第六章　数字 PID 及其算法

在模拟系统中，其过程控制方式就是将被测参数，如温度、压力、流量、成分、液位等，由传感器变换成统一的标准信号送入调节器，在调节器中，与给定值进行比较，然后，把比较出的差值经 PID 运算后送到执行机构，改变进给量，以达到自动调节之目的。这种系统多用电动(或气动)单元组合仪表 DDZ(或 QDZ)来完成。而在数字控制系统中，则是用数字调节器来模拟调节器。其调节过程是首先把过程参数进行采样，并通过模拟量输入通道将模拟量变成数字量，这些数字量由计算机按一定控制算法进行运算处理，运算结果由模拟量输出通道输出，并通过执行机构去控制生产，以达到给定值。

计算机控制系统的优点是：

1. 一机多用。由于计算机运行速度快，而被控对象变化一般都比较慢，因此，用一台计算机可以控制几个到十几个回路，甚至几十个回路，因而可大大地节省设备费用。

2. 控制算法灵活。使用计算机不仅能实现经典的 PID 控制，而且还可以应用直接数字控制，如最快响应无纹波系统，大林算法，以及最优控制等。即使采用经典的 PID 控制，也可以根据系统的需要进行改进。

3. 可靠性高。由于计算机控制算法是用软件编写的，因此比用硬件组成的调节器具有较高的可靠性，且系统维护简单。

4. 可改变调节品质，提高产品的产量和质量。由于计算机控制是严格按照某一特定规律进行的，不会由于人为的因素造成失调。特别是在计算机控制系统中采用直接数字控制，可以按被控对象的数学模型编排程序，使调节品质大为提高，为提高经济效益创造了条件。

5. 安全生产，改善工人劳动条件。

计算机控制的主要任务就是设计一个数字调节器，常用以下控制方法。

1. 程序和顺序控制

程序控制是被控量按照一定的、预先规定的时间函数变化，被控量是时间的函数。如单晶炉的温度控制。

顺序控制可以看作是程序控制的扩展，在各个时期所给出的设定值可以是不同的物理量，每次设定值的给出，不仅取决于时间，还取决于对前段控制结果的逻辑判断。

2. 比例积分微分控制(简称 PID 控制)

调节器的输出是其输入的比例，积分、微分的函数。PID 控制现在应用最广，技术最成熟，其控制结构简单，参数容易调整，不必求出被控对象的数学模型便可以调节，因此无论模拟调节器或者数字调节器大都采用 PID 调节规律。

3. 复杂规律的控制

生产实践中控制系统除了给定值的输入外，还存在大量的随机扰动。另外，性能指标的提法，也不单是过渡过程的品质，而且包括能耗最小，产量最高，产量最好等综合性指标。

对于存在随机扰动的系统，仅用 PID 控制是难以达到满意的性能指标的，因此，根据生产

过程的实际情况，可以引进各种复杂规律的控制。例如串级控制，前馈控制，多变量解耦控制，以及最优控制，自适应控制，自学习控制等。

4. 智能控制

智能控制理论是一种把先进的方法学理论与解决当前技术问题所需要的系统理论结合起来的学科。智能控制理论可以看作是三个主要控制理论领域的交叉或汇合，这三个理论领域是人工智能、运筹学和控制理论。智能控制实质上是一个大系统，是综合的自动化。

随着大规模集成电路技术的发展，微型机和单片机的大量出现及其价格的不断下降，计算控制得到了更加广泛的应用。与此同时，计算机控制技术也日臻完善。由于最优控制、自适应控制、自学习系统，以及智能控制等，都需要用到复杂的数学计算，因此，通常需要高效的控制算法和高性能的计算机才能实现这些复杂规律的控制，用单片机完成这样的任务 是不现实的。因此，在这一章里，主要介绍应用最多的 PID 控制，下一章讲述直接数字控制。单片机已在控制系统中应用，特别是一些单回路调节器，以及 DDZ—Ⅳ型电动单元组合仪表中的调节单元，都广泛采用单片机。

值得说明的是，单片机在控制中的应用，就控制规律和控制理论本身，与微型机控制系统的设计并无严格区别，只是在一些程序设计的处理上不同而已。但在程序和顺序控制中，单片机比微型机有着显著的优点。

6.1 PID 算法的数字实现

PID 调节是 Proportional(比例)、Integral(积分)、Differential(微分)三者的缩写，是连续系统中技术最成熟、应用最为广泛的一个调节方式。PID 调节的实质就是根据输入的偏差值，按比例、积分、微分的函数关系进行运算，其运算结果用以输出控制。在实际应用中，根据被控对象的特性和控制要求，可灵活地改变 PID 的结构，取其中的一部分环节构成控制规律，如比例(P)调节、比例积分(PI)调节、比例积分微分(PID)调节等。特别是在计算机控制系统中，更可以灵活应用，以充分发挥计算机的作用。在这一节里，主要讲一下 PID 调节数字实现方法及程序设计。

6.1.1 PID 算法的数字化

在模拟系统中，PID 算法的表达式为

$$P(t) = K_P\left[e(t) + \frac{1}{T_I}\int e(t)dt + T_D\frac{de(t)}{dt}\right] \tag{6-1}$$

式中，$P(t)$—调节器的输出信号；

$e(t)$—调节器的偏差信号，它等于测量值与给定值之差；

K_P—调节器的比例系数；

T_I—调节器的积分时间；

T_D—调节器的微分时间。

由于计算机控制是一种采样控制，它只能根据采样时刻的偏差值来计算控制量。因此，在计算机控制系统中，必须首先对式(6-1)进行离散化处理，用数字形式的差分方程代替连续系

统的微分方程，此时积分项和微分项可用求和及增量式表示：

$$\int_0^n e(t)\mathrm{d}t = \sum_{j=0}^{n} E(j)\Delta t = T\sum_{j=0}^{n} E(j) \tag{6-2}$$

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} \approx \frac{E(k) - E(k-1)}{\Delta t} = \frac{E(k) - E(k-1)}{T} \tag{6-3}$$

将式(6-2)和式(6-3)代入式(6-1)，则可得到离散的 PID 表达式：

$$P(k) = K_P\{E(k) + \frac{T}{T_I}\sum_{j=0}^{k} E(j) + \frac{T_D}{T}[E(k) - E(k-1)]\} \tag{6-4}$$

式中，$\Delta t = T$——采样周期，必须使 T 足够小，才能保证系统有一定的精度；

$E(k)$——第 k 次采样时的偏差值；

$E(k-1)$——第$(k-1)$次采样时的偏差值；

k——采样序号，$k=0,1,2\cdots$；

$P(k)$——第 k 次采样时调节器的输出。

由于式(6-4)的输出值与阀门开度的位置一一对应，因此，通常把式(6-4)称为位置型 PID 的位置控制算式。

由式(6-4)可以看出，要想计算 $P(k)$，不仅需要本次与上次的偏差信号 $E(k)$和 $E(k-1)$，而且还要在积分项把历次的偏差信号 $E(j)$进行相加，即 $\sum_{j=0}^{k} E(j)$。这样，不仅计算繁琐，而且为保存 $E(j)$还要占用很多内存。因此，用式(6-4)直接进行控制很不方便。为此，我们做如下改动。

根据递推原理，可写出$(k-1)$次的 PID 输出表达式：

$$P(k-1) = K_P\{E(k-1) + \frac{T}{T_I}\sum_{j=0}^{k-1} E(j) + \frac{T_D}{T}[E(k-1) - E(k-2)]\} \tag{6-5}$$

用式(6-4)减去式(6-5)，可得：

$$P(k) = P(k-1) + K_P[E(k) - E(k-1)] + K_I E(k) + K_D[E(k) - 2E(k-1) + E(k-2)] \tag{6-6}$$

式中，$K_I = K_P\frac{T}{T_I}$——积分系数；

$K_D = K_P\frac{T_D}{T}$——微分系数。

由式(6-6)可知，要计算第 k 次输出值 $P(k)$，只需知道 $P(k-1)$，$E(k)$，$E(k-1)$，$E(k-2)$即可，比用式(6-4)计算要简单得多。

