PID 算法原理、调试经验以及代码

前言：下班前收到网友的短信，和我说正在做 PID 的程序，感觉有些困难。我刚好曾经做过 PID 算法，故将自己以前的论文和代码做了整理，写了这篇文章，希望能够对用到 PID 调节器的朋友们有所帮助。也希望更多的朋友能够分享你们的经验，因为我们相信成功源于共享。

1。PID 原理

电池充放电系统中的控制器，根据给定信号和反馈信号相减得到的偏差信号来计算控制量 u，从而控制功率管的占空比 D。从式(4-35)中可知，在 PWM 的频率不变的情况下，即周期寄存器 T_xPR 的值不变的情况下，由控制量 u 改变比较寄存器 T_xCMPR 的值便可以改变功率管的占空比 D。在自动控制系统中，常用的控制器有比例-积分控制器（PI 控制器）、比例-积分-微分控制器（PID 控制器）、分段逼近式控制器，较为新颖的有模糊控制器，神经元网络控制器等，本系统使用的是工业过程控制中广泛应用的 PID 控制器。

按偏差的比例、积分、微分进行控制的控制器称为 PID 控制器。模拟 PID 控制器的原理框图如图 4-7 所示，其中 r(t)为系统给定值，c(t)为实际输出，u(t)为控制量。PID 控制解决了自动控制理论所要解决的最为基本的问题，即系统的稳定性、快速性和准确性。调节 PID 的参数，可以实现在系统稳定的前提下，兼顾系统的带载能力和抗扰能力，同时由于在 PID 控制器中引入了积分项，系统增加了一个零积点，这样系统阶跃响应的稳态误差就为零。

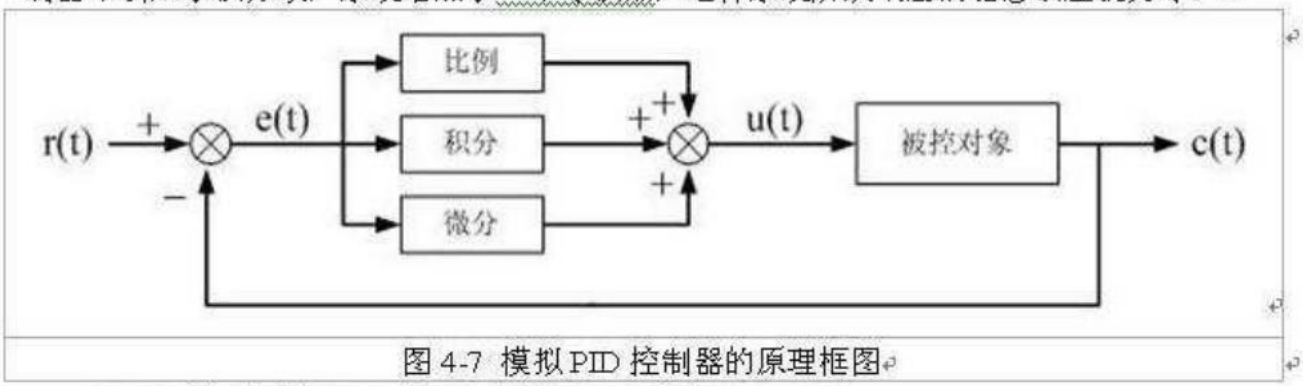

图 4-7 模拟 PID 控制器的原理框图

图 4-7 所示的模拟 PID 控制器的控制表达式为：

$$u(t) = k_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}] \qquad (4\text{-}36)$$

式中，e(t)为系统偏差，e(t)＝r(t)-c(t)；

式中，e(t)为系统偏差，e(t)＝r(t)-c(t)；

k_p为比例系数；

T_i为积分时间常数；

T_d为微分时间常数。

式(4-36)也可以写成：

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt} \quad (4\text{-}37)$$

式中，k_p为比例系数；

k_i为积分系数，$k_i = k_p/T_i$；

k_d为微分系数，$k_d = k_p T_d$。

简单说来，PID 控制器中各校正环节的作用如下[42]：

（1）比例环节 及时成比例地反映控制系统的偏差信号 e(t)，偏差一旦产生，控制器立即产生调节作用，以减少偏差。

（2）积分环节 主要用于消除静差提高系统的无差度。积分作用的强弱取决于积分时间常数 T_i，T_i越大，积分作用越弱，反之则越强。

（3）微分环节 能够反映偏差信号的变化趋势，即偏差信号的变化速率，并能在偏差信号值变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。

