高等学校教学用书

电路的计算机辅助分析与设计

GAODENG XUEXIAO JIAOXUE YONGSHU

冶金工业出版社

高等学校教学用书
电路的计算机辅助分析与设计
沈阳黄金学院　何福民　主编
冶金工业出版社出版
（北京北河沿大街嵩祝院北巷39号）
新华书店总店科技发行所发行
冶金工业出版社印刷厂印刷
787×1092 1/16　印张 15 1/2 字数 365 千字
1989年5月第一版　1989年5月第一次印刷
印数00,001～2,500册
ISBN 7-5024-0413-9
TP·16（课）定价3.10元

前　言

本书系根据冶金、有色高等教育学会“电路理论”学科组对《电路的计算机辅助分析与设计》这门课程的基本要求，按照工业电气自动化专业的教学大纲而编写的。本教材可作为高等学校工业电气自动化、工业自动化仪表、电机与电器制造等电类专业的教学用书。

本书按60～80学时编写，主要内容包括常用的算法程序、网络的矩阵分析与网络的计算机辅助设计。本教材对各种分析与计算方法的介绍均从简到繁、循序渐进、层次分明、思路清晰、重点突出。

本书初稿完成后，曾在校内试用两年。在此期间，先后征求过东北工学院、东北重型机械学院、哈尔滨工业大学、华东冶金学院、吉林电气化专科学校及沈阳冶金机械专科学校等院校任课教师的意见，编者对他们提出的许多宝贵意见深表感谢！

参加本书编写的有沈阳黄金学院的孙振邦（第一、二章）、杨维国（第三章）、何福民（第四、五、六、七章）。本书由何福民任主编，由吉林电气化专科学校高希和、东北重型机械学院庞膺福、沈阳黄金学院金茂勋审阅。

书中不当之处，恳请读者批评指正。

编　者

一九八七年十一月

目　录

第一章　方程求根与函数插值

第一节　研究计算方法的意义

一、概述

人们应用的工具和方法，在一定程度上反映了生产力发展的水平。计算机的应用和计算方法的进步，有力地推动了科学技术的发展。科学技术在发展中提出了大量的科学计算问题，借助计算机运算速度高、存贮信息量大、逻辑判断功能强等特点，能自动地完成复杂的科学技术计算。同时，要用计算机模拟各种物理过程、生产过程，并能对机械、电子、建筑等设计过程进行优化，对生产过程实现最优控制。为了解决上述问题，都需要建立数学模型，并给出一个合适的计算方法。

计算方法的优劣，会影响到能否求出所需的解，并影响到求解过程的快、慢及解的精度。也影响在计算中占用计算机内存的多少，这关系到能否用较少内存完成较复杂的计算。

提倡在科技计算中注重计算方法，讲究程序质量。决不要有了数学式子就原样写入程序中，要先进行分析和整理。也不要用“忽然想到”的方法去解决问题，而要深入考虑能有几种解法，哪种最好。

为说明计算方法的影响，举几个突出的例子：

例如，计算一个$24*24$的行列式。根据行列式的定义，它有$24!$项，每一项都是24个因子的乘积，即有23个乘法。如果忽略计算机在运行中取送数和加减法等操作，用每秒能完成百万次乘法的电子计算机来求这个行列式值，则所需的时间大致是：

$$24!*23*\text{秒}/10^6*\text{天}/86400\text{秒}*\text{年}/365\text{天}\approx 0.45\cdot 10^{12}\text{年}$$

这已超过了目前了解的地球寿命，即按上述算法是不会求出解的。然而，如果改用消去法，则在若干秒内就可用计算机完成这一计算。

又例如，应用行列式法解线性方程组，即克莱姆（Cramer）法则，原则上是可以求解的。但是，要解一个n阶线性方程组，必须解$n+1$个n阶行列式。而每个行列式求解所需的时间已在前例中分析过。当解20阶线性方程组时，所需的时间将是十分可观的。如果方法得当，解同样阶数的线性方程组也就是几个小时的工作量。

