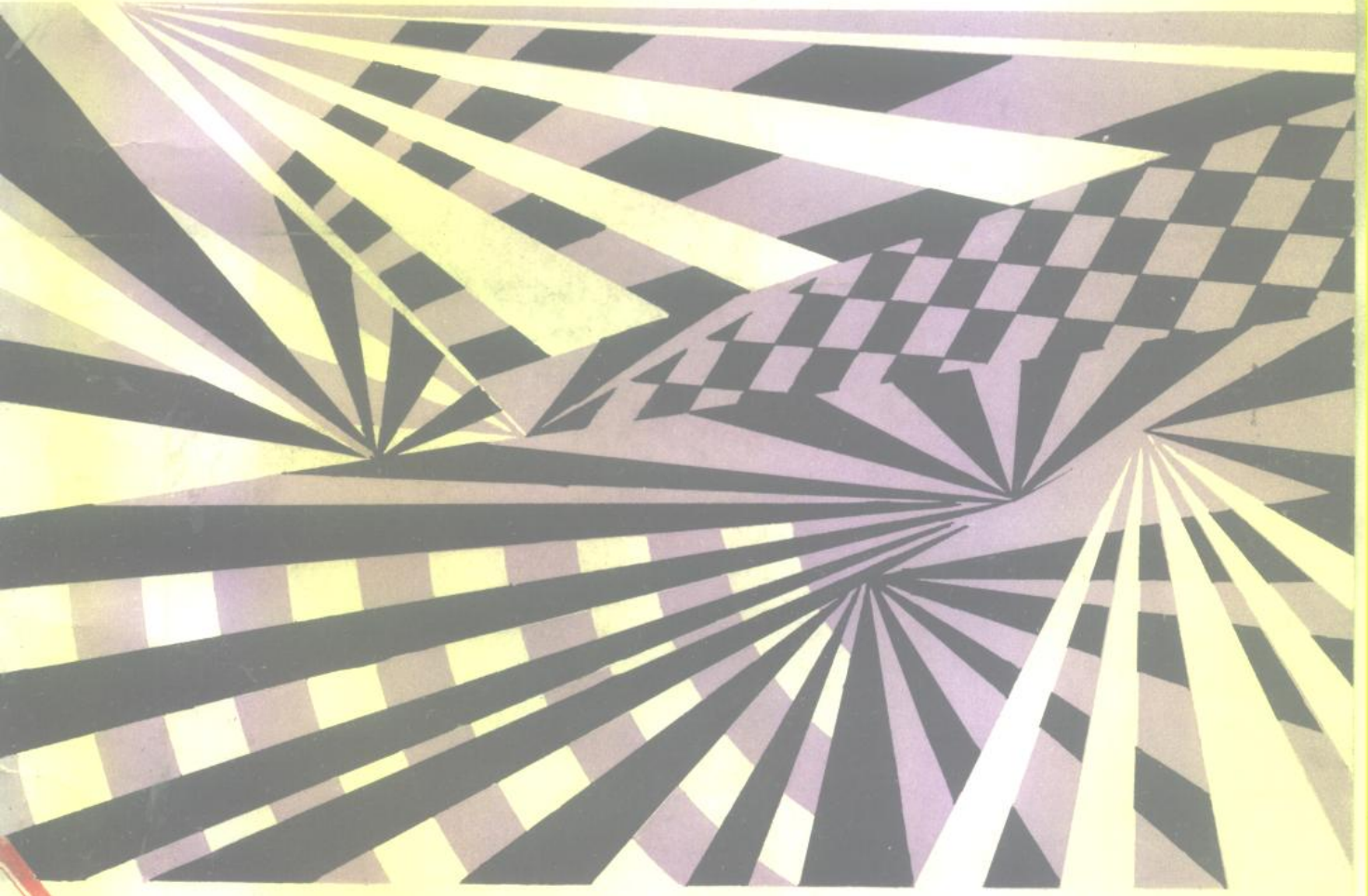
计算机算法与应用

陈礼义　编



天津大学出版社

计算机算法与应用

陈礼义 编

天津大学出版社

内 容 简 介

本书着重讲述在数字计算机上进行科学计算常用数值计算方法。内容包括：算法与误差；非线性方程解法；线性代数方程解法；矩阵特征值算法；函数插值与曲线拟合；数值积分；常微分方程数值解法和最优化方法，书中给出用FORTRAN算法语言编写的程序实例和算法子程序，供读者在实际工程数值计算中参考。

