

科学程序设计引论

——用 Mathematica 和 C 求解计算问题

[美] Joseph L. Zachary 著

裘宗燕 李琦 李建国 译

高等教育出版社

Translation from the English language edition

Introduction to Scientific Programming by Joseph L. Zachary

Copyright 1998, Springer-Verlag New York, Inc.

Published by TELOS The Electronic Library of Science, Santa, California

TELOS is an imprint of Springer-Verlag New York, Inc

All Rights Reserved

图字：01 - 1999 - 0749 号

目 录

前言	(1)
1 计算科学	(1)
1.1 实验、理论和计算	(2)
1.2 解决计算问题	(3)
1.3 向前进	(7)
2 人口密度:数字的计算特性	(9)
2.1 模型	(10)
2.2 方法	(11)
2.3 实现	(11)
2.4 算术表达式	(11)
2.5 有理数	(16)
2.6 <i>Mathematica</i> 中的有理数	(17)
2.7 浮点数	(17)
2.8 <i>Mathematica</i> 中的浮点数	(22)
2.9 评价	(24)
2.10 关键概念	(26)
2.11 练习	(27)
3 厄拉多塞:有效数字及其区间运算	(29)
3.1 模型	(29)
3.2 方法	(30)
3.3 实现	(31)
3.4 实现评价	(33)
3.5 方法评价	(36)
3.6 模型评价	(38)
3.7 问题评价	(39)
3.8 关键概念	(40)
3.9 练习	(40)
4 通向天国之路:舍入误差的积累	(43)
4.1 归纳的模型	(44)
4.2 调和级数求和	(47)
4.3 舍入误差的积累	(48)
4.4 评价	(54)
4.5 关键概念	(57)

4.6	练习	(57)
5	基蒂霍克:自定义函数	(59)
5.1	模型	(59)
5.2	方法	(60)
5.3	实现	(61)
5.4	评价	(66)
5.5	关键概念	(68)
5.6	练习	(68)
6	婴儿潮:符号计算	(71)
6.1	单利率	(71)
6.2	复利率	(75)
6.3	连续利率	(78)
6.4	评价	(81)
6.5	关键概念	(83)
6.6	练习	(83)
7	弹道轨迹:科学的可视化	(87)
7.1	弹道运动	(88)
7.2	科学的可视化	(89)
7.3	运动函数	(90)
7.4	二维图	(94)
7.5	列表	(97)
7.6	多重曲线图	(98)
7.7	参数图	(99)
7.8	动画	(101)
7.9	关键概念	(104)
7.10	练习	(106)
8	莱特海湾的战斗:符号数学	(109)
8.1	固定轨迹	(110)
8.2	任意轨迹	(114)
8.3	阻力的作用	(117)
8.4	分段轨迹	(119)
8.5	最后评价	(122)
8.6	关键概念	(122)
8.7	练习	(123)
9	老麦克唐纳的奶牛:命令式程序设计	(125)
9.1	在 <i>Mathematica</i> 里求解方程	(128)
9.2	二分法	(131)
9.3	一个二分法函数	(134)
9.4	评价	(141)

9 5	关键概念	(144)
9 6	练习	(144)
10	C 语言导引	(147)
10 1	<i>Mathematica</i> 背景	(147)
10 2	C 语言背景	(148)
10 3	C 程序的例子	(148)
10 4	解释器和编译器	(150)
10 5	<i>Mathematica</i> 与 C 的差异	(152)
10 6	关于学习 C 语言	(154)
10 7	厄拉多塞问题	(154)
10 8	基蒂霍克问题	(157)
10 9	关键概念	(158)
10 10	练习	(158)
11	机器人举重:直接的程序	(161)
11 1	连接图的三角学	(162)
11 2	直接程序的组成部分	(164)
11 3	类型	(166)
11 4	表达式	(169)
11 5	简单语句	(171)
11 6	函数 Main	(172)
11 7	库	(173)
11 8	评价	(176)
11 9	关键概念	(177)
11 10	练习	(177)
12	滑块:条件与函数	(181)
12 1	没有摩擦的无限长斜坡	(182)
12 2	有摩擦的无限长斜坡	(185)
12 3	有摩擦的有限长斜坡	(188)
12 4	自定义函数	(190)
12 5	评价	(195)
12 6	关键概念	(196)
12 7	练习	(196)
13	圆杆堆垛:利用函数设计	(199)
13 1	分解问题	(200)
13 2	设计	(203)
13 3	实现	(206)
13 4	评价	(213)
13 5	关键概念	(213)
13 6	练习	(214)

