模糊 PID 自整定控制算法简单理解

模糊控制 PID 的参数自调整原理：CPU 根据系统偏差（偏差＝给定－反馈），和偏差变化率（偏差变化率＝当前周期偏差－上周期偏差）查询相应的模糊控制表，得到 Kp，Ki，Kd 三个参数的整定值，然后进行 PID 运算，真正的运用到实际中也就是一张模糊控制查询表，然后就是查表了，也很简单，关键是表的建立还有专家经验的问题等。

单片机模糊 PID 自整定控制算法的实现及仿真

作者：冯桂宁　蒋翔俊

引言

由于液压伺服系统的固有特性(如死区、泄漏、阻尼系数的时变性以及负载干扰的存在)，系统往往会呈现典型的不确定性和非线性特性。这类系统一般很难精确描述控制对象的传递函数或状态方程，而常规的 PID 控制又难以取得良好的控制效果。另外，单一的模糊控制虽不需要精确的数学模型，但是却极易在平衡点附近产生小振幅振荡，从而使整个控制系统不能拥有良好的动态品质。

本文针对这两种控制的优缺点并结合模糊控制技术,探讨了液压伺服系统的模糊自整定 PID 控制方法，同时利用 MATLAB 软件提供的 Simulink 和 Fuzzy 工具箱对液压伺服调节系统的模糊自整定 PID 控制系统进行仿真，并与常规 PID 控制进行 了比较。此外，本文还尝试将控制系统通过单片机的数字化处理，并在电液伺服实验台上进行了测试，测试证明：该方法能使系统的结构简单化，操作灵活化，并可增强可靠性和适应性，提高控制精度和鲁棒性，特别容易实现非线性化控制。

1、模糊 PID 自整定控制器的设计

本控制系统主要完成数据采集、速度显示和速度控制等功能。其中智能模糊控制由单片机完成，并采用规则自整定 PID 控制算法进行过程控制。整个系统的核心是模糊控制器，AT89C51 单片机是控制器的主体模块。电液伺服系统输出的速度信号经传感器和 A / D 转换之后进入单片机，单片机则根据输入的各种命令，并通过模糊控制算法计算控制量，然后将输出信号通过 D / A 转换送给液压伺服系统，从而控制系统的速度。该模糊控制器的硬件框图如图 1 所示。

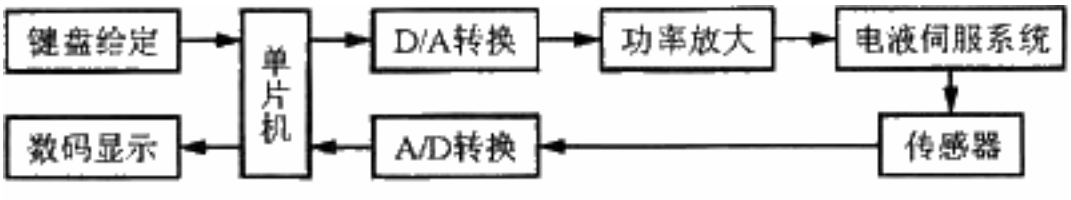

图 1　模糊控制器硬件框图

模糊控制器的主程序包括初始化、键盘管理及控制模块和显示模块的调用等。温度信号的采集、标度变换、控制算法以及速度显示等功能的实现可由各子程序完成。软件的主要流程是：利用 AT89C51 单片机调 A / D 转换、标度转换模块以得到速度的反馈信号，**然后根据偏差和偏差的变化率计算输入量，再由模糊 PID 自整定控制算法得出输出控制量**。启动、停止可通过键盘并利用外部中断产生，有按键输入则调用中断服务程序。该程序的流程图如图 2 所示。

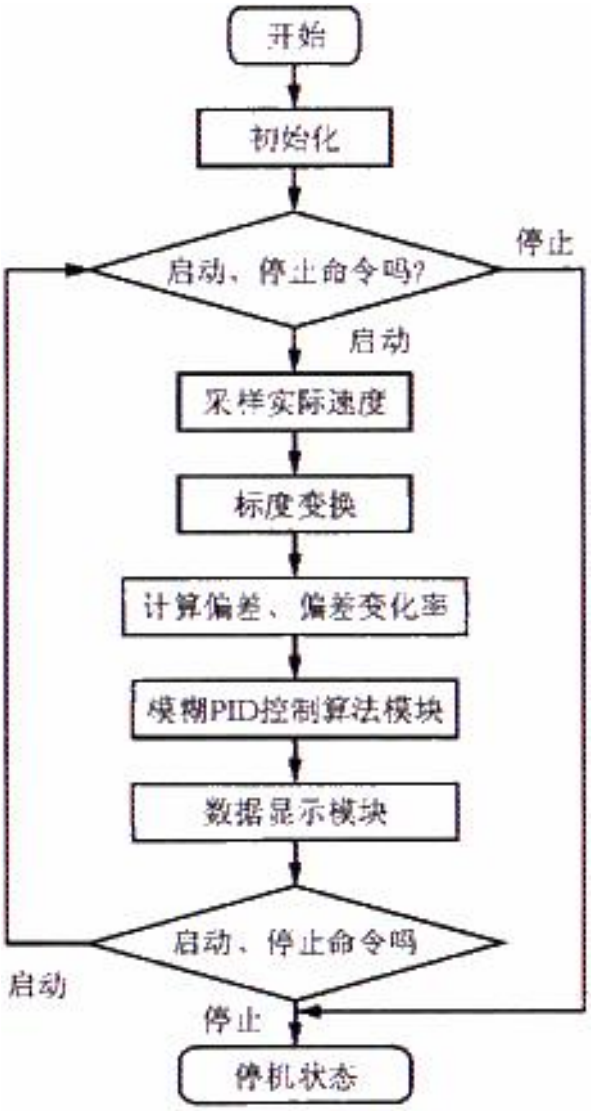

图2　模糊控制器程序控制流程

2、模糊控制器算法研究

采用模糊 PID 自整定控制的目的是使控制器能够根据实际情况调整比例系数 Kp、积分系数 Ki 和微分系数 Kd，以达到调节作用的实时最优。该电液伺服系统的 Fuzzy 自整定 PID 控制系统结构如图 3 所示。

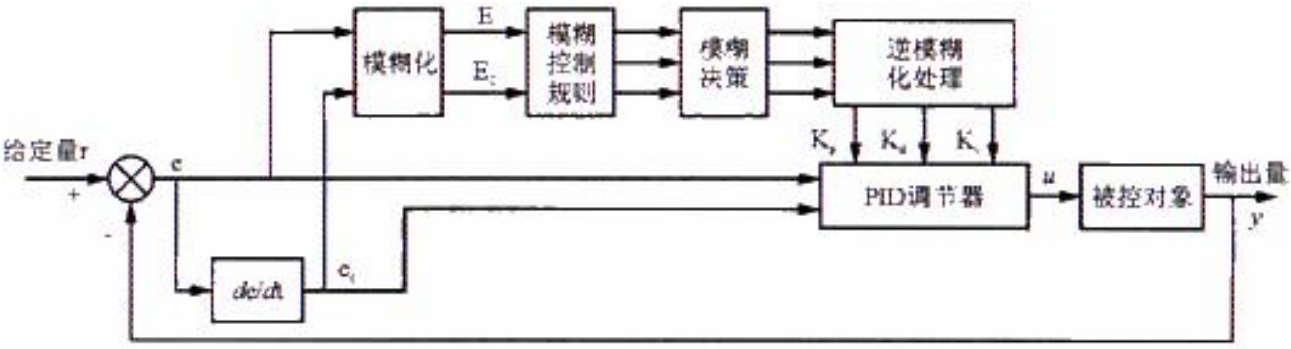

图3 模糊控制器系统结构图

为了简化运算和满足实时性要求，即该调节系统的基本控制仍为 PID 控制，但使 PID 调节参数由模糊自整定控制器根据偏差 e 和偏差变化率 ec 进行自动调整，同时把模糊自整定控制器的模糊部分按 Kp、Ki 和 Kd 分成 3 部分，分别由相应的子推理器来实现。

2.1、输入值的模糊化

模糊自整定 PID 控制器是在 fuzzy 集的论域中进行讨论和计算的，因而首先要将输入变量变换到相应的论域，并将输人数据转换成合适的语言值，也就是要对输入量进行模糊化。结合本液压伺服系统的特性，这里选择模糊变量的模糊集隶属函数为正态分布，具体分布如图 4 所示。根据该规则可把实际误差 e、误差变化率 ec(de / dt)对应的语言变量 E、EC 表示成模糊量。**E、EC 的基本论域为[-6，+6]，将其离散成 13 个等级即[-6，-5，-4，-3，-2，-1，0，+1，+2，+3，+4，+5，+6]。**考虑到控制的精度要求，本设计将[-6，+6]分为负大[NB]、负中[NM]、负小[NS]、零[ZO]、正小[PS]、正中[PM]、正大[PB]等 7 个语言变量，然后由 e、ec 隶属函数根据最大值法得出相应的模糊变量。