本书可做为高等学校非计算数学专业选修课教材，也可作为研究生、工程技术人员的参考书。

计算机算法与应用

陈秋义

天津大学出版社出版
（天津大学内）
河北省邮电印刷厂印刷
新华书店天津发行所发行

开本：787×1092毫米1/16　印张：123/4字数：318千字
1990年12月第一版　1990年12月第一次印刷
印数：1—2500
ISBN7-5618-0226-9
TP·28
定价：2.60元

序　言

数字计算机的广泛应用使得许多复杂、繁琐的工程技术计算问题得到解决，提高了工作效率。因此，在工科院校的课程设置中，数值计算方法已成为一门基本训练课程。工科院校学生和工程技术人员不仅需要对数值计算方法的原理有系统了解，而且更需要掌握使用这些原理解决实际计算的技能。为了提高这种能力需要有一些使用效果好、程序技巧较高的算法程序实例供他们在学习和实际工作中参考。本书就是按照这种要求编写的。

作者从1981年开始在电力及自动化系各专业讲授本课程。根据专业课程设置的需要，吸收专业教师、学生的意见，参考国内外有关教材，编写了这本讲义，经过多次使用修改，最后形成本书的结构。书中各章后面的习题供学生用笔算了解和加深对算法和计算过程的认识。计算程序实例和计算程序练习题供学生在计算机上进行算法和程序设计练习。计算程序实例中的算法子程序可供学生在毕业设计、科研论文及工作中参考，对工程技术人员也有一定参考价值。

本课程应在修完高等数学和算法语言（FORTRAN）后开设。讲授时数为40-50学时，可根据专业课程需要和学时数适当选择章节。在计算机上的练习，一般不少于10小时。练习题数量和内容可按需要选择。在课程讲授中应注意结合专业特点对算法的适用场合、计算工作量、内存占有量、误差特性、收敛性、数值稳定性等方面进行分析、比较和综合。计算机上练习应对学生进行算法、算法语言和程序设计能力进行综合训练和培养。

本书编写过程中，得到贺家李教授的许多帮助和鼓励。宋文南教授详细审阅书稿，提出许多宝贵意见。余贻鑫教授和其他教师、学生对本书提出许多建议。戴积成副教授和康庆平同志对本书的出版给予有力支持。对于他们的热情帮助作者表示感谢。由于本人水平有限，书中的错误和不妥之处在所难免，希望读者批评指正。

编者

一九九〇年一月　于天津

目　录

第一章　算法与误差

现代科学的各个领域几乎都离不开利用数学来描述它的概念和理论，描述它的运动和各变量的依赖关系，其中数值计算起着重要作用。离开计算，一切科学研究和工程设计都是不可能的。掌握数学工具和计算技术是掌握各门科学的关键。

现代数值分析研究应用数字计算机对各种数学问题进行数值计算的方法和理论。它具有两个显著特点：一是数值计算方法和理论都结合数字计算机的特点来研究，注意算法与计算速度、计算机内存消耗的关系；二是在研究算法时要进行误差分析，能估计误差的算法才是有实用价值的算法，而且注意协调计算量和误差的关系，注意解的收敛性与数值稳定性问题。

§1.1　算　　法

一　算法的概念

任何一项计算，总要事先拟定计算方案和规划计算步骤。在解题过程中，常常要处理各种不同条件的限制和变化并随时修改和补充预定的计算方案。

这里所说的算法针对数字计算机而言。由于数字计算机运算速度很高，是人工手算所不能比拟的。为了充分发挥计算机的这种优点，在用计算机解题过程中，应尽量减少人工干预，实现操作自动化。因此在上机解题之前，必须先将所制定的解题方案“告诉”计算机，使计算机按照人们规定的计算顺序去自动执行。用计算机能接受的“语言”来描述解题步骤，这项工作叫做程序设计。

为使计算机的计算过程能够完成某种要求，必须确定交付计算机执行的解题方案中的每个详细步骤，并且将此过程完整地描述出来。所谓算法是对某个数学问题求解方案的完整而明确的描述。这里指的算法仅限于数值计算的算法。

描述算法可以用不同方式，例如，可用通常用的语言和数学公式加以叙述，亦可借助算法语言给出精确的说明；还可用流程图直观地表示算法的整个结构。下面举个简单的例子来说明算法。

设要求解二元一次联立方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2=b_1 \\ a_{21}x_1+a_{22}x_2=b_2 \end{cases}$$