计算机控制是一种离散的采样控制，在计算机控制系统中所使用的是数字 PID 控制器，而式(4-36)和式(4-37)均为模拟 PID 控制器的控制表达式。通过将模拟 PID 表达式中的积分、微分运算用数值计算方法来逼近，便可实现数字 PID 控制。只要采样周期 T 取得足够小，这种逼近就可以相当精确。

将微分项用差分代替，积分项用矩形和式代替，数字 PID 控制器的控制表达式如式(4-38)

$$u(k)=k_p\left\{e(k)+\frac{T}{T_i}\sum_{j=0}^{k}e(j)+\frac{T_d}{T}\left[e(k)-e(k-1)\right]\right\} \quad (4\text{-}38)$$

同样的，式(4-38)也可以写成：

$$u(k)=k_p e(k)+k_i\sum_{j=0}^{k}e(j)+k_d\left[e(k)-e(k-1)\right] \quad (4\text{-}39)$$

其中：$k_i=k_pT/T_i$，$k_d=k_pT_d/T$。

数字 PID 控制器的控制算法通常可以分为位置式 PID 控制算法和增量式 PID 控制算法，本系统使用的是位置式 PID 控制算法，因此下面将讨论如何建立位置式 PID 控制算法的数学模型。

由式(4-39)可得，第 k-1 时刻 PID 调节的表达式为：

$$u(k-1)=k_p e(k-1)+k_i\sum_{j=0}^{k-1}e(j)+k_d\left[e(k-1)-e(k-2)\right] \quad (4\text{-}40)$$

将式(4-39)减式(4-40)，便可得到位置式 PID 控制算法的表达式为：

$$u(k)=u(k-1)+k_p[e(k)-e(k-1)]+k_i e(k)+k_d[e(k)-2e(k-1)+e(k-2)]$$

为了使表达式更为简单，可以将上面的式子展开，合并同类项后可以得到：

$$u(k)=u(k-1)+a_0e(k)+a_1e(k-1)+a_2e(k-2) \quad (4\text{-}41)$$

其中：$a_0=k_p+k_i+k_d=k_p[1+T/T_i+T_d/T]$；

$a_1=-k_p-2k_d=-k_p[1+2T_d/T]$；

$a_2=k_d=k_pT_d/T$。

式(4-41)即为本系统所使用的位置式 PID 控制器的数学模型。

2. 流程图

PID 调节子程序的流程图如图 4-10 所示。当进入 PID 调节子程序时，首先需要根据系统给定值和采样值来计算偏差。为了防止在系统运行初期，由于控制量 u(k)过大使得开关管占空比 D 过大，需要对代入式(4-41)运算的 e(k)做一定的限幅处理。因为瞬间过大的占空比有时候可能会引起过大的电流，从而导致开关管的损坏。另外，在系统进入稳态后，偏差是很小的，如果偏差在一个很小的范围内波动，控制器对这样微小的偏差计算后，将会输出一个微小的控制量，此时输出的控制值在一个很小的范围内，不断改变自己的方向，频繁动作，发生振颤，这样不利于正在充电的蓄电池。因此，当控制过程进入这种状态时，就进入系统设定的一个输出允许带 e_0，即当采集到的偏差 $|e(k)|<e_0$ 时，不改变控制量，使充放电过程能够稳定的进行。

