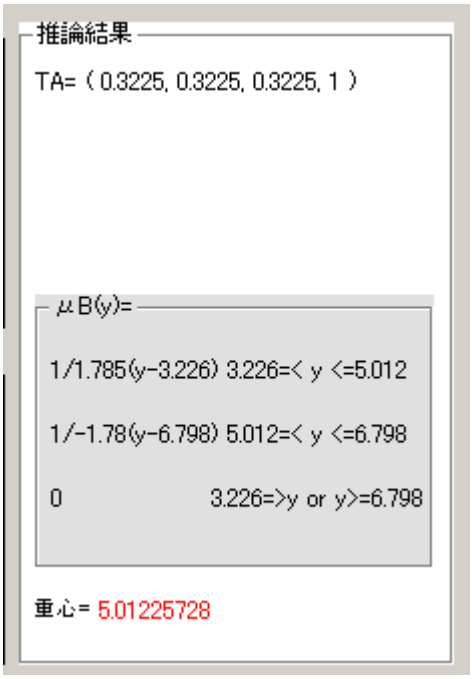
（図１３）

（図１３）から分かるように推論結果は 6.907 となり、これが推論から導き出されたアクセル開度となる。なお（図 １３）は、上から事実、事実と各ルールの距離及びメンバーシップ関数計算過程で使う Kq 、メンバーシップ関数μB (y) 、メンバーシップ関数の重心を表している。

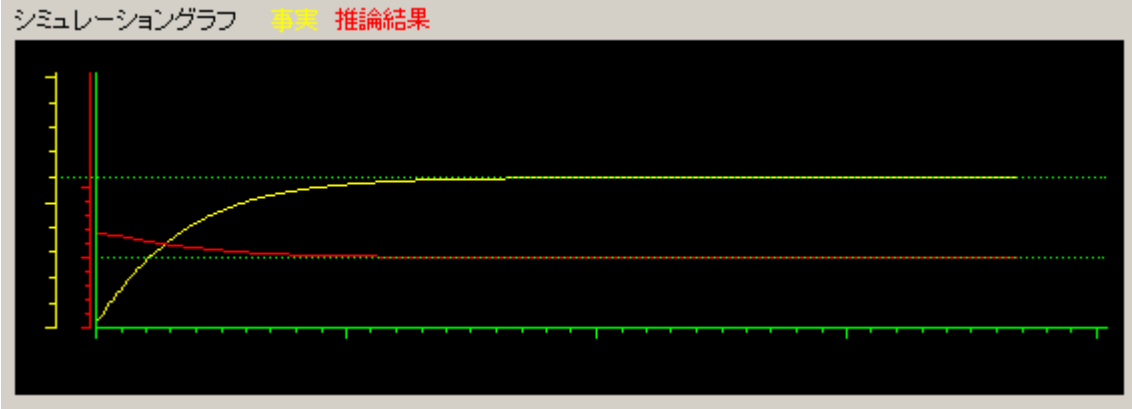
次に、この状態からシミュレーションをスタートさせる。シミュレーション結果のグラフを（図１４）に示す。


前件部
Ta1 = (30, 60, 90, 1)
Ta2 = (0, 30, 60, 1)
Ta3 = (-30, 0, 30, 1)
Ta4 = (-60, -30, 0, 1)
Ta5 = (-90, -60, -30, 1
y
事実
Ta5
Ta4
Ta3
Ta2
Ta1
1
-90
-60
-30
0
30
60
90
後件部
Tb1 = (8, 9, 10, 1)
Tb2 = (5, 7, 9, 1)
Tb3 = (3, 5, 7, 1)
Tb4 = (1, 3, 5, 1)
Tb5 = (0, 1, 2, 1)
y
推論結果
Tb5
Tb4
Tb3
Tb2
Tb1
1
0
1
2
3
5
7
8
9
10

（図１４）


推論結果
TA= (0.3225, 0.3225, 0.3225, 1)
μB(y)=
1/1.785(y-3.226) 3.226=< y <=5.012
1/-1.78(y-6.798) 5.012=< y <=6.798
0 3.226=>y or y>=6.798
重心= 5.01225728

（図１５）



（図１６）

（図１６）のｘ軸は時間ｔを表し、１目盛りが１秒となっている。ｙ軸は、赤軸の方は推論結果の値を表しており、アクセル開度の値となる。１目盛りは１となっている。黄軸は事実となっており、速度を表す。また、１目盛りは１０ｋｍ/h となっている。

結果として、前件部・後件部のグラフから、速度誤差は０に、アクセル開度は約５に収束していることが読み取れる。

さらに（図１６）を見ると、速度 0km/h から制御開始後約 15 秒で速度は目標値 60km/h に収束していることが読み取れる。

次に、前件部をファジィ集合、後件部をシングルトンとしてシミュレーションを行ってみる。ルールは先程のシミュレーションでのルールを基に、後件部のみシングルトンとして変更して以下のように与える。

