Windows 環境での計測制御のためのプログラム開発に関する研究（第３報）

生産技術部
林健一郎，大坪昭文

Windows 上でよく用いられているプログラミング言語の Visual Basic 言語は，GUI 機能を用いて比較的容易にプログラムを作成できるが，その反面できあがったプログラムの実行速度が遅くなるので，特に計測制御のためのプログラムを開発する場合に問題がある．そこで，本報告では，MS-DOS 上で C 言語により開発したファジィ制御のプログラムを Visual C++言語を用いて DLL ファイル化し，それをファジィ制御命令として Visual Basic 言語に追加するという方法で，プログラムの再利用と実行速度の高速化を図った．その結果，ファジィ制御のプログラムは，PID 制御のプログラムと比べ構造が複雑でステップ数も多いが DLL ファイル化は可能であり，DLL ファイル化の効果は PID 制御の場合より一層顕著となることが分かった．

１．はじめに

これまで多くの県内企業においては，パソコンの OS（Operating System）の MS-DOS 上で計測制御のためのプログラム開発を行ってきた．しかしながら，近年 Windows95の登場を契機としてパソコンの OS が MS-DOS から Windows 系へ移行し，それが現在のパソコンの主流となっているため，県内企業においても Windows 上でプログラム開発を行うことが急務な課題となっている．また，従来の MS-DOS 上で C 言語により開発してきたプログラム資産を Windows 上で再利用することも急務な課題となっている．

Windows 上でよく用いられているプログラミング言語の Visual Basic 言語は，極めて強力な GUI (Graphical User Interface) 機能を有するので，Visual C++言語と比較してプログラム開発が容易となるが，その反面プログラムの実行速度が遅くなるという問題がある．このことは計測制御を目的としたプログラム開発において大きな問題と考えられる．本研究では，これらの課題や問題の解決を目的として，プログラムの実行速度の高速化が要求される部分は，Visual C++言語を用いて C 言語で開発したプログラムの DLL（Dynamic Link Library）ファイル化を行い，それを Visual Basic 言語から呼び出して用いることで計測制御におけるプログラムの再利用と実行速度の高速化を図る．

平成13年度においては，PID 制御のプログラムの DLL ファイル化を Visual C++言語を用いて行い，それを Visual Basic 言語に PID 制御命令として追加し，むだ時間のある一次遅れ系を制御対象としたシミュレーション実験によりその有効性を確認した．最終年度である平成14年度においては，平成13年度の PID 制御のプログラムと比較して，プログラムの構造が複雑でステップ数も多いファジィ制御のプログラムにおいても DLL ファイル化が可能であり，その場合 DLL ファイル化の効果がより顕著となることを，平成13年度と同様なシミュレーション実験により確認する．

２．Visual Basic 言語におけるファジィ制御命令の実現

本章では，MS-DOS 上で C 言語により記述したファジィ制御のプログラム（この場合，関数）を，Visual Basic 言語からファジィ制御命令として使用できるように DLL ファイル化する方法を示す．ここで，DLL ファイルは Visual C++言語を用いて図１に示すような手順に従って作成される．

手順１：Win32 Dynamic-Link Library のプロジェクトを作成する．

Visual C++言語においては，１つのプログラムを作成するために必要な複数のファイルがプロジェクトという単位で管理されており，DLL ファイル用のプロジェクトの作成は次のように行われる．

Visual C++言語の統合開発環境である Developer Studio を起動し，「ファイル」メニューから「新規作成」を選択して，「Win32 Dynamic-Link Library」を選択後，プロジェクトの「位置」と「プロジェクト名」を入力する．ここで，プロジェクト名を「fuzzy_mdl」とする

