化学工程计算方法

王树森编

化学工业出版社

化学工程计算方法

王树森　编

化学工业出版社

内 容 提 要

本书主要介绍化学工程程序设计中常用的计算方法，供具有初步 FORTRAN 语言基础的读者使用。全书共分九章，除绪论外，其他各章均为化学化工计算中的常用算法，如非线性代数方程及线性代数方程组的计算方法、数值微分和积分、常微分方程及偏微分方程的数值解法、插值、曲线拟合及过程最优化等。各章内容一般由基本概念、数值方法、应用举例、框图和程序等四部分组成，为加深读者理解，各章均有练习题。

本书可作为高等院校化学化工专业学生应用数学参考书，也可供有关科技人员参考。

化学工程计算方法

王树森 编

责任编辑：徐世峰
封面设计：许 立
*
化学工业出版社出版发行
（北京和平里七区十六号楼）
北京印刷一厂印刷
新华书店北京发行所经销
*
开本787×1092 1/16 印张23¾ 字数599千字
1989年3月第1版 1989年3月北京第1次印刷
印 数 1—3,180
ISBN 7-5025-0378-1/TQ·275
定 价 7.45元

序

随着我国四化建设的发展和科学技术的进步，电子计算机在各个领域、其中包括化学化工计算中的应用越来越广泛。过去用手工计算需要几天甚至几个月才能完成的计算任务，现在用计算机在几秒钟或几分钟内就可能得到解决。因此，计算机应用的普及，受到人们普遍的重视。

应用计算机求解化学化工计算问题，一般来说，它的处理要经过以下几个步骤：

1．化学化工问题的数学表示；

2．根据所得到的数学表示式的类型，选择适当的数值方法；

3．列出程序设计要点，画出程序框图，写出算法程序；

4．将程序和原始数据输入计算机进行计算；

5．对计算结果的物理意义进行解释。

在目前的计算方法教材中，虽然对有关数值计算的基本理论和计算方法作了详尽的介绍，但是它们在化学化工计算的具体应用方面却介绍得较少。这样，以解决化学化工中各种实际计算问题为对象，有关计算方法方面的书就显得非常需要，本书就是为适应这一需要所作的尝试。

本书共分九章。在内容安排上，各章一般先介绍与建立方程或与所介绍的数值方法密切相关的一些基本概念，然后再介绍最常用的数值方法，并给出在化学化工计算中的应用举例，最后给出具有代表性的一些程序框图和FORTRAN程序。为了配合各章所介绍的内容，各章后面还给出了习题。

在应用举例的安排上，第二章至第五章的应用举例，多为化学动力学、化学热力学、以及仪器分析方面的实例，而第六、七两章的应用举例，则以动量、热量、和质量传递过程的分析计算为主，第八章和第九章的举例，分别为实验数据处理和过程最优化。

本书在编写过程中，数学博士唐云副教授审阅了本书原稿中的数学部分，作者在此致以衷心的感谢。由于作者水平所限，书中错误之处在所难免，望批评指正。

目　录

第一章 绪 论

1.1 化学化工问题的数学表示

一个化学化工问题的计算结果是否可靠，首先不是取决于程序的编写，以及计算的精确程度，而取决于我们所采用的化学化工问题的数学表达式，是否真正从本质上描述了这一化学化工过程。要做到这一点，仅有数学知识显然是不够的，必须还要具备足够的化学化工专业知识，以及必要的经验，这样才能准确地写出我们准备求解的化学化工问题的数学表达式。为说明这一点，下面我们不妨举一个例子。

