第 14 卷 第 3 期
1988 年 5 月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 14, No. 3
May, 1988

多变量函数复杂过程的微计算机实时控制程序实现研究

黎 康 保
（西北工业大学）

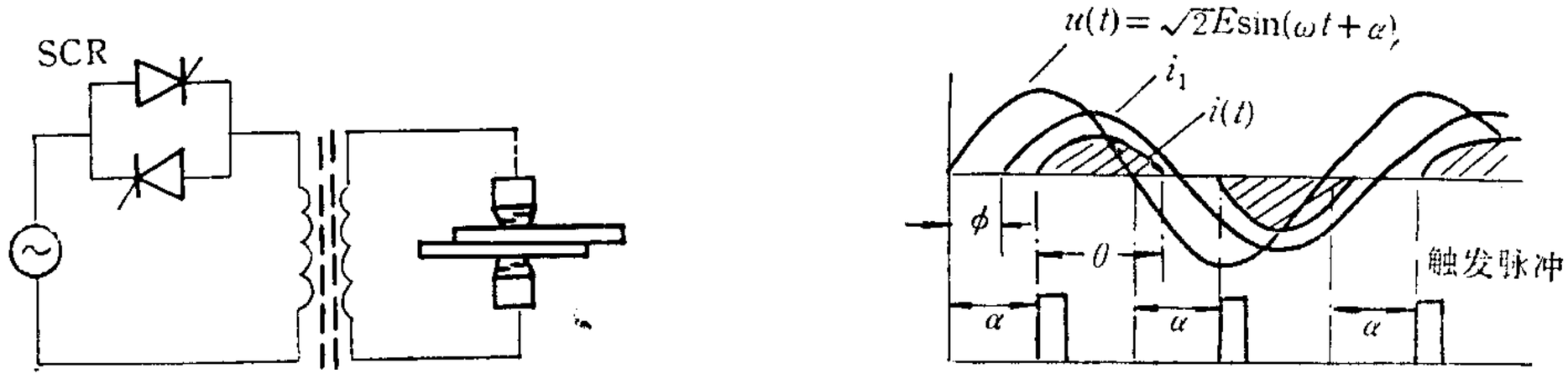
摘 要

对具有多变量函数描述的复杂过程实施计算机实时控制，遇到主控与被控参量之间的期望关系难于推导、过程求解复杂、计算机运算速度远不能满足实时要求，且存储器容量要求过大以致于硬件难于实现. 本文给出一个具有 3—4 变量复杂控制过程程序设计实例，方法容易、计算精度较高、运算速度比函数展开解析法及单个函数表格查找法要快几十倍以上，占用内存空间也十分节省，以致单板微机就可以实现复杂多变量的过程控制.

一、问 题 描 述

在研究计算机过程控制中，常遇到多变量复杂过程的数学描述. 具有复杂多变量函数过程的程序设计问题一直都是计算机控制的一个难题.

在此研究电阻点焊的微计算机控制问题，图 1(a) 表示通过可控硅控制焊接工件电流的原理图. (b) 表示焊接电流 $i(t)$ 与线电压 $u(t)$ 以及变压器负载电流 i_1 之间的波形关系图，有



(a)点焊原理图　　(b)焊接电流波形图

图 1　原理及波形关系图

$$i(t)=\frac{\sqrt{2}}{Z}E[\sin(\omega t+\alpha-\Phi)-\sin(\alpha-\Phi)\cdot e^{-\frac{\omega t}{\mathrm{tg}\Phi}}], \tag{1}$$

$$Z=\sqrt{R^2+\omega^2L^2}, \qquad \Phi=\mathrm{tg}^{-1}\frac{\omega L}{R}.$$

本文于 1985 年 10 月 21 日收到.

用百分比电流 I（即 $I\%$）规格化电流有效值为

$$I = \sqrt{2I'/T}. \tag{2}$$

式中，$I' = \int_0^{\omega t_1} i^2(t)d(\omega t)$，并求得

$$\operatorname{tg}\alpha = \frac{\sin(\theta - \Phi) + \sin\Phi \cdot e^{-\frac{\theta}{\operatorname{tg}\Phi}}}{\cos(\theta - \Phi) + \cos\Phi \cdot e^{-\frac{\theta}{\operatorname{tg}\Phi}}}. \tag{3}$$

(1)—(3) 式是复杂多变量过程描述，没有表示出控制关系，为进行微机控制，必须推导出控制关系表达式．由于电网电压及功率因数变化等原因，致使焊接电流不稳定而影响焊接质量．计算机就是通过控制可控硅控制角 α 的大小来自动调节导通角 θ，保证任何情况下焊接电流的稳定．对此，期望的控制关系应表示为

$$\alpha = F(I, \phi), \tag{4}$$

$$\phi = G(\theta, \alpha). \tag{5}$$

(5) 式由 (3) 式推导，由于三角函数的内部相减及指数关系，获其表达式很困难．(4) 式是由 (1)、(2)、(3) 式联立对四个变量推导，表达式更难获得．这就是说，用解析法推导多变量控制关系往往作不到．

在程序设计方面，正、余弦函数和指数函数可用台劳级数展开，对正弦函数展开为

$$y = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^n \cdot \frac{x^{n+1}}{(2n+1)!}$$

$$\doteq x - \frac{x^3}{3!} + \frac{x^5}{5!} = x\left[1 - x^2\left(\frac{1}{6} - \frac{1}{120}x^2\right)\right]. \tag{6}$$

