PID 控制算法的 C 语言实现一 PID 算法原理

1.PID 常用口诀:

参数整定找最佳，从小到大顺序查

先是比例后积分，最后再把微分加

曲线振荡很频繁，比例度盘要放大

曲线漂浮绕大湾，比例度盘往小扳

曲线偏离回复慢，积分时间往下降

曲线波动周期长，积分时间再加长

曲线振荡频率快，先把微分降下来

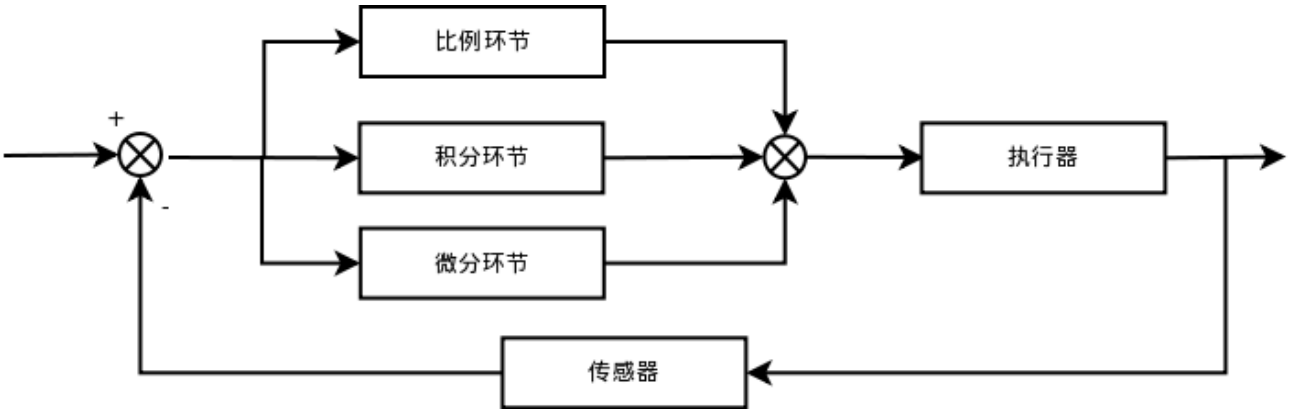
动差大来波动慢。微分时间应加长

理想曲线两个波，前高后低 4 比 1

一看二调多分析，调节质量不会低

最近两天在考虑一般控制算法的 C 语言实现问题，发现网络上尚没有一套完整的比较体系的讲解。于是总结了几天，整理一套思路分享给大家。

在工业应用中 PID 及其衍生算法是应用最广泛的算法之一，是当之无愧的万能算法，如果能够熟练掌握 PID 算法的设计与实现过程，对于一般的研发人员来讲，应该是足够应对一般研发问题了，而难能可贵的是，在我所接触的控制算法当中，PID 控制算法又是最简单，最能体现反馈思想的控制算法，可谓经典中的经典。经典的未必是复杂的，经典的东西常常是简单的，而且是最简单的，想想牛顿的力学三大定律吧，想想爱因斯坦的质能方程吧，何等的简单！简单的不是原始的，简单的也不是落后的，简单到了美的程度。先看看 PID 算法的一般形式：



PID 的流程简单到了不能再简单的程度，通过误差信号控制被控量，而控制器本身就是比例、积分、微分三个环节的加和。这里我们规定（在 t 时刻）：

1. 输入量为 rin(t)；

2. 输出量为 rout(t)；

3. 偏差量为 err(t)=rin(t)-rout(t)；

pid 的控制规律为

$$u(x)=kp(err(t)+\frac{1}{T}\cdot\int err(t)dt+\frac{T_D derr(t)}{dt})$$

理解一下这个公式，主要从下面几个问题着手，为了便于理解，把控制环境具体一下：

1. 规定这个流程是用来为直流电机调速的；

2. 输入量 rin(t)为电机转速预定值；

3. 输出量 rout(t)为电机转速实际值；

4. 执行器为直流电机；

5. 传感器为光电码盘，假设码盘为 10 线；

6. 直流电机采用 PWM 调速 转速用单位 转/min 表示；

不难看出以下结论：

1. 输入量 rin（t）为电机转速预定值（转/min）；

2. 输出量 rout(t)为电机转速实际值（转/min）；

3.偏差量为预定值和实际值之差（转/min）；

那么以下几个问题需要弄清楚：

1.通过 PID 环节之后的 U(t)是什么值呢？

2.控制执行器（直流电机）转动转速应该为电压值（也就是 PWM 占空比）。

3.那么 U(t)与 PWM 之间存在怎样的联系呢？
http://blog.21ic.com/user1/3407/archives/2006/33541.html（见附录 1）这篇文章上给出了一种方法，即，每个电压对应一个转速，电压和转速之间呈现线性关系。但是我考虑这种方法的前提是把直流电机的特性理解为线性了，而实际情况下，直流电机的特性绝对不是线性的，或者说在局部上是趋于线性的，这就是为什么说 PID 调速有个范围的问题。具体看一下
http://articles.e-works.net.cn/component/article90249.htm（见附录 2）这篇文章就可以了解了。所以在正式进行调速设计之前，需要现有开环系统，测试电机和转速之间的特性曲线（或者查阅电机的资料说明），然后再进行闭环参数整定。这篇先写到这，下一篇说明连续系统的离散化问题。并根据离散化后的特点讲述位置型 PID 和增量型 PID 的用法和 C 语言实现过程。

PID 二　　PID 算法的离散化

上一节中，我论述了 PID 算法的基本形式，并对其控制过程的实现有了一个简要的说明，通过上一节的总结，基本已经可以明白 PID 控制的过程。这一节中先继续上一节内容补充说明一下。

1.说明一下反馈控制的原理，通过上一节的框图不难看出，PID 控制其实是对偏差的控制过程；

2.如果偏差为 0,则比例环节不起作用，只有存在偏差时，比例环节才起作用。

3.积分环节主要是用来消除静差，所谓静差，就是系统稳定后输出值和设定值之间的差值，积分环节实际上就是偏差累计的过程，把累计的误差加到原有系统上以抵消系统造成的静差。

4.而微分信号则反应了偏差信号的变化规律，或者说是变化趋势，根据偏差信号的变化趋势来进行超前调节，从而增加了系统的快速性。

好了，关于 PID 的基本说明就补充到这里，下面将对 PID 连续系统离散化，从而方便在处理器上实现。下面把连续状态的公式再贴一下：

$$u(x)=kp(err(t)+\frac{1}{T}\cdot\int err(t)dt+\frac{T_D derr(t)}{dt})$$

假设采样间隔为 T，则在第 K T 时刻：

偏差 err(K)=rin(K)-rout(K)；

积分环节用加和的形式表示，即 err(K)+err(K+1)+……；

微分环节用斜率的形式表示，即[err(K)-err(K-1)]/T;

从而形成如下 PID 离散表示形式：

$$u(k)=K_p(err(k)+\frac{T}{T_i}\sum err(j)+\frac{T_D}{T}(err(k)-err(k-1)))$$

则 u(K)可表示成为：

$$u(k)=K_p(err(k)+K_i\sum err(j)+K_d(err(k)-err(k-1)))$$

至于说 Kp、Ki、Kd 三个参数的具体表达式，我想可以轻松的推出了，这里节省时间，不再详细表示了。

其实到这里为止，PID 的基本离散表示形式已经出来了。目前的这种表述形式属于位置型 PID，另外一种表述方式为增量式 PID，由 U 上述表达式可以轻易得到：

$$u(k-1)=K_p(err(k-1)+K_i\sum err(j)+K_d(err(k-1)-err(k-2)))$$

那么：

$$\Delta u(k)=k_p(error(k)-error(k-1))+k_i error(k)+k_d(error(k)-2error(k-1)+error(k-2))$$

这就是离散化 PID 的增量式表示方式，由公式可以看出，增量式的表达结果和最近三次的偏差有关，这样就大大提高了系统的稳定性。需要注意的是最终的输出结果应该为

u(K)+增量调节值;

PID 的离散化过程基本思路就是这样，下面是将离散化的公式转换成为 C 语言，从而实现微控制器的控制作用。

PID 三　　位置型 PID 的 C 语言实现

上一节中已经抽象出了位置性 PID 和增量型 PID 的数学表达式，这一节，重点讲解 C 语言代码的实现过程，算法的 C 语言实现过程具有一般性，通过 PID 算法的 C 语言实现，可以以此类推，设计其它算法的 C 语言实现。

第一步：定义 PID 变量结构体，代码如下：

```
struct _pid

{
    float SetSpeed;                 //定义设定值
    float ActualSpeed;              //定义实际值
```

```
        float err;                              //定义偏差值
        float err_last;                         //定义上一个偏差值
        float Kp,Ki,Kd;                         //定义比例、积分、微分系数
        float voltage;                          //定义电压值（控制执行器的
变量）
        float integral;                         //定义积分值
}pid;
```

控制算法中所需要用到的参数在一个结构体中统一定义，方便后面的使用。

第二部：初始化变量，代码如下：

```
void PID_init(){
        printf("PID_init begin \n");
        pid.SetSpeed=0.0;
        pid.ActualSpeed=0.0;
        pid.err=0.0;
        pid.err_last=0.0;
        pid.voltage=0.0;
        pid.integral=0.0;
        pid.Kp=0.2;
        pid.Ki=0.015;
        pid.Kd=0.2;
        printf("PID_init end \n");
}
```

统一初始化变量，尤其是 Kp,Ki,Kd 三个参数，调试过程当中，对于要求的控制效果，可以通过调节这三个量直接进行调节。

第三步：编写控制算法，代码如下：

```
float PID_realize(float speed)

{
        pid.SetSpeed=speed;
        pid.err=pid.SetSpeed-pid.ActualSpeed;
        pid.integral+=pid.err;
        pid.voltage=pid.Kp*pid.err+pid.Ki*pid.integral+pid.Kd*(pid.err
-pid.err_last);
        pid.err_last=pid.err;
        pid.ActualSpeed=pid.voltage*1.0;
        return pid.ActualSpeed;
}
```

注意：这里用了最基本的算法实现形式，没有考虑死区问题，没有设定上下限，只是对公式的一种直接的实现，后面的介绍当中还会逐渐的对此改进。

到此为止，PID 的基本实现部分就初步完成了。下面是测试代码：

```
int main(){
        printf("System begin \n");
        PID_init();
        int count=0;
        while(count<1000)
        {
                float speed=PID_realize(200.0);
                printf("%f\n",speed);
                count++;
        }
return 0;
}
```