对于许多化学反应，特别是简单化学反应来说，其反应速率都可以表示成仅与温度有关的函数f_1，与仅与组成有关的函数f_2的乘积，即：

$$r_i = f_1(T)\cdot f_2(c)$$

或

$$r_i = k\cdot f_2(c) \tag{1.1}$$

式中：r_i——反应速率；

T——温度；

c——组成；

k——随温度而变化的项。

当使用不同的化学定律，k与温度的依赖关系也就有所不同。例如当使用阿雷尼厄斯定律，则有：

$$k = k_0 e^{-E/RT} \tag{1.2}$$

式中：k_0——频率因子；

E——反应的活化能；

R——气体常数。

当采用碰撞理论时，则有：

$$k = T^{1/2} e^{-E/RT} \tag{1.3}$$

而采用过渡态理论时[1]，则：

$$k = T\cdot e^{-E/RT} \tag{1.4}$$

显然，三种表达式表示出三种对温度不同的依赖关系。那么究竟采用哪种表达式才能正确地、本质地描述化学过程的反应速率呢？

如果我们所研究的反应是：A和B两种反应物碰撞生成不稳定的中间态AB^*，然后AB^*再分解成产物AB，即：

$$A + B \longrightarrow AB^* \longrightarrow AB \tag{1.5}$$

碰撞理论认为，两反应物之间有效的碰撞次数是控制反应速率的关键，而不稳定态中间物的分解则是无关紧要的。这种理论实际上假设中间产物的分解极为迅速，因而并不影响整个过程的反应速率。过渡态理论则认为，中间产物的分解是控制整个反应速率的关键，这实际上假设中间产物形成的速率非常大，以致于在所有时间内它都以平衡浓度存在，而它是如何形成的，则是无关紧要的。这样，碰撞理论认为式(1.5)的第一阶段是慢的，是控制速率阶段，

而过渡态理论则认为式(1.5)第二阶段是速率控制因素。如果我们确切地知道反应的第一阶段是控制速率阶段，那么我们就应选择碰撞理论表达式来描述反应速率；另一方面，我们若确切地知道反应第二阶段是控制速率阶段，我们就应选择过渡态理论表达式来描述反应速率。所以，这两个理论实际上是互为补充的。

我们如果把式(1.2)、(1.3)、(1.4)归纳成一个方程式：

$$k = k_0' T^m e^{-E/RT} \quad 0 \leqslant m \leqslant 1 \tag{1.6}$$

对式(1.6)两边取对数，然后对T求微分：

$$\frac{\mathrm{d}(\ln k)}{\mathrm{d}T} = \frac{m}{T} + \frac{E}{RT^2} = \frac{mRT + E}{RT^2}$$

因为对于大多数反应来说$mRT \ll E$，所以我们可以忽略mRT项，于是我们可以写作：

$$\frac{\mathrm{d}(\ln k)}{\mathrm{d}T} = \frac{E}{RT^2}$$

或

$$k = k_0 e^{-E/RT}$$

此式即(1.2)式，这说明阿雷尼厄斯定律是碰撞理论以及过渡态理论所表示温度依赖关系的最好近似。通过以上讨论我们可以看出，只要我们对所研究的化学化工过程有深入的了解，对有关化学定律能熟练应用，那么写出正确描述这一化学化工过程的数学表达式应是并不困难的。

一个数学方程，从化学化工的角度来看，它有可能是描述过程机理的较好的表达式，但从数学的角度来看，未必是利用数值方法求解的最佳表达公式。例如有一个一级连续分解反应：

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C \tag{1.7}$$

通过分析计算知道反应速度常数k_1和k_2之间的关系为：

$$k_2 = g(k_2) = k_1 \exp[(k_2 - k_1) t_{\max}] \tag{1.8}$$

式中$t_{\max}$是B的浓度达到最大值时的时间。如果k_1和t_{max}的数值都已知道，那么从原理上讲只要选定k_2的初值，利用(1.8)式迭代求解，就可以求得k_2的近似值。

但式(1.8)中k_2有两个根，根据迭代理论得知[2]，迭代只能收敛于满足下述条件的根：

$$|\mathrm{d}g(k_2)/\mathrm{d}k_2| \equiv |g'(k_2)| < 1 \tag{1.9}$$

将式(1.8)对k_2求导，知当$k_2 < 1/t_{\max}$时可满足上述收敛条件。但我们如果已经知道$k_2 > 1/t_{\max}$，那么就不能利用(1.8)式迭代对k_2求解。

解决这一问题的方法是变更方程的形式，使得它能满足收敛条件。例如对方程(1.8)两边取对数得：

$$\ln k_2 = \ln k_1 + (k_2 - k_1) t_{\max}$$

整理后得：

$$k_2 = f(k_2) = k_1 + [\ln(k_2/k_1)]/t_{\max} \tag{1.10}$$

对k_2求导得：

$$\frac{\mathrm{d}f(k_2)}{\mathrm{d}k_2} = \frac{1}{k_2} \cdot \frac{1}{t_{\max}} \tag{1.11}$$