取前三项，最少要作三次乘法、一次除法和二次减法．用汇编语言在主频 2MHz 的 Z80 微机上作一次乘法或除法要 350μs，正、余弦或指数函数至少 1.4ms，从 (3) 式解出 α 至少 18ms，解 (1)、(2) 式 40ms，总需时间约 60ms．而控制任务要求在 1.5ms 内完成，所以解析法根本不可能实现实时控制．将单个函数作成数据表格，由查表求 $\sin x$、$\cos x$ 或 e^x 一次 0.28ms，$\sqrt{x}$ 需 0.38ms，解 (3) 式＞5ms，解 (1)、(2) 式＞10ms，仍然远不能满足实时控制的要求．

下面研讨上述难题的解决方法．

二、用全式表格函数求解控制关系模型

全式表格函数法是在大计算机 (32 位 FELIX) 上，用高级语言 (FORTRAN) 对过程的全部变量求解表达式(式 (1)—(3))，按精度步长计算出所有数据，从而整理出期望控制关系的数据表格形式，用粗、精两种表格相组合来提高精度和节省内存空间，最优地设计和组织数据的存储结构，就能设计出最满意的程序．

1. 粗值数据表的设计

在 FELIX 机上用 FORTRAN 语言求解 (1)—(3) 式，角度范围 0—180°，粗表精度 0.5%，步长取 0.9°，打印出四个变量的全部数据按表格列出．

由表格整理出期望的多变量控制关系，由 (4)、(5) 式整理出相应的数据表结构，如

表 1　$\Phi = G(\theta, \alpha)$ 数据表

θ			α			ϕ		
时 间 ms	BCD	相 对 HEX	时 间 ms	BCD	HEX	时 间 ms	BCD	HEX
3.8	152	00	7.5	240	*F*0	1.73	069	45
3.8	152	00	7.7	248	*F*8	2.40	096	60
4.0	160	08	7.3	232	*E*8	1.64	066	42
4.0	160	08	7.5	240	*F*0	2.20	088	58
4.0	160	08	7.7	248	*F*8	3.04	122	7*A*
4.2	168	10	7.1	224	*E*0	1.57	063	3*F*
4.2	168	10	7.3	232	*E*8	2.05	082	52
4.2	168	10	7.5	240	*F*0	2.74	110	6*E*
4.2	168	10	7.7	248	*F*8	3.73	149	95
4.4	176	18	6.9	216	*D*8	1.52	061	3*D*
4.4	176	18	7.1	224	*E*0	1.96	078	4*E*
⋮	⋮							⋮
8.8	352	*C*8	2.7	048	30	1.51	060	3*C*
8.8	325	*C*8	2.9	056	38	1.71	068	44
⋮	⋮							⋮
10.0	400	*F*8	1.5	000	00	1.50	060	3*C*
10.0	400	*F*8	1.7	008	08	1.70	068	44
⋮	⋮							⋮
10.0	400	*F*8	4.5	120	78	4.50	180	*B*4

表 1 所示. 同理可得 $\alpha = F(I, \Phi)$ 数据表. 非结点上的数据(精值表相同)可用多元函数插值公式 (7) 求解:

$$y_i = \left[y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}} (x - x_{i-1}) \right] \quad 当 \ Z = Z_{0i} \tag{7}$$

对于表 1, 在 $\theta = 5.4$ms 区间, 有 $\alpha = 6.7$ms 求 Φ 值. 查表得两个结点数 $\alpha_{i-1} = 6.6939$ms, $\alpha_i = 6.7120$ms, 相应 $\Phi_{i-1} = 2.8889$ms, $\Phi_i = 2.9389$ms (表只说明格式, 数据不全), 由 (7) 式得

$$\Phi = \left[\Phi_{i-1} + \frac{\Phi_i - \Phi_{i-1}}{\alpha_i - \alpha_{i-1}} \cdot (\alpha - \alpha_{i-1}) \right] \quad 当 \ \theta = 5.4\text{ms}$$

$$= 2.8889 + \frac{2.9389 - 2.8889}{6.7120 - 6.6939} \cdot (6.7 - 6.6939)$$

$$= 2.91\text{ms}.$$

2. 多变量函数变精度插值表的设计

(1) 插值表设计目的在于大大节省内存空间. 设计思想类似于三角函数表, 在角度很小时通过 $\Delta\theta$ 的弧度值去插值不同的 $\sin\Delta\theta$、$\text{tg}\,\Delta\theta$. 我们的作法是根据函数曲率变化, 在保证量化精度下, 分成若干个不同区间进行变步长插值. 设用纯量化设计的表数据总个数为 N, 而用变步长加插值总个数为 N_v, 结点跨过量化单位 $M = 10$ (即插值表数据个数), 变区间的插值表 $P = 3$ 个, 则采用变步长插值表后数据占用存储空间为

$$N_v = \frac{N}{M} + M \cdot P + 1.$$

若 $N = 1024$，则 $N_v = 133$，节省内存 87%.

多元函数节省量更大,对三元函数表 2，Φ 按 0.05ms 步长应有 68 个数据；取 $I\%$ 步长 0.5%，则 $I\%$ 或 α 数据为 $68 \times 170 = 11560$ 个. 采用插值表，按范围分三个区间插值,每区间按量化精度 $\Delta\Phi$ 分四个粗表，每表有 18 个 $I\%$ 或 α，相应每个 $\Delta\Phi$ 表有 10 个 $\Delta I\%$ 或 $\Delta\alpha$，则存储空间总数为

$$N_v = \frac{68}{4} \times 18 + 3 \times 4 \times 10 = 426 \text{ 个数据},$$

减少内存量 $\frac{11560}{426} \doteq 27$ 倍.

对表 1 计算节省内存量 98%.

(2) 插值表设计. 首先根据变化曲率确定插值区间,从而确定插值表的数目,然后按最小量化步长从表 1 的 Φ、α 列每相邻两个结点中间的数值，确定插值数据个数. 为程序设计方便,应取 2 的整数倍,插值按公式 (7) 计算.