R１: IF A = A1(30, 60, 90, 1) THEN B = 9

R2: IF A = A2(0, 30, 60, 1) THEN B = 7

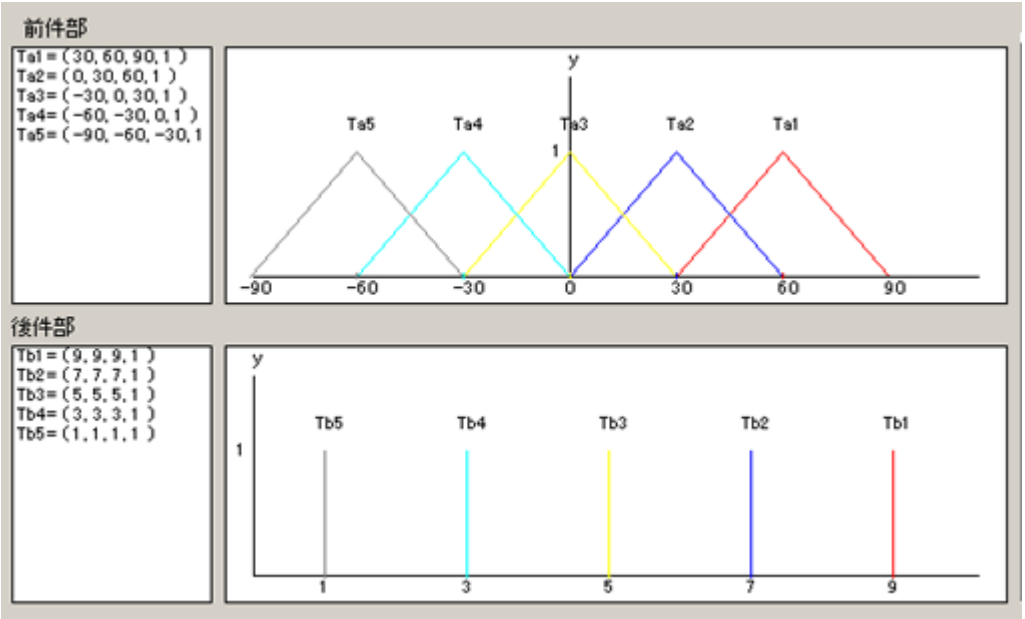
R3: IF A = A3(-30, 0, 30, 1) THEN B = 5

R4: IF A = A4(-60, -30, 0, 1) THEN B = 3

R5: IF A = A5(-90, -60, -30, 1) THEN B = 1

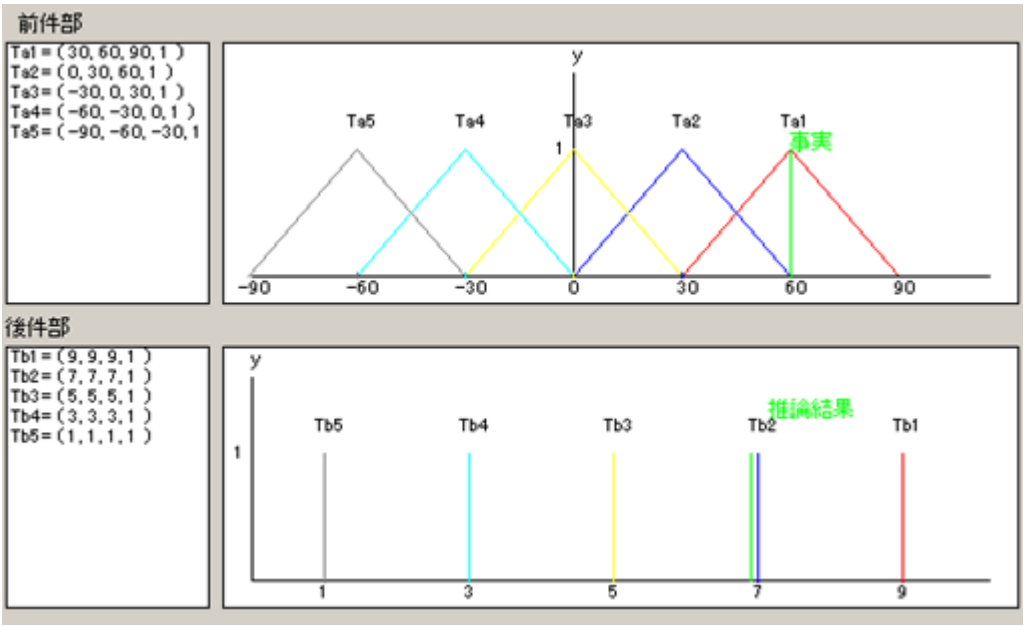
A: 事実
B: 推論結果

なお、シングルトンはメンバーシップ関数の幅を零に収束させた時の極限状態とする。これらを入力した様子を（図１７）に示す。

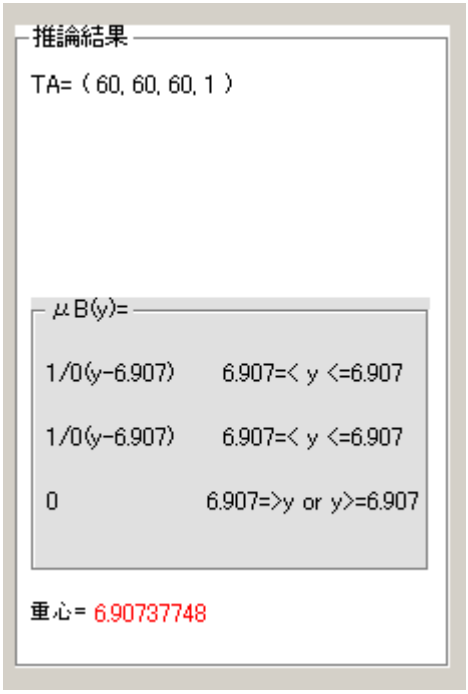


（図１７）

この状態で誤差 60（速度０km/h）を入力し、推論してみる。推論結果を（図１８）に示す。

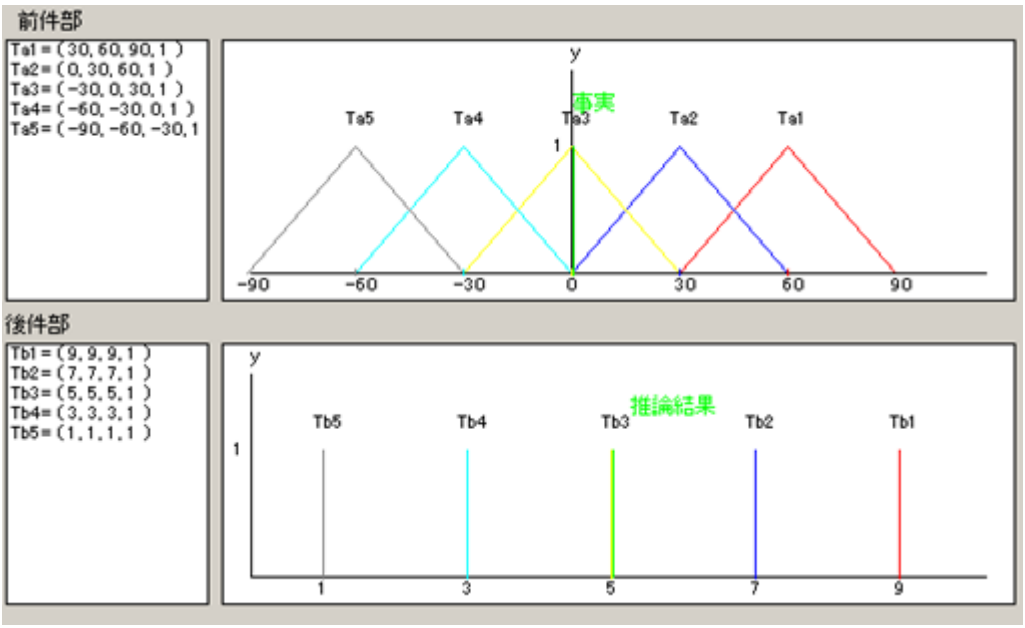


（図１８）

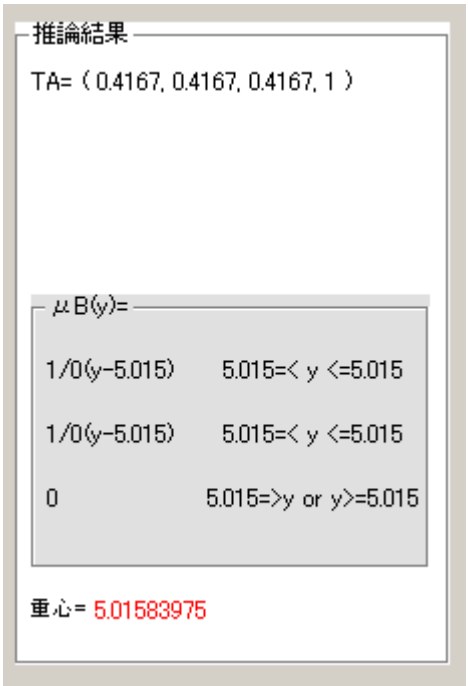


（図１９）

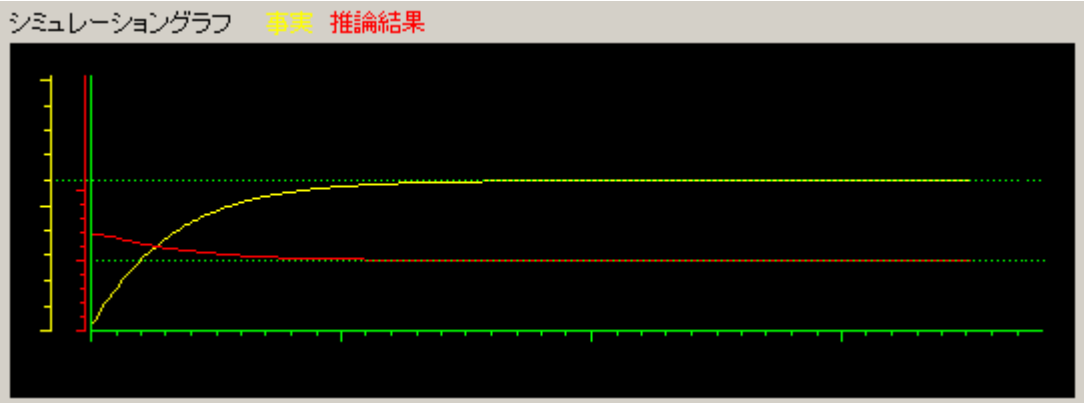
計算結果から、推論結果は 6.907 となっている。この状態からシミュレーションをスタートさせる。シミュレーション結果のグラフを（図 20）に示す。



（図 20）



（図 2 1）



（図 22）

結果として、前回と同じく前件部・後件部のグラフから、速度誤差は０に、アクセル開度は約５に収束していることが読み取れる。

さらに（図 22）を見ると、速度 0km/h から制御開始後約 15 秒で速度は目標値 60km/h に収束していることが読み取れる。

さて、これまで２回のシミュレーションには前件部にファジィ集合を用いたわけだが、「前件部がシングルトンでも推論が可能」という距離型ファジイ推論法の特徴を使い、前件部をシングルトンでおいた状態でもシミュレーションを行ってみようと思う。なお、今回は後件部も実数値でおいてみる。そこで、以下のようなルールを設定した。