因已知

$$k_2 > 1/t_{\max}$$

所以

$$df(k_2)/dk_2<1$$

满足收敛条件。这样利用式(1.10)作为迭代公式对k_2迭代求解，就可以求得我们所要求的根。

对一个化学化工问题进行数学处理时，还经常需要对所研究的体系进行必要的、合理的简化或理想化。因为影响一个化学化工过程的因素可能是多方面的，如果我们不分主次，全部考虑在我们的数学表达式中，将会使计算工作量大大增加，但对提高所求解的精度却没有明显的好处。这一点在传热分析、传质分析中尤其显得突出。

1.2 程 序 框 图

对于一个比较复杂的问题，计算问题中的相互关系也是比较复杂的。为了准确无误地编写算法程序，在编写较复杂的程序之前，总是先用框图的形式表示出解题的逻辑关系，这就是程序框图，也称为流程图。

在编写程序框图的过程中，我们要用到下述框图符号[3]，现分别说明如下：

1．表示各种处理功能的符号

例如，图1.1表示一种代数运算，它表示乘积Mx和b之和取代一个称为A的变量。这里特别要注意，在程序框图中，任何代数算式中的等号都表示：等号左边的量被等号右边的量取代。例如$i=i+1$对于普通代数运算是不成立的，因为i不论等于任何数值，它都不可能等于这个数值再加1。但在程序框图中它表示：新的i等于旧的i加1，或i被$i+1$所取代。

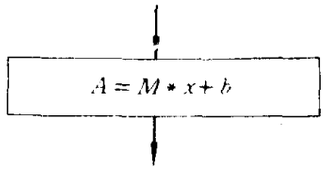

图 1.1 表示处理功能的符号

由于程序框图中代数式中的等号表示一种取代关系，所以上面所举的例子又可用图1.2表示。

2．表示判断的符号

典型的判断符号如图1.3所示。这样一个语句要求计算机判断变量N是否等于或小于50。如果是，下一步运算应沿通路A进行，如果N大于50，对问题的回答是"No"，所以下一步应沿通路B进行。