在精插值表组合数据的方法上，可以进一步将精度由一个量化误差提高到半个量化误差,方法如下.

α 表与表 2 数据组合方式:

$$\text{当 } \Delta\Phi = \Delta\Phi_k, \quad \begin{cases} \Delta I_i = \Delta I_0 \ \Delta I_1 \ \Delta I_2 \ \Delta I_3 \ \Delta I_4 \ \Delta I_5 \ \Delta I_6 \ \Delta I_7 \ \Delta I_8 \ \Delta I_9 \\ \qquad\qquad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \Delta\alpha_i = \underbrace{\Delta\alpha_0 \ \Delta\alpha_1 \ \Delta\alpha_2 \ \Delta\alpha_3 \ \Delta\alpha_4 \ \Delta\alpha_5}_{\alpha = \alpha_{i-1} + \Delta\alpha} \ \underbrace{\Delta\alpha_6 \ \Delta\alpha_7 \ \Delta\alpha_8 \ \Delta\alpha_9}_{\alpha = \alpha_i - \Delta\alpha} \end{cases}$$

表 1 与其精值表的数据组合方式:

$$\text{当 } \Delta\theta = \Delta\theta_k, \quad \begin{cases} \Delta\alpha_j = \Delta\alpha_0 \ \Delta\alpha_1 \ \Delta\alpha_2 \ \Delta\alpha_3 \ \Delta\alpha_4 \ \Delta\alpha_5 \ \Delta\alpha_6 \ \Delta\alpha_7 \\ \qquad\qquad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \Delta\Phi_j = \underbrace{\Delta\Phi_0 \ \Delta\Phi_1 \ \Delta\Phi_2 \ \Delta\Phi_3 \ \Delta\Phi_4}_{\Phi = \Phi_{i-1} + \Delta\Phi} \ \underbrace{\Delta\Phi_5 \ \Delta\Phi_6 \ \Delta\Phi_7}_{\Phi = \Phi_i - \Delta\Phi} \end{cases}$$

(3) 插值表查找算法. 插值表也是多变量函数,具体是对三个变量函数查找,要进行二次判断决定可能四个结点值之一,并决定插值是加进还是减去. 对图 2(a)，任意点 α 可能有 α_1、α_2、α_3、α_4 四个结点可能的粗值和插值组合,用 (8) 式表示:

$$\alpha = \begin{cases} \alpha_1 + \Delta\alpha, & [\Delta\Phi < 0.1\text{ms}, \Delta I < 2.5\%], \\ \alpha_2 + \Delta\alpha, & [\Delta\Phi \geqslant 0.1\text{ms}, \Delta I < 2.5\%], \\ \alpha_3 + \Delta\alpha, & [\Delta\Phi \geqslant 0.1\text{ms}, \Delta I \geqslant 2.5\%], \\ \alpha_4 + \Delta\alpha, & [\Delta\Phi < 0.1\text{ms}, \Delta I \geqslant 2.5\%]. \end{cases} \tag{8}$$

对图 2(b)，任意点 Φ 可能有 Φ_1、Φ_2、Φ_3、Φ_4 四个结点的可能粗值按 (9) 式和插值组合,即:

$$\Phi = \begin{cases} \Phi_1 + \Delta\Phi, & [\Delta\theta < 0.1\text{ms}, \ \Delta\alpha < 0.1\text{ms}], \\ \Phi_2 + \Delta\Phi, & [\Delta\theta \geqslant 0.1\text{ms}, \ \Delta\alpha < 0.1\text{ms}], \\ \Phi_3 + \Delta\Phi, & [\Delta\theta \geqslant 0.1\text{ms}, \ \Delta\alpha \geqslant 0.1\text{ms}], \\ \Phi_4 + \Delta\Phi, & [\Delta\theta < 0.1\text{ms}, \ \Delta\alpha \geqslant 0.1\text{ms}]. \end{cases} \tag{9}$$

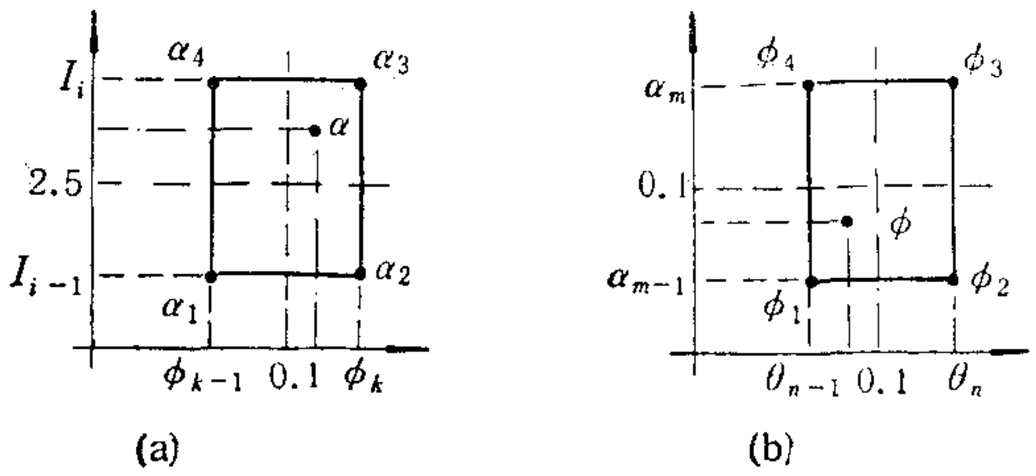


图 2　插值算法

三、数据表的存储结构与数据组织

获得数据表之后，程序设计取决于最优的数据存储结构及数据组织．存储器只能串行存数，即是一维的．现在面对粗值和精值表都是三维的，三维数据表的存储结构是这样组织的：把要查找的目的维(即获取结果那一维)按次序存于存储器，其它二维作为计算确定查找目标维入口地址用，这能最大限度节省存储空间．