R１: IF A = 60 THEN B = ９

R2: IF A = 30 THEN B = ７

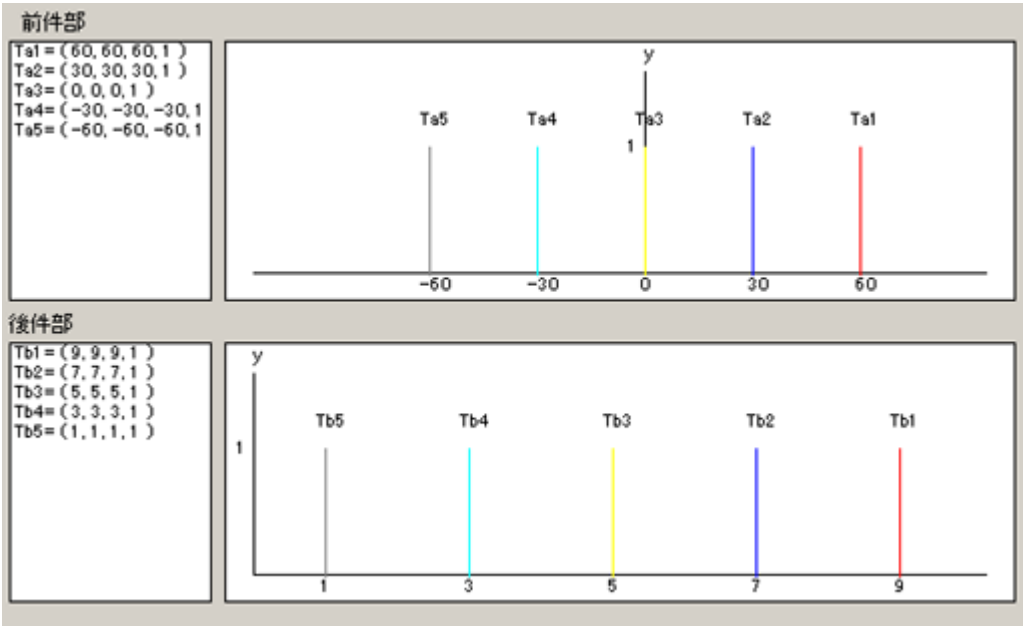
R3: IF A = 0 THEN B = 5

R4: IF A = -30 THEN B = 3

R5: IF A = -60 THEN B = １

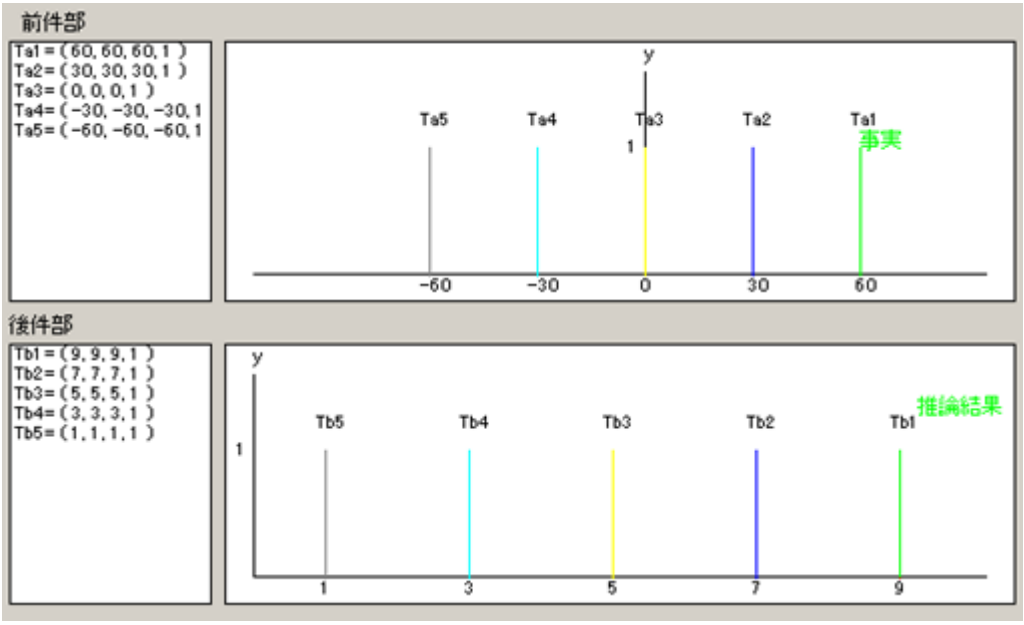
A: 事実
B: 推論結果

なお、シングルトンはメンバーシップ関数の幅を零に収束させた時の極限状態とする。これらのルールを入力した画面を（図 23）に示す。

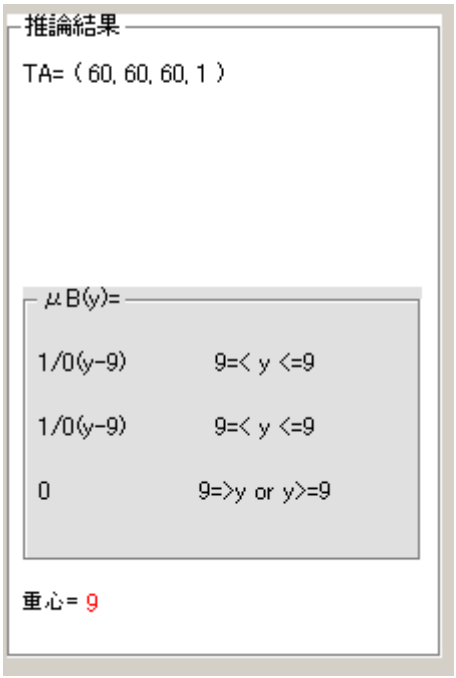


（図 23）

この状態で、前回のシミュレーションと同じく速度誤差 60（速度０km/h）を入力し、推論してみる。推論結果を（図 24）に示す。



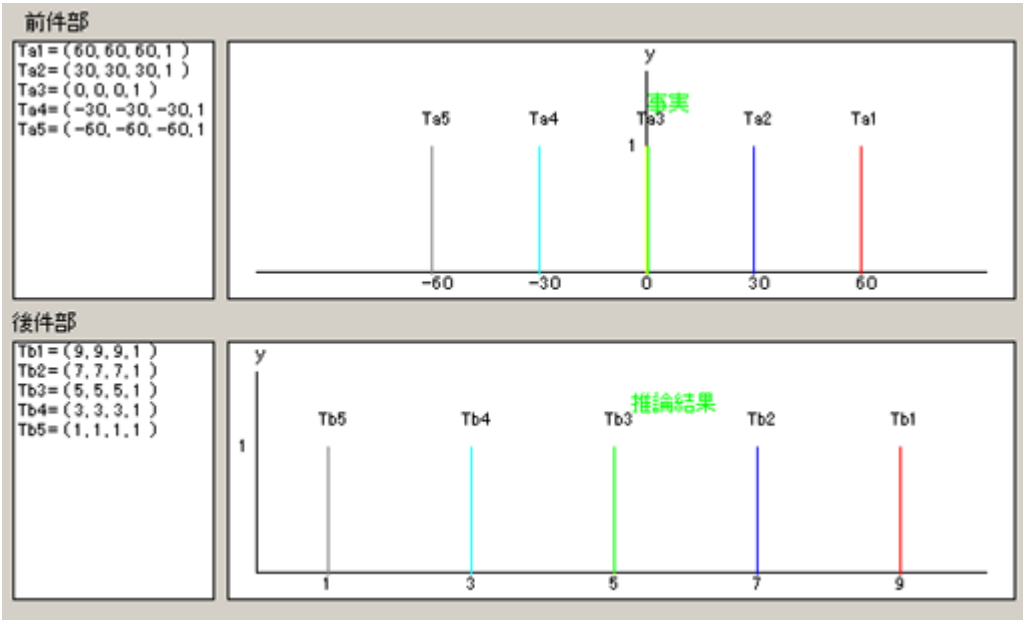
（図 24）



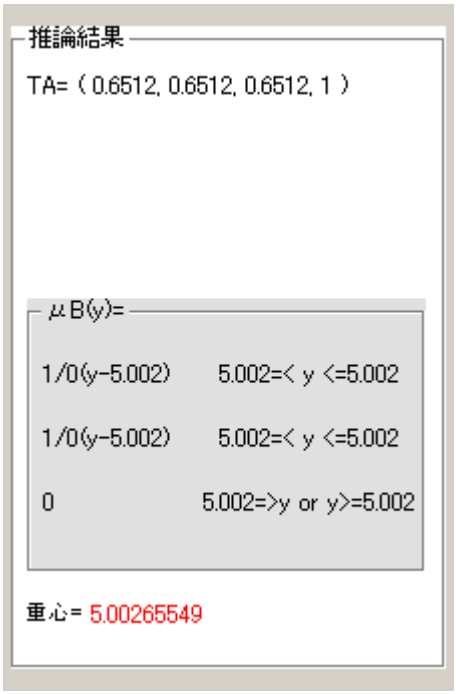
（図 25）

（図 23)、（図 25）を見ると、事実がルール１に当てはまる状態で、きちんとルールに沿った値（推論結果）を返していることが分かる。この事から、距離型ファジィ推論法が正確に分離規則を満たしていることが分かる。また、前件部がシングルトンであっても正確な推論結果が導き出せるという特徴についても証明できた。

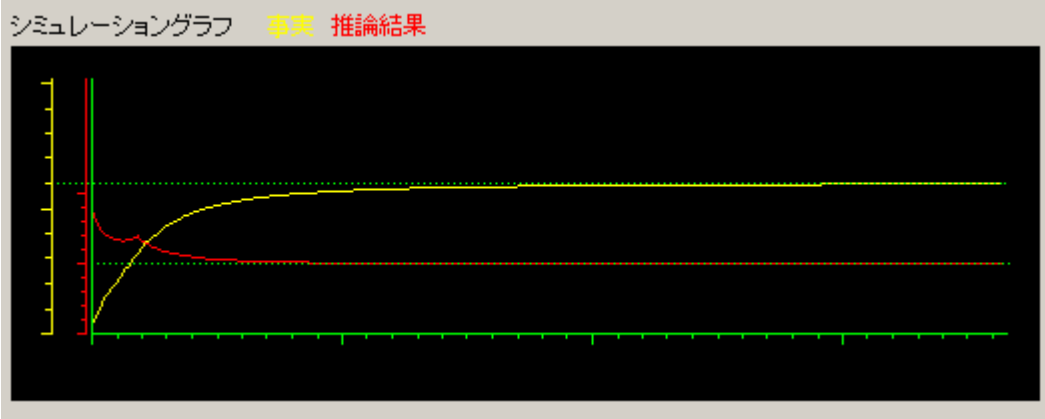
次に、この状態からシミュレーションを実行する。結果を（図 26）に示す。


前件部
Ta1 = (60, 60, 60, 1)
Ta2 = (30, 30, 30, 1)
Ta3 = (0, 0, 0, 1)
Ta4 = (-30, -30, -30, 1
Ta5 = (-60, -60, -60, 1
y
事実
Ta5
Ta4
Ta3
Ta2
Ta1
1
-60
-30
0
30
60
後件部
Tb1 = (9, 9, 9, 1)
Tb2 = (7, 7, 7, 1)
Tb3 = (5, 5, 5, 1)
Tb4 = (3, 3, 3, 1)
Tb5 = (1, 1, 1, 1)
y
推論結果
Tb5
Tb4
Tb3
Tb2
Tb1
1
1
3
5
7
9

（図 26）


推論結果
TA= (0.6512, 0.6512, 0.6512, 1)
μB(y)=
1/0(y-5.002) 5.002=< y <=5.002
1/0(y-5.002) 5.002=< y <=5.002
0 5.002=>y or y>=5.002
重心 = 5.00265549

（図 27）



（図 28）

最後に、前件部をシングルトン、後件部を三角型ファジィ集合でおいた状態でシミュレーションを行う。ルールは以下のように設定した。

R１: IF A = 60 THEN B =B1(8, 9, 10, 1)

R2: IF A = 30 THEN B = B2(5, 7, 9, 1)

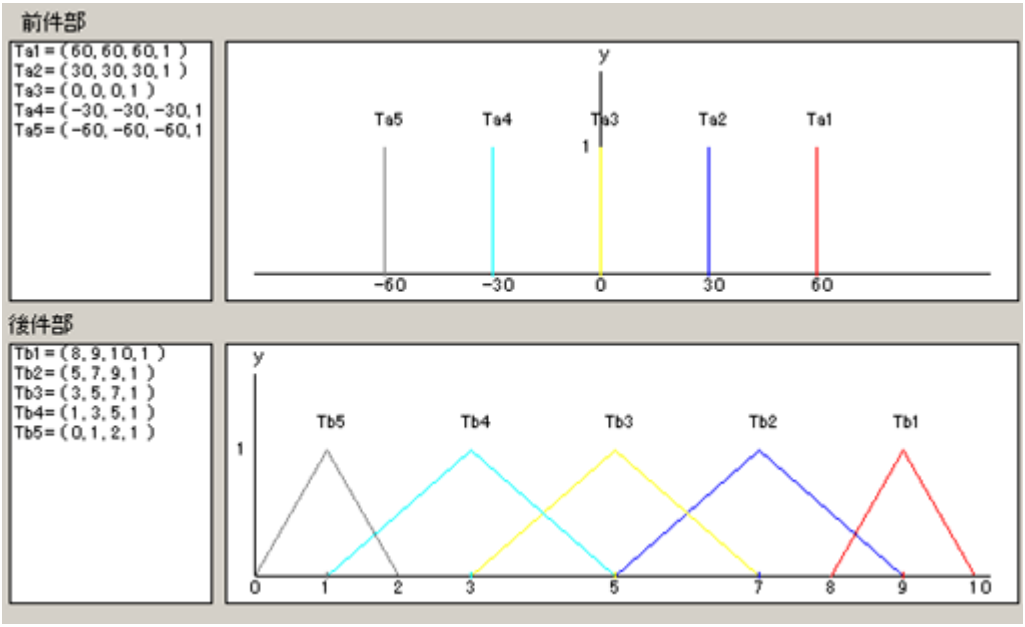
R3: IF A = 0 THEN B = B3(3, 5, 7, 1)

R4: IF A = -30 THEN B = B4(1, 3, 5, 1)

R5: IF A = -60 THEN B = B5(0, 1, 2, 1)

