

第六章 计 算 方 法

进行数值计算（即科学计算），除了要学习计算机语言和程序设计方法外，还要学习计算方法。所谓计算方法指的是用计算机进行计算时的近似计算公式。研究各种数学问题的近似计算公式，是计算数学的任务。计算数学所提供的各种近似计算的方法，称为某数学问题的数值计算方法，简称“计算方法”。

同一个数学问题，往往有一个或多个计算方法。当然它们是有优劣之分的，好的方法应该是误差小，而且占计算机时间少。

在用计算机解题方法中，广泛使用“迭代”这一方法，这是和用人工计算有所不同的。

考虑到不少读者未必专门学过“计算方法”这一门课程，我们在此介绍一些最基本、最常用的计算方法并给出程序。读者用到时可以考虑甚至照抄即可。有的计算机系统提供了“程序库”，即各种常用的计算方法的程序，可以直接调用。

一、方 程 求 根

【例 110】用“二分法”求方程 $x^3 - x - 1 = 0$ 在区间 $[1.3, 1.4]$ 内的一个实的单根，要求精确到小数点后第三位 ($\varepsilon = 0.001$)。

【解】“二分法”的基本思想是，假定 $f(x) = 0$ 在区间 (a, b) 内有一个实单根 x^* 。取 a 和 b 的中点 即 $x_0 = \frac{1}{2}(a + b)$ ，若 $f(x_0)$ 与 $f(a)$ 同号，则 x^* 必不在 (a, x_0) 区间内（而在 (x_0, b) 区间内）。反之，若 $f(x_0)$ 与 $f(a)$ 为异号，则 x^* 必在 (a, x_0) 区间内（参见图 6-1）。

假定 x^* 在 (a_0, x_0) 之间对它再继续进行二分，……直至求得 x_k 值满足精度要求 ($|x_k - x_{k-1}| \leq \varepsilon$) 为止。

程序框图如图 6-2 所示。

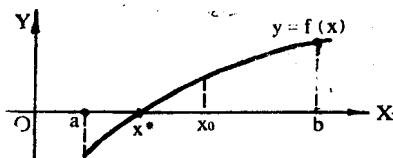


图 6-1

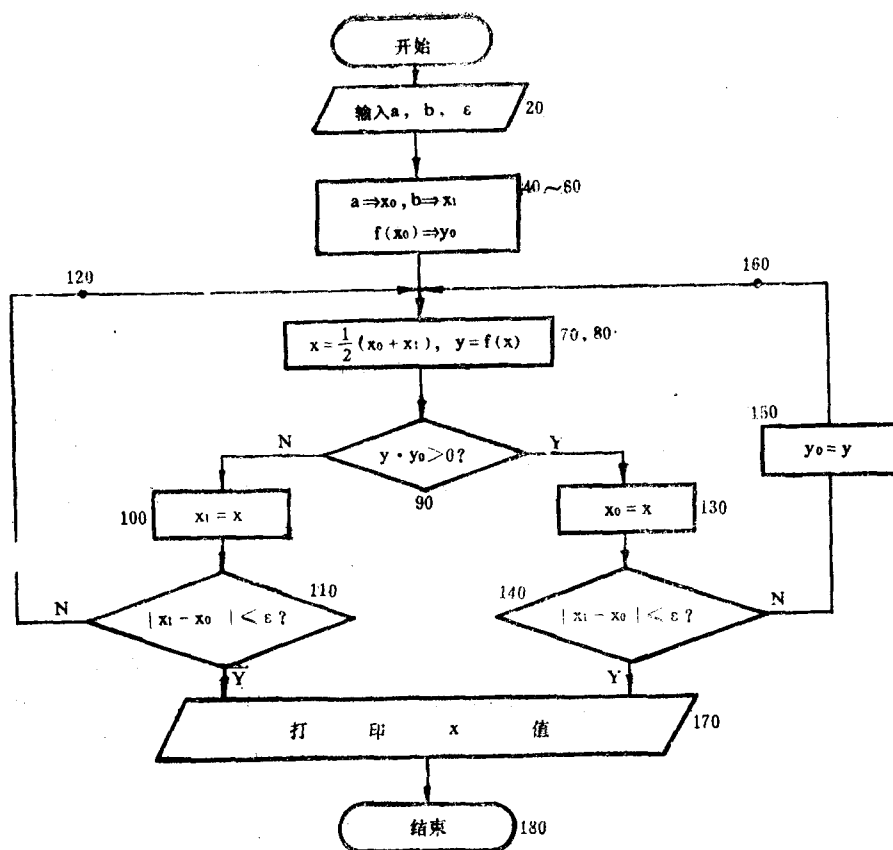


图 6-2

按框图可编写出如下程序;

```

10 PRINT "A,B,E =",
20 INPUT A,B,E
30 PRINT
40 X0 = A
50 X1 = B
60 Y0 = X0 ↑ 3 - X0 - 1
70 X = (X0 + X1) / 2
80 Y = X ↑ 3 - X - 1
85 IF Y · Y0 = 0 GOTO 170
90 IF Y · Y0 > 0 GOTO 130
100 X1 = X
110 IF ABS(X1 - X0) <= E GOTO 170
120 GOTO 70
130 X0 = X
140 IF ABS(X1 - X0) <= E GOTO 170
150 Y0 = Y
160 GOTO 70
  
