微型、小型计算机
应用软件汇编

Application Software

中国科学院成都计算机
应用研究所情报室主编

前　言

《微型、小型计算机应用软件汇编》一书是继1984年《微型、小型计算机应用软件汇编》出版之后，又继续全面系统地收集了微型、小型计算机在各个领域中的应用软件，为推广和开发我国计算机的应用；满足广大从事计算机应用工作者的迫切需要编纂的。我们希望它能成为计算机应用工作者开拓新的应用领域、发展新的应用水平、获取成果捷径不可少的工具。

该书是从近两年来国内发表的有关期刊、资料、会议文献等论文中优选（适当照顾了面）编制成册的，该书内容丰富，有较高的实用价值，是一本难得的内部参考性的情报资料。

该书主点是微型、小型计算机在各领域中的应用软件（包括应用系统软件）和一些编程方法的实践经验。共分上、下两册。

上册内容有：科学工程计算、辅助设计、管理（包括数据库管理）、控制系统、测量、仪器、智能。

下册内容：数据处理与采集、图形、图象、字处理、网络通信、其它、编程技巧方法及实践。

该书因篇幅有限，所选出的文章不能全文登载，但提供了原文出处和省略部份的注明。

在资料的汇集中，因时间关系未一一征求作者意见，有不妥的地方，敬请作者原谅，并对作者积极支持此工作表示衷心感谢。

由于水平有限及时间关系，搜集的资料很不完全，错误难免，敬请读者指正。

参加该书编审、提供资料的同志有：

高树清　凌婉妮　张富容　蔡菊英　徐明保　罗　淳　刘素琴

周永培　单永涛　杨明芳等同志

该书统编：杨明芳、单永涛

制　　图：李晓云

编　　者

1987.8

《微型、小型计算机应用软件汇编》

（上　册）

目　录

科学、工程计算

辅助设计

管理（包括数据库管理）

控制系统

测量、仪器、智能

此为试读,需要完整PDF请访问: www.ertongbook.com

文献名称： **一个用最小乘法拟合曲线的实用程序**

作　　者　赵　璘

使用机型　IBM—PC/XT

使用语言　GWBASIC

用　　途　求拟合曲线表达式

内　　容

问题的提出

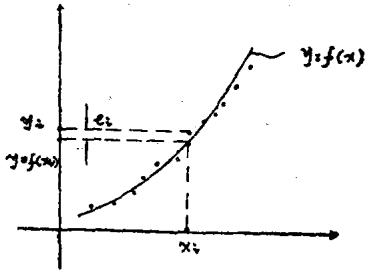
我们知道，在新产品研制，弹道分析及测量工作中，函数有很多是用表格方式给出的。例如：通过实验观测，得出了某个函数$y=f(x)$，在一系列点x_0、x_1、x_2……上的函数值y_0、y_1、y_2……。但对应于x的其它函数值是未知的，这种表格函数不便于分析其性质和变化规律，特别是还不能直接求出表中没有列出的点的函数值，而这种表格函数往往间隔较大，精度较差，因此，人们常希望能根据这些表格数据找到一个能反映数据趋势的近似曲线及表达式来反映原来的函数，这类问题称为求曲线拟合的问题。在各种科学试验及测量工作中有着广泛的应用。

我厂某产品在设计生产中，有一零件形状为杯状体，精度要求高，加工难度大，但该零件只有一组表格数据，且数量少，间隔大，根本不能满足设计要求和工艺要求，因此，需要根据这一组表格函数来得到一个近似表达式，其误差须保证在0.01mm之内，为此，我们采用了次数较低的多项式作为$y=f(x)$的函数，采取的计算方法就是最小二乘法。

最小二乘法简介

为了更好的理解程序，先简单介绍一下最小二乘法。

最小二乘法是用$y=f(x)$来拟合一组数据点(x_1,y_1)、(x_2,y_2)、……(x,y)的常用方法。这个方法的基本思想是使误差平方和：$\delta_1^2+\delta_2^2+\delta_3^2+\cdots\cdots\delta_n^2$减至最小值，故又称最小平方法。这里的$\delta_i$是指第i项误差，即对于给定的$x_i$，$\delta_i$是数据点$y_i$和数据$y=f(x_i)$之间的差，$y=f(x)$是依据拟合曲线求出的，如下图所示：