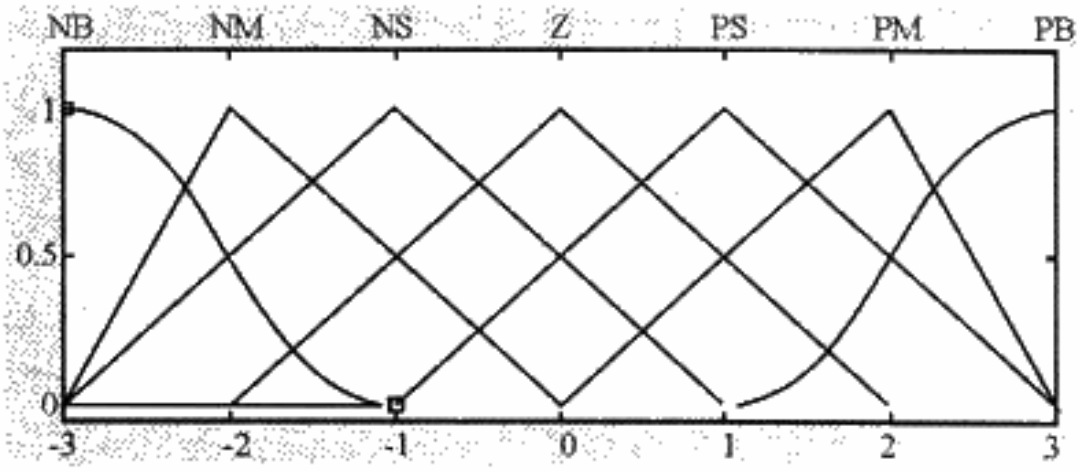

图4 e、e_c隶属函数

2.2、模糊控制规则表的建立

（1）、Kp 控制规则设计

在 PID 控制器中，Kp 值的选取决定于系统的响应速度。增大 Kp 能提高响应速度，减小稳态误差；但是，Kp 值过大会产生较大的超调，甚至使系统不稳定减小 Kp 可以减小超调，提高稳定性，但 Kp 过小会减慢响应速度，延长调节时间。因此，调节初期应适当取较大的 Kp 值以提高响应速度，而在调节中期，Kp 则取较小值，以使系统具有较小的超调并保证一定的响应速度；而在调节过程后期再将 Kp 值调到较大值来减小静差，提高控制精度。Kp 的控制规则如表 1 所列。

表1 K_p的模糊规则表

K_p \ e_c / e	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PM	PM	PS	ZO	ZO
NM	PB	PB	PM	PS	PS	ZO	NS
NS	PB	PM	PM	PS	ZO	NS	NS
ZO	PM	PM	PS	ZO	NS	NM	NM
PS	PS	PS	ZO	NS	NS	NM	NB
PM	PS	ZO	NS	NM	NM	NM	NB
PB	ZO	ZO	NM	NM	NM	NB	NB

	NB	NM	NS	ZO	PS	PM	PB
NB 负大	PB 正大	PB 正大	PM 正中	PM 正中	PS 正小	ZO 零	ZO 零
NM 负中	PB 正大	PB 正大	PM 正中	PS 正小	PS 正小	ZO 零	NS 负小
NS 负小	PB 正大	PM 正中	PM 正中	PS 正小	ZO 零	NS 负小	NS 负小
ZO 零	PM 正中	PM 正中	PS 正小	ZO 零	NS 负小	NM 负中	NM 负中
PS 正小	PS 正小	PS 正小	ZO 零	NS 负小	NS 负小	NM 负中	NB 负大
PM 正中	PS 正小	ZO 零	NS 负小	NM 负中	NM 负中	NM 负中	NB 负大
PB 正大	ZO 零	ZO 零	NM 负中	NM 负中	NM 负中	NB 负大	NB 负大

例如：

PB
ZO 零
NS 负小
NS 负小
NM 负中
NB 负大
NB 负大
NB 负大

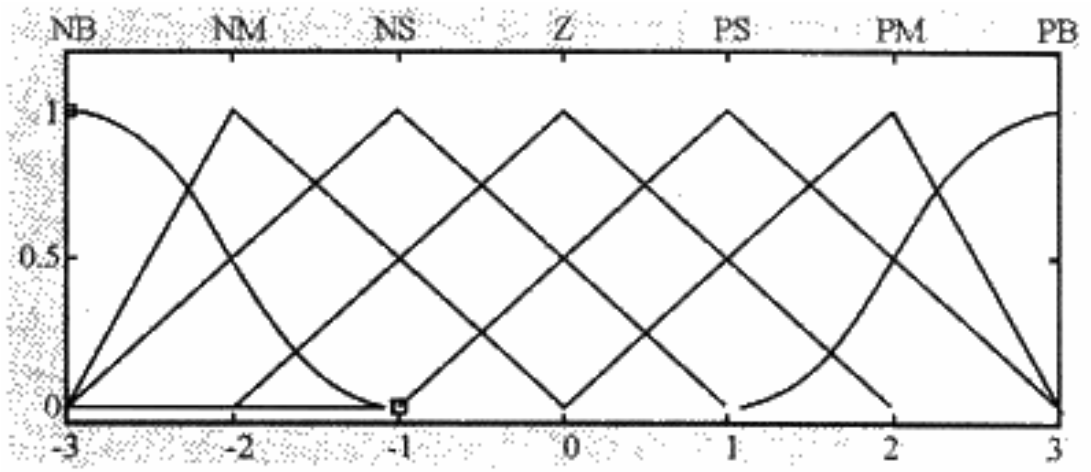

图4 e、e_c隶属函数

隶属度是什么，下面我举个例子说明一吓，例如我说 50 岁以下的不是老年人，70 岁以上的是老年人，那么 60 岁我应该怎样说些，应该说可能是老年人，也可能不是老年人，因为它对于 50 岁的隶属度是 50%，计算工式是(X-50)*0.05，对于 70 的隶属度是 50%，计算工式是(70-x)*0.05。

（2）、Ki 控制规则设计

在系统控制中，积分控制主要是用来消除系统的稳态误差。由于某些原因(如饱和非线性等)，积分过程有可能在调节过程的初期产生积分饱和，从而引起调节过程的较大超调。因此，在调节过程的初期，为防止积分饱和，其积分作用应当弱一些，甚至可以取零；而在调节中期，为了避免影响稳定性，其积分作用应该比较适中；最后在过程的后期，则应增强积分作用，以减小调节静差。依据以上分析，制定的 Ki 控制规则表如表 2 所列。

表2 K_i的模糊规则表

K_i / e_c / e	NB	NM	NS	ZO	PS	PM	PB
NB	NB	NB	NM	NM	NS	ZO	ZO
NM	NB	NB	NM	NS	NS	ZO	ZO
NS	NB	NM	NS	NS	ZO	PS	PS
ZO	NM	NM	NS	ZO	PS	PM	PM
PS	NS	NS	ZO	PS	PS	PM	PB
PM	ZO	ZO	PS	PS	PM	PB	PB
PB	ZO	ZO	PS	PM	PM	PB	PB

	NB	NM	NS	ZO	PS	PM	PB
NB 负大	NB 负大	NB 负大	NM 负中	NM 负中	NS 负小	ZO 零	ZO 零
NM 负中	NB 负大	NB 负大	NM 负中	NS 负小	NS 负小	ZO 零	ZO 零
NS 负小	NB 负大	NM 负中	NS 负小	NS 负小	ZO 零	PS 正小	PS 正小
ZO 零	NM 负中	NM 负中	NS 负小	ZO 零	PS 正小	PM 正中	PM 正中
PS 正小	NS 负小	NS 负小	ZO 零	PS 正小	PS 正小	PM 正中	PB 正大
PM 正中	ZO 零	ZO 零	PS 正小	PS 正小	PM 正中	PB 正大	PB 正大
PB 正大	ZO 零	ZO 零	PS 正小	PM 正中	PM 正中	PB 正大	PB 正大

Ki 的模糊规则表

（3）、Kd 控制规则设计

微分环节的调整主要是针对大惯性过程引入的，微分环节系数的作用在于改变系统的动态特性。系统的微分环节系数能反映信号变化的趋势，并能在偏差信号变化太大之前，在系统中引入一个有效的早期修正信号，从而加快响应速度，减少调整时间，消除振荡．最终改变系统的动态性能。因此，Kd 值的选取对调节动态特性影响很大。Kd 值过大，调节过程制动就会超前，致使调节时间过长；Kd 值过小，调节过程制动就会落后，从而导致超调增加。根据实际过程经验，在调节初期，应加大微分作用，这样可得到较小甚至避免超调；而在中期，由于调节特性对 Kd 值的变化比较敏感，因此，Kd 值应适当小一些并应保持

固定不变；然后在调节后期，Kd 值应减小，以减小被控过程的制动作用，进而补偿在调节过程初期由于 Kd 值较大所造成的调节过程的时间延长。依据以上分析，制定的 Kd 控制规则表如表 3 所列。

表3 K_d的模糊规则表

K_d \ e_r / e	NB	NM	NS	ZO	PS	PM	PB
NB	PS	NS	NB	NB	NB	NM	PS
NM	PS	NS	NB	NM	NM	NS	ZO
NS	ZO	NS	NM	NM	NS	NS	ZO
ZO	ZO	NS	NS	NS	NS	NS	ZO
PS	ZO	ZO	ZO	ZO	ZO	ZO	ZO
PM	PB	NS	PS	PS	PS	PS	PB
PB	PB	PM	PM	PM	PS	PS	PB

	NB	NM	NS	ZO	PS	PM	PB
NB 负大	PS 正小	NS 负小	NB 负大	NB 负大	NB 负大	NM 负中	PS 正小
NM 负中	PS 正小	NS 负小	NB 负大	NM 负中	NM 负中	NS 负小	ZO 零
NS 负小	ZO 零	NS 负小	NM 负中	NM 负中	NS 负小	NS 负小	ZO 零
ZO 零	ZO 零	NS 负小	NS 负小	NS 负小	NS 负小	NS 负小	ZO 零
PS 正小	ZO 零	ZO 零	ZO 零	ZO 零	ZO 零	ZO 零	ZO 零
PM 正中	PB 正大	NS 负小	PS 正小	PS 正小	PS 正小	PS 正小	PB 正大
PB 正大	PB 正大	PM 正中	PM 正中	PM 正中	PS 正小	PS 正小	PB 正大