在很多控制系统中，由于执行机构是采用步进电机或多圈电位器进行控制的，所以，只要给一个增量信号即可。因此，把式(6-4)和式(6-5)相减，得到：

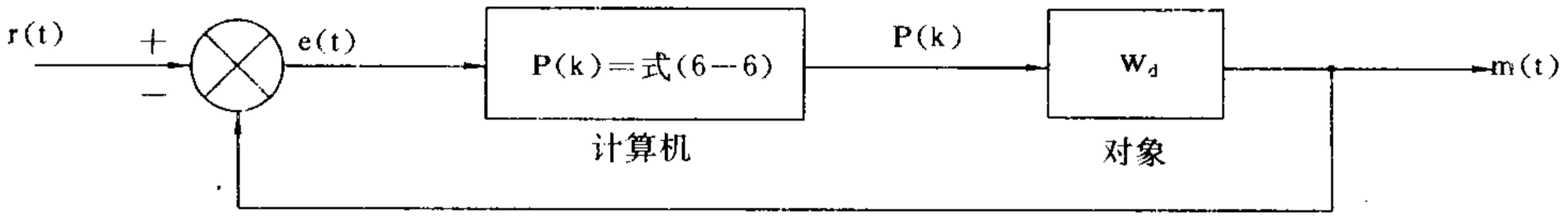
$$\begin{aligned}\Delta P(k) &= P(k) - P(k-1)\\ &= K_P[E(k) - E(k-1)] + K_I E(k) + K_D[E(k)\\ &\quad - 2E(k-1) + E(k-2)]\end{aligned} \tag{6-7}$$

式中，K_I、K_D 同式(6-6)。

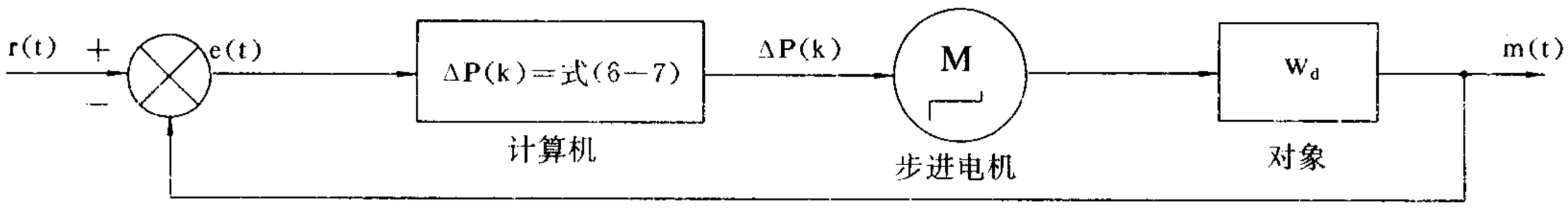
式(6-7)表示第 k 次输出的增量 $\Delta P(k)$，等于第 k 次与第 $k-1$ 次调节器输出的差值，即在第$(k-1)$次的基础上增加(或减少的)量，所以式(6-7)叫做增量型 PID 控制算式。

用计算机实现位置式和增量式控制算式的原理方框图，如图 6-1 所示。

在位置控制算式中，由于全量输出，所以每次输出均与原来位置量有关。为此，这不仅需要



(a) 位置式控制



(b) 增量式控制

图 6—1 *DDC* 控制原理图

对 $e(j)$进行累加，而且计算机的任何故障都会引起 $P(k)$大幅度变化，对生产不利。

增量控制虽然改动不大，然而却带来了很多优点：

1. 由于计算机输出增量，所以误动作影响小，必要时可用逻辑判断的方法去掉；

2. 在位置型控制算法中，由手动到自动切换时，必须首先使计算机的输出值等于阀门的原始开度，即 $P(k-1)$，才能保证手动/自动无扰动切换，这将给程序设计带来困难。而增量设计只与本次的偏差值有关，与阀门原来的位置无关，因而增量算法易于实现手动/自动无扰动切换。

3. 不产生积分失控，所以容易获得较好的调节效果。

增量控制因其特有的优点已得到了广泛的应用。但是，这种控制也有其不足之处：(1)积分截断效应大，有静态误差；(2)溢出的影响大。因此，应该根据被控对象的实际情况加以选择。一般认为，在以晶闸管或伺服电机作为执行器件，或对控制精度要求较高的系统中，应当采用位置型算法，而在以步进电机或多圈电位器作执行器件的系统中，则应采用增量式算法。

此外，除了上述两种控制算法外，还有一种称为速度控制的算法，即

$$V(k)=\frac{\Delta P(k)}{T}$$

$$=\frac{K_P}{T}\{E(k)-E(k-1)+\frac{T}{T_I}E(k)+\frac{T_D}{T}[E(k)-2E(k-1)+E(k-2)]\} \tag{6-8}$$

由于 T 为常量，所以式(6-8)与式(6-7)没有多大区别，故不再详细讨论。

6.1.2 PID 算法程序设计

用 8051 汇编语言进行 PID 程序设计有两种计算方法，一种用定点运算，一种为浮点运算。定点运算速度比较快，但精度低一些；浮点运算精度高，但运算速度比较慢。一般情况下，当速度变化比较慢时，可采用浮点运算。如果系统要求速度比较快，则需采用定点运算的方法。但由于大多数被控对象的变化速度与计算机工作速度相比差异甚远，所以用浮点运算一般都可以满足要求，故本书讨论浮点运算。

下边分别讲一下位置型和增量型两种 PID 程序的设计方法。

一、位置型 PID 算法程序的设计

由式(6-4)可写出第 k 次采样时 PID 的输出表达式为

$$P(k) = K_P E(k) + K_I \sum_{j=0}^{k} E(j) + K_D[E(k) - E(k-1)] \tag{6-9}$$

式中，K_I、K_D 同式(6-6)

为程序设计方便，将式(6-9)作进一步改进，设比例项输出：

$$P_P(k) = K_P E(k)$$

积分项输出：

$$P_I(k) = K_I \sum_{j=0}^{k} E(j)$$

$$= K_I E(k) + K_I \sum_{j=0}^{k-1} E(j)$$

$$= K_I E(k) + P_I(k-1)$$

微分项输出：

$$P_D(k) = K_D[E(k) - E(k-1)]$$

所以，式(6-9)可写为

$$P(k) = P_P(k) + P_I(k) + P_D(k) \tag{6-10}$$

式(6-10)即为离散化的位置型 PID 编程公式，若采用浮点运算，当 K_P、K_I、K_D 分别求出(并转成三字节浮点数)，且存放在指定的内部 RAM 中，则完成式(6-10)位置型浮点运算 PID 运算程序的框图，如图 6-2 所示。