综上所述，研究计算方法与提高程序质量是很重要的，自电子计算机问世以来，已经积累了大批经过反复推敲和改进的算法程序，这里面凝聚了许多人的经验与智慧。对于非计算数学专业的科技工作者，首先要解决的是如何把要分析的物理过程转化为可靠的数学模型，接着便是精心选择计算方法。在决定算法时要善于选用现成的通用算法程序块，以便把注意力用在探索你要解决的问题的特性部分，并做好各程序块间的组合。这样既可以保证重点问题探讨得清楚，又避免了重复性的社会劳动。因此，在本书中给出一些常用算法的通用程序，是经过上机计算，并在解决具体课题中应用过的程序。

二、网络分析与设计中算法的选择

在网络分析与设计中，计算工作量大，求解较为困难。有时，还要把电流、电压、电

容、电感、电磁场等物理量转化为一个便于描述的模型，如转变为数字模拟系统、模拟状态阵等。进而再对建立的数学模型进行分析，找出变化规律，给出定量结果，便于完成其设计计算。进行网络的拓扑分析，靠手算只能分析一些小网络，由于高速电子计算机的出现，利用图论来分析网络已经成为可能。

在电路设计中，进行透彻的解析分析是很重要的。因为只有从较全面的分析、比较中，才能看出方案的优劣。可行的方案，不一定是唯一的方案，更难保证是最好的方案。所以，在现代设计方法中，提倡采用最优化设计方法。最优化设计的计算量大，进行逻辑判断的工作量更大，手算图解是难于完成的。所以要采用计算机辅助分析，即CAA(Computer aided analyis)；并在分析的基础上,完成计算机辅助设计，即CAD (Computer aided design)。

电子产品的功能越来越强，电路设计日趋复杂，又要保证产品有较高的经济、技术指标，于是对电路设计提出了可靠性分析的要求。电网络是由一些电路元件（器件）按照一定规律而连接起来的一个复杂整体，尤其是集成电路的出现、发展及其大量应用，在电路中不可避免地会出现各类受控电源，使电路的分析和设计更为复杂。电网络的特性是由组成网络的元件特性和网络的拓扑结构决定的。这样电网络的特性也与元件的各种参数和性能等许多随机变量有关。可靠设计不同于常规设计，常规设计中把许多物理量都看作是常数，而实际上许多量的取值是随机的，是有一定分布规律的随机变量。可靠性设计中，把具有一定的分布密度函数的物理量，当作随机变量，把许多随机变量决定的物理过程，进行统计描述。这样就可以较客观、准确地反映实际情况，使设计的结果较为符合客观实际，能给出明确的可靠程度。进行可靠性分析与设计时，以数理统计为基础，进行许多随机变量间的函数运算，其分析、计算的工作量是相当大的，用CAA、CAD方法则便于开展这些工作。

三、算法与程序质量的评价

现代化的电子计算机功能不断完善，各种软件也日益丰富，为我们进行电路的计算机辅助分析与设计提供了有利的条件。但是，计算机只能机械地执行人的指令，它不会主动地进行思维，更不可能发挥任何创造性。因此，交给计算机执行的解题方案中的每一个细节都必须准确地加以定义，并把全部解题过程完整地描述出来。本书中所说的“算法”，就是指对解题方案和解题过程的准确而完整的描述。

这里所说的“描述”不是用日常语言，而是用数学语言来进行的，并借助于形式语言（算法语言）对其给出精确地说明。即在程序设计中，用算法语言来描述计算方法和过程，本书在编程序时采用的是BASIC语言。

程序设计只是整个计算链条中的一个环节。科技计算的全过程包括提出物理问题、建立数学模型、选择计算方法、编写和调试程序、运行程序完成实际计算，并对所得到的结果进行考核与评价。这些环节是一个循环往复、渐趋正确的过程。

针对要解决的具体问题，恰当地选择计算方法，以求减少计算工作量，保证计算精度。同时，在具体编写程序时，要注意程序设计技巧，提高运算速度，压缩存贮用量，使程序简短，但要求其通用性强，以达到提高程序设计质量之目的。