```

```

170 PRINT "X=", X
180 END
RUN
A,B,E=? 1.3 , 1.4 , 1E-03?
X= 1.32422

```

【例 111】用“逐步扫描法”确定方程

$$x^3 - x - 1 = 0$$

的粗糙近似值——“初始近似值”。

【解】用计算机解题，求方程的根通常分二步进行。第一步用“逐步扫描法”确定方程的某个粗糙近似值——即初始近似值。第二步将初始近似值“加工”成满足精度要求的根。

设有方程 $f(x) = 0$ ，其根的分布情况可能是很复杂的。假如在 $[a, b]$ 区间内，方程仅有一个实的单根 x^* ，我们可令 $x = a$ ，并按某个预选步长 h (h 可正可负) 一步步往前跨 (h 为正，往右跨。 h 为负，往左跨)。若发现某步的起点 $f(x_0)$ 和终点 $f(x_0 + h)$ 之值为异号或零时 (即 $f(x_0) \cdot f(x_0 + h) \leq 0$)，则 x^* 必定在 $[x_0, x_0 + h]$ 区间内。于是可把这时候的 x_0 或 $x_0 + h$ 作为方程 $f(x) = 0$ 的初始近似值。

用“逐步扫描法”确定方法 $x^3 - x - 1 = 0$ 的初始近似值的 BASIC 程序如下：

```

10 PRINT "A,H=",
20 INPUT A,H
30 PRINT
40 X0=A
50 Y0=X0^3-X0-1
60 X0=X0+H
70 IF (X0^3-X0-1)*Y0>0 GOTO 50
80 PRINT "X0=", X0-H, "OR", "X0=", X0
90 END

```

在不知道方程初始近似值到底在什么位置的情况下，往往把扫描起点定在原点 (即令 $a = 0$)，然后从原点出发，先往 x 轴右边跨 (令 h 为正)，若一直找不到初始近似值，可令 $a = 0$ ， h 为负，往 x 轴反方向 (负向) 向左逐步扫描。

本题若令 $a = 0$ ， $h = 0.1$ (从键盘输入)，运行程序后可得如下结果：

```

RUN
A,H=? 0, 0.1?
X0= 1.3 OR X0=1.4

```

即方程 $x^3 - x - 1 = 0$ 的根在 $[1.3, 1.4]$ 区间内。于是可用更小的步长，更好的算法计算出方程的精确根。

【例 112】用简单迭代法求方程 $x = e^{-x}$ 的根。要求先判断其收敛性，结果精确到 $\epsilon = 0.001$ 。

【解】以 $h = 0.1$ ， $a = 0$ 代入上面例 111 题的程序，即可求出初始根为 0.5 或 0.6。

令 $x_0 = 0.5$ ，则：

$$|g'(x)| = |-e^{-x}|_{x=0.5} \approx 0.606 < 1$$

只要 x 为正, $|g'(x)|$ 总是 < 1 。

故 $x_{i+1} = g(x_i)$ 收敛。

程序如下：

```

10 PRINT "X0,E=";
20 INPUT X0,E
30 PRINT
40 X1=EXP(-X0)
50 IF ABS(X1-X0)<=E GOTO 80
60 X0=X1
70 GOTO 40
80 PRINT "X="; X1
90 END
RUN
X0,E=? .5, 1E-03
X= .566907

```

【例113】用“简单迭代法”求方程 $x^3 - x - 1 = 0$ 在 $x_0 = 1.3$ 附近的精确近似根 ($\epsilon = 0.00001$)。

【解】简单迭代法的基本思想是：

(1) 将方程 $f(x) = 0$ 化为 $x = g(x)$ 的形式；

(2) 用初始近似值 x_0 代入 $g(x)$ 求得 x_1 ；

(3) 若 $|x_1 - x_0| > \epsilon$ (ϵ 为要求达到的精确度)，则将 x_1 代入 $g(x)$ 求得 $x_2, x_3, \dots$ 。

如此重复直到当 $|x_{i+1} - x_i| \leq \epsilon$ 时，则迭代结束。那末 x_{i+1} 或 x_i 即是方程的精确根（严格地说应是较为精确的近似根）。

先将原方程化为 $x = g(x)$ 形式，即：

$$x = \sqrt[3]{x+1} \quad (\text{或 } x = (x+1)^{\frac{1}{3}})$$

程序如下：

```

10 PRINT "X0,E=";
20 INPUT X0,E
30 PRINT
40 X1=(X0+1)^(1/3)
50 IF ABS(X1-X0)<=E GOTO 80
60 X0=X1
70 GOTO 40
80 PRINT "X="; X1
90 END
RUN
X0,E? 1.3, 1E-05
X= 1.32472

```

简单迭代法求方程根的优点是算法逻辑结构简单，但其逼近速度慢，且仅适合于

$x_{k+1} = g(x_k)$ 收敛的情况, 若 $g(x_k)$ 是发散的话, 用此法将永远找不到根。例如本题中若有如下迭代公式:

$$x_{k+1} = x_k^3 - 1 \quad (x = x^3 - 1)。$$

当 x_0 仍取 1.3 时, 则有:

$$x_1 = 1.197, x_2 = 0.715, x_3 = -0.635, \dots$$

显然, 继续迭代下去已毫无价值, 因为 x_k 值将往 x 的负方向离精确近似根处越漂越远。

因此, 在使用迭代法前, 须先判断 $x_{k+1} = g(x_k)$ 是收敛还是发散。

判断 $x_{k+1} = g(x_k)$ 是发散还是收敛可用下面方法: 若 $g(x)$ 具有连续一阶导数, 且对所有 x , 若有 $|g'(x)| \leq q < 1$ (q 为定数), 那末 $x_{k+1} = g(x_k)$ 对于任意 x_0 均收敛, 且 q 越小, 其收敛速度越快。