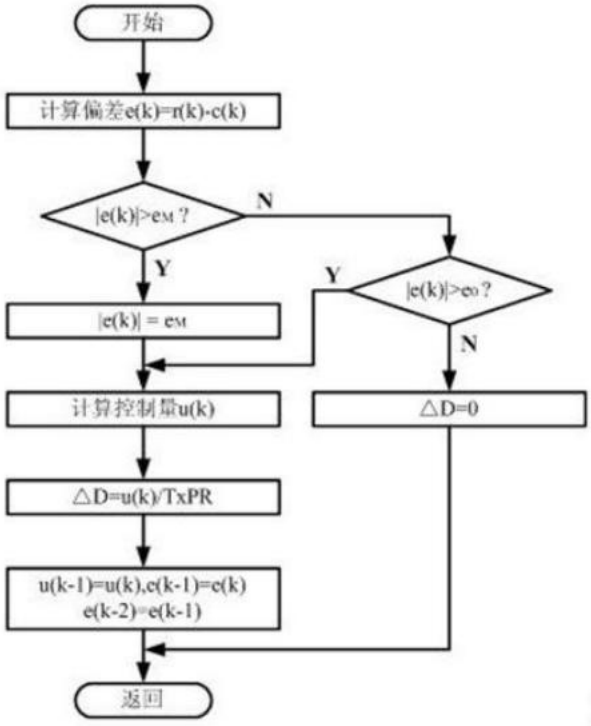

图 4-10 PID 调节子程序流程图

式中，e(t)为系统偏差，e(t)＝r(t)-c(t)；

k_p为比例系数；

T_i为积分时间常数；

T_d为微分时间常数。

式(4-36)也可以写成：

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt} \qquad (4\text{-}37)$$

式中，k_p为比例系数；

k_i为积分系数，$k_i = k_p/T_i$；

k_d为微分系数，$k_d = k_p T_d$。

简单说来，PID 控制器中各校正环节的作用如下[42]：

（1）比例环节 及时成比例地反映控制系统的偏差信号 e(t)，偏差一旦产生，控制器立即产生调节作用，以减少偏差。

（2）积分环节 主要用于消除静差提高系统的无差度。积分作用的强弱取决于积分时间常数 T_i，T_i越大，积分作用越弱，反之则越强。

（3）微分环节 能够反映偏差信号的变化趋势，即偏差信号的变化速率，并能在偏差信号值变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。

计算机控制是一种离散的采样控制，在计算机控制系统中所使用的是数字 PID 控制器，而式(4-36)和式(4-37)均为模拟 PID 控制器的控制表达式。通过将模拟 PID 表达式中的积分、微分运算用数值计算方法来逼近，便可实现数字 PID 控制。只要采样周期 T 取得足够小，这种逼近就可以相当精确。

将微分项用差分代替，积分项用矩形和式代替，数字 PID 控制器的控制表达式如式(4-38)

$$u(k)=k_p\left\{e(k)+\frac{T}{T_i}\sum_{j=0}^{k}e(j)+\frac{T_d}{T}\left[e(k)-e(k-1)\right]\right\} \quad (4\text{-}38)$$

同样的，式(4-38)也可以写成：

$$u(k)=k_p e(k)+k_i\sum_{j=0}^{k}e(j)+k_d\left[e(k)-e(k-1)\right] \quad (4\text{-}39)$$

其中：$k_i=k_pT/T_i$，$k_d=k_pT_d/T$。

数字 PID 控制器的控制算法通常可以分为位置式 PID 控制算法和增量式 PID 控制算法，本系统使用的是位置式 PID 控制算法，因此下面将讨论如何建立位置式 PID 控制算法的数学模型。

由式(4-39)可得，第 k-1 时刻 PID 调节的表达式为：

$$u(k-1)=k_p e(k-1)+k_i\sum_{j=0}^{k-1}e(j)+k_d\left[e(k-1)-e(k-2)\right] \quad (4\text{-}40)$$