我们用多项式：$y=c_1+c_2\cdot x+c_3\cdot x^2+\cdots\cdots C_{n+1}\cdot x^n$来通过M个数据点，为此，我们必须求解下列方程组：

$$\begin{cases} M\cdot C_1+\left(\sum_{i=1}^{M}Xi\right)\cdot C_2+\left(\sum_{i=1}^{M}Xi^2\right)\cdot C_3+\cdots\cdots+\left(\sum_{i=1}^{M}X_i^n\right)\cdot C_{n+1}=\sum_{i=1}^{M}y_i \\ \cdots\cdots \qquad \cdots\cdots \qquad \cdots\cdots \\ \left(\sum_{i=1}^{M}X_i^n\right)\cdot C_1+\left(\sum_{i=1}^{M}X_i^{n+1}\right)\cdot C_2+\left(\sum_{i=1}^{M}X^{n+2}\right)\cdot C_3+\cdots+\left(\sum_{i=1}^{M}X_i^{2n}\right)\cdot C_{n+1}=\sum_{i=1}^{M}X_i^n\cdot y_i \end{cases}$$

8810168

然后确定系数c_1、c_2……c_{n+1}

本程序采用了矩阵求逆的方法解这一组线性方程组，式中C_1～C_{n+1}为要确定的系数，也是本程序所要解决的问题。

程序要点

程序中由行号170～320来完成确定未知常数的系数的工作、由350～600来完成系数矩阵逆矩阵的工作，由行号830～940来完成逆矩阵与右边向量的乘积，从而得出结果。最后由1000～1060来完成输出。

附程序清单如下：

```
 5  CLS
10  PRINT "n, m, d=";
20  INPUT N,M,D
30  DIM X(D), Y(D), A(N, N),
    D(N), B(N, N), V(N, 2×N),
    C(N,N)
50  FOR I=1 TO 141
60  READ X(I),Y(I)
70  NEXT I
80  N1=N
90  FOR I=1 TO N1
100 FOR J=1 TO N1
110 A(I,J)=0
120 NEXT J
130 NEXT I
140 FOR I=1 TO N1
150 D(I)=0
160 NEXT I
170 FOR I=1 TO N1
180 FOR J=1 TO N1
190 IF I+J>2 GOTO 220
200 A(I,J)=D
210 GOTO 250
220 FOR K=1 TO D
230 A(I, J)=A(I, J)+X(K) ^ (I+J
    -2)
240 NEXT K
250 NEXT J
260 FOR K=1 TO D
270 IF I>1 GOTO 300
280 D(I)=D(I)+Y(K)
290 GOTO 310
300 D(I)=D(I)+Y(K)×X(K) ∧ (I-
    1)
310 NEXT K
320 NEXT I
350 FOR S=1 TO N1
360 FOR J=1 TO N1
370 V(S,J)=A(S,J)
380 IF J=S GOTO 410
390 V(S,J+N1)=0
400 GOTO 420
410 V(S,J+N1)=1
420 NEXT J
440 NEXT S
460 FOR S=1 TO N1
470 FOR T=S TO N1
480 IF V(T,S)< >0 GOTD 520
490 NEXT T
500 PRINT "n0"
510 GOTO 1090
520 GOSUB 620
530 C=1/V(S,S)
540 GOSUB 680
550 FOR T=1 TO N1
560 IF T=S GOTO 590
570 C=-V(T,S)
580 GOSUB 720
590 NEXT T
600 NEXT S
610 GOTO 760
620 FOR J=1 TO 2×N1
```

```
630 H = V(S,J)
640 V(S,J) = V(J,J)
650 V(T,O) = H
660 NEXT J
670 RETURN
680 FOR J=1 TO 2×N1
690 V(S,J) =C×V(S,J)
700 NEXT J
710 RETURN
720 FOR J=1 TO 2×N1
730 V(T,J) = V(T,J) +C×V(S,J)
740 NEXT J
750 RETURN
760 PRINT
770 FOR J=N1+1 TO 2×N1
780 FOR I=1 TO N1
790 B(I,J-N1) = V(I,J)
800 NEXT I
810 PRINT
820 NEXT J
830 FOR I=1 TO M
840 FOR J=1 TO 1
850 C(I,J) = D(I)
860 NEXT J
870 NEXT I
880 PRINT J
890 FOR I=1 TO N1
900 FOR J=1 TO 1
902 F(I,J) = 0
904 FOR K=1 TO M
910 F(I,J) = F(I,J) +B(I, K)×C(K,
    J)
920 NEXT K
930 NEXT J
940 NEXT I
950 FOR J=1 TO 1
960 FOR I=1 TO N1
970 PRINT F(I,J)
975 T(I) = F(I,J)
980 NEXT I
990 PRINT
1000 NEXT J
1010 PRINT "Y=" ; T(1) ; "+";
     T (2); "×X+" ; T(3); "×X
     ^2+" ; T(4); "×X^ 3+" ; T
     (5); "×X^4+" ; T(6); "×X
     ^5+" ; T(7); "×X^6+" ; T
     (8); "×X^7"
1011 PRINT
1012 PRINT
1013 PRINT
1015 PRINT "X(I)" , "Y(I)" , "Y"
     , "K"
1020 FOR I=1 TO D
1030 X = X(I)
1040 Q = T(1) + T (2)×X + T (3)×X^
     2+T(4)×X^3+T (5)×X^4+T
     (6)×X^5+T (7)× ^6+T (8) ×
     X^7
1050 R = Y(I) -Q
1053 PRINT X(I), Y(I), Q,
1055 PRINT USING "***, ****" ,
     R
1057 NEXT I
1060 END
1410 DATA 28.0,2.525,27.8,2.334,
     27.6,2.144, 27.4, 1.955, 27.2,
     1.767
1420 DATA 27.0,1.581,26.8,1.396,
     26.6, 1.212, 26.4, 1.030, 26.2,
     0.848
1430 DATA 26.0,0.668,25.8,0.489,
     25.6,0.311, 25.4, 0.134, 25.2,
     -0.042
```

```
1440 DATA  25.0, -0.216, 24.8,
     -0.390, 24.6, -0.562, 24.4,
     -0.733, 24.2, -0.903
```