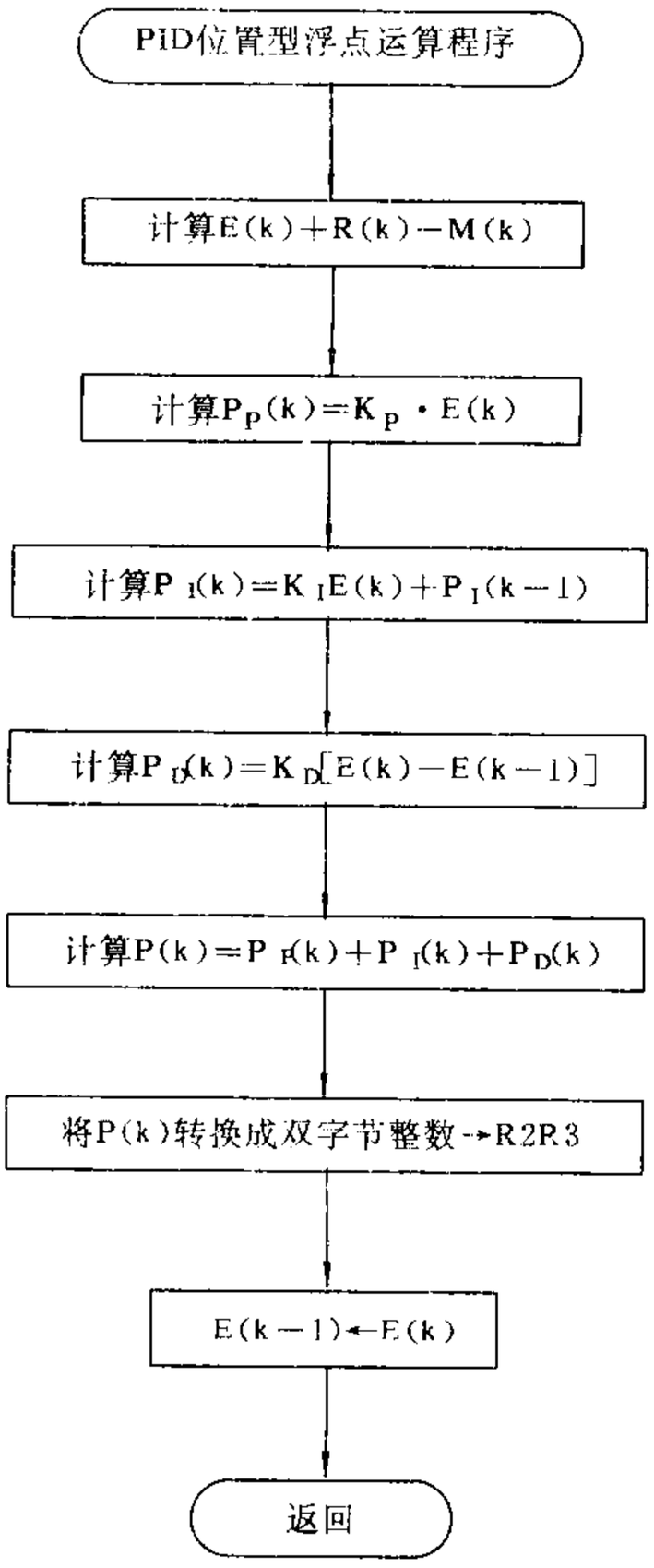


图 6-2　浮点位置型 PID 运算程序流程图

由初始化程序设置初值，使 $E(k-1)=P_I(k-1)=0$。由采样程序进行采样，并经数字滤波后的结果 $M(k)$ 存放在指定单元(浮点数放在以 30H 为首地址的内存单元中)。然后即可调用本程序。程序所占用内存单元的分配，如表 6-1 所示。

表 6-1　　常数项和中间结果存放单元地址分配

内存单元名称	存储单元地址	存放参数	内存单元名称	存储单元地址	存放参数
COEFUR	40H	$R(k)$	BIASE1	4FH	$E(k-1)$
COEFKP	43H	K_P	BIASPI	52H	$P_I(k-1)$
COEFKI	46H	K_I	BIASPP	55H	$P_P(k)$
COEFKD	49H	K_D	MIDDLE	58H	中间结果
BIASEO	4CH	$E(k)$	BIAPID	5BH	$P(k)$

```
PIDPOS:   ORG     8000H
          MOV     R1,#DATA      ;计算 E(k)=R(k)-M(k)
          MOV     R0,#COEFRK
          LCALL   FSUB
          MOV     R1,#BIASE0    ;BIASE0←E(k)
          LCALL   FSTR
```

```
        MOV     R0,#COEFKP      ;计算 P_P(k)=K_PE(k)
        LCALL   FMUL
        MOV     R1,#BIASPP      ;BIASPP←P_P(k)
        LCALL   FSTR
        MOV     R0,#BIASE0      ;计算 K_IE(k)
        MOV     R1,#COEFKI
        LCALL   FMUL
        MOV     R1,#MIDDLE      ;存放中间结果 K_IE(k)
        LCALL   FSTR
        MOV     R0,#BIASPI
        LCALL   FADD            ;计算 P_I(k)=K_IE(k)+P_I(k-1)
        MOV     R1,#BIASPI      ;存 P_I(k),并 P_I(k-1)←P_I(k)
        LCALL   FSTR
        MOV     R0,#BIASE0      ;计算 E(k)-E(k-1)
        MOV     R1,#BIASE1
        LCALL   FSUB
        MOV     R1,#MIDDLE      ;存放中间结果 E(k)-E(k-1)
        LCALL   FSTR
        MOV     R0,#COEFKD
        LCALL   FMUL            ;计算 P_D(k)=K_I[E(k)-E(k-1)]
        MOV     R1,#MIDDLE
        LCALL   FSTR
        MOV     R0,#BIASPI
        LCALL   FADD
        MOV     R1,#MIDDLE
        LCALL   FSTR
        MOV     R0,#BIASPP
        LCALL   FADD            ;计算 P(k)=P_P(k)+P_I(k)+P_D(k)
        MOV     R1,#BIPID       ;BIPID←P(k)
        LCALL   FSTR
        MOV     R0,#BIPID       ;将 PID 三字节浮点数转换成双字节整
                                数→R2R3。符号位→Bit 3CH
        LCALL   FINT
        MOV     4FH,4CH         ;E(k-1)←E(k)
        MOV     50H,4DH
        MOV     51H,4EH
        RET
DATA    EQU     30H
```

二、增量型 PID 算法的程序设计

由式(6-7)可知，增量型 PID 算式为

$$\Delta P(k) = K_P[E(k) - E(k-1)] + K_I E(k) + K_D[E(k) - 2E(k-1) + E(k-2)]$$

设

$$P_P(k) = K_P[E(k) - E(k-1)]$$

$$P_I(k) = K_I E(k)$$

$$P_D(k) = K_D[E(k) - 2E(k-1) + E(k-2)]$$

所以，有

$$P(k) = P_P(k) + P_I(k) + P_D(k) \quad (6\text{-}11)$$

式(6-11)为离散化的增量型 PID 编程表达式。当系数 K_P、K_I、K_D 求出后，分别存放在指定的 RAM 单元中，详见表 6-2。在初始化程序中，将 $E(k-1)$、$E(k-2)$、$P_P(k)$单元清零，而后即可仿照位置型 PID 程序编制出增量型浮点运算 PID 程序。

表 6-2　　常数项及中间结果存放单元地址分配表

存储单元名称	存储单元地址	存放的数据	存储单元的名称	存储单元的地址	存放的数据
COEFUR	40H	$P(k)$	BIASE2	52H	$E(k-2)$
COEFKP	43H	K_P	BIASPP	55H	$P_P(k)$
COEFKI	46H	K_I	MIDLE1	58H	中间结果
COEFKD	49H	K_D	MIDLE2	5BH	中间结果
BIASE0	4CH	$E(k)$	MIDLE3	5EH	中间结果
BIASE1	4FH	$E(k-1)$	BIAPID	61H	$P(k)$

增量型 PID 运算子程序流程图，如图 6-3 所示。

根据流程图可写出如下增量型 PID 浮点运算程序。

```
                ORG     8000H
PIDINC:         MOV     R1,#DATA        ;计算 E(k)=R(k)-M(k)
                MOV     R0,#COEFUR
                LCALL   FSUB
                MOV     R1,#BIASE0      ;BIASE0←E(k)
                LCALL   FSTR
                MOV     R0,#BIASE0      ;计算 E(k)-E(k-1)
                MOV     R1,#BIASE1
                LCALL   FSUB
                MOV     R1,#MIDLE1      ;MIDLE1←E(k)-E(k-1)
                LCALL   FSTR
                MOV     R0,#COEFKP      ;计算 PP(k)=KP[E(k)-E(k-1)]
                LCALL   FMUL
                MOV     R0,#MIDLE2      ;MIDLE2←PP(k)
                LCALL   FSTR
                MOV     R0,#COEFKI      ;计算 PI(k)=KIE(k)
                MOV     R1,#BIASE0
                LCALL   FMUL
```