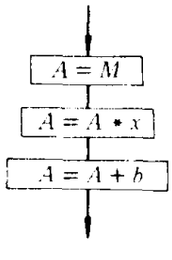

图 1.2 $A=M*x+b$式的框图

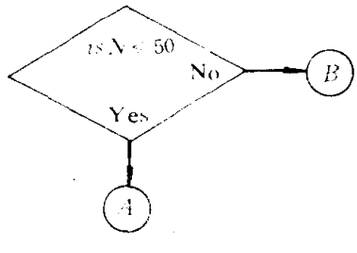

图 1.3 判断符号

3．表示程序的开始、停止和继续

表示程序的开始、停止和继续的符号，如图1.4所示。

4．表示输入或输出的符号

输入表示提供处理所需的信息，输出表示记录处理后的信息，典型的输入及输出符号如图1.5所示。

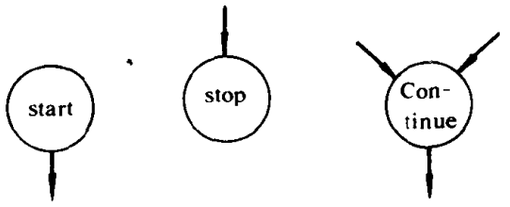

图 1.4　表示程序的开始、停止、继续及返回的符号

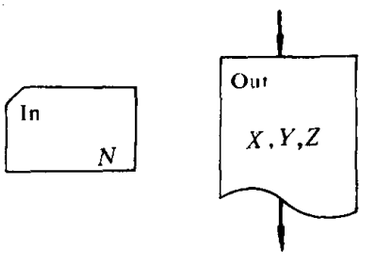

图 1.5　输入及输出符号

下面举几个例子说明程序框图的编写。

例1.1　计算以下式所表示的乘积之和：

$$y = \sum_{i=1}^{n} C_i x_i \tag{1.12}$$

在这个表达式中，n是一个常数，它规定了此题中所包含的c的总数及x的总数。下标i规定了c_i和x_i的特定值。图1.6是进行加和运算的程序框图。

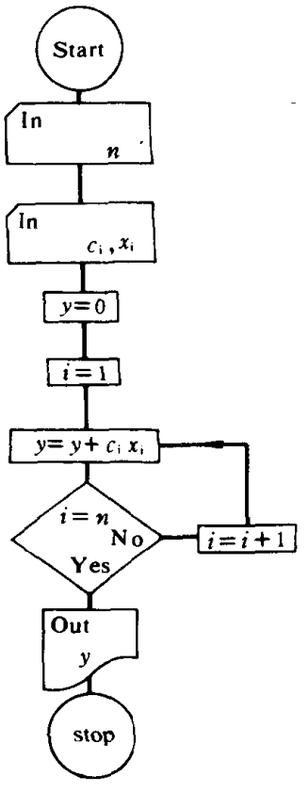

图 1.6　例1.1的框图

用计算机对这个问题求解的第一步，是要规定输入变量的总数n。然后给出每一个c_i和x_i的数值。这些数据构成了输入数据。

为了提供一个正确的起点，对于变量和标码数i都要规定初值。如图所示，我们规定变量y的初值为0，i的初值为1。

计算机所进行的第一次计算是：

$$y = 0 + c_1 x_1$$

过程进入下一步是作出判断。这时i的值为1，如果我们规定n为大于1的某个整数，那么i和我们所规定的n比较，显然i不大于或等于n，出口应是标有“No”的通路，在这个通路上，标码数i由1再加上1变成2，再用这个新的i值计算y。第二次计算是：

$$y_{新值} = y_{前一次的值} + c_2 x_2$$

或

$$y = c_1 x_1 + c_2 x_2$$

这样，i再和n进行比较，如果i仍小于n，过程将重复进行。当$i = n$时，把所得到的y值输出，然后计算停止。

这个例子中重复计算的特点，是许多问题在编写程序框图时的特征之一，它可以有效地利用数字计算机的固有容量。我们把这个例子中所进行的这一计算过程称为循环。也就是说，相同的计算步骤重复进行，而每一次都使i值增加1。可以看到，整型变量i被用作循环计数单元。当然，在许多情况下我们需要作的循环不只是一次，有时是两次，甚至多次。

例1.2　有一组实数，把它们按从大到小的次序排列起来。

处理这类问题，一般是选给定数组的第一个数作为基准，然后用这个数和其他各个数进行比较，一旦发现某一个数比第一个数大，就用这个数和第一个数调换位置，也就是说用这

个较大的数作为基准继续和其他数进行比较，待和所有的数都进行一次比较以后，我们就选择出这个数组中最大的数，我们把它作为 A_1 存贮起来。为确定第二大的数，可重复进行上述过程。但这一次是把第二个数作为基准，用它和第三、第四，以至第 n 个数进行比较。这个过程一直进行下去，我们就把给定的一组数按从大到小的次序排列起来了。完成这一问题的程序框图如图1.7所示。