1. 粗值表存储结构

对 α 表设计存储结构如图 3，α 目标维按其本身数据顺序存放于 T_1、T_2、T_3···区，每区对应于 Φ 维的一个数据，并自构成一个数据表，所以表中一个 Φ 数据唯一地确定一个区的入口地址，由入口地址加上第二维数据 $I\%$ 值就唯一地查找到目标维的结果值 α．

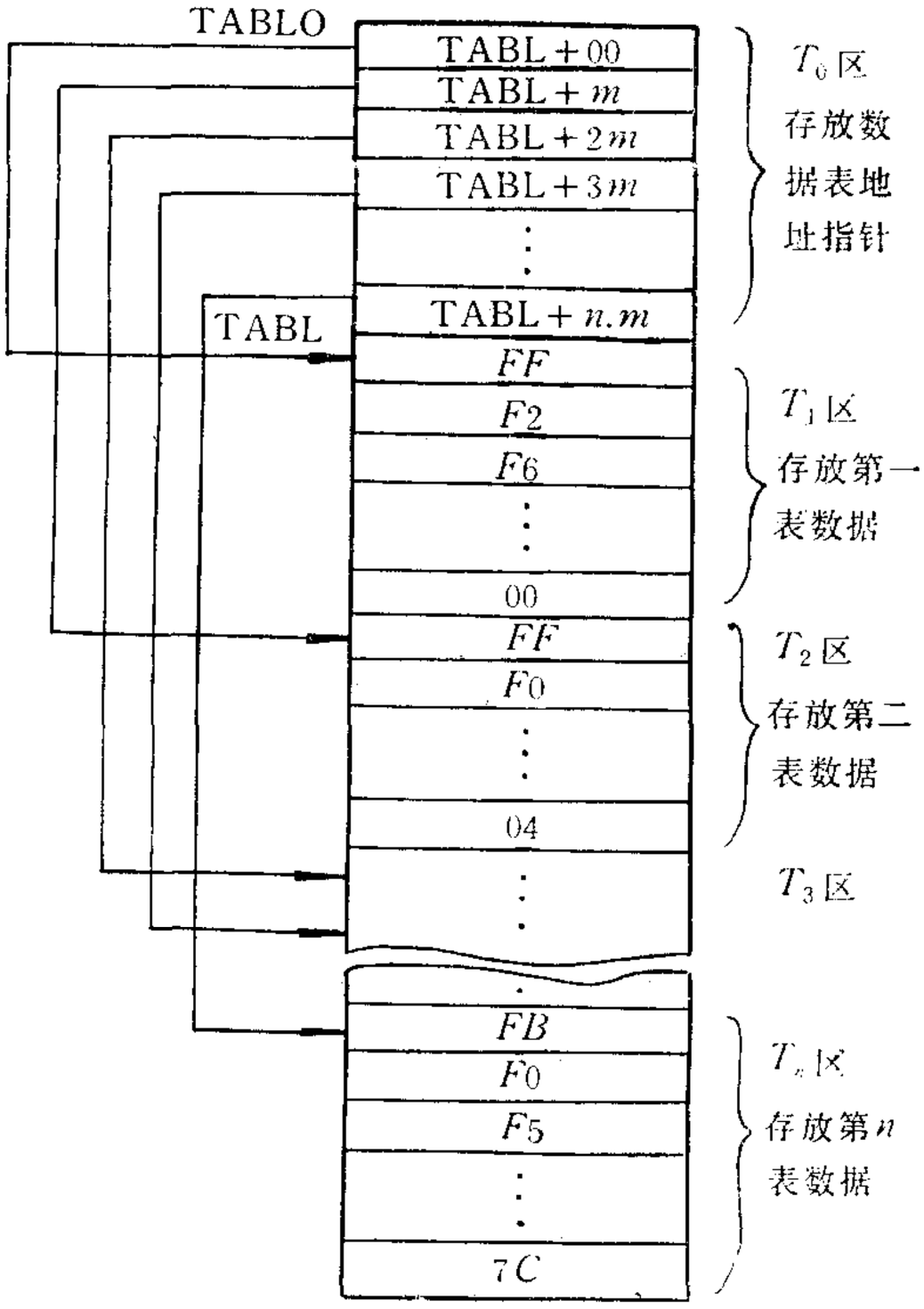
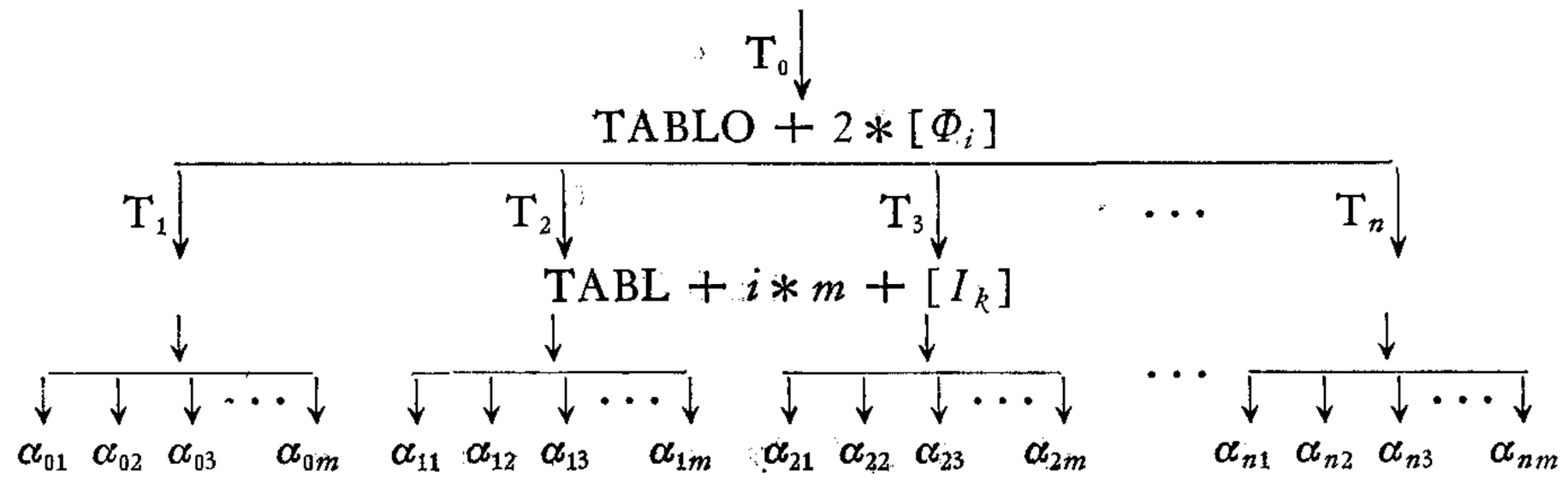