方程组的行列式解法可表述如下：

首先判别

$$d=a_{11}a_{22}-a_{21}a_{12}$$

是否等于零，存在两种可能

（1）如果$d\neq 0$，则令计算机计算

$$x_1=(b_1a_{22}-b_2a_{12})/d$$

$$x_2=(b_2a_{11}-b_1a_{21})/d$$

然后输出计算的结果x_1，x_2。

（2）如果$d=0$，则或是无解，或有无穷多组解，这是奇异的情形。

流程图（或称框图）如图1-1所示，它形象地描述上述算法。

从上例看出流程图是对算法精确而完整的描述，也是编写计算程序的依据。一些比较复杂的计算过程，常常需要先绘制流程图再用算法语言写出计算程序。

流程图的符号可分为两大类：

（1）流线

流程图中用流线表示程序的走向，箭头的方向表示程序执行的方向。

（2）框图

采用不同的几何图形表示不同的程序功能，它可分为以下几种：

a. 输入输出框。它表示数据输入和计算结果打印输出，其图形如图1-2所示。

b. 处理框。它表示程序的某种基本操作功能，一般用矩形框表示，框内用文字和符号表示实现的操作功能，如图1-3所示。

c. 判断框。根据程序执行的条件判断程序执行的走向。一般用菱形框或半圆框表示，如图1-4所示。

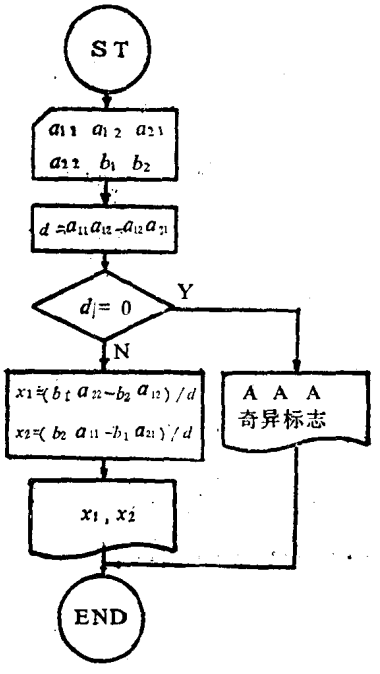


图 1-1 流程图

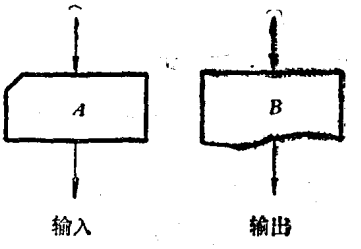


图 1-2 输入、输出框

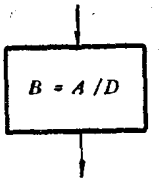


图 1-3处理框

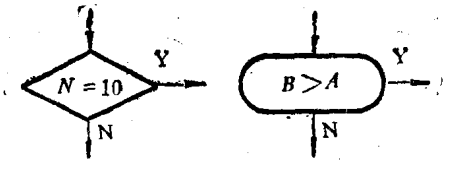


图 1-4 判断框

d. 端框、连接框。用来表示程序的起始、终止和连接。框图内填上“ST”，“END”或数字来表示起始、终止和连接，如图1-5所示。

此外，还有其他框图，如循环框、转子程序框，其图形如图1-6所示。

二 算法的质量标准

计算机运算速度高，存贮数据容量大，并能自动完成极其繁复的计算过程。计算机的解算功能虽然很强，但是否可降低对算法的要求呢？许多事实说明，如果算法选择不当，计算机的利用率就得不到充分发挥，有时甚至不能得到满意的解答。例如求解线性方程组，原则

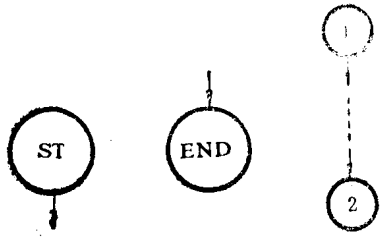


图 1-5 端框，连接框

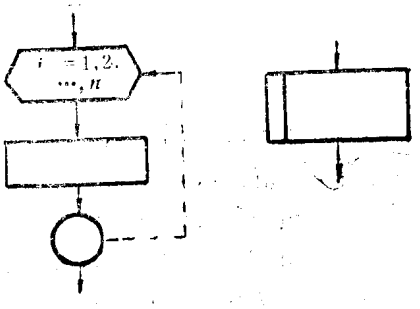


图 1-6 循环框，转子程序框

上可以用行列式解法。如解 n 阶方程组，要计算 $n+1$ 个 n 阶行列式的值，总共需要做 $n!(n-1)(n+1)$ 次乘法运算。设 n 为20，采用每秒一千万次的计算机，要连续计算三千万年才能完成。当然这是没有实际意义的。因此，计算量的大小是衡量算法优劣的一项重要标准。

