

# 直流电机 PID 开发指南

PID 调节器出现于上世纪 30 年代。所谓 PID 控制，就是对偏差进行比例，积分，微分的控制。PID 由 3 个单元组成，分别是比例（P）积分（I），微分（D）单位。在工程实践中，一般 P 是必须的，所以衍生出许多组合的 PID 控制器，如 PD，PI，PID 等。

在我们的微处理器里面，因为控制器是通过软件实现其控制算法的，所以必须对模拟调节器进行离散化处理，这样它只需根据采样时刻的偏差值计算控制量。因此，我们需要使用离散的差分方程式代替连续的微分方程。

假定采样时间很短时（比如 10ms）可做如下处理：

- 1) 用一阶差分代替一阶微分；
- 2) 用累加代替积分。

## 1. 位置闭环控制

位置闭环控制就是根据编码器的脉冲累加测量电机的位置信息，并与目标值进行比较，得到控制偏差，然后通过对偏差的比例，积分，微分进行控制，使偏差趋向于零的过程。

### 1.1 理论分析

根据位置式离散 PID 公式

$$Pwm = Kp * e(k) + Ki * \sum e(k) + Kd [e(k) - e(k-1)]$$

$e(k)$ ：本次偏差

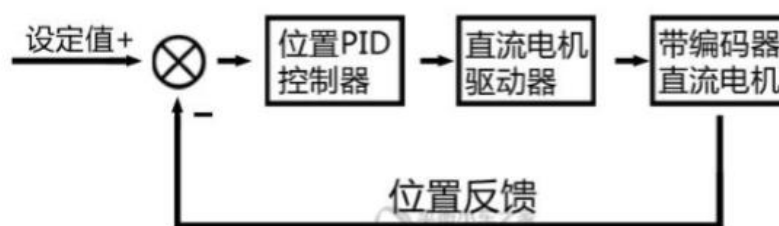
$e(k-1)$ ：上一次偏差

$\sum e(k)$ ： $e(k)$  以及之前的偏差的累积和；其中  $k$  为 1, 2, ...,  $k$ ；

Pwm 代表输出

### 1.2 控制原理图

下图为位置控制原理图，其中需要说明的是，我们这边是通过微机实现 PID 控制的。所以下面的位置 PID 控制器是一个软件实现过程，比如我们的代码里面就是一个我们定义的函数。



### 1.3 C 语言实现

如何把我们以上的理论分析和控制原理图使用 C 语言写出来呢，这是一个有趣且实用的过程。位置式 PID 具体通过 C 语言实现代码如下：

```

int Position_PID (int Encoder,int Target)
{
static float Bias,Pwm,Integral_bias,Last_Bias;

Bias=Encoder-Target;           //计算偏差
Integral_bias+=Bias;           //求出偏差的积分
Pwm=Position_KP*Bias+Position_KI*Integral_bias+Position_KD*(Bias-Last_Bias);
Last_Bias=Bias;                //保存上一次偏差
return Pwm;                    //输出
}

```

入口参数为编码器的位置测量值和位置控制的目标值，返回值为电机控制 PWM(现在再看一下上面的控制原理图是不是更加容易明白了)。

第一行是相关内部变量的定义。

第二行是求出速度偏差，由测量值减去目标值。

第三行通过累加求出偏差的积分。

第四行使用位置式 PID 控制器求出电机 PWM。

第五行保存上一次偏差，便于下次调用。

最后一行是返回。

然后，在定时中断服务函数里面调用该函数实现我们的控制目标：

```
Moto=Position_PID(Encoder,Target_Position);
```