Kd 的模糊规则表

2.3、逆模糊化处理及输出量的计算

对经过模糊控制规则表求得的 Kp、Ki、Kd 采用重心法进行逆模糊化处理(重心法在此就不做详细介绍)的公式如下：

$$u(k)=K_p e(k)+K_i T\sum_{j=1}^{k} e(j)+K_d \Delta e(k)/T$$

式中，u(k)为 k 采样周期时的输出，e(k)为 k 采样周期时的偏差，T 为采样周期，通过输出 u(k)乘以相应的比例因子 Ku 就可得出精确的输出量 u。其公式如下：

$$u=u(k)K_u$$

3、实验结果分析

常规 PID 控制时通过调节 PID 三个参数，就可以得到系统比较理想的响应图，控制效果的优良与参数的调整有很大的关系，也能提高快速性。但三个参数的调整非常繁琐。而且，如果系统环境不断变化，则参数又必须进行重新调整，往往达不到最优。而采用模糊 PID 控制后，通过模糊控制器对 PID 进行非线性的参数整定，可使系统无论是快速性方面还是稳定性方面都达到比较好的效果。

笔者将上述 PID 控制及模糊 PID 控制分别进行了仿真试验，实验分别在单独模糊 PID 控制情况下和模糊 PID 控制两种情况下进行。并在在线运行过程中通过逻辑规则的结果处理、查表和运算完成了对 PID 参数的在线自矫正。系统的偏差绝对值以及偏差的变化绝对值的取值范围可根据实际经验分别确定为[-0.1 cm / s，0.1 cm / s】和[-0.06 cm / s2，0.06 cm / s2]，以而确定相对控制效果较好时 Kp、Ki、Kd 的取值范围为 Kp[-0.3，0.3]、Ki[-0.06，0.06]、Kd[-3，3]。

传统 PID 和模糊 PID 实验所得的曲线分别如图 5 及图 6 所示。从图中可以发现，采用模糊控制策略整定 PID 参数相对于普通 PID 控制策略，其系统的稳态性得到了较大的改善，响应时间大大减少，超调量也得到了一定的改善。

4、结束语

实验证明：该单片机模糊 **PID** 自整定控制器对于电液伺服控制系统具有较好的效果。实践中可以根据

工程控制的具体情况及对超调量、稳定性、响应速度的不同要求，来调整模糊 **PID** 控制器三个参数的取值范围，从而得到不同的控制精度和控制效果。

总之，本文研究的模糊 PID 控制器具有以下一些特点：

(1) 算法简单实用，本质上不依赖于系统的数字模型；

(2) 可充分利用单片机的软件资源，可靠性高，开发速度快；

(3) 克服了传统 PID 控制器操作的困难，提高了系统的智能化程度；

(4) 模糊 PID 控制器棒性好，具有专家控制器的特点，并可推广应用于其它工作领域。

模糊 PID 的简明理解

声明，此纯属本人见解，欢迎拍砖。

最近因为要做一个温控项目，但参数调整很麻烦，所以花了很长的时间去理解模糊自整定 PID。首先我们将论域画分成｛负大，负中，负小，零，正小，正中，正大｝即｛NB，NM，NS，ZO，PS，PM，PB｝如下图：

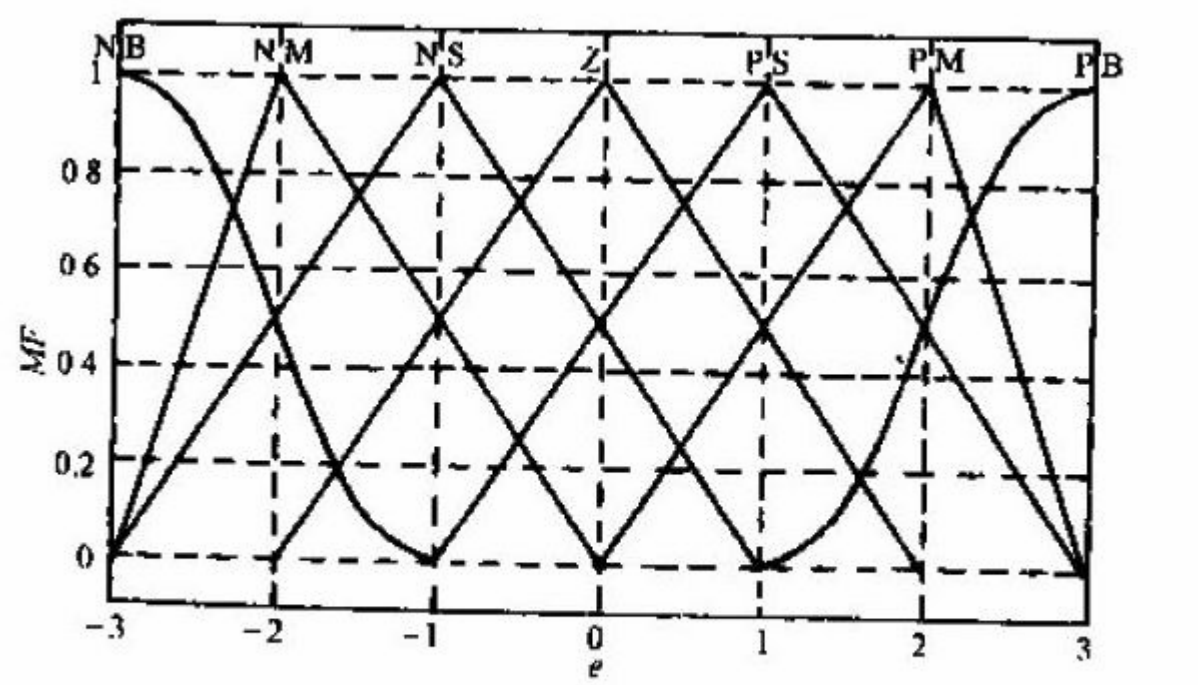

图 3-12　误差的隶属函数

当我们要将 e 或者 ec 模糊化的时候，就要求出 e 或者 ec 对 NB，NM，NS，ZO，PS，PM，PB｝的隶属度。

隶属度是什么，下面我举个例子说明一吓，例如我说 50 岁以下的不是老年人，70 岁以上的是老年人，那么 60 岁我应该怎样说些，应该说可能是老年人，也可能不是老年人，因为它对于 50 岁的隶属度是 50%，计算工式是(X-50)*0.05，对于 70 的隶属度是 50%，计算工式是(70-x)*0.05。

回到正题了，知道了 e 或者 ec 的隶属度后，我们就可以跟椐它的最高隶属度的区域去查规则表，但是有时会有相同隶属度的情况出现，那怎么办？那我们就要用趋势来看是选择那个区域，例如用 NM 和 NS 的交点作例子，如果 e 是趋向正的话，我们选取的是 NM，如果是趋向负的话，我们选择的是 NS。

现在只理解到这，等读懂更多，就再跟大家分响一吓。

模糊 PID 应用

（一）、模糊控制原理

模糊控制是以模糊集理论、模糊语言变量和模糊逻辑推理为基础的一种智能控制方法，它是从行为上模仿人的模糊推理和决策过程的一种智能控制方法。该方法首先将操作人员或专家经验编成模糊规则，然后将来自传感器的实时信号模糊化，将模糊化后的信号作为模糊规则的输入，完成模糊推理，将推理后得到的输出量加到执行器上。

（二）、模糊控制器也称为模糊逻辑控制器由于所采用的模糊控制规则是由模糊理论中模糊条件语句来描述的，因此模糊控制器是一种语言型控制器，故也称为模糊语言控制器.

（三）、模糊控制器的构成

1、模糊化接口（Fuzzy interface）

模糊控制器的输入必须通过模糊化才能用于控制输出的求解，因此它实际上是模糊控制器的输入接口。它的主要作用是将真实的确定量输入转换为一个模糊矢量。对于一个模糊输入变量 e，其模糊子集通常可以作如下方式划分：

（1）＝｛负大，负小，零，正小，正大｝＝｛NB，NS，ZO，PS，PB｝

（2）＝｛负大，负中，负小，零，正小，正中，正大｝＝｛NB，NM，NS，ZO，PS，PM，PB｝

（3）＝｛负大，负中，负小，零负，零正，正小，正中，正大｝＝｛NB，NM，NS，NZ，PZ，PS，PM，PB｝

2、知识库

知识库由数据库和规则库两部分构成。

（1）数据库

数据库所存放的是所有输入、输出变量的全部模糊子集的隶属度矢量值（即经过论域等级离散化以后对应值的集合），若论域为连续域则为隶属度函数。在规则推理的模糊关系方程求解过程中，向推理机提供数据。

（2）规则库

模糊控制器的规则司基于专家知识或手动操作人员长期积累的经验，它是按人的直觉推理的一种语言表示形式。模糊规则通常有一系列的关系词连接而成，如 **if-then、else、also、end、or** 等，关系词必须经过“翻译”才能将模糊规则数值化。最常用的关系词为 **if-then、also**，对于多变量模糊控制系统，还有 and

等。例如，某模糊控制系统输入变量为（误差）和（误差变化），它们对应的语言变量为 E 和 EC，可给出一组模糊规则：）

R1：IF E is NB and EC is NB then U is PB → 规则 1

R2：IF E is NB and EC is NS then U is PM → 规则 2

通常把 if…部分称为“前提部”，而 then…部分称为“结论部”，其基本结构可归纳为 If A and B then C, 其中 A 为论域 U 上的一个模糊子集，B 是论域 V 上的一个模糊子集。根据人工控制经验，可离线组织其控制决策表 R，R 是笛卡儿乘积集上的一个模糊子集，则某一时刻其控制量由下式给出：

规则库是用来存放全部模糊控制规则的，在推理时为“推理机”提供控制规则。规则条数和模糊变量的模糊子集划分有关，划分越细，规则条数越多，但并不代表规则库的准确度越高，规则库的“准确性”还与专家知识的准确度有关。

