首先帮大家解决一下什么是 PID 调节，为什么就要这样的疑惑。
PID 是比例，积分，微分的英文单词的首字母的简称。

下面举个例子说明一下 PID，让大家有个感官的认识，。
一个人闭眼走路，假设他知道自己离目的地有 100 米远，那么他就可以以每秒一米一步这样的速度走向目的地，100 米刚刚好是 100 步，这是一个非常理想化的现象。假设他不知道目的地有多远，目的地可能是 1000 米也有可能是 10000 米，他用每秒每步 3 米得速度向前，很不巧的是这个目的地在 80 米处，他走了 26 步时刚刚好差 2 米，走 27 步有刚刚好又多出 1 米，这就是所谓的稳态误差，如果这个人知道目的地在**大概** 15 米处得地方，开始这个人以每秒一米一步的速度，走完一步然后**目测**一下离目的地还有多远，结果发现还剩下大概 14 米，显然一米一步太慢，因此这个人决定每秒大于一米一步走，得出一条式子，
y=Kpe(t)
其中 y 为下一次要每步要走的距离，**e(t) 为目测距离，也就是偏差，换句话说就是自己走了的距离跟要走的距离也就是目的地的误差，Kp 就是一个常数，假设我们把 Kp 设置为 0.5，y=KPe(t)可以得出 y=7；也就是说那个人下一步要以每秒 7 米得速度走，重复上述的过程，，7+1 共走了 8 米，然后目测一下距离 15 米处还有多远，还有 7 米得误差，所以下一步要走 3.5 米，然后在重复，发现最后会出现一个稳态的误差，也就是多走一步会超出目的地，少走一步又没到目的地。**当然这个上述的例子情况非常特殊，大家可能觉得最后那些误差可以忽略，但是实际应用中，肯定没有人走路的那么特殊，按照这种线性比例下去最后得到的误差会非常大，所以就引入了一个积分的概念，积分的数学几何定义是在区间[a, b]里连续的非负曲线与直线 x=a，x=b 围成的图形的面积。从积分的定义可以得到一个函数

$$y=\frac{1}{T_I}\int e(t)dt$$

其中 Ti 为积分时间，e（t）就是误差了。Y 就是输出，它是个不定积分，事实上把它融入到上述人走路的例子它是个定积分，从 0 到 t 时刻的误差的对时间的积分，也就是说误差曲线 e（t）与时间轴围成的面积，积分时间 Ti 是一个常量，也就是说是自己规定大小，很明显，由上式得 y 为 e（t）与 t 所围成的图形的面积的除以 Ti 的值，Ti 越大 y 越小，Ti 越小 y 越大，大了系统会动荡，所以要慢慢调节系数。
下面是关于积分跟比例的专业阐述：

比例（P）控制
比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。**当仅有比例控制时系统输出存在稳态误差（Steady-state error）。**
积分（I）控制
在积分控制中，控制器的输出与输入误差信号的积分成正比关系。对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误差的 或简称有差系统（System with Steady-state Error）。**为了消除稳态误差，在控制器中必须引入“积分项”。积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。**这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。**因此，比例+积分(PI)控制器，可以使系统在进入稳态后无稳态误差。**
微分调节就是偏差值的变化率。例如，如果输入偏差值线性变化，则在调节器输出侧叠加一

个恒定的调节量。大部分控制系统不需要调节微分时间。因为只有时间滞后的系统才需要附加这个参数。如果画蛇添足加上这个参数反而会使系统的控制受到影响。

举个例子，人去调节窝炉的温度，慢慢调节旋钮，使得温度慢慢变大，要使得温度达到某个固定值，人可以慢慢调节，边看温度边调节，如果开始离这个这目标温度远就快速旋旋钮（比例效果），到最后要使得温度误差小就微调（积分效果），然后实际上温度是有一个惯性在那里，开始你以很快速度调节旋钮的时候温度不会突变，不会一下子就达到稳定值，它慢慢增加到最后，但是不是每个人都是这么有经验，当他看到温度值离目标温度还差这么远，又加快旋转旋钮，最终结果导致实际温度跟目标温度差别非常远，微调也跟本没法调整，最后导致系统的不稳定，**但是如果这个人很有经验，他事先知道这个温度是有惯性的，开始它快速旋转旋钮看温度上升率非常高，也就是温度变化非常快，他就放慢旋转速度了，最后结果是准确的把温度调整到最佳（微分效果）。**（卢军：问题：微分效果是处理惯性的？）人可以是这样子，但是计算机可不会这样调节，那么就要通过一个 PID 得到一个输出值来调节了。

$$y = T_D \frac{de(t)}{dt}$$

下面是一段关于微分的专业阐述：

制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。 自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。**其原因是由于存在有较大惯性组件（环节）或有滞后(delay)组件，具有抑制误差的作用，** 其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化"超前"，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入 "比例"项往往是不够的，比例项的作用仅是放大误差的幅值，而目前需要增加的是"微分项"，它能预测误差变化的趋势，这样，具有比例+微分的控制器，就能 够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。所以对有较大惯性或滞后的被控对象，比例+微分(PD)控制器能改善系统在 调节过程中的动态特性。

综上所述得到一个一条公式，这个就是模拟 PID

$$y = K_P\left[e(t) + \frac{1}{T_I}\int e(t)dt + T_D\frac{de(t)}{dt}\right]$$

下面是关于应用，增量式 PID 算法。其实 PID 的算法可以做很深，但没必要，一般入门级的算法已经在很多场合够用了，这里之所以选用增量式 PID 算法（另外还有位置式 PID 等