在计算方法确定之后，提高运算速度的主要办法是减少重复计算，选用最快的操作、合理安排输入输出。具体措施如下：

（1）程序中多次重复出现的常系数如$\pi/5$、$1/(3\pi)$等，要一次算好，存入变量名中备用，不要用一次算一遍。

（2）多次用到的表达式和函数值要一次算好，并贮存以便备用。相同的下标变量如果多次出现在表达式中，要一次取出赋给一个临时变量，避免重复的进行下标变量的查找和运算。例如计算表达式：

$$x=\frac{1}{\sqrt{2\pi}}\cdot\frac{SIN(a_i)+SIN(a_j)}{a_i^2+a_j^2}$$

$$y=\frac{1}{\sqrt{2\pi}}\cdot\frac{SIN(a_i)-SIN(a_j)}{a_i^2+a_j^2}$$

若是按上式直接写成：

```
10  X=(SIN(A(I))+SIN(A(J)))/SQR(2*3.1416)/(A(1)↑2+A(J)↑2)
20  Y=(SIN(A(I))-SIN(A(J)))/SQR(2*3.1416)/(A(I)↑2+A(J)↑2)
```

这样语句的执行速度远不如下列语句的执行速度快

```
10  X=A(I):Y=A(J)
20  T=0.39894/(X*X+Y*Y)
30  X1=SIN(X):Y=SIN(Y)
40  X=(X1+Y)*T:Y=(X1-Y)*T
```

（3）高维数组的执行速度慢于一维数组，必要时应将二维数组换成若干个一维数组。

对于“按列排”的高维数组，最前边的下标变化最快，循环嵌套时，应尽可能使其对应最内层的循环控制变量。例如有10*10的数组A和B，则对于某些编译程序，循环体1）的执行速度比循环体2）的执行速度要快得多。

```
1)
10                FOR  J=1  TO  10
20                FOR  I=1  TO  10
30                  A(I,J)=B(I,J)
40                     NEXT  I
50                     NEXT  J
2)
10                  FOR  I=1  TO  10
20                  FOR  J=1  TO  10
30                  A(I,J)=B(I,J)
40                  NEXT  J
50                  NEXT  I
```

这是因为编译程序时，第一种写法省去了许多重复性的下标计算。

（4）书写表达式时，应注意各种操作的速度差别：幂计算最慢，除法慢于乘法，乘法慢于加法。因此X↑3，不如写作X*X*X；又如2*A不如写作A+A。此外，X↑0.5不如写为SQR(X)，这是因为X↑0.5通常作为EXP(0.5*LN(X))处理，要两次调用基本函数。

尽量减少与外存设备交换信息；减少人工干预；恰当安排打印量和打印顺序等，都有

利于提高运行速度。

（5）尽量减少运算次数

在计算关于X的多项式的值时，例如

$$P=A+BX+CX^2+DX^3+EX^4+FX^5$$

在程序中写作

```
10        P=A+B*X+C*X↑2+D*X↑3+E*X↑4+F*X↑5
```

不如写作

```
10        P=A+B*X+C*X*X+D*X*X*X+E*X*X*X
            *X+F*X*X*X*X*X
```

更不如写作

```
10        P=A+X*(B+X*(C+X*(D+X*(E+F*X))))
```

最好写成递推化形式。所谓递推化形式，即是将一个复杂的计算过程归结为简单过程的多次重复。这种重复在程序中表现为循环。现以多项式的求值问题为例来介绍递推化方法。

设要对给定的x求下列n次多项式的值

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

前一个例子中已经说明，直接逐项求和的算法是不好的。我们用递推法求多项式的值，用t^k表示x的k次幂，u_k表示式（1-1）右端前$k+1$项的部分和，即

$$t_k=x^k$$
$$u_k=a_0+a_1x+a_2x^2+\cdots+a_{k-1}x^{k-1}+a_kx^k$$

由于x的k次幂等于其$k-1$次幂t_{k-1}再乘上x，而前$k+1$项的部分和u_k等于前k项的部分和u_{k-1}再加上第$k+1$项a_kx^k，因此有

$$\begin{aligned}t_k&=xt_{k-1}\\u_k&=u_{k-1}+a_kt_k\end{aligned}\qquad k=1,2,\cdots n \tag{1-2}$$

