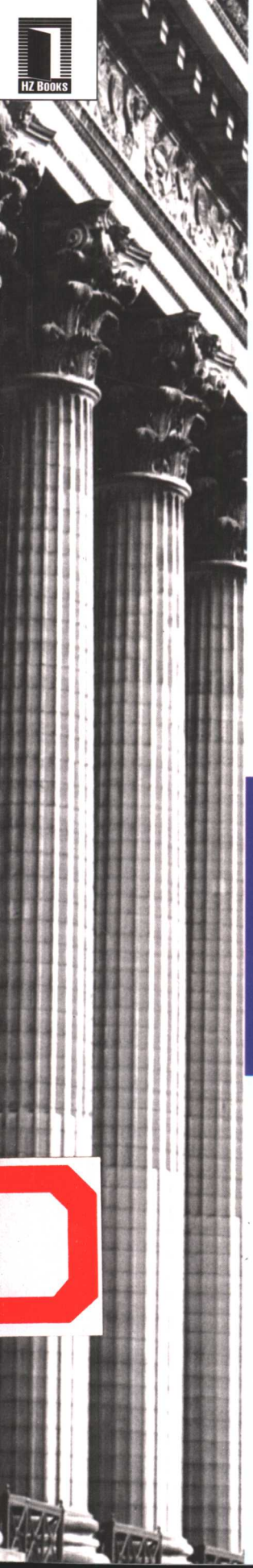
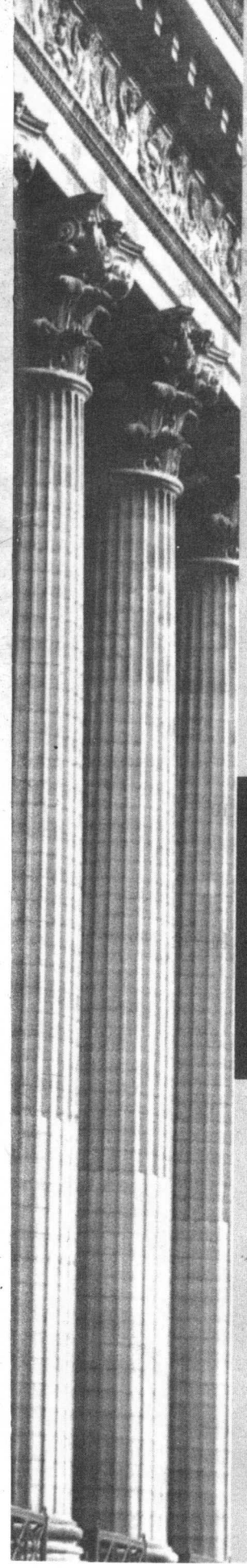


高等院校
计算机教材系列

周 煦 编著

计算机数值计算方法及程序设计

机械工业出版社
China Machine Press



高等院校
计算机教材系列

周 煦 编著

计算机数值计算方法及程序设计

机械工业出版社

本书旨在将数值计算方法和程序设计方法学有机地结合，以便用计算机解决工程和科学技术中的计算问题。本书以数值计算方法的理论为主线，辅以“自顶向下、逐步求精”和典型的模块程序设计方法，全面介绍了解决插值、积分、常微分方程、方程求根、线性方程组等问题的基本思想、计算公式、算法设计、程序框图设计、C语言源程序以及误差分析等内容。本书结构清晰、重点突出、深入浅出，既适合作为高等院校以及成人教育数值计算课程的教材及教师参考书，也适合作为广大科技工作者的自学用书。

图书在版编目（CIP）数据

计算机数值计算方法及程序设计 /周煦编著. －北京：机械工业出版社，2004.10
（高等院校计算机教材系列）
ISBN 7-111-14877-0

Ⅰ.计… Ⅱ.周… Ⅲ.① 电子计算机－数值计算－高等学校－教材 ② 程序设计－高等学校－教材 Ⅳ.① TP301.6 ② TP311

中国版本图书馆CIP数据核字（2004）第071095号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
策划编辑：温莉芳
责任编辑：朱 劼
北京昌平奔腾印刷厂印刷·新华书店北京发行所发行
2004年10月第1版第1次印刷
787mm × 1092mm 1/16 · 19.25印张
印数：0 001 - 4 000册
定价：28.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

计算机科学与技术的广泛应用，已成为现代科学技术和生产力发展的重要标志之一。在现代信息社会里，计算机知识和应用能力已成为当代大学生和工程技术人员知识能力结构的重要组成部分。使用计算机进行科学计算是计算机最先应用的领域，而且至今仍是计算机科学的重要领域。学好计算机数值算法及程序设计，不仅对于使用计算机进行科学计算，顺利解决工程技术中的计算问题有着重要的意义，而且对于学习使用计算机解决其他各种问题也有极大的启迪作用。

