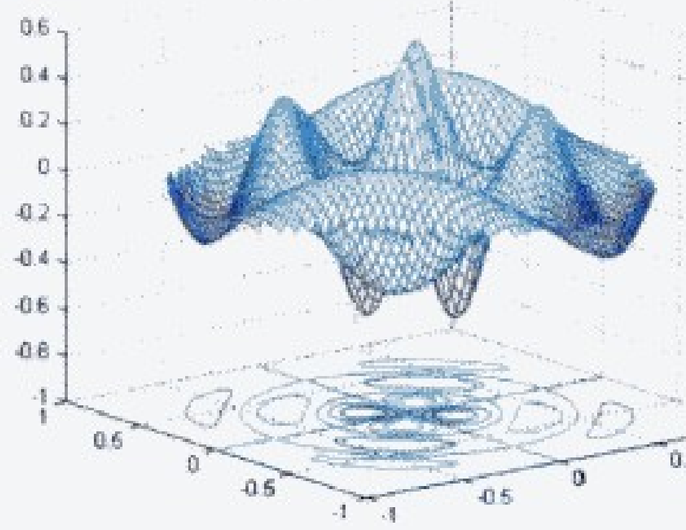
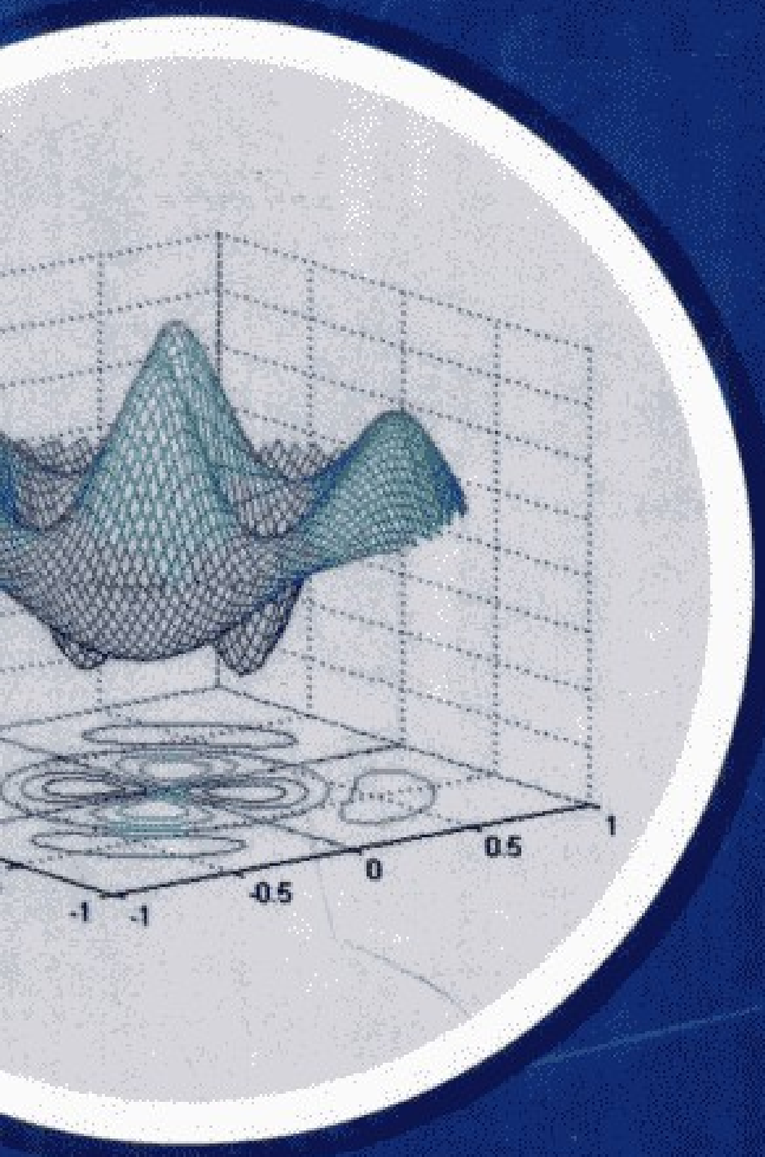
普通高等教育"十一五"国家级规划教材

计算物理基础



◎ 彭芳麟

高等教育出版社
Higher Education Press



ISBN 978-7-04-028355-6

9 787040 283556 >

定价 33.60 元

普通高等教育“十一五”国家级规划教材

计算物理基础

Jisuan Wuli Jichu

彭芳麟

高等教育出版社·北京
HIGHER EDUCATION PRESS BEIJING

内容提要

本书以计算软件 MATLAB 为编程语言，通过实例讲解了计算物理基础知识，如：数据拟合，数值微积分，解常微分方程与偏微分方程，蒙特卡洛方法等。选取的内容基本上遵照了教育部物理学与天文学教学指导委员会所建议的“计算物理基础”教学规范的要求并略有增加。

本书在内容编排与讲解方法上有独特的风格，突出了计算物理是编程语言学习、物理建模、计算方法应用以及用物理思想分析计算结果等四位一体的训练。着重强调了学习编程思路和训练编程技巧，重视实践操作，还增加了物理研究中很有用的可视化技巧如矢量场的可视化、动画技巧和图形界面的设计等。所选用的例子既有科研中的经典案例如混沌、分形等，也有教学中的疑难问题如陀螺、电磁场问题等，全部例子都附有参考程序，以及程序运行所得的图形，使版面显得生动，有利于提高初学者的兴趣。

本书是国家精品课程“计算物理基础”所采用的教材，其中的教学成果曾获 2005 年国家级教学成果二等奖，它反映了我们多年来教改的成果，也是师生之间能者为师，教学相长的见证。本书不仅适合于作基础课的教材，也可用于自学计算物理。对于需要使用计算物理作为工具的科研人员，也是一本很有价值的参考书。

图书在版编目（CIP）数据

计算物理基础 / 彭芳麟. —北京：高等教育出版社，2010. 1

ISBN 978-7-04-028355-6

Ⅰ. 计… Ⅱ. 彭… Ⅲ. 计算物理学-高等学校-教材 Ⅳ. O411.1

中国版本图书馆 CIP 数据核字（2009）第 199184 号

出版发行	高等教育出版社	购书热线	010－58581118
社　　址	北京市西城区德外大街 4 号	咨询电话	400－810－0598
邮政编码	100120	网　　址	http://www.hep.edu.cn
总　　机	010–58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	北京四季青印刷厂	畅想教育	http://www.widedu.com
开　　本	787×960　1/16	版　　次	2010 年 1 月第 1 版
印　　张	28. 75	印　　次	2010 年 1 月第 1 次印刷
字　　数	540 000	定　　价	33. 60 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

版权所有　侵权必究

物料号　28355－00

前　言

自从2004年教育部物理学与天文学教学指导委员会建议开设本科生必修基础课“计算物理基础”以来，我们积极地响应并付诸实践。此前，我们已经为研究生开设了计算物理课，并有一些本科生旁听该门课程。在此基础上，通过检索国外大学相同课程的教学内容与教材，参照教学指导委员会提出的《计算物理基础课程教学规范》，经过数年教学与教改的实践，编写了这本教材。期间我们还出版了《理论力学计算机模拟》和《数学物理方程的MATLAB解法与可视化》两本书，我们的教改成果曾获2005年国家级教学成果二等奖等多种奖项，“计算物理基础”也被评为国家精品课。这几本书记录了我们的教改实践，是师生互教互学，能者为师，共同提高的见证。北京师范大学物理系历届听过此课的学生们为建设这门课程作出了创造性的贡献。唐朝大文学家韩愈在《师说》中说过，“弟子不必不如师，师不必贤于弟子。闻道有先后，术业有专攻，如是而已”。我们对此深有体会。

计算物理借助于计算机来做计算与研究。它涉及建立模型，选择计算方法，用语言编程以及结果分析。这个系统过程缺一不可，只有掌握了这四个方面的技能。才能独立地完成任务。过去这些训练分散在不同的课程中，不利于学生掌握使用这种方法。所以Steven E. Konnin说：“进行计算的本领很难通过标准的大学生物理系课程安排来培养，因为它需要三方面训练（物理学，数值分析和计算机程序设计）的综合，很少有物理系学生毕业时知道如何进行计算；那些具有这种知识的学生通常是在独立工作的过程中，比如通过学习一项研究课题或者一篇毕业论文，而学到一些有限的技巧。”[①]我们赞同这种看法，在编写本书时特别注意这几种技能的综合训练。

本教材定位于普及计算物理基础知识，内容包括数据处理（拟合，插值，快速傅里叶变换等），数值微积分，解线性与非线性方程，解常微分方程与偏微分方程，蒙特卡洛方法等，都是最基本的计算技巧。此外还增加了物理研究中很有用的可视化技巧，如矢量场的可视化，动画技巧和图形界面的设计等。众所周知，计算已经与理论、实验并列为物理研究的主要手段。正如不学微积分是无法学习物理一样，在今天如果不会用计算机做计算也很难真正学好物理。更何况通过学习计算物理，还能学到与其他课程不同的分析方法与研究方法。所以，教学指导委员会才会把计算物理课

① Steven E. Koonin. 计算物理学. 秦克诚，译. 北京：高等教育出版社，1992

列为本科生必修课。

万事开头难，初学者如何迅速入门，是本书最关切的问题。作者深信，只要能入门，深造就有可能。为此，本书选择MATLAB作为编程语言，因为它具有“入门就能用”的特点，同时强大的计算功能与全面的可视化功能也使我们爱不释手。它解决了本课程课时少，内容多的困难。这不是唯一可选择的编程语言，但肯定是一个很好的选择。只要掌握了这种语言，学习其他语言也能触类旁通。我们发现，用软件教学在国内外都已经成为趋势。

教材中不仅介绍数值计算方法，而且都附有物理应用的实例程序。提供参考程序是为了使初学者少走弯路。教材强调了编程技巧的训练，一个物理问题常常会有几个程序，有的是计算方法不同，有的是编程思想不同，通过比较其中的优劣，使学生体会到，不仅要能够解决问题，还要提高解决问题的效率。教材中的例子既有科研中的经典案例如混沌、分形等，也有教学中的疑难问题如陀螺、电磁场问题等，这些问题的物理模型大多是学生所熟悉的，所缺乏的只是数值计算。本教材中的计算正好弥补了这个缺憾。

国外的同类教材有几种不同的编写方法，有的是按数值计算方法顺序讲述，如Steven E. Koonin的《Computational Physics》，Tao Pang的《An Introduction to Computational Physics》；也有的是按物理问题顺序讲述，如Harvey Gould等的《An Introduction to Computer Simulation Methods》；有的虽然是按物理问题顺序讲述，但对用到的数值计算方法是用附录的方式进行简介，如Nicholas J. Giordano等的《Computational Physics》。不同的编写方法有不同的优点，我们已经出版的两本书《理论力学计算机模拟》与《数学物理方程的MATLAB解法与可视化》就是讲常微分和偏微分方程在物理问题中的解法，本书则是按数值计算方法的顺序编排，目的是让初学者能对基本的数值方法有个系统的了解。

本书约需30个学时讲授，上机也需30个学时。使用本教材有较大的灵活性，如果觉得课时紧张，有些扩充性的内容可略去不讲，例如图形界面的制作、曲线拟合工具箱、偏微分工具箱、用指令bvp4c解边值问题与本征值问题等。还有一些物理应用的实例也不一定要在课堂上讲授，这些内容可留给学生课后阅读。

计算物理的起源及发展在张锁春研究员的《计算物理—科学计算—战略计算》[①]一文中有介绍。而在《中国计算物理学会发展20年》[②]一书中则记录了国内计算物理的发展历程。

感谢多年来支持与帮助作者的同事与同行，感谢高等教育出版社及编辑高

① 张锁春. 计算物理—科学计算—战略计算. 计算数学通讯，2000 (2)：2—6

② 张锁春. 中国计算物理学会发展20年. 香港：博士苑出版社，2004

健、李茜、庞永江等为此书所付出辛勤劳动。最后作者还要特别感谢北京大学第一医院院长刘玉村教授及其助手，是他们精湛的医术与高尚的医德帮助身患癌症的我重返工作岗位并能完成此书。

希望本书能为计算物理的基础教学作些铺路的工作，限于自身的水平，书中错漏之处在所难免，敬请读者指正。

彭芳麟

2009年10月

目　　录

第一章　MATLAB简介 ····· **1**

§1.1　MATLAB的操作界面 ····· 1

§1.1.1　操作界面介绍 ····· 1

§1.1.2　指令窗中的功能 ····· 2

§1.1.3　数据存储与显示 ····· 10

§1.2　数据格式与算符 ····· 13

§1.2.1　向量的输入 ····· 13

§1.2.2　矩阵 ····· 14

§1.2.3　列阵 ····· 19

§1.2.4　数据网格 ····· 20

§1.2.5　基元列阵 ····· 22

§1.2.6　结构数组 ····· 22

§1.2.7　字符和文本 ····· 23

§1.3　编程 ····· 24

§1.3.1　编辑程序 ····· 24

§1.3.2　调试程序 ····· 25

§1.3.3　设置搜索路径 ····· 27

§1.3.4　两类程序文件 ····· 28

§1.3.5　流程控制 ····· 31

§1.3.6　数据输入与输出 ····· 38

§1.4　作图 ····· 39

§1.4.1　作图功能概述 ····· 39

§1.4.2　二维曲线作图指令 ····· 40

§1.4.3　复数作图 ····· 42

§1.4.4　基本的三维图 ····· 43

§1.4.5　四维数据的表现——切片函数slice ····· 47

§1.4.6　复变函数图形 ····· 48

§1.4.7　动画 ····· 50

§1.5　画物理场的专用指令 ····· 55

§1.5.1　等值线表现二维标量场 ····· 55

§1.5.2　等值面表现三维标量场 ····· 56

§1.5.3　用箭头、流线表现二维矢量场 ····· 57

§1.5.4 用箭头、流线、锥体、流管、流带表现三维矢量场 ····· 59
§1.6 图形窗口的编辑功能 ····· 66
§1.7 制作图形用户界面 (GUI) ····· 69
§1.8 浮点运算与数值计算的误差 ····· 83
§1.9 物理应用：Logistic模型的周期分岔与混沌现象 ····· 87
§1.10 练习 ····· 94