【例 114】用“加速迭代法”求方程 $x = e^{-x}$ 在 $x_0 = 0.5$ 附近的一个根 ($\varepsilon = 0.000001$)。

【解】加速迭代法的基本思想是:

令 $x^* = \lim x_k$ 是满足方程 $x = g(x)$ 的根, x_{k+1} 为迭代 $k+1$ 次后所得的含有误差的实

根。利用微分中值定理, 则有 $|x^* - \bar{x}_{k+1}| \leq q |x^* - x_k|$ 。整理得

$$x^* - \bar{x}_{k+1} = \frac{q}{1-q} (\bar{x}_{k+1} - x_k), \text{ 这就是误差估计公式。}$$

倘若我们以上式等号右边的误差作为 $\bar{x}_{k+1}$ 的一种补偿, 则所得结果

$$x_{k+1} = \bar{x}_{k+1} + \frac{q}{1-q} (\bar{x}_{k+1} - x_k),$$

将是比 $\bar{x}_{k+1}$ 更接近 x^* 的一个好得多的结果, 于是就大大加速了迭代的进行。这就建立了一种所谓“迭代-加速公式”:

$$\begin{cases} \text{迭代: } \bar{x}_{k+1} = g(x_k) \\ \text{加速: } x_{k+1} = \bar{x}_{k+1} + \frac{q}{1-q} (\bar{x}_{k+1} - x_k) \end{cases}$$

由上题知, 方程 $x = e^{-x}$ 当 $x_0 = 0.5$ 时收敛, 且 $q = -0.6$ 。

用加速迭代法为本题编制的 BASIC 程序如下:

```

10 PRINT "X0,E,Q=",
20 INPUT X0,E,Q
30 PRINT
35 I = 0
40 X1 = EXP(-X0)
50 X1 = X1 + Q/(1-Q) * (X1 - X0)
52 I = I + 1
55 PRINT "X(", I, ")=", X1
60 IF ABS(X1 - X0) <= E GOTO 110
70 X0 = X1
100 GOTO 40
110 END
RUN

```

$$X_0, E, Q = 7.5, 1E-06, -0.62$$

$$X(1) = .566582$$

$$X(2) = .567132$$

$$X(3) = .567143$$

$$X(4) = .567143$$

简单迭代法解本题需迭代10次，得到的精度仅为 $\varepsilon = 10^{-8}$ 。使用加速迭代法解本题，只需迭代三次，且精度达到 $\varepsilon = 10^{-8}$ 。所以加速的效果是相当显著的。

在“迭代-加速公式”中，加速过程由于不需要访问迭代函数 g ，故其计算量可以忽略不计。可以看到增添了这种加速过程取得了显著的效果。这说明了研究算法的重要意义。在其它例题中我们也会碰到类似情况。

【例115】用牛顿迭代法求解方程 $x = e^{-x}$ 在 $x_0 = 0.5$ 附近的一个实根 ($\varepsilon = 0.000001$)。

【解】牛顿法是求解方程 $f(x) = 0$ 的一种重要方法。这种方法的基本思想是，设法将非线性方程 $f(x) = 0$ 逐步转化为某种线性方程求解。

设 $f(x) = 0$ 是要求解的方程， x_0 是一个初始近似根。于是函数 $f(x) = 0$ 在 x_0 附近可用泰勒公式表示为：

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots$$

为了避开计算高阶导数，同时也由于采用了迭代法，可不断校正误差值，因此可近似地取泰勒公式的前二项表示为：

$$f(x_0) + f'(x_0)(x - x_0) = 0。$$

这个近似方程是个线性方程。设 $f'(x_0) \neq 0$ ，解得：

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

若取上式之 x 作为新近似根 x_1 ，此时 $|x_1 - x_0| > \varepsilon$ ，则继续将 x_1 代替上式中的 x_0 ，求得 x_2 ，依此类推，直至 $|x_{k+1} - x_k| \leq \varepsilon$ ，则迭代结束， x_k 或 x_{k+1} 便是方程 $f(x) = 0$ 的根。这种迭代方法称为牛顿法。

牛顿迭代公式的一般形式是：

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

这个公式可改写成如下形式：

$$f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0,$$

可见迭代值 x_{k+1} 实际上是线性方程 $f(x_k) + f'(x_k)(x - x_k) = 0$ 的根。

牛顿法可用如图6-3的曲线表示。方程 $f(x) = 0$ 的根 x^* 在几何上表示曲线 $y = f(x)$ 与 x 轴的交点。设 x_k 是交点 x^* 的某个近似位置，过曲线 $y = f(x)$ 上的对应点 $P_k(x_k, f(x_k))$ 引切线，并将该切线与 x 轴的交点 x_{k+1} 作为根 x^* 的新的近似位置。注意到该切线的方程为