14	牛顿法	(217)
14 1	牛顿法	(218)
14 2	牛顿法的实现	(220)
14 3	二分法实现	(227)
14 4	评价	(229)
14 5	关键概念	(232)
14 6	练习	(233)
15	瓦楞板:多文件程序	(235)
15 1	数值积分	(236)
15 2	矩形法	(238)
15 3	矩形法的实现	(241)
15 4	梯形法	(244)
15 5	梯形法的实现	(247)
15 6	多文件程序	(250)
15 7	矩形法和梯形法的比较	(255)
15 8	关键概念	(255)
15 9	练习	(256)
16	简谐振动:结构与抽象数据类型	(259)
16 1	复数根的牛顿法	(260)
16 2	重返圆杆堆垛问题	(262)
16 3	重返牛顿法	(267)
16 4	评价	(275)
16 5	关键概念	(277)
16 6	练习	(277)
17	杆中的热传导:数组	(281)
17 1	热流模型	(282)
17 2	有限元法	(285)
17 3	实现	(286)
17 4	评价	(297)
17 5	关键概念	(300)
17 6	练习	(300)
18	热传导的可视化:数组作为参数	(303)
18 1	数组作参数	(305)
18 2	文件输入	(309)
18 3	文件输出	(312)
18 4	评价	(318)
18 5	关键概念	(320)
18 6	练习	(320)
附录 A	Mathematica 的功能	(323)

A .1	单位	(323)
A .2	数学排版	(326)
A .3	浮点数仿真	(328)
A .4	任意精度数	(331)
A .5	<i>Mathematica</i> 和 C 一起使用	(335)
附录 B	<i>Mathematica</i> 函数和常数	(343)
附录 C	C 语言库函数	(349)
附录 D	使用 <i>Mathematica 2.2</i>	(351)
D .1	浮点数的语法	(351)
D .2	数学排版	(352)
D .3	特殊常数	(352)
D .4	符号功能	(354)
参考文献		(357)
索引		(359)

前 言

本书将教授科学与工程领域的大学新生如何去求解计算科学问题,这种问题是他们在未来的学术专业生涯中会经常遇到的.虽然本书讨论的是基本概念,但它实际上也提供了一个坚实的基础,有了这种基础,学生就能够继续他们的终身学习.本书还着力于避免通常科学与工程学生的程序设计引论课程里的两个重要缺陷.

首先,程序设计引论课常常忽略实际程序设计的背景情况.如果把目标确定为教学生如何写程序,那么引人之处就在于告诉学生写什么程序.很不幸的是,这意味着完成这种作业的人没有接触到工作中最有意思的部分:把真实世界的问题转化为程序问题.

这样,学生会慢慢地习惯于让别人去帮他们做这种创造性工作,到最后他们完全不知道如何使用自己的程序设计技能去解决在其他地方实际发生的问题.这样培养出来的学生将不会想到用计算去解决他们在科学和工程课里遇到的计算问题,除非课程明确要求他们这样做.即使在课程里有明确要求,许多学生也会是茫然不知所措,直到有人告诉他们应该写什么程序以及大概要怎样地去写.

其次,程序设计引论课通常是为计算机专业的学生设计的.即使要求计算的问题非常有意义,这些问题与一般的科学与工程学生往往也没有太大关系.更根本的问题是,关于程序设计的技术细节可能适合计算机专业的学生,而对于其他科学与工程专业的学生则并不总是合适的.

对浮点数的处理是最好的例证.常规计算机科学基础课几乎用不到浮点数,而科学程序设计则完全是另一种情况,没有浮点数甚至什么也不能做.计算机科学的学生可能要学到 *IEEE* 的浮点数表示标准,而一般科学与工程学生则应该知道浮点数的计算性质,以及它们在表示非精确的物理度量方面的作用.

本书的第1章和第10章包括了一些概述性的内容.除了这两章之外,其余每章的内容都围绕着解决一个科学或工程中的问题展开.每一章开始于对有关问题的描述,然后开发该问题的数学模型,设计一个解决这个模型的计算方法,基于该方法构造一个实现,最后再深入评价由这个实现产生的解答.