文献出处：《红光技讯》1986年1期 1—3页

文献名称 一个最佳分配问题—程序设计中的回溯算法

作　　者 马金良　兰州铁道学院电信系

使用机型

使用语言 BASIC

用　　途 用于解一些组合数相当大的问题

内　　容

本文给出了求解最佳分配问题最优化解的回溯算法，并用 BASIC语言 编写了这一程序，目的是想给出在 BASIC 语言中，实现回溯算法的一个程序模式。

1． 数学模型。设某车间有n个工人，n 个作业任务。每个工人完成某一项任务的时间用矩阵T表示。$T=(T_{i,j})$，$i=1,2,\cdots\cdots n$；$j=1,2,\cdots\cdots n$。$T_{i,j}$表示第i个工人完成第j个任务所需要的时间。工人对任务的分配方案用矩阵 X 表示。$X=(X_{i,j})$，$i=1,2,\cdots\cdots n$；$j=1,2,\cdots\cdots n$。如果第i个工人分配给第j个作业，则 $X_{i,j}=1$，否则$X_{ij}=0$。并且规定是没有一个工人接受两个任务，反之亦然。这样整个任务的完成所需时间

$$T^{*}=\sum_{j=1}^{n}\sum_{i=1}^{n}T_{ij}X_{ij} \qquad \cdots\cdots(1)$$

最优化问题是，如何分配这此任务，使T^{*}取得最小值。

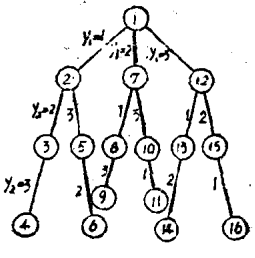
2． 算法分析。为了避免穷举法，我们采用回溯算法。回溯算法在程序设计中占有十分重要的地位。它不是按固定的计算公式，而是通过搜索。这种方法适用于解一些组合数相当大的问题。例如，在本问题中，每个工人对n 个任务的选取方法有n种，因此，n个工人对n个任务的选取方案共有n^{n}种。要从众多的方案中，选取一个最佳方案，并非易事。

为了简化计算，我们引入一个向量y（y_1，x_2，$\cdots\cdots y_n$，）；其中 y_i 的值表示第i个工人被分配给的任务。显然，$1\leqslant y\leqslant n$；$i=1,2\cdots\cdots n$。这样一来，（1）式即变为

$$T^{*}=\sum_{i=1}^{n}T_{i}y_{i} \qquad \cdots\cdots\cdots(2)$$

最优化问题就变成了寻找n 元组（1，2……n）的某种排列，其中任何一种排列都代表了一种对作业的分配方案，而使（2）式最小的那种排列，就是最优化介。这样，我们就从计算 n^{n} 个组合数减少到 $n!$ 。

我们已经看到，寻找满足某些条件（最大或最小）的解，通常情况下，这些候选解是很多的，组成一个解集，构成一个任务树，这个任务树往往增长的很快。使用这种解空间的任务树，搜索就会容易实现。下面我们以工人和任务个数 n = 3 为例，画出其任务树。