计算程序所占用的工作单元的数量称为算法的内存消耗，或占用内存量。尽管计算机能贮存大量信息。但计算大型算题时，有些微型计算机仍不能使用。因此，尽量节约存贮量有经济价值，是衡量算法质量的又一标准。

设计算法时应考虑的另一个因素是逻辑结构问题。虽然计算机能够自动执行极其复杂的计算方案，但是计算方案的每个细节都需要编程人员制定。因此算法的逻辑结构应尽量简单，才能使编制程序、维修程序和使用程序时比较方便。

由此可见，虽然数字计算机是一种强有力的计算工具，但不能因此忽视算法的研究。应该以计算量大小、存贮量多少、逻辑结构是否简单作为评定算法优劣的标准。

三 算法的递推性

计算机上使用的算法常采取递推化的形式。递推化的基本思想是把一个复杂的计算过程归结为简单过程的多次重复。这种重复在程序上表现为循环。递推化的优点是简化程序结构和节省计算量。

下面用多项式求值问题说明递推化方法。

设对于给定的 x 求下列 n 次多项式的值。

$$p(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n \tag{1-1}$$

为了便于叙述，将多项式按降幂的次序排列为

$$p(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0 \tag{1-2}$$

设从前 n 项提出 x，则有

$$p(x)=(a_nx^{n-1}+a_{n-1}x^{n-2}+\cdots+a_2x+a_1)x+a_0$$

经过变换，括号内是 $n-1$ 次多项式，如果对括号内的多项式做同样的变换，又有 $p(x)=((a_nx^{n-2}+a_{n-1}x^{n-3}+\cdots+a_2)x+a_1)x+a_0$ 这样每做一步，最内层的多项式降低一次，最终可将多项式(1-2)变换成如下嵌套的形式。

$$p(x)=(\cdots((a_nx+a_{n-1})x+a_{n-2})x+\cdots+a_1)x+a_0 \tag{1-3}$$

利用(1-3)式结构上的特点，由里向外，一层一层地计算。设用 v_k 表示第 K 层（从里向外数）的值，因此最里面一层为

$$v_1=v_0x+a_{n-1}$$

令初值

$$v_0=a_n$$

第二层为

$$v_2=v_1x+a_{n-2}$$

第K层为

$$v_k=v_{k-1}x+a_{n-k}\qquad k=1,\ 2,\ \cdots,\ n$$

将v_{k-1}，v_{k-2}，$\cdots v_0$值代入得

$$v_k=(\cdots((a_nx+a_{n-1})x+\cdots+a_{n-k+1})x+a_{n-k}$$

以上过程进行到第n层时，即当$K=n$时计算结束。

由以上例子可以看出，这种递推算法不但逻辑结构简单而且由于反复使用一个存储单元存放v，从而节省了内存。

§1.2 误　差

数值分析或数值计算必然存在误差，因此给人以不严格，不准确的错觉。其实计算结果的近似性是正常的不可避免的，绝对的严格和精确是不存在的。当然这种近似性应该是合理的、为研究的问题所容许，否则计算结果便没有意义。因此研究算法的同时也要分析误差。

一　误差的来源

数值计算的误差来源是多方面的。首先是建立数学模型时的误差。为了进行数值计算，首先必须将实际问题用一定的数学表达式描述，即建立合适的数学模型。

在建立数学模型时，通常要根据实际要求做一些简化，忽略一些次要因素，使数值计算不致过分复杂而计算结果又能满足要求。这样建立的“理想化”的数学模型是客观现象近似的描述，这种近似必然产生误差。显然这种误差是正常的，不可避免的。

其次，在解题过程中，也不可避免会产生各种误差。许多数学运算（例如无穷级数求和、微分、积分等）是通过极限过程来定义的，但是在进行数值计算时只能完成有限项算术运算及逻辑运算，因此需将解题方案转化为算术运算与逻辑运算的有限序列。这种加工常表现为某程无穷过程的“截断”，因此产生的误差通常称为“截断误差”。

