PID是最广泛使用的算法，是当之无愧的的万能算法，足以应付一般的研发问题。

PID的最能体现反馈思想的控制算法。

P: 比例 I:积分 D: 微分

PID算法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

IMG_256

其中式中：

比例参数Kp：控制器的输出与输入偏差值成比例关系。系统一旦出现偏差，比例调节立即产生调节作用以减少偏差。特点：过程简单快速、比例作用大，可以加快调节，减小误差；但是使系统稳定性下降，造成不稳定，有余差。

积分参数Ki：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。

微分参数Kd：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。

综上所述，我们需要PID的原因无非就是普通控制手段没有办法使输出快速稳定的到达设定值。比如：温度，转速，液位

控制器的P,I,D项选择

下面将常用的各种控制规律的控制特点简单归纳一下：

(1)、比例控制规律P：采用P控制规律能较快地克服扰动的影响，它的作用于输出值较快，但不能很好稳定在一个理想的数值，不良的结果是虽较能有效的克服扰动的影响，但有余差出现。它适用于控制通道滞后较小、负荷变化不大、控制要求不高、被控参数允许在一定范围内有余差的场合。如：金彪公用工程部下设的水泵房冷、热水池水位控制;油泵房中间油罐油位控制等。

(2)、比例积分控制规律(PI)：在工程中比例积分控制规律是应用最广泛的一种控制规律。积分能在比例的基础上消除余差，它适用于控制通道滞后较小、负荷变化不大、被控参数不允许有余差的场合。如：在主线窑头重油换向室中F1401到F1419号枪的重油流量控制系统;油泵房供油管流量控制系统;退火窑各区温度调节系统等。

(3)、比例微分控制规律(PD)：微分具有超前作用，对于具有容量滞后的控制通道，引入微分参与控制，在微分项设置得当的情况下，对于提高系统的动态性能指标，有着显著效果。因此，对于控制通道的时间常数或容量滞后较大的场合，为了提高系统的稳定性，减小动态偏差等可选用比例微分控制规律。如：加热型温度控制、成分控制。需要说明一点，对于那些纯滞后较大的区域里，微分项是无能为力，而在测量信号有噪声或周期性振动的系统，则也不宜采用微分控制。如：大窑玻璃液位的控制。

(4)、例积分微分控制规律(PID)：PID控制规律是一种较理想的控制规律，它在比例的基础上引入积分，可以消除余差，再加入微分作用，又能提高系统的稳定性。它适用于控制通道时间常数或容量滞后较大、控制要求较高的场合。如温度控制、成分控制等。

鉴于D规律的作用，我们还必须了解时间滞后的概念，时间滞后包括容量滞后与纯滞后。其中容量滞后通常又包括：测量滞后和传送滞后。测量滞后是检测元件在检测时需要建立一种平衡，如热电偶、热电阻、压力等响应较慢产生的一种滞后。而传送滞后则是在传感器、变送器、执行机构等设备产生的一种控制滞后。纯滞后是相对与测量滞后的，在工业上，大多的纯滞后是由于物料传输所致，如：大窑玻璃液位，在投料机动作到核子液位仪检测需要很长的一段时间。

总之，控制规律的选用要根据过程特性和工艺要求来选取，决不是说PID控制规律在任何情况下都具有较好的控制性能，不分场合都采用是不明智的。如果这样做，只会给其它工作增加复杂性，并给参数整定带来困难。当采用PID控制器还达不到工艺要求，则需要考虑其它的控制方案。如串级控制、前馈控制、大滞后控制等。

Kp,Ti,Td三个参数的设定是PID控制算法的关键问题。一般说来编程时只能设定他们的大概数值，并在系统运行时通过反复调试来确定最佳值。因此调试阶段程序需得能随时修改和记忆这三个参数。

而工业控制领域，比如温度等，都是滞后效应很严重的，往往都是mS，甚至是10mS级别的，若直接用负反馈，因为激励与反馈的不同步，必然导致强烈的振荡，所以为了解决这个问题，我们需要引入PID算法，来实现这类滞后效应严重系统的负反馈控制，我们以高频感应加热设备加热工件，从常温25度加热到700度为例做说明：