图 3　粗值存储结构

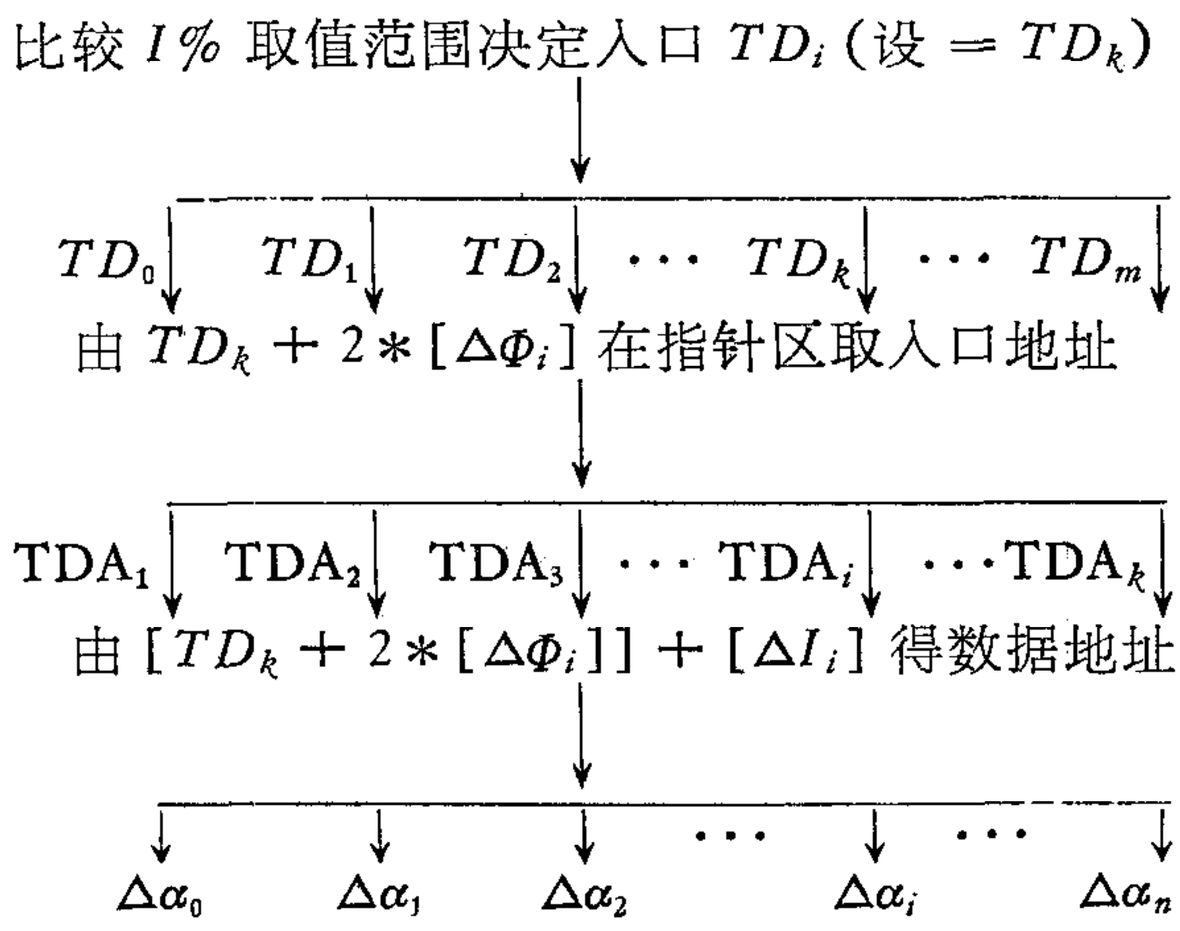
T_0 为地址指针区，每两个单元存放一个目标维 α 表区的入口地址. 各表区数据长度相等设为 m，T_0 区存放的地址内容为 TABL+00，TABL+m，TABL + 2m，TABL+3m···. TABL 为数据第一个表区的入口地址，计算机程序查找过程如下：



其中 $[\Phi_i]$ 是第 i 个 Φ 值经移位处理过的整数，它是 1，2，3……，最大值小于 255. $[I_k]$ 是第 k 个 $I\%$ 被移位及处理过的整数值，范围同前. 对表 1 数据区的长度各不相同，此时 T_0 区分别存放各表区的实际长度再加上总的入口地址作为数据表的入口地址指针.