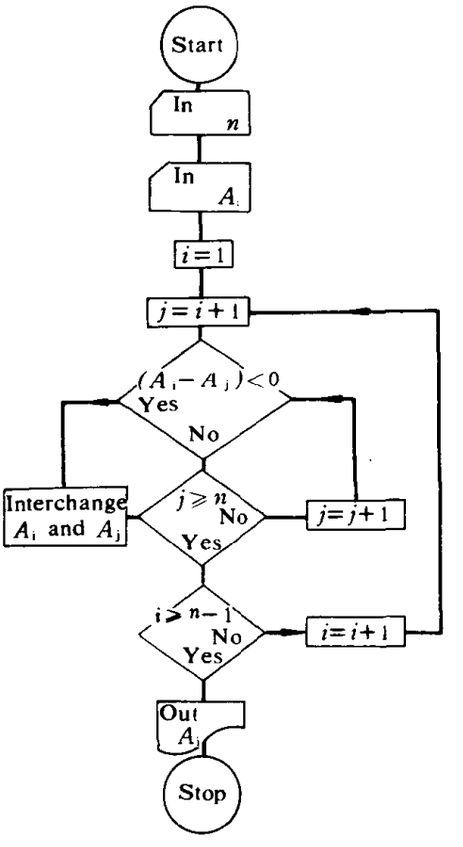

图 1.7　例1.2的框图

检验一个程序框图工作情况是否正常的简单方法，是用一个简单例子试验一下。就本题而言，可用下面的例子进行试验：

$$n = 4 \qquad A_i = \begin{bmatrix} 7 \\ 6 \\ 8 \\ 2 \end{bmatrix}$$

我们注意到在程序框图中用了两个标码数 i 和 j。标有 i 的 A 值专门作为比较基准，而标有 j 的 A 值则为其它各个数。

工作开始时 i 等于1，j 等于 $i+1$ 即 j 等于2。所以要检验 $A_1 - A_2$ 是否小于0。因为 $A_1 = 7$，$A_2 = 6$，所以 $A_1 - A_2$ 并不小于0，沿着标有“No”的通路进入下一个判断符号，此时 $j = 2$，并不大于 n，所以新的 j 增加到3，应比较 A_1 和 A_3。由于 $A_1 - A_3$ 是负值，所以 A_1 和 A_3 进行交换：A_1 变成 8 而 A_3 变成7。因为 $j = 3 \neq 4$，于是 j 增加到4，这时再用新的 A_1 和 A_4 进行比较，即 8 和 2 进行比较，A_1 大于 A_4，此时 $j = n$，所以就求出最大的 A 值，这时这组数的排列次序变为：

$$A_i = \begin{bmatrix} 8 \\ 6 \\ 7 \\ 2 \end{bmatrix}$$

下一步是确定 i 是否大于等于 $n-1$。由于 $i = 1$ 而 $n = 4$，答案是 No，于是 i 增加到2，而 j 变为 3。比较 A_2 和 A_3 后使得第 2 个数与第 3 个数互换位置，这时这一组数的排列变为：

$$A_i = \begin{bmatrix} 8 \\ 7 \\ 6 \\ 2 \end{bmatrix}$$

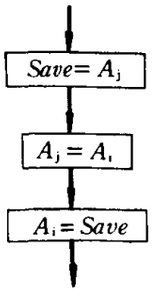

图 1.8　A_i 和 A_j 互换位置

继续进行计算表明，上述结果就是我们要求的最终结果。

在上面的程序框图中有一个步骤是 A_i 和 A_j 互换位置，这个步骤在计算机中是用图1.8所示的流程实现的。

例1.3　计算双重连加式：

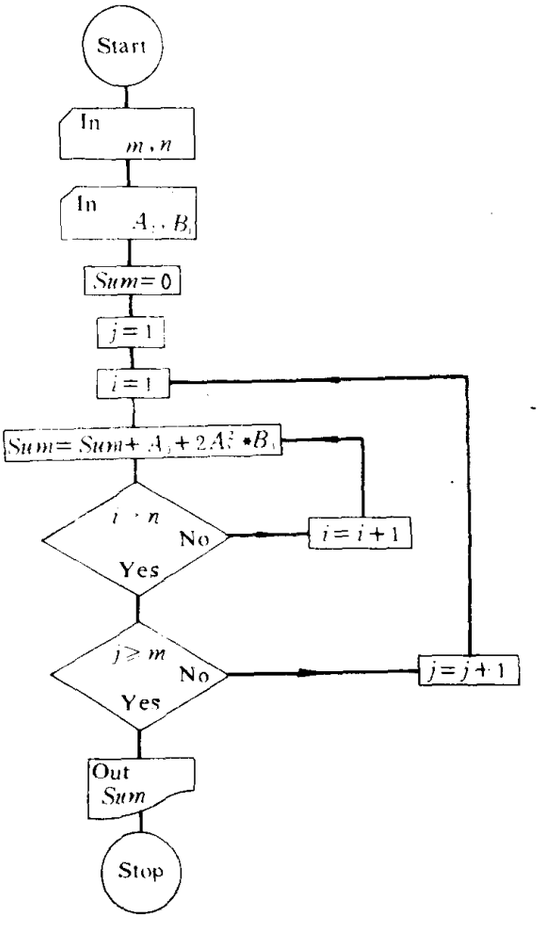

图 1.9 例1.3的框图

$$\mathrm{Sum}=\sum_{j=1}^{m}\sum_{i=1}^{n}\left(A_i+2A_j^2B_i\right)$$

程序框图如图 1.9 所示。首先我们把 i 和 j 的初值都规定为1，然后保持 j 为常数1，使 i 从 $i=1$ 变化到 $i=n$，计算结果得到：

$$\mathrm{Sum}=\sum_{i=1}^{n}\left(A_1+2A_1^2B_i\right)$$

再保持 j 为常数2，仍然使 i 从 $i=1$ 变化到 $i=n$，计算结果得到：

$$\mathrm{Sum}=\sum_{i=1}^{n}\left(A_2+2A_2^2B_i\right)$$

如此继续进行，直到保持 $j=m$，使 i 从 $i=1$ 变化到 $i=n$。把所有连加式再加起来，就得到最终结果。

例 1.4 给出一个二维数组 A_{ij}，其 i 和 j 的范围都是 1 到 n。要求构成一个新的一维数组 D_i，这个一维数组由 A_{ij} 的对角线元素组成。