1、25~600度，100%的全功率加热工件，这是因为温差太大，前期要全功率，先加热到靠近目标温度。之所以考虑在600度，是因为滞后效应，若设定太高，当发现接近700度再停下来，但实际上，温度会冲过700度。当然，600度是一个经验值，以下几个温度点都是经验值，根据实际情况而来。

2、600以上，开启P算法，P就是根据测量值与目标值的误差来决定负反馈的大小。P算法公式：反馈=P\*(当前温度-目标温度)。但因为负反馈是基于存在误差为前提的，所以P算法导致一个问题，永远到不了想要的值：700度。因为到了700度，反馈值就没有了。P算法的开启，进一步逼近了目标温度，假设稳态下可以达到650度，这样就算因为滞后效应导致的延时，也不会超过700度太多。

3、当达到P算法的稳态极限650度附近的时候，比如640度，就应该开启另外一个算法解决P算法引起的极限误差，那就是I算法。I算法就是为了消除这个P算法导致的误差值，毕竟我们想要的是700度，而不是650度。I算法，本质上讲就是获取一个700度下对应的一个驱动值，之后用这个驱动值来取代P算法，那么我们怎么得到这个驱动值呢，唯一的手段就是把之前的误差都累加起来，最后得到一个期望值，这个期望值就是我们想要的驱动值。因为只要与目标值存在误差，那么把这些误差值积累起来再去反馈控制，就能一步步的逼近目标值，这如同水温不够高，再加一点点热水，不够高再加，这样总能达到想要的水温。值得注意的是，I算法不能接入太高，必须要在P算法的后期介入，不然很容易积累过大。这个时候可以引入一个误差门限，比如误差为60，当作6来处理，误差为50，当作5来处理，消除大的误差值，具体根据项目情况决定。

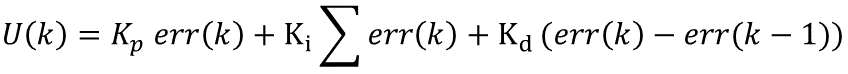
4、当I算法把工件温度加热到很接近目标温度后，那么可以调节的范围就很小了，最后一点点的微动，让调节的每一次的变化，不要太大，这就是D算法。D算法本质上讲就是反对剧烈的变化，所以适用于达到目标温度的时候。

PID算法其实不复杂，但从目前看，很多人都是因为对这三者的使用条件不了解导致的问题，都是从加热一开始，三个要素都上，结果可想而知。P算法是温度接近目标值的时候用，I算法是在P算法到稳态极限的时候用，D算法是达到目标值附近的时候用。实际项目中，D算法一般不用，效果不大。假如非要找一个现实中对应的实物，那么以开关电源为例，TL431基准电源比较器可以认为是P，输出滤波电容C是I，输出滤波电感是D，两者完全等价。它们各自的应用工作点可以认为：假设目标温度700度，600~800度：P算法；640~760度：I算法；690~710度：D算法。具体值，以实验为准，数据仅供参考。

参考网址：[万能算法PID的全面总结 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/162016689)

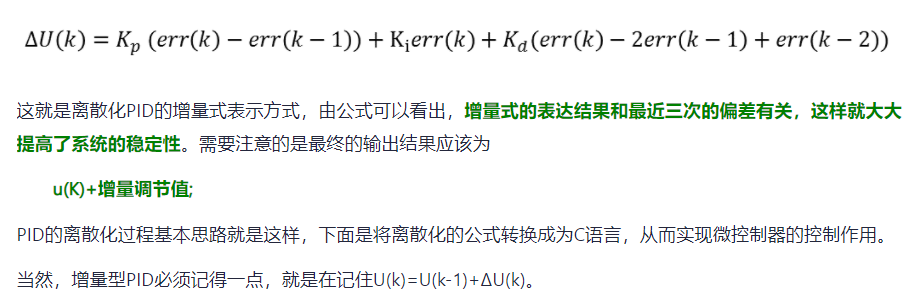
[PID控制算法的C语言实现 - Polaris8 - 博客园 (cnblogs.com)](https://www.cnblogs.com/zhjblogs/p/13731824.html)