3．推理与解模糊接口（Inference and Defuzzy-interface）

推理是模糊控制器中，根据输入模糊量，由模糊控制规则完成模糊推理来求解模糊关系方程，并获得模糊控制量的功能部分。在模糊控制中，考虑到推理时间，通常采用运算较简单的推理方法。最基本的有 Zadeh 近似推理，它包含有正向推理和逆向推理两类。正向推理常被用于模糊控制中，而逆向推理一般用于知识工程学领域的专家系统中。

推理结果的获得，表示模糊控制的规则推理功能已经完成。但是，至此所获得的结果仍是一个模糊矢量，不能直接用来作为控制量，还必须作一次转换，求得清晰的控制量输出，即为解模糊。通常把输出端具有转换功能作用的部分称为解模糊接口。

综上所述，模糊控制器实际上就是依靠微机（或单片机）来构成的。它的绝大部分功能都是由计算机程序来完成的。随着专用模糊芯片的研究和开发，也可以由硬件逐步取代各组成单元的软件功能。

```
// FUZZY.h : main header file for the PROJECT_NAME application
//

#pragma once

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// main symbols


// CFUZZYApp:
// See FUZZY.cpp for the implementation of this class
//

class CFUZZYApp : public CWinApp
{
public:
	CFUZZYApp();

	struct	CCPidStruct	m_pid_para;
	struct	pid_struct	m_pid[32];
	short	accel_value[8*8];
	int	channel;

	void fuzzy_step(long *crisp_inputs, long *crisp_outputs);
	void fuzzify_input(int in_index,long in_val);
```

```
        long get_membership_value(int in_index,int mf_index,long in_val);
        void eval_rule(int rule_index);
        long defuzzify_output(int out_index,long *inputs);
        void PidRun(int n);
        void PidDot(int n);
        void SavePidPara();
        void LoadPidPara();



// Overrides
        public:
        virtual BOOL InitInstance();

// Implementation

        DECLARE_MESSAGE_MAP()
};

extern CFUZZYApp theApp;
```

http://www.pudn.com/ > 模糊 PID 控制器.rar > FUZZY.cpp

```
// FUZZY.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "FUZZY.h"
#include "FUZZYDlg.h"
#include "classext.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//========== 增 量 型 PID 控 制 算 法 ==========
short   temp_value[32];
short   setpt[32];
//==========2D-3D=== 模 糊 控 制 算 法 =======
{NL,NM,NS,ZE,PS,PM,PL}==========

struct   MyFuzzy   m_fuzzy;

// 误差 e :          当前测量值 - 目标值
// 误差变化率 ec:    本次误差   - 上次误差
int          num_inputs   = 2;   //2D
int          num_outputs = 3;   //3D
int          num_rules    = 49; //模糊控制规则
```

```
//===========初始化============================
int num_input_mfs[2] = { 7, 7 };//模糊子集

struct    In    Inputs[2] =
{
      { -50, 50 },// min, max
      { -10, 10 } // min, max
};

long       inmem_points[2][7][4] = //    [e,ec][模糊子集][P1,P2,P3,P4]
{
      {
            { -50, -50, -30, -20 },
            { -30, -20, -20, -10 },
            { -20, -10, -10,      0 },
            { -10,      0,      0,    10 },
            {      0,    10,    10,    20 },
            {    10,    20,    20,    30 },
            {    20,    30,    50,    50 }
      },
      {
            { -10, -10, -6, -4 },
            {    -6, -4,    -4, -2 },
            {    -4, -2,    -2,    0 },
            {    -2,    0,      0,    2 },
            {      0,    2,      2,    4 },
            {      2,    4,      4,    6 },
            {      4,    6,    10, 10 }
      }
};
```

```
int num_output_mfs[3] = { 7, 7, 7 };//输出  ΔKp/ΔKI/ΔKD

struct    Out Outputs[] =
{
      { -1200, 1200 },// MIN MAX
      { -1200, 1200 },// MIN MAX
      { -1200, 1200 } // MIN MAX
};

long       outmem_points[3][7][4] = //[ΔKp, ΔKI, ΔKD][模糊子集][P1]
{
      {
            { -1200 },//ΔKp
            {   -800 },
            {   -400 },
            {         0 },
            {     400 },
            {     800 },
            {   1200 }
      },   {
            { -1200 },//ΔKI
            {   -800 },
            {   -400 },
            {         0 },
            {     400 },
            {     800 },
            {   1200 }
      },   {
            { -1200 },//ΔKD
            {   -800 },
```

```
        {   -400 },
        {      0 },
        {    400 },
        {    800 },
        {   1200 }
    }
};

//==========2D-3D====================================================
int num_rule_ants[49] = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2};//2D
int num_rule_cons[49] = { 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3};//3D

struct  Rule    Rules[49] =
{
    { { 0x00, 0x01 }, { 0xb0, 0xa9, 0xa2 } },
    { { 0x08, 0x01 }, { 0xb0, 0xb1, 0xa2 } },
    { { 0x10, 0x01 }, { 0xa8, 0xb1, 0x9a } },
    { { 0x18, 0x01 }, { 0xa8, 0xb1, 0x9a } },
    { { 0x20, 0x01 }, { 0xa0, 0xb1, 0x9a } },
    { { 0x28, 0x01 }, { 0x98, 0xb1, 0xa2 } },
    { { 0x30, 0x01 }, { 0x98, 0xa9, 0xa2 } },
    { { 0x00, 0x09 }, { 0xb0, 0xb1, 0x82 } },
    { { 0x08, 0x09 }, { 0xb0, 0xa1, 0x82 } },
    { { 0x10, 0x09 }, { 0xa8, 0xa1, 0x8a } },
    { { 0x18, 0x09 }, { 0xa0, 0xa1, 0xa2 } },
    { { 0x20, 0x09 }, { 0xa0, 0xa1, 0xa2 } },
    { { 0x28, 0x09 }, { 0x98, 0xa1, 0x8a } },
```

{ { 0x30, 0x09 }, { 0x98, 0xa1, 0x8a } },

{ { 0x00, 0x11 }, { 0xb0, 0x99, 0x82 } },

{ { 0x08, 0x11 }, { 0xb0, 0x99, 0x82 } },

{ { 0x10, 0x11 }, { 0xa8, 0x99, 0x8a } },

{ { 0x18, 0x11 }, { 0x98, 0x99, 0x92 } },

{ { 0x20, 0x11 }, { 0x98, 0x99, 0x8a } },

{ { 0x28, 0x11 }, { 0x90, 0x91, 0x8a } },

{ { 0x30, 0x11 }, { 0x90, 0x91, 0x8a } },

{ { 0x00, 0x19 }, { 0xb0, 0x91, 0x92 } },

{ { 0x08, 0x19 }, { 0xa8, 0x91, 0x92 } },

{ { 0x10, 0x19 }, { 0xa0, 0x91, 0x92 } },

{ { 0x18, 0x19 }, { 0x90, 0x91, 0x92 } },

{ { 0x20, 0x19 }, { 0x88, 0x91, 0x92 } },

{ { 0x28, 0x19 }, { 0x88, 0x91, 0x9a } },

{ { 0x30, 0x19 }, { 0x88, 0x99, 0x8a } },

{ { 0x00, 0x21 }, { 0xa8, 0x99, 0x82 } },

{ { 0x08, 0x21 }, { 0xa0, 0x99, 0x82 } },

{ { 0x10, 0x21 }, { 0x98, 0x99, 0x8a } },

{ { 0x18, 0x21 }, { 0x88, 0x99, 0x92 } },

{ { 0x20, 0x21 }, { 0x88, 0x99, 0x8a } },

{ { 0x28, 0x21 }, { 0x88, 0x91, 0x9a } },

{ { 0x30, 0x21 }, { 0x88, 0x99, 0x8a } },

{ { 0x00, 0x29 }, { 0x98, 0xb1, 0x82 } },

{ { 0x08, 0x29 }, { 0x98, 0xa1, 0x82 } },

{ { 0x10, 0x29 }, { 0x90, 0xb1, 0x9a } },

{ { 0x18, 0x29 }, { 0x88, 0xa1, 0x92 } },

{ { 0x20, 0x29 }, { 0x80, 0xa1, 0x8a } },

{ { 0x28, 0x29 }, { 0x88, 0x99, 0x9a } },

{ { 0x30, 0x29 }, { 0x88, 0xa1, 0x82 } },

{ { 0x00, 0x31 }, { 0x98, 0xa9, 0xa2 } },

```
        { { 0x08, 0x31 }, { 0x98, 0xa9, 0xa2 } },
        { { 0x10, 0x31 }, { 0x90, 0xb1, 0x9a } },
        { { 0x18, 0x31 }, { 0x88, 0xb1, 0x9a } },
        { { 0x20, 0x31 }, { 0x88, 0xb1, 0x9a } },
        { { 0x28, 0x31 }, { 0x88, 0xa1, 0xa2 } },
        { { 0x30, 0x31 }, { 0x88, 0xa1, 0x82 } }
    };
    //=============================================================================
==============
```

```
    // CFUZZYApp

    BEGIN_MESSAGE_MAP(CFUZZYApp, CWinApp)
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
    END_MESSAGE_MAP()


    // CFUZZYApp construction

    CFUZZYApp::CFUZZYApp()
    {
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
        int i;
        memset(&m_pid_para,0,sizeof(struct CCPidStruct ));
```

```
    LoadPidPara();
    channel=0;
    memset(&m_pid,0,sizeof(struct pid_struct)*32);
    for(i=0;i<32;i++)
    {
        temp_value[i]=0;
        setpt[i]=0;
    }
    memset(&m_fuzzy,      0, sizeof(struct MyFuzzy));
}


// The one and only CFUZZYApp object

CFUZZYApp theApp;


// CFUZZYApp initialization

BOOL CFUZZYApp::InitInstance()
{
    // InitCommonControls() is required on Windows XP if an application
    // manifest specifies use of ComCtl32.dll version 6 or later to enable
    // visual styles.  Otherwise, any window creation will fail.
    InitCommonControls();

    CWinApp::InitInstance();

    AfxEnableControlContainer();
```

```
    CFUZZYDlg dlg;
    m_pMainWnd = &dlg;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        //  dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        //  dismissed with Cancel
    }

    // Since the dialog has been closed, return FALSE so that we exit the
    //  application, rather than start the application's message pump.
    return FALSE;
}



void CFUZZYApp::fuzzy_step(long *crisp_inputs, long *crisp_outputs)
{//  进行模糊运算
    int         in_index, rule_index, out_index;
    long        in_val;
    long        dx;

    for(in_index = 0;in_index < num_inputs;in_index++)
    {// 求各输入的模糊等级隶属度
```

if(crisp_inputs[in_index]>Inputs[in_index].max)