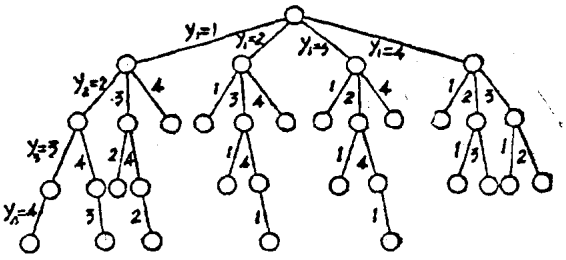


任务树的每一条边上标有一个yi的可能值。比如最左边的一枝，就表示给第i个工个分配第i个任务。对于n = 3，分配方案共有3！ = 6种，分别对应了任务树的六个不同分枝。搜索按下述方法进行，从根开始产生任务树的各节点。一个正在产生儿子的节点称为扩展节点。如果对一个扩展节A，一旦产生了它的儿子B，就把B当作新的扩展节点。在完成对子树B（以B为根的子树）的穷尽搜索之后，将A重新变成扩展节点，继续生成A的下一个儿子。这样方法称为深度优先状态生成法。上图中任务树各节点的编号就是较深度优先原则给出的。

3. 算例。回溯算法的优点是建立和扫描任务树时，不断用约束条件来判死那些不可能产生最佳解的节点。以减少工作量。否则就会变成穷举法。下面给出一个具体算例，进行说明。设 n = 4，工人完成每个任务所需时间矩阵 T为：

$$T = \begin{pmatrix} 5 & 15 & 22 & 18 \\ 25 & 10 & 9 & 35 \\ 17 & 37 & 25 & 6 \\ 16 & 30 & 18 & 24 \end{pmatrix}$$

选取初值 $R = \sum_{i=1}^{4} T_{i,i} = 64$。在进行搜索时，按深度优先原则对树的每一枝都进行判断，当所需的时间T*大于R时，搜索停止，并判死这一节点，不再向下生成儿子（如果存在的话）。否则，将此时间T*做为R的新值（R总是取最小时间）。重复上面过程，继续生成和扫描另一枝。下面的图是本算例按回溯算法实际生成的状态任务树。



任务树的第二枝（$y_1 = 1$，$y_2 = 2$，$y_3 = 4$，$y_7 = 3$）化费时间 T* = 39，小于R(R = 64)，故R取新用39。第三枝（$y_1 = 1$，$y_2 = 3$，$y_3 = 2$）化费时间T* = 51。T*大于R值，故搜索停止，判死该节点，不再向下生成儿子。其它情况完全类似。通过运行程序可知，最佳分配方案是 y = (1, 2, 4, 3)；任务完成最小时间 T* = 39。

回溯算法的BASIC程序如下：

```
5  INPUTN: DIM  Y(N), S(N)
V(N), T(N,N)
15  FOR I=1 TON
20  FOR J=1 TON
25  INPUT T(I,J)
30  NEXT  J
35  NEXT  I
40  FOR  I=1  TO  N:R=R+
T  (I,I):NEXT  I
50  FOR  I=1  TO  N:V(I)=
I:NEXT I
60  K=1:Y(K)=0:S(O)=0
65  Y(K)=Y(K)+1
70  1FY(K)>N THEN 130
80  GOSUB 500
85  1FY(K)>N THEN 130
90  S(K)=S(K-1)+T(K,Y(K))
100 1FK<N  AND  S(K)<R
THEN  200
110 1FK<N THEN 65
115 1FS(K)>=R THEN 130
120 R=S(K)
125 FOR I=1  TON:V(I)=Y(I)
```

```
:NEXT I
  130 K=K-1
  135 IF K=0 THEN 150
  140 GOTO 65
  150 FOR I=1 TO N:PRINT
V(I):NEXTI
  160 PRINT "MINI TIMET=";R
  165 END
  200 K=K+1:Y(K)=0:
GOTO65
  500 FOR I=1 TO K-1
  510 IFY(I)=Y(K) THEN 540
  520 NEXTI
  530 RETURN
  540 Y(K)=Y(K)+1:GOTO 500
```