使用计算机进行科学计算时，人们最为关心的是计算机是否能够完成预期的计算任务。当前的计算机，就其本身运算的实质而言，它所能进行的仅仅是加、减、乘、除四则运算与逻辑运算。人们经常遇到的一些计算问题，例如求三角函数等超越函数的值、函数的导数、积分函数的原函数、微分方程的通解等，这些都是计算机无能为力的。但是，如果所求的是近似的数值解，那么计算机就大有用武之地，而且一般均可以满足预先提出的计算精度要求。而这正是本书所要完成的任务。

数值计算方法既是一门古老的学科，又是一门新兴的学科，电子计算机的产生和发展大大地促进了数值计算方法的发展，现在数值计算方法已深入到各个学术领域；程序设计方法学是随着计算机的产生和发展而发展起来的一门科学，只有把数值计算方法和程序设计紧密地结合起来，把算法变为计算机能直接执行的程序，才能真正使用计算机帮助人们解决各种极其复杂的计算任务。目前，以上两门学科都有许多专著和教材，但将两者有机地结合起来形成一门独立的课程尚不多见，本书试图将数值计算方法和程序设计方法学融为一体，这也是一种尝试。

本书使用计算功能强大的C语言编写出各种常用算法的程序，且均在计算机上调试通过，只要有适当的环境，就可以直接在计算机上用来执行某些计算任务。本书在介绍各种算法时，大体上按照基本思想、计算公式、算法设计、程序框图设计、源程序和误差的顺序进行叙述。全书以算法为主线，突出结构化、模块化设计思想，结合算法介绍“自顶向下，逐步求精”和典型模块程序设计的方法。本书作者在大学从事了40年的教学工作，有着丰富的教学经验。本书是作者近20年来开设数值算法及程序设计课程的总结。本书内容系统、概念明确、条理清楚、由浅入深、语言流畅、图文并茂，既适合作为高等院校的本科生、专科生、研究生以及成人教育学生及教师的教材或教学参考用书，也可以作为广大科技工作者的自学用书。

在本书的编写过程中，得到了全国高等学校计算机学会和机械工业出版社的关怀和支持。华中科技大学的张凤祥教授和机械工业出版社华章分社编辑为本书提出了很多中肯的建议，在此谨向上述部门、单位以及支持本书编写、出版的同志们、老师们表示衷心的感谢。

由于编者的水平有限，加上编写匆忙，难免有所疏漏，恳请读者予以批评指正。

编　者

2004年3月

目　录

第1章 概 论

内容提要 对于21世纪高等院校的学生来说，使用计算机解决其业务领域中遇到的问题是必须具备的能力，否则将无法适应社会主义经济建设和科学技术飞速发展的形势。计算机数值计算方法是计算机科学领域首先研究和发展的对象，而且至今仍是计算机科学领域的一个重要的研究对象。计算机数值计算方法这门课程涉及计算机程序设计和数值计算方法两方面的内容，本章是本门课程的理论基础。本章将概括介绍计算机数值计算方法的性质及研究对象，数值计算方法的基本概念与办法，数值计算中的误差以及程序设计方法等内容。

1.1 数值计算及程序设计课程的性质及其研究对象

随着科学技术和社会的发展，大量复杂的计算问题不断出现在人们的面前。在计算机没有问世之前，为了解决某些复杂的计算问题，不少科学家献出了大半生，甚至毕生的精力。例如，1867年法国天文学家达拉姆尼（Dalamny）花了整整20年的时间，求解了一个天体运动的摄动级数展开式，以解决月球运行轨道的数值解。18世纪英国数学家香克斯用了毕生的精力，于1873年把圆周率π计算到了小数点后707位。把科学家的精力主要用于一些具体的计算上，这并非解决复杂计算问题的好方法，于是人们开始研究妥善解决计算问题的方式。

1.1.1 数值计算及程序设计的研究对象

由于科学技术的发展，提出了将科学家从繁琐的计算工作中解放出来的要求，而实际上有些复杂的计算问题根本无法用人工计算完成，例如，1948年美国原子能研究中有一项计算问题，需要进行900万道运算，大约相当于1500名工程师一年的计算工作量； 要准确预报天气变化情况，就要求解成千上万个偏微分方程组，这大概相当于数万名工程师一年的计算工作量。不难看出，上述列举的计算问题都有这样一个特点——它们都是人工计算（包括使用电动计算机这样的计算工具）所难于或根本无法胜任的计算任务。为解决这样的计算任务，一门新的科学——计算机数值计算方法及计算机程序设计方法学应运而生。