把本书的各部分联系在一起的线索是通过 *Mathematica* 和 C 表述的科学程序设计的精华.我非常仔细地在计算问题求解的广阔领域中展开这些本质性的东西.为此目的,我选择了一些题目,它们使我能够以一种比较自然的顺序引出这些最基本的程序设计概念,介绍它们并做细致的解释.虽然每章里都有不少篇幅用于讨论实现方面的问题,它们还是包含了许多关于问题描述、模型构造、方法开发和结果评价的内容,这就使实现问题被摆到了适当的位置.

程序设计语言

无论对初学者,还是对高级的程序员,都没有一种单一的最佳语言.程序设计语言不过是一种复杂的工具.技术高超的程序设计者就像一个机械师傅,有他们自己配置的一组工具.对程序设计语言的合理选择依赖于许多因素,如手中程序设计问题的性质,是否存在可以使用的类似问题的解,程序设计者的基础,以及他们工作中所用的计算环境等.

由于这些认识,我在本书第一部分使用 *Mathematica*,第二部分用的是 C. *Mathematica* 和 C 代表了在科学程序设计中广泛使用的两类工具.*Mathematica* 是一个计算机代数系统,而 C 是一种常规的程序设计语言.

Mathematica 是一个系统,用它能够做数值、符号和图形数学.它同时也是用在该系统里的一种程序设计语言的名字.我在书的第一部分选用 *Mathematica* 而不是 C 有许多原因.首先, *Mathematica* 比 C 更容易理解和使用.其次,在 *Mathematica* 里只要经过很少的训练就可以做许多有趣的事情.学生们很容易在自己的课程中发现 *Mathematica* 的直接应用;最后,学生在学习 *Mathematica* 时掌握的大部分基本概念都可以带到 C 的学习中去.在本书的后半部分将详细讨论这些问题.

Mathematica 并不是仅有的计算机代数系统.它主要竞争对手有 Maple 和 Macsyma 等.我也可以在书中使用 *Mathematica* 的地方使用这些系统.事实上,我已经出版过本书的另一个版本,其中用的就是 Maple.

C 是一种常规程序设计语言,它广泛使用于包括科学程序设计在内的各类程序设计工作中.虽然计算机代数系统,例如 *Mathematica*,有极强的能力,熟练使用常规程序设计语言仍然是计算机科学家的基本技能.常规程序设计语言已经有 40 多年的发展历史,存在着大量用例如 Fortran 或 C 写的程序,供重复使用.常规程序设计语言比计算机代数系统更灵活,对于大的应用而言它们的执行速度也更快.

实际中有许多常规程序设计语言,其中在科学程序设计方面使用最多的两个是 C 和 Fortran.这里选择了 C 而不是 Fortran,主要是因为我自己多用 C 写程序.当然我也可以用 Fortran 代替 C 写一本与本书类似的书.实际情况是, Fortran 和 C 之间类似的地方多于它们的差异,至少从科学程序设计的角度看情况就是这样.

虽然本书的特色是讨论两种程序设计语言,但它并不是由互不相干的两部分组成的.我用 *Mathematica* 讨论一些问题,如数的计算性质,用运算和函数构造表达式,函数库的开发利用,以及围绕着程序设计者定义的函数组织程序等.在关于 *Mathematica* 的一半中讨论的这些问题都将延续到关于 C 的一半里去.这意味着关于 C 的讨论不是从空白开始,我们将从关于 *Mathematica* 的讨论继续下去.

本书不企图覆盖 *Mathematica* 或者 C 的全部.*Mathematica* 是个巨大的系统,不能设想这样一本书能讨论完它的各个方面.而 C 则包含一些方面,它们或是与科学程序设计无关,或是非常简单,可以在任何 C 的参考书中找到,读者可以在学完本书后自己去了解.我的想法是在计算问题求解的情况下,展现出每种语言有代表性和有用的部分.

随着对本书的撰写进程,我逐渐认识到 *Mathematica* 的一些特别有用的特性,这些特性值得讨论,但它们又很难自然地放进正常叙述中.我没有采取把这些讨论强塞进本书主体的方式,而是把它们放在附录 A.有几个章节里提到这些论题,读者可以自己决定如何使用这个部分.