文献出处：《计算机时代》1986年第3期22—23页

编者注：本文还有《参考文献》等略

文献名称 **微处理机计算的线性插值程序**

作　　者 徐有福　二军医大第一附属医院

使用机型 微处理机

使用语言 Z80汇编语言

用　　途 适于微处理机简捷、快速、高精度的计算程序

内　　容

微处理机在用于工业控制和数据处理中，经常会遇到线性插值运算。在计算数字中，线性插值法虽然有现成的公式和计算框图，但是运用这种算法，每插补一点需要进行乘、除运算，计算量大。由于微处理机指令中没有乘、除指令，为了进行乘除运算还需编制乘、除子程序，这样既占存贮器空间，又增加了运算时间，不能实现实时处理目的。本文介绍一种适合微处理机计算的简捷、快速、高精度的线性插值方法。

如图1所示，如果已知两点A、B，在中间要插补N－1个相等间隔的点。

下面为了说明问题方便起见，我们令

y(i)　　为插补值。i＝1，2，…N－1。

Δy＝y(N)－y(0)，　　为A、B两点之间的差值。

N　　，　　为A、B两点之间的间隔数。

ΣΔy　　，为累加和。

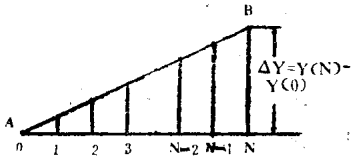


图1　线性插值示意图

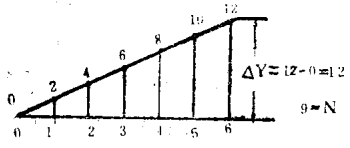


图2　线性插值的一个例子

这一方法的要点是从第一点开始，每插补一点，加上差值Δy与间隔数N比较，如果大，然后作减法，每减去一个N，插补值y(i)加1；如果小，则不减，插补值保持不变，然后进行下一点插补。

此为试读，需要完整PDF请访问：www.ertongbook.com

为了更好地说明这一问题，让我们来看一个简单的例子。在这个例子中，y(0)=0，y(6)=12，N=6。（图2）

插补从第一点开始，一开始累加和ΣΔy置零，插补值y(i)=y(0)=0。

1. 插补第一点，累加和加差值作为新累加和，结果为0+12=12，与间隔数N比较，如大于或等于N，然后作减法，每减去一个N，y(i)加1。现在因12>6，所以将12-6=6作为新累加和，插补值y(i)加1，变为1。因新累加和与N比较，仍够减，所以6-6=0为新累加和，插补值y(i)再加1，变为2。

2. 插补第二点，将第一点插补后的累加和加差值作为新累加和，现为0+12=12，与间隔数N比较，因12>6，所以新累加和为12-6=6，插补值y(i)再加1，变为3。新累加和与N比较，仍够减，所以现在新累加和为6-6=0，插补值再加1，变为4。

……

依此类推，直到最后一点插补结束。插补后的数据表在图2中。

对于线性下降的，也可用上述的方法进行插补，只不过在这种情形下，每减去一个N，插补值是减1而不是加1。

下面是用Z80汇编语言编写的线性插值子程序。

程序的设计是假设需插补的两点A、B，即起点和终点值已分别存放在存贮器地址ADDRO和ADDRN中。在本程序中，寄存器对HL存放插补数据的地址，DE存放间隔数N；BC存放A、B两点之间的差值Δy，，存贮单元COUNT存放插补个数N-1，存贮单元TEMP存放1或FFH（即-1），取决于所插补的点是线性上升还是线性下降，下一个地址存放插补值y(i)。

其程序如下：

```
COUNT   EQU   nn1
TEMP    EQU   nn2
        ORG   2000H
START:  LD    DE, ADDRO,  数据起始地址→DE
        LD    HL, ADDRN,  数据结束地址→HL
        CALL  LINSET      ；调用线性插值子程序
        END   START
```