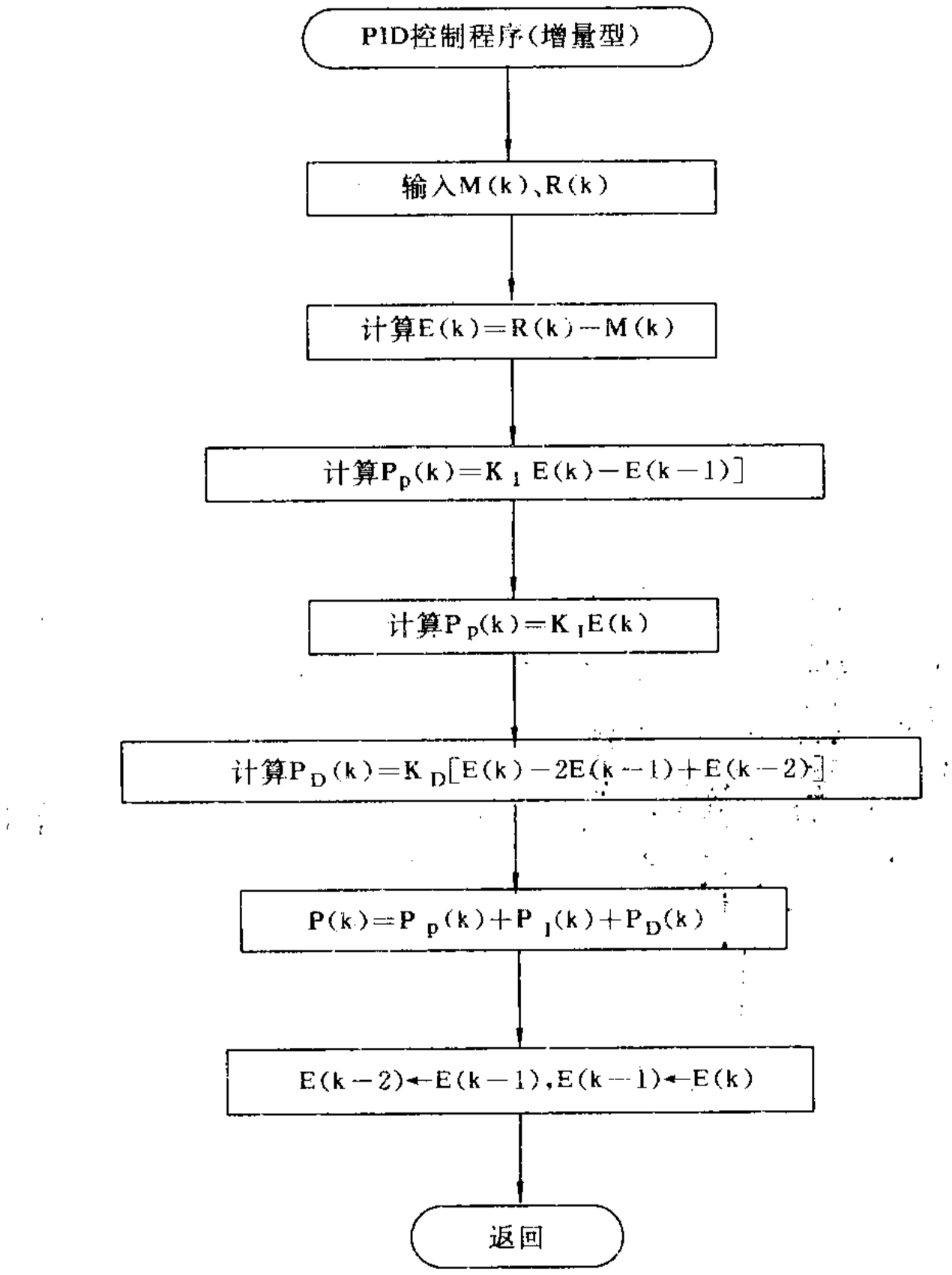


图 6-3　增量型 *PID* 运算程序流程图

```
MOV     R1,#MIDLE3      ;保护 P_I(k)
LCALL   FSTR
MOV     R0,#BIASPP      ;计算 P_P(k)+P_I(k)
LCALL   FADD
MOV     R1,#MIDEL2      ;MIDLE2←P_P(k)+P_I(k)
LCALL   FSTR
MOV     R0,#MIDLE1      ;计算 E(k)-2E(k-1)
MOV     R1,#BIASE1
LCALL   FSUB
MOV     R1,#MIDLE3
LCALL   FSTR
MOV     R0,#BIASE2      ;计算 E(k)-2E(k-1)+E(k-2)
LCALL   FADD
MOV     R1,#MIDLE3
LCALL   FSTR
MOV     R0,#COEFKD      ;计算 K_D[E(k)-2E(k-1)+E(k-2)]
LCALL   FMUL
MOV     R1,#MIDLE1
LCALL   FSRT
MOV     R0,#MIDLE2      ;计算 P(k)=P_P(k)+P_I(k)+P_D(k)
```

```
        LCALL   FADD
        MOV     R1,#BIAPID      ;存 P(k)
        LCALL   FSTR
        MOV     R0,#BIAPID      ;将 PID 三字节浮点数转换成双字节整
                                  数→R2R3。符号位→bit  3CH
        LCALL   FINT
        MOV     52H,4FH         ;E(k-2)←E(k-1)
        MOV     53H,50H
        MOV     54H,51H
        MOV     4FH,4CH         ;E(k-1)←E(k)
        MOV     50H,4DH
        MOV     51H,4EH
        RET
DATA    EQU     30H
```

以上两个程序运行完以后,运算结果为双字节定点数,在 R2R3 寄存器中再调用 PID 输出程序,即可输出 PID 控制。关于 PID 输出程序的设计,可根据执行机构的具体情况(如阀门、步进马达等)酌情处理。

6.2 数字 PID 调节中的几个实际问题

数字 PID 调节器在实际应用中,根据系统对被控参数的要求、D/A 转换器的输出位数,以及对于干扰的抑制能力的要求等,还有许多具体问题需要解决。这是 PID 数字调节器的设计中往往被人忽视,然而实际应用中又不得不正视的一些实际问题。如正、反作用问题,积分饱和问题、限位问题、字节变换问题、电流电压输出问题、干扰抑制问题,以及手动/自动无扰动切换问题等。这在模拟仪表中常需要采取改变线路,或更换不同类型调节器的办法加以解决。在计算机系统中,则可通过改变软件的方法,很容易地实现。在这一节里,我们将介绍几个主要问题。

6.2.1 正、反作用问题

在模拟调节器中,一般都是通过偏差进行调节的。偏差的极性与调节器输出的极性有一定的关系,且不同的系统有着不同的要求。例如,在煤气加热炉温度调节系统中,被测温度高于给定值时,煤气进给阀门应该关小,以降低炉膛的温度。又如在炉膛压力调节系统中,当被测压力值高于给定值时,则需将烟道阀门开大,以减小炉膛压力。在调节过程中,前者称作反作用,而后者称为正作用。模拟系统中调节器的正、反作用是靠改变模拟调节器中的正、反作用开关的位置来实现的。而在由计算机所组成的数字 PID 调节器中,可用两种方法来实现。一种方法是通过改变偏差 $E(k)$的公式来完成。其作法是,正作用时,$E(k)=M(k)-R(k)$,反作用时,则 $E(k)=R(k)-M(k)$,程序的其它部分不变。另一种方法是,计算公式不变,例如都按公式(6-10)、(6-11)计算,只是在需要反作用时,在完成 PID 运算之后,先将其结果求补,而后再送到