1. 数值计算方法

随着社会的进步，出现了许多复杂的科学技术问题。为了解决这些问题，先要根据提出的问题和条件建立数学模型，然后进行解算。从数学模型求解的角度看，有解析法和数值解法、图解法等。例如，已知物体作匀变速直线运动时，其速度随时间变化的规律为$v=v(t)=at$，其中a为常量，今欲求$t=0$到t这段时间内物体所通过的路程s。显然，由物理学可知:

$$\begin{aligned} s &= \int_0^t v(t)\mathrm{d}t \\ &= \int_0^t at\mathrm{d}t \\ &= \frac{1}{2}at^2\Big|_0^t \end{aligned}$$

$$= \frac{1}{2}at^2 \tag{1-1}$$

式（1-1）即为此数学模型的解析解。

若问题变得复杂一些，所给出的运动不是匀变速直线运动，而是非匀变速直线运动，如 $v(t)=\sin 2t$ 或 $\frac{\sin t}{t}$，则式（1-1）就无法使用解析法求出，而只能使用数值解法、图解法求出其近似解。

通常把求数值解的问题称为数值计算问题，把求数值解的方法称为数值计算方法，简称为计算方法或数值方法。

计算方法是数学的一个分支，是一门古老的科学，已有几百年的历史。它的研究方向是求解各种数学问题的数值方法及有关的理论，其主要内容有插值计算、数值积分计算、微分方程数值解法、方程求根和线性方程组的数值解法等。

2. 科学计算

计算方法虽已有几百年的历史，但由于只能依赖于人工计算，所以其发展速度极其缓慢。而且科学技术和社会的发展，给人们提出了许多人工计算所无法完成的计算，这一类计算不是人类尚未认识的计算问题，而是人们能够计算但由于计算工作量太大而无法完成的计算任务。随着电子计算机的诞生和飞速发展，这一类计算任务已经得到了解决。我们将此类计算问题称为科学计算。

人工计算无法完成，而必须借助于计算机完成的科学技术计算问题称为科学计算。例如，前提到的原子能数学模型计算，天气预报计算都属于科学计算。在科学技术飞速发展的今天，大部分复杂的计算都属于科学计算。

3. 计算机数值计算方法

要解决科学计算问题，当然要研究计算的方法，于是随着计算机科学的发展，一门使用计算机解决数值计算问题的科学产生了，这就是计算机数值计算方法。研究科学计算的方法、途径及其理论的科学称为计算机数值计算方法。在电子计算机出现之前，数值计算方法发展缓慢，由于计算工作量极大，人工计算无法完成，所以实用性不大。但在电子计算机出现后，过去无法计算的问题得到了解决，从而极大地提高了数值计算的实用性，同时数值计算方法也在适应计算机解题的过程中获得了飞速的发展，产生了计算机数值计算方法。

在学习计算机数值计算方法的时候，要注意两点：

1) 计算机只会做四则运算，更严格地讲它只会做加法，其他运算都是转化为加法来完成的，例如求对数、三角运算、求幂、求平方根，求积分、解算微分方程、解线性方程组等，最终都要化作加法计算。因此，在使用计算机进行数值计算时研究算法就显得特别重要。

2) 计算机数值算法就其本质而言是离散的。按照高等数学中连续的概念，就实数来说，在任意两个相异的实数间有无穷多个实数。但对于计算机来说，给计算机输入初始数据时，只能逐个地输送，输入数据的个数是有限的，不可能输入无穷多个数据；让计算机输出计算结果时，也只能是逐个数据输出，个数也是有限的，不可能输出无穷多个数据。从计算机的角度看，计算机处理的对象是数据，数据是一个一个的，不是“连续”的，而是“离散”的。处理数据也是一个操作一个操作地进行，两个操作彼此是独立的，不存在“连续”的关系。无疑，离散与连续两者是既相互区别又相互联系。就数的概念而言，从初等数学的“离散”

上升到高等数学的“连续”，而现在又要从高等数学的“连续”再回到计算机数值算法的“离散”，即要完成一个离散—连续—离散这样一个思想方法上的否定之否定的螺旋式上升过程。只有这样，才能从本质上认识计算机数值算法。

4. 程序设计方法学

使用计算机帮助我们解决问题的一个先决条件，就是要让计算机理解并执行人确定的各种指令。程序是人们根据解题需要而设计、编制的解题方法和步骤，从计算机执行动作的角度而言，程序也是一系列用计算机语言表示的指令的有序集合。要让计算机帮助解决问题，就必须设计、编制程序。人们用计算机能够识别的语言设计、编制、调试程序的过程就是程序设计。为了用计算机语言编写出高质量的程序，使计算机能正确、快速、顺利地帮助人们解决问题，就必须对程序设计的过程及方法进行研究。研究程序结构、功能，程序设计的原则、方法及其理论的科学就是程序设计方法学。