第二章 迭代–分形图形 ····· 99
§2.1 迭代与分形 ····· 100
§2.1.1 分形树 ····· 100
§2.1.2 Sierpinski三角形 ····· 109
§2.1.3 科赫雪花曲线 ····· 116
§2.1.4 L系统 ····· 119
§2.2 复变函数迭代 ····· 120
§2.2.1 Julia集 ····· 120
§2.2.2 Mandelbrot集 ····· 124
§2.3 分形与分形维 ····· 126
§2.4 练习 ····· 132

第三章 数值微分与数值积分 ····· 134
§3.1 数值微分与数值积分 ····· 134
§3.1.1 数值微分的算法 ····· 134
§3.1.2 数值积分的算法 ····· 135
§3.2 MATLAB指令 ····· 138
§3.2.1 差分运算(diff) ····· 138
§3.2.2 梯度计算(gradient) ····· 138
§3.2.3 离散拉普拉斯算符(del2) ····· 139
§3.2.4 梯形积分 (trapz) ····· 140
§3.2.5 累计梯形积分 (cumtrapz) ····· 140
§3.2.6 函数积分 (quad, quadl) ····· 141
§3.2.7 函数的二重积分 (dblquad) ····· 142
§3.2.8 函数的三重积分(triplequad) ····· 142
§3.3 用符号工具箱计算微积分 ····· 143
§3.4 环形电流的磁场——物理场的可视化 ····· 144
§3.5 分子振动的半经典量子化 ····· 147
§3.6 练习 ····· 151

第四章　数据处理 ····· **154**
§4.1　插值 ····· 154
§4.1.1　插值分类 ····· 154
§4.1.2　拉格朗日插值法 ····· 155
§4.1.3　分段三次埃尔米特插值 ····· 158
§4.1.4　MATLAB插值指令 ····· 159
§4.2　曲线拟合 ····· 161
§4.2.1　曲线拟合的最小二乘法 ····· 161
§4.2.2　多项式拟合 ····· 162
§4.2.3　多项式拟合的指令polyfit ····· 163
§4.2.4　图形窗口的曲线拟合功能 ····· 164
§4.2.5　指数拟合 ····· 166
§4.2.6　线性最小二乘法的一般形式 ····· 167
§4.3　曲线拟合工具箱 ····· 168
§4.4　求方程的零点 ····· 176
§4.4.1　求单调连续函数$f(x)$的实数根的算法 ····· 176
§4.4.2　MATLAB解方程的指令 ····· 178
§4.4.3　搜寻函数的极小值 ····· 181
§4.5　快速傅里叶变换 ····· 183
§4.5.1　离散傅里叶变换 ····· 183
§4.5.2　傅里叶级数与傅里叶积分 ····· 185
§4.5.3　快速傅里叶变换的MATLAB指令 ····· 187
§4.5.4　快速傅里叶变换的算法 ····· 194
§4.6　练习 ····· 195

第五章　解常微分方程 ····· **198**
§5.1　龙格–库塔法 ····· 198
§5.1.1　基本思想 ····· 198
§5.1.2　二阶龙格–库塔法 ····· 200
§5.1.3　三阶与四阶龙格–库塔法 ····· 201
§5.2　变步长的龙格–库塔法 ····· 202
§5.3　常微分方程组的初值问题 ····· 203
§5.4　用MATLAB解常微分方程 ····· 205
§5.5　刚性问题 ····· 213
§5.6　事件 ····· 216
§5.7　误差 ····· 220
§5.8　性能 ····· 224

§5.9 物理应用 ····· 225
§5.9.1 刚体绕瞬心的转动方程 ····· 225
§5.9.2 弹簧摆运动 ····· 234
§5.9.3 圆锥陀螺运动 ····· 237
§5.10 边值问题和本征值 ····· 243
§5.10.1 边值问题的直接积分 ····· 243
§5.10.2 打靶法求弦振动方程本征值 ····· 246
§5.10.3 一维薛定谔方程的定态解 ····· 247
§5.11 用指令bvp4c解边值问题与本征值问题 ····· 250
§5.11.1 用指令bvp4c解一维本征值问题 ····· 251
§5.11.2 有两个解的边值问题 ····· 254
§5.11.3 马蒂厄方程的本征值 ····· 256
§5.11.4 艾登方程 ····· 258
§5.11.5 Falkner-Skan边值问题 ····· 260
§5.11.6 在$x=0$处有突变的问题 ····· 262
§5.12 练习 ····· 264

第六章 混沌 ····· 268
§6.1 单摆——从周期运动到混沌 ····· 268
§6.1.1 单摆的动力学方程 ····· 268
§6.1.2 周期运动 ····· 270
§6.1.3 有阻尼有驱动的情况——耗散系的混沌 ····· 283
§6.2 倒摆与达芬方程 ····· 290
§6.2.1 倒摆的运动方程 ····· 290
§6.2.2 倒摆的混沌运动 ····· 291
§6.3 自激振动——范德波尔方程 ····· 298
§6.3.1 运动方程 ····· 298
§6.3.2 VDP方程通向混沌的道路 ····· 300
§6.3.3 吸引子类型及其频谱 ····· 305
§6.3.4 分岔 ····· 309
§6.4 洛伦茨方程——奇怪吸引子 ····· 315
§6.4.1 倍周期窗与费根鲍姆数 ····· 324
§6.4.2 由阵发通向混沌 ····· 325
§6.4.3 庞加莱截面图 ····· 327
§6.5 练习 ····· 328

第七章　解偏微分方程 ····· **330**
§7.1　差分法解热传导方程 ····· 330
§7.1.1　显式差分公式 ····· 330
§7.1.2　隐式公式与平均隐式公式 ····· 335
§7.2　差分法解弦振动方程 ····· 338
§7.2.1　显式格式 ····· 339
§7.2.2　初始条件 ····· 339
§7.2.3　实例 ····· 340
§7.3　差分法与松弛法解椭圆型方程 ····· 344
§7.3.1　显式差分公式 ····· 344
§7.3.2　边界条件 ····· 345
§7.3.3　迭代法与松弛法 ····· 346
§7.4　偏微分方程的工具箱（PDETOOL） ····· 349
§7.4.1　PDETOOL的功能演示 ····· 350
§7.4.2　PDETOOL中方程的输入格式 ····· 351
§7.4.3　边界条件 ····· 352
§7.4.4　可解问题的分类 ····· 352
§7.4.5　解题步骤 ····· 353
§7.4.6　解一维与三维问题 ····· 353
§7.4.7　拉普拉斯方程与泊松方程 ····· 354
§7.4.8　热传导方程 ····· 356
§7.4.9　波动方程 ····· 358
§7.4.10　平面区域的本征值问题 ····· 363
§7.5　特殊函数 ····· 371
§7.5.1　勒让德函数 ····· 373
§7.5.2　贝塞尔函数 ····· 376
§7.5.3　调用MAPLE计算特殊函数 ····· 379
§7.5.4　平面波展开 ····· 386
§7.5.5　环形电流的磁场 ····· 391
§7.5.6　高斯积分 ····· 399
§7.6　练习 ····· 402

第八章　蒙特卡罗方法 ····· **404**
§8.1　蒙特卡罗方法的发展过程 ····· 404
§8.2　随机变量、密度函数与分布函数 ····· 406
§8.3　大数定理与中心极限定理 ····· 408
§8.4　随机数与随机抽样 ····· 412

§8.4.1 随机数的产生 ····· 413
§8.4.2 随机抽样 ····· 413
§8.5 计算定积分 ····· 417
§8.6 热力学的平衡态 ····· 422
§8.7 麦克斯韦速率分布律 ····· 424
§8.8 链式反应的模拟 ····· 426
§8.9 迭代函数系统 (IFS) ····· 429
§8.10 分形生长模型 ····· 435
§8.10.1 抛射沉积模型 ····· 435
§8.10.2 森林和薄膜的关联生长 ····· 436
§8.10.3 DLA ····· 438
§8.11 练习 ····· 440

参考文献 ····· 443

第一章　MATLAB简介

MATLAB是Matrix Laboratory（矩阵工作室）的缩写,是一个功能强大、界面友好的优秀数学软件；也称之为科学技术计算语言（the language of technical computing）。每年MATLAB都会推出新的版本，本书应用的是MATLAB7.3的版本。MATLAB易学好用，可以使非专业编程人员的物理工作者花费较少的时间也能编出高质量的程序，从而有更多时间与精力用到物理问题的研究上面。

本书是使用MATLAB作为编程语言讲授计算物理基础,为了方便没有学过MATLAB 的本科生使用。本章先介绍一些在本书中用到的MATLAB基本知识，其内容限制在6课时左右。对学过MATLAB的读者，可以跳过本章，直接阅读其后的章节，想深入地学习研究MATLAB知识可以参考其他书籍。

微分与积分、矩阵运算、解常微分方程与偏微分方程是高等数学与数学物理方法的基本内容，是学习理论物理的数学基础，没有这些知识是不可能学习理论物理的。用计算机去计算微分、积分与矩阵，解常微分方程与偏微分方程，可以提高计算能力与计算效率，这成了计算物理的基本内容。由于计算机的强大计算功能，使用随机数作研究也是计算物理的基本内容。可以肯定，今后计算物理的用途会越来越多。所以，在理论、实验和科学计算已经在物理研究中成三足鼎立的时代，不会用计算机作计算，必有诸多不便之处。

用MATLAB完成这些计算，必须掌握MATLAB的计算功能、编程方法与作图技巧。本章将对这三个方面加以介绍。

§1.1　MATLAB的操作界面

§1.1.1　操作界面介绍

安装MATLAB以后，在WINDOWS的窗口会出现MATLAB的图标，用鼠标双击MATLAB的图标将看到如图1.1所示的MATLAB的操作界面(Desktop)。

在操作界面上有四个窗口（图中用圆圈标出），分述如下：

Command Window　　指令窗口。是进行计算、运行指令与程序的基本工作环境。

Current Directory　　当前工作目录。其功能相当于目录管理器。

Command History　　自动记录用过的指令。用鼠标可以复制、删除或运

	行这些指令。
Workspace	工作内存，显示当前在内存中存储的变量，可用矩阵编辑器编辑它们。

图1.1中显现有三个窗口，隐藏了一个窗口（工作内存窗口），点击它的名称可以将它显现出来。菜单Desktop是对窗口进行管理的各种指令，例如只要显现指令窗口，用鼠标依次逐层点击菜单Desktop/Desktop Layout/Command Window only。如要恢复默认设置则依次逐层点击菜单Desktop/Desktop Layout/Default。

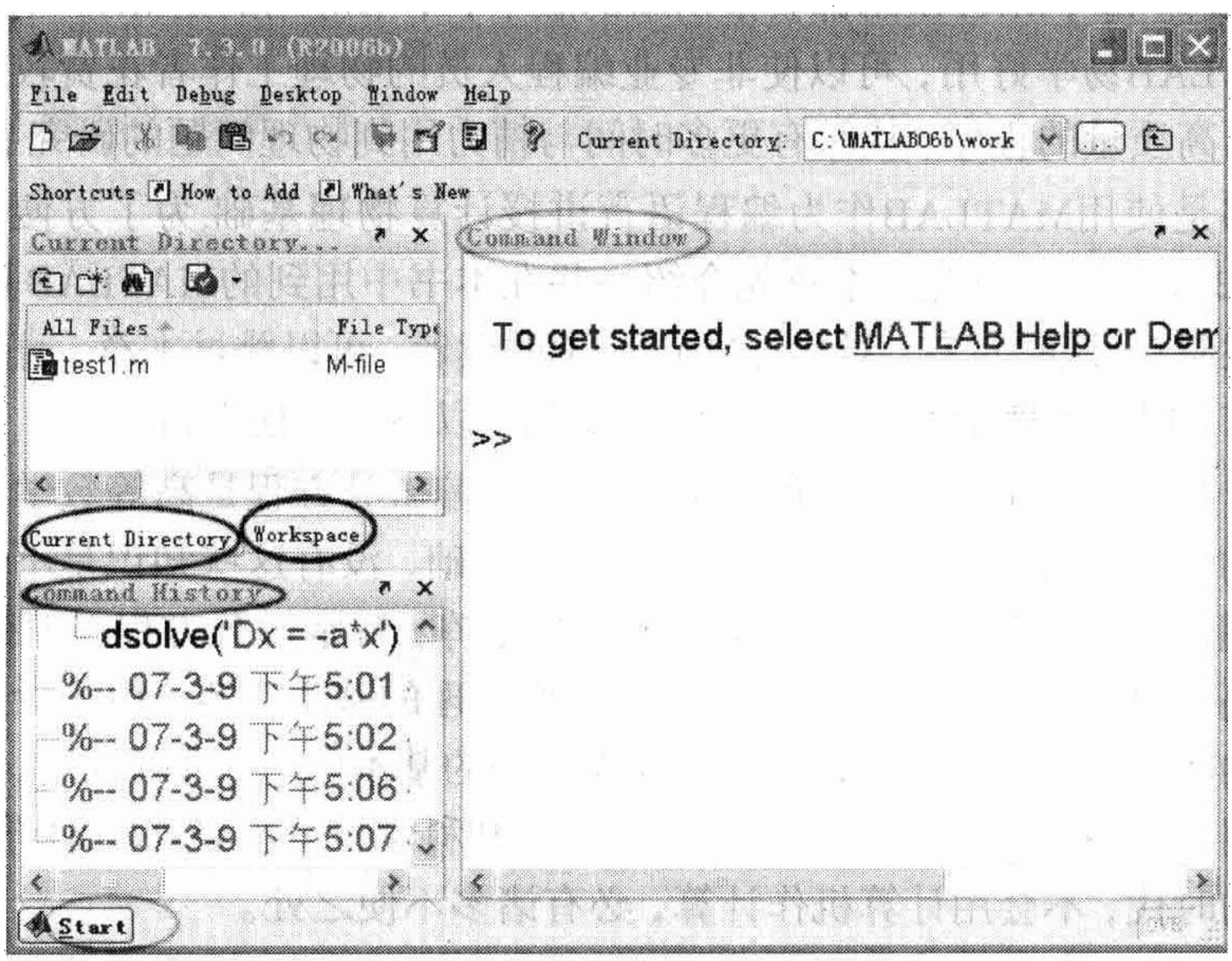


图 1.1 MATLAB的操作界面

要退出MATLAB的方法有三种：用鼠标点击操作界面窗口的右上角的× 号，用鼠标点击菜单File/Exit MATLAB或者直接在指令窗中键入quit后回车。

工作界面左下的Start（图中也用圆圈标出）按钮可以打开帮助界面(help)、演示界面(demos)和其他工作界面。在demos界面下的Matlab/Desktop Tools and Development Environment目录下有关于这些窗口的视频，是Flash 形式的动画，有英文解说，可以很快地了解窗口的详细用途。