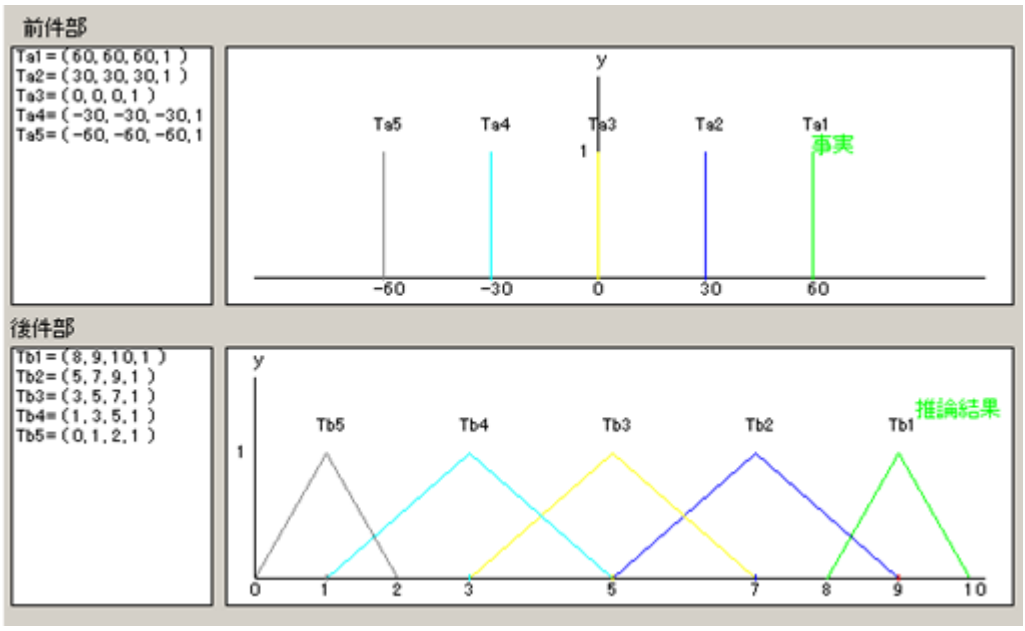
A: 事実
B: 推論結果

なお、シングルトンはメンバーシップ関数の幅を零に収束させた時の極限状態とする。これらのルールを入力した画面を（図 29）に示す。

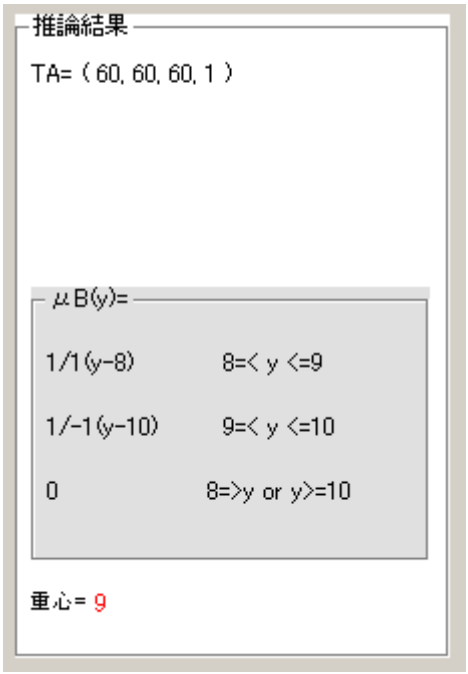


（図 29）

この状態で、前回のシミュレーションと同じく速度誤差 60（速度０km/h）を入力し、推論してみる。推論結果を（図 30）に示す。



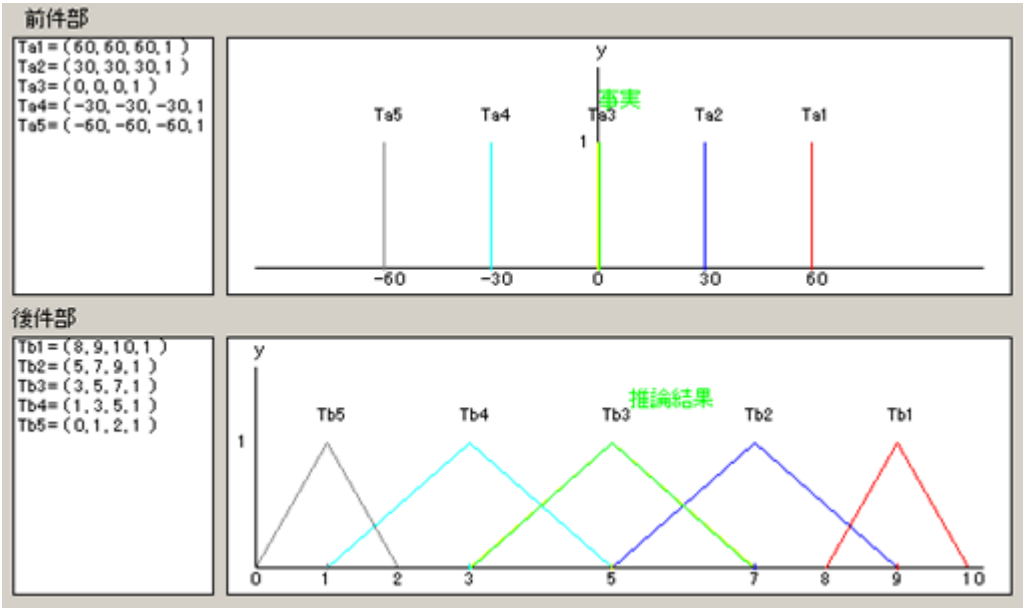
（図 30）



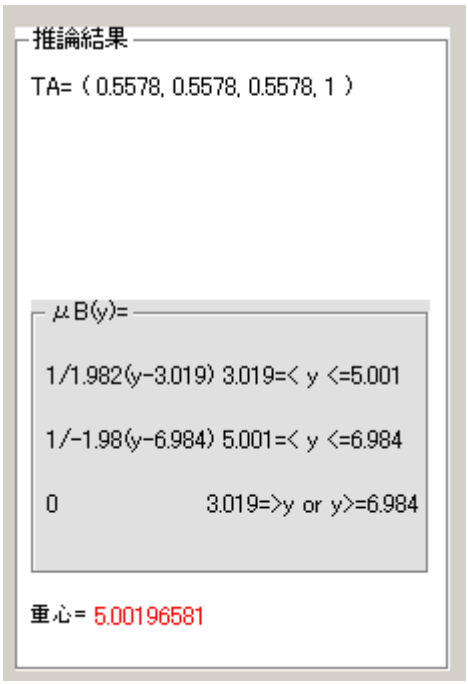
（図 31）

（図 30）を見ると、前回の前件部をシングルトンでおいた状態でのシミュレーションと同じような結果になったことがわかる。

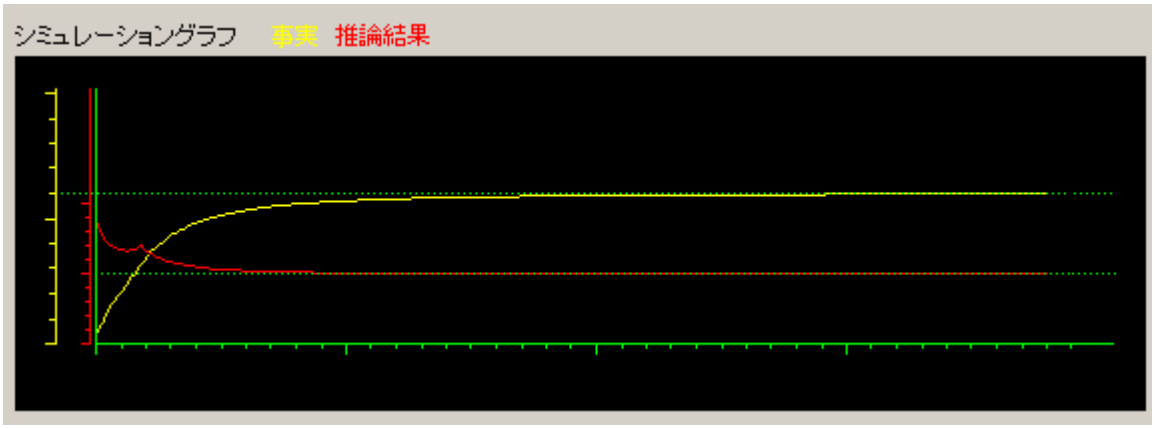
次に、この状態からシミュレーションを実行する。結果を（図 32）に示す。



（図 32）



（図 33）



（図 34）

（図 33）、（図 34）などから、正確な推論・制御が実行できていることが分かる。前回のシミュレーションと比較してみると、グラフに多少の違いが見られるが、同じく正確に制御が実行できている。

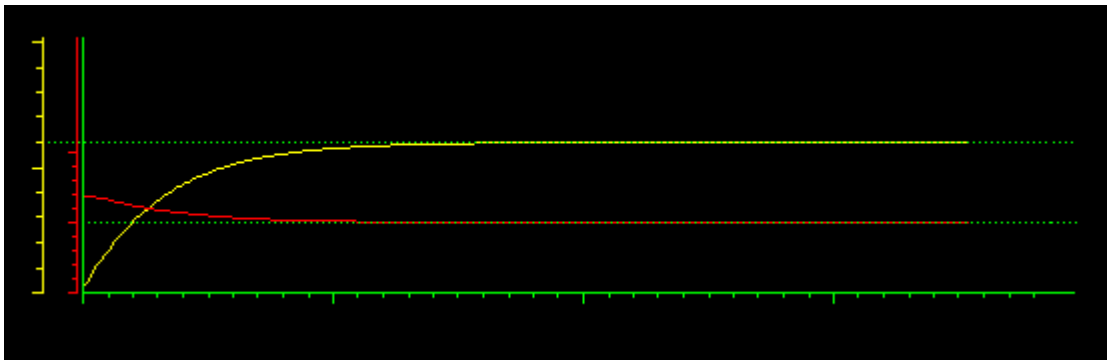
４－４　考察

実験結果から、距離型ファジィ推論法が実際の制御に使えることは証明できた。今回シミュレーションは以下のように４種類実行した。

1、　前件部と後件部を共に三角型ファジィ集合でおいた状態
2、　前件部を三角型ファジィ集合、後件部をシングルトンで置いた状態
3、　前件部と後件部を共にシングルトンでおいた状態
4、　前件部をシングルトン、後件部を三角型ファジィ集合でおいた状態