等），因为增量式 PID 算法运算量少，非常适合单片机的应用。
显然要想给单片机运算，就必须是数字量，而上述的 PID 是模拟 PID，我们要将他数字化，离散化。
其中积分在上面说到的，他的几何意义就是求 e（t）与时间轴 t 围成的图形的面积，将这个面积分成 T 等分　，T=0 到 T=1 跟 e（t）围成的面积加上 T=1 到 T=2 跟 e（t）围成的面积一直累加。。。。。直到 T-1 到 T 跟 e（t）围成的面积刚好就是整个 e（t）与 t 时间轴围成的面积，刚刚好是 e（t）对 t 的积分，如果 T 无限大，那么就可以分割成无限个小图形那么这个图形的面积就可以用 T[e（1）+e(2)+………+e(T-1)+e(T)]来代替积分效果，而这个 T 等分就是 AD 在整个时间轴 t 中采样的点，显然越快的 AD 在相同的时间 t 里面采样的点越多，换句话说就是 T 更接近无限大。因此积分可以用累和代替。

下面为积分的专业的解释

定义

设函数 f(x)在[a,b]上有界，在[a，b]中任意插入若干个分点

a=x0<x1<...<xn-1<xn=b

把区间[a，b]分成 n 个小区间

[x0，x1]，...[xn-1，xn]。

在每个小区间[xi-1，xi]上任取一点 ξi(xi-1≤ξi≤xi)，作函数值 f(ξi)与小区间长度的乘积 f(ξi)△xi，并作出和

$$s=\sum_{i=1}^{n} f(\xi_1)\Delta x_1$$

如果不论对[a,b]怎样分法，也不论在小区间上的点 ξi 怎样取法，只要当区间的长度趋于零时，和 S 总趋于确定的极限 I，

这时我们称这个极限 I 为函数 f(x)在区间[a,b]上的定积分，

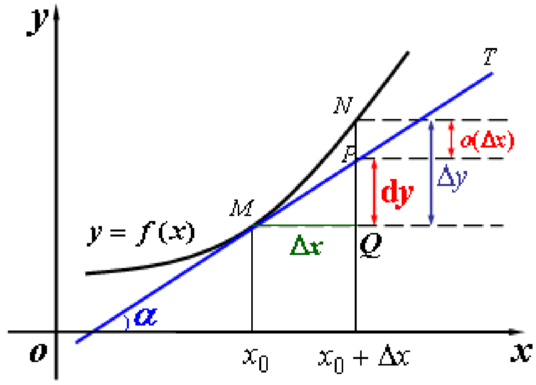
记作

$$\int_{b}^{a} f(x)\,dx$$

微分用差分代替，先说明一下微分的几何意义

二、微分的几何意义

如图，



$$QP = MQ \cdot \tan\alpha$$
$$= \Delta x \cdot f'(x_0)$$
$$= \mathrm{d}y$$

dy 就是切线纵坐标对应的增量.

当$|\Delta x|$很小时，在点M的附近，

切线段MP可近似代替曲线段MN.

我们可以想象把上图中的 f（x）换成 e（t），x 轴换成 t 轴，把△x 换成△t，当△t 非常小的时候曲线 MN 等价于直线 MN，△y 就等于 dy，所以

$$y = T_D \frac{de(t)}{dt}$$

可以用 Td*[e（t）-e（t-1）]/ △t,同样△t 就是采样时间~越小越好。

因此模拟 PID 离散化得到在 k-1 时刻的输出

$$u_{k-1} = K_p[e_{k-1} + \frac{T}{T_i}\sum_{j=0}^{k-1} e_j + T_d \frac{e_{k-1} - e_{k-2}}{T}]$$

因此得到一个增量

$$\Delta u_k = u_k - u_{k-1} = K_p[e_k - e_{k-1} + \frac{T}{T_i} e_k + T_d \frac{e_k - 2e_{k-1} + e_{k-2}}{T}]$$

$$= K_p(1 + \frac{T}{T_i} + \frac{T_d}{T})e_k - K_p(1 + \frac{2T_d}{T})e_{k-1} + K_p \frac{T_d}{T} e_{k-2}$$

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

$$A = K_p(1 + \frac{T}{T_i} + \frac{T_d}{T})$$

$$B = K_p(1 + \frac{2T_d}{T})$$

$$C = K_p \frac{T_d}{T}$$

其中的 T 为采样时间

，如果计算机控制系统采用恒定的采样周期 T，一旦确定 A、B、C(系数的选取是 PID 的关键这里不做讨论)

增量式 PID 控制算法与位置式 PID 算法相比，计算量小得多，因此在实际中得到广泛的应用。

位置式 PID 控制算法也可以通过增量式控制算法推出递推计算公式：

$$u_k = u_{k-1} + \Delta u,$$

就是目前在计算机控制中广泛应用的数字递推 PID 控制算法。

：

下面是程序

```
typedef struct PID
{
int SetPoint; //设定目标 Desired Value
long SumError; //误差累计
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;


static PID sPID;
static PID *sptr = &sPID;
/*==============================================================
============================================
Initialize PID Structure  PID 参数初始化
==============================================================
============================================*/
void IncPIDInit(void)
{
sptr->SumError = 0;
sptr->LastError = 0; //Error[-1]
sptr->PrevError = 0; //Error[-2]
sptr->Proportion = 0; //比例常数 Proportional Const
sptr->Integral = 0; //积分常数 Integral Const
sptr->Derivative = 0; //微分常数 Derivative Const
sptr->SetPoint = 0;
}

/*==============================================================
============================================
增量式 PID 计算部分
==============================================================
============================================*/
int IncPIDCalc(int NextPoint)
{
register 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
//存储误差，用于下次计算
sptr->PrevError = sptr->LastError;
sptr->LastError = iError;
//返回增量值
```

```
return(iIncpid);
}
```