将式(4-39)减式(4-40)，便可得到位置式 PID 控制算法的表达式为：

$$u(k)=u(k-1)+k_p[e(k)-e(k-1)]+k_i e(k)+k_d[e(k)-2e(k-1)+e(k-2)]$$

为了使表达式更为简单，可以将上面的式子展开，合并同类项后可以得到：

$$u(k)=u(k-1)+a_0e(k)+a_1e(k-1)+a_2e(k-2) \quad (4\text{-}41)$$

其中：$a_0=k_p+k_i+k_d=k_p[1+T/T_i+T_d/T]$；

$a_1=-k_p-2k_d=-k_p[1+2\,T_d/T]$；

$a_2=k_d=k_p\,T_d/T$。

式(4-41)即为本系统所使用的位置式 PID 控制器的数学模型。

2. 流程图

PID 调节子程序的流程图如图 4-10 所示。当进入 PID 调节子程序时，首先需要根据系统给定值和采样值来计算偏差。为了防止在系统运行初期，由于控制量 u(k)过大使得开关管占空比 D 过大，需要对代入式(4-41)运算的 e(k)做一定的限幅处理。因为瞬间过大的占空比有时候可能会引起过大的电流，从而导致开关管的损坏。另外，在系统进入稳态后，偏差是很小的，如果偏差在一个很小的范围内波动，控制器对这样微小的偏差计算后，将会输出一个微小的控制量，此时输出的控制值在一个很小的范围内，不断改变自己的方向，频繁动作，发生振颤，这样不利于正在充电的蓄电池。因此，当控制过程进入这种状态时，就进入系统设定的一个输出允许带 e_0，即当采集到的偏差 $|e(k)|<e_0$ 时，不改变控制量，使充放电过程能够稳定的进行。

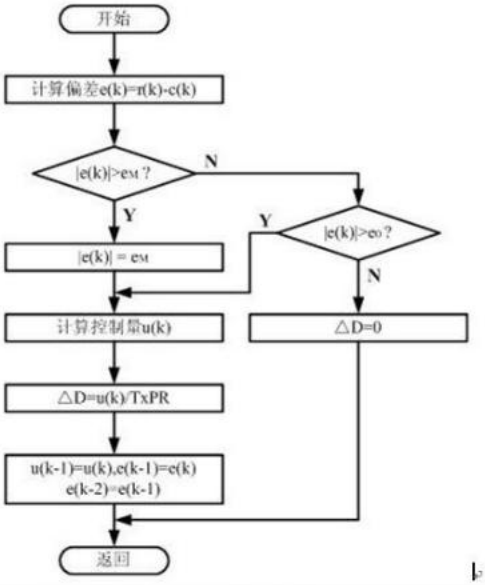

图 4-10 PID 调节子程序流程图

2#
发表于 2007-11-2 22:21
3。PID 代码

```
//定义变量
float Kp;                       //PI 调节的比例常数
float Ti;                       //PI 调节的积分常数
float T;                        //采样周期
float Ki;
float ek;                       //偏差 e[k]
float ek1;                      //偏差 e[k-1]
float ek2;                      //偏差 e[k-2]
float uk;                       //u[k]
signed int uk1;                 //对 u[k]四舍五入取整
signed int adjust;              //调节器输出调整量

//变量初始化
Kp=4;
Ti=0。005;
T=0.001;
// Ki=KpT/Ti=0.8，微分系数 Kd=KpTd/T=0.8,Td=0.0002,根据实验调得的结果确定这些参数
ek=0;
ek1=0;
ek2=0;
uk=0;
uk1=0;
adjust=0;

int piadjust(float ek)  //PI 调节算法
{
if( gabs(ek)<0.1 )
{
adjust=0;
}
else
{
uk=Kp*(ek-ek1)+Ki*ek;  //计算控制增量
ek1=ek;

uk1=(signed int)uk;
if(uk>0)
{
if(uk-uk1>=0.5)
{
uk1=uk1+1;
}
}
if(uk<0)
{
if(uk1-uk>=0.5)
{
uk1=uk1-1;
}
}
adjust=uk1;
}


return adjust;
}
```

下面是在 AD 中断程序中调用的代码。

```
。。。。。。。。。。
else //退出软启动后，PID 调节,20ms 调节一次
{
```

```
EvaRegs.CMPR3=EvaRegs.CMPR3+piadjust(ek);//误差较小 PID 调节稳住
if(EvaRegs.CMPR3>=890)
{
EvaRegs.CMPR3=890; //限制 PWM 占空比
}
}
。。。。。。。。。。。。。。。。
```