2. 插值表——精值存储结构

所有插值表都是三维的，设计的存储结构如图 4 示，各个插值表的入口地址分别为 $TD_0, TD_1 \cdots TD_m$，每个表结构分指针区和数据区. 指针区每项放 16 位精值表的入口地址，数据区按表顺序存放数据，每个数据表的长度不相同，$0^{\#}$ 表 TAB_1 长度 L_1，TAB_2 长度 L_2，···. $k^{\#}$ 表 TAC_1 长度 k_1，TAC_2 长度 k_2，···. $m^{\#}$ 表各表长度分别为 m_1，m_2，··· 等. 以精插值表 2 $\Delta\alpha = f(\Delta\Phi, \Delta I\%)$ 为例，查找过程如下：



3. 数据结构组织

图 3 的存储结构中，如何查找 T_0 区的地址指针问题就是计算确定目标数据的地址问题，这取决于除目标维以外的其它两维数据的结构组织.

对 Φ 的结构组织可以把 α 表中所有 Φ 的粗值数据集合在一起，做为第一项，减去初值得第三、四项. 由于粗值结点的精插值设计为 2 的整数倍 8，故只要将输入的任何 Φ 值右移三位，移出的值存于另一寄存器中，就代表精值，记为 $[\Delta\Phi]$；而留在本寄存器的便是粗值，并形成顺序整数 0，1，2，……16，记为 $[\Phi]$，这样就实现了粗精分离. 将图 3 T_0 区的起址 TABLO 加上 $2 * [\Phi]$ 便得 T_0 区的入口地址，取其内容得目标数据 α 表的