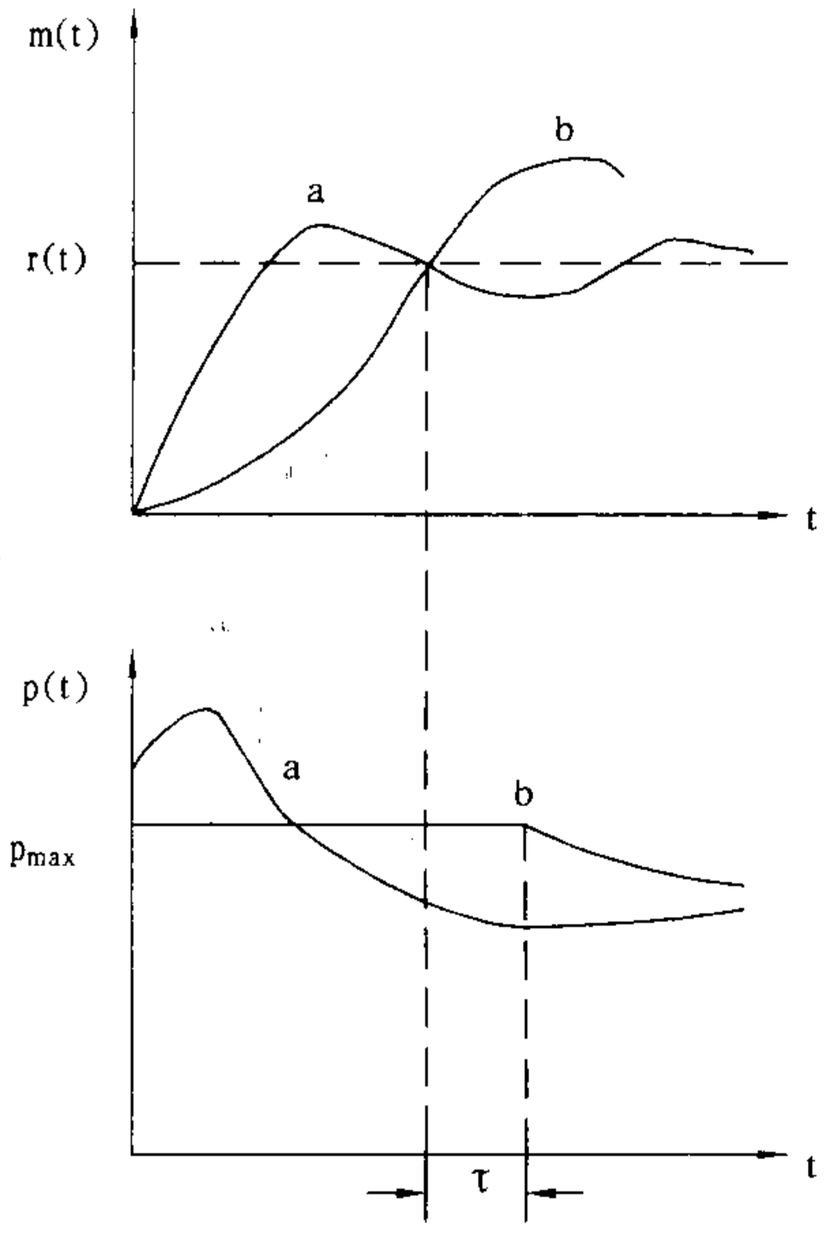


图 6—4　PID 位置算法的积分饱和现象
a. 理想情况的控制；
b. 有限制时产生积分饱和

D/A 转换器进行转换，并进而输出。

6.2.2　饱和作用的抑制

在模拟系统中，由于积分作用，将使调节器的输出达到饱和。为了克服这种现象，人们又研究出抗积分饱和型 PID 调节器，如 DDZ—Ⅲ型仪表中的 DTT—2105 型调节器。同样，在数字 PID 调节器中也存在着同样的问题。

在自动调节系统中，由于负载的突变，如开工、停工、或给定值的突变等，都会引起偏差的阶跃，即 $|E(k)|$ 增大。因而，根据式(6-9)计算的位置型 PID 输出 $P(k)$ 将急剧增加或减小，以至超过阀门全开(或全关)时的输入量 P_{max}(或 P_{min})。此时的实际控制量只能限制在 P_{max}(图 6—4 中曲线 b)，而不是计算值(图 6—4 中的曲线 a)。此时系统输出 $M(k)$ 虽然不断上升，但由于控制量受到限制，其增长速度减慢，偏差 $E(k)$ 将比正常情况下持续更长时间保持在正值，而使式(6-9)中的积分项有较大的累计值。当输出超过给定值 $r(t)$ 后，开始出现负偏差，但由于积分项累计值很大，还要经过相当一段时间 τ 后控制量 $p(t)$ 才能脱离饱和区，这样就使系统的输出出现明显的超调。很明显，在位置型 PID 算法中，饱和现象主要是由积分项引起的，所以称之为“积分饱和”。这种现象引起大幅度的超调，使系统不稳定。

为了消除积分饱和影响，人们研究了很多方法，如遇限削弱积分法，有效偏差法，以及积分分离法等。下边主要介绍前两种方法，关于积分分离法，将在下一节中进行讲述。

一、遇限削弱积分法

这种修正方法的基本思想是：一旦控制量进入饱和区，则停止进行增大积分的运算。具体地说，在计算 $P(k)$ 值时，首先判断上一采样时刻控制量 $P(k-1)$ 是否已超过限制范围，如果已超出，将根据偏差的符号，判断系统的输出是否在超调区域，由此决定是否将相当偏差计入积分项，见图 6—5。该程序流程图，如图 6—6 所示。