显然应该取初值为

$$\begin{aligned}t_0&=1\\u_0&=a_0\end{aligned} \tag{1-3}$$

利用初值（1-3），对$k=1, 2, \cdots$，直到n反复执行式（1-2），最终求得的u_n就是多项式的值。

按式（1-2）、（1-3）编写的程序为

```
10        D$="用递推法求n次多项式的值": PRINT  D$
20        INPUT  "N, X"; N, X
30        PRINT  "N="; N; TAB(20); "X="; X
40        T=1:READ  U
50        FOR  K=1  TO  N
60        T=X*T:READ  A:U=U+A*T
70        NEXT  K:PRINT "U="; U
```

80 END

90 DATA a_0, a_1, a_2, …a_n

说明：当没有汉卡时删去第10句。

上述递推算法的每一步要做两次乘法，总的计算量为$2n$次乘法。

四、压缩存贮用量的措施

压缩存贮用量有两个方面：一是把程序写的简捷；二是节约数据占用的空间。

除精心考虑程序的整体安排外，恰当地应用复合语句可以节省内存。BASIC 程序是由一系列语句行号组成的，每一行号可包含一个或几个语句，但不得超过80个字符。语句行号是用来保证正确地执行预定的进程顺序。虽然输入语句时可以按任意顺序输入到内存中去，但在执行程序时是严格按语句行号的次序进行的。一个语句行号后含有若干个语句时，称为复合语句。一些袖珍机和微机允许采用复合语句，即在一个语句行号后，用冒号“:”把单句隔开，形成复合语句。本书的一些程序中，采用了复合语句。因为每个行号都要占4个字节内存，若一个行号能容纳几个语句，就节省了一些用来表示语句行号的内存。当使用冒号组成复合语句时，应注意程序的执行路线以防止出错。

节约数据空间的关键是恰当地运用数组，因为存贮空间的需求量主要由一些大数组来决定。这里的根本问题仍是选择计算方法。不同的计算方法所需要的数组容量不同。计算方法确定之后，就要在程序整体规划中精心安排数组，尽可能不引入只用一次的数组，应使一个数组有多种用途。能用简单变量代替的就不用数组。当数据量过大时，可采用外存的方法，用到这些数据时分别将其由各种外存调到内存中去。

五、框图的研究

框图是描述算法的一种语言，它能直观地显示算法的全貌，对计算方案给予形象的描述。在进行程序设计之前，先画出程序框图，清晰地表示出程序的执行过程，便于在程序设计中安排输入、输出，建立数组和循环体，表明子程序的设置和程序块间的组装。使设计者做到心中有数，思路清晰，语言简明。

同时框图给读程序带来很大方便，它醒目地显示了程序的走向，表达了程序设计的总体方案。尤其是程序较长时，借助框图能较快地明确程序的脉络，每一部分的作用和各部分之间的相互关系，在读程序时框图会起到路标的作用。

对于调试和运行程序，框图也有鲜明的指导作用。如果程序有误，可以对照框图检查程序；如果运行时间较长，借助框图可随时掌握程序执行的情况。

框图有繁有简。繁者是细框详图，便于设计和读清楚程序的细节；简者是粗框略图，它线条鲜明，能简捷地表明设计思路。可根据程序的长短、繁简情况选择框图的画法，也可以简略的粗框图与详细的细框图同时应用。

一种算法实质上是解决某一数学问题的一个处方，它包括一套指令，一步一步地按照它去执行，就能得到所讨论的问题的解。

对于某一个算法可以用不同的方式进行表达，其中框图是最好的方式之一，是一种算法的通用表达方式，它与程序设计中所使用的语言和运行时所应用的计算机类型无关，通用性好，能最一般地表明程序设计的思路，反映程序执行的过程和对其中各个环节的安排。

本书中所给出的框图中，应用了表1-1所示的几种图框。

表 1-1　　图　　框

图　　形	名　　称	意　　义
开始	启 始 框	表示开始启动
	输入输出框	指明输入输出过程
	处 理 框	表示要进行的工作（不包括比较判断）
	检 查 框	表示算法的判断检查部分
结束	结 束 框	表示运行过程结束
⟶	箭　　头	指示各框执行的顺序