{

crisp_inputs[in_index]=Inputs[in_index].max;

}

if(crisp_inputs[in_index]<INPUTS[IN_INDEX].MIN)

crisp_inputs[in_index]="Inputs[in_index].min;" fuzzify_input(in_index,crisp_inputs[in_index]); for(rule_index="0;rule_index" num_rules;rule_index++) 运 行 规 则 库 eval_rule(rule_index); for(out_index="0;out_index" num_outputs;out_index++) 模 糊 输 出 crisp_inputs); crisp_outputs[out_index]="dx;" 计 算 P="1+ Δ P" I="1+ Δ I" D="1+ Δ D" ="==" dx="defuzzify_output(out_index," m_fuzzy.crisp_scale[out_index]="dx;" void CFUZZYApp::fuzzify_input(int in_index,long 计 算 模 糊 输 入 的 隶 属 度 int i; for(i="0;i" num_input_mfs[in_index];i++) m_fuzzy.fuzzy_inputs[in_index][i]="get_membership_value(in_index,i,in_val);" } 求各输入的模糊等级隶属度 in_index: 输 入 类 型 : 0:e 1:ec mf_index: 模 糊 等 级 {NL, PL} in_val: 输 入 (e,ec) 值 inmem_points[in_index][mf_index][0] inmem_points[in_index][mf_index][1] inmem_points[in_index][mf_index][2] inmem_points[in_index][mf_index][3] : P2 P3 ****** ******* * ********** ********* ********** P4 CFUZZYApp::get_membership_value(int in_index,int mf_index,long in_val) { long dx,dy,dt; --如果输入值超出 模糊等级{NB, NM, NS, ZE, PS, PM, PB} 图形范围,则隶属度为零. < inmem_points[in_index][mf_index][0]) return 0; P1 if(in_val> inmem_points[in_index][mf_index][3]) return 0;//P4

if(in_val <= inmem_points[in_index][mf_index][1])

{//如果 输入值 < P2

if(inmem_points[in_index][mf_index][0] == inmem_points[in_index][mf_index][1])

{//如果 P1 = P2 梯形

return 1000;

}else

{// in_val 在 P1,P2 之间

// in_val - P1

//dt=---------- * 1000

// P2 - P1

```
                dy=in_val - inmem_points[in_index][mf_index][0];
                dy=dy*1000;
                dx=inmem_points[in_index][mf_index][1] - inmem_points[in_index][mf_index][0]; //
                dt=dy/dx;
                return    dt;
            }
        }
        if (in_val >= inmem_points[in_index][mf_index][2])
        {//如果  输入值  >= P3
            if(inmem_points[in_index][mf_index][2] == inmem_points[in_index][mf_index][3])
            {//如果  P3 = P4    梯形
                return 1000;
            }else
            {// in_val  在  P3,P4  之间
             //              P4 - in_val
             //dt=---------------------- * 1000
             //              P4    -    P3
                dy=inmem_points[in_index][mf_index][3] - in_val;
                dy=dy*1000;
                dx=inmem_points[in_index][mf_index][3] - inmem_points[in_index][mf_index][2];
                dt=dy/dx;
                return    dt;
            }
        }
        return 1000;
    }

    void CFUZZYApp::eval_rule(int rule_index)
    {// evaluator_rule    运行规则库
```

```
int         in_index,out_index,mf_index,ant_index,con_index;
int         val;
long      rule_strength = 1000;//隶属度设为 1.000

//找出该规则, 2 个输入  隶属度最小值
//如果该规则隶属度为 0,则该规则条件未满足
for(ant_index = 0;ant_index < num_rule_ants[rule_index];ant_index++)
{// 2 个输入(0:e      1:ec )中  找最小值
      val = Rules[rule_index].antecedent[ant_index];
      in_index = (val & 0x07);              //in_index:输入类型:    0:e      1:ec
      mf_index = ((val & 0x38) >> 3);//mf_index:模糊等级  {NL, NM, NS, ZE, PS, PM, PL}

      if(rule_strength>m_fuzzy.fuzzy_inputs[in_index][mf_index])
      {//找最小值
            rule_strength = m_fuzzy.fuzzy_inputs[in_index][mf_index];
      }
}

//如果该规则隶属度大于零,则 fuzzy_outputs[][]=该规则隶属度
m_fuzzy.rule_strengths[rule_index] = rule_strength;

for(con_index = 0;con_index < num_rule_cons[rule_index];con_index++)
{// 3 个输出
      val = Rules[rule_index].consequent[con_index];
      out_index = (val & 0x03);
      mf_index = ((val & 0x38) >> 3);
```