$$y = f(x_k) + f'(x_k)(x - x_k),$$

这样得到的交点 x_{k+1} 必满足方程

$$f(x_k) + f'(x_k)(x - x_k) = 0,$$

因而就是牛顿公式的计算结果。故牛顿法也称切线法。

牛顿迭代法具有平方收敛性。

$$\text{令 } f(x) = e^{-x} - x,$$

$$\therefore f'(x) = -e^x - 1.$$

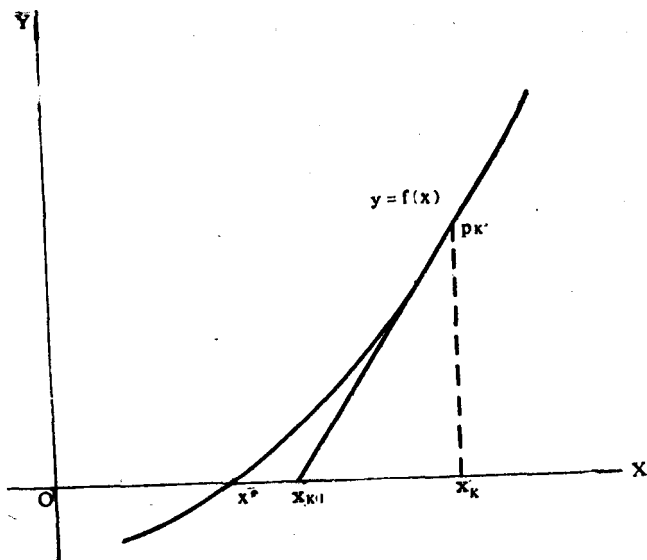


图 6-3

程序如下：

```

10 PRINT "X0,E =",
20 INPUT X0,E
30 PRINT
35 I=0
40 I=I+1
50 X1=X0+(EXP(-X0)-X0)/(EXP(-X0)+1)
60 PRINT "X(", I, ")=", X1
70 IF ABS(X1-X0)<=E GOTO 100
80 X0=X1
90 GOTO 40
100 END
RUN
X0,E=? .5 , 1E-06
X(1)= .566311
X(2)= .567143
X(3)= .567143
    
```

【例116】 用“快速弦截法”求方程 $e^{-x} - x = 0$ 的根。已知 $x_0 = 0.5$, $x_1 = 0.6$ 。要求精确到 0.000001。

【解】 牛顿法的突出优点是收敛速度快。但其明显缺点是需要事先计算 $f'(x)$ ，如果函

数 $f(x)$ 比较复杂 使用牛顿法反而显得不方便。为避开计算导数 我们以差商

$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}},$$

来代替牛顿公式中的导数 $f'(x_k)$ 。上式实际上是函数值增量与自变量增量之比, 即

$$\Delta f(x) / \Delta x。$$

也即是 $f(x)$ 在 (x_{k-1}, x_k) 之间的平均变化率(图 6-4), 故可近似地表示为 $f'(x)$ 之值。

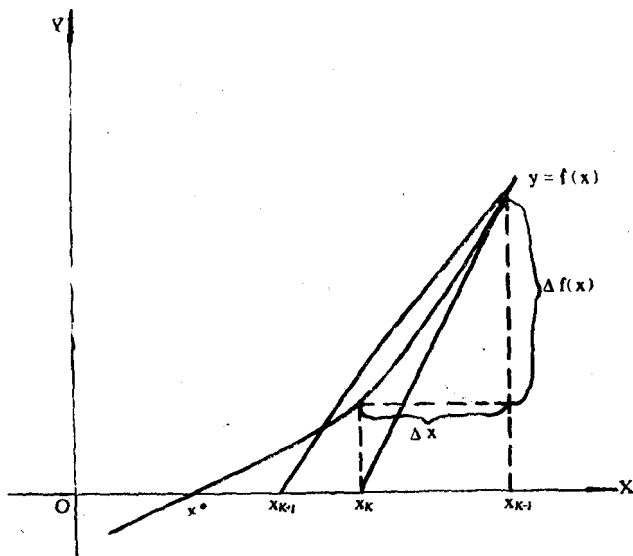


图 6-4

于是有如下快速弦截公式：

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{f(x_k) \cdot (x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}。$$

快速弦截法也是迭代法。不过, 我们前面提到过的迭代法有个共同点, 就是它们在计算 x_{k+1} 时只用到上一步的值 x_k , 这类迭代法称“一步迭代”。而快速弦截法在计算 x_{k+1} 时要用到前两步的结果 x_k, x_{k-1} 。因此快速弦截法是一种“多步迭代”。故在使用快速弦截公式时, 计算前须先给出两个初始值 x_0, x_1 。

程序如下：

```

10 PRINT "X0,X1,E=",
20 INPUT X0,X1,E
30 PRINT
40 I=0
50 I=I+1
60 F0=EXP(-X0)-X0
70 F1=EXP(-X1)-X1
80 X=X1-F1/(F1-F0)*(X1-X0)
90 PRINT "X(", I, ")=", X
100 IF ABS(X-X0)<=E GOTO 140

```

```

110 X0=X1
120 X1=X
130 GOTO 50
140 END
RUN
X0,X1,E=? .5, .6, 1E-06
X(1)=.567545
X(2)=.567141
X(3)=.567143
X(4)=.567143

```

二、计算 n 次多项式的值

【例 117】编写计算 n 次多项式 $P(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ ，在 x 处之值的 BASIC 通用程序。

【解】在计算机上计算多项式的值，通常将多项式 $P(x)$ 改写成如下形式：

$$\begin{aligned}
 P(x) &= a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n \\
 &= (\cdots((a_0x + a_1)x + a_2)x + \cdots + a_{n-1})x + a_n.
 \end{aligned}$$

此法称“秦九韶法”，是我国宋代数学家秦九韶最先提出的。外国文献中以霍纳（Horner）命名此法，但较之秦九韶却晚了五、六世纪。

秦九韶法的特点在于，它通过一次式的反复计算（乘、加），归结为重复计算 n 个一次式 $(a_{i-1}x + a_i)$ 来实现。这在计算机程序里可以十分方便的实现。