这个界面上的其他一些图标与WINDOWS 下的其他软件如WORD的图标的用法相似，这里不再细述。

§1.1.2 指令窗中的功能

在指令窗口的运算反映了MATLAB的基本特征，下面试举几例来说明如何在指令窗口进行数值计算、符号计算、作图和运行程序。在下面的例子中，>>是 MAT-

LAB 的运算提示符，只有在这个符号后面才能从键盘输入运算式，输入完成以后必须按回车键以后才能得到结果，即例子中没有>>符号的行所显示的内容。

例1

```
>> A=52.738                                  %对A赋值
A = 52.738                                   %按回车键以后屏幕显示结果
>> a=3.2477e2                                %对a赋值
a = 324.77                                   %屏幕显现的结果
>> B=A+a                                     %加法计算A+a
B = 377.51
>> a=3.2477e-2                               %重新给a赋值
a = 0.032477
>> B=A+a                                     %重新计算A+a
B = 52.77
>> S='Hello'                                 %输入字符串时要加单引号
S = Hello
```

例1表明，在数值计算中，如果计算A+a，则要先给A和a赋值，赋值的方法是用等号。注意MATLAB的变量名是区分大小写的，所以A与a是不同的变量。如果对a重复赋值，则最后一次赋值才是a的实际值。如果将字符串赋予变量，以后可以用该变量名调用这个字符串，如上面对字符串Hello的操作就是如此，以后调用S就能调出字符串Hello。

将计算结果显现在屏幕上要占用一定的时间，会影响计算速度。有些复杂运算过程的中间结果，是不需要显示的，而在算式后加上分号以后，计算结果就不会显现在屏幕上，这样在有大量计算的程序中，可以节省大量的用来显示中间计算结果的时间，从而提高程序运行的速度。初学者往往由于不注意使用语句后面的分号，造成程序运行缓慢。

数据可以用常规的十进制小数表示法，如A的赋值；也可以用科学计数法即10的幂次用e或E加正负数表示，如1e2=10^2，1e-2=10^{-2}，例1中对a的两次赋值分别使用了正负幂次。

变量名用字母打头，后面可以跟字母、数字、下划线，数目不限。但只有变量名的前面63个符号有效。要查看一个变量的内容，只需键入其名称，如输入A，B，a等。需要强调的是，决不可以用数字作变量名的开头，比如变量名1a，2a是无效的，但a1，a2是合法的。也不可以用中文作变量名，比如用速度、位移等名词作变量名是不行的，但可以用它们的拼音作变量名以便记忆。变量名中也不可以包括空格、运算符号与标点。

在指令窗的所有输入过的变量名都会自动被 MATLAB 保存在工作内

存 (Workspace) 中，可以在工作内存窗口中查看，只要不关闭指令窗就可以调用它们。如果变量是一个矩阵，则点击工作内存中的变量名看到的数据以表格形式出现，可以对其中的数据进行修改，这就是MATLAB的矩阵编辑器。用指令whos可以查看工作内存中所有变量的名称。清除某个变量的语句格式为指令clear + 变量名，清除所有变量用指令clear all。清除屏幕上所有显示内容的指令是clc。

在窗口中还可以用以下指令来完成相应的工作。用后面介绍的help帮助系统可以更详细地了解它们的用法。

who	列出内存中的变量名	type	显示指定文件的内容
whos	列出内存中的变量名及其性质	which	列出文件所在的目录
clear	清除内存	dbtype	显示文件中带行号的内容
clc	清除工作窗中的显示内容	disp	显示文字或变量内容
clf	清除图形窗中的显示内容	edit	编辑指定的文件
what	查看指定目录下的文件名	close all	关闭所有窗口
exist	查找变量或文件	dir	列出指定目录下的文件

在例1中使用了加法算符+，在MATLAB中使用的其他数学运算符还有：

运算符	+	−	*	/	sqrt	∧	( )
含义	加	减	乘	除	开方根	幂	括号

例2 计算

$$2\times(300-2\mathrm{i})^2+\frac{1}{\sqrt{5+2\mathrm{i}}}$$

```
>>2*(300-2*i)^2+1/sqrt(5+2*i)                    %输入计算式
ans = 1.7999e+005 -2400.1i                      %回车后屏幕显示的计算结果
```

例2表明，MATLAB可用复数计算，运算符号和表达式与数学上使用的形式相近，i或j是默认的虚数单位，计算结果默认用ans表示。在MATLAB7.6以前的版本中，计算式只能使用圆括号，不能使用中括号与花括号。在7.6版本中已经可以使用中括号。

MATLAB可以进行关系运算，使用的关系运算符是:

运算符	<	<=	>	>=	==	~=
含义	小于	小于等于	大于	大于等于	等于	不等于

进行关系运算时，如果结果为真，输出值为1，如果结果为假，输出值为0。

```
>>    a=3,    b=5                                %给变量a,b赋值
a=3
b=5
```

```
>>  a>b                                    %比较是否a>b
ans=0                                      %输出值为0,即表达式a>b为假
>>  a<b                                    %比较是否a<b
ans=1                                      %输出值为1,即表达式a<b为真
```

MATLAB的逻辑运算符有：

运算符	&	—	~	xor
含义	与	或	非	异或

下面是例子：

```
>> A=[1,  3,  -1, 0, 7, 2]                 %输入向量A
>> B=~(A>2)                                %找出A中不大于2的元素的位置
B =1     0     1     1     0     1
>> D=A.*B                                  %找出 A 中不大于2的元素
D =1     0    -1     0     0     2
>> C=(A>0)&(A<3)                           %找出 A 中大于0小于3的元素的位置
C =1     0     0     0     0     1
>> E=A.*C                                  %找出 A 中大于0小于3的元素
E =1     0     0     0     0     2
```

应用c=xor(a,b)的结果如下:

a	0	0	非0	非0
b	0	非0	0	非0
c	0	1	1	0

MATLAB的许多计算功能都是用指令来实现的，这些指令相当于一批可以直接调用的小程序。只要按照设定的格式输入，就会得到固定格式的输出。这些指令的使用十分简单，使用方法大致相同，我们以解方程的指令solve作个示范。

例3 解三角方程

$$\cos 2x + \sin x = 1$$

```
>>s = solve('cos(2*x)+sin(x)=1')
s =  0
     pi
     1/6*pi
     5/6*pi
```

这里解三角方程是用指令solve来实现的，输出的解以变量s命名，所得结果是有四个分量的一个矢量。注意输入的方程要用单引号括起来。

例4 解三次方程

$$x^3 - 2x^2 = x - 1$$

```
>> s = solve( ' x ^ 3 - 2 * x ^ 2 = x - 1' )
s =   1/6*(28+84*i*3^(1/2))^(1/3)+14/3/(28+84*i*3^(1/2))^(1/3)
+2/3-1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)
+2/3+1/2*i*3^(1/2)*(1/6*(28+84*i*3^(1/2))^(1/3)
-14/3/(28+84*i*3^(1/2))^(1/3))

-1/12*(28+84*i*3^(1/2))^(1/3-7/3/(28+84*i*3^(1/2))^(1/3)
+2/3-1/2*i*3^(1/2)*(1/6*(28+84*i*3^(1/2))^(1/3)
-14/3/(28+84*i*3^(1/2))^(1/3))
```

指令solve不只可以解三角方程，也可以解代数方程等。这里是解三次方程。如果要替换成可变精度的浮点数形式，所用的指令是vpa(S，D)，其中S表示要替换的对象，D表示要用几位有效数字。例如上述结果变成有10位有效数字的浮点数的方法是：

```
vpa(s, 10)
ans =   2.246979604   +  0.1e-9*i
       -0.8019377357  +  0.3e-9*i
              0.5549581323.5e-9*i
```

对于不同的MATLAB版本，小数点后面的数字有微小的差别。

例5 解方程

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

```
>>syms  a  b  c  x                    %建立符号变量a,b,c,x
>>f=a*x^2+b*x+c                       %建立符号变量的二次方程
>>solve(f)
ans =  [1/2/a*(-b+(b^2-4*a*c)^(1/2))]
       [1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

例5是用solve解符号变量方程。求解前先用指令syms建立有关的符号变量a，b，c，x 以及要解的符号变量方程，然后才能使用解方程的指令solve。如果不指明系数与变量，则以x作为默认的待求解的变量，如上面所做的那样。也可以对指定变量求解，如指定b是变量解同一个方程：

```
>>  b = solve(f,b)
```

```
b = -(a*x^2+c)/x
```

符号变量的知识可以参看符号工具箱的介绍。

例6 电阻值为x，y，z三个电阻组成如图1.2所示的星形网络，已经测出三个端点的电阻为r_1，r_2，r_3,求三个电阻值x，y，z。

根据欧姆定律，可以列出下面的非线性方程组，其中r_1, r_2, r_3为已知，x, y, z是未知量。

$$\begin{cases} \dfrac{1}{x}+\dfrac{1}{y+z}=\dfrac{1}{r_1} \\ \dfrac{1}{y}+\dfrac{1}{z+x}=\dfrac{1}{r_2} \\ \dfrac{1}{z}+\dfrac{1}{x+y}=\dfrac{1}{r_3} \end{cases}$$

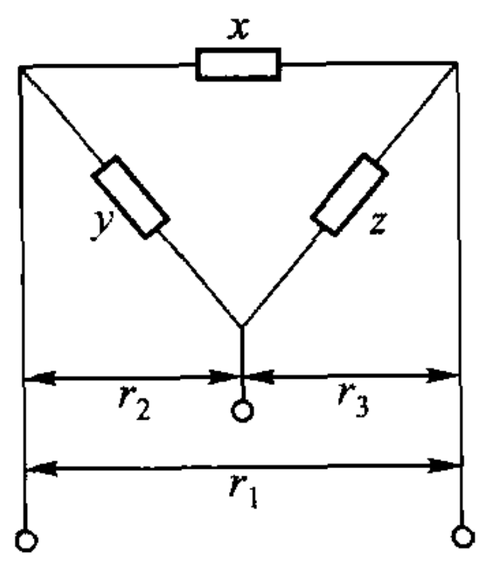


图 1.2 三个电阻组成的星形网络

这里要解是非线性的代数方程组，仍然可以用指令solve求解，操作是：

```
>> syms x y z r1 r2 r3
>> [x,y,z] = solve(1/x+1/(y+z)-1/r1,1/y+1/(z+x)-1/r2,1/z+1/(x+y)
-1/r3)
x = 1/2*(-2*r1*r3-2*r2*r1+r1^2-2*r2*r3+r3^2+r2^2)/(r1-r3-r2)
y = -1/2*(-2*r1*r3-2*r2*r1+r1^2-2*r2*r3+r3^2+r2^2)/(r1+r3-r2)
z = -1/2*(-2*r1*r3-2*r2*r1+r1^2-2*r2*r3+r3^2+r2^2)/(r1-r3+r2)
```

读者如果用笔求一下这个方程的解，就能对笔算与机算的差别有较深的体会，也许能增加学习计算物理的兴趣。

MATLAB解常微分方程的指令也很简单，下面是一个例子。

例7 解常微分方程

$$\frac{\mathrm{d}}{\mathrm{d}t}x=-ax$$

```
>> dsolve('Dx = -a*x')
ans = C1*exp(-a*t)
```

这里，dsolve是解常微分方程的指令，符号D代表$\dfrac{\mathrm{d}}{\mathrm{d}t}$。

下面的例子更复杂一点，涉及了一些新的操作，如矢量的输入、矩阵的建立、使用正弦函数计算还有画图。

例8 以x表示从0到6间隔为0.1的所有数，计算$\sin x, \sin 4x$，并画出图形。

操作步骤如下：

```
>> x=0:0.1:6;                                  %建立矢量x
>> A=[x; 4*x];                                 %建立两行的矩阵A
>> B=sin(A);                                   %计算A的正弦函数值
>> plot(x,B(1,:),x,B(2,:))                     %画两条正弦曲线
```

下面解释一下这些语句的用途。第一句是输入矢量x，输入方法是给出起点值、终点值以及其中的间距（步长）。

第二句是建立矩阵A，A的第一行是矢量x，第二行是4*x，两行数据用分号隔开，再用中括号括起来，就构成一个矩阵。将它们组成一个矩阵是为了同时计算所有的函数值，因为计算正弦函数的指令sin具有这种功能。

第三句是以A的每一个元素的数值作为弧度值，用sin计算相应的正弦函数值，又得到一个新的矩阵B。它也是有两行元素的矩阵。

第四句是作图。plot是画图指令，所画的图形在一个新的窗口中，其用法是在它括号后面注上变量的数据和相应的函数值即可。B(1，:) 和 B(2，:) 分别是矩阵B的第一行和第二行元素。因为要画两个图形，所以有两组变量数据和两组函数值。

在每句指令后面都加上了分号，也就是不显示运算结果，这是因为本例运算结果的数据较多，显示会花费不少时间，所以取消了显示。读者要在计算机上亲自运行一下，如果想看看计算过程的数据，那就在输入时去掉语句后面的分号。

例8表明，MATLAB可以用向量、矩阵运算，在MATLAB中所有的初等函数都可以用指令计算，§1.10中的习题3列出了所有的初等函数指令。

将画正弦曲线的plot语句替换成comet(x，B(1，:))，就能看到一个光点沿正弦曲线移动，产生动画效果。

MATLAB对变量、向量、矩阵、函数赋值或命名的格式都很相似，都是：

名称=(赋值对象或命名对象)

对变量命名时不需要说明是否复数，多少维，精度多少。MATLAB都是当作双精度复数处理。如果是矩阵，MATLAB会根据输入的数值自动扩充以满足需要。

有几个特殊的变量名是MATLAB保留的，列举如下，读者一般不宜改变它们。

pi	圆周率
eps	2.2204e-016，是软件计算时能分辨的数值的最小相对间隔
Inf	大于10^{308}的数据如输入10^{1024}或5/0
NaN	非数如0/0
0	小于10^{-322}的数
realmin	最小浮点数，$2.225073858507201 \times 10^{-308}$
realmax	最大浮点数，$1.797693134862316 \times 10^{308}$

画图指令plot的用法十分丰富，下面是它的基本用法：

plot(x1，y1，x2，y2，...)　　画多条曲线

plot(y)　　以元素序号作自变量x，以y作为函数画图

plot(x，y，'color style marker')　　指定颜色，线型，标志

颜色：　青'c'，洋红'm'，黄'y'，红'r'，绿'g'，蓝'b'，白'w'，黑'k'

线型：　实线'－'，短划线'－－'，点线':'，点划线'－.'

标志：　点'·'，加号'+ '，圆圈'o'，星号'*'，叉号'×'，四方形's'，钻石形'd'，五角形'p'，六角形'h'，三角形（向上'v'，向下'∧'，向左'<'，向右'>'）

例如，要将前面的曲线改画成一条红色的点状曲线，并将数据点用+号标志出来的操作如下：