if(m_fuzzy.fuzzy_outputs[out_index][mf_index]<M_FUZZY.RULE_STRENGTHS[RULE_INDEX]) 模糊输出 int } * { long < 找最大值 CFUZZYApp::defuzzify_output(int out_index,long *inputs) temp1,temp2; mf_index,in_index; num_output_mfs[out_index]="7" crisp_outputs[]="//"

```
fuzzy_outputs[][0]*outmem_points[][0][0]+...+fuzzy_outputs[][6]*outmem_points[][6][0]
---------------------------------------------------------------------------------- fuzzy_outputs[][0]+ .. +fuzzy_outputs[][6]
for (mf_index="0;mf_index" num_output_mfs[out_index];mf_index++)
temp1="m_fuzzy.fuzzy_outputs[out_index][mf_index];" temp2="outmem_points[out_index][mf_index][0];"
summ="0;" temp1; product="0;" + (temp1 temp2);
m_fuzzy.fuzzy_outputs_image[out_index][mf_index]="m_fuzzy.fuzzy_outputs[out_index][mf_index];"
m_fuzzy.fuzzy_outputs[out_index][mf_index]="m_fuzzy.rule_strengths[rule_index];" if (summ> 0)
        {
            m_fuzzy.crisp_outputs[out_index] = product / summ;
            return m_fuzzy.crisp_outputs[out_index];
        }else
        {// 无输入规则    NO_RULES
            return m_fuzzy.crisp_outputs[out_index];
        }
    }


    //==位置型 PID 控制算法:计算 u(0)============================
    //
    // u(0) = Kp*e(k)    +    4/KI* Σe(i)    +    KD/4 [e(k) - e(k-1)]
    //
    //=========================================================
    /*void CFUZZYApp::PidDot(int n)
    {//
            long kp,ki,kd,work,de;
            struct pid_struct *p;
            long sv,pv;

            p=&m_pid[n];
```

```
sv=theApp.accel_value[n];//theApp.m_profile.B.m_SetValue[n];
pv=temp_value[n];
de=pv-sv;//当前误差 e    -100
//=====防止数据过大==================================
if(de>200)    de=200;
if(de<-200) de=-200;

p->e[2] = p->e[0];
p->e[1] = p->e[0];


p->e[0] =      de;//e0

m_fuzzy.crisp_inputs[0]=p->e[0];//当前误差 e
m_fuzzy.crisp_inputs[1]=0;


kp=m_pid_para.kp[n];//放大 1000 倍


work=kp*de;
work = work/10;

if(work>1000000)      work=1000000;
if(work<-1000000)    work=-1000000;


work-=m_pid_para.bias[n];

p->U0=work;
```

```
            p->this_Un=work;

            p->this_Output=-p->this_Un;

//          if(p->this_Output>=4000) p->this_Output=4000;
//          if(p->this_Output<0)        p->this_Output=0;

}
*/

//=============================================================================
//Δu(k)=u(k) -u(k-1)                                                          //
//          =Kp{ [e(k)-e(k-1)] + (4/KI)*e(k) + (KD/4)*[e(k)-2e(k-1)+e(k-2)] }//
//          =Kp{ E1 + (4/KI)*E2    +    KD/4*E3 }                             //
//=============================================================================
// E1 = [e(k)-e(k-1)]                                                         //
// E2 = e(k)                                                                  //
// E3 = [e(k)-2e(k-1)+e(k-2)]                                                 //
//=============================================================================
//误差 e :              当前测量值   -   目标值                               //
//误差变化率 ec:        本次误差     -   上次误差                             //
//=============================================================================
//PID 周期为 4 秒=4000ms                                                      //
//                                                                           //
//                                                                           //
//=============================================================================
/*void CFUZZYApp::PidRun(int n)
{//
      long kp,ki,kd,work,wtmp,sum,de,scale,kp0,ki0,kd0,integral;
      struct pid_struct *p;
```

```
long sv,pv;

p=&m_pid[n];

sv=setpt[n];
pv=temp_value[n];
integral = theApp.m_pid_para.integral[n];// 放大 10 倍//开始调节     放大 10 倍
sum=sv-pv;

if(sum>=0)
{
    if(sum>=integral)
    {//未到开始调节下限
        p->this_Output=4000;
        theApp.accel_value[n]=pv;
        p->this_Un=0;
        p->prev_Un=0;
        return;
    }
}else
{
    sum=-sum;
    if(sum>=integral)
    {// 超过开始调节上限
        p->this_Output=0;
        theApp.accel_value[n]=pv;
        p->this_Un=0;
        p->prev_Un=0;
        return;
    }
```

```
}

theApp.accel_value[n]+=theApp.m_pid_para.accel[n];//加速速率     放大 10 倍
if(theApp.accel_value[n]>sv)
{
      theApp.accel_value[n]=sv;
}

if(p->first==0)
{//计算 U0
            theApp.accel_value[n]=pv;
            PidDot(n);
            p->first=0xff;
}else
{
             p->prev_Un=p->this_Un;//保存上次 Un 记录
             p->e[2] = p->e[1];
             p->e[1] = p->e[0];

             sv=theApp.accel_value[n];//theApp.m_profile.B.m_SetValue[n];
             pv=temp_value[n];

             de=pv-sv;//当前误差 e
             //=====防止数据过大==================================
             if(de>200)    de=200;
             if(de<-200) de=-200;
             p->e[0] = de;
             p->E1     = p->e[0]- p->e[1];//当前误差变化率 ec
             p->E2     = p->e[0];
             p->E3     = p->e[0]- 2*p->e[1]+p->e[2];
```

```
                //====模糊推理==================================
                m_fuzzy.crisp_inputs[0]=p->e[0];//当前误差 e
                work=p->E1;   //当前误差变化率 ec   加权
                wtmp=m_fuzzy.crisp_inputs[1];
                sum=wtmp+work;
                sum=sum/2;
                m_fuzzy.crisp_inputs[1]=sum;
                fuzzy_step(m_fuzzy.crisp_inputs,m_fuzzy.crisp_outputs);

                //====PID 运算=============================================
                kp0=m_pid_para.kp[n];//放大 1000 000 倍        2000 * 1000
                ki0=m_pid_para.ki[n];//放大 10000 倍            1666 * 100
                kd0=m_pid_para.kd[n];//放大 10000 倍             1333 * 100
//              if(ki0>0)
//              {
//                    ki0=400/ki0;
//              }else   ki0=0;

                kp=m_fuzzy.crisp_scale[0]*kp0;
                ki=m_fuzzy.crisp_scale[1]*ki0;
                kd=m_fuzzy.crisp_scale[2]*kd0;

                kp=kp/1000; //放大 1000 倍
                ki=ki/1000; //放大 10 倍
                kd=kd/100;   //放大 10 倍

                //------dUn=Kp{ E1 + KI*E2 + KD*E3 }----
                work = kp*p->E1 + ki*p->E2 + kd*p->E3; //-1333
                work=work/10;
```

```
                //========对 this_Un 进行加权处理，提高抗干扰能力======================
                if(work>1000000)      work=1000000;
                if(work<-1000000)    work=-1000000;
                sum=work;
                //sum=work+p->dUn;
                //sum=sum/2;
                p->dUn=sum;

                p->this_Un=p->prev_Un+p->dUn;
                p->this_Output=-p->this_Un;

//              if(p->this_Output>=4000) p->this_Output=4000;
//              if(p->this_Output<0)         p->this_Output=0;

        }
}
*/




void CFUZZYApp::PidRun(int n)
{//
        long kp,ki,kd,work,wtmp,sum,de,scale,kp0,ki0,kd0,integral;
        struct pid_struct *p;
        long sv,pv;

        p=&m_pid[n];
```

```
sv=setpt[n];
pv=temp_value[n];
integral = theApp.m_pid_para.integral[n];// 放大 10 倍//开始调节     放大 10 倍
sum=sv-pv;

if(sum>=0)
{
    if(sum>=integral)
    {//未到开始调节下限
        p->this_Output=4000;
        theApp.accel_value[n]=pv;
        p->this_Un=0;
        p->prev_Un=0;
        return;
    }
}else
{
    sum=-sum;
    if(sum>=integral)
    {// 超过开始调节上限
        p->this_Output=0;
        theApp.accel_value[n]=pv;
        p->this_Un=0;
        p->prev_Un=0;
        return;
    }
}

if(p->first==0)
{//计算 U0
```

```
            theApp.accel_value[n]=pv;
            p->first=0xff;
            p->integral=0;
            m_fuzzy.crisp_inputs[1]=0;
}


theApp.accel_value[n]+=theApp.m_pid_para.accel[n];//加速速率     放大 10 倍
if(theApp.accel_value[n]>sv)
{
      theApp.accel_value[n]=sv;
}

sv=theApp.accel_value[n];//theApp.m_profile.B.m_SetValue[n];

de=pv-sv;//当前误差 e
//=====防止数据过大==================================
if(de>200)    de=200;
if(de<-200) de=-200;
p->e[1] = p->e[0];
p->e[2] = p->e[1];
p->e[0] = de;
p->E1     = p->e[0]- p->e[1];//当前误差变化率 ec


//====模糊推理==================================
m_fuzzy.crisp_inputs[0]=p->e[0];    //当前误差 e
m_fuzzy.crisp_inputs[1]=p->E1;       //当前误差变化率 ec
fuzzy_step(m_fuzzy.crisp_inputs,m_fuzzy.crisp_outputs);
```

```
//====PID 运算==============================================
kp0=m_pid_para.kp[n];//放大 1000 000 倍          2000 * 1000
ki0=m_pid_para.ki[n];//放大 10000 倍              1666 * 100
kd0=m_pid_para.kd[n];//放大 10000 倍               1333 * 100

kp=m_fuzzy.crisp_scale[0]*kp0;
ki=m_fuzzy.crisp_scale[1]*ki0;
kd=m_fuzzy.crisp_scale[2]*kd0;

kp=kp/1000; //放大 1000 倍
ki=ki/1000; //放大 10 倍
kd=kd/100;    //放大 10 倍


//=======限制积分项=======================
p->integral=p->integral+de;
if(p->integral>2000)    p->integral = 2000;
if(p->integral<-2000) p->integral =-2000;

work = kp*de + ki*p->integral+    kd*(p->e[1]-p->e[2]);

//p->test_kd= -(kd*(p->e[1]-p->e[2]))/10;

work=work/10;
work-=m_pid_para.bias[n];
work=-work;

//========对 this_Un 进行加权处理，提高抗干扰能力=======================
if(work>4000)      work=4000;
```

```
    if(work<-4000)    work=-4000;
    p->this_Output=work;
}


void CFUZZYApp::SavePidPara()
{//
     FILE *fp;
     CString filename;

     filename="pid.dat";
     fp=fopen(filename,"wb");
     fwrite((struct CCPidStruct    *)&(m_pid_para),sizeof(struct CCPidStruct),1,fp);
     fclose(fp);
}


void CFUZZYApp::LoadPidPara()
{//
    FILE *fp;
    CString filename;
    filename="pid.dat";

    fp=fopen(filename,"rb");
    if(fp==NULL)
    {
        SavePidPara();
    }else
    {
        fread((struct CCPidStruct    *)&(m_pid_para),sizeof(struct CCPidStruct),1,fp);
```

```
            fclose(fp);
        }
    }

    // FUZZY.cpp : Defines the class behaviors for the application.
    //

    #include "stdafx.h"
    #include "FUZZY.h"
    #include "FUZZYDlg.h"
    #include "classext.h"

    #ifdef _DEBUG
    #define new DEBUG_NEW
    #endif

    //==========  增  量  型  PID  控  制  算  法
============================================================

    short   temp_value[32];
    short   setpt[32];
    //==========2D-3D===  模  糊  控  制  算  法
=======
{NL,NM,NS,ZE,PS,PM,PL}=========================

    struct    MyFuzzy   m_fuzzy;

    // 误差 e :          当前测量值 - 目标值
    // 误差变化率 ec:    本次误差   - 上次误差
    int           num_inputs  = 2;  //2D
    int           num_outputs = 3;  //3D
    int           num_rules   = 49; //模糊控制规则
```

```
//===========初始化============================
int    num_input_mfs[2] = { 7, 7 };//模糊子集

struct      In     Inputs[2] =
{
      { -50, 50 },// min, max
      { -10, 10 } // min, max
};

long inmem_points[2][7][4] = //    [e,ec][模糊子集][P1,P2,P3,P4]
{
      {
            { -50, -50, -30, -20 },
            { -30, -20, -20, -10 },
            { -20, -10, -10,      0 },
            { -10,      0,      0,    10 },
            {      0,    10,    10,    20 },
            {    10,    20,    20,    30 },
            {    20,    30,    50,    50 }
      },
      {
            { -10, -10, -6, -4 },
            {    -6, -4,    -4, -2 },
            {    -4, -2,    -2,    0 },
            {    -2,    0,      0,    2 },
            {      0,    2,      2,    4 },
            {      2,    4,      4,    6 },
            {      4,    6,    10, 10 }
      }
```

```
};
int    num_output_mfs[3] = { 7, 7, 7 };//输出  ΔKp/ΔKI/ΔKD

struct      Out  Outputs[] =
{
      { -1200, 1200 },// MIN MAX
      { -1200, 1200 },// MIN MAX
      { -1200, 1200 } // MIN MAX
};

long outmem_points[3][7][4] = //[ΔKp, ΔKI, ΔKD][模糊子集][P1]
{
      {
            { -1200 },//ΔKp
            {   -800 },
            {   -400 },
            {         0 },
            {     400 },
            {     800 },
            {   1200 }
      },    {
            { -1200 },//ΔKI
            {   -800 },
            {   -400 },
            {         0 },
            {     400 },
            {     800 },
            {   1200 }
      },    {
            { -1200 },//ΔKD
```

```
            {   -800 },
            {   -400 },
            {      0 },
            {    400 },
            {    800 },
            {   1200 }
        }
    };


    //==========2D-3D=====================================================
    int   num_rule_ants[49] = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2};//2D
    int   num_rule_cons[49] = { 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3};//3D

    struct      Rule Rules[49] =
    {
        { { 0x00, 0x01 }, { 0xb0, 0xa9, 0xa2 } },
        { { 0x08, 0x01 }, { 0xb0, 0xb1, 0xa2 } },
        { { 0x10, 0x01 }, { 0xa8, 0xb1, 0x9a } },
        { { 0x18, 0x01 }, { 0xa8, 0xb1, 0x9a } },
        { { 0x20, 0x01 }, { 0xa0, 0xb1, 0x9a } },
        { { 0x28, 0x01 }, { 0x98, 0xb1, 0xa2 } },
        { { 0x30, 0x01 }, { 0x98, 0xa9, 0xa2 } },
        { { 0x00, 0x09 }, { 0xb0, 0xb1, 0x82 } },
        { { 0x08, 0x09 }, { 0xb0, 0xa1, 0x82 } },
        { { 0x10, 0x09 }, { 0xa8, 0xa1, 0x8a } },
        { { 0x18, 0x09 }, { 0xa0, 0xa1, 0xa2 } },
        { { 0x20, 0x09 }, { 0xa0, 0xa1, 0xa2 } },
```

