PID 算法原理、调试经验以及代码

学习总结 2010-09-26 22:40:27 阅读 0 评论 0 字号：大中小 订阅

1、PID 控制原理

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。

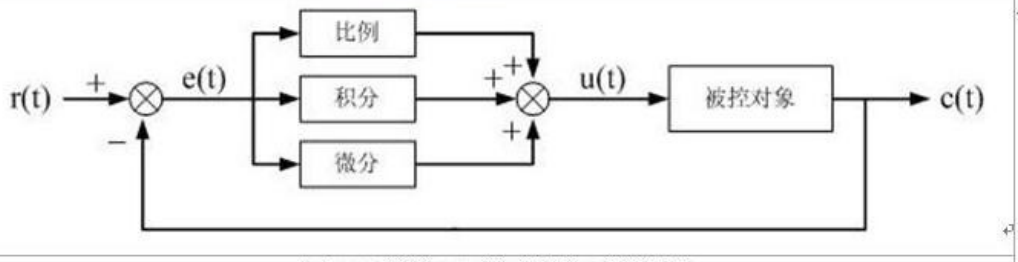

图 4-7 模拟 PID 控制器的原理框图

图 4-7 所示的模拟 PID 控制器的控制表达式为：

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

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，$k_d = k_pT_d$。

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。

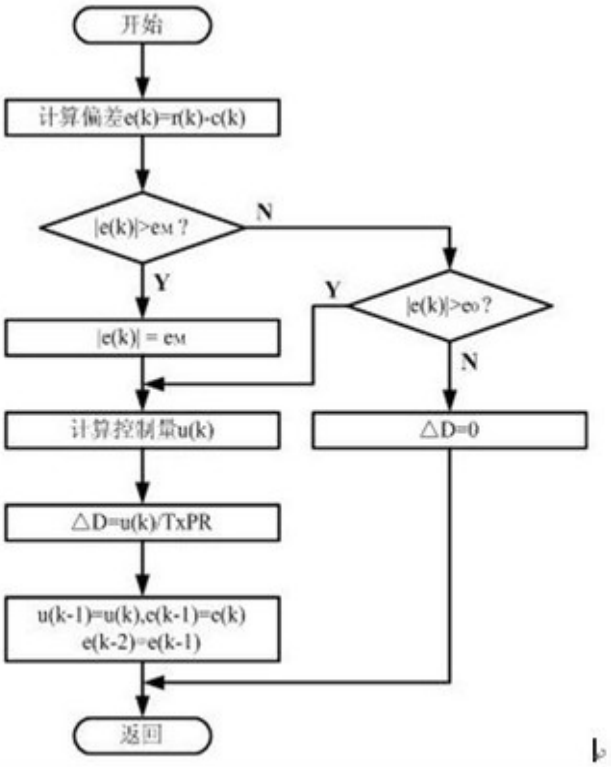

图 4-10 PID 调节子程序流程图

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。

附，完整 PID 代码：

```
//声明变量
//定义变量
float Kp;                  //PID 调节的比例常数
float Ti;                  //PID 调节的积分常数
float T;                   //采样周期
float Td;                  //PID 调节的微分时间常数
float a0;
float a1;
float a2;
float ek;                  //偏差 e[k]
float ek1;                 //偏差 e[k-1]
float ek2;                 //偏差 e[k-2]
float uk;                  //u[k]
int uk1;                   //对 uk 四舍五入求整
int adjust;                //最终输出的调整量
//变量初始化，根据实际情况初始化
    Kp=;
    Ti=;
    T=;
       Td=;
       a0=Kp*(1+T/Ti+Td/T);
       a1=-Kp*(1+2*Td/T);
       a2=Kp*Td/T;
// 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 pid(float ek)
{
    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
        {
            adjust=0;
        }
    else
        {
            uk=a0*ek+a1*ek1+a2*ek2;
            ek2=ek1;
            ek1=ek;
          uk1=(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;

}
float gabs(float ek)
{
       if(ek<0)
       {
            ek=0-ek;
       }
       return ek;
}
```