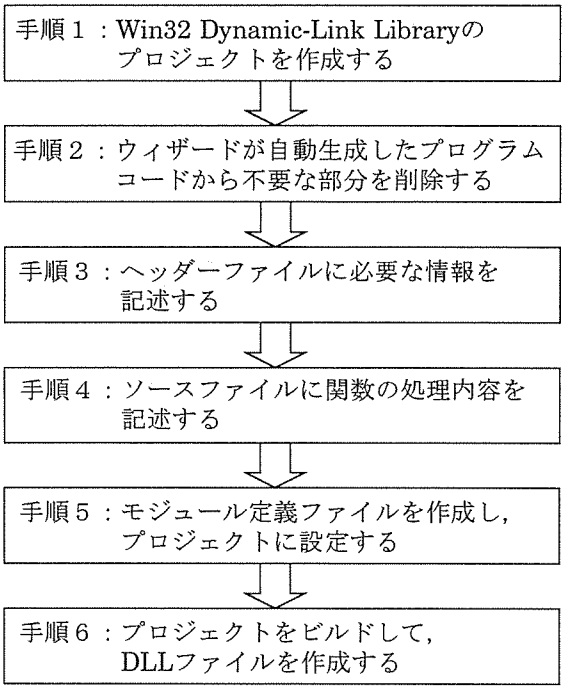

図１　DLLファイルを作成する手順

ので，このプロジェクトにおいて作成されるDLL ファイル名は「fuzzy_mdl.dll」となる．また，「位置」で指定したディレクトリにはこのプロジェクトで作成される種々のファイルが格納される．次の「Win32 Dynamic-Link Library － ステップ1/1」ウィンドウにおいては「シンボルをエクスポートするDLL」を選択する．

手順２：ウィザードが自動生成したプログラムコードから不要な部分を削除する．

Developer Studio ではウィザードと呼ばれる機能により，プログラム作成に必要なファイルがその中にプログラムコードの雛形を伴って自動的に生成される．ここでは，ヘッダーファイルの「fuzzy_mdl.h」とソースファイルの「fuzzy_mdl.cpp」について修正を行うが，それぞれDLL ファイル作成に不要なプログラムコードの部分を削除しておかねばならない．

手順３：ヘッダーファイルに必要な情報を記述する．

Visual Basic 言語からDLLファイルに収録されたファジィ制御の関数を呼び出せるようにするためには，手順２のヘッダーファイルにおいて関数のプロトタイプ宣言を行わねばならない．具体的には，図２の DLL 作成用ヘッダーファイルに示すように，関数「fuzzy_cont()」の型宣言の前に関数のプロトタイプ宣言として「FUZZY_MDL_API」と「__stdcall」を記述する．

手順４：ソースファイルに関数の処理内容を記述する．

ここでは，手順２で作成されたソースファイルにおいて，DLL ファイルに収録するファジィ制御の関数の処理内容を記述することになる．Visual Basic 言語から呼び出す関数には型宣言の前に手順３のヘッダーファイルにおけるプロトタイプ宣言と同様な記述を，一方，DLL ファイルの内部だけで使用される関数には C 言語における通常の記述を行う．具体的には図３の DLL 作成用ソースファイルに示すような記述となるが，この場合 Visual Basic 言語の方から呼び出される関数は「fuzzy_cont()」だけである．

手順５：モジュール定義ファイルを作成し，プロジェクトに設定する．

モジュール定義ファイルには，Visual Basic 言語から DLL ファイルに収録した関数を呼び出す際の関数名を記述しなければならない．このモジュール定義ファイルを用いなくても DLL ファイ

```
 1: // fuzzy_mdl.h : DLL作成用ヘッダーファイル
 2: // ファジィ制御(min-max-重心法)
 3:
 4: #ifdef FUZZY_MDL_EXPORTS     // DLLファイルを作成する場合
 5: #define FUZZY_MDL_API __declspec(dllexport)
 6: #else                        // VC++によりDLLの関数を呼び出す場合
 7: #define FUZZY_MDL_API __declspec(dllimport)
 8: #endif
 9:
10: // 他のプログラムから呼び出されるファジィ制御の関数のプロトタイプ宣言
11: FUZZY_MDL_API double __stdcall fuzzy_cont(double u0, double v0);
```