处理这类问题的方法可以是使 i 和 j 分别从 1 变化到 n，然后比较 i 和 j 是否相等，如果相等，元素 A_{ij} 即为所求数组的元素，将其输出。它的程序框图如图1.10所示。

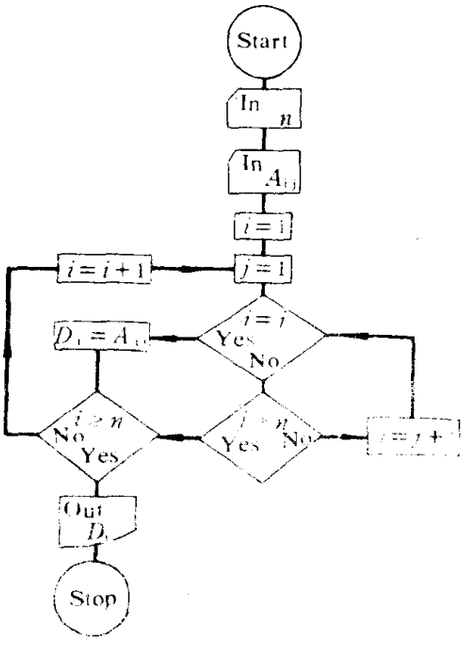

图 1.10 例1.4的框图之

但考虑到我们输入给计算机的全部数据，每一个数据都占有一个存贮单元，所以我们只要把A_{ij}的全部对角线元素A_{ii}从它们的存贮单元调出来，就构成我们所要求的一维数组D_i。其程序框图如图1.11所示。

1.3 FORTRAN程序

程序框图编好之后，就可以根据程序框图，应用任何一种语言编写出计算机的算法程序。但在工程计算中应用最普遍的语言是FORTRAN语言。

关于FORTRAN语言，近年来国内出版了不少专著进行介绍。FORTRAN是英文FORMULA TRANSLATOR的缩写，意思是“公式翻译”。FORTRAN语言于1956年首次出现在美国，其后不断发展，形成很多种类，但最流行的是FORTRAN Ⅱ和FORTRAN Ⅳ。后来，美国标准化协会于1966年正式公布了两个美国标准文本：基本FORTRAN（相当于FORTRAN Ⅱ）和FORTRAN（相当于FORTRAN Ⅳ）。本书所介绍的源程序都是用FORTRAN语言编写的。为了叙述方便，我们把FORTRAN语言简称FORTRAN，把用FORTRAN语言编写的源程序简称为FORTRAN程序。下面的例子是根据事先编好的程序框图（图1.12）用FORTRAN语言写出的算法程序。