```
>> plot(x,B(1,:),'r:+')
```

显然，如果矢量x与矢量y分别是一个多边形的顶点的x坐标与y坐标，则可用 plot 画出这个多边形。由于两点画一线，为了使多边形图形封闭，需要将起点的坐标在矢量的最后重复输入一次。例如画一个如图 1.3 的三角形，它的顶点在 A(1，1)，B(2.3，1.5)，C(3，1)，方法如下：

```
>> x=[1,  2.3,  3,  1];
   y=[1,  1.5,  1,  1];
>> plot(x,y)
```

将plot语句替换成fill(x，y，'g')则是将三角形填色，g表示是填充绿色。

对于已经编好的程序，运行时是调用相应的程序名。例如下面的几个例子。

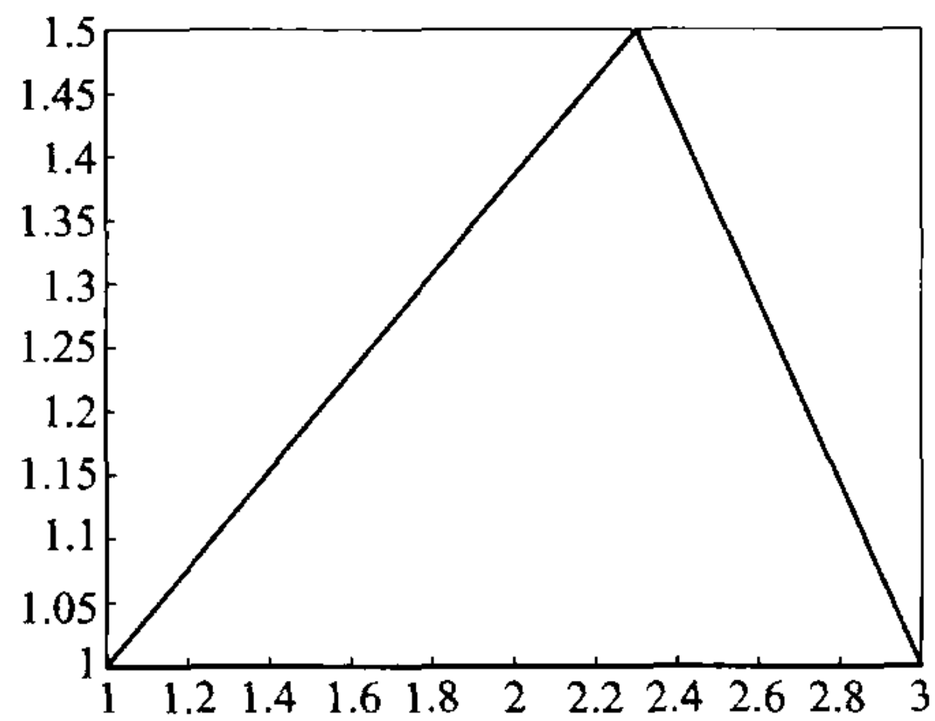


图 1.3　画三角形

例9　运行程序

```
>> ballode                    %MATLAB自带的演示小球落地弹跳的程序
>> funtool                    %函数计算器，可以对函数进行计算
>> taylortool                 %函数的泰勒展开计算器
>> demo                       %打开程序演示窗口
```

仔细查看demo的演示程序，是学习MATLAB的捷径。指令demo打开的窗口如图1.4所示。

图中右边的窗口里对演示的内容和使用方法作了详细的说明，点击左边窗口中的各项都会在右边窗口显示图标式的程序目录，再点击右边的图标，就可以看到相应的演示程序。

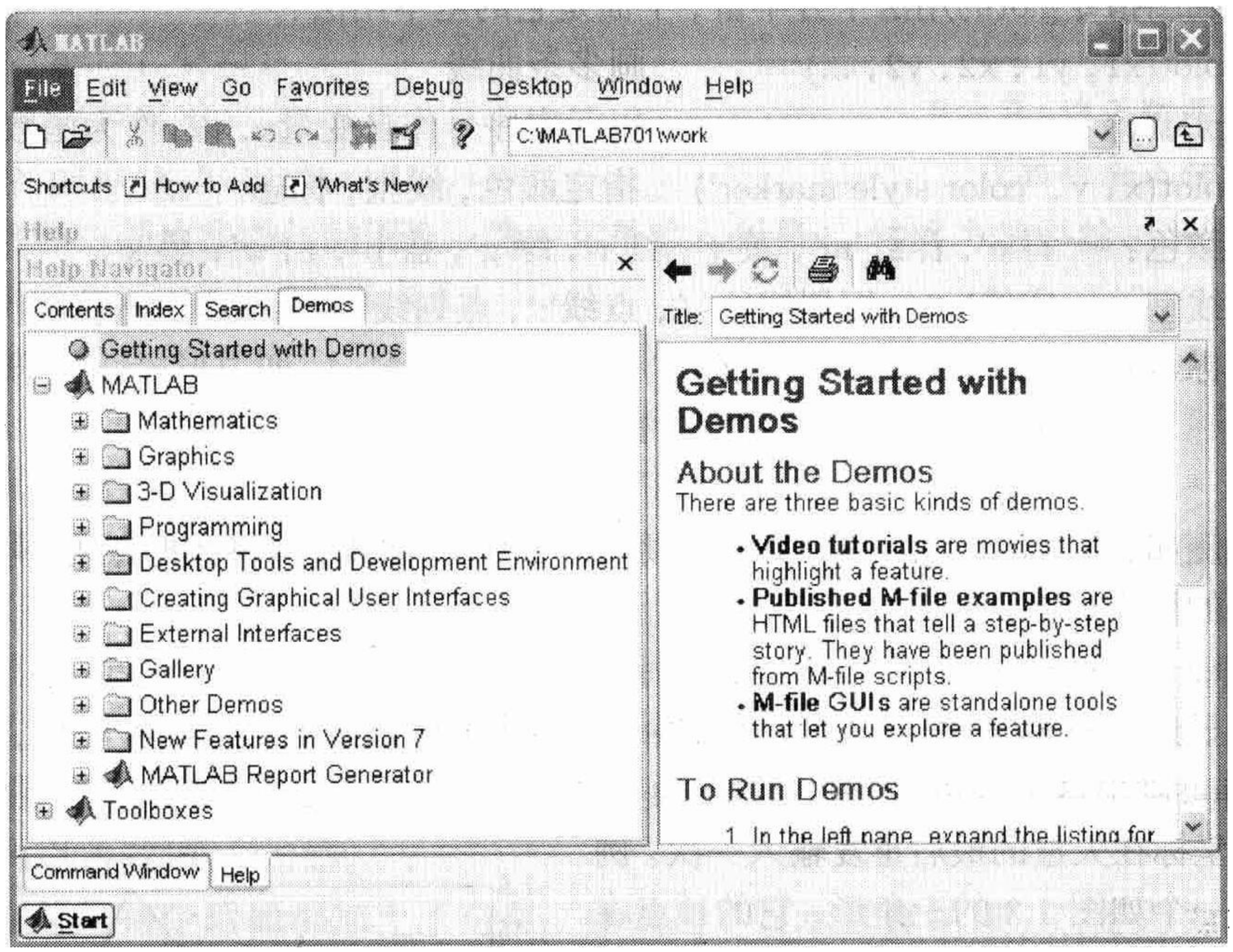


图 1.4 MATLAB的演示界面

§1.1.3 数据存储与显示

数据运算与存储的默认格式都是16位十进制数(二进制双精度数)，双精度数范围在$10^{-322} \sim 10^{308}$。数据显示有如下10种方式。

MATLAB命令	说明
short(默认)	3位整数4位小数的定点数，超出范围则用short e显示
long	2位整数14位小数的定点数，超出范围则用long e显示
short e	5个数字的浮点数，其中1位整数4位小数
long e	16个数字的浮点数，其中1位整数15位小数
short g	显示5个数字，自动选择定点数或浮点数
long g	显示15个数字，自动选择定点数或浮点数
hex	16位十六进制数
bank	银行用两位小数的定点数
+	显示数字的符号是正、负或零
rational	分数近似

改变显示方式的指令为format long(或者将long换成其他格式指令)，或直接用菜单。

行间距默认格式为 loose，即行间有空行，用 format compact 可压缩行间的空行。

字体设置可用菜单file/preference/command window/font & colors，比如选用Sans Serif，plain，18 是一种字体大小比较适合的等线体，而且这种字体能在程序中显示中文。

下面列表对比了数字234.7896和45.6985678945128在不同显示格式下在屏幕上出现的数据，读者应注意其中的差别，不要由于显示方式的差别造成对计算结果和数据精度的错误理解。

format	234.7896	45.6985678945128
short	234.7896	45.6986
long	2.347896000000000e+002	45.69856789451280
short e	2.3479e+002	4.5699e+001
long e	2.347896000000000e+002	4.569856789451280e+001
short g	234.79	45.699
long g	234.7896	45.6985678945128

例10　已知矩阵A，计算

(1) 矩阵A的逆矩阵B；

(2) A与B相乘所得的单位矩阵C；

(3) 矩阵C的平方根矩阵D；

(4) 矩阵D的绝对值矩阵E。

矩阵求逆的指令是inv，求平方根的指令是sqrt，求绝对值的指令是abs，矩阵相乘用*表示。

下面用 short（左边）和 short g（右边）两种方式显示计算结果，注意对比它们的区别，尤其注意矩阵 C 中为零的值在两种显示方式中出现的差别。

```
>> format short                          >>format short g
>> A                                     >>A
A =                                      A =
0.4447    0.9218    0.4057               0.4447    0.92181    0.40571
0.6154    0.7382    0.9355               0.61543   0.73821    0.93547
0.7919    0.1763    0.9169               0.79194   0.17627    0.9169
```

```
>> B = inv(A)
B =
 2.5956   -3.9226    2.8535
 0.8950    0.4383   -0.8432
-2.4139    3.3037   -1.2119

>> C = B*A
C =
1.0000  -0.0000  0
0        1.0000  0
0.0000   0.0000  1.0000

>> D = sqrt(C)
D =
1.0000    0 + 0.0000i 0
0         1.0000      0
0.0000    0.0000      1.0000

>> E = abs(D)
E =
1.0000   0.0000    0
0        1.0000    0
0.0000   0.0000    1.0000
```

```
>> B = inv(A)
B =
 2.5956   -3.9226    2.8535
 0.89504   0.43833  -0.84323
-2.4139    3.3037   -1.2119

>> C = B*A
C =
1             -4.4409e-016    0
0              1              0
1.1102e-016   3.0531e-016    1

>> D = sqrt(C)
D =
1              0 +2.1073e-008i  0
0              1                0
1.0537e-008    1.7473e-008      1

>> E = abs(D)
E =
1             2.1073e-008    0
0             1              0
1.0537e-008   1.7473e-008    1
```

数值计算难免产生误差，产生误差的原因有多种，首先，十进制的数用二进制来表示，就可能产生误差，其次，计算机是用有限的位数来表示数据，对于超出这个位数的数据就要进行取舍，这又带来了误差。在计算中由于采用的近似计算方法不同，对计算精度的要求不同，都会带来误差，这些误差积累起来所形成的误差就可能导致计算结果发生偏差。在MATLAB中，数值计算的默认设置是保留16位十进制的数值，所以在例10中，矩阵A与其逆矩阵B相乘的结果矩阵C应当是单位矩阵，但这个结果是在上述精度的意义上而言的，在short显示方式下，矩阵C 的非对角元素都是零，而在short g显示方式下，非对角元素的第16位小数并不是零。如果对它不停地进行开方计算，这种误差就会累积起来达到很大，以至使计算结果完全不可靠。

对于不熟悉的指令，可以查阅在线帮助系统。例如读者可以运行以下指令了解相关的知识。

```
>> help format          %查阅显示格式format 的简单含义
>> doc format           %查阅显示格式format在说明书中的详尽解释
>> help ops             %可查看所有的运算符号
>> help                 %显示所有的帮助文件目录
```

§1.2 数据格式与算符

MATLAB可处理的数据格式很多、很灵活，甚至可以一起处理数据、字符、函数和文本。为了大量地高效地快速地处理数据，MATLAB是将所有要处理的数据统一作成矩阵形式，然后对矩阵进行运算。所用的数据可能是一维、二维、三维甚至更高的维数，在MATLAB中统一称之为列阵(array)。在使用中，根据情况会作如下区分。

- 标量(scalar)　　单个数据
- 矢量（向量）(vector)　　一行或一列数据
- 矩阵(matrix)　　m行n列的二维数据
- 列阵(array)　　三维以上的数据
- 数据网格(meshgrid)　　供画图用的数据格式
- 基元列阵(cell array)　　不同类型数据混合组成的列阵
- 结构列阵(structure array)　　有域名的不同类型数据组成的列阵

从这个分类中可以看出，它们的关系是，标量的组合生成矢量，矢量的组合生成矩阵，矩阵的组合扩充生成列阵，数据的混合排列形成新的格式……

§1.2.1 向量的输入

如果已经知道向量的全部数据，可以通过直接输入生成向量，如下面行向量a的输入，输入时数据用逗号或空格分隔，全部数据用括号括起来。对于列向量，输入时数据之间用分号分隔，或者先按行向量输入，再转置成列向量，如下面列向量b的输入。

如果向量中各个数据之间的间隔是固定的，那么只要输入第一个数据、间隔(也称步长)和最后一个数据，再将三者用冒号隔开，如下面向量c的输入。

如果要在两个数据之间生成确定的若干个线性等间隔的数据，或者是对数等间隔的数据，可以使用专门的指令linspace和logspace，如下面生成的向量d和向量e。

以下是用指令生成向量：