例如，指数函数e^x可展开成下面幂级数形式

$$e^x=1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}+\cdots \tag{1-4}$$

在实际计算时无法得到其右端无穷多项和，只能截取有限项求和

$$S_n(x)=1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!} \tag{1-5}$$

以部分有限项之和$S_n(x)$作为e^x的值，必然存在误差。根据泰勒（Taylor）余项定理，$S_n(x)$关于e^x的截断误差为

$$R_n(x)=\frac{x^{n+1}}{(n+1)!}e^{\theta x},\ 0<\theta<1 \tag{1-6}$$

第三，由于数字计算机上只能用有限的数位来表示一个数，对于计算过程中超出限定位数的尾数将在运算中舍去，从而造成误差，这种误差称为舍入误差。在两个相近的数相减，或者两个数值大小相差很大的数相除以后再乘一很大的数时，舍入误差相当大。因此在计算时应该尽量避免这种情况发生。

上述情况说明，在数值计算过程中会出现各种误差。因此，应当在满足精度要求的前提下选择和设计合适的算法。

二 误差限和有效数字

误差限和有效数字用来定量表示误差的大小，并且在计算程序中控制误差。误差限的定义为

设 Z 是准确值 Z^* 的某个近似值，如果能够估计出误差 Z^*-Z 的取值范围，即指定一个正小数 ε 使得

$$|Z^*-Z|<\varepsilon \tag{1-7}$$

则称这样的 ε 为近似值 Z 的误差限。如果(1-7)式成立，那么，在有允许误差 ε 的情况下，近似值 Z 和真值 Z^* 可以看成是“重合”的，或者说，在有允许误差 ε 的情况下，结果 Z 是“准确”的。

有效数字用来说明近似值的准确程度。例如对于 π 的值，$Z^*=\pi=3.141592654\cdots$，取近似值 $Z_1=3.14$，它们之间的差为 $0.001592654\cdots$，Z_1 数的意义是个位数为 3，十分位数为 1，百分位为 4，千分位以后的数为零，因此它只有 3 位数准确，称有 3 位有效数字。另外，近似值 $Z_2=3.1416$，其误差为 $-0.000007346\cdots$，误差的绝对值小于 0.00001，在小数点后第四位的半个单位以内，即 $0.00001<0.00005$，而且从这一位到 Z_2 的第一位非零数字共有 5 位，因此 Z_2 有 5 位有效数字。然而对于近似值 $Z_3=3.1415$，其误差为 $0.000092654\cdots$，其绝对值小于 0.0001 即小于小数点后第 3 位的半个单位，故 Z_3 仅有 4 位有效数字。这就是常用 3.1416 而不用 3.1415 作为 π 的近似值的原因。

由以上例子可见，如果近似值 Z 的误差限是某一位上的半个单位，而且从该位直到 Z 的第一位非零数字共有 n 位，则近似值 Z 便有 n 位有效数字。

在理解有效数字的意义时，应该明确，有效数字是用来表示数的精度大小，而不是用来指示小数点的位置。

有效数字和误差限有对应的关系，两者可以互相转换。例如 $Z_2=3.1416$ 与 π 的误差小于小数点后第 4 位的半个单位，即小于 0.00005，这时

$$|Z_2-\pi|\leqslant 0.00005$$

所以误差限为 0.00005。误差限常用于计算程序中，检查计算结果的误差，控制计算过程。读者可以考虑一下，如果用试探法求 $x^2-2=0$ 的近似解，要求有 4 位有效数字的精度，试问误差限是多少。

第二章　非线性方程求解

§2.1 引　言

在科学计算中时常遇到求解函数方程$f(x)=0$的数值计算问题。$f(x)$可能是超越函数，例如含有三角函数或者对数函数等特殊函数，也可以是多项式，这时$f(x)=0$为代数方程。非线性方程的解法可分为直接求解法和间接求解法。直接求解法可以用一组公式求得一个解，这个解一定是精确解。例如用二次方程的求根公式求解就是一种直接求解法。间接求解法是重复进行某种数值计算过程，最后得到一个近似解，解的误差可以估计和控制。例如下面方程