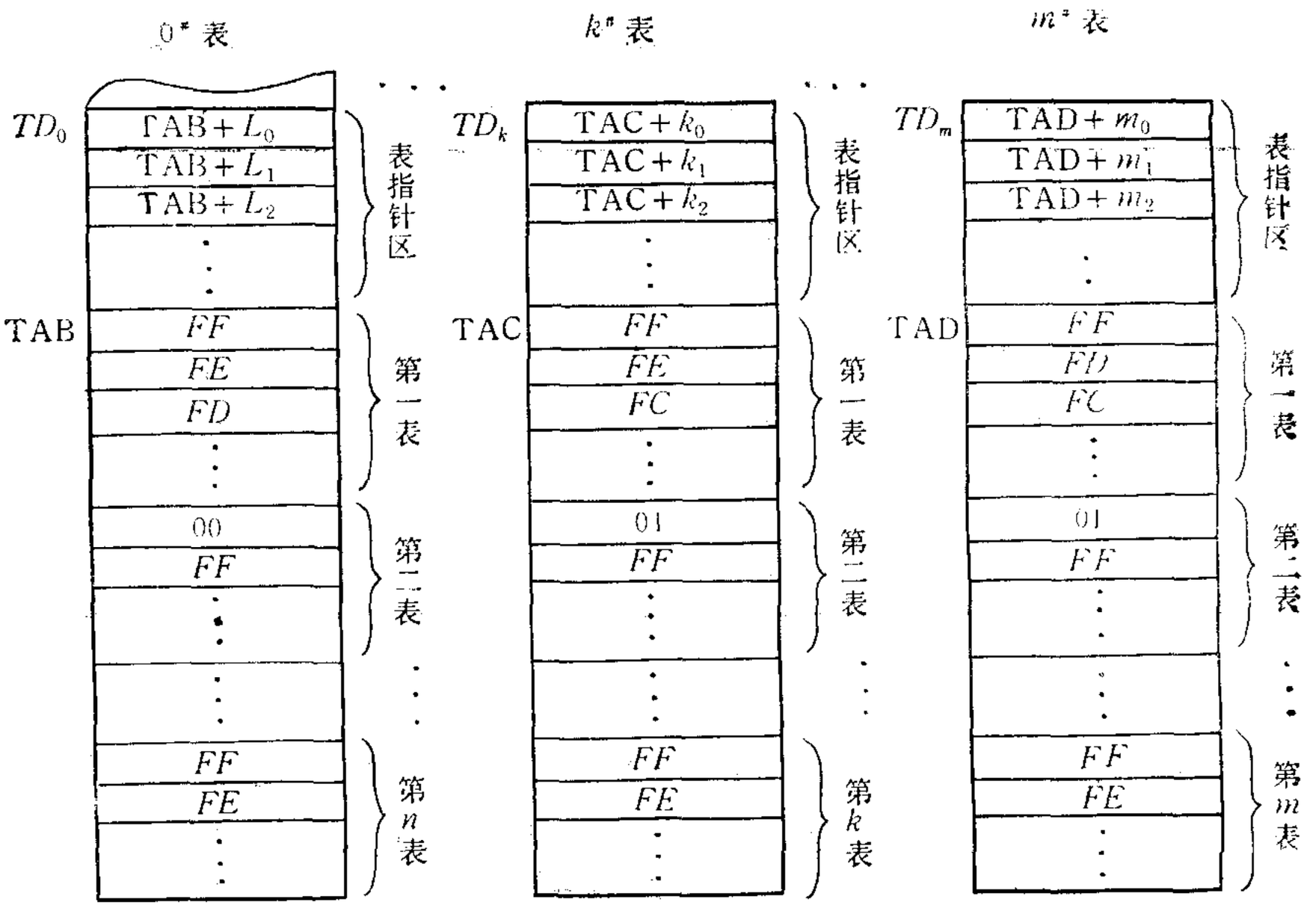


图 4　插值表——精值存储结构

表 2　$\Delta\alpha = f(\Delta\phi, \Delta I\%)\ I\% \leqslant 60\%$

$\Delta\phi$		$\Delta I\%$		$\Delta\alpha$		
ms	HEX	BCD	HEX	ms	BCD	HEX
0.00	00	0.5	01	−0.028	−1	FF
0.00	00	1.0	02	−0.056	−2	FE
0.00	00	1.5	03	−0.084	−3	FD
0.00	00	2.0	04	−0.112	−4	FC
0.00	00	2.5	05	−0.140	−6	FA
0.00	00	3.0	06	+0.112	+4	04
0.00	00	3.5	07	+0.084	+3	03
0.00	00	4.0	08	+0.056	+2	02
0.00	00	4.5	09	+0.028	+1	01
0.05	01	0.0	00	+0.005	00	00
0.05	01	0.5	01	−0.023	−1	FF
0.05	01	1.0	02	−0.051	−2	FE
0.05	01	1.5	03	−0.079	−3	FD
0.05	01	2.0	04	−0.107	−4	FC
0.05	01	2.5	05	−0.135	−5	FB
0.05	01	3.0	06	+0.117	+5	05
0.05	01	3.5	07	+0.089	+4	04
0.05	01	4.0	08	+0.061	+2	02
0.05	01	4.5	09	+0.033	+1	01
0.10	02	00	00	+0.010	+0	00
0.10	02	05	01	−0.018	−1	FF
⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 3 $I\%$ 值结构组织

$I\%$	减初值	序 值	$I\%$	减初值	序 值
15	00	0	60	45	9
20	05	1	65	50	10
25	10	2	70	55	11
30	15	3	75	60	12
35	20	4	80	65	13
40	25	5	85	70	14
45	30	6	90	75	15
50	35	7	95	80	16
55	40	8	100	85	17

入口地址.

$I\%$ 的数据结构组织,以 5% 作为粗值分档(小于 5% 作为精插值),将其列于表 3 的第一项,减去初值为第二项. 它为两位 BCD 码,可以直接巧妙地处理为十六进制的顺序整数,列于表中的第三项. 方法是将十位和个位的 BCD 码分离,将十位的 BCD 码在寄存器中右移三位,然后比较个位. 若个位值大于等于 5,则在移位后的十位寄存器中加 1,并在个位寄存器中减 5. 这样十位寄存器中的数就自然成为十六进制的顺序整数,记为 $[I]$,个位寄存器中存放的正好就是精值 $[\Delta I]$. 这样,粗值的查找便为

$$\alpha_{粗} \leftarrow [\mathrm{TABLO} + 2 * [\phi]] + [I].$$

精值的查找首先根据给定 $I\%$ 的范围决定一个精插值表的入口地址,设为 TD_k,则查找为

$$\Delta\alpha \leftarrow [TD_k + 2 * [\Delta\phi]] + [\Delta I].$$

真实的 α 值为

$$\alpha = \alpha_{粗} + \Delta\alpha.$$

四、程 序 设 计

通过以上工作分析,程序设计分两部分:

1. 建立数据库

根据数据表的存储结构将表数据存入指定的内存空间,用汇编语言伪指令定义语句可以将表数据依次存入内存空间的单元中.

定义和存入 α 表粗值的存储结构数据:

```
        ORG   ××××H
TABLO:  DEFW  TABL + 00     ;存放指针区 T0,共有 17 个粗
        DEFW  TABL + 18      值数据入口地址,每个数据
        DEFW  TABL + 36      表有 18 个数据.
        DEFW  TABL + 54
          ⋮
```

```
TABL:   DEFB 316-61          ; 17 个表数据依次排列，其中 61、
        DEFB 302-60            60 均为表的初值
        DEFB 290-60
        DEFB 277-60
        :
        DEFB 184-60
```

图 4 的精值结构对表 4 有：

```
        ORG  ××××H
TDO:    DEFW TAB + 00 ┐
        DEFW TAB + 10 │  ; I% ≤ 60% 范围共有四个
        DEFW TAB + 20 │    精值表，表长为 10 个数
        DEFW TAB + 30 ┘    据

TAB:    DEFB   00 ┐
        DEFB  −01 │
        DEFB  −02 │
        DEFB  −03 │
        DEFB  −04 │
        DEFB  −06 │
        DEFB   04 │
        DEFB   03 │
        DEFB   02 │
        DEFB   01 │
        DEFB   00 ┘        ; 其它精值表数据
          :
TD1:    DEFW TAC + 00      ; 60% < I% ≤ 70% 范围四个精
          :                  值表
TAC:    DEFB 00            ; 精值表数据
          :
TD2:    DEFW TAD + 00      ; I% > 70% 范围四个精值表
          :
TAD:    DEFB 00            ; 精值表数据
          :
```