设 P 单元为存放一次式 “ $a_{i-1}x + a_i$ ” 的结果。程序如下：

```

10 PRINT "N,X=",
20 INPUT N,X
30 PRINT
40 READ P
50 FOR I=1 TO N
60 READ P1
70 P=P*X+P1
80 NEXT I
90 PRINT "P= ", P
100 END
110 DATA ..... (依次填入多项式各系数值)

```

现利用上面程序计算 6 次多项式 $P(x) = 8.3x^6 + 0.65x^5 - 2.3x^4 - 1.4x^3 + 4.7x^2 + 5.2x - 7.5$ 在 $x = 5.45$ 处之值。

先将 DATA 语句改写为：

```
110 DATA 8.3, 0.65, -2.3, -1.4, 4.7, 5.2, -7.5
```

将程序输入计算机后，运行程序，在键盘输入 6 和 5.45，即可打印出如下结果：

RUN

N,X=? 6 , 5.45 2

P= 218529

如果上例中 x 有 M 个已知点(设 $x=5.2, -7.5, 2, 5.4, 6.7, 7, 8.9$), 那末对应的多项式值可用如下的多个已知点通用程序来进行计算, 而无需分 M 次进行计算。

求多已知值 $x(m)$ 的 n 次多项式值的秦九韶解法的BASIC通用程序如下:

```
10 PRINT "N,M=",
20 INPUT N,M
30 PRINT
40 DIM P(N), X(M)
50 FOR I=0 TO N
60 READ P(I)
70 NEXT I
80 FOR J=0 TO M
90 READ X(J)
100 A=P(0)
110 FOR I=1 TO N
120 A=A * X(J) + P(I)
130 NEXT I
140 PRINT "X=", X(J), "P=", A
150 NEXT J
160 END
170 DATA <a0>, <a1>, ..., <an>
180 DATA <x1>, <x2>, ..., <xm>
```

现将DATA语句改写如下:

```
170 DATA 8.3, 0.65, -2.3, -1.4, 4.7, 5.2, -7.5
180 DATA 5.2, -7.5, 2, 5.4, 6.7, 7, 8.9
```

运行程序后, 人机回答记录如下:

RUN

N,M=? 6 , 7 2

X= 5.2 P= 6095.26

X= -7.5 P= 25873.3

X= 2 P= 130.7

X= 5.4 P= 7089.96

X= 6.7 P= 16813

X= 7 P= 20033.5

X= 8.9 P= 52344.3

三、计算数值积分

【例 118】试编写用梯形法计算定积分

$$y = \int_a^b f(x) dx$$

的通用 BASIC 子程序。

【解】在实际问题中，常需要用到数值积分。很多数学分支，如微分方程和积分方程的求解，也都是以积分计算为基础的。

梯形法积分的基本思想是：

如图 6-5，已知函数 $f(x)$ 在有限区间 (a, b) 上连续（分段连续也可以），则定积分

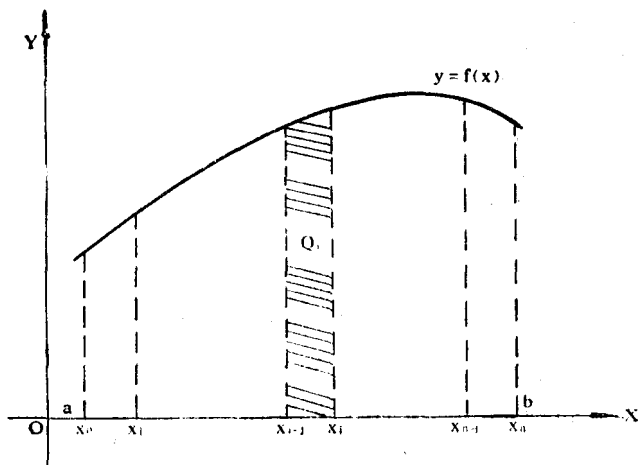


图 6-5

$$\int_a^b f(x) dx$$

存在。若在 (a, b) 内 $f(x) > 0$ （实际计算时可为任意，这里是为了便于分析问题而假设的），则积分值就是曲线 $f(x)$ 与直线 $x = a$ ， $x = b$ 及 x 轴所围成的曲边梯形的面积。

将 (a, b) 分成 n 等分，则积分值 y 应为所有小曲边梯形的面积之和 $(\sum Q_i)$ 。以小梯形面积近似代替小曲边梯形面积，则有：

$$T_i = (x_i - x_{i-1}) \cdot \frac{f(x_i) + f(x_{i-1})}{2} = \frac{b-a}{n} \cdot \frac{f(x_i) + f(x_{i-1})}{2},$$

$$\therefore y = \int_a^b f(x) dx \approx \sum_{i=1}^n T_i = \sum_{i=1}^n \left[\frac{b-a}{n} \cdot \frac{f(x_{i-1}) + f(x_i)}{2} \right].$$

若以 h 表示 $\frac{b-a}{n}$ （即每个小梯形的高），则有：

$$\int_a^b f(x) dx \approx h \sum_{i=1}^n \left[\frac{f(x_{i-1}) + f(x_i)}{2} \right].$$

为了不使计算函数 $f(x_i)$ 的工作重复，将上式展开后作如下演变和整理：

$$\begin{aligned} \int_a^b f(x) dx &= h \left[\frac{f(x_0) + f(x_1)}{2} + \frac{f(x_1) + f(x_2)}{2} \right. \\ &\quad \left. + \dots + \frac{f(x_{n-1}) + f(x_n)}{2} \right] \\ &= h \left[\frac{f(x_0) + f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i) \right]. \end{aligned}$$