下面是经过 1000 次的调节后输出的 1000 个数据（具体的参数整定过程就不说明了，网上这种说明非常多）：

83.000001	78.301526	110.009898	133.460162	150.799612	163.620593	173.100594
11.555000	79.812136	111.134811	134.291942	151.414626	164.075347	173.436838
59.559675	81.321929	112.245652	135.113308	152.021959	164.524422	173.768895
28.175408	82.800304	113.342615	135.924419	152.621696	164.967877	174.096796
52.907421	84.268909	114.425860	136.725382	153.213951	165.405795	174.420594
38.944152	85.713108	115.495564	137.516332	153.798781	165.838235	174.740352
51.891699	87.143455	116.551897	138.297401	154.376315	166.265257	175.056096
46.141651	88.553005	117.595029	139.068697	154.946626	166.686967	175.367915
53.339054	89.946960	118.625116	139.830352	155.509812	167.103377	175.675818
51.509998	91.322078	119.642331	140.582499	156.065958	167.514610	175.979886
55.908450	92.680996	120.646826	141.325237	156.615146	167.920681	176.280136
55.944631	94.022234	121.638767	142.058701	157.157471	168.321682	176.576656
58.970680	95.347186	122.618307	142.782985	157.693012	168.717670	176.869444
59.882936	96.655242	123.585603	143.498218	158.221871	169.108719	177.158600
62.225001	97.947180	124.540813	144.204509	158.744097	169.494862	177.444121
63.537254	99.222808	125.484079	144.901969	159.259826	169.876198	177.726087
65.527707	100.482601	126.415549	145.590726	159.769078	170.252740	178.004510
67.011058	101.726572	127.335383	146.270843	160.271991	170.624605	178.279458
68.810646	102.955049	128.243715	146.942486	160.768588	170.991799	178.550967
70.355318	104.168125	129.140691	147.605718	161.258996	171.354406	178.819094
72.042040	105.366066	130.026459	148.260674	161.743264	171.712487	179.083860
73.595658	106.549019	130.901149	148.907425	162.221494	172.066080	179.345315
75.207620	107.717187	131.764909	149.546109	162.693737	172.415265	179.603504
76.745444	108.870756	132.617870	150.176794	163.160075	172.760077	179.858466

180.110241	188.564488	193.425214	196.219859	197.826629	198.750448	199.281571
180.358866	188.707429	193.507408	196.267115	197.853799	198.766067	199.290541
180.604388	188.848592	193.588568	196.313778	197.880631	198.781497	199.299421
180.846849	188.987995	193.668715	196.359851	197.907131	198.796736	199.308165
181.086262	189.125644	193.747847	196.405363	197.933284	198.811776	199.316815
181.322699	189.261576	193.826004	196.450296	197.959122	198.826628	199.325345
181.556172	189.395801	193.903175	196.494672	197.984629	198.841303	199.333789
181.786733	189.528364	193.979391	196.538492	198.009823	198.855788	199.342115
182.014396	189.659258	194.054643	196.581753	198.034705	198.870087	199.350336
182.239222	189.788528	194.128963	196.624494	198.059275	198.884218	199.358462
182.461226	189.916170	194.202349	196.666678	198.083520	198.898162	199.366479
182.680475	190.042233	194.274828	196.708363	198.107481	198.911943	199.374396
182.896971	190.166702	194.346393	196.749493	198.131129	198.925538	199.382228
183.110768	190.289633	194.417073	196.790138	198.154493	198.938970	199.389943
183.321881	190.411007	194.486854	196.830267	198.177566	198.952229	199.397586
183.530369	190.530867	194.555777	196.869889	198.200349	198.965320	199.405110
183.736239	190.649236	194.623820	196.909019	198.222843	198.978257	199.412555
183.939545	190.766119	194.691027	196.947656	198.245062	198.991033	199.419891
184.140301	190.881544	194.757390	196.985803	198.267001	199.003643	199.427152
184.338555	190.995531	194.822919	197.023493	198.288662	199.016092	199.434307
184.534321	191.108087	194.887626	197.060701	198.310059	199.028390	199.441389
184.727651	191.219243	194.951536	197.097449	198.331178	199.040542	199.448363
184.918558	191.329005	195.014633	197.133733	198.352049	199.052536	199.455264
185.107080	191.437382	195.076965	197.169558	198.372645	199.064373	199.462073
185.293243	191.544428	195.138496	197.204940	198.392982	199.076067	199.468802
185.477080	191.650111	195.199273	197.239872	198.413066	199.087617	199.475442
185.658625	191.754504	195.259270	197.274378	198.432911	199.099019	199.481995
185.837886	191.857565	195.318547	197.308436	198.452499	199.110280	199.488475
186.014930	191.959350	195.377060	197.342089	198.471846	199.121407	199.494857
186.189745	192.059857	195.434856	197.375309	198.490953	199.132381	199.501183
186.362382	192.159119	195.491918	197.408125	198.509819	199.143240	199.507404
186.532859	192.257135	195.548283	197.440523	198.528439	199.153940	199.513578
186.701207	192.353919	195.603919	197.472520	198.546842	199.164511	199.519639
186.867437	192.449511	195.658886	197.504114	198.565003	199.174957	199.525656
187.031605	192.543890	195.713145	197.535309	198.582945	199.185270	199.531579
187.193713	192.637105	195.766734	197.566127	198.600648	199.195457	199.537437
187.353802	192.729137	195.819654	197.596546	198.618147	199.205514	199.543230
187.511884	192.820032	195.871912	197.626594	198.635415	199.215440	199.548936
187.667997	192.909776	195.923517	197.656258	198.652474	199.225262	199.554583
187.822151	192.998410	195.974472	197.685546	198.669313	199.234930	199.560149
187.974384	193.085920	196.024791	197.714486	198.685955	199.244503	199.565647
188.124700	193.172360	196.074478	197.743047	198.702378	199.253928	199.571073
188.273148	193.257700	196.123558	197.771265	198.718611	199.263275	199.576436
188.419728	193.341993	196.172016	197.799113	198.734625	199.272468	199.581730

199.586961	199.762524	199.863486	199.921513	199.954863	199.974049	199.985079
199.592118	199.765490	199.865199	199.922496	199.955424	199.974379	199.985262
199.597220	199.768422	199.866879	199.923452	199.955979	199.974699	199.985442
199.602260	199.771314	199.868549	199.924415	199.956538	199.975014	199.985623
199.607218	199.774169	199.870186	199.925348	199.957073	199.975326	199.985803
199.612132	199.776992	199.871813	199.926275	199.957623	199.975645	199.985984
199.616974	199.779775	199.873419	199.927198	199.958146	199.975939	199.986170
199.621764	199.782527	199.874997	199.928108	199.958671	199.976249	199.986327
199.626486	199.785247	199.876563	199.929019	199.959189	199.976546	199.986508
199.631156	199.787938	199.878109	199.929903	199.959693	199.976832	199.986668
199.635757	199.790590	199.879620	199.930788	199.960203	199.977125	199.986846
199.640316	199.793204	199.881136	199.931653	199.960689	199.977414	199.987006
199.644808	199.795787	199.882613	199.932509	199.961191	199.977688	199.987169
199.649249	199.798338	199.884088	199.933353	199.961665	199.977969	199.987321
199.653636	199.800860	199.885527	199.934187	199.962156	199.978247	199.987481
199.657959	199.803343	199.886971	199.935002	199.962619	199.978525	199.987633
199.662246	199.805802	199.888371	199.935816	199.963098	199.978782	199.987800
199.666457	199.808225	199.889783	199.936617	199.963543	199.979061	199.987948
199.670635	199.810624	199.891142	199.937420	199.964014	199.979312	199.988094
199.674752	199.812986	199.892518	199.938195	199.964448	199.979576	199.988237
199.678815	199.815326	199.893845	199.938971	199.964907	199.979825	199.988386
199.682833	199.817642	199.895180	199.939733	199.965330	199.980077	199.988526
199.686798	199.819915	199.896485	199.940477	199.965772	199.980335	199.988675
199.690715	199.822175	199.897783	199.941228	199.966201	199.980569	199.988815
199.694583	199.824388	199.899057	199.941961	199.966625	199.980812	199.988965
199.698409	199.826587	199.900322	199.942685	199.967046	199.981053	199.989090
199.702177	199.828755	199.901562	199.943392	199.967458	199.981300	199.989231
199.705905	199.830902	199.902797	199.944111	199.967868	199.981522	199.989359
199.709582	199.833006	199.904010	199.944804	199.968263	199.981755	199.989491
199.713209	199.835097	199.905222	199.945491	199.968664	199.981984	199.989629
199.716788	199.837155	199.906392	199.946181	199.969047	199.982213	199.989757
199.720339	199.839194	199.907576	199.946854	199.969437	199.982427	199.989889
199.723826	199.841210	199.908720	199.947518	199.969817	199.982648	199.990012
199.727276	199.843191	199.909875	199.948165	199.970193	199.982860	199.990133
199.730690	199.845168	199.910985	199.948824	199.970565	199.983080	199.990253
199.734054	199.847096	199.912108	199.949456	199.970943	199.983298	199.990373
199.737378	199.849024	199.913193	199.950083	199.971297	199.983501	199.990493
199.740657	199.850905	199.914287	199.950714	199.971668	199.983704	199.990614
199.743901	199.852784	199.915352	199.951326	199.972011	199.983914	199.990734
199.747111	199.854621	199.916423	199.951930	199.972363	199.984114	199.990854
199.750260	199.856449	199.917459	199.952532	199.972712	199.984309	199.990960
199.753393	199.858238	199.918505	199.953125	199.973047	199.984500	199.991072
199.756474	199.860016	199.919527	199.953714	199.973388	199.984698	199.991180
199.759526	199.861757	199.920526	199.954290	199.973726	199.984887	199.991289

199.991398	199.994170	199.996051	199.997325	199.998170	199.998745	199.999123
199.991507	199.994241	199.996100	199.997365	199.998199	199.998766	199.999135
199.991616	199.994313	199.996148	199.997403	199.998218	199.998774	199.999152
199.991718	199.994391	199.996191	199.997434	199.998247	199.998785	199.999161
199.991837	199.994459	199.996249	199.997474	199.998267	199.998803	199.999172
199.991922	199.994531	199.996288	199.997512	199.998296	199.998805	199.999183
199.992025	199.994602	199.996340	199.997543	199.998316	199.998826	199.999201
199.992123	199.994680	199.996389	199.997583	199.998339	199.998834	199.999203
199.992214	199.994748	199.996438	199.997614	199.998368	199.998845	199.999224
199.992314	199.994805	199.996480	199.997640	199.998387	199.998863	199.999232
199.992412	199.994868	199.996538	199.997669	199.998416	199.998871	199.999243
199.992503	199.994928	199.996578	199.997689	199.998436	199.998883	199.999261
199.992604	199.994989	199.996629	199.997711	199.998459	199.998894	199.999263
199.992701	199.995049	199.996678	199.997740	199.998488	199.998905	199.999284
199.992792	199.995109	199.996712	199.997760	199.998508	199.998923	199.999292
199.992878	199.995175	199.996746	199.997789	199.998537	199.998931	199.999304
199.992967	199.995226	199.996787	199.997809	199.998556	199.998943	199.999321
199.993047	199.995295	199.996824	199.997838	199.998585	199.998954	199.999323
199.993136	199.995346	199.996855	199.997858	199.998590	199.998972	199.999344
199.993216	199.995416	199.996896	199.997880	199.998605	199.998974	199.999352
199.993305	199.995466	199.996927	199.997909	199.998616	199.998995	199.999364
199.993385	199.995536	199.996967	199.997929	199.998634	199.999003	199.999381
199.993474	199.995593	199.997005	199.997958	199.998642	199.999014	199.999390
199.993554	199.995653	199.997036	199.997978	199.998654	199.999032	199.999401
199.993637	199.995713	199.997076	199.998007	199.998665	199.999034	199.999412
199.993726	199.995759	199.997113	199.998027	199.998676	199.999055	199.999430
199.993806	199.995811	199.997145	199.998049	199.998694	199.999063	199.999432
199.993881	199.995859	199.997185	199.998078	199.998702	199.999074	199.999453
199.993952	199.995902	199.997216	199.998098	199.998714	199.999092	199.999461
199.994024	199.995960	199.997256	199.998127	199.998725	199.999094	199.999473
199.994101	199.995999	199.997294	199.998147	199.998743	199.999115	