これらのシミュレーション結果のグラフを順に並べて以下に表示する。

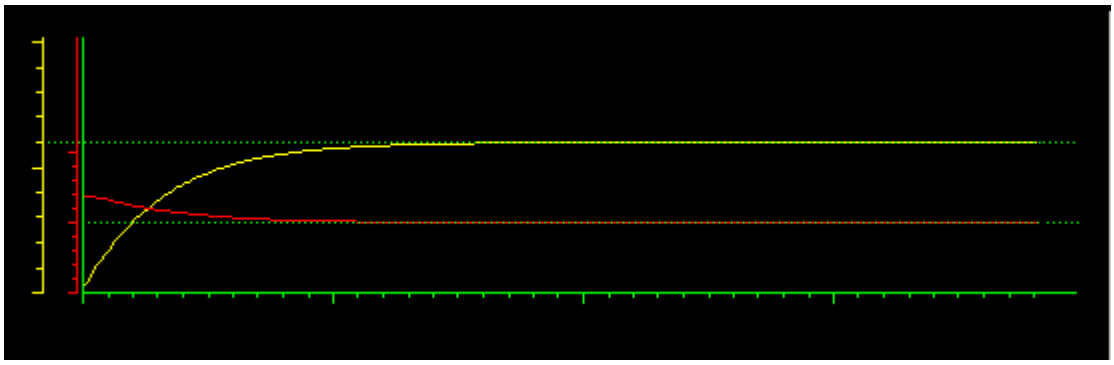
前件部：三角型ファジィ集合　　後件部：三角型ファジィ集合



黄：事実　　赤：推論結果

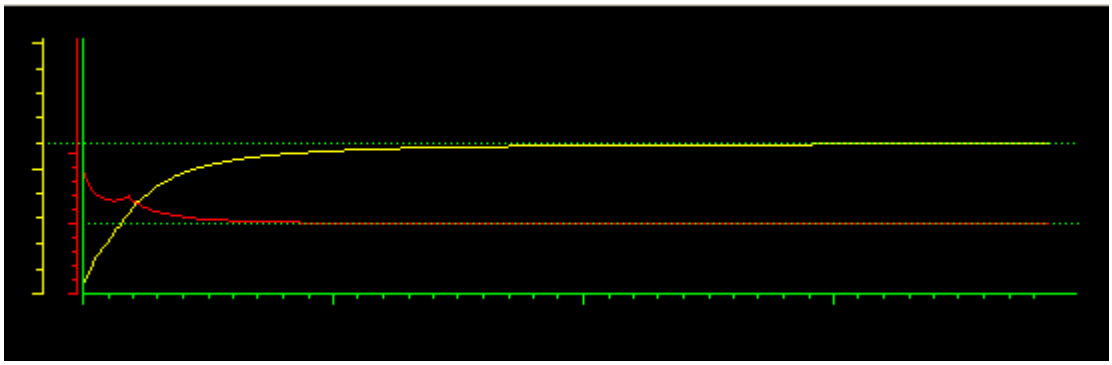
（図 35）

前件部：三角型ファジィ集合　　後件部：シングルトン



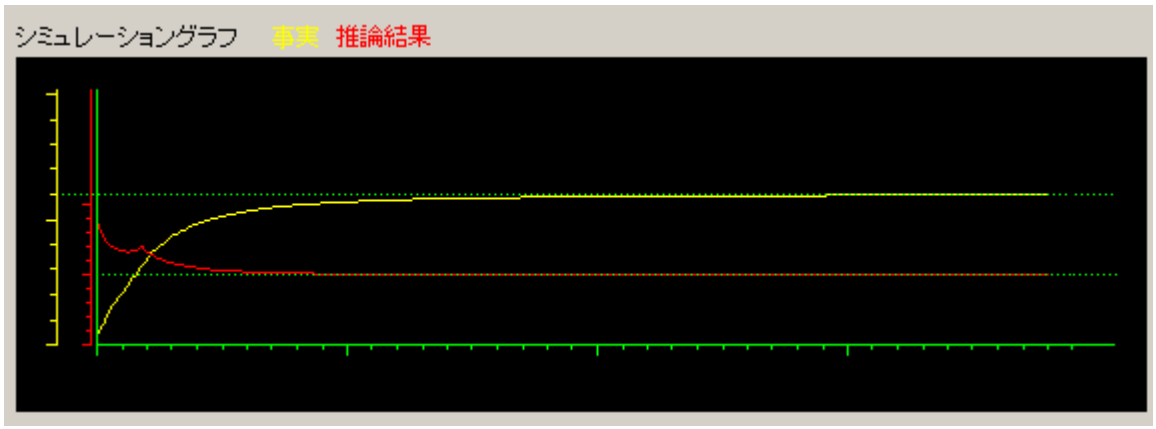
（図 36）

前件部：シングルトン　　後件部：シングルトン



（図 37）

前件部：シングルトン　　後件部：三角型ファジィ集合



（図 38）

それぞれのグラフから、前件部、後件部をファジィ集合、もしくはシングルトンのどちらでおいた場合でもきちんと正確な制御が行われていることは分かる。

しかし、それぞれのグラフを見比べると、まず前件部が三角型ファジィ集合で与えられた（図 35）、（図 36）のシミュレーション結果グラフでは、推論結果は滑らかな曲線を描いているのに対して、前件部をシングルトンでおいた（図 37）、（図 38）のグラフでは、推論結果は下降、上昇、下降というふうに、凸の部分ができている。

このシミュレーションでは事実（速度誤差）をファジィ集合ではなく、実数で受け取り推論している。本研究に使われている距離型ファジィ推論法は、ファジィ集合間の距離を基に推論を行っているわけだが、ファジィ集合間の距離を測る場合、上の２つのシミュレーションに見られるように、前件部が三角型ファジィ集合で、その前件部に事実が実数（シングルトン）として入った場合、この三角型ファジィ集合とシングルトンがどのように重

なったとしても、距離は０にはならない。これは二章三節を見てもらうとわかる。距離が０にならないとどうなるかというと、例えば以下のようなルールがある場合、

IF　X が A なら　THEN　Y を B にせよ

この X と A の距離が０になった場合、Y は B になる訳だが、距離が０にならなければ Y は B にはならず、距離が０に近ければ近いほど B に近いものになる。これは距離型ファジィ推論法の特性で漸近特性という。

このようにファジィ集合間の距離の関係で、３つのシミュレーション結果のグラフは滑らかなものや凸のあるもの、というふうに違ったものになった。推論結果の動きは、実際に推論エンジンに各シミュレーションで行ったルールを入力し、事実をドラッグして推論結果の動きを見てみるとわかる。

実際の制御では、今回のシミュレーションのように事実を実数として取得する場合が多い。従来のファジィ推論法ではこの実数を前件部にルールとして与えることは不可能であったが、この距離型ファジィ推論法を用いれば容易に実数を用いた推論が可能である。実験の結果からファジィ集合はどのように与えても推論､制御は可能であることは分かった。ファジィ集合またはシングルトンなど、ルールの与え方によって推論結果には微妙な変化がある訳だが、それに関してはその制御に見合ったルールの与え方をとるべきであると私は考える。

また、シミュレーションではそれぞれ目標速度に達するのに約 15 秒程度かかったわけだが、この到達時間は前件部・後件部のファジィ集合を移動または変化させることで同じく変化させることが可能である。

例として、以下のようにルールを変化させてみる。

R 1 :

IF x = A1(13.5, 43.5, 73.5, 1) THEN y = B1(7, 9, 10, 1)

R2:

IF x = A2(-14, 16, 46, 1) THEN y = B2(6, 8, 10, 1)

R3:

IF x = A3(-43, 0, 30, 1) THEN y = B3(4.9, 5.9, 6.9, 1)

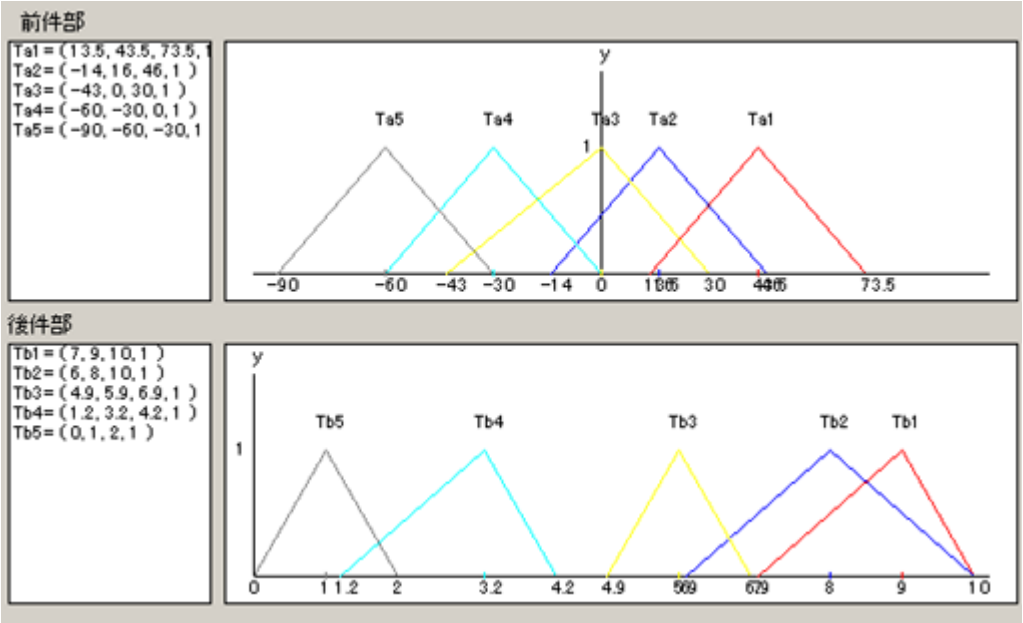
R4:

IF x = A4(-60, -30, 0, 1) THEN y = B4(1.2, 3.2, 4.2, 1)

R5:

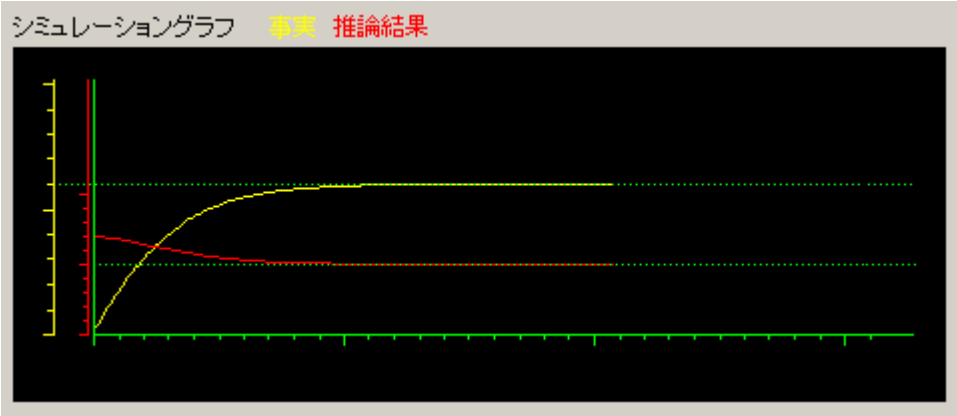
IF x = A5(-90, -60, -30, 1) THEN y = B5((0, 1, 2, 1)

これらのファジィ集合を図示したものを（図 39）に示す。



（図 39）

このルールを基に開始速度０km/h からシミュレーションをスタートしてみると、グラフは（図 40）のようになる。

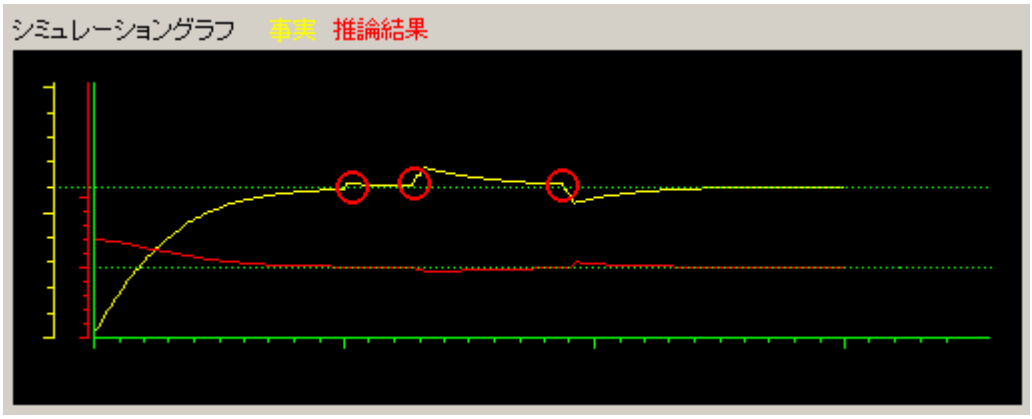


（図 40）

（図 40）から読み取れるように、約 10 秒で目標速度に達することがわかる。

このようにファジィルールを変更することで、目標値への到達速度を変化させることが可能となる。本研究で開発した推論エンジンに搭載されているシミュレーション機能は、シミュレーション実行中にもファジィ集合を視覚的にマウスで動かして推論結果の変化をリアルタイムで見ることができる為、簡単に適切なファジィルールが設定可能である。例えば、めちゃくちゃにルールを設定しておき、シミュレーションを実行しながら速度が目標値に収束するようにファジィ集合を移動させることができる。

また実際の制御では外乱により、値が急に変化することもある。そこで、事実を変化させる機能を使い、シミュレーション途中に速度を無理やり変化させてみた。その様子を（図 41）に示す。



（図 41）

（図 41）の赤丸が速度変化させたところである。図を見ると、速度が変化しても再び目標値に収束していることが分かる。このことから距離型ファジィ推論法に基づく制御は、実際の制御で起こり得る外乱にも対応できると言える。