5. 计算机数值计算方法及程序设计

计算机数值计算方法和程序方法学的交叉就是计算机数值计算方法及程序设计。确定科学计算的算法，并运用计算机语言将算法编制成程序，用以解决科学计算问题，这就是计算机数值计算方法及程序设计的研究对象。

1.1.2 学习计算机数值方法及程序设计的重要性

在学习了计算机程序设计语言后，应不失时机地学习计算机数值计算方法及程序设计，原因如下。

(1) 选择正确的计算机算法是解决问题的必要前提

前面已提到，要借助计算机解决问题，就必须编写正确的程序，而要编写出正确的程序，除了掌握计算机语言外，还必须选择与确定正确的算法。从数学的角度看，算法就是解题的方法和步骤；从使用计算机的角度看，计算机算法就是按解题要求选择合适的计算机操作所组成的操作序列。没有正确的计算机算法也就不可能借助计算机解决问题。请注意，我们强调的是计算机算法，是计算机解题的方法与步骤，即计算机能够识别和理解的解题方法与步骤。如果方法很先进，但计算机不能识别和理解，那么计算机也就无法帮助我们解决问题。例如，人们可以利用恒等变换通过数学推导解题，但计算机却不能使用数学上的恒等变换来解题，计算机解题的方法是离散的，我们只能使用计算机能识别和理解的方法去解题。因此，选择正确的计算机算法是使用计算机解决问题的必要前提，也是计算机数值计算方法及程序这门课程研究的重要内容。

(2) 使计算机能够完成它所不会或无法完成的那些运算

前面说到，计算机只会做加法，其他运算都是先转化为四则运算，最终转化为加法来进行的。所以如何将其他运算转化为四则运算，最终转化为加法，也是本门课程所要研究的一个内容。

值得一提的是，有些计算任务虽然看起来只有加、减、乘、除，但实际上其计算工作量却大得惊人，有时连计算机也只能“望洋兴叹”！在人们心目中，计算机进行加法运算的速度极快，似乎不管计算工作量如何巨大，计算机都能及时完成，但这仅是人们的一种错觉，事实并非如此，确实存在着计算机都无法完成的计算任务。如计算一个25阶行列式的值，按

行列式理论，展开后共有25!项，每项内包含25个数字相乘，即计算每一项需要做24次乘法，由此计算整个行列式的值需要进行的乘法次数为:

$$25! \times 24 \approx 3.7227 \times 10^{26}$$

要让计算机完成这么多次的乘法运算，需要多久的时间？现以2.5×10^9次/秒（相当于2.5G）的计算机来进行此项运算任务，该计算机一年能完成的计算次数为:

$$365 \times 24 \times 60 \times 60 \times 2.5 \times 10^9 = 7.884 \times 10^{16}$$

根据以上计算，该计算机完成25!行列式所需的时间为:

$$T = 3.7227 \times 10^{26} \div (7.884 \times 10^{16})$$
$$\approx 4.7 \times 10^9 \text{（年）}$$

这是一个天文数字，说明计算机实际上是无法完成如此巨大的计算任务的。那么计算机是不是连一个25阶行列式的值都无法计算呢？当然不是。事实上，只要选择一种合适的算法，该行列式只要几秒钟就能计算完成，几秒钟和几十亿年，这是多么鲜明的对照，而这也说明研究算法的重要性。

(3) 提高程序编写质量

要正确使用计算机，人们必须首先编写出高质量的程序。要编写出高质量的程序，就必须满足两个基本条件:

1) 选择正确的算法。程序和算法两者是相辅相成的，程序是算法的表达方式，算法是程序所要表达的实质内容。高质量的程序离不开高质量的算法。

2) 使用正确的程序设计方法。当前，随着计算机科学的迅速发展，一门新的科学——程序设计方法学正在逐步走向成熟。运用程序设计方法学所阐明的原则、原理与技术来编写程序，就能保证程序的正确性，增加程序的可读性和可维护性，这对于大型程序的设计有特别重要的意义。

如何满足以上两个条件，也是本门课程所要研究的重要方面。

(4) 促进思维发展

选择算法和编写程序来解决科学计算问题，这是一种与运用数学演绎推理解决计算问题不同的思维方式，即离散的思维方式。连续和离散，这两种思维方式既相互矛盾，又相互统一，相辅相成。掌握算法和编写程序的方法，对于完善思维方式，培养和提高思维能力，促进思维的发展具有重要的意义。