PID 四　　增量型 PID 的 C 语言实现

上一节中介绍了最简单的位置型 PID 的实现手段，这一节主要讲解增量式 PID 的实现方法，位置型和增量型 PID 的数学公式请参见我的系列文《PID 控制算法的 C 语言实现二》中的讲解。实现过程仍然是分为定义变量、初始化变量、实现控制算法函数、算法测试四个部分，详细分类请参加《PID 控制算法的 C 语言实现三》中的讲解，这里直接给出代码了。

```
#include<stdio.h>
#include<stdlib.h>
```

```
struct _pid{
	float SetSpeed;                    //定义设定值
	float ActualSpeed;                 //定义实际值
	float err;                           //定义偏差值
	float err_next;                    //定义上一个偏差值
	float err_last;                    //定义最上前的偏差值
	float Kp,Ki,Kd;                    //定义比例、积分、微分系数
}pid;

void PID_init(){
	pid.SetSpeed=0.0;
	pid.ActualSpeed=0.0;
	pid.err=0.0;
	pid.err_last=0.0;
	pid.err_next=0.0;
	pid.Kp=0.2;
	pid.Ki=0.015;
	pid.Kd=0.2;
}

float PID_realize(float speed){
	pid.SetSpeed=speed;
	pid.err=pid.SetSpeed-pid.ActualSpeed;
	float incrementSpeed =
pid.Kp*(pid.err-pid.err_next)+pid.Ki*pid.err+pid.Kd*(pid.err-2*pid.er
r_next+pid.err_last);
	pid.ActualSpeed+=incrementSpeed;
	pid.err_last=pid.err_next;
	pid.err_next=pid.err;
	return pid.ActualSpeed;
}

int main(){
	PID_init();
	int count=0;
	while(count<1000)
	{
		float speed=PID_realize(200.0);
		printf("%f\n",speed);
		count++;
	}
	return 0;
}
```

运行后的 1000 个数据为：

83.000000	102.955040	142.782974	166.265259	180.110214	188.273132	193.085907
11.555000	104.168114	143.498199	166.686951	180.358841	188.419724	193.172333
59.559677	105.366058	144.204498	167.103378	180.604370	188.564484	193.257675
28.175406	106.549004	144.901962	167.514587	180.846817	188.707428	193.341965
52.907425	107.717178	145.590714	167.920670	181.086243	188.848587	193.425186
38.944149	108.870743	146.270844	168.321671	181.322662	188.987976	193.507385
51.891701	110.009888	146.942474	168.717667	181.556137	189.125626	193.588531
46.141655	111.134796	147.605713	169.108704	181.786682	189.261566	193.668686
53.339050	112.245636	148.260651	169.494858	182.014359	189.395798	193.747818
51.510002	113.342598	148.907410	169.876175	182.239182	189.528351	193.825974
55.908447	114.425842	149.546082	170.252731	182.461197	189.659256	193.903152
55.944637	115.495552	150.176773	170.624588	182.680435	189.788513	193.979370
58.970676	116.551880	150.799576	170.991791	182.896942	189.916168	194.054626
59.882942	117.595009	151.414597	171.354401	183.110733	190.042221	194.128952
62.224998	118.625099	152.021927	171.712479	183.321854	190.166702	194.202332
63.537247	119.642311	152.621674	172.066086	183.530334	190.289627	194.274811
65.527702	120.646812	153.213913	172.415268	183.736206	190.411011	194.346375
67.011047	121.638756	153.798752	172.760086	183.939514	190.530884	194.417053
68.810638	122.618294	154.376282	173.100601	184.140274	190.649246	194.486832
70.355309	123.585594	154.946594	173.436844	184.338531	190.766144	194.555756
72.042023	124.540794	155.509781	173.768890	184.534302	190.881561	194.623810
73.595642	125.484062	156.065918	174.096786	184.727631	190.995544	194.691010
75.207603	126.415535	156.615112	174.420578	184.918533	191.108109	194.757370
76.745430	127.335365	157.157440	174.740326	185.107056	191.219254	194.822906
78.301514	128.243698	157.692993	175.056076	185.293228	191.329025	194.887619
79.812126	129.140671	158.221848	175.367889	185.477066	191.437408	194.951523
81.321915	130.026443	158.744095	175.675797	185.658615	191.544449	195.014633
82.800293	130.901138	159.259811	175.979858	185.837891	191.650146	195.076950
84.268898	131.764893	159.769073	176.280121	186.014923	191.754517	195.138489
85.713097	132.617859	160.271973	176.576630	186.189743	191.857590	195.199265
87.143440	133.460159	160.768585	176.869431	186.362381	191.959366	195.259277
88.552994	134.291931	161.258987	177.158569	186.532852	192.059875	195.318542
89.946945	135.113297	161.743271	177.444092	186.701202	192.159134	195.377060
91.322067	135.924408	162.221481	177.726044	186.867432	192.257141	195.434845
92.680977	136.725372	162.693726	178.004471	187.031601	192.353928	195.491913
94.022224	137.516327	163.160065	178.279419	187.193710	192.449509	195.548264
95.347176	138.297394	163.620575	178.550934	187.353790	192.543884	195.603912
96.655235	139.068695	164.075333	178.819046	187.511871	192.637085	195.658859
97.947174	139.830353	164.524399	179.083817	187.667984	192.729126	195.713135
99.222801	140.582489	164.967865	179.345276	187.822128	192.820007	195.766724
100.482597	141.325226	165.405777	179.603470	187.974365	192.909760	195.819641
101.726562	142.058685	165.838226	179.858429	188.124680	192.998383	195.871902

195.923508	197.656250	198.652466	199.225266	199.554611	199.743942	199.852798
195.974472	197.685547	198.669312	199.234955	199.560181	199.747147	199.854645
196.024796	197.714478	198.685944	199.244522	199.565674	199.750305	199.856461
196.074493	197.743042	198.702377	199.253967	199.571106	199.753433	199.858261
196.123566	197.771255	198.718597	199.263290	199.576462	199.756516	199.860031
196.172028	197.799118	198.734619	199.272507	199.581757	199.759567	199.861786
196.219879	197.826630	198.750443	199.281601	199.586990	199.762573	199.863510
196.267136	197.853806	198.766068	199.290588	199.592148	199.765549	199.865219
196.313797	197.880630	198.781494	199.299454	199.597244	199.768478	199.866898
196.359879	197.907120	198.796722	199.308212	199.602280	199.771378	199.868561
196.405380	197.933289	198.811768	199.316864	199.607254	199.774231	199.870209
196.450317	197.959122	198.826614	199.325409	199.612167	199.777054	199.871826
196.494690	197.984634	198.841278	199.333847	199.617020	199.779846	199.873428
196.538513	198.009827	198.855759	199.342178	199.621811	199.782593	199.875015
196.581787	198.034698	198.870056	199.350403	199.626541	199.785309	199.876572
196.624512	198.059265	198.884186	199.358521	199.631210	199.787994	199.878113
196.666702	198.083527	198.898132	199.366547	199.635818	199.790649	199.879639
196.708374	198.107483	198.911911	199.374466	199.640366	199.793259	199.881149
196.749512	198.131134	198.925507	199.382294	199.644867	199.795853	199.882629
196.790146	198.154495	198.938934	199.390015	199.649307	199.798401	199.884094
196.830261	198.177567	198.952194	199.397644	199.653687	199.800919	199.885544
196.869888	198.200348	198.965286	199.405167	199.658020	199.803406	199.886978
196.909012	198.222839	198.978226	199.412613	199.662292	199.805862	199.888397
196.947647	198.245056	198.990997	199.419952	199.666519	199.808289	199.889786
196.985809	198.266998	199.003616	199.427200	199.670685	199.810684	199.891174
197.023483	198.288666	199.016068	199.434357	199.674805	199.813049	199.892532
197.060699	198.310059	199.028366	199.441422	199.678864	199.815384	199.893875
197.097443	198.331177	199.040512	199.448410	199.682877	199.817688	199.895203
197.133728	198.352036	199.052505	199.455307	199.686844	199.819962	199.896515
197.169556	198.372635	199.064346	199.462112	199.690750	199.822220	199.897812
197.204941	198.392975	199.076050	199.468842	199.694626	199.824432	199.899094
197.239883	198.413071	199.087601	199.475479	199.698441	199.826630	199.900360
197.274384	198.432907	199.099014	199.482040	199.702209	199.828796	199.901611
197.308456	198.452499	199.110275	199.488510	199.705933	199.830933	199.902847
197.342102	198.471848	199.121399	199.494904	199.709610	199.833054	199.904068
197.375320	198.490952	199.132385	199.501221	199.713242	199.835144	199.905273
197.408127	198.509811	199.143234	199.507462	199.716827	199.837204	199.906464
197.440521	198.528442	199.153946	199.513611	199.720367	199.839233	199.907639
197.472519	198.546829	199.164520	199.519699	199.723862	199.841248	199.908798
197.504105	198.565002	199.174973	199.525696	199.727310	199.843231	199.909943
197.535309	198.582932	199.185287	199.531631	199.730713	199.845200	199.911072
197.566116	198.600647	199.195465	199.537476	199.734085	199.847137	199.912186
197.596542	198.618134	199.205521	199.543259	199.737411	199.849045	199.913284
197.626587	198.635406	199.215454	199.548965	199.740692	199.850937	199.914368

199.915436	199.951385	199.972015	199.983932	199.990799	199.994751	199.997025
199.916489	199.951996	199.972366	199.984131	199.990906	199.994812	199.997055
199.917526	199.952591	199.972717	199.984329	199.991028	199.994873	199.997101
199.918564	199.953186	199.973053	199.984528	199.991135	199.994934	199.997131
199.919571	199.953766	199.973389	199.984726	199.991257	199.994995	199.997177
199.920578	199.954346	199.973724	199.984909	199.991364	199.995056	199.997208
199.921570	199.954910	199.974045	199.985107	199.991470	199.995117	199.997253
199.922546	199.955475	199.974380	199.985291	199.991577	199.995178	199.997284
199.923523	199.956024	199.974701	199.985474	199.991684	199.995239	199.997314
199.924469	199.956573	199.975021	199.985657	199.991791	199.995300	199.997345
199.925415	199.957108	199.975327	199.985840	199.991898	199.995361	199.997375
199.926346	199.957642	199.975632	199.986023	199.992004	199.995422	199.997406
199.927261	199.958176	199.975937	199.986191	199.992096	199.995483	199.997437
199.928177	199.958694	199.976242	199.986374	199.992203	199.995544	199.997467
199.929077	199.959213	199.976532	199.986542	199.992294	199.995605	199.997498
199.929962	199.959717	199.976822	199.986710	199.992401	199.995667	199.997528
199.930832	199.960220	199.977112	199.986877	199.992493	199.995712	199.997559
199.931702	199.960724	199.977402	199.987045	199.992584	199.995773	199.997589
199.932556	199.961212	199.977676	199.987213	199.992676	199.995819	199.997620
199.933395	199.961700	199.977966	199.987366	199.992767	199.995880	199.997650
199.934235	199.962173	199.978241	199.987534	199.992859	199.995926	199.997681
199.935059	199.962646	199.978516	199.987686	199.992950	199.995987	199.997711
199.935867	199.963120	199.978790	199.987839	199.993042	199.996033	199.997742
199.936676	199.963577	199.979050	199.987991	199.993134	199.996094	199.997772
199.937469	199.964035	199.979309	199.988144	199.993225	199.996140	199.997803
199.938248	199.964478	199.979568	199.988297	199.993301	199.996185	199.997833
199.939026	199.964920	199.979828	199.988449	199.993393	199.996231	199.997864
199.939789	199.965363	199.980072	199.988586	199.993469	199.996277	199.997894
199.940536	199.965790	199.980331	199.988739	199.993561	199.996323	199.997925
199.941284	199.966217	199.980576	199.988876	199.993637	199.996368	199.997955
199.942017	199.966644	199.980820	199.989014	199.993713	199.996414	199.997986
199.942749	199.967056	199.981064	199.989151	199.993790	199.996460	199.998016
199.943466	199.967468	199.981293	199.989288	199.993866	199.996506	199.998047
199.944168	199.967880	199.981537	199.989426	199.993942	199.996552	199.998077
199.944870	199.968277	199.981766	199.989563	199.994019	199.996597	199.998093
199.945557	199.968674	199.981995	199.989685	199.994095	199.996643	199.998123
199.946243	199.969070	199.982224	199.989822	199.994171	199.996689	199.998138
199.946915	199.969452	199.982437	199.989944	199.994247	199.996735	199.998169
199.947586	199.969833	199.982666	199.990067	199.994324	199.996780	199.998184
199.948242	199.970215	199.982880	199.990189	199.994400	199.996826	199.998215
199.948883	199.970581	199.983093	199.990311	199.994476	199.996872	199.998230
199.949524	199.970947	199.983307	199.990433	199.994537	199.996902	199.998260
199.950150	199.971313	199.983521	199.990555	199.994614	199.996948	199.998276
199.950775	199.971664	199.983719	199.990677	199.994675	199.996979	199.998306