1. 位置型PID





1. 增量型PID



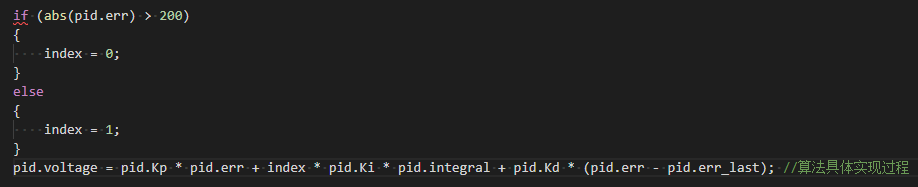


**结论：从最终数据的结果显示来看，增量式PID数据的稳定性要好于位置型PID的；**

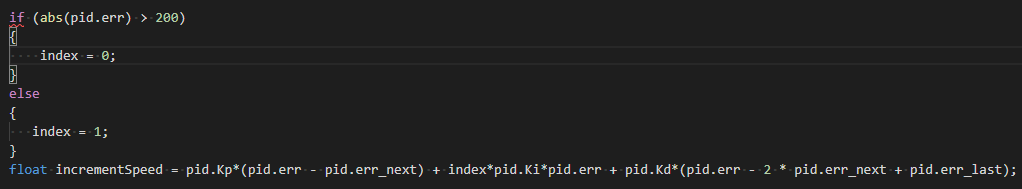
1. 积分分离PID

|  |  |
| --- | --- |
| IMG_256 | **结论：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。** |

a、位置型积分分离PID

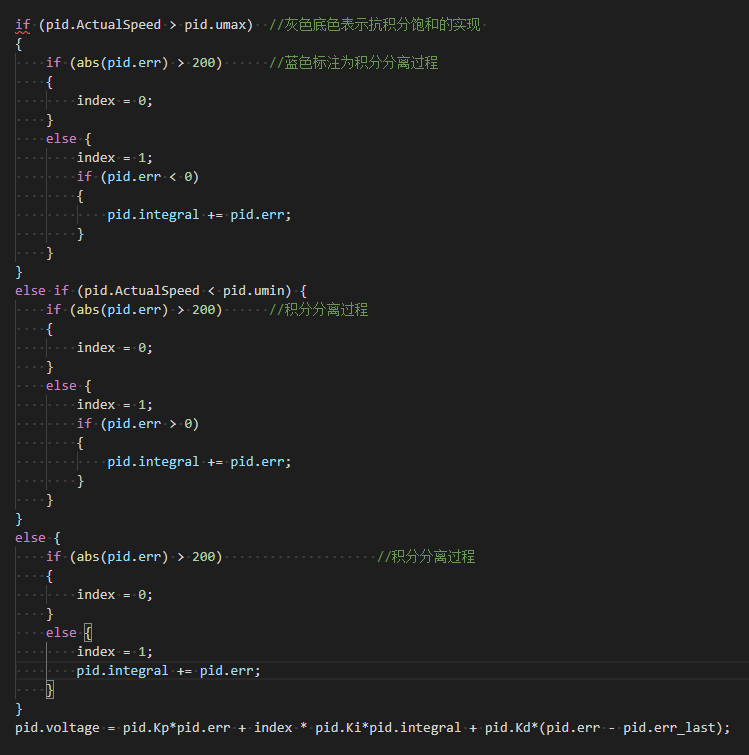


b、增量型积分分离PID

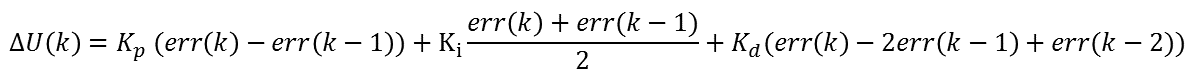


1. 抗积分饱和的PID控制算法

|  |  |
| --- | --- |
| IMG_256 | **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** |



1. 梯形积分的PID控制算法



pid.voltage=pid.Kp\*pid.err+index\*pid.Ki\*pid.integral/2+pid.Kd\*(pid.err-pid.err\_last); //梯形积分 相对于实现六的pid.voltage代码变换

**结论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。梯形积分能增加精度；**

1. 变积分的PID控制算法

|  |  |
| --- | --- |
| IMG_256 | 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\*(pid.err-pid.err\_last); |

IMG_256

**系统的稳定速度非常快**

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，**