线性插值子程序LINSET如下：

```
LINSET: LD      BC,0000H；置累加和初值
        PUSH    BC       ；进栈
        PUSH    HL       ；进栈保护
        OR      A
        SBC     HL, DE   } 计算间隔数N
        PUSH    HL       ；间隔数N进栈保护
```

```
        DEC     HL
        LD      (COUNT),HL      }插补个数存入计数存贮单元
        POP     HL      ；间隔数N—HL
        EX      (SP),HL ；数据结束地址→HL，间隔数N→栈项
        EX      DE，HL  ；数据起止地址→HL，结束地址→DE
        LD      A，(DE) ；终点数据y(N)→A
        SUB     (HL)    ；计算差值Δy=y(N)-y(0)
        LD      E，01H  ；1→E
        JR      NC  UPWARD；Δy≥0，转UPWARD
        NEG             ；Δy<0，取绝对值
        LD      E，FFH  ；-1→E
UPWARD；LD      D，(HL) ；插补值y(i)初值→D
        LD      (TEMP),DE；
        LD      C，A    ；差值Δy→BC
        POP     DE      ；间隔数N→DE
        INC     HL      ；第一点插补数据地址→HL
LOOP  ：EX      (SP)，HL；累加和ΣΔy→HL，数据地址→栈顶
        ADD     HL，B  C；累加和加差值作为新累加和ΣΔy=ΣΔy+Δy
COMP  ：OR      A
        SBC     HL，DE  }计算Δy-N
        JR      C  RESUM；不够减，转RESUM
        PUSH    HL      ；新累加和ΣΔy进栈
        LD      HL,(TEMP)
        LD      A，H
        ADD     A，L    /够减，插补值加1或减1
        LD      H，A
        LD      (TEMP),HL}够减。
        POP     HL      ；累加和ΣΔY→HL
        JR      COMP    ；累加和与N继续比较
RESUM；ADD      HL，DE ；不够减，恢复
        EX      (SP),HL；数据地址→HL，累加和ΣΔy→栈顶
        PUSH    HL      ；数据地址进栈
        LD      HL,(TEMP)；
        LD      A，H    ；插补值y(i)→A
        POP     HL      ；数据地址→HL
        LD      (HL)，A ；插补
        INC     HL      ；指向下一个数据地址
```

```
        POSH    HL          ; 进栈
        LD      HL,(COUNT); 插补个数→HL
        DEC     HL          ; 插补个数减1
        LD      (COUNT),HL;
        LD      A, L        }判插补结束否
        OR      H           }
        JR      Z  END      ; 结束，转END
        POP     HL          ;
        JR      LOOP        ; 未结束，继续
END:    POP     HL          ;
        POP     HL          ;
        RET                 ;
```

文献出处：《微型电脑》1985年5期47—49页

文献名称 **水准网测量平差通用程序及其使用说明**
作　　者 练高树　原昆明军区后勤部设计勘测所
使用机型 PC—1500
使用语言 BASIC
用　　途 平差测量
内　　容

一、图形及其编号（图1）

1. 已知点数为K，结点数为M，待定点数为P（不含结点数）。

2. 水准路线数为L。

3. 水准点编号：已知点为A，B，C，……Z。结点编号为a、b，c，……z。待定点水准路线序号和方向编为1，2，3，……，P。（注：结点a必须有一条路线与已知点相连，其余结点也必须有一条路线与序号排在它前面的结点相连，或与一已知点相连，以便推算各结点近似高程。另外，如要评定不是结点的某些点的精度时，可将这些点作为结点）。

4. 各水准路线序号分别为Z_1，Z_2，Z_3，……Z_L。水准路线方向取水准路线起始点和终点（二者均为已知点或结点）。如图1所示，第一条路线 $Z_1 = A - b$（或Ab），……，第四条为 $Z_4 = b - B$（或bB），第六条也为 $Z_6 = b - B$（或bB）。

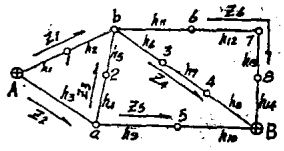


图1

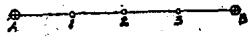


图2

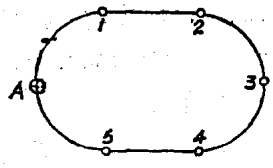


图3

5. 各水准路线的高差段数分别为N_1，N_2，N_3，N_L。

6. 各水准路线中的各段高差值按水准路线序号和方向编为$h_1, h_2, h_3, \cdots\cdots, h_N$ ($N=N_1+N_2+N_3+\cdots\cdots+N_L$)，相应的距离为$L_1$，$L_2$，$L_3$，$\cdots\cdots L_N$(单位：公里)。

二、特殊情况的处理

1. 附合水准路线（图2）

将图中待定点1，2，3中的任何一点作为结点即可平差。

2. 闭合水准路线（图3）

同理将图中待定点1，2，3，4，5任何一点作为结点即可平差。

三、程序功能

本程序可以平差最多由26个已知点，26个结点和若干个水准点组成的水准网。信息和数据一次输入，在输入过程中可以及时校对，及时改错。程序按间接观测平差进行，计算并打印出各结点高程平差值，各待定点高程平差值，各段高差改正数Vhi，每公里高差中误差Mk，各结点高程中误差Mh。