199.998322	199.998581	199.998779	199.998978	199.999176	199.999329	199.999435
199.998352	199.998596	199.998795	199.998993	199.999191	199.999344	199.999435
199.998367	199.998611	199.998810	199.999008	199.999207	199.999344	199.999451
199.998398	199.998627	199.998825	199.999023	199.999222	199.999359	199.999451
199.998413	199.998642	199.998840	199.999039	199.999237	199.999359	199.999466
199.998444	199.998657	199.998856	199.999054	199.999252	199.999374	199.999466
199.998459	199.998672	199.998871	199.999069	199.999268	199.999374	199.999481
199.998489	199.998688	199.998886	199.999084	199.999283	199.999390	199.999481
199.998505	199.998703	199.998901	199.999100	199.999298	199.999390	199.999496
199.998520	199.998718	199.998917	199.999115	199.999298	199.999405	199.999496
199.998535	199.998734	199.998932	199.999130	199.999313	199.999405	199.999512
199.998550	199.998749	199.998947	199.999146	199.999313	199.999420	199.999512
199.998566	199.998764	199.998962	199.999161	199.999329	199.999420	

PID 五　　积分分离的 PID 控制算法

通过三、四两篇文章，基本上已经弄清楚了 PID 控制算法的最常规的表达方法。在普通 PID 控制中，引入积分环节的目的，主要是为了消除静差，提高控制精度。但是在启动、结束或大幅度增减设定时，短时间内系统输出有很大的偏差，会造成 PID 运算的积分积累，导致控制量超过执行机构可能允许的最大动作范围对应极限控制量，从而引起较大的超调，甚至是震荡，这是绝对不允许的。

为了克服这一问题，引入了积分分离的概念，其基本思路是 当被控量与设定值偏差较大时，取消积分作用；当被控量接近给定值时，引入积分控制，以消除静差，提高精度。其具体实现代码如下：

```
        pid.Kp=0.2;
        pid.Ki=0.04;
        pid.Kd=0.2;    //初始化过程


if(abs(pid.err)>200)
        {
        index=0;
        }else{
        index=1;
        pid.integral+=pid.err;
        }
        pid.voltage=pid.Kp*pid.err+index*pid.Ki*pid.integral+pid.Kd*(p
id.err-pid.err_last);        //算法具体实现过程
```

其它部分的代码参见《PID 控制算法的 C 语言实现三》中的讲解，不再赘述。同样采集 1000 个量，会发现，系统到 199 所有的时间是原来时间的 1/2,系统的快速性得到了提高。

199.003571	199.527420	199.775894	199.893707	199.949585	199.976074	199.988663
199.036804	199.543182	199.783371	199.897263	199.951248	199.976852	199.989044
199.068924	199.558426	199.790588	199.900665	199.952896	199.977631	199.989395
199.099960	199.573135	199.797577	199.903992	199.954437	199.978378	199.989761
199.129974	199.587372	199.804337	199.907181	199.955963	199.979095	199.990097
199.158981	199.601120	199.810867	199.910278	199.957428	199.979797	199.990417
199.187012	199.614426	199.817154	199.913284	199.958847	199.980453	199.990753
199.214111	199.627289	199.823257	199.916168	199.960205	199.981125	199.991058
199.240311	199.639694	199.829147	199.918976	199.961548	199.981735	199.991348
199.265640	199.651718	199.834839	199.921677	199.962830	199.982361	199.991653
199.290115	199.663315	199.840347	199.924286	199.964066	199.982925	199.991913
199.313797	199.674561	199.845673	199.926804	199.965271	199.983505	199.992203
199.336655	199.685410	199.850815	199.929245	199.966431	199.984055	199.992447
199.358795	199.695908	199.855789	199.931610	199.967545	199.984604	199.992706
199.380157	199.706039	199.860596	199.933884	199.968628	199.985107	199.992950
199.400818	199.715851	199.865234	199.936081	199.969666	199.985611	199.993179
199.420792	199.725311	199.869736	199.938217	199.970673	199.986069	199.993408
199.440109	199.734482	199.874069	199.940277	199.971649	199.986557	199.993607
199.458771	199.743332	199.878281	199.942276	199.972595	199.987000	199.993835
199.476807	199.751907	199.882324	199.944183	199.973511	199.987442	199.994034
199.494263	199.760162	199.886261	199.946045	199.974380	199.987869	199.994232
199.511124	199.768173	199.890045	199.947830	199.975235	199.988281	199.994431

199.994598	199.998795	199.999756	199.999771	199.999771	199.999771	199.999771
199.994797	199.998825	199.999771	199.999771	199.999771	199.999771	199.999771
199.994965	199.998856	199.999771	199.999771	199.999771	199.999771	199.999771
199.995132	199.998886	199.999771	199.999771	199.999771	199.999771	199.999771
199.995285	199.998917	199.999771	199.999771	199.999771	199.999771	199.999771
199.995453	199.998962	199.999771	199.999771	199.999771	199.999771	199.999771
199.995605	199.998978	199.999771	199.999771	199.999771	199.999771	199.999771
199.995743	199.999023	199.999771	199.999771	199.999771	199.999771	199.999771
199.995895	199.999054	199.999771	199.999771	199.999771	199.999771	199.999771
199.996017	199.999084	199.999771	199.999771	199.999771	199.999771	199.999771
199.996155	199.999115	199.999771	199.999771	199.999771	199.999771	199.999771
199.996277	199.999146	199.999771	199.999771	199.999771	199.999771	199.999771
199.996414	199.999191	199.999771	199.999771	199.999771	199.999771	199.999771
199.996521	199.999207	199.999771	199.999771	199.999771	199.999771	199.999771
199.996643	199.999252	199.999771	199.999771	199.999771	199.999771	199.999771
199.996750	199.999283	199.999771	199.999771	199.999771	199.999771	199.999771
199.996872	199.999298	199.999771	199.999771	199.999771	199.999771	199.999771
199.996964	199.999313	199.999771	199.999771	199.999771	199.999771	199.999771
199.997070	199.999329	199.999771	199.999771	199.999771	199.999771	199.999771
199.997162	199.999344	199.999771	199.999771	199.999771	199.999771	199.999771
199.997269	199.999359	199.999771	199.999771	199.999771	199.999771	199.999771
199.997360	199.999374	199.999771	199.999771	199.999771	199.999771	199.999771
199.997437	199.999390	199.999771	199.999771	199.999771	199.999771	199.999771
199.997528	199.999405	199.999771	199.999771	199.999771	199.999771	199.999771
199.997604	199.999435	199.999771	199.999771	199.999771	199.999771	199.999771
199.997681	199.999435	199.999771	199.999771	199.999771	199.999771	199.999771
199.997772	199.999466	199.999771	199.999771	199.999771	199.999771	199.999771
199.997849	199.999466	199.999771	199.999771	199.999771	199.999771	199.999771
199.997910	199.999496	199.999771	199.999771	199.999771	199.999771	199.999771
199.997986	199.999512	199.999771	199.999771	199.999771	199.999771	199.999771
199.998047	199.999527	199.999771	199.999771	199.999771	199.999771	199.999771
199.998108	199.999542	199.999771	199.999771	199.999771	199.999771	199.999771
199.998169	199.999557	199.999771	199.999771	199.999771	199.999771	199.999771
199.998245	199.999573	199.999771	199.999771	199.999771	199.999771	199.999771
199.998306	199.999588	199.999771	199.999771	199.999771	199.999771	199.999771
199.998352	199.999603	199.999771	199.999771	199.999771	199.999771	199.999771
199.998398	199.999619	199.999771	199.999771	199.999771	199.999771	199.999771
199.998459	199.999634	199.999771	199.999771	199.999771	199.999771	199.999771
199.998489	199.999649	199.999771	199.999771	199.999771	199.999771	199.999771
199.998550	199.999680	199.999771	199.999771	199.999771	199.999771	199.999771
199.998596	199.999680	199.999771	199.999771	199.999771	199.999771	199.999771
199.998642	199.999710	199.999771	199.999771	199.999771	199.999771	199.999771
199.998703	199.999710	199.999771	199.999771	199.999771	199.999771	199.999771
199.998734	199.999741	199.999771	199.999771	199.999771	199.999771	199.999771

199.999771	199.999771	199.999771	199.999771	199.999771	199.999771	199.999771

```
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
199.999771  199.999771  199.999771  199.999771  199.999771  199.999771
```

PID 六 抗积分饱和的 PID 控制算法 C 语言实现

所谓的积分饱和现象是指如果系统存在一个方向的偏差，PID 控制器的输出由于积分作用的不断累加而加大，从而导致执行机构达到极限位置，若控制器输出 U(k)继续增大，执行器开度不可能再增大，此时计算机输出控制量超出了正常运行范围而进入饱和区。一旦系统出现反向偏差，u(k)逐渐从饱和区退出。进入饱和区越深则退出饱和区时间越长。在这段时间里，执行机构仍然停留在极限位置而不随偏差反向而立即做出相应的改变，这时系统就像失控一样，造成控制性能恶化，这种现象称为积分饱和现象或积分失控现象。