4。PID 调节经验总结

PID 控制器参数选择的方法很多，例如试凑法、临界比例度法、扩充临界比例度法等。但是，对于 PID 控制而言，参数的选择始终是一件非常烦杂的工作，需要经过不断的调整才能得到较为满意的控制效果。依据经验，一般 PID 参数确定的步骤如下[42]：

(1) 确定比例系数 Kp

确定比例系数 Kp 时，首先去掉 PID 的积分项和微分项，可以令 Ti=0、Td=0，使之成为纯比例调节。输入设定为系统允许输出最大值的 60%～70%，比例系数 Kp 由 0 开始逐渐增大，直至系统出现振荡；再反过来，从此时的比例系数 Kp 逐渐减小，直至系统振荡消失。记录此时的比例系数 Kp，设定 PID 的比例系数 Kp 为当前值的 60%～70%。

(2) 确定积分时间常数 Ti

比例系数 Kp 确定之后，设定一个较大的积分时间常数 Ti，然后逐渐减小 Ti，直至系统出现振荡，然后再反过来，逐渐增大 Ti，直至系统振荡消失。记录此时的 Ti，设定 PID 的积分时间常数 Ti 为当前值的 150%～180%。

(3) 确定微分时间常数 Td

微分时间常数 Td 一般不用设定，为 0 即可，此时 PID 调节转换为 PI 调节。如果需要设定，则与确定 Kp 的方法相同，取不振荡时其值的 30%。

(4) 系统空载、带载联调

对 PID 参数进行微调，直到满足性能要求。

上面的实际是 PI 调节的代码，现附上 PID 的。

```
1.  //声明变量
2.
3.  //定义变量
4.  float Kp;                    //PID 调节的比例常数
5.  float Ti;                    //PID 调节的积分常数
6.  float T;                     //采样周期
7.  float Td;                    //PID 调节的微分时间常数
8.  float a0;
9.  float a1;
10. float a2;
11.
12. float ek;                    //偏差 e[k]
13. float ek1;                   //偏差 e[k-1]
14. float ek2;                   //偏差 e[k-2]
15. float uk;                    //u[k]
16. int uk1;                    //对 uk 四舍五入求整
17. int adjust;                 //最终输出的调整量
18.
19. //变量初始化，根据实际情况初始化
20.     Kp=;
21.     Ti=;
22.     T=;
23.         Td=;
24.
25.         a0=Kp*(1+T/Ti+Td/T);
26.         a1=-Kp*(1+2*Td/T);
27.         a2=Kp*Td/T;
28. // Ki=KpT/Ti=0.8，微分系数 Kd=KpTd/T=0.8,Td=0.0002,根据实验调得的结果确定这些参数
29.     ek=0;
30.     ek1=0;
31.     ek2=0;
32.     uk=0;
33.         uk1=0;
34.         adjust=0;
35.
36.
37. int pid(float ek)
38. {
```

```
39.       if(gabs(ek)<ee) //ee 为误差的阀值，小于这个数值的时候，不做 PID 调整，避免误差较小时频繁调节引起震荡。
ee 的值可自己设
40.           {
41.                   adjust=0;
42.           }
43.       else
44.           {
45.                   uk=a0*ek+a1*ek1+a2*ek2;
46.                   ek2=ek1;
47.                   ek1=ek;
48.                uk1=(int)uk;
49.
50.                if(uk>0)
51.                    {
52.               if(uk-uk1>=0.5)
53.                       {
54.                  uk1=uk1+1;
55.                       }
56.                    }
57.           if(uk<0)
58.               {
59.              if(uk1-uk>=0.5)
60.                       {
61.                 uk1=uk1-1;
62.                       }
63.               }
64.
65.               adjust=uk1;
66.       }
67.           return adjust;
68.
69.   }
70.
71.   float gabs(float ek)
72.   {
73.           if(ek<0)
74.           {
75.                   ek=0-ek;
76.           }
77.           return ek;
78.   }
```