```
>>a=[3, 5, 7, 8]或>>a=[3 5 7 8]          %行向量a的两种输入方法
>>b=[3;5;7;8]                            %列向量b，b是行向量a的转置
>>b=a'                                   %用转置算符'直接由a生成b
>>c=1:2:13                               %从1到13间隔为2的数列
>>d=linspace(0,2*pi,20)                  %零到2π之间等间隔的20个数
>>e=logspace(1,3,50)                     %10^1 ~ 10^3之间产生50个对数值
```

向量其实就是矩阵中的行向量或列向量，所以向量运算与下面讲的矩阵运算法则相同。

§1.2.2 矩阵

§1.2.2.1 矩阵的生成、标识与修改

1. 矩阵的输入

直接输入一个矩阵的方法如下例所示，即同一行的数据用逗号或空格分隔，各行数据用分号隔开，全部数据用方括号括起来。

```
>> A=[2, 4, 5;  9, 7, 1]
A =  2     4     5
     9     7     1
```

MATLAB允许使用“空”矩阵，即没有数据的矩阵，生成的方法是：

```
>> C=[ ]
C=[ ]
```

MATLAB还可以用NaN，Inf作为矩阵中的元素，它们和空矩阵一样，有些特殊的用处。在以后的例子将会用到这些知识。

2. 矩阵的标识方式

为了能从矩阵中找到任意一个数据或者任意一批数据，必须给其中的数据编号，这就是矩阵元素的标识方式。编号的方式是：矩阵名（行号，列号），如下面的 A(i,j)。使用冒号可以表示一行元素，一列元素或者几个相连续的元素，如A(i,:)是第i行全部元素， A(:,j) 是第 j 列全部元素。A(2:4, j) 是第 j 列中第 2 到第 4 个元素，A(i, 2:4) 是第 i 行中第 2 到第 4 个元素。英文单词 end 可表示矩阵某行或某列的最后一个元素，如A(end, j) 与 A(end-1, j) 表示的是第 j 列的倒数第 1 与倒数第 2 个元素，A(i, end) A(i, end-1) 则是第 i 行的倒数第 1 与倒数第 2 个元素。如果按照第 1 列，第 2 列，第 3 列…… 的顺序将矩阵所有的元素排列一列，那么就可以用一个标号来表示矩阵元素如A(k)。这些标识方式归纳如下：

A(i, j)	第 i 行 j 列元素
A(:, j)	第 j 列所有元素

A(i,:)	第 i 行所有元素
A(2:4, j)	第 2 到 4 行的第 j 列元素
A(i, 2:4)	第 2 到 4 列的第 i 行元素
A(end, j)	第 j 列的最后一个元素
A(end-1,j)	第 j 列的倒数第二个元素
A(i, end)	第 i 行的最后一个元素
A(i, end-1)	第 i 行的倒数第二个元素
A(k)	将第一列，第二列……排成一列后其中的第 k 个元素

3. 指令生成矩阵

一些特殊的矩阵用指令生成既简单又方便，如：

zeros	零矩阵
ones	全部元素为1的矩阵
eye	单位矩阵
rand	均匀分布的随机数矩阵
magic	建立幻方阵
randn	n 维正态分布的随机数矩阵
cell	建立空的基元列阵
diag	建立对角矩阵或提取矩阵对角元
logspace	生成对数等分的行向量
linspace(a,b,n)	在a，b之间生成等间距的n个数

用这些指令可以生成向量，矩阵，三维列阵，四维列阵……使用的格式雷同，只要按格式输入相应的参数就可以，例如：

`>>zeros(5)`	生成 5×5 的零矩阵
`>>zeros(1,4)`	生成 1×4 的零矩阵
`>>zeros(2,4)`	生成 2×4 的零矩阵
`>>zeros(2,4,6)`	生成 2×4×6 的零元素组成的三维列阵
`>>ones(3,5)`	生成 3×5 的全部元素为1的矩阵
`>>rand(3,50)`	生成 3×50 的均匀分布的随机数矩阵

4. 合并与删除

已经建好的矩阵可以进行改造，如增加元素，删除元素，合并矩阵，将元素重新排列等。

给矩阵增加元素只要指明该元素所在的位置就行，MATLAB会自动将矩阵扩充到需要的大小，对于没有输入的元素，MATLAB将自动以零来代替。如

```
>> M=[2, 3;  7, 9]
M =  2     3
```

```
      7     9
>> M(3,3)=1                          %给2×2的矩阵M增加一个元素M(3,3)
M =  2     3     0                   %矩阵M自动扩充成3×3的矩阵
     7     9     0
     0     0     1
```

对矩阵可以删除指定的行或者指定的列，方法是令被删除的行或列等于[ ]。但不能只删除一个元素。如

```
>> M(3, :)=[ ]                       %将矩阵M的第3行删除
M =  2     3     0
     7     9     0
```

用[ ]将矩阵括在一起就完成了矩阵的合并。例如

```
>> P=[M,  M]                         %将矩阵M按行排列
P =  2     3     0     2     3     0
     7     9     0     7     9     0
>> Q=[M;  M]                         %将矩阵M按列排列
Q =  2     3     0
     7     9     0
     2     3     0
     7     9     0
```

还有几个指令是专门用以对矩阵进行修改的，列举如下：

repmat	将矩阵按指定的行数或列数排列
reshape	将矩阵中的所有元素按新指定的行与列重新排列
flipud	将矩阵上下翻转
fliplr	将矩阵左右翻转
rot90	将矩阵逆时针转90度

§1.2.2.2 矩阵运算

矩阵运算有很多种形式，我们将它们归纳为两大类。

1. 用指令对矩阵进行操作

- 对矩阵内每个元素单独运算：如指令sin(A)对矩阵的各个元素求正弦值。
- 对矩阵的一列或一行元素运算：如指令sum(A)是对A的各列求和。
- 对整个矩阵运算：如指令inv(A)求整个矩阵的逆。

2. 矩阵之间的数学运算

矩阵之间可以进行下列两种不同的运算。

- 数组运算：两个矩阵的对应元素之间的数学运算。

• 矩阵运算：两个矩阵按矩阵运算法则进行的运算。

两种运算使用不同的运算符号，其差别是数组运算符号多个圆点。如数组乘法记为A.*A，矩阵乘法则记为A*A，所有运算符号列表如下：

	转置	加	减	乘	右除	左除	幂
矩阵运算算符	A′（取共轭）	+	−	*	/	\	∧
数组运算算符	A.′（不取共轭）	+	−	.*	./	.\	.∧

下面试举几例，先建立矩阵A：

```
>> A=[ 1 1 1 ; 2  2  2 ;  3  3  3 ]
A =  1     1     1
     2     2     2
     3     3     3
```

再对比两种运算的结果：

```
>> A.*A     %数组乘法                    >> A*A     %矩阵乘法
ans = 1     1     1                     ans =  6     6     6
      4     4     4                            12    12    12
      9     9     9                            18    18    18
```

本例表明，尽管参与运算的矩阵相同，但数组乘法与矩阵乘法的结果显然不同。

矩阵运算符号右除“/”的规则与矩阵运算法则中的规定相同，而矩阵运算符号左除“\”是矩阵运算法则中没有的运算。其实符号左除“\”和右除“/”是用两个很简洁的符号表示代表两个不同程序，它们的含义是：

若

$$A * B = C$$

则已知A和C求B时用左除，

$$B = A \backslash C$$

已知C和B求A时用右除，

$$A = C/B$$

例如：

```
>> A=[2, 8, 10; 11, 3,  7; 6,  9,  4 ];
>> B=[12,  9,  4;  5,  2,  21; 8, 4,  6];
>> C=A*B
C = 144    74   236
    203   133   149
```

```
     149     88    237
>> A\C     %用左除求矩阵B
ans = 12     9     4
       5     2    21
       8     4     6
```

```
>> C/B     %用右除求矩阵A
ans =  2     8    10
      11     3     7
       6     9     4
```

左除法可解形式为$AX = b$方程组。例如下面的例子：

$$\begin{pmatrix} 3 & 5 & -7 \\ 2 & -12 & 3 \\ -1 & 9 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ -5 & 6 \\ 2 & 7 \end{pmatrix}$$

```
>> A = [ 3, 5, -7; 2,-12, 3; -1, 9, 8 ]; b = [ 34; -56; 27 ];
>> X = A \ b
ans  =   0.5474
         4.3854
        -1.4901
```

3. 矩阵运算指令

下面是一部分常用的矩阵运算指令：

sum(A)	求各列元素之和	det(A)	求矩阵行列式的值
prod(A)	求各列元素之积	eig(A)	求本征值
max(A)	求各列的最大的元素	min(A)	求各列的最小的元素
median(A)	求各列的中位元素	sort(A)	使各列元素按递增排序
mean(A)	求各列的平均值	std(A)	求各列的标准差
trace(A)	求矩阵的迹	rank(A)	求矩阵的秩
cumsum(A)	求各列元素累计和	cumprod(A)	求各列元素累计积
kron	张量积	cross	向量叉积

例如：

```
>> A=[44,      92,      40;
      61,      73,      93;
      79,      17,      91]
>> sum(A)
ans = 184   182   224
>> trace(A)
ans =   208
>> eig(A)
ans =  195.26
```

```
6.3712 +      31.263i
6.3712 -      31.263i
```

所有计算函数值的指令都可以应用于矩阵，我们在第一节曾用过正弦函数对矩阵进行计算，计算的结果是将矩阵的每一个元素作为因变量所对应的函数值，并且得到一个新的大小与原来矩阵相同的由函数值组成的矩阵。

§1.2.3 列阵

列阵是一种多维数组，可以看成是矩阵的一种推广。这里我们作一个形象的比喻，矩阵由行元素与列元素构成，把它看成书本的一页，每个元素相当一页书中的一个字符，那么一本书就相当三维的列阵，将书本作一排列堆积，就形成了四维列阵。类似于行指标，列指标的记法，我们不妨把第三个指标记之为层，第四个指标记之为块…… 可以把行、列、层、块……都统称为维指标(dim)，则它们的次序是：

dim=1(行指标)，2（列指标），3（层指标），4（块指标），……

m行n列l层k块……的列阵总的元素数目为$m \times n \times l \times k \times \cdots$。

对列阵元素的标识也是指明它对应的行、列、层、块……指标，就好比我们是在找第几本书中的第几页的第几行的第几个字符……

由于屏幕上只能显示二维数据，所以在建立一个列阵之后，在屏幕上是逐个显示其中所包含的矩阵，例如

```
>> ones(4,3,2)                    %生成4行3列2层的全部元素为1的三维列阵
ans(:,:,1) = 1     1     1        %第一层是4行3列的矩阵,全部元素为1
             1     1     1
             1     1     1
             1     1     1
ans(:,:,2) = 1     1     1       %第二层也是4行3列的矩阵,全部元素为1
             1     1     1
             1     1     1
             1     1     1
```

从这里可以看到常用的将列阵降维的方法，就是给定列阵中的一个指标值，所得的列阵就降了一维，如这里的三维列阵在给定层指标1，2以后，所得到的是两个矩阵，也就是只剩下二维了。

下面以交换矩阵指标次序的指令flipdim举例说明维指标的应用：

```
>> A=[1, 2; 3, 4; 5, 6]                                  %建立矩阵A
A =  1     2
     3     4
     5     6
```

```
>> B=flipdim(A,1)                    %按A的行指标进行交换
B =  5     6                         %第三行变为第一行
     3     4                         %第二行保持不动
     1     2                         %第一行变为第三行
>> c=flipdim(A,2)                    %按A的列指标交换
c =  2     1                         %第一列变为第二列
     4     3                         %第二列变为第一列
     6     5
```

§1.2.4 数据网格

数据网格是MATLAB为画图而构造的数据结构。在选定坐标变量的变化范围后，用指令meshgrid 或ndgrid生成。

二维的数据网格有两个元素数目相同的矩阵，三维的数据网格有三个元素数目相同的三维列阵。多维的数据网格以此类推。

下面是由矢量$\boldsymbol{x}$，$\boldsymbol{y}$生成的二维数据网格$\boldsymbol{X}$，$\boldsymbol{Y}$：

```
>> x = [ 1 3 5 ];     y = [ 2 4 6 ];
>> [X, Y] = meshgrid( x, y )
X = 1     3     5
    1     3     5
    1     3     5
Y = 2     2     2
    4     4     4
    6     6     6
```

由$\boldsymbol{X}$，$\boldsymbol{Y}$在对应位置上每取一个元素，就表示平面上一个点的坐标，所以$\boldsymbol{X}$，$\boldsymbol{Y}$共代表了九个点的坐标，分别是：

```
(1, 2)   (3, 2)   (5, 2)
(1, 4)   (3, 4)   (5, 4)
(1, 6)   (3, 6)   (5, 6)
```

下面的例子表明由矢量$\boldsymbol{x}$，$\boldsymbol{y}$生成的函数$\boldsymbol{z}$仍是矢量，矢量$\boldsymbol{z}$只有三个值；而数据网格$\boldsymbol{X}$，$\boldsymbol{Y}$生成的函数$\boldsymbol{Z}$则是矩阵，矩阵$\boldsymbol{Z}$有九个值，这九个值与平面上的九个点相对应。所以利用数据网格$\boldsymbol{X}$，$\boldsymbol{Y}$才能画出空间的曲面。

```
>> z = x.^2 + 2*y
z = 5     17     37
>> Z = X.^2 + 2*Y
Z = 5     13     29
```

```
     9     17     33
    13     21     37
```

三维数据网格的数据结构与二维数据网格相似，但是要用三个三维列阵表示。下面这个例子是用三个矢量$\boldsymbol{x}$，$\boldsymbol{y}$，$\boldsymbol{z}$组成了三维的数据网格$\boldsymbol{X}$，$\boldsymbol{Y}$，$\boldsymbol{Z}$：