{ { 0x28, 0x09 }, { 0x98, 0xa1, 0x8a } },

{ { 0x30, 0x09 }, { 0x98, 0xa1, 0x8a } },

{ { 0x00, 0x11 }, { 0xb0, 0x99, 0x82 } },

{ { 0x08, 0x11 }, { 0xb0, 0x99, 0x82 } },

{ { 0x10, 0x11 }, { 0xa8, 0x99, 0x8a } },

{ { 0x18, 0x11 }, { 0x98, 0x99, 0x92 } },

{ { 0x20, 0x11 }, { 0x98, 0x99, 0x8a } },

{ { 0x28, 0x11 }, { 0x90, 0x91, 0x8a } },

{ { 0x30, 0x11 }, { 0x90, 0x91, 0x8a } },

{ { 0x00, 0x19 }, { 0xb0, 0x91, 0x92 } },

{ { 0x08, 0x19 }, { 0xa8, 0x91, 0x92 } },

{ { 0x10, 0x19 }, { 0xa0, 0x91, 0x92 } },

{ { 0x18, 0x19 }, { 0x90, 0x91, 0x92 } },

{ { 0x20, 0x19 }, { 0x88, 0x91, 0x92 } },

{ { 0x28, 0x19 }, { 0x88, 0x91, 0x9a } },

{ { 0x30, 0x19 }, { 0x88, 0x99, 0x8a } },

{ { 0x00, 0x21 }, { 0xa8, 0x99, 0x82 } },

{ { 0x08, 0x21 }, { 0xa0, 0x99, 0x82 } },

{ { 0x10, 0x21 }, { 0x98, 0x99, 0x8a } },

{ { 0x18, 0x21 }, { 0x88, 0x99, 0x92 } },

{ { 0x20, 0x21 }, { 0x88, 0x99, 0x8a } },

{ { 0x28, 0x21 }, { 0x88, 0x91, 0x9a } },

{ { 0x30, 0x21 }, { 0x88, 0x99, 0x8a } },

{ { 0x00, 0x29 }, { 0x98, 0xb1, 0x82 } },

{ { 0x08, 0x29 }, { 0x98, 0xa1, 0x82 } },

{ { 0x10, 0x29 }, { 0x90, 0xb1, 0x9a } },

{ { 0x18, 0x29 }, { 0x88, 0xa1, 0x92 } },

{ { 0x20, 0x29 }, { 0x80, 0xa1, 0x8a } },

{ { 0x28, 0x29 }, { 0x88, 0x99, 0x9a } },

{ { 0x30, 0x29 }, { 0x88, 0xa1, 0x82 } },

```
        { { 0x00, 0x31 }, { 0x98, 0xa9, 0xa2 } },
        { { 0x08, 0x31 }, { 0x98, 0xa9, 0xa2 } },
        { { 0x10, 0x31 }, { 0x90, 0xb1, 0x9a } },
        { { 0x18, 0x31 }, { 0x88, 0xb1, 0x9a } },
        { { 0x20, 0x31 }, { 0x88, 0xb1, 0x9a } },
        { { 0x28, 0x31 }, { 0x88, 0xa1, 0xa2 } },
        { { 0x30, 0x31 }, { 0x88, 0xa1, 0x82 } }
    };
    //================================================================================
==============
```

```
    // CFUZZYApp

    BEGIN_MESSAGE_MAP(CFUZZYApp, CWinApp)
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
    END_MESSAGE_MAP()


    // CFUZZYApp construction

    CFUZZYApp::CFUZZYApp()
    {
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
        int i;
```

```
    memset(&m_pid_para,0,sizeof(struct CCPidStruct ));
    LoadPidPara();
    channel=0;
    memset(&m_pid,0,sizeof(struct pid_struct)*32);
    for(i=0;i<32;i++)
    {
        temp_value[i]=0;
        setpt[i]=0;
    }
    memset(&m_fuzzy,      0, sizeof(struct MyFuzzy));
}


// The one and only CFUZZYApp object

CFUZZYApp theApp;


// CFUZZYApp initialization

BOOL CFUZZYApp::InitInstance()
{
    // InitCommonControls() is required on Windows XP if an application
    // manifest specifies use of ComCtl32.dll version 6 or later to enable
    // visual styles.  Otherwise, any window creation will fail.
    InitCommonControls();

    CWinApp::InitInstance();

    AfxEnableControlContainer();
```

```
    CFUZZYDlg dlg;
    m_pMainWnd = &dlg;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        //  dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        //  dismissed with Cancel
    }

    // Since the dialog has been closed, return FALSE so that we exit the
    //  application, rather than start the application's message pump.
    return FALSE;
}



void CFUZZYApp::fuzzy_step(long *crisp_inputs, long *crisp_outputs)
{//  进行模糊运算
    int         in_index, rule_index, out_index;
    long        in_val;
    long        dx;

    for(in_index = 0;in_index < num_inputs;in_index++)
```

```
{// 求各输入的模糊等级隶属度
    if(crisp_inputs[in_index]>Inputs[in_index].max)
    {
        crisp_inputs[in_index]=Inputs[in_index].max;
    }
    if(crisp_inputs[in_index] inmem_points[in_index][mf_index][3]) return 0;//P4

if(in_val <= inmem_points[in_index][mf_index][1])
{//如果 输入值 < P2
    if(inmem_points[in_index][mf_index][0] == inmem_points[in_index][mf_index][1])
    {//如果 P1 = P2  梯形
        return 1000;
    }else
    {// in_val 在 P1,P2 之间
     //          in_val - P1
     //dt=---------------------- * 1000
     //             P2   -   P1

        dy=in_val - inmem_points[in_index][mf_index][0];
        dy=dy*1000;
        dx=inmem_points[in_index][mf_index][1] - inmem_points[in_index][mf_index][0]; //
        dt=dy/dx;
        return   dt;
    }
}
if (in_val >= inmem_points[in_index][mf_index][2])
{//如果 输入值 >= P3
    if(inmem_points[in_index][mf_index][2] == inmem_points[in_index][mf_index][3])
    {//如果 P3 = P4  梯形
        return 1000;
```

```
            }else
            {// in_val 在 P3,P4 之间
             //              P4 - in_val
             //dt=---------------------- * 1000
             //              P4    -    P3
                dy=inmem_points[in_index][mf_index][3] - in_val;
                dy=dy*1000;
                dx=inmem_points[in_index][mf_index][3] - inmem_points[in_index][mf_index][2];
                dt=dy/dx;
                return    dt;
            }
        }
        return 1000;
}

void CFUZZYApp::eval_rule(int rule_index)
{// evaluator_rule   运行规则库
        int         in_index,out_index,mf_index,ant_index,con_index;
        int         val;
        long      rule_strength = 1000;//隶属度设为 1.000

        //找出该规则, 2 个输入  隶属度最小值
        //如果该规则隶属度为 0,则该规则条件未满足
        for(ant_index = 0;ant_index < num_rule_ants[rule_index];ant_index++)
        {// 2 个输入(0:e      1:ec )中  找最小值
            val = Rules[rule_index].antecedent[ant_index];
            in_index = (val & 0x07);              //in_index:输入类型:    0:e      1:ec
            mf_index = ((val & 0x38) >> 3);//mf_index:模糊等级  {NL, NM, NS, ZE, PS, PM, PL}

            if(rule_strength>m_fuzzy.fuzzy_inputs[in_index][mf_index])
```

```
            {//找最小值
                rule_strength = m_fuzzy.fuzzy_inputs[in_index][mf_index];
            }
        }

        //如果该规则隶属度大于零,则 fuzzy_outputs[][]=该规则隶属度
        m_fuzzy.rule_strengths[rule_index] = rule_strength;

        for(con_index = 0;con_index < num_rule_cons[rule_index];con_index++)
        {// 3 个输出
            val = Rules[rule_index].consequent[con_index];
            out_index = (val & 0x03);
            mf_index = ((val & 0x38) >> 3);
            if(m_fuzzy.fuzzy_outputs[out_index][mf_index] 0)
        {
            m_fuzzy.crisp_outputs[out_index] = product / summ;
            return m_fuzzy.crisp_outputs[out_index];
        }else
        {// 无输入规则   NO_RULES
            return m_fuzzy.crisp_outputs[out_index];
        }
    }


    //==位置型 PID 控制算法:计算 u(0)==========================
    //
    // u(0) = Kp*e(k)   +   4/KI* Σe(i)   +   KD/4 [e(k) - e(k-1)]
    //
    //=========================================================
    /*void CFUZZYApp::PidDot(int n)
```

```
{//
        long kp,ki,kd,work,de;
        struct pid_struct *p;
        long sv,pv;

        p=&m_pid[n];

        sv=theApp.accel_value[n];//theApp.m_profile.B.m_SetValue[n];
        pv=temp_value[n];
        de=pv-sv;//当前误差 e    -100
        //=====防止数据过大==================================
        if(de>200)    de=200;
        if(de<-200) de=-200;

        p->e[2] = p->e[0];
        p->e[1] = p->e[0];


        p->e[0] =      de;//e0

        m_fuzzy.crisp_inputs[0]=p->e[0];//当前误差 e
        m_fuzzy.crisp_inputs[1]=0;


        kp=m_pid_para.kp[n];//放大 1000 倍


        work=kp*de;
        work = work/10;
```

```
            if(work>1000000)      work=1000000;
            if(work<-1000000)    work=-1000000;


            work-=m_pid_para.bias[n];

            p->U0=work;
            p->this_Un=work;

            p->this_Output=-p->this_Un;

//          if(p->this_Output>=4000) p->this_Output=4000;