这就是可在计算机上较为方便地进行计算定积分

$$y = \int_a^b f(x) dx$$

之值的梯形法求积公式。当上式的 n 值足够大时，积分值就与理论值相差无几了。

用梯形求积公式编写的通用子程序如下：

```

500 REM SUBROUTINE T-INTEGRATION
510 H = (B - A) / N
520 T = (FNF(A) + FNF(B)) / 2
530 FOR X = A + H TO B - H STEP H
540 T = T + FNF(X)
550 NEXT X
560 T = T * H
570 RETURN

```

现调用上面子程序，计算定积分

$$y = \int_0^1 e^{-x^2} dx \quad (\text{取 } n = 100)。$$

主程序可以这样编写：

```

10 REM MAIN PROGRAM FOR T-INTEGRATION
20 PRINT "A,B,N=",
30 INPUT A,B,N
40 PRINT
50 DEF FNF(X) = EXP(-X * X)
60 GOSUB 500
70 PRINT "THE INTEGRATION VALUE IS, ", T
80 END
RUN
A,B,N=? 0, 1, 100
THE INTEGRATION VALUE IS, .746818

```

【例119】试编写用辛普生 (Simpson) 求积公式计算定积分

$$y = \int_a^b f(x) dx$$

之值的通用BASIC子程序。并编写调用此通用子程序计算定积分

$$y = \int_0^1 \frac{\sin x}{x} dx$$

之值的BASIC主程序。

【解】利用二端点 a ， b 作结点所构成的一次多项式，是一条过 $(a, f(a))$ ， $(b, f(b))$ 的直线：

$$p_1(x) = \frac{x-b}{a-b} f(a) + \frac{x-a}{b-a} f(b)。$$

所以用二点公式所求得积分值即是梯形公式：

$$T = \frac{b-a}{2} [f(a) + f(b)]$$

所得的结果。

若除端点 a 和 b 外，再补充中点 $c = \frac{a+b}{2}$ 作为结点构成一个二次多项式

$$p_2(x) = \frac{(x-c)(x-b)}{(a-c)(a-b)} f(a) + \frac{(x-a)(x-b)}{(c-a)(c-b)} f(c) + \frac{(x-a)(x-c)}{(b-a)(b-c)} f(b),$$

它应该是过三点 $(a, f(a))$, $(c, f(c))$, $(b, f(b))$ 的抛物线。这就得到如下三点公式：

$$S = \frac{b-a}{6} [f(a) + 4f(c) + f(b)].$$

此式称为辛普生公式。

由于抛物线 $y = p_2(x)$ 比直线 $y = p_1(x)$ 更接近原函数曲线 $(y = f(x))$ ，因此用辛普生公式求得的积分值比用梯形求积公式求得的积分值，更接近原函数的积分值。

若将 (a, b) 分成 n 等分，对子区间 (x_{i-1}, x_i) ($i = 1, 2, \dots, n$)，则有：

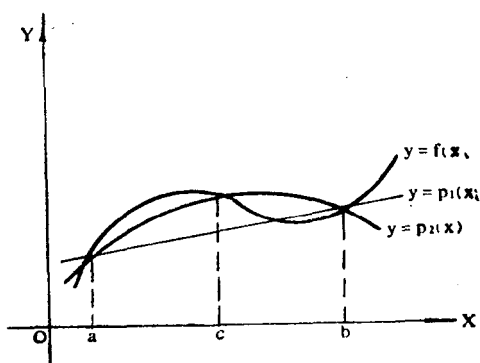


图 6-6

$$S_i = \frac{h}{6} [f(x_{i-1}) + 4f(x_{i-\frac{1}{2}}) + f(x_i)],$$

$$\begin{aligned} \therefore S &= \int_a^b f(x) dx \approx \sum_{i=1}^n S_i = \frac{h}{6} \sum_{i=1}^n [f(x_{i-1}) + 4f(x_{i-\frac{1}{2}}) + f(x_i)] \\ &= \frac{h}{6} \{ [f(x_0) + 4f(x_{1-\frac{1}{2}}) + f(x_1)] \\ &\quad + [f(x_1) + 4f(x_{2-\frac{1}{2}}) + f(x_2)] + \dots \\ &\quad + [f(x_{n-1}) + 4f(x_{n-\frac{1}{2}}) + f(x_n)] \} \\ &= \frac{h}{6} \{ [f(x_0) - f(x_n)] + 2 \sum_{i=1}^n [2f(x_{i-\frac{1}{2}}) + f(x_i)] \} \\ &= \frac{h}{3} \left\{ \frac{f(a) - f(b)}{2} + \sum_{i=1}^n [2f(x_{i-\frac{1}{2}}) + f(x_i)] \right\}. \end{aligned}$$

用辛普生求积公式编写计算定积分

$$y = \int_a^b f(x) dx$$

之值的通用 BASIC 子程序如下 其中 $h = (b-a)/n$ ：

```
1000 REM SUBROUTINE FOR SIMPSON-INTEGRATION
1010 H = (B - A) / N
1020 S = (FNF(A) - FNF(B)) / 2
```

```

1030 FOR X=A+H TO B STEP H
1040 S=S+2*FNF(X-H/2)+FNF(X)
1050 NEXT X
1060 S=S*H/3
1070 RETURN