Mathematica 是 Wolfram 研究公司的产品,该公司销售在各种计算机上的 *Mathematica* 系统(包括专业版本和学生版本).书中程序符合该系统的 3.0 版,以及老些的 2.2 版.有不多的几个例外,附录 B 告诉读者,如果使用 2.2 版,程序中的哪些地方应做点修改.

C 编译程序可以从许多公司得到.本书里所有的 C 程序可以在任何满足 ANSI C 标准的编译程序上使用,几乎所有近 10 年开发的 C 编译程序都满足这个要求.实际上,对这些程序也可以使用任何符合 ANSI 标准 C++ 的编译程序,只要能用 ANSI C 的标准输入输出库.

知识基础

为能采用这里的讨论方式,即每章开始于提出一个问题,然后通过模型、方法、实现和评价去接近问题的解,我不得不对使用本书的学生的数学和科学基础做一些假设.我假定他们的水平是大学科学或工程专业的新生.

16 个问题中的 7 个牵涉到高中或大学的物理内容,涉及到质量、速度、加速度、力、引力、摩擦、拉力、弹簧和热.当提出这种物理问题时,我给出了对所涉及到的物理原理的一些说明.学生原来已经了解一些有关概念情况下才有帮助.

另外的 9 个问题属于数学方面的.这些问题,以及书中的所有模型,都要求一个适当的数学基础.这里使用了许多代数、几何和直角三角形的三角学.我还一次用到余弦定理,一次积分和三次微分.虽然不完全了解其中的模型,也可能理解由它们导出的实现.但学生最好至少具备基本微积分课程的预备知识.

书中的许多方法依赖于数值分析的基本概念,而有关实现则使用了 *Mathematica* 或者 C.这里并不假定学生在这些方面有任何先修知识.要成为一个成功的程序设计者,作为一个学生而言必须乐于去实践.如果讨厌去尝试某些事情直到完全弄懂它们,这样的学生永远不可能成功.

章节概览

这里对各章的以下方面作一概述:引出各章内容的问题,理解模型所需要的数学基础,进而引出的计算方法,开发一个求解问题的实现所需要的 *Mathematica* 和 C 语言的基础.

第 1 章探究科学的实验、理论和计算途径之间的差别.以第谷·布拉赫(Tycho Brahe)、约翰尼斯·开普勒(Johannes Kepler)和依萨克·牛顿(Isaac Newton)的工作作为例子.这里仔细地讨论了约翰尼斯·开普勒经过 5 年努力,通过计算确定火星的轨道.用这个例子说明计算问题求解的五步方法.这正是书中后面章节里使用的方

法。

第2章开始对 *Mathematica* 的讨论。这儿的问题是确定地球上的平均人口密度, 以此作为出发点学习如何使用算术表达式完成简单计算。本章大部分内容在讨论 *Mathematica* 中有理数和浮点数的差别。

第3章讨论古代昔勒尼(Cyrene)的厄拉多塞(Eratosthenes)在公元前225年测定地球周长时使用的简单实验和几何论证。由此引出对计算中由于物理测量的非精确性带来的问题。这里要解释如何使用有效数字和区间算术的方法对付非精确性数据。还讨论了变量和赋值的概念。

第4章引出了一个很有趣的问题: 在一个桌上摆起 n 块超出桌边的同样大小的砖块, 最多可以伸出桌边多远。这是本书中第一个不那么显而易见的模型。为开发这个模型, 我们使用了重心的概念。这章的许多内容是关于确定一系列不同 n 值的项的求和问题。这也使我能够讨论在精确的但(有时是非常)慢的有理数算术, 与快速的但(有时是灾难性的)不精确的浮点数计算之间的平衡折中。

第5章用毕达哥拉斯(Pythagoras)定理开发了一个模型和两个方法, 解决的问题是确定一个小山顶上看地平线的可见距离。第一个方法牵涉到减两个数, 它们的差异仅仅是在一些非常重要的数位上。由此导致计算结果有效数位的灾难性损失。第二个方法用到代数化简, 避免了第一个方法的问题。在基于这个方法的实现中提出了程序设计者自己定义函数的问题。