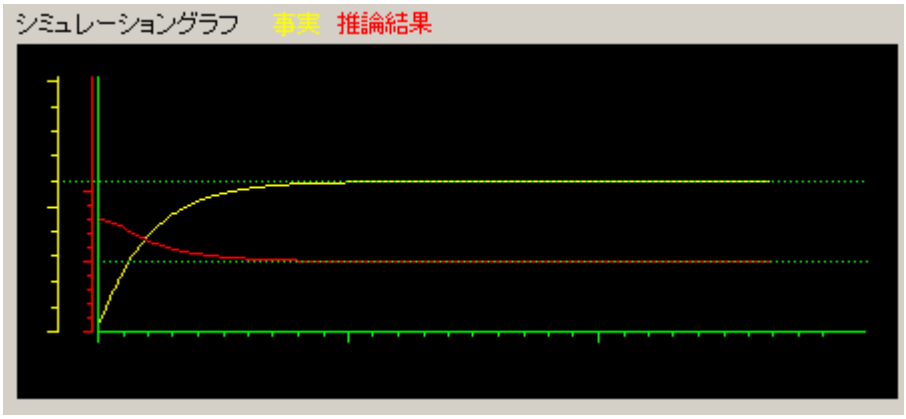
さて、今回の実験ではルール数を５つに設定してシミュレーションを実行したわけだが、ルール１とルール５のみを抜き出し、ルール数を２つにしてシミュレーションを実行してみるとどうなるだろうか。従来の推論法の例として、第二章で述べた max-min 重心法を考えた場合、各ルールはそれぞれ重なり合っていないと推論できないことが分かる。よって、従来の推論法ではこの場合推論は不可能である。対して本研究で用いた距離型ファジィ推論法の場合、各ファジィ集合と事実の距離を基に推論する為、推論が可能となる。以下にその様子を示す。まず、ルールを次のように設定する。

Ｒ１: IF A = A1(30, 60, 90, 1) THEN B = B1(8, 9, 10, 1)

Ｒ5: IF A = A5(-90, -60, -30, 1) THEN B = B5(0, 1, 2, 1)

A: 事実
B: 推論結果

このルールを基に誤差 60（開始速度 0km/h）からシミュレーションを実行してみる。シミュレーション結果グラフを（図 42）に示す。



（図 42）

（図 42）から、ルール数を５つに設定した時と同様に確実な制御ができていることがわかる。さらに目標値への到達時間、曲線の滑らかさなど、ルールを５つに設定した状態よりも性能が良いと言える。

　このように、距離型ファジィ推論法を用いると他の推論法に比べてルール数を大幅に少なくすることができ、ルール自体も非常にすっきりとまとめることができる。

第五章　結章

距離型ファジィ推論法に基づいて新しい推論エンジンを開発し、数値実験によりその有効性を示した。今後の改善点として次の 4 点があげられる。1) キーボードとマウスを併用した操作を拡張することで、推論エンジンの扱いやすさをさらに向上させることができる。2) π型・三角型・台形型などのファジィ集合が入り混じった状態など、あらゆるファジィ集合に対応できるように開発すれば、多種類の概念を定量化できる。3)シミュレーション条件など、ユーザーが自由にいつでも変更可能にすることでシミュレーションをよりリアルにすることができる。4)VisualBasic、VisualC++で開発した双方の推論エンジンを互いにリンクできれば、より効率的に作業が可能である。

参考文献

1） 王：距離型ファジィ推論法 Part1～18，FSS 論文集（1995～2001）

2） 村上　周太：ファジィ制御，日刊工業新聞社（1993）

3） 土谷　武士，深谷　健一：メカトロニクス入門，森北出版株式会社（1994）

4） 土井　滋貴，那須　靖弘，上田　悦子：Win32API 完璧マスタ，CQ 出版社（2001）

5） 河西　朝雄：VisualC++プログラミングブック，技術評論社（2000）

6） 谷尻　豊寿，谷尻　かおり：ステップアッププログラミング VisualBasic5，技術評論社（1998）

謝辞

本研究を行うにあたり、ファジィ推論についての適切な助言、ご指導を承りました王碩玉教授に深く感謝いたします。

また、研究を進めるにあたって様々な面で助けてくださった王研究室の同士に深く感謝いたします。

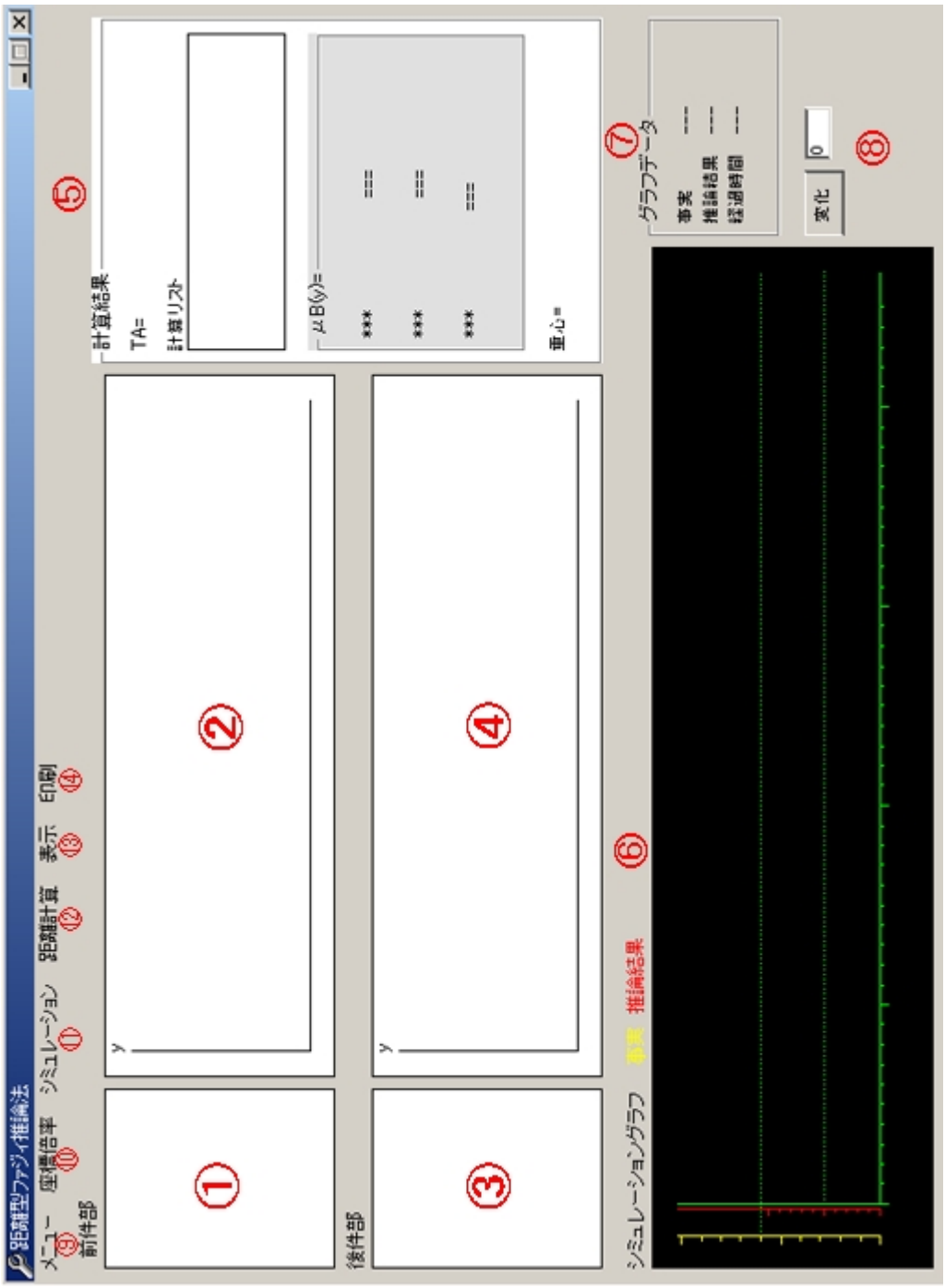
最後に、4 年間の大学生活を支えてくださった両親に心から感謝いたします。

付録

付録１　VisualBasic により開発した推論エンジンの使用方法

付録１－１　各コントロール使用方法

この節では、推論エンジンの各機能及び使用方法について説明する。まず、以下に画面の全体図及び各コントロールの使用方法を示す。なお、今後（図 1）に示す番号の順に説明を行う。


距離型ファジィ推論法
メニュー
座標倍率
シミュレーション
距離計算
表示
印刷
前件部
後件部
計算結果
TA=
計算リスト
μB(y)=
重心=
グラフデータ
事実
推論結果
経過時間
変化
シミュレーショングラフ
事実
推論結果

（図 1）

① 前件部データ表示ボックス

ルール順に前件部データが表示される。例として、Ta1=(0,1,2,1)の場合 Ta1 の１はルール番号を表し、右辺はファジィ集合のパラメータを表す。

② 前件部ファジィ集合表示ボックス

前件部のファジィ集合をルールごとに色分けして図として表示する。ここではマウスを使うことでさまざまなファジィ集合の操作が可能である。

③ 後件部データ表示ボック

ルール順に後件部データが表示される。主な内容は①と同じである。

④ 後件部ファジィ集合表示ボックス

後件部のファジィ集合をルールごとに色分けして図として表示する。②と同じくマウスによる様々な操作が可能である。

⑤ 計算結果表示ウィンドウ

上から事実、事実と前件部ファジィ集合の距離、推論結果のメンバーシップ関数、推論結果のメンバーシップ関数の重心、という順で表示する。

⑥ シミュレーショングラフ表示ウィンドウ

ｘ軸に時間ｔ、ｙ軸に黄色の軸として事実、赤色の軸として推論結果を表す。シミュレーションを実行することで、一定時間ごとにその時々の値（経過時間、事実、推論結果）をもとにグラフ化し、表示する。

⑦ グラフデータ表示ウィンドウ

シミュレーショングラフのデータが一定時間で更新されるごとに、その時のデータを表示する。

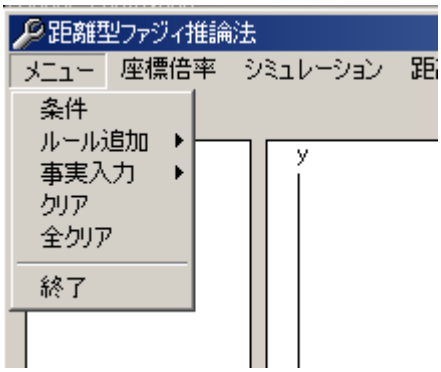
⑧　事実変更ボタン

シミュレーション実行中に使用する。テキストボックスに現在の事実をどれだけ増加・減少させたいかを入力し、変化ボタンを押すことで事実が変化する。これを使うことで急な外乱などの事実の変化をシミュレーションすることがで

きる。

⑨　メニュー

ルール数設定・ルール追加・事実入力・画面初期化など様々な用途に使う。以下にメニューをクリックした様子を（図 2）示す。



（図 2）

「条件」を押すと、ルール数の設定ウィンドウが表示される。ここで、ルール数を設定する。「ルール追加」を押すとさらに、「データ入力」・「自動セット前件部」・「自動セット後件部」というメニューが表示される。

データ入力：　ファジィについて知識のある人はこれを使い直接、前件部・後件部のファジィ集合のパラメータを入力する。

自動セット前件部：　ファジィについて知識の無い人はこれを使うことで、ファジィ集合の形を選択するだけでコンピューターが自動で前件部にファジィ集合のパラメータを入力してくれる。