図２　DLL 作成用ヘッダーファイル

```
 1: // fuzzy_mdl.cpp : DLL作成用ソースファイル
 2: // ファジィ制御(min-max-重心法)
 3:
 4: #include "stdafx.h"      // 自動生成されたWindows用のヘッダーファイルの取り込み
 5: #include "fuzzy_mdl.h"   // DLL作成用ヘッダーファイルの取り込み
 6:
 7: // 自動生成されたDLLのエントリポイント
 8: BOOL APIENTRY DllMain( HANDLE hModule,
 9:                        DWORD  ul_reason_for_call,
10:                        LPVOID lpReserved
11:                      )
12: {
13:     switch (ul_reason_for_call)
14:     {
15:         case DLL_PROCESS_ATTACH:
16:         case DLL_THREAD_ATTACH:
17:         case DLL_THREAD_DETACH:
18:         case DLL_PROCESS_DETACH:
19:             break;
20:     }
21:     return TRUE;
22: }
23:
24: // ここからファジィ制御(min-max-重心法)の関数の記述
25:
26: /* 使用する標準関数のヘッダーファイルの取り込み */
27: #include <stdio.h>
28: #include <math.h>
29: #include <string.h>
30: #include <process.h>
31:
32: /* ファジィ制御に使用する関数のプロトタイプ宣言 */
33: void init(void);                        /* ファジィ制御用定数の初期化 */
34: void rule_open(void);                   /* 制御規則の呼出 */
35: void init_memb(void);                   /* 前・後件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の初期化 */
36: double find_membf(char *);              /* 前件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の頂点位置を求める */
37: double *find_memb(char *);              /* 後件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の配列ﾎﾟｲﾝﾀを求める */
38: void ant_match(double, double, int);          /* 前件部適合度を求める */
39: void _implication(int, double, double *);    /* 各規則による推論結果を求める */
40: void _unify(void);                      /* 統合化 */
41: double _defuzzification(void);          /* 非ファジィ化 (重心法)  */
42: double triangle(double, double, double);     /* 三角型ﾒﾝﾊﾞｰｼｯﾌﾟ関数 */
43: double _min(double, double);            /* ｍｉｎ演算 */
44: double _max(double, double);            /* ｍａｘ演算 */
45:
46: /* 定数の定義 */
47: #define N_RULES     49                  /* 制御規則の最大数(7×7分割) */
48: /* 前件部パラメータ */
49: #define TRNS        6.                  /* 台集合区間[-1,1]->[-6,6]の変換定数 */
50: #define ANT_WIDTH   2.                  /* 前件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の幅 */
51: /* 後件部パラメータ */
52: #define DIV         65                  /* 後件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の離散化点数 */
53: #define CON_WIDTH   8.                  /* 後件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の幅 */
54:
55: /* 外部変数の定義 */
56: char rule[N_RULES][3][3];               /* 制御規則 */
```