(5) 推动计算机科学的发展

计算机的出现，为计算机数值计算方法和程序设计方法学的发展提供了条件。在计算机没有出现以前，数值计算方法已存在了几百年，但其发展极其缓慢，因为当时的计算工具无法完成数值计算方法所要求的巨大的计算工作量，只有在计算机出现后，才能完成巨大的计算工作量。计算机的出现为数值计算方法的实现提供了保证，数值计算方法才能够取得突飞猛进的发展。程序设计更是在计算机出现后因需要编写机器语言程序或高级程序语言程序才发展起来的，没有计算机也就不可能出现程序设计方法学。反过来，数值计算方法和程序设计方法学的繁荣和发展，又丰富了计算机科学的内容和领域，推动了计算机科学的发展。

综上所述，在学习过计算机高级程序设计语言后，不失时机地学习计算机数值计算方法及程序是非常必要的，这有助于提高我们使用计算机解决科学计算问题的能力。

1.1.3 课程的基本要求

通过本门课程的学习，读者应当达到以下一些基本要求:

1) 了解数值计算方法的基本概念与方法，程序设计的原则和基本方法。

2) 掌握插值、数值积分、常微分方程、超越方程求根、线性方程组等常用的数值计算方法的基本理论。

3) 根据算法进行编码。具备使用高级语言编写一般常用算法的源程序的能力，并培养输入、运行、调试、修改程序的能力。

4) 借助于电子计算机解决常用算法范围内的实际科学计算问题。

5) 学习按照自向下、逐步求精和结构化程序设计方法，完善思维方式，不断提高思维能力。

1.2 数值计算方法的基本方法与途径

1.2.1 离散变量与离散化

离散是相对于连续而言的，我们称在有限集合或可数集合中取值的变量为离散变量。把连续变量变换为离散变量的过程称为离散化。

在计算机进行数值计算时采用离散变量或将连续变量离散化是由其本身的基本特点所决定的，因为计算机只能一个一个地计算数值，更由于存储单元的关系，一个变量在某一时刻只能有一个值，当某个变量取得新值的时候，按照计算机存储“取之不尽，新来旧去”原则，变量原来的值将为新值所替代。因此，按计算机工作的方式，变量的值只能是离散值。计算机输出计算结果时，也是一个个具体的数值，即离散的数值，而不是一个解析式。这里值得一提的是，计算机算法解题也是离散方式的，即通过一个个具体的操作进行的，一旦一系列操作完成，问题也就随之解决。计算机的操作主要有“+”、“-”、“×”、“/”、“与”、“或”、“非”、“传送”、“比较”等，使用这些操作对已知数据进行加工和变换，计算机最终以离散的数字表达解题的结果。由此可知，其解题过程是一个动态的变化过程。

当需要将一个连续变量离散化时，常用的方法是在一定的区间上运用解析式计算出一批结点处的函数值，并将自变量及其对应的函数值列成表格。列表是表示离散函数和离散变量最常用的形式之一。

使用列表方式将连续变量离散化时，如何确定每个结点处的自变量值？最常用的方法是等距结点法。例如，指数函数$y=e^x$是一个连续函数，今欲在某个区间（如[1.4125, 1.5000]）将其离散化。具体的做法是从$x=1.4125$开始，每增加一个定值（如0.0005）计算一个y值（当然可以根据具体问题的需要确定这个定值），并将对应的x值和y值列在表中，这样就将指数函数离散化了。表1-1是从该表中截取的一部分。经过这样的处理后，变量x已不再是连续变量，不能随意取值，而是成为每隔步长$h=0.0005$取值的离散变量了。

表1-1　x值和y值的列表

x	1.4125	1.4130	1.4135	1.4140	1.4145	1.4150
y	4.106208	4.108262	4.110316	4.112372	4.114429	4.116486

值得指出的是，数值计算中将连续变量离散化是指对最终计算结果的处理，而在问题的

处理过程中往往要使用相反的过程，即把离散变量连续化。例如，在使用计算机进行函数插值处理时，首先是依据给定的条件，将离散变量连续化，将其变换为连续变量，这就是构造插值公式；然后根据需要计算的插值点，使用插值公式计算出插值结点处的函数值，此时实际上又将变量离散化了，同时问题也在此过程中获得了解决。由此可见，在解决问题的过程中，连续和离散通常是交叉使用的，两者相辅相成。因此这里介绍离散，并不是要用离散来否定连续，而是要把两者紧密地结合起来，完善思维方式，以利于问题的解决。

