51 单片机 PID 算法程序(三)增量式 PID 控制算法

当执行机构需要的<mark>不是控制量的绝对值</mark>,而是控制量的增量(<mark>例如去驱动步进电动机</mark>)时,需要用 PID 的"增量算法"。

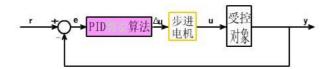


图3-2(b) 增量式PID控制算法的简化示意图

增量式 PID 控制算法可以通过(2-4)式推导出。由(2-4)可以得到控制器的第 k-1 个采样时刻的输出值为:

$$\mathbf{u}_{k-1} = K_{P}[\mathbf{e}_{k-1} + \frac{T}{T_{i}} \sum_{j=0}^{k-1} \mathbf{e}_{j} + T_{d} \frac{\mathbf{e}_{k-1} - \mathbf{e}_{k-2}}{T}]$$
(2-5)

将(2-4)与(2-5)相减并整理,就可以得到增量式 PID 控制算法公式为:

$$\Delta \mathbf{u}_{k} = \mathbf{u}_{k-1} = K_{p} \left[\mathbf{e}_{k} - \mathbf{e}_{k-1} + \frac{T}{T_{i}} \mathbf{e}_{k} + T_{d} \frac{\mathbf{e}_{k} - 2\mathbf{e}_{k-1} + \mathbf{e}_{k-2}}{T} \right]$$

$$= K_{p} \left(1 + \frac{T}{T_{i}} + \frac{T_{d}}{T} \right) \mathbf{e}_{k} - K_{p} \left(1 + \frac{2T_{d}}{T} \right) \mathbf{e}_{k-1} + K_{p} \frac{T_{d}}{T} \mathbf{e}_{k-2}$$

$$= A \mathbf{e}_{k} - B \mathbf{e}_{k-1} + C \mathbf{e}_{k-2} \tag{2-6}$$

其中

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

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

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

由(2-6)可以看出,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,只要使用前后三次测量的偏差值,就可以由(2-6)求出控制量。

增量式 PID 控制算法与位置式 PID 算法(2-4)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:

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \Delta \boldsymbol{u}_{(2-7)}$$

(2-7) 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。

增量式 PID 控制算法 C51 程序

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

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)
{
                               //当前误差
register int iError, iIncpid;
iError = sptr->SetPoint - NextPoint;
                               //增量计算
iIncpid = sptr->Proportion * iError
                               //E[k]项
- sptr->Integral * sptr->LastError
                               //E[k-1]项
                               //E[k-2]项
+ sptr->Derivative * sptr->PrevError;
                               //存储误差,用于下次计算
sptr->PrevError = sptr->LastError;
sptr->LastError = iError;
                               //返回增量值
return(iIncpid);
}
```