2. 设计查找程序

根据数据表的存储结构和数据结构的组织情况可以设计出查找程序。对 $\alpha = F(I\%, \Phi)$ 求解查找程序的流程图如图 5 所示。

对于求 $\Phi = G(\theta, \alpha)$，表数据的存储结构相同，数据结构的组织方法相同，只是数据

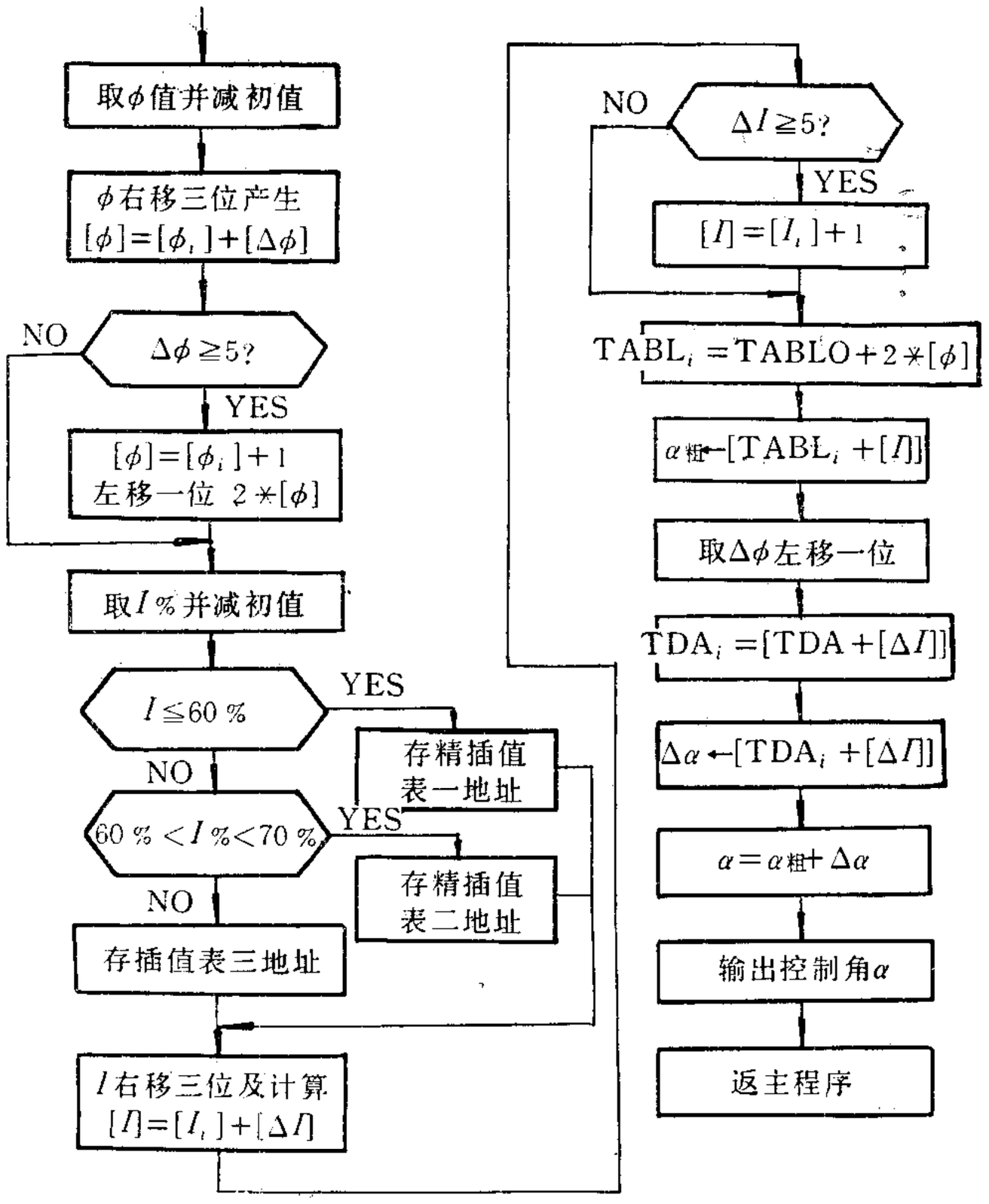


图 5 $\alpha = F(I\%, \phi)$ 查找程序流程图

值、表的长度和表数目不同，而程序设计的思想和形式与图 5 流程图是相类同的.

五、结 论

全式表格函数求解法，借用大计算机和高级语言作开发手段，解决了具有多变量复杂函数的过程控制表达式求取的困难，通过粗精组合的多变量函数表结构设计，大大地节省了内存空间，并在 8 位微机中能保证获得足够的 8 位程序设计精度. 在所构思的存储结构安排和数据结构的组织下，在时钟 2MHz 的 Z80 微机控制系统中，汇编语言程序设计获得的结果是：求解 $\alpha = F(I\%, \Phi)$ 三个变量过程需时 344μs，求解三变量过程 $\Phi = G(\theta, \alpha)$ 需时 227μs，即求解复杂多变量函数式 (1)—(3) 共只需 571μs；而前述只求解一次 $\sin x$ (或 $\cos x$) 就需 1400μs，作一次有符号的乘法就需 350μs，可见本方法研究的重要意义. 两个查找程序占存储空间 240 字节，数据表及地址 1093 字节，共占空间 1333 字节. 程序长期运行两年多，工作正确可靠.

邱义林助理工程师、吴 禄讲师为本程序设计研制作了大量具体工作，在此深表感谢.

参 考 文 献

[1] Li Kangbao, et al., Research on Microprocessor Controller for Spot Welding, The International Conference

on Quality and Reliability in Welding, Collection Volume 4, D-2, June, 1984.
[2] Lance A. Leventhal, Z80 Assembly Language Programming, 1979, by Adam Osbome & Associates, Inc.
[3] 王广芳等，数据结构，湖南科学技术出版社，1984，4.
[4] 莫勒 H. H.，数据结构与程序设计技术，孙永强、张　然译，科学出版社，1984，8.
[5] 张德荣等，计算方法与算法语言，人民教育出版社，1982，4.
[6] 黎康保，微型计算机在飞机空中瞄准系统中的应用，西北工业大学学报，3(1985)，第二期.

RESEARCH ON MICROCOMPUTER PROGRAMMING PERFORMED IN REAL-TIME FOR COMPLICATED CONTROL PROCESS WITH MULTI-VARIABLE FUNCTION

Li Kangbao

(Northwestern Polytechnical University)

Abstract

While performing microcomputer control in real-time for complicated control process with multi-variable function, some very difficult problems are often met: Those, for instance, the expected relations between the master control parameters and that of the slave may be hard to reduce; the processing is so complicated that the computer operating speed is far from satisfying the requirements of the real-time system; the size of memory capability needed is too vast to realize by the hardware, etc. In this paper, a practical example of programming for the process with 3 to 4 variables is discussed. The programming is easy, the computing accuracy is higher, and the operating speed is more than tens of times faster as compared with the analytic solution and data-table look-up for a single function, and also saves memory area as well. In this way, a single board microcomputer is capable of performing the control of a process with complicated multi-variable.