$$a\sin^2 x+bx+c=0$$

因为没有一个现成的能表达其解的代数式，故只能寻找近似计算办法，这就是所谓的数值计算方法。

数值计算方法的内容十分广泛，不仅仅限于上述例子中的方程求解。本书后面涉及的各种数学方程的算法大多数属于数值计算方法。数值计算方法通常有以下特点：

（1）数值计算一般来说只能得到近似解，如果在计算方法或者计算过程上做一些考虑，这种近似可以改善，得出更精确的结果。

（2）数值计算的概念比较简单，不需引入复杂的数学知识。

（3）数值计算方法是针对数字计算机的，计算过程适合在数字计算机上执行。

（4）数值计算有时不能得出结果。这个问题将在后面各种数值计算问题中具体讨论。

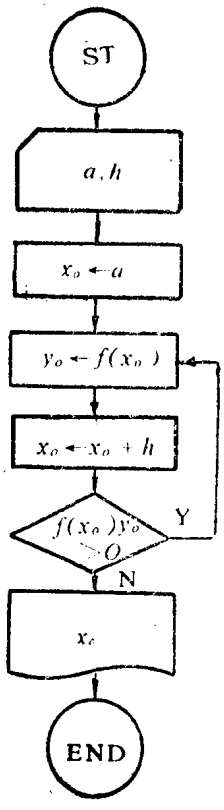


图 2-1　扫描法流程图

下面就函数方程$f(x)=0$的数值计算方法问题进行讨论。在这个问题的数值计算中，如果对于有解区间不清楚，可以先采用“扫描”的方法求得粗糙的近似有解区间，然后再用其他效率较高的方法求精确的结果。“扫描”法又称搜索法。其步骤是假设$f(x)$在某个区间(a, b)内有一个实单根x^*，从端点$x=a$出发，按预选的步长h一步步从左向右移动，每移动一步进行一次根的“扫描”，并检查每一步的起点x_0和终点x_0+h的函数值是否同号。如果$f(x_0)$与$f(x_0+h)$非同号，即$f(x_0)\cdot f(x_0+h)\leqslant 0$，有根区间必为$(x_0, x_0+h)$，这时可取$x_0$或$x_0+h$作为根的初始近似值。如果$f(x_0)$与$f(x_0+h)$同号，继续向前扫描一步，比较$f(x_1)$与$f(x_1+h)$是否同号，其中$x_1=x_0+h$。这种过程持续到$b$点。图2-1的流程图表示这种方法的计算过程。

§2.2 二 分 法

二分法不是采用等步长扫描，而是改为在有根区间(a, b)取中点$x_0=\frac{1}{2}(a+b)$，计算中点x_0的函数值$f(x_0)$，检查$f(a)$与$f(x_0)$是否同号。如果同号，说明根x^*处于中点右侧，令$a_1=x_0$，$b_1=b$，否则x^*必在x_0左侧（如图2-2所示）令$a_1=a$，$b_1=x_0$，这样新的有解区间为(a_1, b_1)，其长度为区间(a, b)的一半。对新的有根区间(a_1, b_1)采取相同步骤，用中点$x_1=\frac{1}{2}(a_1+b_1)$将有根区间(a_1, b_1)分为两半，再检验根在x_1的哪一侧，又得到新的有根区间(a_2, b_2)，其长度为(a_1, b_1)的一半。通过二分过程可以得到一个近似根的序列