//          if(p->this_Output<0)        p->this_Output=0;

}
*/

//=============================================================================
//Δu(k)=u(k) -u(k-1)                                                          //
//          =Kp{ [e(k)-e(k-1)] + (4/KI)*e(k) + (KD/4)*[e(k)-2e(k-1)+e(k-2)] }//
//          =Kp{ E1 + (4/KI)*E2    +    KD/4*E3 }                             //
//=============================================================================
// E1 = [e(k)-e(k-1)]                                                         //
// E2 = e(k)                                                                  //
// E3 = [e(k)-2e(k-1)+e(k-2)]                                                 //
//=============================================================================
//误差 e :              当前测量值    -    目标值                             //
//误差变化率 ec:        本次误差      -    上次误差                           //
//=============================================================================
//PID 周期为 4 秒=4000ms                                                      //
```

```
//                                                                              //
//                                                                              //
//==============================================================================
/*void CFUZZYApp::PidRun(int n)
{//
    long kp,ki,kd,work,wtmp,sum,de,scale,kp0,ki0,kd0,integral;
    struct pid_struct *p;
    long sv,pv;

    p=&m_pid[n];

    sv=setpt[n];
    pv=temp_value[n];
    integral = theApp.m_pid_para.integral[n];//  放大 10 倍//开始调节     放大 10 倍
    sum=sv-pv;

    if(sum>=0)
    {
        if(sum>=integral)
        {//未到开始调节下限
            p->this_Output=4000;
            theApp.accel_value[n]=pv;
            p->this_Un=0;
            p->prev_Un=0;
            return;
        }
    }else
    {
        sum=-sum;
        if(sum>=integral)
```

```
        {// 超过开始调节上限
            p->this_Output=0;
            theApp.accel_value[n]=pv;
            p->this_Un=0;
            p->prev_Un=0;
            return;
        }
    }

    theApp.accel_value[n]+=theApp.m_pid_para.accel[n];//加速速率     放大 10 倍
    if(theApp.accel_value[n]>sv)
    {
        theApp.accel_value[n]=sv;
    }

    if(p->first==0)
    {//计算 U0
            theApp.accel_value[n]=pv;
            PidDot(n);
            p->first=0xff;
    }else
    {
             p->prev_Un=p->this_Un;//保存上次 Un 记录
             p->e[2] = p->e[1];
             p->e[1] = p->e[0];

             sv=theApp.accel_value[n];//theApp.m_profile.B.m_SetValue[n];
             pv=temp_value[n];

             de=pv-sv;//当前误差 e
```

```
//=====防止数据过大==================================
if(de>200)    de=200;
if(de<-200) de=-200;
p->e[0] = de;
p->E1      = p->e[0]- p->e[1];//当前误差变化率 ec
p->E2      = p->e[0];
p->E3      = p->e[0]- 2*p->e[1]+p->e[2];

//====模糊推理==================================
m_fuzzy.crisp_inputs[0]=p->e[0];//当前误差 e
work=p->E1;   //当前误差变化率 ec   加权
wtmp=m_fuzzy.crisp_inputs[1];
sum=wtmp+work;
sum=sum/2;
m_fuzzy.crisp_inputs[1]=sum;
fuzzy_step(m_fuzzy.crisp_inputs,m_fuzzy.crisp_outputs);

//====PID 运算==========================================
kp0=m_pid_para.kp[n];//放大 1000 000 倍         2000 * 1000
ki0=m_pid_para.ki[n];//放大 10000 倍             1666 * 100
kd0=m_pid_para.kd[n];//放大 10000 倍              1333 * 100
//              if(ki0>0)
//              {
//                    ki0=400/ki0;
//              }else    ki0=0;

kp=m_fuzzy.crisp_scale[0]*kp0;
ki=m_fuzzy.crisp_scale[1]*ki0;
kd=m_fuzzy.crisp_scale[2]*kd0;
```

```
                kp=kp/1000; //放大 1000 倍
                ki=ki/1000;    //放大 10 倍
                kd=kd/100;    //放大 10 倍

                //------dUn=Kp{ E1 + KI*E2 + KD*E3 }----
                work = kp*p->E1 + ki*p->E2 + kd*p->E3; //-1333
                work=work/10;

                //========对 this_Un 进行加权处理，提高抗干扰能力======================
                if(work>1000000)     work=1000000;
                if(work<-1000000)    work=-1000000;
                sum=work;
                //sum=work+p->dUn;
                //sum=sum/2;
                p->dUn=sum;

                p->this_Un=p->prev_Un+p->dUn;
                p->this_Output=-p->this_Un;

//              if(p->this_Output>=4000) p->this_Output=4000;
//              if(p->this_Output<0)        p->this_Output=0;

        }
}
*/

void CFUZZYApp::PidRun(int n)
```

```
{//
    long kp,ki,kd,work,wtmp,sum,de,scale,kp0,ki0,kd0,integral;
    struct pid_struct *p;
    long sv,pv;

    p=&m_pid[n];

    sv=setpt[n];
    pv=temp_value[n];
    integral = theApp.m_pid_para.integral[n];//  放大 10 倍//开始调节     放大 10 倍
    sum=sv-pv;

    if(sum>=0)
    {
        if(sum>=integral)
        {//未到开始调节下限
            p->this_Output=4000;
            theApp.accel_value[n]=pv;
            p->this_Un=0;
            p->prev_Un=0;
            return;
        }
    }else
    {
        sum=-sum;
        if(sum>=integral)
        {// 超过开始调节上限
            p->this_Output=0;
            theApp.accel_value[n]=pv;
            p->this_Un=0;
```

```
            p->prev_Un=0;
            return;
        }
    }

    if(p->first==0)
    {//计算 U0
            theApp.accel_value[n]=pv;
            p->first=0xff;
            p->integral=0;
            m_fuzzy.crisp_inputs[1]=0;
    }


    theApp.accel_value[n]+=theApp.m_pid_para.accel[n];//加速速率     放大 10 倍
    if(theApp.accel_value[n]>sv)
    {
        theApp.accel_value[n]=sv;
    }

    sv=theApp.accel_value[n];//theApp.m_profile.B.m_SetValue[n];

    de=pv-sv;//当前误差 e
    //=====防止数据过大==================================
    if(de>200)    de=200;
    if(de<-200) de=-200;
    p->e[1] = p->e[0];
    p->e[2] = p->e[1];
    p->e[0] = de;
    p->E1     = p->e[0]- p->e[1];//当前误差变化率 ec
```

```
//====模糊推理==================================
m_fuzzy.crisp_inputs[0]=p->e[0];    //当前误差 e
m_fuzzy.crisp_inputs[1]=p->E1;       //当前误差变化率 ec
fuzzy_step(m_fuzzy.crisp_inputs,m_fuzzy.crisp_outputs);


//====PID 运算=============================================
kp0=m_pid_para.kp[n];//放大 1000 000 倍          2000 * 1000
ki0=m_pid_para.ki[n];//放大 10000 倍              1666 * 100
kd0=m_pid_para.kd[n];//放大 10000 倍              1333 * 100

kp=m_fuzzy.crisp_scale[0]*kp0;
ki=m_fuzzy.crisp_scale[1]*ki0;
kd=m_fuzzy.crisp_scale[2]*kd0;

kp=kp/1000; //放大 1000 倍
ki=ki/1000;     //放大 10 倍
kd=kd/100;     //放大 10 倍


//=======限制积分项=======================
p->integral=p->integral+de;
if(p->integral>2000)    p->integral = 2000;
if(p->integral<-2000) p->integral =-2000;

work = kp*de + ki*p->integral+    kd*(p->e[1]-p->e[2]);

//p->test_kd= -(kd*(p->e[1]-p->e[2]))/10;
```

```
	work=work/10;
	work-=m_pid_para.bias[n];
	work=-work;

	//========对 this_Un 进行加权处理，提高抗干扰能力======================
	if(work>4000)	work=4000;
	if(work<-4000)	work=-4000;
	p->this_Output=work;
}


void CFUZZYApp::SavePidPara()
{//
	 FILE *fp;
	 CString filename;

	 filename="pid.dat";
	 fp=fopen(filename,"wb");
	 fwrite((struct CCPidStruct	*)&(m_pid_para),sizeof(struct CCPidStruct),1,fp);
	 fclose(fp);
}


void CFUZZYApp::LoadPidPara()
{//
	FILE *fp;
	CString filename;
	filename="pid.dat";
```

```
        fp=fopen(filename,"rb");
        if(fp==NULL)
        {
            SavePidPara();
        }else
        {
            fread((struct CCPidStruct    *)&(m_pid_para),sizeof(struct CCPidStruct),1,fp);
            fclose(fp);
        }
    }
```