防止积分饱和的方法之一就是抗积分饱和法，该方法的思路是在计算 u(k)时，首先判断上一时刻的控制量 u(k-1)是否已经超出了极限范围： 如果 u(k-1)>umax，则只累加负偏差; 如果 u(k-1)<umin，则只累加正偏差。从而避免控制量长时间停留在饱和区。直接贴出代码，不懂的看看前面几节的介绍。

```
struct _pid{
        float SetSpeed;                         //定义设定值
        float ActualSpeed;                    //定义实际值
        float err;                                 //定义偏差值
        float err_last;                         //定义上一个偏差值
        float Kp,Ki,Kd;                         //定义比例、积分、微分系数
        float voltage;                          //定义电压值（控制执行器的变
量）
        float integral;                         //定义积分值
        float umax;
        float umin;
}pid;

void PID_init(){
        printf("PID_init begin \n");
        pid.SetSpeed=0.0;
        pid.ActualSpeed=0.0;
        pid.err=0.0;
        pid.err_last=0.0;
        pid.voltage=0.0;
        pid.integral=0.0;
        pid.Kp=0.2;
       pid.Ki=0.1;              //注意，和上几次相比，这里加大了积分环节
的值
        pid.Kd=0.2;
        pid.umax=400;
        pid.umin=-200;
        printf("PID_init end \n");
}
float PID_realize(float speed){
        int index;
        pid.SetSpeed=speed;
        pid.err=pid.SetSpeed-pid.ActualSpeed;

       if(pid.ActualSpeed>pid.umax)    //灰色底色表示抗积分饱和的实现
        {

              if(abs(pid.err)>200)           //蓝色标注为积分分离过程
               {
                      index=0;
               }else{
                      index=1;
                      if(pid.err<0)
                      {
```

```
                        pid.integral+=pid.err;
                    }
                }
        }else if(pid.ActualSpeed<pid.umin){
                if(abs(pid.err)>200)            //积分分离过程
                {
                        index=0;
                }else{
                        index=1;
                        if(pid.err>0)
                        {
                        pid.integral+=pid.err;
                        }
                }
        }else{
                if(abs(pid.err)>200)
//积分分离过程
                {
                        index=0;
                }else{
                        index=1;
                        pid.integral+=pid.err;
                }
        }

        pid.voltage=pid.Kp*pid.err+index*pid.Ki*pid.integral+pid.Kd*(p
id.err-pid.err_last);

        pid.err_last=pid.err;
        pid.ActualSpeed=pid.voltage*1.0;
        return pid.ActualSpeed;
}
```

最终的测试程序运算结果如下，可以明显的看出系统的稳定时间相对前几次来讲缩短了不少。

100.000000	127.653938	168.149734	185.400513	193.338928	196.959244	198.612015
30.000000	137.468842	170.611786	186.628952	193.894257	197.213043	198.727829
95.000000	139.967911	173.205124	187.737457	194.404160	197.445572	198.834000
65.500000	146.934479	175.339691	188.766006	194.870834	197.658768	198.931290
103.750000	149.954224	177.470551	189.699692	195.299072	197.854111	199.020477
92.175003	155.144211	179.298065	190.561951	195.691193	198.033203	199.102219
115.237503	158.157745	181.063431	191.347580	196.050888	198.197311	199.177139
112.173752	162.174561	182.616440	192.071030	196.380341	198.347763	199.245804
126.794380	164.953079	184.086655	192.731674	196.682465	198.485626	199.308746

199.366425	199.986298	199.999710	199.999939	199.999939	199.999939	199.999939
199.419296	199.987442	199.999725	199.999939	199.999939	199.999939	199.999939
199.467758	199.988495	199.999756	199.999939	199.999939	199.999939	199.999939
199.512161	199.989441	199.999771	199.999939	199.999939	199.999939	199.999939
199.552872	199.990326	199.999786	199.999939	199.999939	199.999939	199.999939
199.590179	199.991135	199.999817	199.999939	199.999939	199.999939	199.999939
199.624390	199.991867	199.999817	199.999939	199.999939	199.999939	199.999939
199.655716	199.992554	199.999847	199.999939	199.999939	199.999939	199.999939
199.684464	199.993179	199.999847	199.999939	199.999939	199.999939	199.999939
199.710785	199.993744	199.999863	199.999939	199.999939	199.999939	199.999939
199.734924	199.994263	199.999863	199.999939	199.999939	199.999939	199.999939
199.757034	199.994751	199.999878	199.999939	199.999939	199.999939	199.999939
199.777298	199.995178	199.999893	199.999939	199.999939	199.999939	199.999939
199.795883	199.995590	199.999893	199.999939	199.999939	199.999939	199.999939
199.812912	199.995941	199.999908	199.999939	199.999939	199.999939	199.999939
199.828537	199.996292	199.999924	199.999939	199.999939	199.999939	199.999939
199.842834	199.996582	199.999924	199.999939	199.999939	199.999939	199.999939
199.855972	199.996887	199.999939	199.999939	199.999939	199.999939	199.999939
199.867981	199.997116	199.999939	199.999939	199.999939	199.999939	199.999939
199.879013	199.997391	199.999939	199.999939	199.999939	199.999939	199.999939
199.889099	199.997574	199.999939	199.999939	199.999939	199.999939	199.999939
199.898361	199.997803	199.999939	199.999939	199.999939	199.999939	199.999939
199.906845	199.997971	199.999939	199.999939	199.999939	199.999939	199.999939
199.914612	199.998154	199.999939	199.999939	199.999939	199.999939	199.999939
199.921753	199.998291	199.999939	199.999939	199.999939	199.999939	199.999939
199.928268	199.998444	199.999939	199.999939	199.999939	199.999939	199.999939
199.934280	199.998581	199.999939	199.999939	199.999939	199.999939	199.999939
199.939743	199.998703	199.999939	199.999939	199.999939	199.999939	199.999939
199.944794	199.998810	199.999939	199.999939	199.999939	199.999939	199.999939
199.949371	199.998917	199.999939	199.999939	199.999939	199.999939	199.999939
199.953629	199.999008	199.999939	199.999939	199.999939	199.999939	199.999939
199.957474	199.999084	199.999939	199.999939	199.999939	199.999939	199.999939
199.961029	199.999176	199.999939	199.999939	199.999939	199.999939	199.999939
199.964279	199.999237	199.999939	199.999939	199.999939	199.999939	199.999939
199.967270	199.999298	199.999939	199.999939	199.999939	199.999939	199.999939
199.969986	199.999359	199.999939	199.999939	199.999939	199.999939	199.999939
199.972504	199.999405	199.999939	199.999939	199.999939	199.999939	199.999939
199.974792	199.999466	199.999939	199.999939	199.999939	199.999939	199.999939
199.976898	199.999496	199.999939	199.999939	199.999939	199.999939	199.999939
199.978821	199.999542	199.999939	199.999939	199.999939	199.999939	199.999939
199.980591	199.999588	199.999939	199.999939	199.999939	199.999939	199.999939
199.982208	199.999619	199.999939	199.999939	199.999939	199.999939	199.999939
199.983688	199.999649	199.999939	199.999939	199.999939	199.999939	199.999939
199.985062	199.999680	199.999939	199.999939	199.999939	199.999939	199.999939

199.999939	199.999939	199.999939	199.999939	199.999939	199.999939	199.999939

199.999939	199.999939	199.999939	199.999939	199.999939	199.999939	199.999939

199.999939	199.999939	199.999939	199.999939	199.999939
199.999939	199.999939	199.999939	199.999939	199.99993
199.999939	199.999939	199.999939		

PID 七　　梯形积分的 PID 控制算法 C 语言实现

先看一下梯形算法的积分环节公式

$$\int_0^t e(t)dt = \sum_{i=0}^{k} \frac{e(i)+e(i-1)}{2}T$$

作为 PID 控制律的积分项，其作用是消除余差，为了尽量减小余差，应提高积分项运算精度，为此可以将矩形积分改为梯形积分，具体实现的语句为：

```
pid.voltage=pid.Kp*pid.err+index*pid.Ki*pid.integral/2+pid.Kd*(pid.err-pid.err_last);   //梯形积分
```

其它函数请参见本系列教程六中的介绍

最后运算的稳定数据为：199.999878，较教程六中的 199.9999390 而言，精度进一步提高。

PID 八　　变积分的 PID 控制算法 C 语言实现

变积分 PID 可以看成是积分分离的 PID 算法的更一般的形式。在普通的 PID 控制算法中，由于积分系数 ki 是常数，所以在整个控制过程中，积分增量是不变的。但是，系统对于积分项的要求是，系统偏差大时，积分作用应该减弱甚至是全无，而在偏差小时，则应该加强。积分系数取大了会产生超调，甚至积分饱和，取小了又不能短时间内消除静差。因此，根据系统的偏差大小改变积分速度是有必要的。

变积分 PID 的基本思想是设法改变积分项的累加速度，使其与偏差大小相对应：偏差越大，积分越慢；偏差越小，积分越快。

这里给积分系数前加上一个比例值 index：

当 abs(err)<180 时，index=1;

当 180<abs(err)<200 时，index=（200-abs(err)）/20;

当 abs(err)>200 时，index=0;

最终的比例环节的比例系数值为 ki*index;

具体 PID 实现代码如下：

```
  pid.Kp=0.4;
  pid.Ki=0.2;        //增加了积分系数
  pid.Kd=0.2;



float PID_realize(float speed){
  float index;
  pid.SetSpeed=speed;
  pid.err=pid.SetSpeed-pid.ActualSpeed;

  if(abs(pid.err)>200)                    //变积分过程
  {
  index=0.0;
  }else if(abs(pid.err)<180){
  index=1.0;
  pid.integral+=pid.err;
  }else{
```

```
        index=(200-abs(pid.err))/20;
        pid.integral+=pid.err;
        }
        pid.voltage=pid.Kp*pid.err+index*pid.Ki*pid.integral+pid.Kd*(p
id.err-pid.err_last);

        pid.err_last=pid.err;
        pid.ActualSpeed=pid.voltage*1.0;
        return pid.ActualSpeed;
}
```

最终结果可以看出，系统的稳定速度非常快（测试程序参见本系列教程 3）：

120.000000	201.361328	200.121704	200.007263	200.000397	200.000046	200.000046
64.000000	197.143387	199.875870	199.993256	199.999619	199.999954	199.999954
148.800003	201.225632	200.102432	200.006088	200.000336	200.000046	200.000046
96.959999	197.687561	199.896851	199.994370	199.999680	199.999954	199.999954
165.632004	201.089340	200.086136	200.005081	200.000275	200.000046	200.000046
120.934395	198.122787	199.914230	199.995300	199.999725	199.999954	199.999954
177.300476	200.958511	200.072372	200.004257	200.000229	200.000046	200.000046
139.081223	198.472076	199.928635	199.996063	199.999756	199.999954	199.999954
185.469742	200.836655	200.060776	200.003555	200.000198	200.000046	200.000046
152.898834	198.753296	199.940582	199.996719	199.999802	199.999954	199.999954
191.139313	200.725555	200.051010	200.002975	200.000168	200.000046	200.000046
163.452988	198.980423	199.950500	199.997253	199.999832	199.999954	199.999954
195.022278	200.625870	200.042801	200.002487	200.000137	200.000046	200.000046
171.538986	199.164398	199.958755	199.997711	199.999863	199.999954	199.999954
197.635025	200.537506	200.035904	200.002075	200.000107	200.000046	200.000046
177.753738	199.313843	199.965622	199.998077	199.999893	199.999954	199.999954
199.350967	200.459900	200.030090	200.001740	200.000092	200.000046	200.000046
182.546188	199.435547	199.971344	199.998398	199.999908	199.999954	199.999954
200.439255	200.392258	200.025223	200.001465	200.000076	200.000046	200.000046
186.254608	199.534912	199.976105	199.998657	199.999924	199.999954	199.999954
201.093094	200.333679	200.021118	200.001221	200.000061	200.000046	200.000046
189.134460	199.616211	199.980057	199.998886	199.999939	199.999954	199.999954
201.450439	200.283203	200.017700	200.001007	200.000046	200.000046	200.000046
191.379044	199.682877	199.983353	199.999084	199.999954	199.999954	199.999954
201.609268	200.239899	200.014832	200.000839	200.000046	200.000046	200.000046
193.135010	199.737640	199.986099	199.999237	199.999954	199.999954	199.999954
201.638611	200.202866	200.012421	200.000702	200.000046	200.000046	200.000046
194.513870	199.782700	199.988403	199.999359	199.999954	199.999954	199.999954
201.586670	200.171295	200.010391	200.000580	200.000046	200.000046	200.000046
195.600708	199.819855	199.990326	199.999451	199.999954	199.999954	199.999954
201.486694	200.144470	200.008698	200.000488	200.000046	200.000046	200.000046
196.460571	199.850525	199.991928	199.999542	199.999954	199.999954	199.999954