1.2.2 逼近

用简单函数$y(x)$近似地代替函数$f(x)$称为近似代替，也称为逼近。其中，$f(x)$称为被逼近函数或被近似函数，$y(x)$称为逼近函数或近似函数。逼近函数与被逼近函数之差

$$E(x)=f(x)-y(x)$$

称为逼近的误差或余项。

使用逼近方法，一方面使问题的解决得以简化，同时也会带来误差。尽管使用逼近方法来解决问题会带来误差，但在一些场合下人们不得不用逼近来解决问题。例如，已知的函数不是用解析式表示，而是用列表或图形的方式给出；计算工作量极大，简化问题、节省计算工作量已成为解决问题的关键或函数的表达式极复杂，分析函数性质较困难，在这种情况下，人们往往使用逼近方法来解决问题。

对于一个已给定的函数，可以用各种已知的函数来进行逼近，为了简化问题，一般都用简单函数作为逼近函数。所谓简单函数，通常是指可以用四则运算进行计算的函数，这种函数的一般形式是有理分式函数（即多项式的商），而其中最简单，也是最常用的逼近函数形式是多项式

$$P_n(x)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-2}x^2+a_{n-1}x+a_n \tag{1-2}$$

通常将多项式作为逼近函数的主要原因有：多项式是所有函数中研究得最多，各种性质研究得最透的函数；其计算误差易于控制；所有的函数几乎均可近似地展开为多项式。

逼近是数值计算中经常应用的概念与方法，它在插值、数值积分、常微分方程数值解、方程求根等方面均有广泛的应用。从某种意义上说，没有逼近也就没有数值计算。

1.2.3 递推

将一个复杂的计算过程转化为简单过程的多次重复的数学方法称为递推。把一个复杂的计算过程转化为一个简单过程，这种具体实施过程称为递推化。为了叙述方便，本书将需多次重复的简单过程称为递推结构或递推过程。在程序设计中，递推是用循环来实现的，循环是计算机在解决问题时最强有力的手段之一。从某种意义上说，解决复杂计算问题的关键可以归结为设计出或寻找到简单而又合理的递推结构，这也是算法的关键问题之一。现以设计计算多项式$P_n(x)$为例来说明递推。从不同的角度考虑，计算多项式$P_n(x)$的递推结构大体有如下三种。

1. 方案一

此种方案采用计算一项累加一项的递推结构。在计算每一个累加项时，其乘幂又采用累乘递推结构。从递推结构角度上看，本计算方案是一个递推结构中嵌套着另一个递推结构；从程序设计的角度看，是一个循环中嵌套着另一个循环的双重循环。这种递推结构的N-S流程

图如图1-1所示，其C语言程序如下所示:

```
#include<stdio.h>
{
    int i,j,n;
    float a,m,s=1.0;/*此处设常数项a_n=1.0*/
    printf("Input variables n and x=\n");/*提示输入*/
    scanf("%d%f",&n,&x);
    for(i=1;i<=n;i++)
        {
        printf("Enter variable a=\n");
        scanf("%f",&a);              /*输入系数值*/
        m=1.0;
        for(j=1;j<=i;j++)
            {
            m*=x;                    /*求乘幂*/
            }
        s=s+a*m;                     /*累加各项*/
        }
    printf("s=%f\n",s);
}
```

关于本程序的几点说明如下:

1) 注意，内循环的终值恰好是外循环当时的循环变量值。因此当$i=1$时，内循环正好求x的一次幂，$i=2$时，内循环求x的二次幂，以此类推，当$i=n$时，内循环求的是x^n。

2) 语句printf("Enter variable a=\n")和printf("Enter variable a=\n")的作用并不是输入变量值，而是提示紧跟下面的输入语句要输入什么变量，这可以增加程序的可读性，提高程序运行的效率。

3) /*……*/是C语言的注释内容，是程序的一个组成部分。不过它与程序的执行毫无关系，但加入注释可以大大地增加程序的可读性。

以上几点虽然不是程序设计的主要方面，但恰当应用以上技巧对于灵活进行程序设计，提高程序的可读性和程序的维护是有很大好处的。

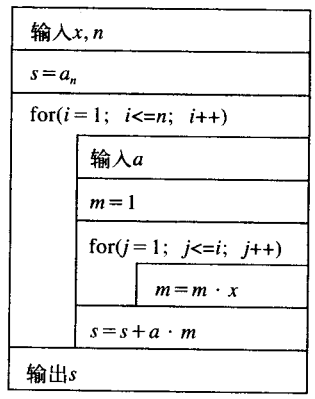


图1-1　方案一的流程图

2. 方案二

分析方案一中x乘幂的计算，每个x的乘幂都是从x的1次开始，一次一次乘x计算而得，计算无继承性，即前面计算的结果不能为后面的计算所利用。为了改进此计算过程，可以设想:

$$x^n=x^{n-1}\cdot x$$

即计算x^n时，若x^{n-1}已计算得出结果，则计算x^n时不必从x的1次开始，只要将x^{n-1}乘以x即可获得x^n。从递推的角度看，累乘和累加可以同时递推:

