很多同学都不清楚 PID 是个什么东西，因为很多不是自动化的学生。他们开口就要资料，要程序。这是明显的学习方法不对，起码，首先，你要理解 PID 是个什么东西。
本文以通俗的理解，以小车纵向控制举例说明 PID 的一些理解。
首先，为什么要做 PID？由于外界原因，小车的实际速度有时不稳定，这是其一，要让小车以最快的时间达达到既定的目标速度，这是其二。速度控制系统是闭环，才能满足整个系统的稳定要求，必竟速度是系统参数之一，这是其三.

小车调速肯定不是线性的，外界因素那么多，没人能证明是线性的。如果是线性的，直接用 P 就可以了。比如在 PWM=60%时，速度是2M/S，那么你要它3M/S，就把 PWM 提高到90%。因为90/60=3/2，这样一来太完美了。完美是不可能的。

那么不是线性的，要怎么怎么控制 PWM 使速度达到即定的速度呢？即要快，又要准，又要狠。（即快准狠）系统这个速度的调整过程就必须通过某个算法调整，一般 PID 就是这个所用的算法。

可能你会想到，如果通过编码器测得现在的速度是2.0m/s,要达到2.3m/s 的速度，那么我把 pwm 增大一点不就行了吗？是的,增大 pwm 多少呢？必须要通过算法,因为 PWM 和速度是个什么关系，对于整个系统来说，谁也不知道。要一点一点的试，加个1%,不够，再加1%还是不够，那么第三次你还会加1%吗？很有可能就加2%了。通过 PID 三个参数得到一个表达式：△PWM=a *△V1+b *△V2+c *△V3,a b c 是通过 PID 的那个长长的公式展开，然后约简后的数字，△V1，△V2，△V3 此前第一次调整后的速度差，第二次调整后的速度差,第三次。。。。。一句话，PID 要使当前速度达到目标速度最快，需要建立如何调整 pwm 和速度之间的关系。
输入输出是什么：
输入就是前次速度，前前次速度，前前前次速度。
输出就是你的 PWM 应该增加或减小多少。

为了避免教科书公式化的说明，本文用口语化和通俗的语言描述。虽然不一定恰当，但意思差不多，就是那个事。如果要彻头彻尾地弄 PID，建议多调试，写几个仿真程序。

PID 一般有两种：位置式 PID 和增量式 PID。在小车里一般用增量式，为什么呢？位置式 PID 的输出与过去的所有状态有关，计算时要对 e（每一次的控制误差）进行累加，这个计算量非常大，而明没有必要。而且小车的 PID 控制器的输出并不是绝对数值，而是一个△，代表增多少，减多少。换句话说，通过增量 PID 算法，每次输出是 PWM 要增加多少或者减小多少，而不是 PWM 的实际值。

下面均以增量式 PID 说明。

这里再说一下 P、I、D 三个参数的作用。P=Proportion，比例的意

思，I 是 Integral，积分，D 是 Differential 微分。

打个比方，如果现在的输出是1，目标输出是100，那么 P 的作用是以最快的速度达到100，把 P 理解为一个系数即可；而 I 呢？大家学过高数的，0的积分才能是一个常数，I 就是使误差为0而起调和作用；D 呢？大家都知道微分是求导数，导数代表切线是吧，切线的方向就是最快到至高点的方向。这样理解，最快获得最优解，那么微分就是加快调节过程的作用了。

公式本来需要推导的，我就不来这一套了。直接贴出来：

$$\Delta u_k = u_k - u_{k-1} = Kp(e_k - e_{k-1} + \frac{T}{Ti}e_k + Td\frac{e_k - 2e_{k-1} + e_{k-2}}{T})$$

$$= Kp(1 + \frac{T}{Ti} + \frac{Td}{T})e_k - Kp(1 + \frac{2Td}{T})e_{k-1} + Kp\frac{Td}{T}e_{k-2})$$

$$= Ae_k + Be_{k-1} + Ce_{k-2}$$

其中 $A = Kp(1 + \frac{T}{Ti} + \frac{Td}{T})$；

$B = Kp(1 + \frac{2Td}{T})$；

$C = Kp\frac{Td}{T}$。

看看最后的结果：

△Uk=A*e(k)+B*e(k-1)+C*e(k-2)