四、程序运行

按机器显示输入各有关数据，输入后立即打印出来，以便校核。每输完一组数据后，机器显示“IS there error？”，此时校核该组打印数据，如无错，打入①，如有错，则打入1，机器啸叫三声后按显示，重新输入该组数据，直至输入完毕。

具体操作是：按RUN或DEF S开工后，显示“K=？”时，打入已知点数。

显示“L=？”时，打入水准路线数，如图1示，L=6。

显示“H_A=？”时，打入已知点A的高程。

显示“Z_1=？”时，打入Z_1路线方向，如图1示，Z_1=Ab-b（或Ab）。

显示“N_1=？”时，打入Z_1路线的高差段数，如图1示，N_1=2。

```
10: "S" CLEAR : WAIT 0
15: INPUT "K=" ; K, "M=" ; M :
    LPRINT "K=" ; K; "M=" ; M
    : K = K - 1
20: INPUT "P=" ; P, "L=" ; L :
    LPRINT "P=" ; P; "L=" ; L
    : L = L - 1
22: S = K + M + P : LF 1
24: DIM H(S), Z(L), N(L), HH(L),
    S(L), U(L)
26: FOR I = 0TO K : A$ = "H" + CHR
    $ (65 + I) + "=" : PRINT A$;
28: INPUT H (I) : LPRINT A$ ; H
    (I) : CLS : NEXT I : LF 1
30: C$ = "IS there error" : PRINT
    C$; : INPUT C
32: CLS : IF C = 1 BEEP 3; GOTO 26
35: FOR I = 0TO L : A$ = STR$ ( I +
    1 ) + "=" : PRINT "Z" ; A$;
38: INPUT Z$ : LPRINT "Z" + A$;
    Z$ : Z = ASC (LEFT$ (Z$, 1) )
39: A = Z - 65 : IF Z)96 LET A = Z -
    96 + K
40: Z = ASC (RIGHT$ ( Z$, 1)) :
    B = Z - 65 : IF Z) 96LFT B = Z -
    96 + K
42: Z (I) = A + B/ 100 : CLS : PRINT
    "N" , A$;
44: INPUT N (I) : LPRINT "N" + A
    $; N(I) : N = N + N (I) : CLS :
    NEXT I : LF 1
46: N( 0 ) = N( 0 ) - 1 : N = N - 1 :
```

```
    PRINT C$;  : INPUT C
48: CLS : IF C=1 BEEP 3 : N=0 :
    GOTO 35
50: DIM H1(N), L(N), P(M-1),
    PF(M-1)
52: FOR I=0TO N : A$="h"+STR
    $(I+1)+"=" : PRINT A$;
54: INPUT H1(I) : LPRINT A$, H1
    (I) : CLS : NEXT I : LF 1
56: PPINT C$; : INPUT C
58: CLS: IF C=1 BEEP 3 : GOTO52
60: FOR I=0TO N : A$="L"+STR
    $(I+1) + "=" : PRINT A$;
62: INPUT L(I) : LPRINT A$; L(I)
    : CLS : NEXT I : LF 1
64: PRINT C$; : INPUT C
66: CLS: IF C=1 BEEP 3 : GOTO60
68: FOR I=0TO L : U=U+N(I) : T
    =U-N(I)+1 : IF I=0LET T=
    T-1
70: FOR J=TTO U : HH(I)=HH(I)
    +H1(J) : S(I)=S(I)+L(J) :
    NEXT J : NEXT I
72: FOR I=K+1TO K+M : O=I-K
    -1 : FOR J=0TO L : T=INT Z
    (J) : U=100*(Z(J)-T)
74: IF T=IOR U=I LET P(Q)=P
    +1/S(J):PF(Q)=P(Q)
76: IF T=IAND U<I LET H(I)=H
    (I)+H(U)-HH(J) : SU=SU+1
78: IF U=IAND T<ILET H(I)=H
    (I)+H(T)+HH(J) : SU=SU+1
80: NEXT J : H(I)=H(I)/SU=0 :
    NEXT I
82: Z=0 : FOR I=K+1 TO K+M :
    O=I-K-1 : SU=0 : PF=0 : FOR
    J=0TO L
84: T=INT Z(J) : U=100*(Z(J)-
    T)
86: IF T=IAND U>K LET PF=PF
    +1/(S(J)+1/PF(U-K-1))
88: IF T=ILET SU=SU+(H(U)-
    HH(J))/S(J) : IF U<=K LET
    PF=PF+1/S(J)
90: IF U=IAND T>KLET PF=PF+
    1/(S(J)+1/PF(T-K-1))
92: IF U=ILET SU=SU+(H(T)+
    HH(J))/S(J) : IF T<=K LET
    PF=PF+1/S(J)
94: NEXT J : H=SU/P(Q)
96: IF ABS(H-H(I)>>1E-50R
    ABS(PF-PF(Q)>>1E-4LET
    Z=1
98: H(I)=H : PF(Q)=PF : NEXT I
100:IF Z=1GOTO 82
102:C=0 : O=K+M : FOR I=0TO
    L : T=INT Z(I) : U=100*(Z(I)
    -T)
104:U(I)=H(U)-H(T)-HH(I) : MK
    =MK+U(I)*U(I)/S(I)
106:C=C+N(I) : S=C-N(I)+1 : IF
    I=0 LET S=S-1
108:FOR J=STO C : L(J)=U(I)*L
    (J)/S(I) : H1(J)=H1(J)+L(J)
109:IF S=CLET H=H(T)+H1(J) :
    GOTO 118
110: Q=Q+1 : IF J=S LET H(Q)=
    H(T)+H1(J) : GOTO 118
112:IF J=CLET Q=Q-1 : H=H(Q)
    +H1(J) : GOTO 118
114:H(Q)=H(Q-1)+H1(J)
118:NEXT J : IF ABS(H-H(U)>>
    1E-3 GOTO 150
120:NEXT I : MK=SQR(MK/(L+I
```

```
-M) )
122:FOR I=1TO M:LPRINT "H"
+CHR$ (96+I) + "=", INT
<H(K+I)*1E4+0.5)/1E4
123:NEXT I:LF 1:IF P=0GOTO
128
124:FOR I=1TO P:A$= "H (" +
STR$ I+ ") ="
126:LPRINT A$; INT (H (K+M+
I) *1E4+0.5) /1E4:NEXT I:
LF 1
128:FOR I=0TO N
130:LPRINT "Uh<" +STR$ (I+1)
+ ") =", INT (L(I) *1E4+
0.5)/10; "mm"
132:NEXT I:LF 1:LPRINT "MK
=", INT (MK*1E4+0.5) /10;
"mm":LF 1
140:FOR I=0TO M-1:PF(I)=MK
/JPF(1)
142:LPRINT "Mh" +CHR$ (97+I)+
"=", INT (PF(I)*1E4+0.5)
/10; "mm"
145:NEXT I:GOTO 200
150:WAIT 500:PRINT "ERROR"
200:END
```