$$\begin{cases} t=t\cdot x \\ s=s+a\cdot t \end{cases}$$

这种递推结构的N-S流程图如图1-2所示，其C语言程序如下所示:

```
#include<stdio.h>
{
    int i,n;
    float a,t,s=1.0;/*设常数项a_n=1.0*/
    printf("Input variables n and x=\n");/*提示输入*/
    scanf("%d%f",&n,&x);
    for(i=1;i<=n;i++)
        {
        printf("Enter variable a=\n");
        scanf("%f",&a);                    /*输入系数值*/
        t=t*x;                             /*求乘幂*/
        s=s+a*t;
        }
    printf("s=%f\n",s);
}
```

在本程序中，请注意求乘幂的累乘语句t＝t*x和求多项式各项之和的累加语句s＝s＋a*t处于同一个循环中。

从框图和程序可以看出，方案二较方案一简明，且可节省计算工作量。因而方案二优于方案一。

3. 方案三

进一步的分析可以发现，方案二中的递推将累乘和累加在同一个循环中实现，且各自独立进行。进一步的思考提出，能否把累乘和累加结合在一起进行？于是就产生了第三种递推方案。这种递推方案是把多项式$P_n(x)$计算过程转化为:

$$P_n(x)=(\cdots((a_0\cdot x+a_1)\cdot x\cdots)\cdot x+a_n$$

这种计算方案的递推结构为前面的计算结果乘以x，加上一个新系数，如此递推下去，即可将多项式的值计算出来。这种递结构的N-S流程图如图1-3所示，其C语言程序如下所示。

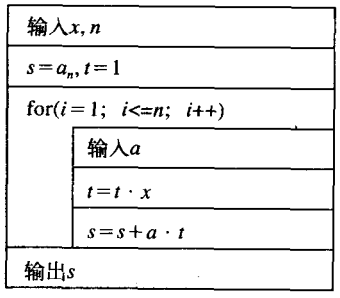


图1-2 方案二的流程图

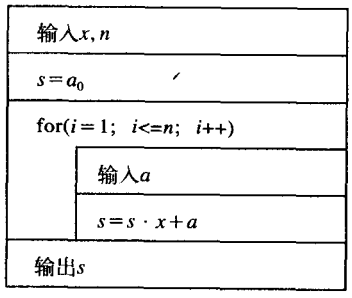


图1-3 方案三的流程图

```
#include<stdio.h>
{
    int i,n;
    float a,t,s=1.0;          /*设乘幂x^n的系数a_0=1.0*/
    printf("Input variables n and x=\n");/*提示输入*/
```

```
    scanf("%d%f",&n,&x);
    for(i=1;i<=n;i++)
        {
            printf("Enter variable a=\n");
            scanf("%f",&a);                    /*输入系数值*/
            s=s*x+a;
        }
    printf("s=%f",s);
}
```

在此程序中应注意的是，第一个送入s单元的值已不是a_n（常数项），而是a_0（最高次乘幂的系数），并且求乘幂的累乘与求各项和的累加在一个语句s＝s＊x＋a中实现。

通过以上计算多项式$P_n(x)$值的三种递推结构分析比较，可获得如下一些有用的结论:

1) 解决复杂计算问题的关键在于能否找到（或设计出）一个递推结构，例如，只要找到上面所提供的三种递推结构中的任意一个，就能计算出多项式$P_n(x)$，余下的问题仅是程序编码问题。

2) 一个计算问题的递推结构不一定是唯一的，通常是多种多样的。这就提示我们，当找到一种解决计算问题的递推结构时，不要停止思考，而应考虑是否还有别的更优的递推结构，即应继续思考。

3) 解决同一计算问题的递推结构也有优劣之分。上述所给出的计算多项式$P_n(x)$的三种方案，虽然都能正确地计算出多项式$P_n(x)$的值，但显然方案三最优，方案二次之，方案一最差。因而当我们对某一计算问题设计出递推结构时，对于能否计算出结果而言，可以暂告一段落，但从学习和研究角度看不能到此为止，而应当提出有没有别的递推结构能计算出正确的结果？这种递推结构是否是最优的？这样才能不断提高设计算法的能力。

1.2.4 常用基本递推结构

稍加注意计算机程序，即可发现这样一个事实：程序中存在着一些功能单一、结构相对稳定的小程序段。只要程序中某处需要完成某种功能，完成该种功能的相应小程序段便在该处出现（或被引用），本书为叙述方便，暂把这种功能单一、结构相对稳定的小程序段称之为常用基本递推结构，或称之为基本算法小模块。实际上，这些基本算法小模块也是离散、逼近、递推在程序中的综合运用。通过高级程序语言的学习，读者已接触了一些常用基本递推结构，如累加、累乘、换值、选极值、使用误差限控制计算等，而且数值计算程序基本上是这些常用基本递推结构复合而成。下面将这些常用的递推结构进行简要归纳，作为继续学习的基础。