这里 KP 是 P 的值，TD 是 D 的值，1/Ti 是 I 的值，都是常数，哦，还有一个 T，T 是采样周期，也是已知。而 A B C 是由 P I D 换算

来的，按这个公式，就可以简化计算量了，因为ＰＩＤ是常数，那么ＡＢＣ可以用一个宏表示。这样看来，只需要求e(k) e(k-1) e(k-2)就可以知道△Uk 的值了，按照△Uk 来调节 PWM 的大小就OK 了。PID 三个参数的确定有很多方法，不在本文讨论范围内。采样周期也是有据可依的，不能太大，也不能太小。

. . . . . .

. . . . . .

PID 实际编程的过程的，要注意的东西还是有几点的。PID 这东西可以做得很深。

1 PID 的诊定。凑试法，临界比例法，经验法。

2 T 的确定，采样周期应远小于过程的扰动信号的周期，在小车程序中一般是 ms 级别。

3 目标速度何时赋值问题，如何更新新的目标速度?这个问题一般的人都乎略了。目标速度肯定不是个恒定的，那么何时改变目标速度呢?

4 改变了目标速度，那么 e(k) e(k-1) e(k-2)怎么改变呢？是赋 0 还是要怎么变？

5 是不是 PID 要一直开着？

6 error 为多少时就可以当速度已达到目标？

7 PID 的优先级怎么处理，如果和图像采集有冲突怎么办？

8 PID 的输入是速度，输出是 PWM，按理说 PWM 产生速度，但二者不是同一个东西，有没有问题？

9 PID 计算如何优化其速度？指针，汇编，移位？都可以试！

```
//*****************************************************
//定义 PID 结构体
//*****************************************************
typedef struct PID
{
    int SetPoint; //设定目标  Desired Value
    double Proportion; //比例常数  Proportional Const
    double Integral; //积分常数  Integral Const
    double Derivative; //微分常数  Derivative Const
    int LastError; //Error[-1]
    int PrevError; //Error[-2]
} PID;
//*****************************************************
//定义相关宏
//*****************************************************
#define P_DATA 100
#define I_DATA   0.6
#define D_DATA   1
```

```
#define HAVE_NEW_VELOCITY 0X01
//*****************************************************
//声明 PID 实体
//*****************************************************
static PID sPID;
static PID *sptr = &sPID;
//*****************************************************
//PID 参数初始化
//*****************************************************
void IncPIDInit(void)
{
sptr->LastError = 0; //Error[-1]
sptr->PrevError = 0; //Error[-2]
sptr->Proportion = P_DATA; //比例常数  Proportional Const
sptr->Integral = I_DATA; //积分常数 Integral Const
sptr->Derivative = D_DATA; //微分常数  Derivative Const
sptr->SetPoint =100;    目标是 100
}
//*****************************************************
//增量式 PID 控制设计
//*****************************************************
int IncPIDCalc(int NextPoint)
{
    int iError, iIncpid; //当前误差
    iError = sptr->SetPoint - NextPoint; //增量计算
    iIncpid = sptr->Proportion * iError //E[k]项
                - sptr->Integral * sptr->LastError //E[k－1]项
                + sptr->Derivative * sptr->PrevError; //E[k－2]项
     sptr->PrevError = sptr->LastError;     //存储误差，用于下次计算
     sptr->LastError = iError;
     return(iIncpid);                                    //返回增量值
}
Int g_CurrentVelocity;
Int g_Flag;
void main(void)
{
      DisableInterrupt
InitMCu();
      IncPIDInit();
g_CurrentVelocity=0;     //全局变量也初始化
g_Flag=0;                      //全局变量也初始化
EnableInterrupt;
      While(1)
{
```

```
    if (g_Flag& HAVE_NEW_VELOCITY)
        {
            PWMOUT+= IncPIDCalc(CurrentVelocity);
              g_Flag&=~ HAVE_NEW_VELOCITY;
}
}
}
//*****************************************
//采样周期 T
//*****************************************
Interrrupt TIME void
{
    CurrentVelocity =GetCurrentVelocity;
    g_Flag|= HAVE_NEW_VELOCITY;
}
```