第6章研究的问题是预测美国人口的增长趋势。在第2-5章的讨论集中在数值计算, 而这里考虑的是符号计算程序。在这里用到 *Mathematica* 的符号计算函数, 如 Factor、Solve 和 Limit 等, 开发了一些简单的、复合的或连续的增长率模型, 并将它们应用到人口增长的问题上。

第7章用抛射体弹道轨道可视化的问题研究了 *Mathematica* 的 Plot 等函数。这章开发了一个模型, 解决由抛射体的初始速度分解出其垂直和水平分量。这是我们第一次使用三角学。这里还要解释如何生成二维可视材料, 包括图、参数图形和动画等。

第8章考查拉一条船沿着特定的一维轨道运动所需要的力。这需要开发一个函数, 它以一个符号轨迹表达式、船的质量和船的拉力系数作为参数, 生成一个功率消耗表达式作为结果。这个模型和它的实现都依赖于微分, 以便能够从船的位置得到速度和加速度。我用 *Mathematica* 的关系运算建立合取结构, 并使用它的 While 结构描述船的轨迹。

第9章提出了一个问题, 归结起来是要找到一个弧的半径, 该弧把一个圆分割为相等的两块。有关模型依赖于圆和三角形的性质, 并需要一点三角学。最后结果是一个很简单的超越方程, 无法得到符号形式的解, 这促使我们考虑使用二分方法。经过随后的几个步骤, 我们开发了一个实现该方法的 *Mathematica* 函数, 其中用到赋值、If、While 和 Module 表达式。我们的意图并不是深入展现 *Mathematica* 的命令式功能, 而是预示一下 C 程序设计的一些情况。

第10章是从 *Mathematica* 到 C 的转折点。到这里为止, 学生除学到了许多其他东西之外, 应该已经牢固地掌握了数的计算性质、面向表达式的函数式程序设计, 以及自定义函数的使用等等。C 语言当然也有数、表达式和自定义函数, 这意味着我们可

以从已有的基础上继续前进,开始讨论 C 中面向语句的过程式程序设计问题.本章列举了 *Mathematica* 和 C 的不同点,并在一个高的层次上研究了前面章节里某些 *Mathematica* 程序的 C 版本.

第 11 章描述了控制机器人的踝、膝和髋关节,使其能够做下蹲运动的问题.通过两个简化假设,我们开发了一个三角化的模型,其中机器人肩膀的位置完全由它的躯干长度和膝的角度确定.这个模型的 C 实现引出关于直接的程序的讨论,其中接触到 C 的数值类型、表达式、赋值语句、数学库和输入输出库等.

第 12 章用滑块在斜面上运动的计算问题,研究了 C 的 if 语句及其他相关结构.在这里讨论自定义函数的实现,引出了一些概念,如函数头部、函数体、函数原型和函数的运行时行为等.在这章开发的若干越来越具一般性的模型都基于三角学.

第 13 章的问题是确定摞起来的一些圆木的中心坐标.模型中大量使用了三角学.与本书中其他章里的三角学模型不同,这个模型用到非直角三角形以及余弦定理.这里没有介绍 C 的新东西,但是模型的实现说明了在程序设计、实现和测试中函数分解的作用.

第 14 章从确定靠在方块上一个杆的位置开始.虽然这个模型仅仅依赖于相似三角形的性质和某些代数运算,但得到的是一个 4 次多项式方程.我们开发和实现牛顿法去求解这个方程.在此同时介绍了 C 语言 while 和 do 循环的使用.

第 15 章的问题是确定要制成一张 10 m 长的瓦楞板所需钢板的长度.要解决这个问题,就要确定刻画了波纹的曲线的长度,这需要算一个积分.这个问题引出对数值积分的矩形法和梯形法的讨论,以及它们的实现.在实现中讨论了 for 循环的使用,将函数作为参数的可能性,以及把一个小程序划分成多个文件的优越性.

第 16 章提出的问题是确定滑块阻尼谐波振荡器的滑块的位置(作为时间的函数).这里并不想去建立有关的模型,而是指出要确定有关的解,就要找出一个二次方程的复根.随后说明了为什么牛顿法可以容易地推广到求复根的情况.牛顿法这个版本的实现中涉及到使用 C 语言中结构,及自定义类型和抽象数据类型的问题.