自動セット後件部： 自動セット前件部と同じく、コンピューターが自動で後件部にファジィ集合のパラメータを入力してくれる。

「事実入力」をクリックすると、さらに「データ入力」、「自動入力」というメニューが表示される。

データ入力： ルール追加と同じく、ファジィについて知識のある人はこれを使い、事実のファジィ集合のパラメータを入力する。

自動入力： コンピューターが自動で事実を入力してくれる。あとはこれをマウスで移動などさせることにより推論を行う。

「クリア」を実行すると、現在のルールはそのままに事実ファジィ集合データや現在図として表示されている事実・推論結果のファジィ集合、シミュレーショングラフなどクリアしてルール入力直後の画面に初期化することができる。ルールはそのままに新たな事実で推論したい時や、シミュレーションを開始させたい時に使用する。

「全クリア」を実行すると「クリア」とは違い、ルール自体の全データを消去することができる。また、「クリア」と同じく前件部・後件部の各ファジィ集合表示ボックスの初期化、シミュレーショングラフの初期化も実行する。新たなルールで推論したい時や、シミュレーションを実行したい時に使用する。

⑩ 座標倍率メニュー

前件部・後件部ファジィ集合表示ボックスのｘ軸・ｙ軸座標倍率の変更に使う。変更中の画面を（図 3）に示す。

（図 3）

初期値は図のように x 軸 40、y 軸 70 となっている。ファジィ集合に使う各パラメータが大きい時にはファジィ集合表示ボックスが狭くなり、画面にファジィ集合が収まりきらなくなるという事態が発生するが、この数値を小さくすれば表示ボックスを広く使うことができ、全てのファジィ集合が表示可能となる。

⑪　シミュレーションメニュー

シミュレーションに関する設定はこのメニューで行う。「開始」、「インターバル」、「グラフ範囲」の 3 種類があるが、「インターバル」はシミュレーションの経過時間の変更を行う。一番上の 10ms が 0.01 秒間隔で最も速く、最後の１s が１秒間隔で最も遅い。「グラフ範囲」に関しては、現段階では１種類しか選択することができない。このメニューに関しては今後発展させていく予定である。

⑫　距離計算メニュー

このメニューをクリックすると、「三角型」、「台形型」、「π型」、「三角型･シングルトン」、「シングルトン」の５つの選択肢が表示される。これらは距離計算のみを実行するもので、「三角型」は２つの三角型ファジィ集合間の、「台形型」は２つの台形型ファジィ集合間の、「π型」は２つのπ型ファジィ集合間の、「三角型・シングルトン」は三角型ファジィ集合とシングルトンの、「シングルトン」は２つのシングルトン間の、それぞれ距離を計算することができる。あまり使う機能ではないが、距離型ファジィ推論の特徴である距離計算というものがどういうものなのかを、様々なファジィ集合を使って見ることができる。

⑬　表示

⑤の計算結果表示ウィンドウへの距離計算値表示の ON・OFF 切り替えを行う。

⑭　印刷

画面全体をプリントアウトすることができる。シミュレーション終了後、結果グラフなどをプリントアウトする時などに使用する。

付録１－２　マウスイベント

次に、マウスイベントについて説明する。ファジィ集合の移動や事実入力など、マウスでの操作を行うことで作業を大幅に快適化することができる。このマウスイベントは前に述べた（図１）の②、④のファジィ集合表示ボックスでの操作によるものが主になる。この表示ボックスでマウス操作を行うことで、以下のような処理が可能となる。

①　各ファジィ集合の移動

②　ファジィ集合の各パラメータの移動

③　ファジィ集合のパラメータ詳細変更

④　全ルールの前件部もしくは後件部のファジィ集合移動

⑤　事実直接入力

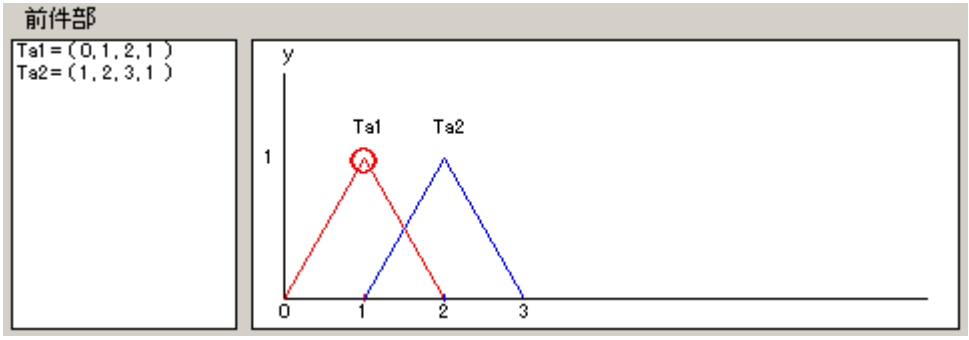
⑥　y 軸移動

これらの操作方法を以下で説明する。

①　各ファジィ集合の移動

これは 1 つのファジィ集合を全体的に移動することができる。なお、高さも変更が可能である。これは、1 つのファジィ集合だけを、各パラメータは変えずに並行移動したい

時や、高さのみを変更したい時などに使用する。マウス操作場所については、（図 4）に示す。

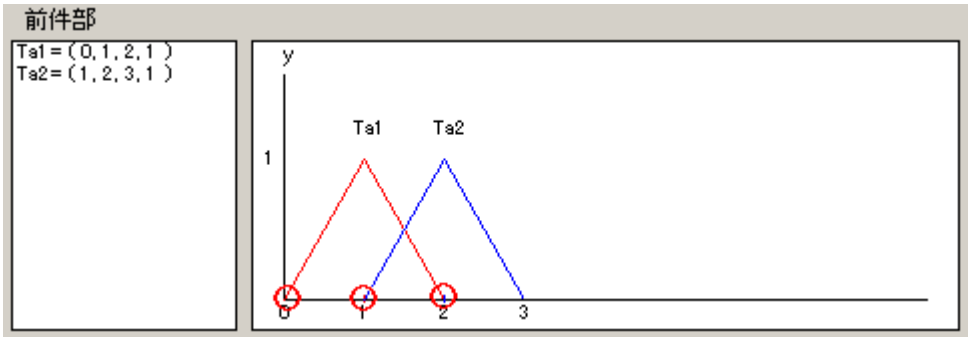


（図 4）

（図 4）の赤丸の部分（ファジィ集合のルールごとのラベル）をドラッグすることで、移動が可能になる。なお、移動量は 0.1 ごとになっている。

② ファジィ集合の各パラメータの移動

ファジィ集合の x 軸上にある各パラメータを変更することができる。これは、ファジィ集合自体の形を変更したい時に使用する。マウス操作場所については、（図 5）に示す。

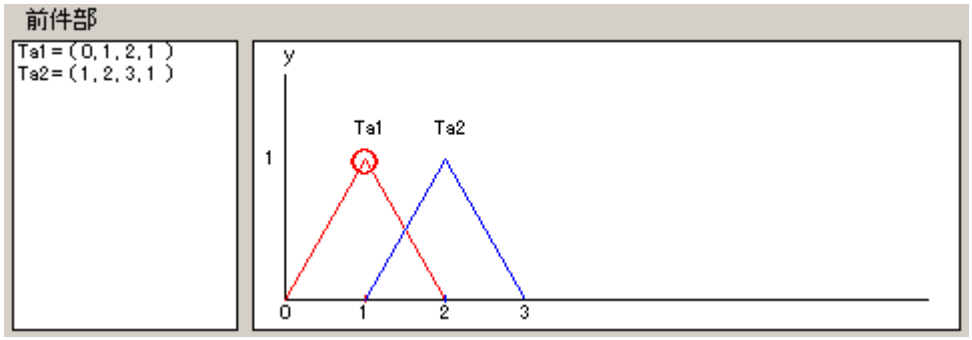


（図 5）

（図 5）の赤丸をドラッグすることで、移動可能となる。移動量は 0.1 ごととなっている。

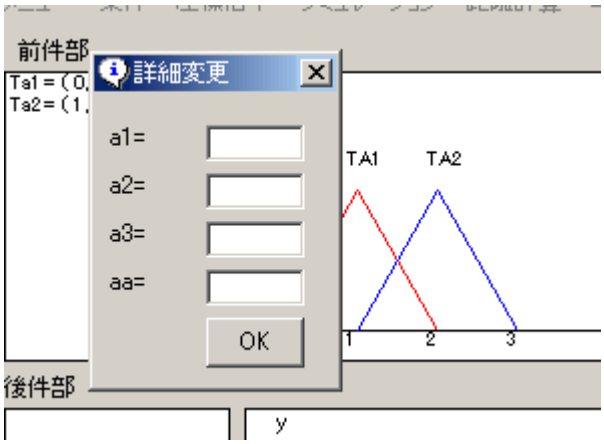
③　ファジィ集合のパラメータ詳細変更

データ入力し直すことにより、ファジィ集合のパラメータを変更する事ができる。①及び②の方法では、ドラッグすることで 0.1 づつファジィ集合の移動が可能である。これは、0.1 よりも小さい量、または大幅な移動をさせたい時など、また新たにファジィ集合のパラメータを入力し直したい時などに有効である。マウス操作場所については、(図 6) に示す。



(図 6)

(図 6) の赤丸を右クリックすることで、(図 7) のようなウィンドウが表示される。

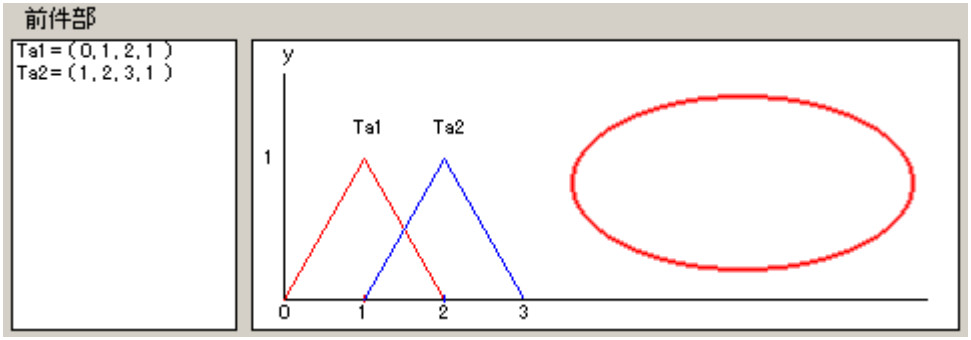


(図 7)