复制代码

3#
发表于 2007-11-3 09:06
楼主真是大大大大大大的好人啊！！！！真的是超级感谢你啊！

这篇文章我收下了！！！

不过在下还有一个不情之请，不知道楼主能不能吧 PID 的程序上传一下撒！

如果不方便的话，也没有关系！我会努力，会加油的！真的是再次谢谢你了！！！

还有其他问题的时候，还要麻烦楼主啊！！！

4#
发表于 2007-11-3 10:33
PID 并不复杂，程序代码我已经写出来了，因为这是整个程序里面的一小段，所以也不好全贴出来了。PID 的关键我觉得在于参数的调试，需要不断的做实验，分析，再调整，再做实验，最终选择一组适合自己项目的参数。我自己写过一个 PID 的仿真软件，不过是用于开关电源中 BUCK 电路的 PID 仿真软件，最近整理之后，拿出来给大家，有需要的朋友敬请期待哦，呵呵。

也希望您能够多多支持我们HELLODSP，陪伴我们一起成长！尽自己的力量去帮助更多需要帮助的朋友们

5#
发表于 2007-11-3 11:02
呵呵，谢谢，版主！
我们期待这你的成果，希望像你这样好心的前辈能有取得更大的成绩，帮助我们。
我会尽自己的努力的，会继续支持HELLODSP的！

6#
发表于 2007-11-3 14:17
"不过是用于开关电源中BUCK电路的PID仿真软件"
"不过是用于开关电源中BUCK电路的PID仿真软件"
我期待着这个地出现，谢谢楼主

7#
发表于 2007-11-3 19:32
谢谢分享你的成果
致敬！

8#
发表于 2007-11-4 23:21
PID仿真软件已经发布了，需要的朋友点击下面的页面去看看吧：

＝＝PID控制器仿真软件【基于BUCK电路】＝＝

9#
发表于 2007-11-5 12:54
一定仔细研究，谢谢哈

10#
发表于 2007-11-5 14:35
好帖子,谢谢楼主!!!

11#
发表于 2007-11-5 20:19
原创！好东东，收藏！

12#
发表于 2007-11-21 19:53
喜欢经验之谈
！！！！！！！！！！！

13#
发表于 2007-11-22 12:36
喜欢经验之谈

14#
发表于 2007-11-22 15:14
感谢楼主，好贴，收藏！

15#
发表于 2007-12-7 14:30
回复 6# 的帖子
谢谢，得好好研读一下

16#
发表于 2007-12-23 20:30
好文章，谢谢楼主分享经验。

17#
发表于 2007-12-27 13:20
支持开源，倡导开源，

18#
发表于 2007-12-28 19:04
xxxxxxxxxxxxxxxxx

19#
发表于 2008-1-2 23:36
呵呵
呵呵，看过，谢了，想起了很久用汇编写了个 PI 算法，写得老长。

20#
发表于 2008-1-8 01:31
不够清楚呀。。。。。。。

21#
发表于 2008-1-10 11:53
太感谢楼主了，这么好的贴了，要拼了老命的顶啊

22#
发表于 2008-1-10 12:03
赞！希望楼主贴代码(*^__^*) 嘻嘻……

23#
发表于 2008-3-3 20:32
顶一个,很感谢版主的分享

24#
发表于 2008-3-5 11:03
感谢楼主
!!!!

25#
发表于 2008-3-19 08:40
谢谢

26#
发表于 2008-3-21 19:22
感谢楼主 正在思考 PID 的问题

27#
发表于 2008-3-23 20:26

```
if(uk>0)
{
if(uk-uk1>=0.5)
{
uk1=uk1+1;
}
}
if(uk<0)
{
if(uk1-uk>=0.5)
{
uk1=uk1-1;
}
}
```

这段代码什么意思？？？请教！！！

28#
发表于 2008-3-24 09:37
很详细的资料,谢谢。。。

29#
发表于 2008-3-26 09:10
haohaohaohaohao

30#
发表于 2008-3-26 09:19
楼主伐的东西真实太好了