# PID 通俗易懂的讲解

最近从网上看到了一种对 PID 的解释，比较通俗易懂，也好记住，经过自己的整理后说明如下。
控制模型：你控制一个人让他以 PID 控制的方式走 110 步后停下。

**（1）P 比例控制，就是让他走 110 步，他按照一定的步伐走到一百零几步（如 108 步）或 100 多步（如 112 步）就停了。**

*说明：*

*P 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）。*

**（2）PI 积分控制，就是他按照一定的步伐走到 112 步然后回头接着走，走到 108 步位置时，然后又回头向 110 步位置走。在 110 步位置处来回晃几次，最后停在 110 步的位置。**

*说明：*

*在积分 I 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。*

**（3）PD 微分控制，就是他按照一定的步伐走到一百零几步后，再慢慢地向 110 步的位置靠近，如果最后能精确停在 110 步的位置，就是无静差控制；如果停在 110 步附近（如 109 步或 111 步位置），就是有静差控制。**

*说明：*

*在微分控制 D 中，控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 P”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 P+微分 D（PD）控制器能改善系统在调节过程中的动态特性*

```
/*================================================================================================
    PID Function

    The PID (比例、积分、微分) function is used in mainly
    control applications. PIDCalc performs one iteration of the PID
    algorithm.
    While the PID function works, main is just a dummy program showing
    a typical usage.
================================================================================================*/
```

```
typedef struct PID {                    // 结构体
        double 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
        double LastError;               // 最后误差数 Error[-1]
        double PrevError;               // 之前误差数 Error[-2]   <Previous>
        double SumError;                // 误差积分 Sums of Errors
} PID;                                  //
/*=========================================================================================================
   PID 计算部分
=========================================================================================================*/
double PIDCalc( PID *pp, double NextPoint )
{
    double dError,    //当前微分
           Error;     //偏差
        Error = pp->SetPoint - NextPoint;           // 偏差 =设定目标-输入值
        pp->SumError += Error;                      // 积分 =积分+偏差
        dError = pp->LastError - pp->PrevError;     // 当前微分 =最后误差-之前误差
        pp->PrevError = pp->LastError;              // 更新 “之前误差”
        pp->LastError = Error;                      // 更新 “最后误差”
        return (pp->Proportion * Error              // 比例项 =比例常数 x 偏差
            +   pp->Integral * pp->SumError         // 积分项 =积分常数 x 误差积分
            +   pp->Derivative * dError             // 微分项 =微分常数 x 当前微分
        );
}
/*=========================================================================================================
   Initialize PID Structure
=========================================================================================================*/
void PIDInit (PID *pp)
{
    memset ( pp,0,sizeof(PID));
}
/*=========================================================================================================
    Main Program
=========================================================================================================*/
double sensor (void)                    // Dummy Sensor Function 输入
{
    return 100.0;
}
void actuator(double rDelta)            // Dummy Actuator Function 输出
{

}
void main(void)
{
    PID         sPID;                   // PID Control Structure 结构
    double      rOut;                   // PID Response (Output) 输出变量
    double      rIn;                    // PID Feedback (Input)  输入变量
    PIDInit ( &sPID );                  // Initialize Structure  初始化结构
    sPID.Proportion = 0.5;              // Set PID Coefficients  设定 PID 系数 比例常数
    sPID.Integral   = 0.5;              // 积分常数
    sPID.Derivative = 0.0;              // 微分常数
    sPID.SetPoint   = 100.0;            // Set PID Setpoint      设定目标
    for (;;) {                          // Mock Up of PID Processing
        rIn = sensor ();                // Read Input            取输入值
        rOut = PIDCalc ( &sPID,rIn );   // Perform PID Interation  PID 运算
        actuator ( rOut );              // Effect Needed Changes  输出
    }
```