ここでテキストボックスにそれぞれファジィ集合のパラメータを入力し、OK を押すことでファジィ集合のパラメータが変更される。

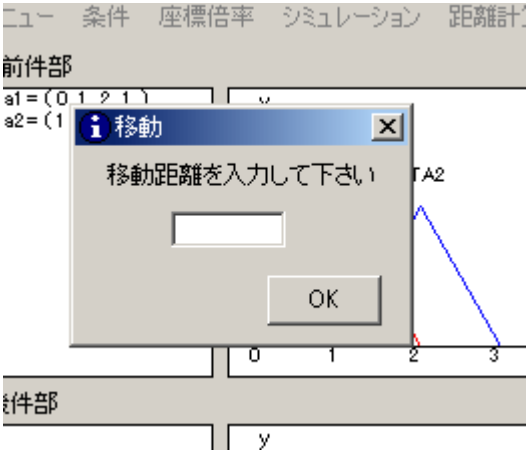
④　前件部もしくは後件部の全ファジィ集合移動

ルールの前件部、もしくは後件部のファジィ集合を全て平行移動することができる。これは、ルール自体をいくらか数値分移動させたい時等に使用する。マウスの操作場所については、（図 8）に示す。



（図 8）

（図 8）の赤丸部分を SHIFT キー＋クリックすることで、（図 9）のようなウィンドウが表示される。

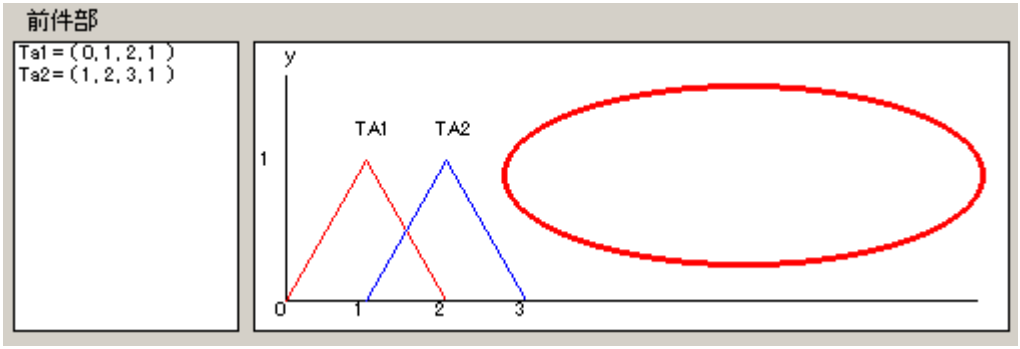


（図 9）

ここで、テキストボックスに任意の移動量を入力して OK をクリックすれば、ファジィ集合を全て、任意の移動量分移動させることができる。

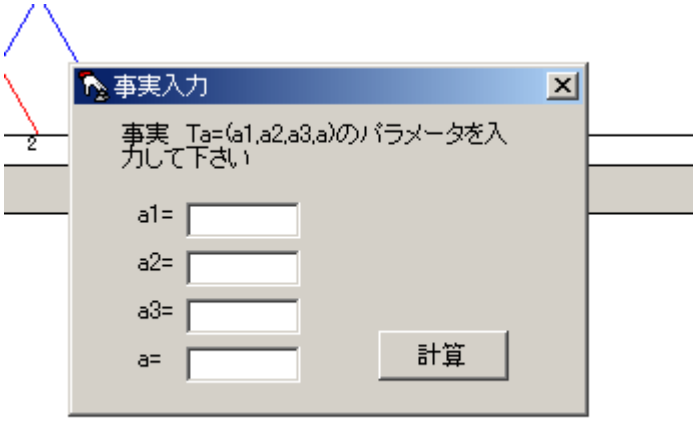
⑤　事実直接入力

メニューを使用せずに、直接事実を入力することができる。ただし、データ入力のみ可なので、自動入力はメニューから行う必要がある。これは、メニューを使用せずに事実を即座に入力できるショートカットの役目をする。マウスの操作場所については、（図10）に示す。



（図 10）

（図 10）の赤丸の部分、背景の部分をダブルクリックすると、（図 11）のようなウィンドウが表示される。

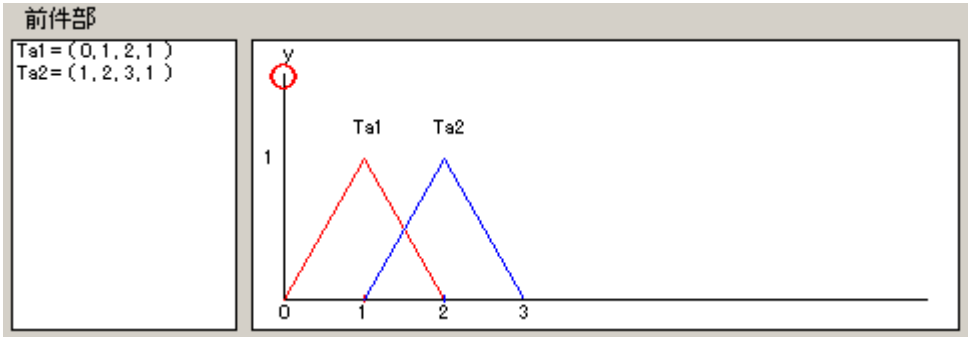


（図 11）

ここで、テキストボックスに事実のファジィ集合を任意に入力して計算ボタンをクリックすれば、即座に事実入力及び推論が可能である。

⑥　y 軸移動

y 軸の位置を変更することができる。ルールで－の値を持つファジィ集合などがある場合、それらのファジィ集合は初期画面では表示ボックスの左側に隠れてしまうわけだが、y 軸をファジィ集合表示ボックスの中央に持ってくるなどすることで、これらの問題を回避することができる。マウスの操作場所については（図 12）に示す。



（図 12）

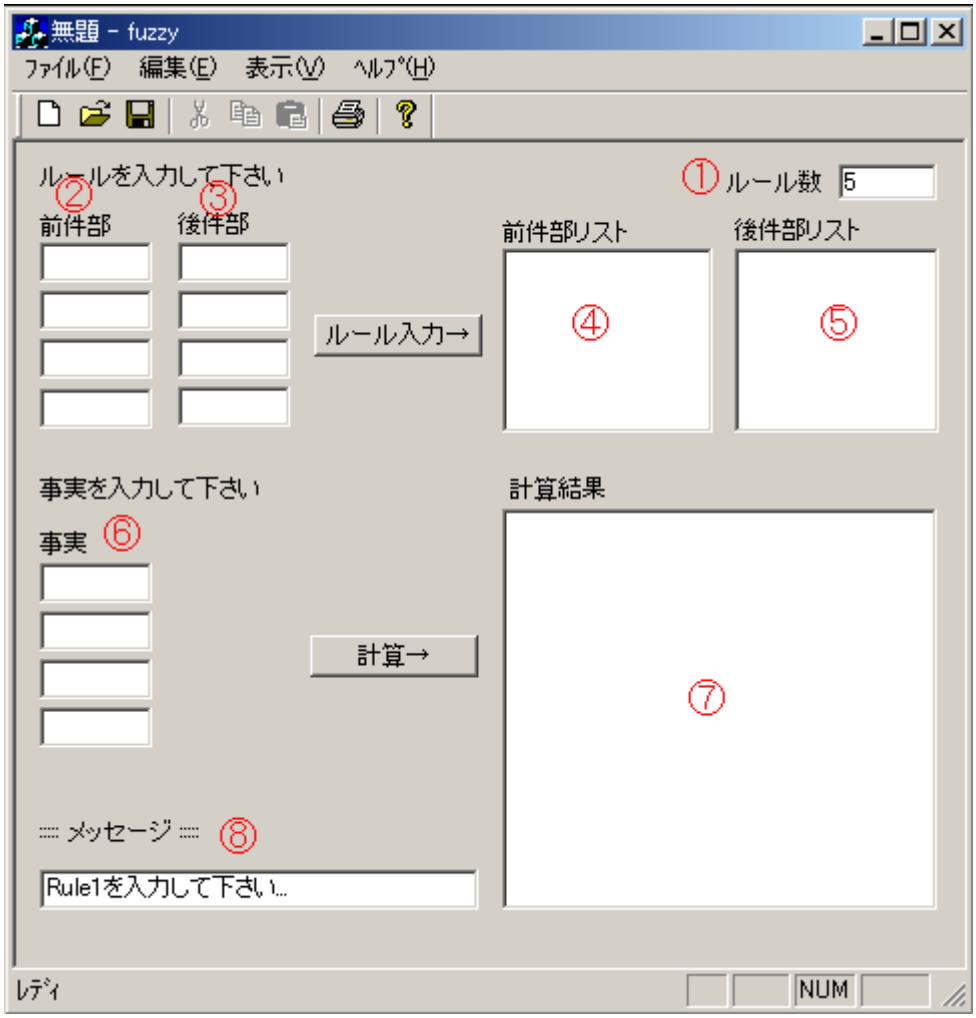
（図 12）の赤丸の部分をドラッグすると、y 軸が移動可能となる。

以上、これらのマウス操作を使うことでファジィ推論をより円滑かつ快適に行うことができる。

付録２　VisualC++により開発した推論エンジンの使用方法

付録２－１　各部の名称

この節では、推論エンジン起動画面の各部の名称・役目について説明する。（図 13）に起動画面を示す。なお、番号順に説明していく。



（図 13）

① ルール数入力ボックス

ここで、ルール数の設定を行う。初期値はルール数５となっている。任意の数値を入れるだけで、その時点からルール数を変更できる。

② 前件部ファジィ集合パラメータ入力ボックス

前件部のファジィ集合各パラメータを入力する。

③ 後件部ファジィ集合パラメータ入力ボックス

後件部のファジィ集合各パラメータを入力する。

④　ファジィ集合データ表示ボックス

前件部のファジィ集合のパラメータがルール追加される度にリスト表示される。

⑤　件部ファジィ集合データ表示ボックス

③と同じく、後件部のファジィ集合のパラメータがルール追加される度にリスト表示される。

⑥　実パラメータ入力ボックス

事実のファジィ集合パラメータを入力する。

⑦　計算結果表示ボックス

ここには事実のパラメータ、前件部ファジィ集合と事実の距離、推論結果のメンバーシップ関数を計算するのに必要な Kq の計算値、推論結果のメンバーシップ関数、推論結果のメンバーシップ関数の重心、という順でリスト表示される。

⑧　コメント

入力手順などを表示してくれる。このメッセージに沿って入力を進めれば簡単にファジィ推論を行える。

付録２－２　使用方法及び使用手順

この節では、この推論エンジンの主な使用方法及び使用手順について説明する。この推論エンジンは、グラフィック表示などは一切なく、ファジィ集合も全てデータ入力制となっている。

推論を行うにあたって、以下のような手順を参考に示す。

１、　ルール数を入力

２、　前件部及び後件部ファジィ集合のパラメータ入力

３、　入力ボタンを押して登録する。２～３の動作をルール数の分繰り返す。

４、　事実を入力する。

５、　計算ボタンを押す。

この手順で推論が可能である。推論結果は計算結果表示ボックスにリスト表示される。これは、何度か推論した後でもさかのぼって過去のデータを参照できるようになっている。

新たにルールを入力し直したい場合には、もう一度１の手順に戻り、ルール数の分登録を繰り返せば、新たなルール設定で推論が可能になる。そのままのルールで良い場合は４の手順からを繰り返せば、事実のみを変更して推論が可能である。

全てのデータを初期化して新たに推論を始めたい場合は、「ファイル」→「新規作成」を押し、新しいウィンドウにすれば良い。