文献出处：《设计资料》1986年第2期44—45页

文献名称 利用特征矩阵计算膜系反射率的 BASIC 程序

作　　者 吴泽耀

使用机型 紫金—Ⅱ、苹果机系统

使用语言 BASIC

用　　途 膜系反射率的计算

内　　容

一、说明：

本程序用来计算膜系的反射率，可适用于任意入射角和任意光学厚度的多层介质膜所组成的膜系，每输入一组数据，即可输出该膜系的光谱反射率。

二、计算公式：

已知膜系的光学参数 n_0，n_1，n_2 ……n_k，n_{k+1}，n_1d_1，n_2d_2……n_kn_k，φ_0，则任意一个波长λ的反射率计算步骤为：

1. 由公式 $n_0\sin\varphi_0 = n_1\sin\varphi_1 = \cdots = n_j\sin\varphi_j$ (j=1, 2……k+1) 计算φ_j*

2. 由公式 $\eta_{pj} = n_j/\cos\varphi_j$ 和 $\eta_{sj} = n_j\cos\varphi_j$ (j=0, 1, 2……k+1) 分别计算P分量和S分量的各有效折射率值。

3. 由公式 $\delta_j = \frac{2\pi}{\lambda} n_jd_j\cos\varphi_j$ (j=1, 2, ……K) 计算δ_j。

4. 对P分量，计算 K+1 个矩阵的连乘积

$$\begin{pmatrix} B \\ C \end{pmatrix} \prod_{j=1}^{K} \begin{pmatrix} \cos\delta_j & i\sin\delta_j/\eta_{pj} \\ i\eta_{pj}\sin\delta_j & \cos\delta_j \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \eta_{pk+1} \end{pmatrix} = \begin{pmatrix} a_3+ia_4 \\ a_1+ia_2 \end{pmatrix}$$

5. 将膜系的导纳Y简化为：

$$Y = \frac{C}{B} = \frac{a_1+ia_1}{a_3+ia_4}$$

此为试读，需要完整PDF请访问：www.ertongbook.com