200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954

200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954
200.000046	200.000046	200.000046	200.000046	200.000046	200.000046	200.000046
199.999954	199.999954	199.999954	199.999954	199.999954	199.999954	199.999954

200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046

199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954

200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046

199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954

200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046

199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954

200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046
199.999954
200.000046

PID 九　　专家 PID 与模糊 PID 的 C 语言实现

本节是 PID 控制算法的 C 语言实现系列的最后一节，前面 8 节中，已经分别从 PID 的实现到深入的过程进行了一个简要的讲解，从前面的讲解中不难看出，PID 的控制思想非常简单，其主要问题点和难点在于比例、积分、微分环节上的参数整定过程，对于执行器控制模型确定或者控制模型简单的系统而言，参数的整定可以通过计算获得，对于一般精度要求不是很高的执行器系统，可以采用拼凑的方法进行实验型的整定。

然而，在实际的控制系统中，线性系统毕竟是少数，大部分的系统属于非线性系统，或者说是系统模型不确定的系统，如果控制精度要求较高的话，那么对于参数的整定过程是有难度的。专家 PID 和模糊 PID 就是为满足这方面的需求而设计的。专家算法和模糊算法都归属于智能算法的范畴，智能算法最大的优点就是在控制模型未知的情况下，可以对模型进行控制。这里需要注意的是，专家 PID 也好，模糊 PID 也罢，绝对不是专家系统或模糊算法与 PID 控制算法的简单加和，他是专家系统或者模糊算法在 PID 控制器参数整定上的应用。也就是说，智能算法是辅助 PID 进行参数整定的手段。

其实在前面几节的讲述中，已经用到了专家 PID 的一些特例行为了，从第五节到第八节都是专家系统一些特列化的算法，对某些条件进行了局部的判定，比如如果偏差太大的话，就去除积分项，这本身就是含有经验的专家系统。

专家系统、模糊算法，需要参数整定就一定要有整定的依据，也就是说什么情况下整定什么值是要有依据的，这个依据是一些逻辑的组合，只要找出其中的逻辑组合关系来，这些依据就再明显不过了。下面先说一下专家 PID 的 C 语言实现。正如前面所说，需要找到一些依据，还得从 PID 系数本身说起。

1.比例系数 Kp 的作用是加快系统的响应速度，提高系统的调节精度。Kp 越大，系统的响应速度越快，系统的调节精度越高，但是容易产生超调，甚至会使系统不稳定。Kp 取值过小，则会降低调节精度，使响应速度缓慢，从而延长调节时间，是系统静态、动态特性变差；

2.积分作用系数 Ki 的作用是消除系统的稳态误差。Ki 越大，系统的静态误差消除的越快，但是 Ki 过大，在响应过程的初期会产生积分饱和的现象，从而引起响应过程的较大超调。若 Ki 过小，将使系统静态误差难以消除，影响系统的调节精度；

3.微分系数 Kd 的作用是改善系统的动态特性，其作用主要是在响应过程中抑制偏差向任何方向的变化，对偏差变化进行提前预报。但是 kd 过大，会使响应过程提前制动，从而延长调节时间，而且会降低系统的抗干扰性。

反应系统性能的两个参数是系统误差 e 和误差变化律 ec，这点还是好理解的：

首先我们规定一个误差的极限值，假设为 Mmax；规定一个误差的比较大的值，假设为 Mmid；规定一个误差的较小值，假设为 Mmin；

当 abs（e）>Mmax 时，说明误差的绝对值已经很大了，不论误差变化趋势如何，都应该考虑控制器的输入应按最大（或最小）输出，以达到迅速调整误差的效果，使误差绝对值以最大的速度减小。此时，相当于实施开环控制。

当 e*ec>0 时，说明误差在朝向误差绝对值增大的方向变化，此时，如果 abs(e)>Mmid，说明误差也较大，可考虑由控制器实施较强的控制作用，以达到扭转误差绝对值向减小的方向变化，并迅速减小误差的绝对值。此时如果 abs(e)<Mmid，说明尽管误差是向绝对值增大的方向变化，但是误差绝对值本身并不是很大，可以考虑控制器实施一般的控制作用，只需要扭转误差的变化趋势，使其向误差绝对值减小的方向变化即可。

当 e*err<0 且 e*err(k-1)>0 或者 e=0 时，说明误差的绝对值向减小的方向变化，或者已经达到平衡状态，此时保持控制器输出不变即可。

当 e*err<0 且 e*err(k-1)<0 时，说明误差处于极限状态。如果此时误差的绝对值较大，大于 Mmin，可以考虑实施较强控制作用。如果此时误差绝对值较小，可以考虑实施较弱控制作用。

当 abs(e)<Mmin 时，说明误差绝对值很小，此时加入积分，减小静态误差。

上面的逻辑判断过程，实际上就是对于控制系统的一个专家判断过程。（未完待续）

PID 十　　模糊算法简介

在 PID 控制算法的 C 语言实现九中，文章已经对模糊 PID 的实质做了一个简要说明。本来打算等到完成毕业设计，工作稳定了再着力完成剩下的部分。鉴于网友的要求和信任，抽出时间来，对模糊 PID 做一个较为详细的论述，这里我不打算做出仿真程序了，但就基本概念和思路进行一下说明，相信有 C 语言基础的朋友可以通过这些介绍性的文字自行实现。这篇文章主要说明一下模糊算法的含义和原理。

实际上模糊算法属于智能算法，智能算法也可以叫非模型算法，也就是说，当我们对于系统的模型认识不是很深刻，或者说客观的原因导致我们无法对系统的控制模型进行深入研究的时候，智能算法常常能够起到不小的作用。这点是方便理解的，如果一个系统的模型可以轻易的获得，那么就可以根据系统的模型进行模型分析，设计出适合系统模型的控制器。但是现实世界中，可以说所有的系统都是非线性的，是不可预测的。但这并不是说我们就无从建立控制器，因为，大部分的系统在一定的条件和范围内是可以抽象成为线性系统的。问题的关键是，当我们系统设计的范围超出了线性的范围，我们又该如何处理。显然，智能算法是一条很不错的途径。智能算法包含了专家系统、模糊算法、遗传算法、神经网络算法等。其实这其中的任何一种算法都可以跟 PID 去做结合，而选择的关键在于，处理的实时性能不能得到满足。当我们处理器的速度足够快速时，我们可以选择更为复杂的、精度更加高的算法。但是，控制器的处理速度限制了我们算法的选择。当然，成本是限制处理器速度最根本的原因。这个道理很简单，51 单片机和 DSP 的成本肯定大不相同。专家 PID 和模糊 PID 是常用的两种 PID 选择方式。其实，模糊 PID 适应一般的控制系统是没有问题。文章接下来将说明模糊算法的一些基本常识。

模糊算法其实并不模糊。模糊算法其实也是逐次求精的过程。这里举个例子说明。我们设计一个倒立摆系统，假如摆针偏差＜5°，我们说它的偏差比较“小”；摆针偏差在 5° 和 10° 之间，我们说它的偏差处于“中”的状态；当摆针偏差＞10° 的时候，我们说它的偏差有点儿“大”了。对于“小”、“中”、“大”这样的词汇来讲，他们是精确的表述，可问题是如果摆针偏差是 3° 呢，那么这是一种什么样的状态呢。我们可以用“很小”来表述它。如果是 7° 呢，可以说它是“中”偏“小”。那么如果到了 80° 呢，它的偏差可以说“非常大”。而我们调节的过程实际上就是让系统的偏差由非常“大”逐渐向非常“小”过度的过程。当然，我们系统这个调节过程是快速稳定的。通过上面的说明，可以认识到，其实对于每一种状态都可以划分到大、中、小三个状态当中去，只不过他们隶属的程度不太一样，比如 6° 隶属于小的程度可能是 0.3，隶属于中的程度是 0.7，隶属于大的程度是 0。这里实际上是有一个问题的，就是这个隶属的程度怎么确定？这就要求我们去设计一个隶属函数。详细内容可以查阅相关的资料，这里没有办法那么详细的说明了。
http://baike.baidu.com/view/150383.htm（见附录 3）这里面有些说明。那么，知道了隶属度的问题，就可以根据目前隶属的程度来控制电机以多大的速度和方向转动了，当然，最终的控制量肯定要落实在控制电压上。这点可以很容易的想想，我们控制的目的就是让倒立摆从隶属“大”的程度为 1 的状态，调节到隶属“小”的程度为 1 的状态。当隶属大多一些的时候，我们就加快调节的速度，当隶属小多一些的时候，我们就减慢调节的速度，进行微调。可问题是，大、中、

小的状态是汉字，怎么用数字表示，进而用程序代码表示呢？其实我们可以给大、中、小三个状态设定三个数字来表示，比如大表示用 3 表示，中用 2 表示，小用 1 表示。那么我们完全可以用 1*0.3+2*0.7+3*0.0=1.7 来表示它，当然这个公式也不一定是这样的，这个公式的设计是系统模糊化和精确化的一个过程，读者也可参见相关文献理解。但就 1.7 这个数字而言，可以说明，目前 6° 的角度偏差处于小和中之间，但是更偏向于中。我们就可以根据这个数字来调节电机的转动速度和时间了。当然，这个数字与电机转速的对应关系，也需要根据实际情况进行设计和调节。

前面一个例子已经基本上说明了模糊算法的基本原理了。可是实际上，一个系统的限制因素常常不是一个。上面的例子中，只有偏差角度成为了系统调节的参考因素。而实际系统中，比如 PID 系统，我们需要调节的是比例、积分、微分三个环节，那么这三个环节的作用就需要我们认清，也就是说，我们需要根据超调量、调节时间、震荡情况等信息来考虑对这三个环节调节的比重，输入量和输出量都不是单一的，可是其中必然有某种内在的逻辑联系。所以这种逻辑联系就成为我们设计工作的重点了。下一篇文章将详细分析 PID 三个变量和系统性能参数之间的联系。

PID 十一 （PID 系列完结篇） 模糊 PID 的参数整定

这几天一直在考虑如何能够把这一节的内容说清楚，对于 PID 而言应用并没有多大难度，按照基本的算法设计思路和成熟的参数整定方法，就算是没有经过特殊训练和培训的人，也能够在较短的时间内容学会使用 PID 算法。可问题是，如何能够透彻的理解 PID 算法，从而能够根据实际的情况设计出优秀的算法呢。