用框图描述算法举例：

例如要解二元一次联立方程组

$$a_{11}x_1+a_{12}x_2=b_1$$
$$a_{21}x_1+a_{22}x_2=b_2$$

这个方程组的行列式解法可表述为

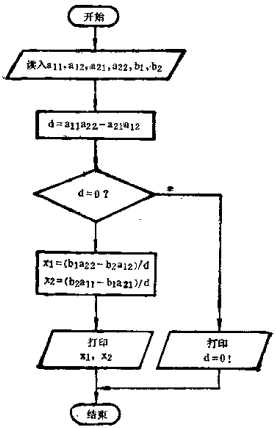

图 1-1　解方程组程序框图

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

首先应判别这个行列式的值是否为零，可能出现两种情况：

1）如果$d\neq 0$，则可按下式求出计算结果

$$x_1=(b_1a_{22}-b_2a_{12})/d$$
$$x_2=(b_2a_{11}-b_1a_{21})/d$$

2）如果$d=0$，则或者无解，或有无穷多组解，这是奇异的情形。

利用上述的图框符号，可以把解方程组的过程，用框图语言形象地描述出来，如图1-1所示。

六、误差的产生

在解题过程，不可避免地会产生各种误差。产生误差的原因有：截断误差、舍入误差等。

在微分、积分及无穷级数求和等运算中，计算机只能完成有限次的算术及逻辑运算。而这些数学运算是通过极限过程来定义的，因而产生了截断误差。

例如计算指数函数e^x的前n项和

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

其截断误差为

$$R_n(x)=\frac{x^{n+1}}{(n+1)!}e^{\theta x}$$

其中 $0<\theta<1$

计算过程中会遇到数据位数很多，甚至会是无穷小数，然而受计算字长的限制，用机器代码表示的数据必须舍入成一定位数，产生舍入误差。

第二节 方程求根

一、方程实根所在区间的确定

科学技术中的许多问题常常归结为解函数方程

$$f(x)=0$$

的问题。这里，$f(x)$可能是超越函数；也可能是代数多项式。尤其常遇到$f(x)$是多项式的情况。当$f(x)$是多项式时，我们称$f(x)=0$为代数方程。使$f(x)=0$成立的x^*称为该方程的根。求方程根x^*的具体过程，一般按两步进行：

第一步，确定根所在的某个区间，即首先确定一个根的粗糙近似值，一般称它为“初始近似值”。

第二步，求满足一定精度要求的近似根x^*，即从初始近似值开始，用求根的方法得到预定准确度的根。

如何求得根的初始近似值呢？关键在于确定$f(x)=0$的实单根所在的区间。有时，方程$f(x)=0$的根的分布可能是很复杂的。因此，我们假定$f(x)$在某个区间$[a,\ b]$内有且仅有一个实数单根x^*。我们从一个端点出发，比如从左端点$x=a$开始，按预先选定的某个步长h一步步地向右跨，每跨一步进行一次根的“扫描”，即检查每一步的起点x_0和终点x_0+h的函数值是否同号。若$f(x_0)$与$f(x_0+h)$的值为异号（图1-2），则

$$f(x_0)\cdot f(x_0+h)\leqslant 0$$

这时，所求的根x^*必在点x_0与x_0+h之间，即

$$x^*\in[x_0,\ x_0+h]$$

这样，就找到了方程实根的所在区间。可以把x_0、x_0+h或该区间内的某个值做为根的初始近似值。

上述求方程$f(x)=0$的近似根的过程，可以用框图进行描述（图1-3），按着这个框图可编出如下的程序

```
10  D$="这是一个求方程f(x)=0的近似值的程序": PRINT  D$
20  READ  A, H: XA=A
30  DEF  FNS(X)=F(X)
40  YA=FNS(XA)
50  XA=XA+H
60  IF  YA*FNS(XA)>=0  THEN  40
70  PRINT  "XA=";  XA;  TAB(20);  "XA+H=";  XA+H
80  END
90  DATA  a, h
```