```

现调用以上子程序，计算定积分

$$y = \int_0^1 \frac{\sin x}{x} dx$$

主程序如下：

```

10 REM MAIN PROGRAM FOR SIMPSON-INTEGRATION
20 PRINT "A,B,N=",
30 INPUT A,B,N
40 PRINT
50 DEF FNF(X)=SIN(X)/X
60 GOSUB 1000
70 PRINT "THE INTEGRAL VALUE IS: ", S
80 END

```

运行程序并配合键盘输入，可打印出如下结果：

```

A,B,N=? 0, 1, 100
THE INTEGRAL VALUE IS, .946074

```

【例 120】试编写用柯特斯 (Cotes) 求积分公式计算定积分

$$y = \int_a^b f(x) dx$$

值的 BASIC 通用程序。

【解】如果在三点公式 (辛普生) 的端点 a 、 b 及中点 c 之间再各增加一中点

$$d = a + \frac{b-a}{4}, \quad e = a + \frac{3}{4}(b-a),$$

也不难推导出下列五点公式：

$$c = \frac{b-a}{90} [7f(a) + 32f(d) + 12f(c) + 32f(e) + 7f(b)].$$

此即柯特斯求积公式。

将 (a, b) 分成 n 等分，对每个等分段再应用上面求积公式，便可得到精度较高的总积分值。

对于区间 (x_{i-1}, x_i) ($i=1, 2, \dots, n$) 有：

$$C_i = \frac{b-a}{90n} [7f(x_{i-1}) + 32f(x_{i-\frac{3}{4}}) + 12f(x_{i-\frac{1}{2}}) + 32f(x_{i-\frac{1}{4}}) + 7f(x_i)],$$

$$\therefore \int_a^b f(x) dx \approx \sum_{i=1}^n C_i = \frac{h}{90} \sum_{i=1}^n [7f(x_{i-1}) + 32f(x_{i-\frac{3}{4}}) + 12f(x_{i-\frac{1}{2}}) + 32f(x_{i-\frac{1}{4}}) + 7f(x_i)].$$

为了不使计算量重复，也为了便于编程序，我们可将上式展开如下：

$$\begin{aligned} \int_a^b f(x) dx &\approx \frac{h}{90} \left\{ [7f(x_0) + 32f(x_{1-\frac{3}{4}}) + 12f(x_{1-\frac{1}{2}}) \right. \\ &\quad + 32f(x_{1-\frac{1}{4}}) + 7f(x_1)] + [7f(x_1) \\ &\quad + 32f(x_{2-\frac{3}{4}}) + 12f(x_{2-\frac{1}{2}}) + 32f(x_{2-\frac{1}{4}}) \\ &\quad + 7f(x_2)] + \cdots + [7f(x_{n-1}) + 32f(x_{n-\frac{3}{4}}) \\ &\quad + 12f(x_{n-\frac{1}{2}}) + 32f(x_{n-\frac{1}{4}}) + 7f(x_n)] \left. \right\} \\ &= \frac{h}{90} \left\{ 7[f(x_0) - f(x_n)] + \sum_{i=1}^n [32f(x_{i-\frac{3}{4}}) \right. \\ &\quad + 12f(x_{i-\frac{1}{2}}) + 32f(x_{i-\frac{1}{4}}) + 14f(x_i)] \left. \right\} \end{aligned}$$

用柯特斯公式编写的求积通用子程序如下：

```

1000 REM SUBROUTINE FOR COTES-INTEGRATION
1010 H = (B - A) / N
1020 C = 7 * (FNF(A) - FNF(B))
1030 FOR X = A + H TO B STEP H
1040 C = C + 32 * FNF(X - H * 0.75) + 12 * FNF(X - H/2) + 32 * FNF(X - H/4) + 14 *
      FNF(X)
1050 NEXT X
1060 C = C * H/90
1070 RETURN

```

利用柯特斯求积公式编写的主程序与辛普生主程序相同。

【例121】用变步长的梯形求积分公式计算积分

$$y = \int_0^1 \frac{\sin x}{x} dx$$

之值。要求精确到 $\epsilon = 10^{-5}$ 。本题要求先画框图，后根据框图写出程序。

【解】前面介绍的梯形法求积公式、辛普生求积公式和柯特斯求积公式，对积分精度仅取决于步长（即取决于等分数 n 。 n 越大，则 h 越小，精度越高）。但步长太小将会导致计算量的增加，步长太大又难以保证精度，而事先要确定一个恰当的步长却是很难做到的。

实际计算中往往采用变步长的求积方案。即在步长逐次折半的过程中反复利用求积公式计算积分值。直至前后二次积分值之差的绝对值满足精度要求为止。

由前知，应用梯形法公式对每个子区间 (x_{i-1}, x_i) 上的积分值有：

$$T_i = \frac{h}{2} [f(x_{i-1}) + f(x_i)].$$

经过二分（步长折半）再增加一个新分点 $x_{i-\frac{1}{2}}$ 后，对子区间 (x_{i-1}, x_i) ，其积分值为：

$$\begin{aligned} T_i &= \frac{h}{4} [f(x_{i-1}) + f(x_{i-\frac{1}{2}})] + \frac{h}{4} [f(x_{i-\frac{1}{2}}) + f(x_i)] \\ &= \frac{h}{4} [f(x_{i-1}) + 2f(x_{i-\frac{1}{2}}) + f(x_i)] \end{aligned}$$

$$\therefore \int_a^b f(x) dx \approx \frac{h}{4} \sum_{i=1}^n [f(x_{i-1}) + f(x_i)] + \frac{h}{2} \sum_{i=1}^n f(x_{i-\frac{1}{2}})$$

$$\text{令 } \frac{h}{2} \sum_{i=1}^n [f(x_{i-1}) + f(x_i)] = T_{1.}$$

$$\text{则 } \int_a^b f(x) dx \approx \frac{T_{1.}}{2} + \frac{h}{2} \sum_{i=1}^n f(x_{i-\frac{1}{2}}) \dots \text{令为 } T_{2.}$$