通过讲述公式和基本原理肯定是最能说明问题的，可是这样的话怕是犯了“专家”的错误了。对于门槛比较低的技术人员来讲，依然不能透彻理解。可是说的入耳了，能不能透彻说明也是一个问题，所以斟酌了几天，整理了一下思路才开始完成 PID 系列文章的最后一篇。

我所说的最后一篇不代表 PID 的功能和发展就止步与此，仅仅是说明，透过这一些列的文章，基本上已经可以涵盖 PID 设计的要点，至于更深入的研究，就交给有需要的读者去做。

上一节中大致讲述了一下模糊算法。实际上模糊算法的很多概念在上一节中并没有深入的解释。举的例子也只是为了说明模糊算法的基本含义，真正的模糊算法是不能这么设计的，当然也不会这么简单。模糊算法的核心是模糊规则，如果模糊规则制定的出色，那么模糊算法的控制效率就高。其实这是智能算法的一般特性，规则是系统判断和处理的前提。那么就说说 PID 的规则该怎么制定。

我们知道，模糊算法的本质是对 PID 的三个参数进行智能调节。那么首先要提出的问题是如何对 PID 的参数进行调节？这个问题其实是参数整定的问题，现实当中有很多整定方法。可是我们需要从根本上了解为什么这么整定，才能知道该如何建立数学模型进行分析。那么要回答如何整定参数的问题，就需要先明白 PID 参数的作用都是什么？对系统有什么影响？

我们从作用和副作用两个方面说明参数对系统的影响。

1.比例环节 Kp，作用是加快系统的响应速度，提高系统的调节精度，副作用是会导致超调;

2.积分环节 Ki，作用是消除稳态误差，副作用是导致积分饱和现象;

3.微分环节 Kd，作用是改善系统的动态性能，副作用是延长系统的调节时间。

理解了上述问题，那么就可以“辩证施治，对症下药”了。比如说，如果系统响应速度慢，我们就加大 Kp 的取值，如果超调量过大我们就减小 Kp 的取值等等。可是问题这些语言的描述该如何用数学形式表达出来呢。我们所知道的，反馈系统的实质就是系统的输出量作为反馈量与系统的输入量进行作差，从而得到系统的误差 e，那么这个误差 e 就能够反应目前系统所处的状态。误差 e 可以表明目前系统的输出状态到底偏离要求多少。而误差 e 的变化律 ec，表示误差变化的速度。这样，我们可以根据这两个量的状态来分析三个参数此时应该如何取值，假如 e 为负方向比较大，ec 也为负方向增大状态，此时比例环节要大一些，从而加快调节速度，而积分环节要小一些，甚至不加积分环节，从而防止负方向上出现饱和积分的现象。微分环节可以稍加一些，在不影响调节时间的情况下，起到改善系统动态性能的作用。

附录 1

看到有不少人问到底如何让 UK 值与 PWM 占空比值对应，进而实现占空比输出和输出控制电压对应。

（注意，我这里讨论的前提是输出控制的是电压，不是 PWM 方波。PWM 输出后要经过滤波整形再输出控制。）

前提条件：

输出电压控制电压范围是 0-10V。

给定、反馈、输出电压采样输入电压范围是 0-5V（经过运放）。

使用单片机 AD 为 10 位 AD 芯片。

那么 10 位 AD 芯片电压采集得到的数据范围就是 0-1024。

PWM 为 8 位可调占空比方波，0 对应输出占空比为 0 的方波，255 对应输出占空比 100%的方波，127 对应输出 50%的方波。

比如当前给定是 2.5V，反馈电压是 1V。（KP,KI,KD 等系数略，关于 PID 算法的整数实现我在前文中有论述如何实现）。

那么经过 AD 采样

1、给定 2.5V 对应为 512

2、反馈 1V 对应为 205

假定经过 PID 计算得到的 UK 为 400

也就意味着输出电压应当为（400*（UPWM 峰值电压））/1024

那么 UK 对应的 PWM 占空比是多少呢？

我们知道，UK=1024 对应占空比为 100，也就是 PWM 的占空比系数为 255。可知，PWM 系数 = UK/4;

那么 400 就应当对应系数 400/4=100。

也就是输出电压=400*10/1024=3.9V

同时，由于采样精度以及 PWM 输出占空比精度控制的问题，将导致输出电压和期望值不是那么线性，所以，我在项目内加入了输出电压采样的控制。

采样 AD 输入为 0-5V，所以，对于输出 0-10V 有一个缩小的比例。

输出 10V 则采样值对应为 255

输出 5V 则采样之对应 127

可知，3.9V 对应 AD 结果为 97

采样输出电压值，可以针对性的调整一下占空比输出，从而得到误差允许范围内的一个控制输出电压。

同时，经过一些加速控制的手段。可以比较迅速的达到控制的目的。

下文中的 UK 控制方法是针对增量式 PID 控制而来做的。

```
/*****************************************************/

void    PWMProcess(void)

{

    uint16 idata temp;

    uint16 idata UKTemp;

 temp = 0;

    UKTemp = 0;

    if( Pwm.ChangeFlag_Uint8 != 0 )   //判断是否需要改变占空比

 {                    //是否需要改变占空比和你的被控系统特性有关

     Pwm.ChangeFlag_Uint8 = 0;

  UKTemp = PID.Uk_Uint16 + SwIn.AddValue_Uint16;

 //计算 UK 控制量

 //控制量和计算值以及一个开关量有关，我这里的开关量是系统需要的时候叠加在控制量上的一个变量。

  if(UKTemp>999)

  {

            UKTemp = 999;

  }
```

//这里只所以是 999 封顶而不是 1024 是因为我的系统 PWM 的峰值电压是 12V 导致。

```
        while(1)                          //如果输出电压和期望电压相差 Delta，则继续调整占空比，直到在误差以内

        {

      ADChPro(UPWMADCH);          //测量输出电压

   if( ADPool.Value_Uint16[UPWMADCH] == UKTemp)
```

```
    {
                return;
    }
    if( ADPool.Value_Uint16[UPWMADCH] > UKTemp)   //如果当前电压大于输出电压，减小占空比
    {
        if( ( ADPool.Value_Uint16[UPWMADCH] - UKTemp ) > UDELTA )
     {
         temp = ADPool.Value_Uint16[UPWMADCH] - UKTemp;  //
      temp = temp / 2;        //下降可以加速下降，所以下降参数加倍
      if( Pwm.DutyCycle_Uint8 > temp )
      {
                        Pwm.DutyCycle_Uint8 = Pwm.DutyCycle_Uint8 - temp;
      }
      else
      {
                        Pwm.DutyCycle_Uint8 = 0;
      }
     }
     else
     {
                    return;
     }
    }
    else            //如果当前电压小于输出电压
    {
        if( ( UKTemp - ADPool.Value_Uint16[UPWMADCH] ) > UDELTA )
```

```
    {
        temp = UKTemp - ADPool.Value_Uint16[UPWMADCH];
     temp = temp / 4;  //上升处理不要超调，所以每次只+一半
     if( (255-Pwm.DutyCycle_Uint8) > temp )
     {
                    Pwm.DutyCycle_Uint8 += (temp/2);
     }
     else
     {
                    Pwm.DutyCycle_Uint8 = 255;
     }
    }
    else
    {
                return;
    }
   }
          DisPlayVoltage();
          PWMChangeDuty(Pwm.DutyCycle_Uint8);  //改变占空比
   Delay(10,10);

       }
 }
}
/*****************************************************/
```

附录 2

直流电机 PWM 调速系统中控制电压非线性研究

引言

由于线性放大驱动方式效率和散热问题严重，目前绝大多数直流电动机采用开关驱动方式。开关驱动方式是半导体功率器件工作在开关状态，通过脉宽调制 PWM 控制电动机电枢电压，实现调速。目前已有许多文献介绍直流电机调速，宋卫国等用 89C51 单片机实现了直流电机闭环调速；张立勋等用 AVR 单片机实现了直流电机 PWM 调速；郭崇军等用 C8051 实现了无刷直流电机控制；张红娟等用 PIC 单片机实现了直流电机 PWM 调速；王晨阳等用 DSP 实现了无刷直流电机控制。上述文献对实现调速的硬件电路和软件流程的设计有较详细的描述，但没有说明具体的调压调速方法，也没有提及占空比与电机端电压平均值之间的关系。在李维军等基于单片机用软件实现直流电机 PWM 调速系统中提到平均速度与占空比并不是严格的线性关系，在一般的应用中，可以将其近似地看作线性关系。但没有做深入的研究。本文通过实验验证，在不带电机情况下，PWM 波占空比与控制输出端电压平均值之间呈线性关系；在带电机情况下，占空比与电机端电压平均值满足抛物线方程，能取得精确的控制。本文的电机闭环调速是运用 Matlab 拟合的关系式通过 PID 控制算法实现。

1 系统硬件设计

本系统是基于 TX-1C 实验板上的 AT89C52 单片机，调速系统的硬件原理图如图 1 所示，主要由 AT89C52 单片机、555 振荡电路、L298 驱动电路、光电隔离、霍尔元件测速电路、MAX 232 电平转换电路等组成。

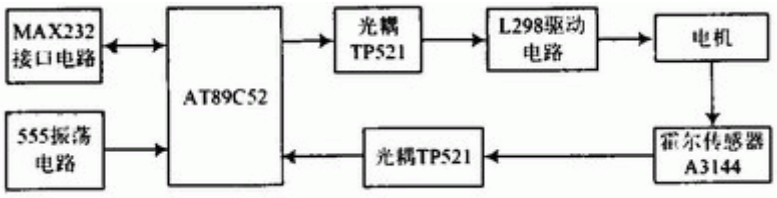


图 1 闭环控制系统示意图

2 系统软件设计

系统采用模块化设计，软件由 1 个主程序，3 个中断子程序，即外部中断 0、外部中断 1，定时器 0 子程序，PID 算法子程序，测速子程序及发送数据到串口显示子程序组成，主程序流程图如图 2 所示。外部中断 0 通过比较直流电平与锯齿波信号产生 PWM 波，外部中断 1 用于对传感器的脉冲计数。定时器 0 用于对计数脉冲定时。测得的转速通过串口发送到上位机显示，通过 PID 模块调整转速到设定值。本实验采用 M / T 法测速，它是同时测量检测时间和在此检测时间内霍尔传感器所产生的转速脉冲信号的个数来确定转速。由外部中断 1 对霍尔传感器脉冲计数，同时起动定时器 0，当计数个数到预定值 2 000 后，关定时器 0，可得到计 2 000 个脉冲的计数时间，由式计算出转速：

$$n=60f / K=60N / (KT) \quad (1)$$

式中：n 为直流电机的转速；K 为霍尔传感器转盘上磁钢数；f 为脉冲频率；N 为脉冲个数；T 为采样周期。

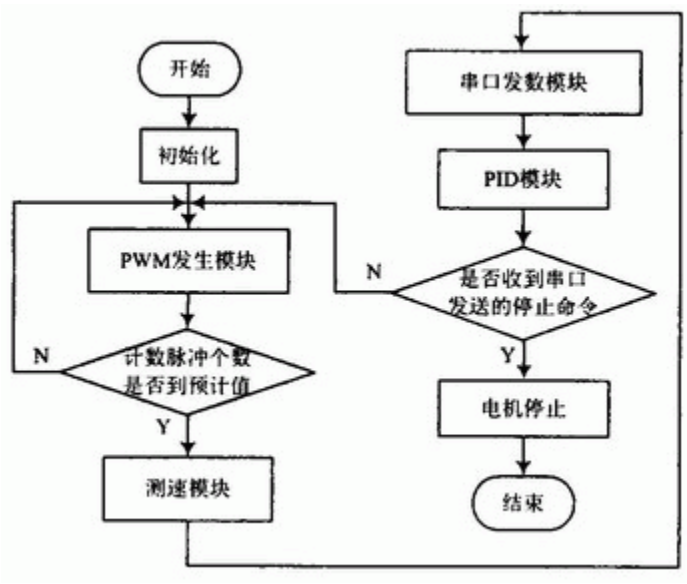


图 2 主程序流程图

3 实验结果及原因分析

3．1 端电压平均值与转速关系

3．1．1 实验结果

实验用的是永磁稳速直流电机，型号是 EG-530YD-2BH，额定转速 2 000～4 000 r／min，额定电压 12 V。电机在空载的情况下，测得的数据用 Matlab 做一次线性拟合，拟合的端电压平均值与转速关系曲线如图 3(a)所示。相关系数 R-square：0．952 1。拟合曲线方程为：

$$y=0.001\ 852x+0.296\ 3 \quad (2)$$

由式(2)可知，端电压平均值与转速可近似为线性关系，根据此关系式，在已测得的转速的情况下可以计算出当前电压。为了比较分析，同样用 Matlab 做二次线性拟合，拟合的端电压平均值与转速关系曲线如图 3(b)所示。相关系数 R-square：0．986 7。