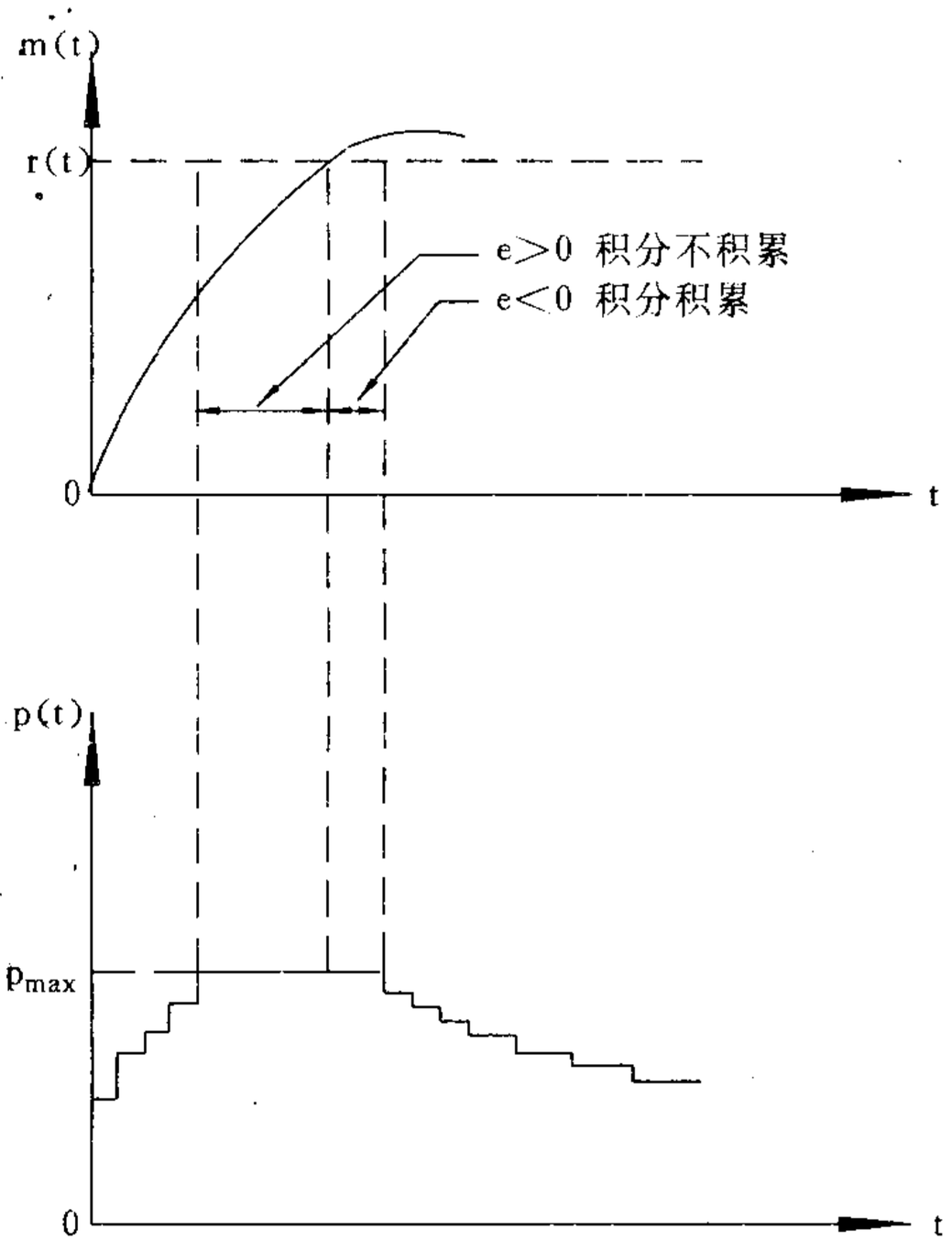


图 6—5　遇限消弱积分法克服积分饱和

二、有效偏差法

当用式(6-4)位置型 *PID* 算式算出的控制量超出限制范围时，控制量实际上只能取边界值，即

$$P(k) = P_{max}(\text{通常为 } 100\% \text{ 阀位})$$

或

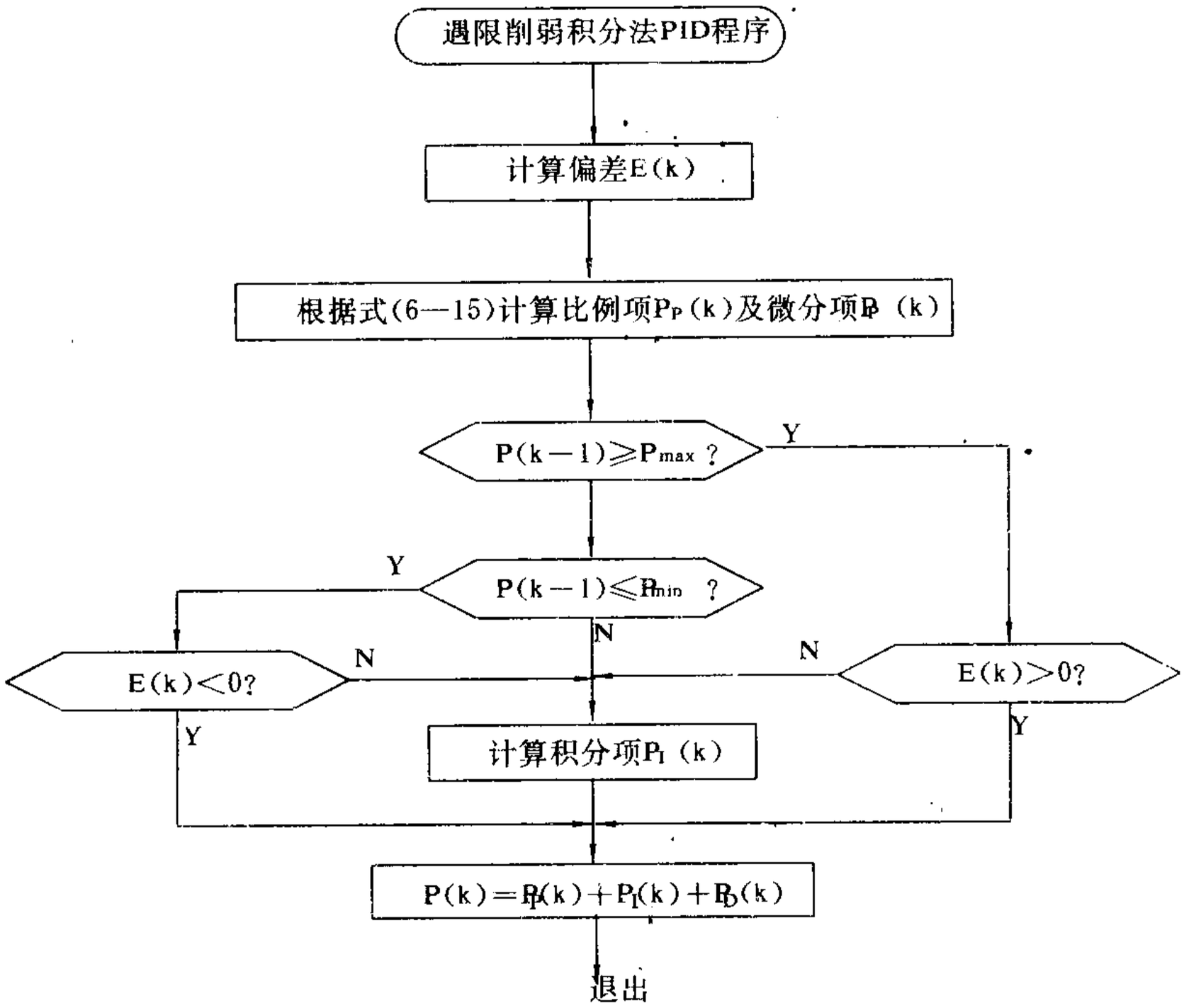


图 6—6　遇限削弱积分的 PID 算法流程图

$P(k) = P_{min}$(通常为 0% 阀位)

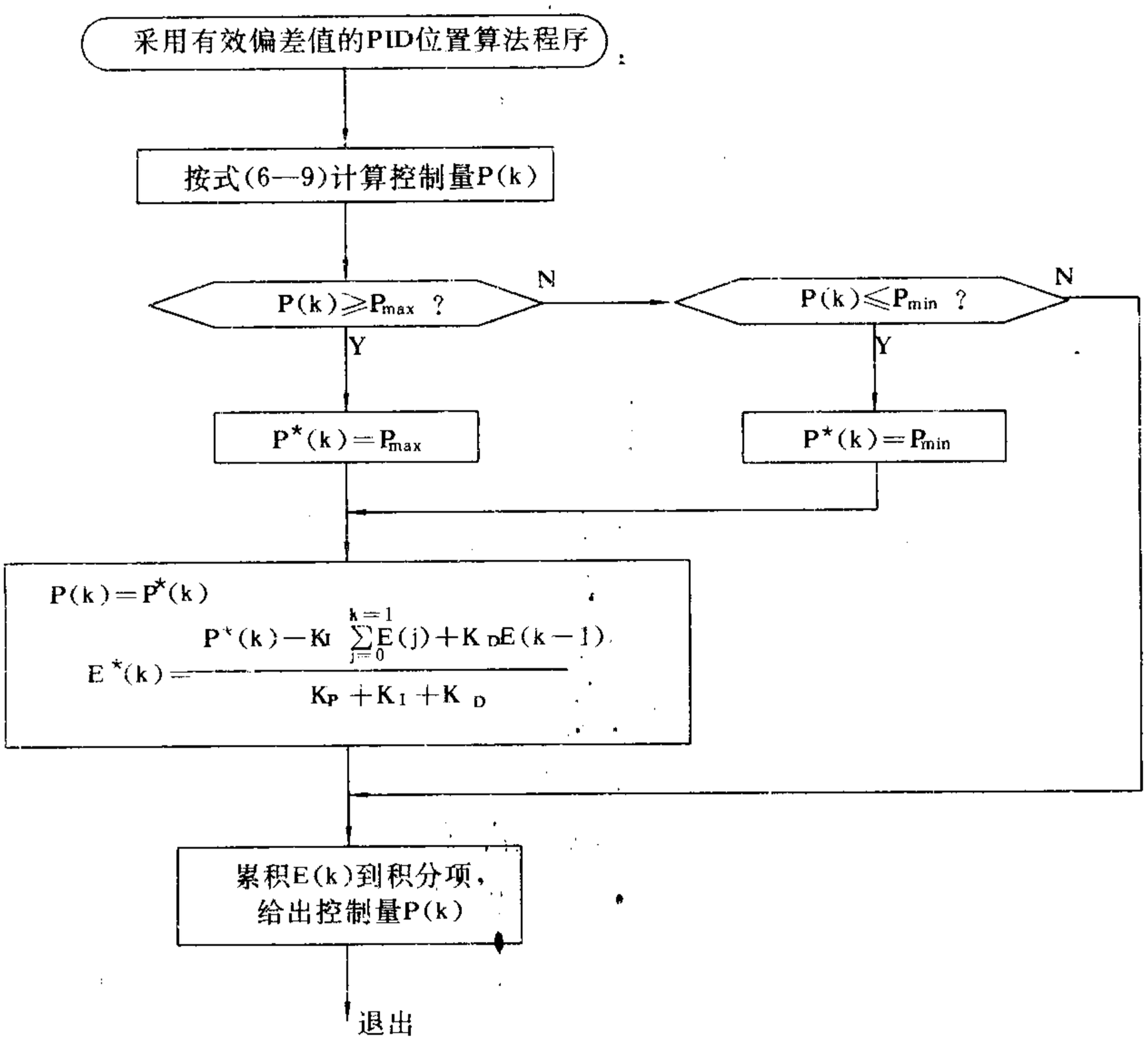


图 6—7　采用有效偏差值的 PID 位置算法流程图

有效偏差法的实质是将相当于这一控制量的偏差值作为有效偏差值进行积分，而不是将实际偏差进行积分。

如果实际的控制量为 $P(k)=P^*$（P_{max}或 P_{min}），则有效偏差可按式(6—4)逆推出，即

$$E^*(k)=\frac{P^*(k)-K_I\sum_{j=0}^{k-1}E(j)+K_DE(k-1)}{K_P+K_I+K_D} \tag{6-12}$$

该算法的程序框图，如图 6—7 所示。

对于增量型 *PID* 算法，由于执行机构本身是存储元件，在算法中没有积分累积，所以不容易产生积分饱和现象，但可能出现比例和微分饱和现象，其表现形式不是超调，而是减慢动态过程。这种现象对很多系统来讲，影响并不很大，故不再详述。