我们可看出，这里的 $T_{1.}$ 和 $T_{2.}$ 实际上分别代表了二分前后的积分值。由于 $T_{1.}$ 中所计算的是老结点值（二分前之积分值）而 $T_{2.}$ 中又包含有 $T_{1.}$ ，因此在计算 $T_{2.}$ 时只要调用 $T_{1.}$ 并计算新结点 $x_{i-\frac{1}{2}}$ 的值（二分后增加的）即可。可见变步长梯形公式（也称递推公式）由于避免了老结点的重复计算，从而使总计算量在步长相同的情况下（指变步长中的最后步长）与应用定步长梯形法求积分值所需的计算量几乎一样。然而它的优点远不止于此，它能以 2^n (n 为二分次数) 的收敛速度自动找到最佳步长，从而求出既能满足精度要求，又使步长最小的积分值。

用变步长梯形公式（递推公式）求定积分

$$y = \int_0^1 \frac{\sin x}{x} dx$$

值的程序框图如图6-7所示。

程序如下：

```

10 PRINT "A, B, E =",
20 INPUT A, B, E
30 PRINT
40 H = B - A
50 T1 = H/2 * (SIN(A)/A + SIN(B)/B)
60 S = 0
70 X = A + H/2
80 S = S + SIN(X)/X
90 X = X + H
100 IF X < B GOTO 80
110 T2 = T1/2 + S * H/2
120 IF ABS(T2 - T1) <= E GOTO 160
130 T1 = T2
140 H = H/2
150 GOTO 60
160 PRINT "Y =", T2
170 END
RUN
A, B, E = ? 1E-38, 1, .00001
Y = .946079

```

【注】 键盘输入的 A 值应为机器零 ($1E-38$)，否则 $\text{SIN}(A)/A$ 为无定义。

【例 122】编写用龙贝 (Romberg) 方法计算积分

$$y = \int_0^1 \frac{\sin x}{x} dx$$

值的 BASIC 程序 ($\varepsilon = 10^{-8}$)。

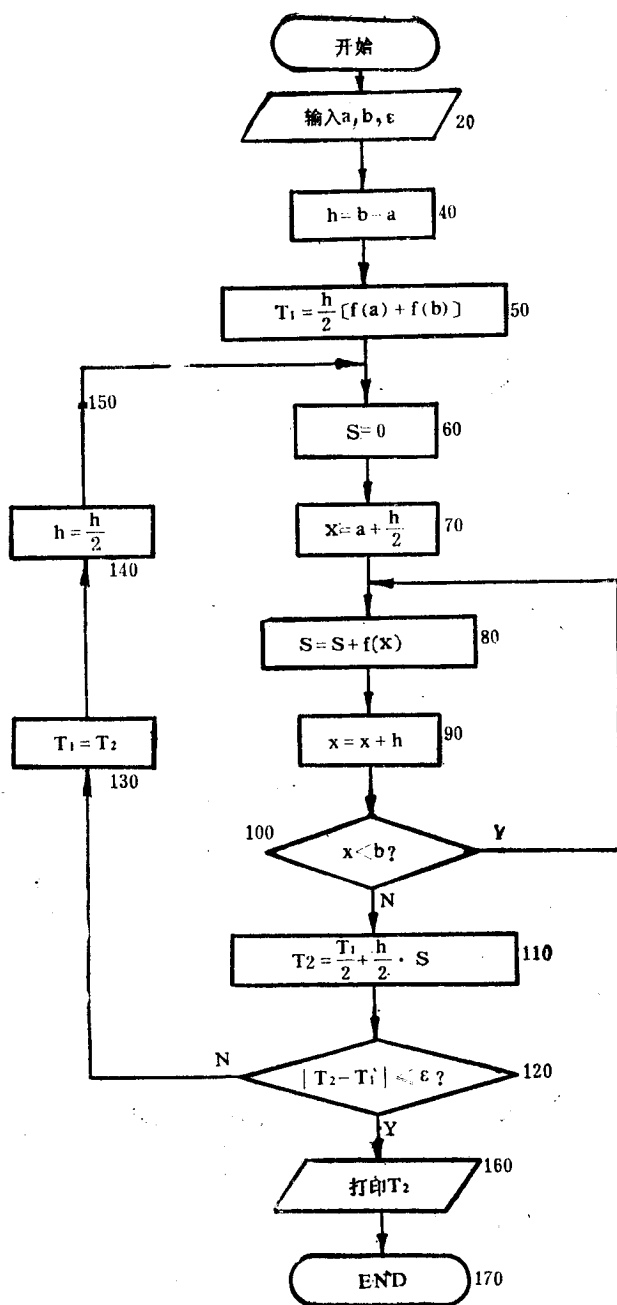


图 6-7

【解】 梯形法的截断误差是与 h^2 成正比的，故当步长折半后，截断误差将减至 $\frac{1}{4}$ 。

$$\frac{I^* - T_{2n}}{I^* - T_{1n}} \approx \frac{1}{4}$$

其中 I^* ——精确积分值，
 T_{1n} ——二分前积分值，
 T_{2n} ——二分后积分值。