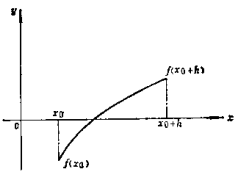

图 1-2　方程实单根所在的区间

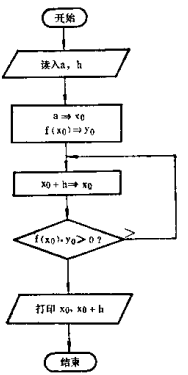

图 1-3　逐步扫描找方程初始近似值

例 1-1　考察方程

$$f(x)=x^3-x-1=0$$

显然有

$$f(0)=-1<0$$
$$f(2)=5>0$$

可以看出在〔0，2〕区间内至少有一个实根。

如果应用上面的程序，从$x_0=0$出发，取步长$h=0.5$向右进行根的扫描。则程序中检查框内判别值的符号变化如表1-2所示。

表 1-2 用扫描法确定根的初始近似值

x	0	0.5	1.0	1.5	2.0
$f(x_0)\cdot f(x_0+h)$	+	+	+	−	−

从表1-2可知，$x^*\in[1.0, 1.5]$。

对于步长h，取值要恰当。只有步长足够小，才能利用逐步扫描法得到足够精确的近似根。但步长越小，扫描的次数就要增加，从而使计算量变大。如果要求找到精度较高的根，仅用这种逐步扫描方法是不切实际的。

基于上述原因，通常求根过程分两步走，先用逐步扫描方法找到根的某个初始近似值x_0；再用某种求根方法，逐步使近似值向精确值靠近。

以下介绍几种使方程的根逐步精确化的方法，如二分法、叠代法等。

二、二分法求根

设方程$f(x)=0$在区间$[a, b]$内有且仅有一个实根x^*。取根存在的区间的中点

$$x_0=\frac{1}{2}(a+b)=b_1$$

用$x_0(b_1,0)$把区间$[a,b]$分成两段，然后进行根的扫描，判定所求的根在区间$[a, b_1]$段，还是在区间$[b_1, b]$段中。如果$f(x_0)$与$f(a)$是同号，即$f(a)\cdot f(x_0)>0$，则$x^*\in[b_1,b]$区间；否则，$x^*\in[a, b_1]$。

若$x^*\in[a, b_1]$，则取$a=a_1$，$x_0=b_1$。则$x^*\in[a_1, b_1]$区间，且$|a_1b_1|=\frac{1}{2}|a\ b|$（图1-4）。

对于被压缩了的区间$[a_1, b_1]$再应用上述的二分法，如此反复地二分下去，便可得到一系列有根区间

$$[a, b],\ [a_1, b_1],\ [a_2, b_2],\ \cdots\cdots,\ [a_k, b_k],\ \cdots$$

其中每个区间都是前一个区间的一半，因此区间(a_k, b_k)的长度为

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

显然，这样无限地应用二分法，便可使这些区间最终收缩于一点x^*，该点x^*就是所求的根。

总之在二分法求根过程中，某一有根区间$[a_k, b_k]$的中点为

$$x_k=\frac{1}{2}(a_k+b_k)$$

x_k是一个根的近似值，而在二分过程中则可得到一个近似根的序列

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

这个序列是以根x^*为极限的。

在分析、解决实际问题时，要求根满足一定的精度即可。不可能，也不必要无限地应用二分法。由于

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

当我们要求根的准确度为ε时，则要求

$$|b_{k+1}-a_{k+1}|<\varepsilon \quad \text{即可}$$

图1-5是二分法求根的框图

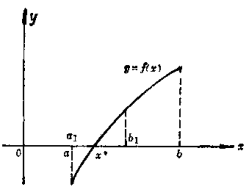

图 1-4 二分法求根

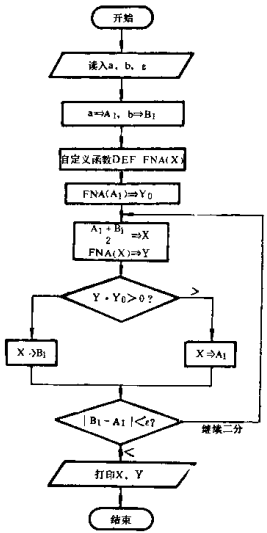

图 1-5 二分法求根的框图

按框图1-5写出二分法求根的程序

```
10    D$="这是二分法求根的程序": PRINT  D$
20    READ  A, B, E
30    A1=A : B1=B : K=0
40    DEF  FNA(X)=F(X)
50    YA=FNA(A1)
```

```
60   X=(A1+B1)/2 : K=K+1
70   Y=FNA(X)
80   IF  Y*YA>0  THEN  110
90   B1=X
100  GOTO  120
110  A1=X
120  IF  ABS(B1-A1)>E  THEN  60
130  PRINT  K
140  PRINT  "ROOT  IS"; X
150  END
160  DATA  a, b, ε
```