第 17 章假定一个热源作用于一根绝热银棒的一端,要求确定棒中心点的温度,作为时间的函数.我们从稳定状态下热传导的三个基本事实出发,根据热力学第一原理,开发了该问题的一个有限元模型.这个模型是纯代数的.随后的工作是解释怎样使用有限元方法求解这个模型.最后用 C 语言实现了这个方法.

第 18 章处理的问题是假定连续加热,完成第 17 章中温度情况的可视化.采用的方法是写一个 C 语言程序,它向一个文件中写入一系列 *Mathematica* 命令.而后,当这些命令被装入 *Mathematica* 系统时,就能生成我们需要的动画.本章作为这本书的结尾,还说明了没有一种工具能满足所有计算问题的需求.在这里的实现中还解释了如何把数组作为参数,以及如何从文件输入、向文件输出.

我们在本书中开发的绝大部分模型、方法和实现都不涉及除几何、代数及直角三角形的三角学之外的数学.这使科学和工程的新学生可以接受这本书,而又不影响我们展示整个问题求解过程的目的,不过,其中也确实有几个部分用到稍微深一点的数学:

第 8 章里由位置表达式出发,用微分去确定速度和加速度的符号表达式.在模型

和实现里都用到微分,这使得本章对那些没有开始学习微积分的学生是困难的.好在可以跳过这章,不会对整本书的连续性产生大影响.

第 13 章对非直角三角形使用余弦定理.不熟悉余弦定理的学生会弄不清楚模型是如何导出的.但这不会阻碍对主要内容的理解,这里关注的是 C 程序的设计、实现和调试.

第 15 章的要点是定义积分,但是把它叙述为纯几何问题:确定由曲线围起的面积.

第 14 章和第 16 章讨论了牛顿法,完整理解该方法的工作原理需要理解函数的一阶导数.而实现本身并不依赖于微分.

教学特征和辅助材料

本书有许多有助于学生学习的特色,并附有一张磁盘和有关网址.

在每章的最后都复习了本章中展开的关键概念.

除第 1 章而外,每章都带有一组内容广泛的练习.

除第 1 和第 10 章外,其余的章都用两种方式组织.首先是每章都像常规一样划分为节和小节.此外,在计算问题求解的五个步骤的转换点,书边上都用小图形标记.这些小图形的意义在第 1 章里解释.

书后有 4 个附录,其中包含了一些辅助信息.附录 A 讨论一些常用的 *Mathematica* 有趣功能,这些东西在正文里没有介绍.附录 B 是有关正文中使用的所有 *Mathematica* 函数的概述.附录 C 是正文中使用的所有 C 函数的概述.附录 D 讨论了如果在学习时使用的是 *Mathematica* 2.2 版本,必须对书中程序做什么改动.最后是建议进一步阅读的参考文献.

对有程序的各章,随书磁盘里有 *Mathematica* notebook 格式的所有代码.这些 notebook 中的代码可供学生直接使用,而不必自己重新键入.其中的说明文字是对书本的补充.磁盘里包含了 3.0 版和 2.2 版形式的 notebook.

书中某些地方用到我特别为本书建立的 *Mathematica* 库.有关库的实现以及它们使用的说明都(以 3.0 和 2.2 版的两种格式)包含在磁盘里.

磁盘包含了书里描述的每个 C 语言程序的完整实现.在许多情况下,书上列出的只是程序里特别有趣的片段.如果不这样做,一个长程序就会占好几页.磁盘里的程序是完整的,整理好的,并且包含解释性的注释.

两个网页中包含了许多在线材料,作为对书的补充.其中许多是非常有用的.例如学生的实验室工作.网页地址是:

<http://www.telospub.com/catalog/MATHEMATICA/IntroSciProgMma.html>

<http://www.cs.utah.edu/zachary/IntroSciProgMma.html>

这些材料包括一些 Java 小程序,基于 HTML 的教学材料,以及一些 *Mathematica* notebook.磁盘里还包含了若干例子.有关材料的开发是一个正在进行的项目,今后几年里还会不断积累.欢迎大家通过因特网使用这些在线材料,或者下载它们.

我的 email 地址是 zachary@cs.utah.edu.我将很高兴听到你的问题、评论和建议.