1. 累加基本递推结构

累加基本递推结构的流程图如图1-4和图1-5所示。图1-4是累加递推基本结构的一般形式，其主要部分是累加单元在循环外赋初值（可以为0，也可以为一个具体数据，这要依据具体情况而定）。紧接进入循环，循环体为一个累加结构，读入一个数，将该数累加到和值单元（即累加单元）中去。凡是累加结构都有形式为

$$s=s+a$$

的累加语句，其中s为累加单元，各数据相加后的和值仍放置在该单元中。a为累加数。其基本操

作过程是输入一个数据，累加一个数据。如果所要加的数据已事先输入，则输入部分即可省去。

图1-5所示是一个计数基本递推结构，它是累加的一种特殊形式，图中的循环体是指程序要解决的问题所对应的程序段，计数语句$M=M+1$是与循环体共处于同一个循环中的一个语句，其中M单元即为计数单元，由流程图可知，每循环一次，循环体执行一次，M单元的值增加1，M单元把循环的次数记录了下来，此即称为计数基本递推结构的由来。为了叙述方便，后面本书将累加基本递推结构简称为累加。

需要说明的是，图1-4和图1-5所给出的流程图并不是一个完整的流程图，仅给出了累加部分的流程，以下给出的基本递推结构的流程图与累加相同，不再进行说明，请读者注意。

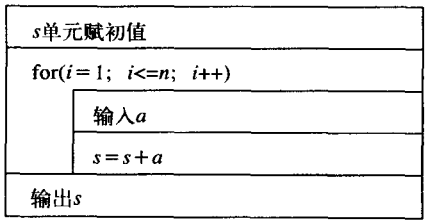


图1-4 累加基本结构

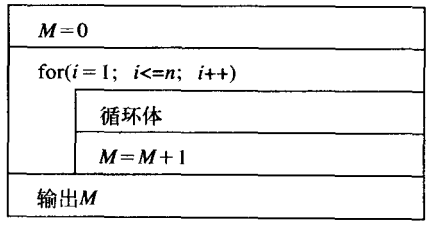


图1-5 计数基本结构

2. 累乘基本递推结构

累乘基本递推结构的流程图如图1-6、图1-7、图1-8所示。图1-6是一般的累乘基本递推结构形式，它的主要部分是输入一个数，就将该数累乘到乘积上。注意其中的“直到$a=1E+37$”，1E+37称为结束标志，1E+37是最后一个输入数，当a单元中读入1E+37后，循环就结束。使用结束标志的好处是不必清点共要输入多少个数据，依据多少个数据确定循环的终值，这里只要把结束标志作为最后一个数据就可以使循环正常结束。设置结束标志的值时应注意，该值绝对不能是所要处理数据范围内的数据对象。此外还要注意，累乘单元在循环体外必须赋初值1（或相乘数中的一个），否则就得不到所要的乘积。

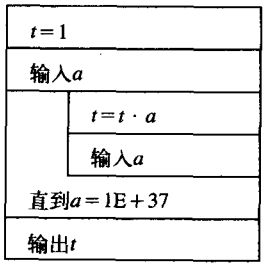


图1-6 累乘基本结构

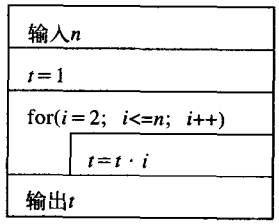


图1-7 阶乘基本结构

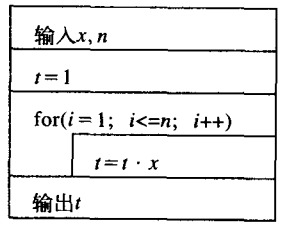


图1-8 乘幂基本结构

累乘基本递推结构还有两种经常使用的结构，图1-7所示是求取阶乘的基本递推结构，它的特点是所乘的数是自然数列，使用阶乘递推结构时要注意阶乘的终值不能太大，否则很容易造成溢出。图1-8所示是求取乘幂的基本递推结构，它的特点是所乘的数据是同一个值。累乘基本递推结构也是程序中经常应用的基本算法小模块，为了叙述方便，本书在后面将这三种结构分别简称为累乘、阶乘和乘幂。

3. 换值基本递推结构

换值是指将两个存储单元中的值相互交换。例如$x=5$，$y=10$，则换值的结果应该是