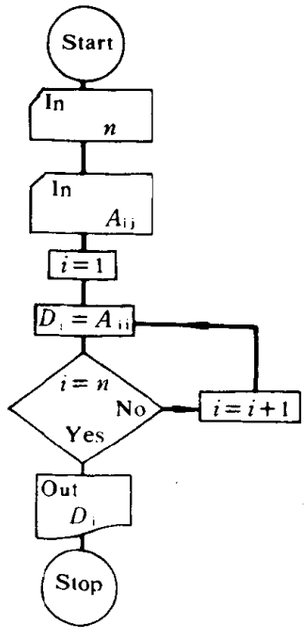

图 1.11 例1.4的框图之二

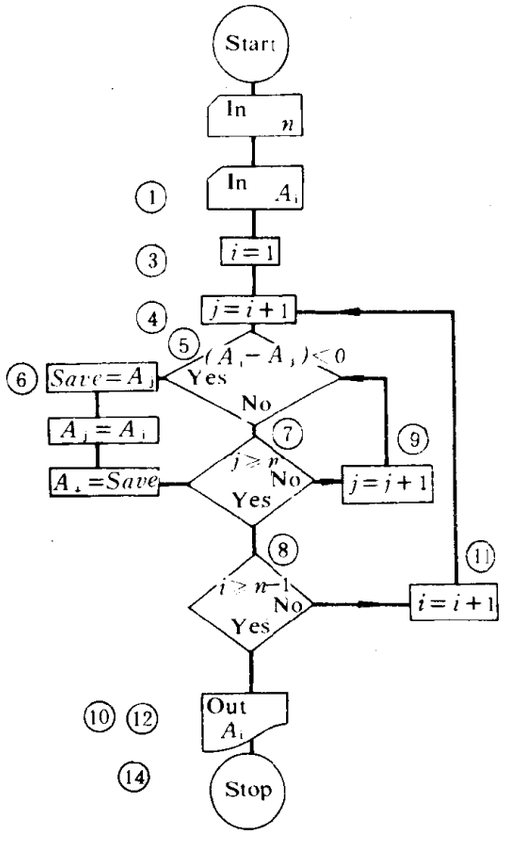

图 1.12 某程序的框图

```
      DIMENSION  A(1000)
100   FORMAT(I4)
101   FORMAT(E14.8)
      READ(5, 100)N
      M=N-1
1     READ(5, 101)(A(I), I=1, N)
3     I=1
4     J=I+1
5     IF(A(I)-A(J))6, 7, 7
6     SAVE=A(J)
      A(J)=A(I)
      A(I)=SAVE
7     IF(J-N) 9, 8, 9
9     J=J+1
      GO TO 5
8     IF(I-M) 11, 10, 11
11    I=I+1
      GO TO 4
10    DO 12 I=1, N
      WRITE (6, 101) A(I)
12    CONTINUE
14    CALL EXIT
      END
```

习　　题

1．由 n 个数 A_i 构成的一维数组，绘制求此数组中最小数的程序框图。

2．绘制计算 $n!$ 的程序框图。

3．n 个数的平均值可由下述关系式求得：

$$A_{vg}=\frac{1}{n}\sum_{i=1}^{n}A_i$$

绘制计算 A_{vg} 的程序框图。

4．由 n 个数 A_i 构成的一维数组，绘制输出此数组各个数的绝对值的程序框图。

5．求平方根的迭代公式为：

$$x_{i+1}=\frac{1}{2}\left(x_i+\frac{N}{x_i}\right)$$

式中 N 为欲求平方根的数，x_0 为 N 的平方根的初始估计值。绘制程序框图，要求迭代过程进行到 x_{i+1} 和 x_i 之差小于或等于某给定误差值。

6．数 M 和 Q 的值已经给定，现要从 M 中连续减去 Q，$\frac{1}{2}Q$、$\frac{1}{4}Q$，$\frac{1}{8}Q$ 等。设计一程序框图，用以确定全数小于给定值 E 所需要的最少的相减次数。若相减 100 次后仍得不到所要求的解，则停止运算。

7．给定 x 的两个线性函数：

$$y_1=f_1(x)\qquad y_2=f_2(x)$$

设计一程序框图，用以确定能满足下式的 x 的值：

$$|y_2-y_1|\leqslant \text{error}$$

式中error为给定误差值。

8．求下述一元二次方程的根：

$$ax^2+bx+c=0$$

一元二次方程求根的计算公式是：

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

方程的两个根可能是实重根，两个相异实根，或一对共轭复根，均需按照判别式 b^2-4ac 的情况来判定。对各种情况的求根公式如下：

$$-\frac{b}{2a}\pm\frac{\sqrt{4ac-b^2}}{2a}i \qquad \text{当}b^2-4ac<0$$

$$-\frac{b}{2a} \qquad \text{当}b^2-4ac=0$$

$$-\frac{b}{2a}\pm\frac{\sqrt{b^2-4ac}}{2a} \qquad \text{当}b^2-4ac>0$$

根据判断条件编制程序框图。

9．某实验需按如下规律计算函数值

$$f(x)=\begin{cases}11.6x^{1/2}-18.4x^{3/2}+87.2x^{5/2}+x^{7/2} & 0\leqslant x\leqslant 1\\ 29.6x^{1/2}-185.5x^{3/2}+655.7x^{5/2} & 1\leqslant x\leqslant 2\\ x^{1/2}-0.2x^{3/2} & 2\leqslant x\leqslant 3\\ x^{1/2} & 3\leqslant x\leqslant 4\end{cases}$$