致 谢

本书的成型过程应当追溯到我与 Chris Johnson 从 1993 年开始的合作, 当时 Chris 引导我关心基础的计算科学教育问题. 我们一起设计了一个科学程序设计的引论课程, 开发了讲义和一些在线材料(这就是本书的雏形), 并第一次开设了这个课程. 从 1994 年开设以来, 这个课已经在犹他大学讲了 7 次. 开课之后, Chris 就一直鼓动我把讲义和在线材料整理成一本教科书. 如果没有从他那里得到的动力和见识, 我不可能开始做这件事. 如果没有他的帮助和鼓励, 我也不可能完成这个工作.

Eric Eide 和 Ken Parker 在 1993 年末到 1994 年初的讲义初稿和在线材料开发中起了重要作用. 初始课程使用 Maple、C 和 Fortran. Eric 写了当时讲义中许多关于 Fortran 和 C 的东西, 他文笔的准确和清晰令人佩服. Ken 开发了有关 Maple 的初始材料, 提出了后来放进讲义的许多问题. 我把其中的不少问题放进了本书. 在后来的日子里, Ken 还回答了我许许多多的关于物理、数学、以及数值分析方面的问题.

“本科生计算工程与科学”(The Undergraduate Computational Engineering and Science, UCES)项目是能源系通过使用计算改革本科生教学计划的一个创意. 1993 年起我成为这个项目的特许成员. 该项目后来发展到有大约 50 位教育家和科学家参加. 这个项目为我提供了丰富的专业信息, UCES 项目组的成员们对本书也有不少贡献. 在一个面向问题的科学程序设计课程中同时使用一个符号代数系统和一个常规程序设计语言的想法就是该项目的第一次会议上提出的. 我在本书中始终使用的解决计算问题的五步策略则是在后来的一次会议上提出的.

Tom Marchioro 是 UCES 项目的执行主任. 他从一开始就支持和鼓励本书的工作. UCES 项目的创始人 Jim Corones, Tom Marchioro 之前的项目执行主任 Dave Martin, 以及 Tom 都为原始的课程讲义和在线材料开发提供了资助. 我希望他们的投资得到了回报.

Judy Zachary 阅读了作为本书雏形的基于 Maple 的许多章节的早期草稿, 并对改进其技术内容和表现形式提出了许多建议. 其他评阅人的意见主要是整体性的. Judy 的建议与他们互为补充. 此外, Judy 还是我的许多初步想法的公正评判者.

Andre Weideman 在 1996 - 97 学年中访问了犹他大学, 用这本书的 Maple 版本作为教材两次讲授课程. 他提出了一些改进有关数值分析的处理和练习的建议. 和他共事令我非常愉快.

本书的评阅人, Alkis Akritas、Tom Marchioro、Vaidy Sunderam、Bob White 和 John Ziebarth 提出了一些很有价值的建议. 他们关于加强 *Mathematica* 部分的建议特别重要.

Frank Stenger 最早在犹他大学的计算机引论课程中使用计算机符号代数系统. 当我对某些问题缺乏想法时就曾去找他, 他给我提了两个建议.

Wolfram 研究公司的 Paul Wellin 回答了我一些关于 *Mathematica* 的问题, 并寄给我几个不同版本的 *Mathematica* 系统.

从国家科学基金的本科生教育分部得到的一个“课程和教学计划开发”项目资助

支持了我们为本科生计算科学教育开发在线材料的工作,特别是支持了本书的写作.

感谢犹他大学计算机系的学生、教师和职员,是他们的全体使我在教学生涯的第一个十年里极富成果.

Joe Zachary
Salt Lake City
April 11, 1997

计算科学

第谷·布拉赫(1546—1601)是丹麦天文学家。在丹麦国王资助下,第谷在丹麦的 Hven 岛上建立了文艺复兴时期欧洲的第一个天文台。虽然直到他死后八年才发明望远镜,第谷还是能想出办法精确地测量天体位置,其误差只有一度的 $1/30$ 到 $1/60$ 。为了完成这项壮举,第谷发明了一系列仪器,包括六分仪(1569)、墙象限仪(1582)以及可携带的浑天仪(1591)等。在二十多年的时间里,第谷一直潜心于系统地测量和记录太阳、月亮和行星的位置。