```
>> x = [ 1 4 7 ];     y = [ 2 5 8 ];     z = [ 3 6 9 ];
>> [ X, Y, Z ] = meshgrid( x, y, z )
X(:,:,1) = 1     4     7
           1     4     7
           1     4     7
X(:,:,2) = 1     4     7
           1     4     7
           1     4     7
X(:,:,3) = 1     4     7
           1     4     7
           1     4     7
Y(:,:,1) = 2     2     2
           5     5     5
           8     8     8
Y(:,:,2) = 2     2     2
           5     5     5
           8     8     8
Y(:,:,3) = 2     2     2
           5     5     5
           8     8     8
Z(:,:,1) = 3     3     3
           3     3     3
           3     3     3
Z(:,:,2) = 6     6     6
           6     6     6
           6     6     6
Z(:,:,3) = 9     9     9
           9     9     9
           9     9     9
```

这里的$\boldsymbol{X}$，$\boldsymbol{Y}$，$\boldsymbol{Z}$都是三维的列阵，这个数据网格表示了空间的27 个点的坐标。当每个列阵的第三个指标（层指标）都取1时，表示的是z = 3的各个点

的坐标：

```
( 1, 2, 3 )    ( 4, 2, 3 )    ( 7, 2, 3 )
( 1, 5, 3 )    ( 4, 5, 3 )    ( 7, 5, 3 )
( 1, 8, 3 )    ( 4, 8, 3 )    ( 7, 8, 3 )
```

§1.2.5 基元列阵

基元列阵(cell array)可以将不同类型的数据按照与矩阵形式相似的结构组织起来加以应用。

例如下面就建立一个基元列阵：

```
>> G{1,1}=3;
>> G{1,2}=[1,2; 3,5];
>> G{2,1}='good';
>> G{2,2}='sin(x)';
>> G
G = [3]          [2×2  double]
    'good'        'sin(x)'
```

这里是以基元为单位作显示，其中有一个元素是矩阵，所以只给出了它们的维数和数据精度。如要查看它的内容，可以将它们调出来，方法与调用矩阵元素的方法相似。请注意显示基元与显示内容之间的区别。

```
>> G{1,1}
ans = 3
>> G{1,2}
ans = 1     2
      3     5
>> G{2,1}
ans = good
>> G{2,2}
ans = sin(x)
```

基元列阵用途广泛，读者可以参阅本书的参考文献。

§1.2.6 结构数组

结构数组是比基元列阵更复杂更灵活的数据结构，它将整个数组分成不同的“域”，每个域都赋予一个特定的名称，不同的域可以存放不同类型的数据。所以结构数组是一种有域名的数据，它类似于关系数据库中数据的结构。建立结构数组的语句是：

s = struct('field1', values1, 'field2', values2, ⋯)

其中field1，field2是域名，values1，values2是相应的数据。数据元素要与域名相对应。

例如，下面的指令产生一个结构数组s，域名是type，color，x，同时输入了两组数据。第一组数据在三个域的值是big，red，3；第二组数据在三个域的值是little，red，4。可用s(1)，s(2)查看每组数据的内容。如域名是文本型的字符串，要用引号括起来；如数据是基元列阵，要用花括号括起来，其中的文本型的字符串还是要用引号，而标量数据则不用引号。

```
>> s = struct('type',{'big','little'},'color',{'red'},'x',{3,4})
s =
    1×2 struct array with fields:
    type
    color
    x
>> s(1)
ans =
       type:  'big'
      color:  'red'
          x:  3
>> s(2)
ans = type:  'little'
      color:  'red'
          x:  4
```

如果想查看其中一个域type的内容，操作如下

```
>> s.type
ans = big
    ans = little
```

它表示在域type中有两组数据，其值为big和little。

对结构数组操作的指令还有fieldnames，getfield，rmfield，setfield。在解非线性方程或解常微分方程时，为了说明解方程的条件就要用结构数组。结构数组也是一种用途广泛的数据结构，读者可以参看本书的参考文献。

§1.2.7 字符和文本

输入文本的方法是用单引号，它输入的是$1\times n$的列阵。如

```
>> s = 'Hello'
```

是一个1×5的列阵。在MATLAB内部它是作为数来保存，但不是浮点数。如下的操作是将字符列阵转换为浮点数的列阵，每个数代表相应的字符的ASCII码。

```
>> a = double(s)
a = 72    101    108    108    111
```

而反向操作

```
>> s = char(a)
```

把数再变成字符。指令char是显示编码表中的字符，当编号是32到127时，显示的是ASCII码表中的字符。

可以用方括号把字符合并为一个大的字符串，如水平合并和垂直合并的操作分别是

```
>> h = [ s, 'world' ]
h = Hello world
>> v = [ s; 'world' ]
v = Hello
    world
>> k=['the','y']
k =they
```

§1.3 编　　程

§1.3.1 编辑程序

MATLAB有自备的程序编辑器，用以编辑、调试和管理程序。下面编辑两个小程序来说明其用法。首先编写一个简单而又实用的程序cc.m，步骤如下：

- 输入edit后回车，打开编辑器窗口。当然也可以用工具栏的图标或菜单打开编辑器。
- 在编辑器窗口输入以下内容：

```
clear all                    %清除内存中所有变量
close all                    %关闭所有图形窗口
clc                          %清屏
```

- 用文件名cc.m存盘。
- 在指令窗中运行cc，程序中的三条指令将一起执行，等效于从指令窗将三条指令逐条输入的效果。

再编写一个用矩阵画图的程序test1.m。

- 输入edit后回车，打开编辑器窗口。当然也可以用工具栏的图标或菜单打开编辑器。
- 输入以下内容：

```
x=0:0.1:6;                                  %设置自变量
A=[x; 4*x];                                 %构造矩阵
B=sin(A);                                   %求正弦函数值
plot(x,B(1,:),x,B(2,:))
```

- 用文件名test1.m存盘。
- 在指令窗中运行test1。

在编辑程序时应注意程序的排版格式，以保持程序的可读性，如后面讲到的for循环结构语句，if分支结构语句，尤其是以后常用的函数文件，它们都有一定的格式；在程序中可以加入一些说明性的文字，这些文字要用%开头，在%后面的语句都不会执行；在程序中，有时一个语句太长，在一行写不完就要用...分行，这样形式上为两行的语句在结构上仍属于一行，执行时不会出现错误；某个语句的运行结果不需要在屏幕上显示时可以在该语句末尾加上分号。

文件的命名规则基本与Windows操作系统的要求相似，但有几点不同，如不能用中文作文件名，即禁用“作业.m”等之类的文件名，因为MATLAB不能识别中文；也不能用数字作为文件名开头，或“1.m”或“3-2.m”作文件名，因为数字是参与运算的。

§1.3.2 调试程序

在输入程序时，程序编辑器具有自动检查功能，对于某些语法错误或文件格式错误会及时显示，如循环语句忘记输入end，或者几个嵌套的循环少一个end，在脚本文件内输入了子函数文件，在程序中输入中文字而没有加上注解符号，或者指令输入错误如将function输入成了funtiont等，这时程序中的错误之处会变成不同颜色以显示错误，或者在文件存盘时会提示错误，编写者应该根据提示及时修正这些错误。

程序初步编写完成以后，需要检查修正错误以及完善提高程序的质量，以保证程序能顺利高效地运行，这个过程称之为程序调试。调试程序是在程序编辑器中进行的，对于很简单的程序，如果把握性大，直接运行即可，MATLAB会自动检查出有错误的语句。复杂一点的程序，可以逐句运行，看看每句执行的情况，如果有错，随时修正。更复杂的程序，可以分块设置间断点，然后分块调试。调试好一部分程序，再调试下一部分程序。

以刚才编写的程序test1.m为例，调试过程如下：

先键入edit test1，即用编辑器打开程序test1，将光标移至第一行，按下F12键，

或者点击鼠标右健，于是在行号处出现一个红圆点，这表明在第一行设置了一个间断点，再按下F5，程序开始运行第一行语句，并在第一行语句的红点旁出现一个绿色箭头。表明程序已经运行到这里。以后的运行可有几种选择，第一种是要逐句运行，那么就按F10健，每按一次运行一行语句；第二种是连续运行，那么就要按F5键，程序将运行到下一个间断点才会停下来。如果下面没有设置间断点，程序就运行到最后。第三种选择是按F11，那么除了有和F10键相同的功能之外，还能进入到程序中调用的函数文件中去逐句运行而不是一步跳过。如果要停止这项操作，可按Shift+F11。程序运行时，如果语句有错误，就会停止在有错误的语句，并在指令窗口用红色字体给出对错误的说明。这里说明一下，如果要清除所设置的间断点，那么将光标移到有红圆点处，按下F12键即可，或者再用鼠标右键点击红点也可以。

这些操作都是直接用键盘上的功能键来完成的，也可以用菜单或工具栏中的图标来完成这些操作，读者只要看看菜单上的与功能键对应的名称就可以知道如何使用。将光标移到图标上，会显示图标的说明，由此也可以知道图标的用法。

程序调试完成以后，还可生成一个评价报告，它分析了程序各部分的运行时间，用户可以据此改进程序。例如点击菜单Tools/Open Profiler，将打开程序评价窗口(profiler)，在程序名的对话框中，用光标选定要评价的程序，回车即可，程序运行以后，会自动生成一个评价报告。读者可以运行一个较复杂的程序以体会此项功能。

程序在调试好以后，就可以使用了。但这并不意味着这个程序一定是正确的，也就是编程工作到此还不能算结束。因为目前只能说明这个程序符合编程语法的要求，而一个程序的正确性不仅与算法有关，更重要的是还与构造算法的物理模型、物理思想和物理概念有关。为了检验程序的正确与否，往往要选用以下方法：

① 根据物理模型，对可能得到的结果进行一些定性分析，用以预测计算结果。在可用解析方法求出一些简单解时，可把解析结果与程序计算的结果进行对比。

② 改变程序中的参数重新进行计算，分析比较所得的结果，看它们表现出的规律性是否能互相印证。

③ 如有可能，对物理模型再设计一个不同的算法，编辑新的程序进行计算，以检验旧程序。

还有一点很重要又往往容易被人所忽略的是，每次在运行一个新程序时，应该在指令窗中输入clear all，以清空内存中所有的变量。如果不这样做，一旦发生原来内存中的变量与现在所用的程序变量名相同的情形，则原来的变量赋值

就会影响新程序的运行，导致程序计算结果不正确或者程序不能运行。这时，没有经验的用户往往只在程序中查找原因，而不知道是清空内存的问题，以致浪费了时间还不能解决问题。

§1.3.3 设置搜索路径

MATLAB运行的程序必须在其搜索路径之下的目录中，否则就不能运行。所以设置搜索路径也就是选定MATLAB的工作目录范围，这通常是指要把用户自己建立的新目录放入这个范围中去。MATLAB本来已有的目录都是默认的工作目录，也就是已经在搜索路径之下。

用File/Set Path打开路径设置(Set Path)窗口如图1.5所示。

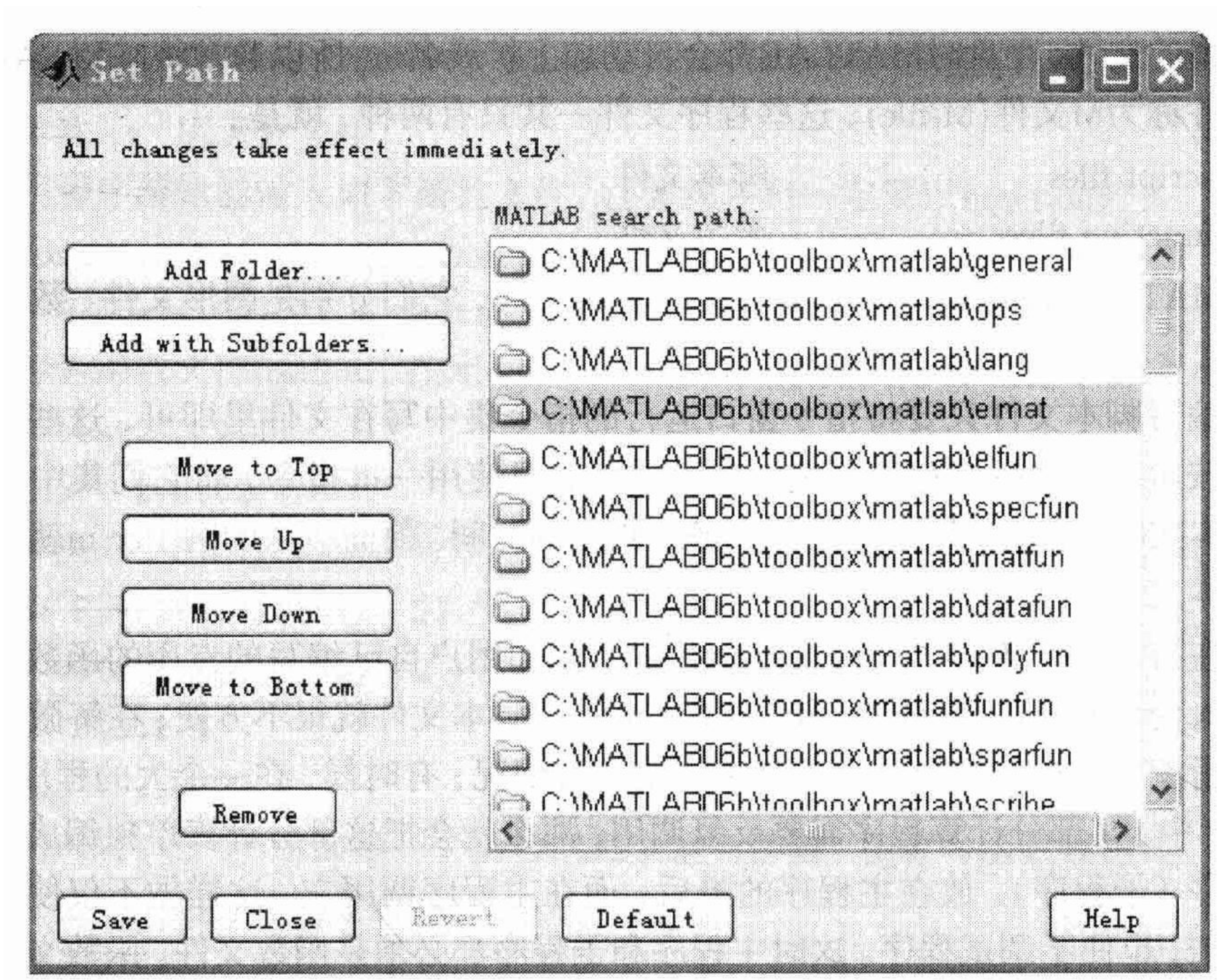


图 1.5　MATLAB 的路径设置窗口

其中所列出目录就是搜索路径之下的目录。自编的程序可以存在 MATLAB 目录下的 work 文件夹，这是 MATLAB 提供给用户存储文件的文件夹，它在 MATLAB 的搜索路径之下。如果想将一个新目录加入到搜索路径下，按照窗口中各个按钮上的指示进行操作即可。窗口中各个按钮的含义为：

Add Folder	把指定的目录加入搜索路径
Add with subfolders	把指定目录及其子目录加入搜索路径

Move to Top	将选定的目录移到最上面
Move to Bottom	将选定的目录移到最下面
Move Up	将选定的目录向上移动
Move Down	选定的目录向下移动
Remove	将选定的目录移出搜索路径
Save	保存新加入的目录
Revert	取消本次操作
Default	恢复默认设置

§1.3.4 两类程序文件

MATLAB的程序文件必须以m为扩展名，否则不能运行。只要是用程序编辑器编辑的文件，存盘时MATLAB都会自动加上扩展名m。所以通常将MATLAB程序文件称为M文件(M-file)，这些程序文件一共只有两种，就是：

script files	脚本文件
function files	函数文件

另外MATLAB还有以fig，dat等为扩展名的文件，它们分别是图形文件、数据文件等。

编写脚本文件只要将指令窗口运行的指令集中写在文件里即可，这些指令都是按出现的先后顺序执行的。对于需要反复使用一批指令，将它们集中编写在脚本文件，下次使用时就能节约输入指令的时间。前面所举的例子cc.m就是一个脚本文件。

在有些情况下脚本文件是不能用的，例如用户自己编写的专用的函数，需要输入变量的值并要求输出函数相应的值，用脚本文件就很不方便；在解微分方程时所编写的表示微分方程的文件也是这种情况；有时候，在一个大的程序（主程序）中，某部分计算程序需要反复调用，那么就会把这部分计算单独编成一个小程序（子程序），放在主程序的最后，而在主程序调用它。这样做不仅使主程序简洁，也便于调试程序。这时主程序和子程序都必须是函数文件。函数文件的使用比脚本文件复杂，初学者往往不能很好地区别和使用这两种文件，需要加强这方面的学习。

下面先介绍如何用函数文件建立用户函数。

§1.3.4.1 函数与函数文件

MATLAB内部有计算全部初等函数的程序，直接用指令调用即可，如 sin，cos， tan，log，log10，exp…… 取负数的平方根和对数会自动地得出相应的复数。MATLAB也提供了许多更高级的特殊函数，如勒让德函数legendre，贝塞尔函数besselj，bessely……这些函数大多数能接受复数。要看基本函数列表，只

需键入

