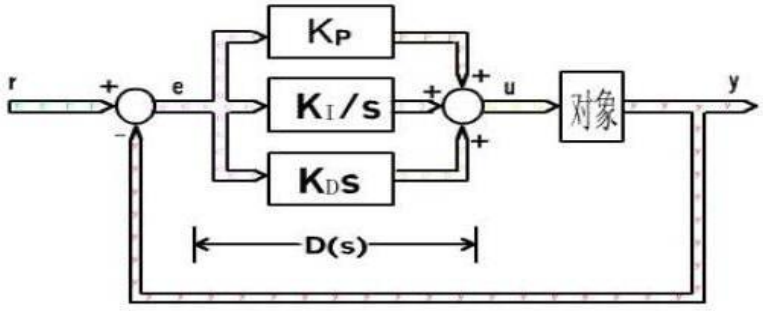
PID 控制算法

1 算法概述

PID 是一个闭环控制算法。因此要实现 PID 算法，必须在硬件上具有闭环控制，就是得有反馈。比如控制一个电机的转速，就得有一个测量转速的传感器，并将结果反馈到控制路线上。以前对于闭环控制的一个最朴素的想法就只有 P 控制，将当前结果反馈回来，再与目标相减，为正的话，就减速，为负的话就加速。现在知道这只是最简单的闭环控制算法。

PID 是比例(P)、积分(I)、微分(D)控制算法。但并不是必须同时具备这三种算法，也可以是 PD,PI,甚至只有 P 算法控制。PID 算法的结构图如下图：



比例(P)、积分(I)、微分(D)控制算法各有作用：

比例，反应系统的基本（当前）偏差 e(t)，系数大，可以加快调节，减小误差，但过大的比例使系统稳定性下降，甚至造成系统不稳定；

积分，反应系统的累计偏差，使系统消除稳态误差，提高无差度，因为有误差，积分调节就进行，直至无误差；

微分，反映系统偏差信号的变化率 e(t)-e(t-1)，具有预见性，能预见偏差变化的趋势，产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除，因此可以改善系统的动态性能。但是微分对噪声干扰有放大作用，加强微分对系统抗干扰不利。

积分和微分都不能单独起作用，必须与比例控制配合。

2 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。

3 公式：

比例（P）控制器

$$u(t) = K_p e(t)$$

比例十积分（PI）控制器

$$u(t) = K_p[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau]$$

比例十积分十微分（PID）控制器

$$u(t) = K_p[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{d(e(t)}{dt}]$$

式中 K_P——比例放大系数；T_I——积分时间； T_D——微分时间。

4 pid 算法源代码：

```
float PID(float e,float kp,float ki,float kd)
{
    static float e_s=0,sum=0;//e_s 用于保存上一次的误差值，用于计算微分项。Sum 用于计算累加和，计算积分项。
    float r;
    sum+=e;//计算积分累加和
```

```
    r=kp*e+ki*sum+kd*(e-e_s);//从左至右分别是比例项、积分项、微分项
    e_s=e;//保存这一次的误差值用于下一次微分计算
    return r;
}
```

5 平衡车控制示例

```
void loop()
{
  float kp=30,ki=0.0,kd=10,r1,r2;
  if (sign==0) return;//sign 为数据更新标志，每隔 10ms 更新一次，也就是说以下代码每隔 10ms 控制一次。
  sign=0;
  kd = (float)analogRead(0)/1024*200;
  r1=PID1(Angle[0],kp,ki,kd);//PID1、PID2 函数就是第四节的 PID 函数，为了区分左右轮，所以分成两个。
  r2=PID2(Angle[0],kp,ki,kd);
  SetMotor(r1,r2);//设置电机转速。
}
```