约翰尼斯·开普勒(1571—1630)是德国天文学家。在第谷生命的最后一年半里开普勒做了他的助手。第谷去世后,开普勒拥有了第谷用毕生精力积累起来的天文观测数据。开普勒有自己的目标:发现行星运动背后的一般规律。为了找到隐藏在第谷的数据中的规律,开普勒经过了许多年努力,做了无穷无尽的计算,最后他发表了下面关于行星运动的三大定律,前两条发表于 1609 年,最后一条发表于 1619 年:

1. 行星的运行轨道是个椭圆,太阳位于椭圆的一个焦点上;
2. 连接行星和太阳的线段在等时间内扫过的面积相等;
3. 行星轨道周期的平方与它到太阳的平均距离的立方成正比。

依萨克·牛顿(1642—1727)是英国物理学家。牛顿一生的建树很多,包括微积分的发明,还有他发现的著名的运动定律和万有引力定律。牛顿有如此深刻的见识,他看到这些定律以同样形式适用于一切物体,无论是从树上落下的苹果,还是在轨道上运行的行星。虽然开普勒由第谷的数据中得到他的定律,但他并没有去解释为什么行星的运动遵循这些定律,牛顿的定律则提供了这个解释。事实上,开普勒的定律可以

从牛顿定律中导出,作为它们的推论。

采用现代的语言,我们可以说:第谷是实验科学家,牛顿是理论科学家,而开普勒则是计算科学家。今天,有许多科学家在使用这些词描述他们的工作,例如,实验化学家,理论物理学家,计算流体力学家等等。作为现代科学技术基础的就是对科学实验的、理论的和计算的研究。

1.1 实验、理论和计算

通向科学的这三种研究不可能完全划清界限,科学家和工程师的工作明显有一些重叠。第谷、牛顿和开普勒在自己的工作中都做了许多实验、理论和计算工作。

实验科学家的工作是观察自然界如何运转。第谷的实验包括设计、制造和使用六分仪、象限仪以及浑天仪等,完成了具有史无前例的精确性的天文观测。今天的实验科学家设计、制造并使用超级对撞机来测量高能亚原子粒子反应,设计、制造并使用“宇宙背景辐射探索者”卫星来观察宇宙的背景微波辐射。

理论科学家用数学语言去解释和预测自然的行为。牛顿运动定律解释了物体在外力作用下如何运动,他的万有引力定律解释了万有引力与质量和距离的关系。与此类似,今天的理论科学家提出了一些理论,解释高能亚原子粒子如何相互作用,解释在大爆炸之后宇宙的演化过程等。

理论通常要求得到实验的支持,而实验也常常是为了去验证理论。例如,大爆炸理论说宇宙的今天是经过数以十亿年计的时间演化而来的,其初始形式是无法想象的稠密和能量集中的亚原子粒子。按照这个理论,在宇宙开始时最关键的几百微秒中发生的一些高能反应与人们在高能超级对撞机中观察到的反应一样。另一方面,大爆炸学说的一个观点是宇宙背景辐射是每分钟都在变化的,这个预言已经被宇宙背景辐射探索者卫星观察到了。

计算科学家使用实验和理论知识,设法为自然界的某些方面建立基于计算机的模型,然后用这种模型去模拟自然界的行爲,并设法去建立模拟中观察到的现象与自然界之间的联系。从某种角度看,计算科学家是在建立自然界的模拟翻版,以此来作他们的实验。物理实验有时可能太困难、太昂贵,或者根本无法做,这正是科学的计算研究的用武之地。

计算科学的工作有时会引出新的理论或者新的实验,开普勒关于行星轨道的计算工作就是这样。开始时开普勒相信行星的轨道是圆形的,通过对第谷测量数据的大量计算,他终于认识事实并非如此,进而发现了行星运动的三大定律。

计算科学也可以用来检验理论。例如,计算科学家可以利用有关的理论和实验知识,按照大爆炸刚刚开始的情况在计算机上建立一个宇宙的模拟模型。运行这个模拟系统,观察其演变过程。如果这个大爆炸模拟产生的宇宙不像真实宇宙,这就可能是一个证据,说明作为模拟基础的理论有问题。

计算科学最重要的用途,也许是如果没有计算机那些建立好的理论就无法应用。例如,用计算机模拟预测天气的变化,估计地震对一个地区的可能破坏情况,评价一种新飞行器设计的空气动力性质等。