図３　DLL 作成用ソースファイル(その１)

```
57: int rule_count;                     /* 選択された制御規則の総数 */
58: double rule_match[N_RULES];         /* 各規則の適合度 */
59: double top_memb[N_RULES][2];        /* 各規則の前件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数頂点位置 */
60: double *con_memb[N_RULES];          /* 各規則の後件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数配列ポインタ */
61: double c_d[N_RULES][DIV];           /* 各規則の推論結果のﾒﾝﾊﾞｰｼｯﾌﾟ関数の格納用 */
62: double e[DIV];                      /* 統合化後のﾒﾝﾊﾞｰｼｯﾌﾟ関数の格納用 */
63: int flag=0;                         /* 初期化終了判断用変数(0: 初期化必要) */
64:
65: /* 離散化した後件部ﾒﾝﾊﾞｰｼｯﾌﾟ関数の配列 */
66: double pb[DIV],pm[DIV],ps[DIV],zo[DIV],ns[DIV],nm[DIV],nb[DIV];
67:
68: /* 他のプログラムから呼び出されるファジィ制御の関数 */
69: FUZZY_MDL_API double __stdcall fuzzy_cont(double u0,     /* 入力１ */
70:                                           double v0)     /* 入力２ */
71: {
72:     int i;              /* カウンタ */
73:     double w0;          /* 出力 */
74:
75:     if(flag==0) init(); /* ファジィ制御用定数の初期化 */
76:
77:     /* 入力の制限 */
78:     if (u0<-1.0)        u0=-1.0;    /* 入力１の下限 */
79:     else if(u0>1.0)     u0=1.0;     /* 入力１の上限 */
80:     if (v0<-1.0)        v0=-1.0;    /* 入力２の下限 */
81:     else if(v0>1.0)     v0=1.0;     /* 入力２の上限 */
82:
83:     u0 *=TRNS;              /* 台集合変換[-1,1]->[-6,6] */
84:     v0 *=TRNS;
85:
86:     for(i=0;i<DIV;i++)      /* 統合化用の配列のリセット */
87:         e[i]=0.0;
88:
89:     /*** STEP 1:前件部適合度を求める ***/
90:     for(i=0;i<rule_count;i++)
91:     {
92:         ant_match(u0,v0,i);
93:     }
94:
95:     /*** STEP 2:各規則による推論結果を求める ***/
96:     for(i=0;i<rule_count;i++)
97:     {
98:         if(rule_match[i] > 0.0)
99:         {   /* 前件部適合度＞０ならば含意演算を実行 */
100:            _implication(i,rule_match[i],con_memb[i]);
101:        }
102:    }
103:
104:    /*** STEP 3:統合化 ***/
105:    _unify();
106:
107:    /*** STEP 4:非ファジィ化 ***/
108:    w0=_defuzzification();
109:
110:    return(w0);     /* ファジィ制御器からの出力を返す */
111: }
```

図３　DLL 作成用ソースファイル(その２)

```
1: ; fuzzy_mdl.def: モジュール定義ファイル
2: ; ファジィ制御(min-max-重心法)のDLLファイル作成用
3: ;
4: LIBRARY fuzzy_mdl.dll
5:
6: EXPORTS
7:    fuzzy_cont
```

図4　モジュール定義ファイル

ルを作成することはできるが，Developer Studio により DLL ファイル作成時に自動設定される関数名は，得てして使い難い名前に設定される．

図4にファイル名「fuzzy_mdl.def」のモジュール定義ファイルを示しているが，そこで「LIBRARY」の所には DLL ファイル名，「EXPORTS」の所には呼び出す際の関数名が記述されている．このモジュール定義ファイルをプロジェクトに追加するためには，「プロジェクト」メニューから「設定」を選択し，「プロジェクトの設定」ウィンドウの「リンク」タブの中の「プロジェクトオプション」欄の末尾に「/def:"fuzzy_mdl.def"」と記述することになる．

手順6：プロジェクトをビルドして，DLL ファイルを作成する．

Developer Studio の「ビルド」メニューから「ビルド」を選択すると，以上一連の手順に間違いがない場合には，ファジィ制御の関数の DLL ファイル「fuzzy_mdl.dll」が作成される．この DLL ファイルは Visual Basic 言語においてその参照宣言で指定するディレクトリにコピーしておかねばならない．

なお，MS-DOS 上で C 言語で記述したファジィ制御のプログラムでは，エラー発生時にメッセージを表示するようなプログラムとなっているが，Windows 上でメッセージを表示するようにするには，そのプログラムを一部変更しなければならない．具体的には，メッセージを表示する場合には C 言語の「printf 関数」に代えて，Windows が提供する関数セットである API(Application Programming Interface) の「MessageBox 関数」などを Visual C++言語から使用することになる．また，キーボードからパラメータ入力を行う場合にも，今回は使用していないが Visual Basic 言語の「InputBox 関数」を用いて入力を行い，その値を DLL ファイル化したファジィ制御の関数へ引数で渡すような処理の追加が必要となってくる．

3．Visual Basic 言語によるファジィ制御のシミュレーション実験

本章では，Visual C++言語で DLL ファイル化したファジィ制御の動作確認と実行速度の評価を，Visual Basic 言語を用いて記述したシミュレーション実験により行う．

本シミュレーション実験で制御対象として用いたむだ時間のある一次遅れ系の伝達関数を次式に示す．

$$G(s)=\frac{e^{-2s}}{1+50s}$$

むだ時間のある一次遅れ系は，実際の制御対象を近似的にモデル化する際最もよく用いられるものであり，ここでは，一般的なケースである遅れがむだ時間より支配的となるような伝達関数のパラメータとしている．