```
>> help elfun
```

要列出特殊函数可以键入

```
>> help specfun
```

某些函数，比如方根、正弦函数，是内建的，是MATLAB的内核的一部分，所以非常有效，但计算的细节则不容易得到。另一些函数，如双曲正弦函数、伽马函数，是用M文件来建立的。可以查看文件的内容或者修改它。

用户在计算中要用到的函数也可能是MATLAB中没有的，那么就可以用这些基本函数来建造自己专用的单变量或多变量函数，方法有三种：一是用函数文件；二是用函数句柄符号@建立匿名函数；三是用符号变量。

例11　建立带参数θ 的关于x 的函数$f(x,\theta) = \cos^2(x^2) + \theta$，求$x = 3$，$\theta = 2.1$时的函数值。

方法一　用函数文件。

用程序编辑器输入如下函数文件，注意第一行一定要以function开头。

```
function F = test2(x,theta)
F = cos(x.^2).^2 + theta;
```

存为函数文件test2.m，按照test2(x,theta)的格式调用。即在指令窗输入

```
>>test2(3,2.1)
ans = 2.9302
```

方法二　用函数句柄符号@，这样建立的函数在MATLAB中叫匿名函数。

```
>> F = @(x,t)cos(x.^2).^2 + t
F =@(x,t)cos(x.^2).^2 + t
>> F(3,2.1)
ans = 2.9302
```

输入时，符号@后面第一个括号内是变量与参量，接着写函数表达式。

方法三　用符号变量。

首先介绍一下符号变量，这是与数值变量不同的一种新的变量，符号变量可以不赋值就进行运算，因此可用于公式推导，而数值变量必须赋值以后才能运算，所以不能用于推导公式。符号运算与数值运算的指令是完全不同的。依次点击start\demos\toolbox\symbolic 查看符号工具箱的演示程序，可以初步了解其用法。下面用符号变量建立同样的函数。

```
>> syms x theta                                  %建立符号变量
>> FFF= cos(x.^2).^2 + theta                     %用变量建立函数
FFF = cos(x^2)^2+theta
>> subs(FFF, {'x','theta'},{3,2.1})              %计算函数值
```

```
ans = 2.9302
```

符号函数的赋值相对复杂一些，不是直接代入数值，而是利用赋值指令subs，按照一定的格式输入相应的参数，在这里，依次输入的是函数名、变量名及变量对应的赋值，其中还要使用括号与引号。

§1.3.4.2 两类程序文件的比较

两类程序文件的格式与用途有明显的差别，一般来说，函数文件可以完成脚本文件的功能，而脚本文件却不具有函数文件的功能。

函数文件的编写格式与调用格式都是规定的，不能任意改变，说明如下：

① 函数文件格式。

第一行是函数定义行，必须按照如下顺序书写：

function[输出变量Q1,Q2,…]=函数文件名(输入变量P1,P2,…)

在程序编辑完成以后存盘时，MATLAB会自动以这个语句里使用的函数文件名存盘。虽然用户也可以用另外的文件名存盘，建议还是以这个语句中使用的函数文件名存盘为好。另外主函数文件在调用子函数文件只能用这个名字，因为子函数文件包含在主函数文件内，没有单独的文件名。

这里要强调的是，这种格式可以灵活变化，比如可以不要输入变量，有时也可能不要输出变量。

② 函数文件调用格式：

[输出变量Q1, Q2, …]=函数文件名(输入变量P1, P2, …)

函数文件的调用格式必须与编写时使用的格式完全相同。例如当编写函数文件时要求输入变量，则调用时一定要遵照这种格式输入变量，包括变量的顺序也不能改变。如果只在指令窗口输入文件名，那么程序是不能运行的。

③ 函数文件内可以包含一个或多个函数文件，称为子函数文件。

④ 每个函数文件都是使用独立的内存空间，所以它们的变量不能共享。要在不同的函数空间使用同一个变量，必须**在不同的空间传递变量**。

在不同的空间传递变量，有几种不同的方法可用，如用指令global建立各个子空间都能共享的全局变量，或用指令evalin进行跨空间传递变量的值，还可以用指令assignin对变量跨空间赋值，其中使用全局变量的方法最为简单。

两类程序文件的区别归纳总结如下：

	脚本文件	函数文件
格式	任意	必须以function开头
功能	没有输入输出变量	有输入输出变量
内存	与指令窗口共用	独立函数空间
调用法	文件名不可加变量	文件名可加变量
内部子程序	无	有(即子函数文件)

示意图1.6说明了两类程序文件对内存空间的使用及变量的位置，脚本文件的变量是存放在指令窗口的内存中，而每个函数文件都会生成一个独自的内存空间用以存放自己的变量。

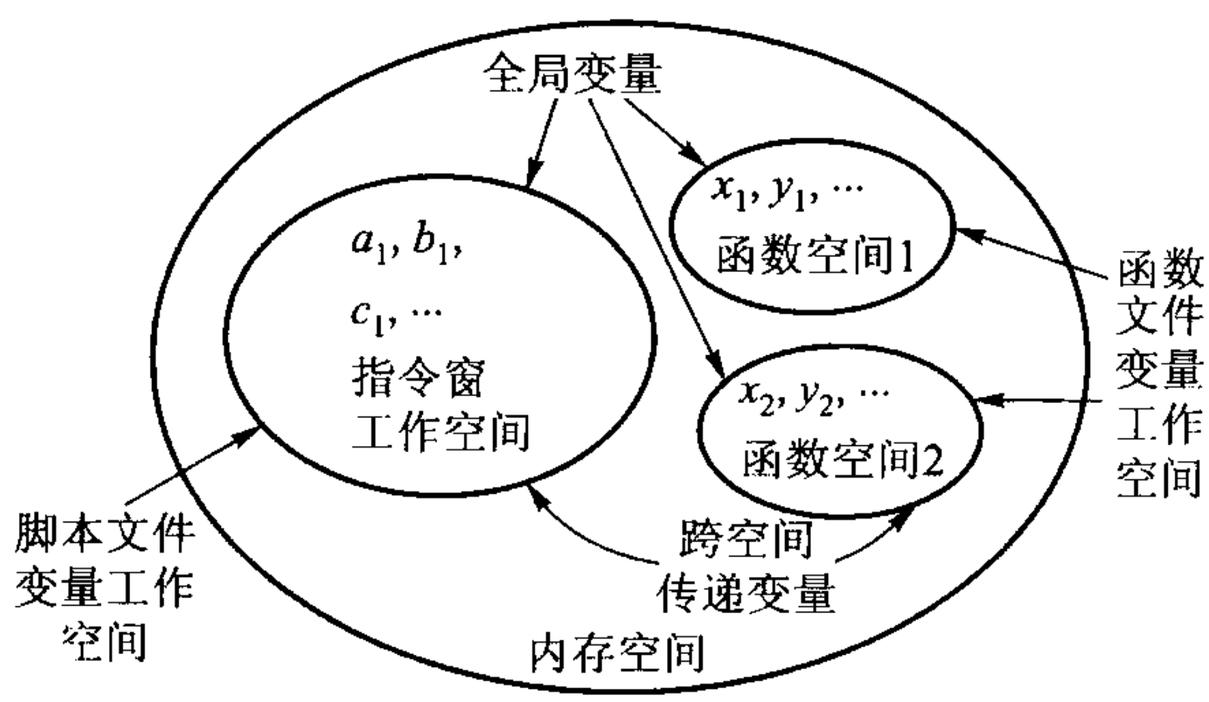


图 1.6　内存空间分配及变量存储位置示意图

§1.3.5　流程控制

程序中语句一般是按语句的先后顺序执行，要想使程序中的语句按规定的次序执行，就要使用流程控制语句。常用流程控制有循环结构和分支结构，这些结构都可以嵌套使用，分述如下。

§1.3.5.1　循环结构

循环结构有两种，一种是已经知道循环的次数，那么就可以用下面第一种结构的形式来控制称之为 for 循环，另一种是不能事先确定循环的次数，只能设定中止循环的条件，那么就用第二种循环来控制称之为 while 循环。

1. for 循环语句

使用for循环的语句结构如下：

for　循环变量=起始值: 步长: 终止值

　　循环体

end

步长的默认值为1，可以取正值或负值。取正值必须起始值小于终止值，取负值则要求起始值大于终止值。

在很多情况下，for循环可以用矢量或矩阵来进行操作，这是MATLAB提高运算速度的关键，称之为矢量化编程。例如编写一个对数表，不恰当的做法是编写如下程序：

```
x=0;
for  k=1:1001
     y(k)=log10(x);
     x=x+0.01;
end
```

恰当的程序如下，它不仅简洁而且运算速度快。

```
x=0:0.01:10;        y=log10(x);
```

下面是for循环和矢量化编程应用的实际例子。

例12 画图形$\sin n\pi x(n=1,2,3,4;\ \ x=0:0.001:1)$。