例 1-2 求方程

$$f(x)=x^3-x-1=0$$

在区间 (1, 1.5) 内的实根。要求准确到小数点后的第 2 位，即$e=0.005$。

用二分法求根，这里已知$a=1$，$b=1.5$，$e=0.005$，且$f(a)<0$。

取区间(a, b)的中点$x_0=1.25$，即x_0把(a, b)区间二等分。由于$f(x_0)<0$，即 $f(x_0)$与$f(a)$同号，所以根必在x_0的右侧。这时应取：$a_1=x_0=1.25$；$b_1=b=1.5$，得到了新的有根区间 (a_1, b_1)。

再把区间 (a_1, b_1) 二等分，其中点$x_1=1.375$，并进行根的扫描。如此反复二分下去，二分法求根过程记录于表1-3中。

表 1-3 二分法求根

k	a_k	b_k	x_k	$f(x_k)$的符号
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	1.3203	−
6	1.3203	1.3281	1.3242	−

只要二分 6 次，便达到了预定的精度

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

所以，$x_k=x_6=1.3242$便是所要求的近似根。

在运行给出的二分法求根通用程序时，显然第160句应添写为

```
160  DATA  1, 1.5, 0.005
```

而第40句（自定义函数语句）是

```
40  DEF  FNA(X)=X*X*X-X-1
```

作了上述两条语句的修改便可运行

```
RUN↵
```

6

ROOT IS 1.3242

三、叠代法与叠代法的加速

叠代法是一种重要的逐次逼近方法。这种方法用某个固定公式反复校正根的近似值，使其逐步精确化，最后得到满足预定精度要求的结果。对于要解的具体方程$f(x)=0$，要首先设法把它化成下面的型式

$$x=g(x) \tag{1-4}$$

由于方程式(1-4)的右端含有未知量x，故称为隐式方程。在函数$g(x)$的定义域中任取一值x_0作为初始值，把x_0代入$g(x)$中得到$g(x_0)=x_1$，作为方程解的下一个近似值。再把x_1代入$g(x)$中，把$g(x_1)=x_2$，…，则有

$$x_{k+1}=g(x_k) \tag{1-5}$$

这样，从给定的初始值x_0出发，按显式(1-5)得到一个方程解的数列

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

如果这个数列有极限，则称叠代格式(1-5)是收敛的，这时数列x_k的极限

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

就是方程$f(x)$的根。

例 1-3 求方程

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

在$x=1.5$附近的一个根。

把方程式按式(1-4)改写成隐式形式

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

从所给的初始值$x_0=1.5$开始，按式(1-5)作叠代计算

$$x_1=\sqrt[3]{x_0+1}=\sqrt[3]{1.5+1}=1.35721$$

比较x_0和x_1的值可知，x_0并不是所给方程的根。再用x_1作根的近似值进行叠代计算，又得

$$x_2=\sqrt[3]{x_1+1}=\sqrt[3]{1.35721+1}=1.33086$$

由于x_2与x_1间仍有偏差，再以x_2作根的近似值进一步作叠代计算，并重复进行上述叠代计算，表1-4中记录了各次叠代的结果。

表 1-4 叠代记录

k	x_k	k	x_k
0	1.5	5	1.32476
1	1.35721	6	1.32473
2	1.33086	7	1.32472
3	1.32588	8	1.32472
4	1.32491		

从表1-4可以看到，当仅取六位数字时，x_7与x_8已完全相同，故可以认为x_7是所给方程的根，即

$$x^*=1.32472$$

对上例的叠代过程，可作如下的几何解释（图1-6）：