

数字 PID 控制器的设计

一、引言

在计算机控制系统中，PID（比例-积分-微分）控制是最经典、应用最广泛的控制策略。尽管现代控制理论层出不穷，但 PID 凭借其结构简单、物理意义明确、鲁棒性强等优点，依然占据了工业控制 90% 以上的份额。与模拟 PID 不同，数字 PID 是基于计算机采样和离散化算法实现的。本文将重点探讨数字 PID 控制器的设计原理、算法形式的选择以及针对工程实际问题的软件改进设计。

二、数字 PID 的离散化原理

模拟 PID 控制规律描述的是连续时间变量，其数学表达为：

$$u(t) = K_p[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt}]$$

其中，

$$K_p$$

为比例系数，

$$T_i$$

为积分时间常数，

$$T_d$$

为微分时间常数。

在计算机控制系统中，由于计算机只能处理离散的数字信号，我们需要对上述公式进行离散化处理。通常设定采样周期为

$$T$$

，用求和代替积分，用差分代替微分。在第

k

次采样时刻，控制算法可转化为差分方程，这是数字 PID 设计的基础。

三、 两种核心算法形式的设计与比较

在软件设计中，数字 PID 主要有两种标准形式，设计时需根据执行机构的特性进行选择。

3.1 位置式 PID 控制算法

位置式 PID 的输出

$$u(k)$$

直接对应执行机构的实际位置（如阀门的开度、电压的大小）。

设计特点： 需要累加历次偏差值，计算量相对较大，且必须保存历史数据。

缺点： 一旦计算机出现故障或重启，输出值可能发生大幅跳变，导致执行机构动作剧烈，引发安全事故。此外，容易产生积分饱和现象。

适用场景： 适用于本身没有记忆功能的执行机构，如没有积分部件的模拟电路驱动。

3.2 增量式 PID 控制算法

增量式 PID 计算的是本次控制量的增量

$$\Delta u(k) = u(k) - u(k - 1)$$

。计算机只输出控制量的变化值，实际执行机构（如步进电机或带记忆的调节阀）根据增量进行动作。

设计优点： 算法简单，只与最近三次的采样偏差有关，不需要累加历史误差，误动作影响小。即使计算机故障，执行机构也能保持原位，安全性高。手动/自动切换时可以实现无扰动切换。

适用场景： 计算机控制技术的首选方案，特别是步进电机控制或带保持位

的 D/A 转换系统。

四、 数字 PID 的改进设计

标准的 PID 算法在实际工程中往往难以满足所有要求，因此在软件设计时必须引入改进措施，这也是数字 PID 设计的核心难点。

4.1 积分分离算法

在启动或误差较大时，标准 PID 的积分项会迅速累积，导致系统响应出现巨大的超调量。

设计策略： 设定一个误差阈值

$$\epsilon$$

。当误差

$$|e(k)| > \epsilon$$

时，取消积分作用（PD 控制），快速消除误差；当误差

$$|e(k)| \leq \epsilon$$

时，引入积分作用（PID 控制），消除稳态误差。这种“分段控制”能兼顾快速性与稳定性。

4.2 抗积分饱和设计

当执行机构达到极限（如阀门全开）时，若偏差继续存在，积分项会持续增加（计算值虚高），导致偏差反向时，控制器需要很长时间才能“退饱和”，造成系统迟钝。

设计策略： 在软件中增加限幅逻辑。一旦计算出的控制量超出限制范围，便停止积分项的累加，或直接将积分项修正为限幅值，确保系统能随时响应反向变化。

4.3 带有死区的 PID 控制

对于频繁波动的系统，微小的干扰会导致控制器频繁动作，加剧执行机构磨损。

设计策略： 引入死区概念。当偏差绝对值小于某个微小值时，认为偏差为 0，控制器不动作。这牺牲了微小的精度，换取了系统的稳定性和执行机构的寿命。

五、 采样周期的选择

数字 PID 的设计离不开采样周期

$$T$$

的确定。

$$T$$

太小，对计算机精度和速度要求高，且容易引入高频噪声；

$$T$$

太大，会丢失信号细节，降低系统稳定性和抗干扰能力。

设计原则： 根据香农采样定理，

$$T$$

应满足信号复现要求。在工程上，通常根据被控对象的响应速度来定。例如，流量控制系统

$$T$$

选 1-5 秒，而伺服电机控制系统

$$T$$

可能需要毫秒级。

六、 结论

数字 PID 控制器的设计不仅仅是公式的代码化，更是一项系统工程。在设计过程中，工程师必须首选确定采用位置式还是增量式算法，这决定了系统的架构；其次，必须针对具体的物理对象，加入积分分离、抗饱和、数字滤波等改进算法，以应对非线性与干扰；最后，合理选择采样周期和整定参数。只有经过这样严谨设计和优化的数字 PID 控制器，才能在计算机控制系统中发挥出高精度、高稳定性的控制效果。