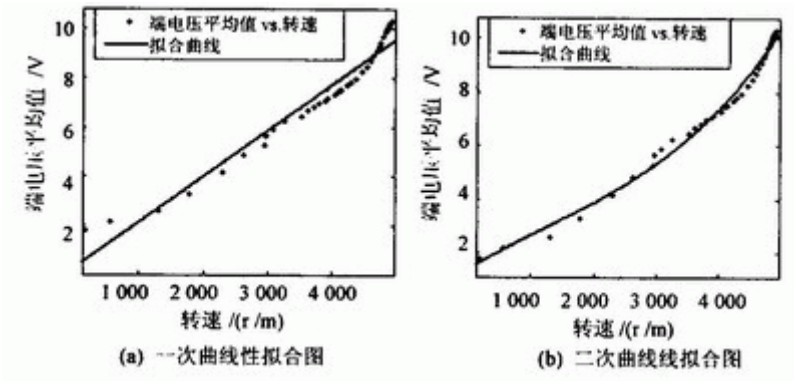


图 3 端电压平均值与转速关系曲线图

3．1．2 原因分析

比较图 3(a)可知，当转速在 0～1 500 r / min 和 4 000～5 000 r / min，端电压平均值与转速间存在的非线性，用二次曲拟合如图 3(b)所示，拟合相关系数较高。由图 3(a)可见，当电机转速为 0 时电机两端电压平均值约为 1．3 V。这是因为电机处于静止状态时，摩擦力为静摩擦力，静摩擦力是非线性的。随着外力的增加而增加，最大值发生在运动前的瞬间。电磁转矩为负载制动转矩和空载制动转矩之和，由于本系统不带负载，因此电磁转矩为空载制动转矩。空载制动转矩与转速之间此时是非线性的。电磁转矩与电流成正比，电流又与电压成正比，因此此时电压与转速之间是非线性的。

当转速在 2 000～4 000 r / min 线性关系较好，占空比的微小改变带来的转速改变较大，因此具有较好的调速性能。这是因为随着运动速度的增加，摩擦力成线性的增加，此时的摩擦力为粘性摩擦力。粘性摩擦是线性的，与速度成正比，空载制动转矩与速度成正比，也即电磁转矩与电流成正比，电流又与电压成正比，因此此时电压与转速之间是线性的。当转速大于 4 000 r / min。由于超出了额定转速所以线性度较差且调速性能较差。此时用二次曲线拟合结果较好，因为当电机高速旋转时，摩擦阻力小到可以忽略，此时主要受电机风阻型负荷的影响，当运动部件在气体或液体中运动时，其受到的摩擦阻力或摩擦阻力矩被称为风机型负荷。对同一物体，风阻系数一般为固定值。阻力大小与速度的平方成正比。即空载制动转矩与速度的平方成正比，也即电磁转矩与速度的平方成正比，电磁转矩与电流成正比，电流又与电压成正比，因此此时电压与转速之间是非线性的。

3．2 占空比与端电压平均值关系

3．2．1 实验结果

拟合占空比与端电压平均值关系曲线如图 4 所示。相关系数 R-square：0．998 4。拟合曲线方程为：

$$y = 0.000\ 645\ 3x^{5.081} + 5.79 \qquad (3)$$

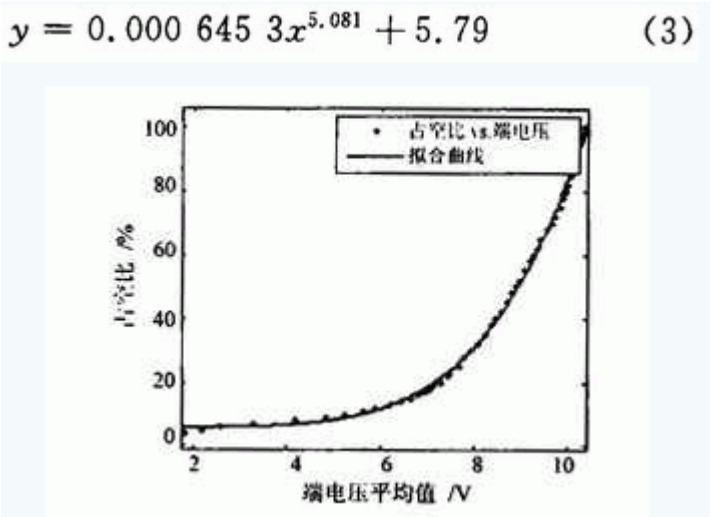


图 4 占空比与端电压平均值关系曲线图

如图 4 所示，占空比与端电压平均值满足抛物线方程。运用积分分离的 PID 算法改变电机端电压平均值，可以运用此关系式改变占空比，从而实现了 PWM 调速。

用示波器分别测出电压的顶端值 Utop 与底端值 Ubase，端电压平均值 Uarg 满足关系式：

$$U_{arg} = U_{base} + \alpha(U_{top} - U_{base}) \qquad (4)$$

其中：α 为占空比。

正是由于所测得的电机端电压底端值 Ubase 不为 0，所以得出的占空比与端电压平均值之间关系曲线为抛物线。若将电机取下，直接测 L298 的 out1 与 out2 输出电压。所测得的电机端电压底端值 Ubase 约为 0，所得的占空比与端电压平均值满足线性关系，即令式(4)中 Ubase 为 0，式(4)变为：

$$U_{arg} = \alpha U_{top} \qquad (5)$$

3．2．2 原因分析

将电机取下后，直接测 L298 的输出端之间的电压，占空比与端电压平均值满足关系式(5)，说明整个硬件电路的设计以及软件编程的正确性。从电机反电势角度分析，当直流电机旋转时，电枢导体切割气隙磁场，在电枢绕组中产生感应电动势。由于感应电动势方向与电流的方向相反，感应电动势也即反电势。直流电机的等效模型如图 5 所示。图 5(a)表示电机工作在电动机状态。图 5(b)表示电机工作在发电机状态。

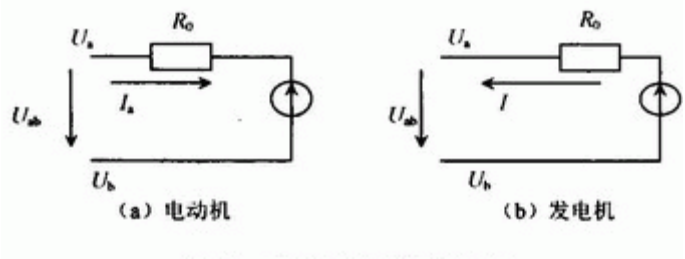


图 5 直流电机等效电路

如图 5(a)所示，电压平衡方程为：

$$U = E_a + I_a R_a + 2\Delta u_b \qquad (6)$$

式中：U 为外加电压；Ia 为电枢电流；Ra 为电枢绕组电阻；2△Ub 为一对电刷接触压降，一般取 2△Ub 为 0．5～2 V；Ea 为电枢绕组内的感应电动势。电机空载时，电枢电流可忽略不计，即电流 Ia 为 0。空载时的磁场由主磁极的励磁磁动势单独作用产生。给电机外加 12 V 的额定电压，由(6)可得反电势：

$$E_a = U - 2\Delta U_b \qquad (7)$$

以 40%的占空比为例，电机端电压 Uab 是测量中的电压平均值 Uarg，其值为 8．34 V，测量中的电压底端值 Ubase 约为 7 V。由式(7)可得 Ea 的值范围应在 6．34～7．84 V。由图 5(b)可见，此时 Uab 的值是测得的底端值 Ubase 即电机的电动势 Ea 为 7 V。

当 PWM 工作在低电平状态，直流电机不会立刻停止，会继续旋转，电枢绕组切割气隙磁场，电机此时工作在发电机状态，产生感应电动势 E。

$$E = C_e \Phi_n \qquad (8)$$

式中：Ce 为电机电动势常数；Φ 为每级磁通量。由于电机空载，所以图 5(b)中无法形成回路。用单片机仿真软件 Proteus 可直观的看出在 PWM 为低电平状态，电机处于减速状态。低电平持续时间越长，电机减速量越大。正是由于在低电平期间，电机处于减速状态，由式(8)可知，Ce，Φ 均为不变量，转速 n 的变化引起 E 的改变。此时 Uab 的值等于 E 的值。电机在低电平期间不断的减速，由于 PWM 周期较短，本文中取 20 ms，电机在低电平期间转速还未减至 0，PWM 又变为高电平了。这样，就使测得的 Ubase 值不为 0。以 40%的占空比为例，当 PWM 工作在低电平状态，测得 Ubase 的值约为 7 V。由式(8)可知，当正占空比越大，转速也就越大，同时减速时间越短，感应电势 E 的值越大。所以 Ubase 的值也就越大。

4 结语

重点分析了直流电机 PWM 调速过程中控制电压的非线性，对非线性的影响因素做了详细的分析。由于 PWM 在低电平期间电压的底端值不为 0，导致了占空比与电机端电压平均值之间呈抛物线关系。因此，可用得出的抛物线关系式实现精确调速。本系统的非线性研究可为电机控制中非线性的进一步研究提供依据，在实际运用中，可用于移动机器人、飞行模拟机的精确控制。

附录 3

隶属函数(membership function)，用于表征模糊集合的数学工具。对于普通集合 A，它可以理解为某个论域 U 上的一个子集。为了描述论域 U 中任一元素 u 是否属于集合 A，通常可以用 0 或 1 标志。用 0 表示 u 不属于 A，而用 1 表示属于 A ，从而得到了 U 上的一个二值函数 x A（u），它表征了 U 的元素 u 对普通集合的从属关系，通常称为 A 的特征函数，为了描述元素 u 对 U 上的一个模糊集合的隶属关系，由于这种关系的不分明性，它将用从区间[0，1]中所取的数值代替 0，1 这两值来描述，记为（u），数值（u）表示元素隶属于模糊集的程度，论域 U 上的函数 μ 即为模糊集的隶属函数，而（u）即为 u 对 A 的隶属度。