$$x_0, x_1, x_2, \cdots x_k, \cdots$$

此序列以根x^*为极限。

当二分过程进行到第K步时，x_k进一步趋近于x^*，因此

$$|x^*-x_k| \leqslant \frac{1}{2}(b_k-a_k)=b_{k+1}-a_{k+1}$$

当有根区间(a_{k+1}, b_{k+1})的长度小于误差限ε时，x_k近似等于x^*。

二分法虽然不是一种有效的计算方法，但是它的递推过程简单，增加计算函数值的次数可以逐步逼近真实解。当迭代K次以后有根区间缩小为原来(a, b)的2^{-k}，即

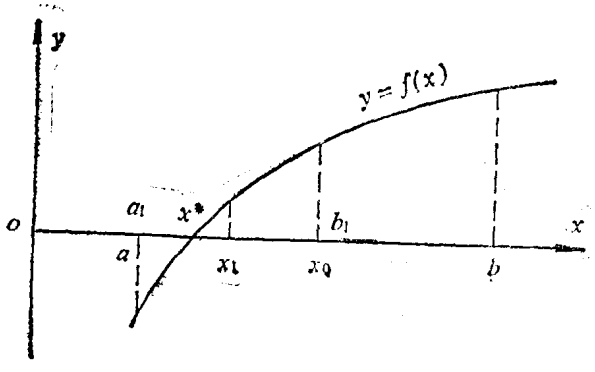


图 2-2 二分法计算过程

$$(b_k-a_k)=\frac{1}{2^k}(b-a)$$

二分法的求解过程如图2-3所示。

例如，求方程$f(x)=x^3-x-1=0$在区间$(1, 1.5)$的实根。要求准确到小数点后第二位。

用二分法，$a=1$，$b=1.5$，且$f(a)<0$，取区间(a, b)的中点$x_0=1.25$，将区间二等分。由于$f(x_0)$与$f(a)$同号，故有解区间在x_0右侧，应令$a_1=x_0=1.25$，$b_1=b=1.5$，从而得到新的有根区间(a_1, b_1)。

对于区间(a_1, b_1)再取中点$x_1=1.375$进行根的二分计算，由于本题$f(x)=0$在(a,b)内只有一个实根，因而$f(x_1)$仍然只要同$f(a)$比较符号即可，图2-3也仅仅适用于这种情况。

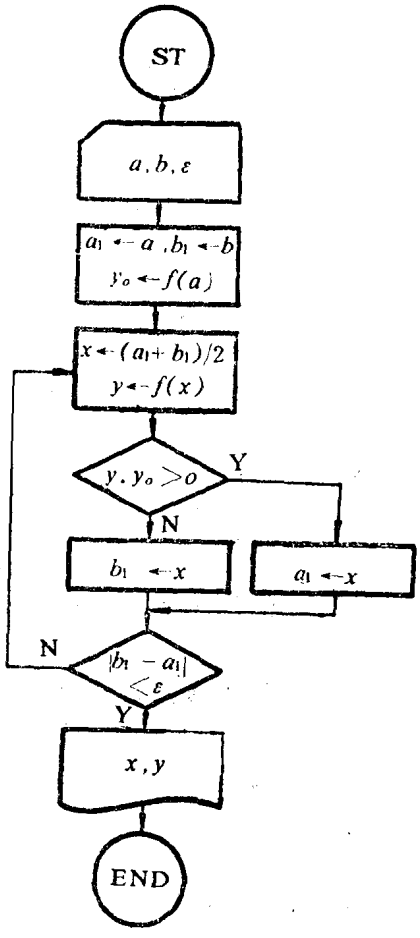


图 2-3 二分法流程图

如此反复进行，每次计算必须根据误差的要求来确定二分法次数。若要求准确到小数点以后第二位，按照误差的有效位数概念，误差限应小于0.005，从误差估计式

$$|x^*-x_k| \leqslant b_{k+1}-a_{k+1}=\frac{1}{2^{k+1}}(b-a)$$

不难看出，当$K=6$时可满足精度要求。

$$|x^*-x_6| \leqslant 0.005$$

以上二分法计算过程的结果如下表2-1。

表 2-1

k	a_k	b_k	x_k	$f(x_k)$符号	f_a符号
0	1	1.5	1.25	-	
1	1.25	1.5	1.375	+	-
2	1.25	1.375	1.3125	-	+
3	1.3125	1.375	1.3438	+	-
4	1.3125	1.3438	1.3281	+	+
5	1.3125	1.3281	1.3205	-	+
6	1.3205	1.3281	1.3242	-	-

如果有根区间内有若干个实数根，可以先用扫描法从a开始，以步长h逐步扫描，当找到第一个有根区间时即采用二分法步骤求第一个根。接着继续用扫描法确定第二个有解区间，再用二分法求根，直到求出（a，b）中的全部实根。这种求根过程可以由本章后面给出的计算程序实例2-1的计算程序来完成。

§2.3 迭 代 法

一 一般迭代法

迭代法是一种逐次逼近法，在求解非线性代数方程时经常使用。迭代法的基本思想是把求解方程$f(x)=0$等效地变换成为迭代公式$x=g(x)$，然后用它的迭代公式$x_{k+1}=g(x_k)$反复迭代，校正根的近似值使之逐步接近精确值，最后得到满足精度要求的结果。