三、限位问题

在某些自动调节系统中，为了安全生产，往往不希望调节阀“全开”或“全关”，而是有一个上限限位 P_{max}，和一个下限限位 P_{min}。因此，要求调节系统的输出不能为 100%或 0%，亦即要求调节器输出限制在一定的幅度范围内，即 $P_{min}\leqslant P\leqslant P_{max}$。在具体系统中，不一定上、下限位都需要，可能只有一个下限或上限限位，例如，在加热炉控制系统中，为防止加热炉熄灭，不希望加热炉的燃料（重油、煤气或天然气）管道上的阀门完全关闭，这就需要设置一个下限限位。为此，可以在 PID 输出程序中进行上、下限比较，其程序流程图，如图 6—8 所示。

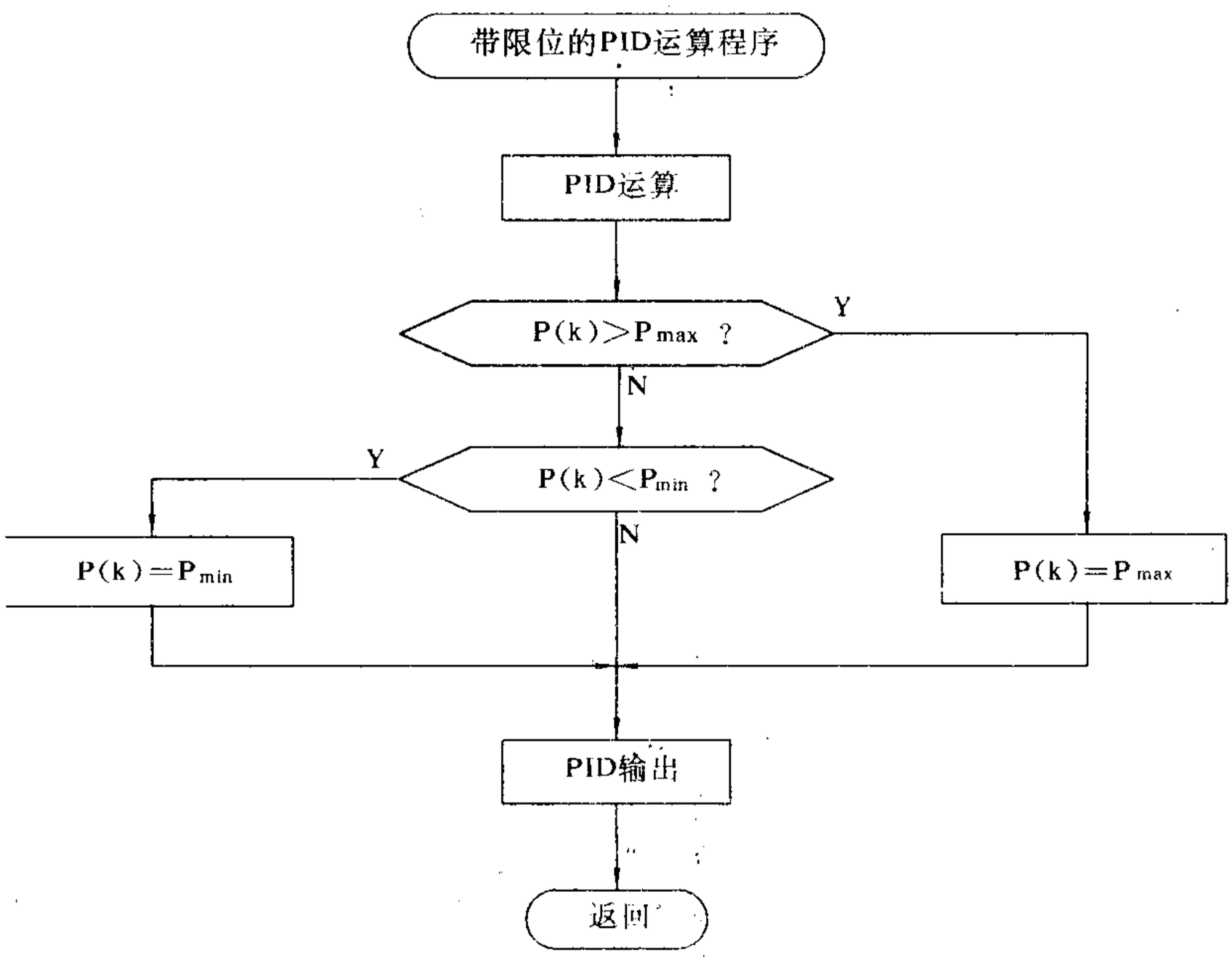


图 6—8　带上、下限限位的 PID 程序

为了提高调节品质，当程序判断输出为 P_{max}（或 P_{min}）后，也可按有限偏差重新求出 $P(k)$值。

6.2.3　手动/自动跟踪及手动后援问题

在自动调节系统中，必须在由手动到自动切换时，能够实现自动跟踪，即在由手动到自动切换时刻，使 PID 的输出等于手动时的阀位值，然后在此基础上，即可按采样周期进行自动调节。为达此目的，必须使系统能采样两种信号：(1)自动/手动状态；(2)手动时的阀位值。

另外，当系统切换到手动时，要能够输出手动控制信号。例如，在用 DDZ—Ⅲ型电动执行机构作为执行单元时，手动输出电流应为 4～20mA。能够完成这一功能的设备，我们称之为手

动后援。在计算机调节系统中，手动/自动跟踪及手动后援是系统安全可靠运行的重要保障。在计算机系统中，手动/自动跟踪及手动后援，可以采用多种方法实现，特别是单片机的位操作指令，更给这一设计带来很大的方便。下边介绍两种实现手动/自动跟踪及手动后援的方法。