表1　ファジィ制御規則

ΔE \ E	*NB*	*NM*	*NS*	*ZO*	*PS*	*PM*	*PB*
PB	*ZO*	*PS*	*PM*	*PB*	*PB*	*PB*	*PB*
PM	*NS*	*ZO*	*PS*	*PM*	*PB*	*PB*	*PB*
PS	*NM*	*NS*	*ZO*	*PS*	*PM*	*PB*	*PB*
ZO	*NB*	*NM*	*NS*	*ZO*	*PS*	*PM*	*PB*
NS	*NB*	*NB*	*NM*	*NS*	*ZO*	*PS*	*PM*
NM	*NB*	*NB*	*NB*	*NM*	*NS*	*ZO*	*PS*
NB	*NB*	*NB*	*NB*	*NB*	*NM*	*NS*	*ZO*

NB：Negative Big　　*PB*：Positive Big
NM：Negative Medium　　*PM*：Positive Medium
NS：Negative Small　　*PS*：Positive Small
ZO：Zero

また，本シミュレーション実験におけるファジィ制御では，制御規則表を規則で埋め尽くした表1に示すようなファジィ制御規則を用いるが，このファジィ制御規則は制御の現場で実際良く用いられるものである．なお，表1において E は偏差(＝目標値－制御量)，ΔE は偏差の変化分を表している．

Visual Basic 言語を用いたファジィ制御のシミュレーションプログラムは，ファジィ制御の関数に関するコーディング部分を除いて，ここでも先に作成した PID 制御のシミュレーションプログラムを用いる．従って，このシミュレーションプログラムの標準モジュールにおいて，DLL ファイルに収録されているファジィ制御の関数を Visual Basic 言語から呼び出せるように，関数の参照宣言を次のように行わなければならない．

```
Public Declare Function fuzzy_cont Lib
"DLL ファイルのあるディレクトリ名
¥fuzzy_mdl.dll" (ByVal u As Double,
ByVal v As Double) As Double
```

図5は Visual Basic 言語のプロジェクトの実行命令を用いてファジィ制御のシミュレーションプログラムを実行した結果であるが，そのシミュレーションの画面上では，制御量の数値とグラフの表示によりファジィ制御の時間経過の様子を見ることができる．なお，このシミュレーション実験においては，制御対象のむだ時間のある一次遅れ系に対して，できるだけ早くオーバーシュートなしで制御量が初期値の0から目標値の1に収束するように，ファジィ制御器の制御パラメータの調整を行っている．

ここで，DLL ファイル化したファジィ制御のプログラムの実行速度を評価するために，Visual Basic 言語で新たに作成したファジィ制御のプログラムとの実行速度の比較実験を行う．この実行速度の比較実験では，Visual Basic 言語のプロジェクト

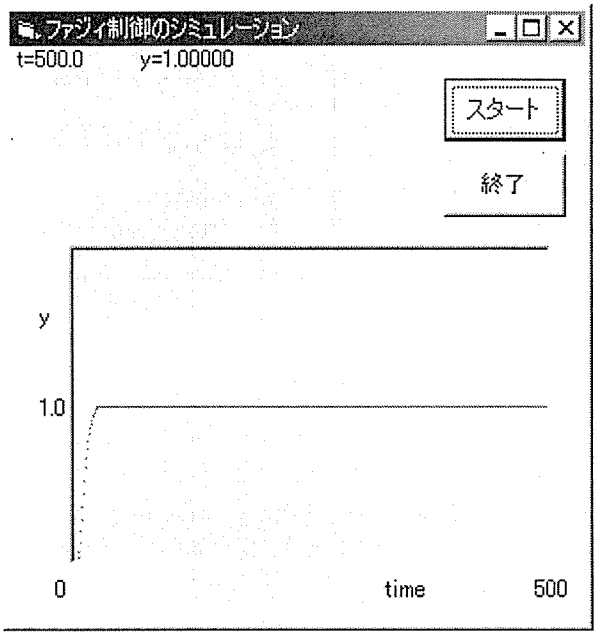

図5　ファジィ制御のシミュレーション結果

の実行命令を用いて，ファジィ制御の命令を指定回数だけ実行させたときの両者の所要時間を計測する．時間の計測には Visual Basic 言語の現在時刻を取得する「Timer 関数」を使用して，実験の開始時刻と終了時刻との時間差から所要時間を算出する．