例2.1 求方程$x^3-x-1=0$ 在$x=1.5$ 附近的一个根。

首先将方程改写成如下形式

$$x=\sqrt[3]{x+1} \tag{2-1}$$

用所给的初始近似值$x_0=1.5$代入（2-1）式的右端得到

$$x_1=\sqrt[3]{1.5+1}=1.35721$$

由于x_0不满足（2-1）式。用x_1作为近似值代入（2-1）式的右端，又得到

$$x_2=\sqrt[3]{1.35721+1}=1.33086$$

重复同样步骤，可以逐步求得更精确的值。这种逐步校正过程称为迭代过程，迭代计算公式应写成

$$x_{k+1}=\sqrt[3]{x_k+1} \qquad k=0,\ 1,\ 2,\ \cdots \tag{2-2}$$

表2-2为各步迭代计算的结果。从表中可以看出，如果仅取六位有效数字，计算到x_7时实际上已满足方程（2-1）式的要求，因而得到的根为$x^*=1.32472$。

表 2-2

K	x_k	K	x_k
0	1.5	5	1.32476001
1	1.35720881	6	1.32473595
2	1.33086091	7	1.32471948
3	1.32588377	8	1.32471825
4	1.32494937		

应当注意，上述的迭代计算公式不是唯一的，例题中还可以有如下的迭代公式

$$x=x^3-1$$

总之，对于一般形式的方程式$f(x)=0$，可将它转化为如下迭代公式

$$x=g(x) \tag{2-3}$$

此式的等号两边皆含有未知量x，故称为隐式形式。如果将给出的某个近似值x_k代入（2-3）式右端，则（2-3）式变为可进行迭代计算的显式形式

$$x_{k+1}=g(x_k) \tag{2-4}$$

这样，从给定的初始近似值x_0出发，按显式公式（2-4）可以得到一个数列

$$x_0,\ x_1,\ \cdots x_k\cdots$$

如果这个数列有**极限**，则称迭代公式（2-4）是收敛的，这时数列x_k的极限

$$x^*=\lim_{k\to\infty} x_k$$

就是方程式（2-3）的根。显然收敛性是迭代法中很重要的问题。

下面几何解释迭代过程。方程式$x=g(x)$的求根问题在用几何图形解释时，须确定曲线$y=g(x)$与直线$y=x$的交点p^*（见图2-4和2-5）。对于给定的某个初值x_0，在曲线$y=g(x)$上得到对应的点p_0，其纵坐标为$g(x_0)=x_1$。也就是说，过p_0点引平行于x轴的直线，交直线$y=x$于点Q_1。过Q_1再作平行于y轴的直线，交曲线$y=g(x)$于p_1点。从图2-4和图2-5中不难看出，近似值$x_1=g(x_0)$即为p_1的横坐标。按图中箭头指示的路径继续作下去，可求得近似值x_1，x_2，…。如果迭代收敛，则点序列p_1，p_2，…将越来越逼近所求的交点p^*。

然而迭代的过程不总是令人满意的，有时候迭代过程不收敛。例如上述例题，若不采用

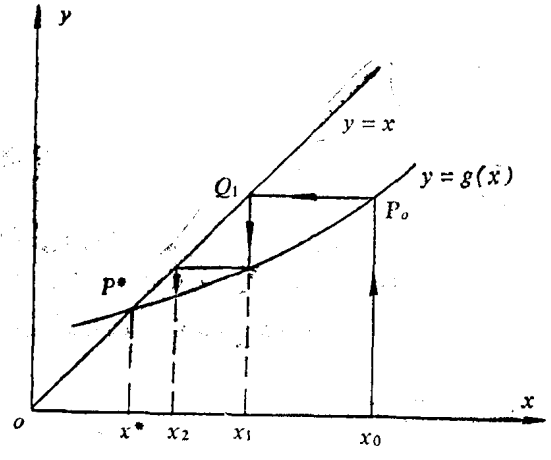


图 2-4　迭代过程收敛

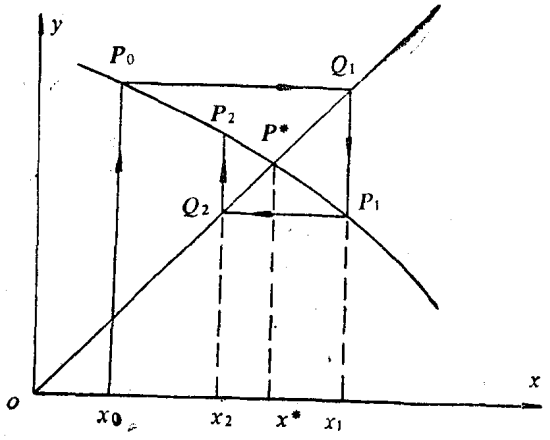


图 2-5　迭代过程收敛