## 1.4 参数整定

首先我们需要明确我们的控制目标，也就是满足控制系统的 3 个要求：

1. 稳定性
2. 快速性
3. 准确性

具体评估指标有最大超调量，上升时间，静差等。

最大超调量是响应曲线的最大峰值与稳态值的差，是评估系统稳定性的一个重要指标。

上升时间是指响应曲线从原始工作状态出发，第一次到达输出稳态值所需的时间，是评估系统快速性的一个重要指标

静差是指被控量的稳定值与给定值之差，一般用于衡量系统的准确性。

关于 P I D 三个参数作用可以大致又不完全的概况为：P 用于提高响应速度，I 用于减小静差，D 用于抑制震荡。

在实践生产过程中，不同的控制系统对控制器效果的要求不一样，比如平衡车，倒立摆对系统的快速性要求很高，响应太慢会导致系统失控。智能家居里面的门窗自动开合系统，对快速性要求就不高，但是对于稳定性和准确性的要求就很高。所以需要严格控制系统的超调量和静差。所以 PID 参数在不同的控制系统中是不一样的。只要我们理解了每个 PID 参数的作用，我们就可以应对工程中各种项目的 PID 参数整定了。

位置控制的调节经验可以总结为：先只用 P 控制，增大 P 系数到系统震荡之后加入微分控制以增大阻尼，消除震荡之后在根据系统对响应和静差等的具体要求，调节 P 和 I 参数。

一般而言，一个控制系统的控制难度，一般取决于系统的转动惯量和对响应速度的要求等。转动惯量越小，对响应速度要求越低。PID 参数就越不敏感。

## 2. 速度闭环控制

速度闭环控制就是根据单位时间获取的脉冲数测量电机的速度信息，并与目标值进行比较。得到控制偏差，然后通过对偏差的比例，积分，微分进行控制。使偏差趋向于零的过程。

一些 PID 的要点在位置控制中已经有讲解。

这里需要说明的是，这里速度控制 20ms 一次，一般建议 10ms 或者 5ms。因为我们使用的是 7.4V 电池，电机额定工作电压 12V。所以速度较慢，20ms 可以延长获取速度的单位时间。提高编码器的采值。

### 2.1 理论分析

根据增量式离散 PID 公式

$$Pwm += Kp[e(k) - e(k-1)] + Ki * e(k) + Kd[e(k) - 2e(k-1) + e(k-2)]$$

$e(k)$  : 本次偏差

$e(k-1)$  : 上一次偏差

$e(k-2)$  : 上上次的偏差

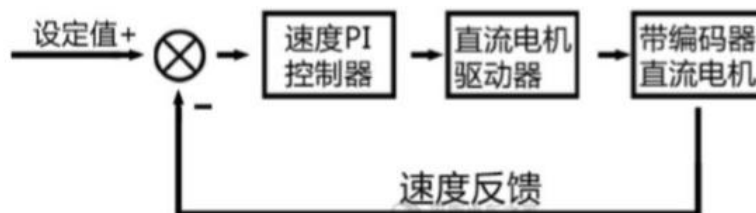
Pwm 代表增量输出

在我们的速度闭环控制系统里面只使用 PI 控制，因此对 PID 控制器可简化为以下公式：

$$Pwm += Kp[e(k) - e(k-1)] + Ki * e(k);$$

### 2.2 控制原理图

下图为速度控制原理图。其中需要说明的是，我们这边是通过微机实现 PID 控制的，所以下面速度 PI 控制器是一个软件实现的过程，比如我们的代码里面就是一个我们定义的函数。



### 2.3 C 语言实现

增量式 PI 控制器具体通过 C 语言实现代码如下：

```

int Incremental_PI (int Encoder,int Target)
{
    static float Bias,Pwm,Last_bias;
    Bias=Encoder-Target;           //计算偏差
    Pwm+=Velocity_KP*(Bias-Last_bias)+Velocity_KI*Bias; //增量式 PI 控制器
    Last_bias=Bias;                //保存上一次偏差
    return Pwm;                    //增量输出
}

```

入口参数为编码器的速度测量值和速度控制的目标值，返回值为电机控制 PWM。

第一行是相关内部变量的定义。

第二行是求出速度偏差，由测量值减去目标值。

第三行使用增量 PI 控制器求出电机 PWM。

第四行保存上一次偏差，便于下次调用。

最后一行是返回。

然后，在定时中断服务函数里面调用该函数实现我们的控制目标：

```

Moto=Incremental_PI(Encoder,Target_Velocity);
Set_Pwm(Moto);           //===赋值给对应 MCU 的 PWM 寄存器

```