表2の①は Visual Basic 言語で記述したファジィ制御のプログラムを用いる場合の，②は DLL ファイル化したファジィ制御のプログラムを用いる場合の所要時間を，ファジィ制御命令の実行回数の1万回と10万回の各々10回計測してそれらの平均値を求めた結果である．この実験結果の①の平均値を②の平均値で割った結果から，Visual C++言語により DLL ファイル化したファジィ制御の方が Visual Basic 言語で記述したファジィ制御と比較して，約24倍もの実行速度となっていることが分かる．なお，このことはむだ時間のある一次遅れ系のパラメータの如何にかかわらず成立する．

表2　ファジィ制御のプログラムの実行速度の比較
(① Visual Basic 言語で記述，② DLL ファイル化)

方法 / 回数	①	②	①／②
1万	13.04[s]	0.54[s]	24.15
10万	130.93[s]	5.41[s]	24.20

平成13年度において，DLL ファイル化した PID 制御のプログラムと Visual Basic 言語で新たに作成した PID 制御のプログラムとの本年度と同様な実行速度の比較実験を行った．その結果，DLL ファイル化した PID 制御の方が Visual Basic 言語で記述した PID 制御と比較して，約3倍の実行速度となっていた．平成13年度に得た結果と本年度得た結果から；ファジィ制御のプログラムのように PID 制御のプログラムと比べプログラムの構造が複雑でステップ数も多い場合でも DLL ファイル化は可能であり，その場合 DLL ファイル化の効果が一層顕著に表れると言うことができる．ちなみに，ファジィ制御のプログラムの場合が PID 制御のプログラムの場合と比較して約8倍の DLL ファイル化の効果が表れている．

なお，本シミュレーション実験においては，NEC 社製のパーソナルコンピュータ PC-9821Ra43 (CPU：Celeron，クロック：433MHz) を使用し，OS の Windows98 (Second Edition) の環境下で Microsoft の Visual Basic (Version 6.0) を用いて作成したプログラムを実行した．

4．おわりに

本報告においては，MS-DOS 上で C 言語で開発したファジィ制御のプログラムの DLL ファイル化を Visual C++言語を用いて行い，それを Visual Basic 言語へファジィ制御命令として追加を行った．そして，制御対象がむだ時間のある一次遅れ系のシミュレーション実験により，追加したファジィ制御命令の動作確認と実行速度の評価を行った．その結果，C 言語で開発したファジィ制御のプログラムを DLL ファイル化し再利用したものは，新たにファジィ制御のプログラムを Visual Basic 言語で記述したものと比較して，約24倍もの実行速度を得ることができた．ちなみに，ファジィ制御のプログラムを DLL ファイル化した効果は PID 制御のプログラムを DLL ファイル化した効果の約8倍となった．

なお，本研究を進めるにあたり多数の文献を参考としたが，例えば，その内以下に示すような文献が挙げられる．

参考文献

1) 矢沢久雄，Visual Basic プログラマのための Visual C++入門，アスキー，2000.
2) 横山直隆，Visual Basic による制御実習入門，㈱シータスク，1999.
3) 山住富也，森博，小池慎一，理系のための Visual Basic 6.0実践入門，技術評論社，1999.
4) 北山洋幸，ひとつ上を目指す VB ユーザーのための Visual Basic システムプログラミング入門，技術評論社，1999.
5) 蔵守伸一，手にとるように分かる Visual Basic 関数事典，オーム社，1999.
6) 林健一郎，白仁田和彦，大坪昭文，佐賀県工業技術センター研究報告書，No.10，pp.79-85，2001.