一、简易方法

当调节系统要求不太高，或为节省资金，可自行设计一个简易的手动/自动跟踪、及手动后援系统，如图 6—9 所示。

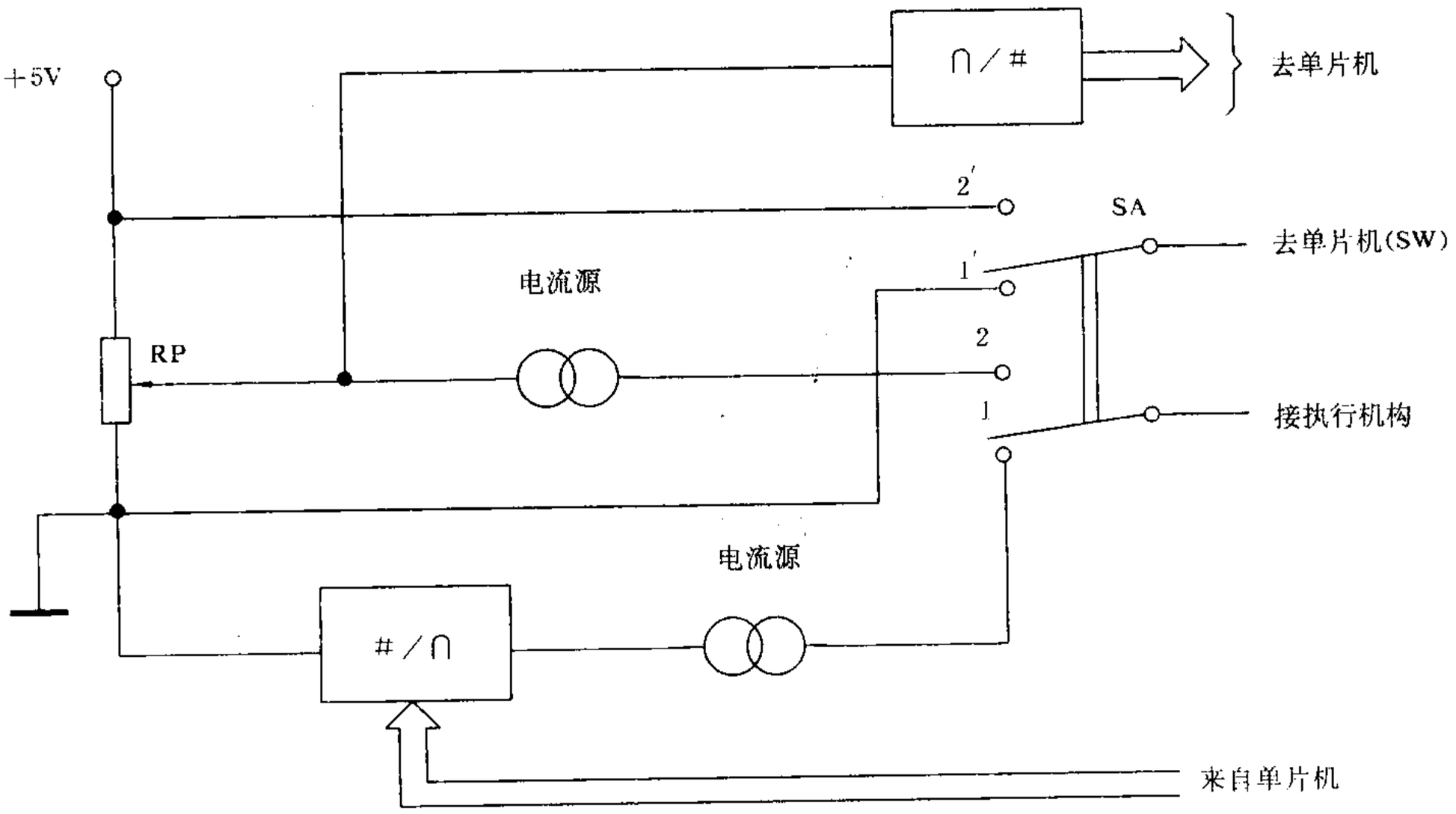


图 6—9　自动跟踪及手动后援原理系统图

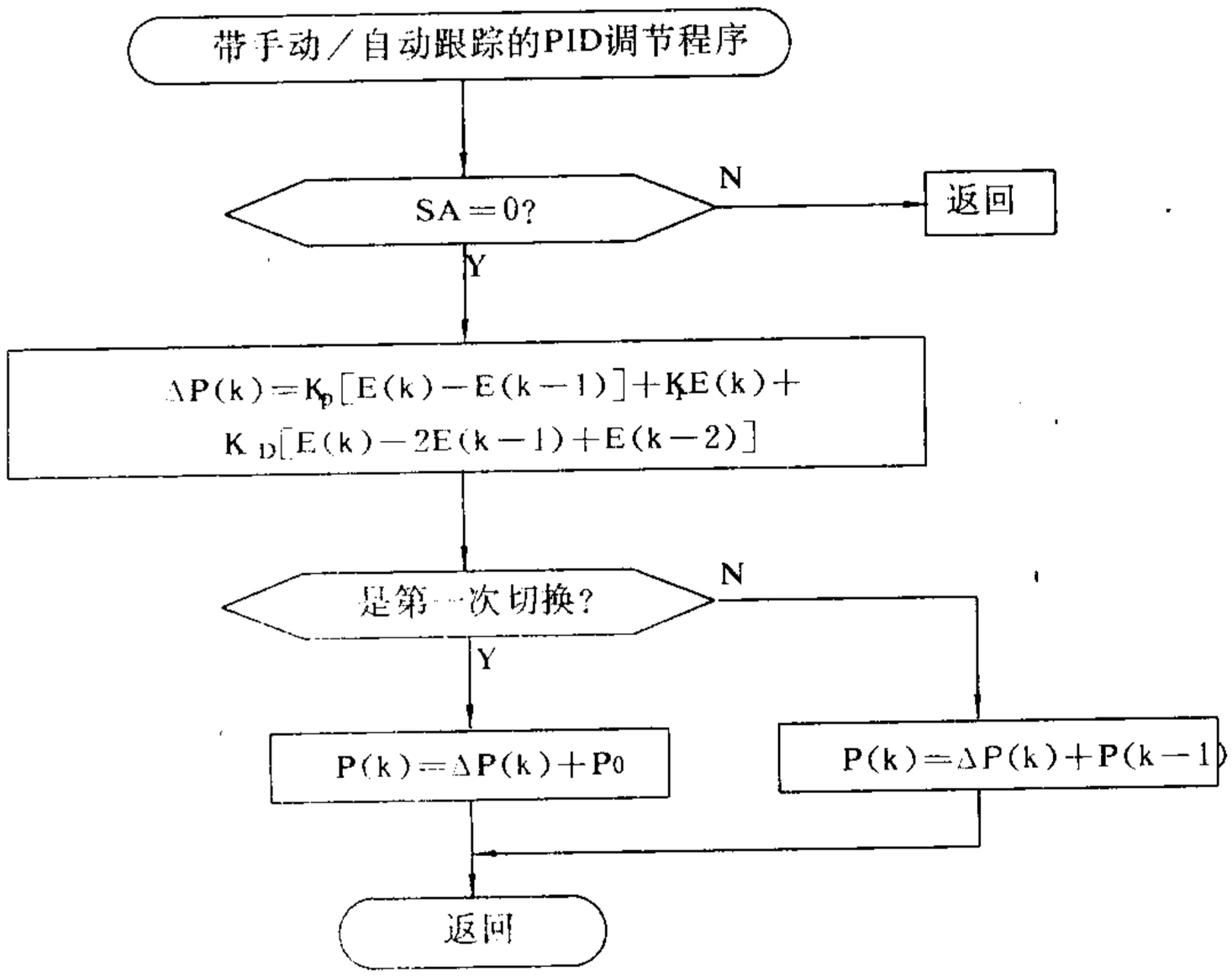


图 6—10　带手动/自动跟踪的 PID 控制流程图

在图 6—9 中，双刀双掷选择开关 SA 有两个位置，当 SA 处于 1—1′位置时，开关量 SW＝0，表示自动方式，执行机构由 D/A 转换器的输出带动；当开关 SA 在 2—2′位置上，开关量 SW＝1，表示手动方式，此时执行机构由电位器 RP 控制。图中的两个电流源为电压—电流变换器，输出范围为 4～20mA。A/D 转换器用来检测手动时阀门的位置。

系统的工作原理是，首先根据系统的要求设置开关 SA 的位置。在 PID 运算之前，先判断 SW 的状态，如果为 1(手动状态)，则不进行 PID 运算，直接返回主程序。若 SW＝0，调节系统处于自动状态，此时，首先进行增量型 PID 运算，然后再加上经 A/D 转换器检测到的手动状态下的阀位值，作为本次 PID 控制的输出值，即

$$P(k) = \Delta P(k) + P_0 \tag{6-13}$$

其中，ΔP(k)—增量型 PID 计算值；

P_0—手动时阀的开度。

注意，此过程只存在于由手动→自动的第一次采样（调节）过程中。以后的 P_0 值，则按公式

$$P(k) = \Delta P(k) + P(k-1) \tag{6-14}$$

处理。

实现上述控制过程的软件流程图，如图 6—10 所示。

请注意，此系统在由自动→手动切换时，必须先使手操作器（电位器）的输出值与 *D/A* 转换器的输出值一致，然后再切换，才能实现自动→手动无扰动切换。

二、利用模拟仪表的操作器

实现手动/自动无扰动切换及手动后援，也可以利用模拟仪表的操作器，如 *DFQ*2000 和 *DFQ*2100 型等。这种方法的优点是手动后援和阀位指示在操作器上均已安排好，这样可节省系统的开发时间。这种方法的基本思想与前面讲的方法大致相同。同样要检测手动/自动状态及手动输出阀位值，然后对手动/自动开关状态进行分析，详见图 6—10。

6.3 几种发展的 PID 算法

由于计算机控制系统的灵活性，除了按式(6-6)和式(6-7)进行标准的 PID 控制计算外，尚可根据系统的实际要求，对 PID 控制进行改进，以达到提高调节品质的目的。下面介绍几种非标准的 PID 算式。

6.3.1 不完全微分的 PID 算式

在上面介绍的标准 PID 算式中，当有阶跃信号输入时，微分项输出急剧增加，容易引起控制过程的振荡，导致调节品质下降。为了解决这一问题，同时要保证微分作用有效，可以仿照模拟调节器的方法，采用不完全微分的 PID 算式，其传递函数表达式为：

$$\frac{P(s)}{E(s)} = K_P^*\left[1 + \frac{1}{T_I^* S} + \frac{T_D^* S}{1 + \frac{T_D^*}{K_d^*} S}\right] \tag{6-15}$$

式中，$P(s)$—PID 输出量算子形式；

$E(s)$—偏差信号算子形式；

K_P^*—实际比例放大系数；

T_I^*—实际积分时间；

T_D^*—实际微分时间；

K_d^*—实际微分增益。

将式(6-15)分成比例积分和微分两部分，则

$$P(s) = P_{PI}(s) + P_D(s)$$

其中，$P_{PI}(s) = K_P^*\left[1 + \frac{1}{T_I^* S}\right]E(s)$