若已知 x，编出计算函数值的程序框图，并用FORTRAN 语言写出计算程序。

10．编写完成下列各题计算的FORTRAN 程序：

a. 求$1^2+2^2+\cdots\cdots+99^2$

b. 求$1+\frac{1}{1}+\frac{1}{2}+\cdots\cdots+\frac{1}{20}$

c. $z=a+\ln\frac{x^2+b}{2}$

其中：$x=1.0,2.0,0.1$；$a=0.1,0.8,0.05$；

$b=1.0,10.0,1.0$

d. 计算渐开线上M点的坐标

$$x=\cos t+t\sin t$$
$$y=\sin t-t\cos t$$

其中：$t=0,0.9,0.1$

e. 已知

$$e=1+\frac{1}{1!}+\frac{1}{2!}+\cdots\cdots+\frac{1}{n!}$$

求它的近似值

$$s_n=1+\frac{1}{1!}+\frac{1}{2!}+\cdots\cdots+\frac{1}{n!}$$

直至满足

$$|s_{n+1}-s_n|<10^{-7}$$

参 考 文 献

[1] Levenspiel, O.: **"Chemicel Rheaction Engineering"**, Znd Edition, John Wiley & Sons, 1972.

[2] Stark, P.A.: **"Introduction to Numerical Methods"**, MacMillan, 1970.

[3] Ketter, R. L. and Prawel, S. P.: **"Modern Methods of Engineering Computaion"**, McGraw-Hill, 1969.

第二章 非线性方程

在这一章我们将讨论非线性方程的性质和求解问题。所谓非线性方程，是指含有变量的乘积、乘方，或变量的三角函数、指数函数如$\sin x$、$\cos x$、e^x、$\ln x$等的代数方程。这一类方程经常出现在热力学计算、动力学计算、化学平衡计算、以及电化学计算中。在化工设计计算中也会出现非线性方程。因此，方程的根通常对应于我们所要求的压力、温度、浓度、反应速率常数，反应器的几何尺寸等。

除了个别特殊情况以外，非线性代数方程的求解，没有普遍适用的代数解法可以使用，在某些情况下甚至不能确定解是否存在。因此，非线性代数方程的求解，通常采用迭代法。

非线性代数方程的根，可以是实根，也可以是虚根，鉴于我们的研究内容，我们只介绍方程实根的解法。

2.1 迭代法和收敛速率

2.1.1 迭 代 法

如前所述，非线性代数方程的求解，也就是要求出满足一般式为$f(x)=0$的非线性方程的变量x的值。在有些情况下，x值很容易求得，例如：

$$f(x)=ax^2+bx+c=0$$

只要进行因式分解，或利用公式

$$x=[-b\pm(b^2-4ac)^{1/2}]/2a$$

就很容易求出x的值。但是，如果x的幂高于2，例如：

$$f(x)=ax^5+bx^3+cx^2+dx+e=0$$

或x本身就是幂，例如：

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

上述简单的方法就不适用，这时就要用迭代法求根。在使用迭代法时，需要选择一个合适的迭代函数$g(x)$和迭代初值x_0。这样，利用下述关系式就可求得根ξ的近似值：

$$x_{r+1}=g(x_r) \qquad r=0,1,2,\cdots \tag{2.1}$$

$g(x)$的选择应满足这样的条件：使得在根这一点的函数值等于这个根，即：

$$\xi=g(\xi)$$

如果存在着一个整数r_0，使得所有$r\geqslant r_0$时误差

$$|x_r-\xi|<\epsilon$$

那么我们就称由式(2.1)所规定的迭代序列x_0，x_1，x_2，……是收敛的。其中ϵ是一个很小的正数。由于ξ是我们要求的未知数，所以通常我们假设当相邻两迭代值之差小于ϵ时，则此迭代过程收敛，即：

$$|x_{r+1}-x_r|<\epsilon$$

如果随r的增加，$|x_{r+1}-x_r|$连续增加，那么我们就称此迭代过程是发散的。在实际运算中，我们总是在程序中设置一个允许迭代次数的上限，以免收敛很慢或根本不收敛时运算无休止