```
x=[0:0.001:1];                                  %变量x是矢量
y=sin(pi*[1:4]'*x);                             %函数值y是矩阵
plot(x,y)                                       %用y的每行对x作图(新用法)
```

在本例中，计算本来需要对x,n分别作for循环计算，即是二重循环，而在这里是用矩阵运算来替代。图1.7为所画图形。

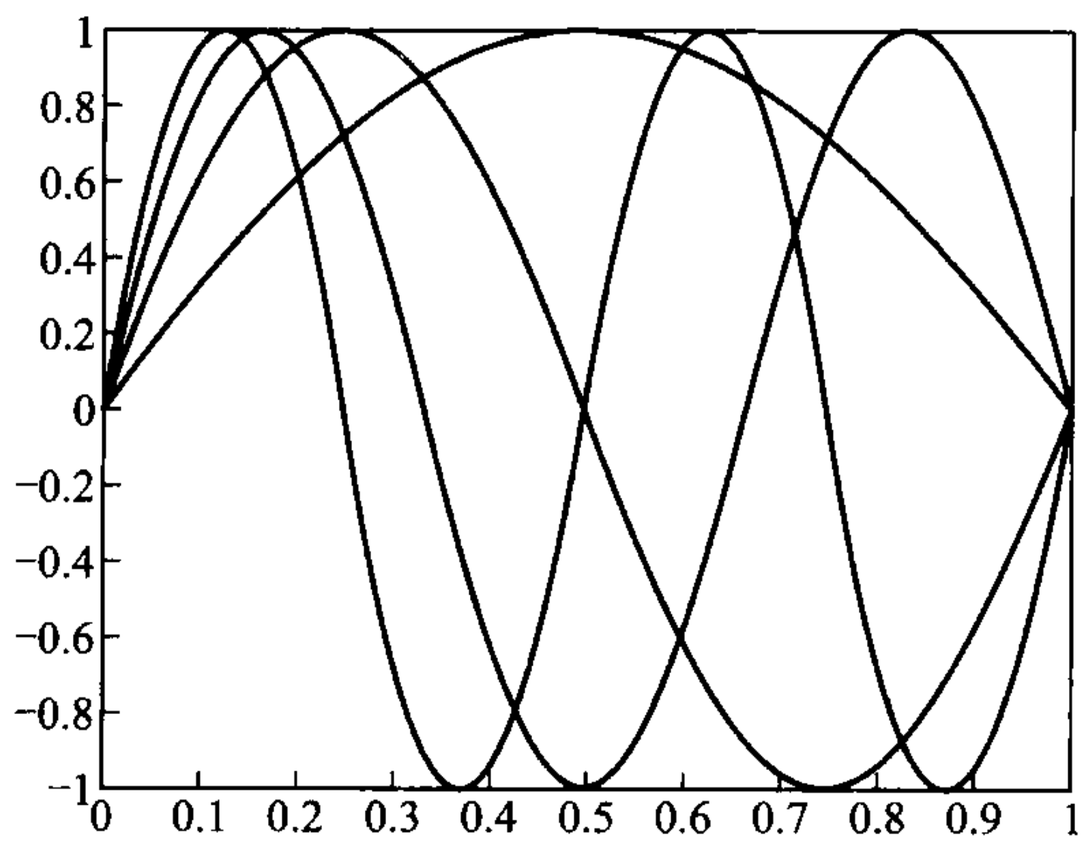


图 1.7 四条正弦曲线

例13 双缝干涉实验装置如图1.8所示。波长为λ振幅为A_0的单色光通过间距为d的两个狭缝，在屏幕上形成干涉条纹。两束光在屏幕上的光程差为

$$\Delta L=L_1-L_2=\sqrt{\left(y-\frac{d}{2}\right)^2+z^2}-\sqrt{\left(y+\frac{d}{2}\right)^2+z^2}$$

形成的相位差为$\theta = \Delta L\dfrac{2\pi}{\lambda}$，在屏幕上干涉光强度为$4A_0^2\cos^2\dfrac{\theta}{2}$。

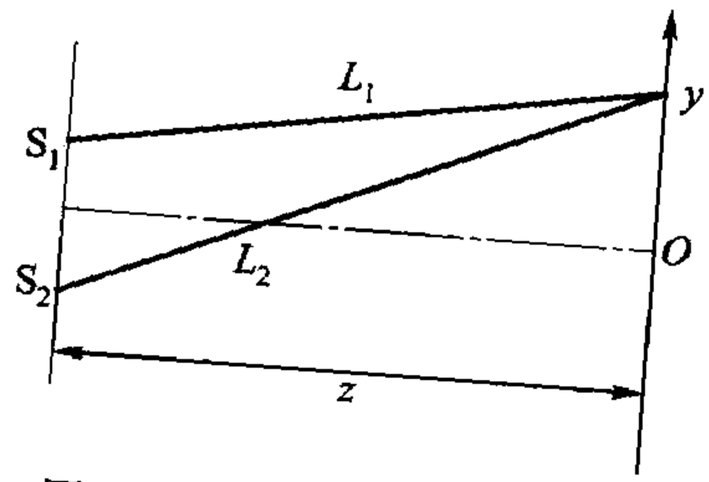


图 1.8　双缝干涉实验示意图

计算程序如下，所得图形如图1.9所示。

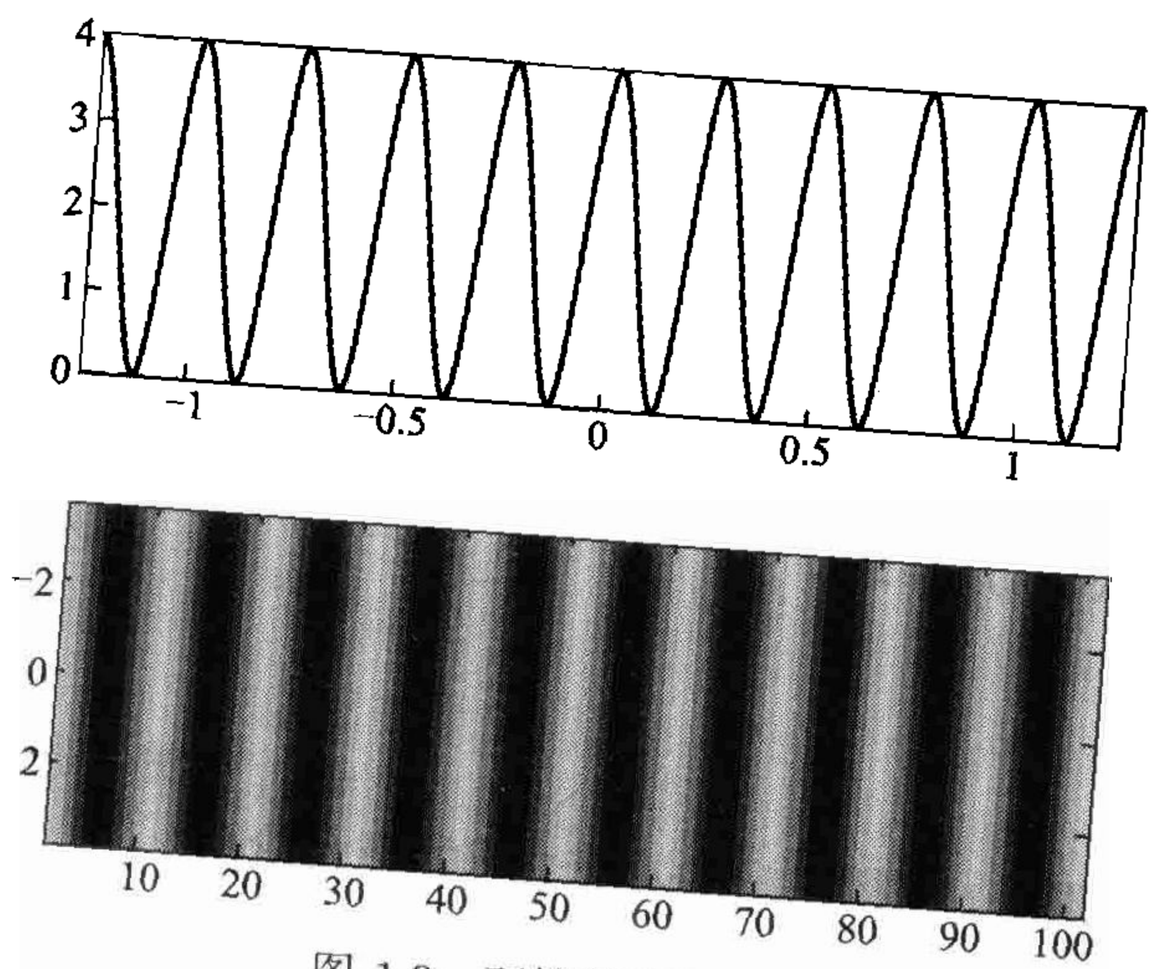


图 1.9　双缝干涉图像

指令colormap是表示颜色与图像的数值之间的对应关系，参数可取gray，hot，cool，bone，copper，pink，flag，prism，jet 等项，在 Demo 程序中目录 MATLAB/Graphics/ Images and Matrices的演示可以看到这些指令的颜色显示效果。指令image是用颜色显示矩阵元素而形成图像。

```
clear all
ym=1.25;                          %屏上观测点离中心的最大距离
y=linspace(-ym,ym,101);           %屏上取n个观测点
d=2;                              %两个光源距离
z=1000;                           %屏到缝的距离
lambda=5e-4;                      %设置光的波长
L1=sqrt((y-1).^2+z^2);            %屏上一点到两个光源的距离
```

```
L2=sqrt((y+1).^2+z^2);
phi=2*pi*(L2-L1)/lambda;                    %计算相位差
I=4*(cos(phi/2)).^2;                        %利用相位差计算光强
subplot(2,1,1)
plot(y,I)                                   %用曲线表示干涉条纹
axis([-1.25  1.25  0  4])
subplot(2,1,2)
B=I*255/5;                                  %定标取255个级别,使I/5对应最亮
image(B);                                   %以图案表示干涉条纹
colormap(gray(255));                        %用灰度级别显示图案
```

2. while循环语句

使用while循环结构的语句如下：

while　表达式

**　　循环体**

end

例如要找出小于10^{10}的最大阶乘，用的程序是

```
n=1
while prod(1:n)<1e10
   n
   n=n+1;
end
```

屏幕上显示出结果是13，即13！是小于10^{10}的最大阶乘。

§1.3.5.2 分支结构

分支结构是通过条件控制来选择要执行的语句，常用的有 if 语句与 switch 语句。

1. if 语句

首先介绍 if 语句，它有三种结构，就是单分支结构、双分支结构和多分支结构。

单分支结构如下，满足条件表达式时执行相应的语句，否则跳过执行其后的语句：

if　条件表达式1

**　　语句体**

end

双分支结构如下，满足条件表达式1执行语句体1，否则执行语句体2：

if　条件表达式1

语句体1

else

语句体2

end

多分支结构如下，可以根据需要的条件数目规定应执行的语句内容：

if　条件表达式1

语句体1

elseif　条件表达式2

语句体2

……

elseif　条件表达式n

语句体n

else

语句体n+1

end

例如可以用 if 语句来建立分段函数

$$y=\begin{cases} x^2 & (0\leqslant x) \\ -x^2 & (x\leqslant 0) \end{cases}$$

程序如下：

```
x=input('请输入x的值，x= ')
if x>=0
   y=x^2
else
   y=-x^2
end
```

这里的 input 是要求键盘输入的指令，详细用法见本节后面的介绍。

2. switch 语句

语句 switch 结构都是多分支的，根据判断表达式的值不同来执行不同的语句。

switch　判断表达式（标量或字符串）

case　值1

语句1

case　值2

语句2

……

otherwise

语句3

end

例14 编一个程序，根据输入的分子种类画出Maxwell速率分布曲线。Maxwell速率分布函数是

$$f(v)=4\pi\left(\frac{m}{2\pi\kappa T}\right)^{3/2}v^2\exp\left(\frac{-mv^2}{2\kappa T}\right)$$

其中κ=1.381e-23，$0\leqslant v\leqslant 1500$，$T$=300。

考虑氮分子和氧分子，它们的摩尔质量 mu 分别是 28e-3 kg·mol^{-1} 和 32e-3 kg·mol^{-1}，相应的分子质量是 m=mu/NA，NA=6.022e23 是阿伏伽德罗常量。

实际程序如下：

```
clear all
disp('此程序计算室温下Maxwell分布律')
y=input('输入一种分子名称:  N,  O','s')
switch   y
    case ( 'N' )
         mu=28e-3;
    case( 'O' )
         mu=32e-3;
    otherwise
    warning('只能输入大写N,O')
end
T=300;   v=0:1500;
k=1.381e-23;   NA=6.022e23;
m=mu/NA;
f=4*pi*(m/(2*pi*k*T))^(3/2)*exp(-m*v.^2./(2*k*T)).*v.^2;
plot(v,f)
```

这里的disp和warning的解释见本节后面的介绍。程序在调用时按照屏幕显示的提示输入分子种类N或O才能得到结果，如果输入不当则会提示错误。图1.10中所画是氮分子的速率分布。

初学者感到疑惑的是，这几种循环语句的功能很相近，有时为了完成一种计算可以采用其中任何一种语句，其实这其中还是有优劣之分。例如想要做n次循环，可以用语句

```
for k=1:n  ...... end
```

也可以用

```
while  k<n  ......, k=k+1, end
```

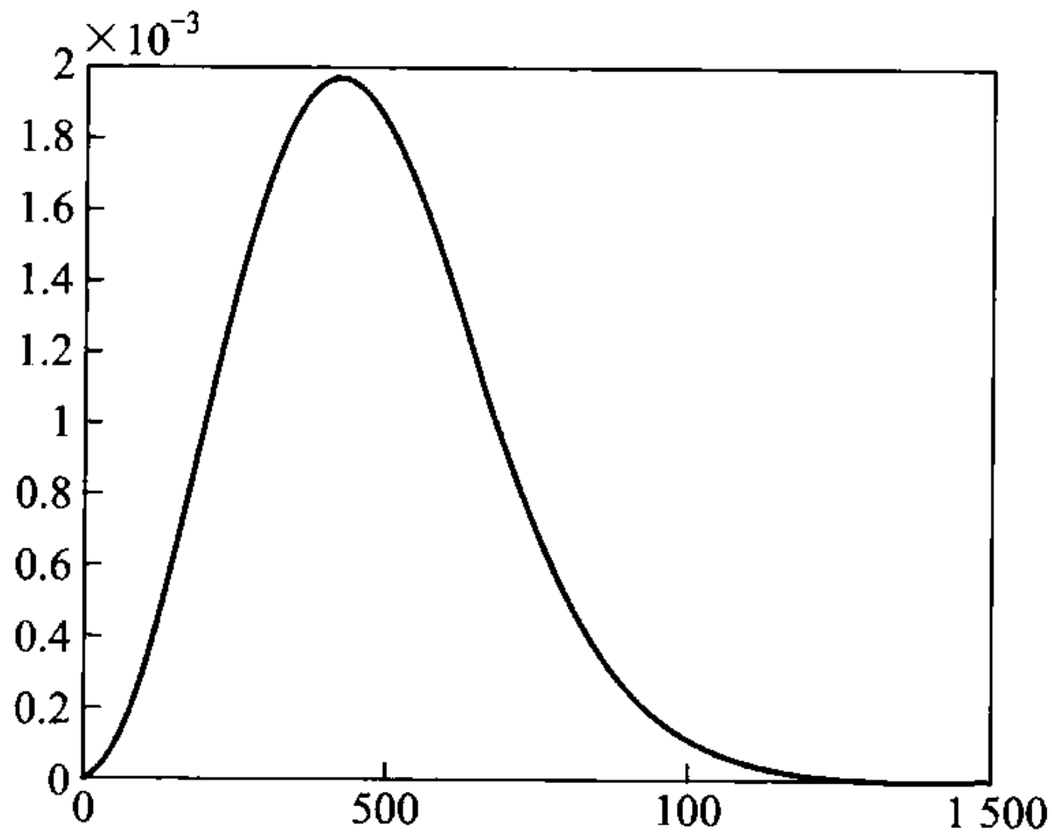


图 1.10　Maxwell速率分布曲线

但是后者由于多一次计算，所以不如前者好。所以在程序中应该仔细地选用效率高的循环语句。

§1.3.5.3　其他指令

还有一些指令也可以控制程序中语句执行状态，下面简单列举如下，有关用法可以参看help帮助系统。

input(' … ')	显示引号中提示文字，并将键盘输入的数值、字符串或表达式赋予指定的变量。
input(' … ', ' s')	将键盘输入的内容一概作为字符串赋予指定的变量。
disp(' … ')	在屏幕上显示引号中的内容。
pause	使程序运行暂停，按任意键恢复运行。
pause(n)	使程序运行暂停n秒。
break	用于for，while，if和try语句的终止，如果循环是嵌套的，则只从最内的一层退出。
continue	在for 和while循环中跳过其后的指令，执行下一个循环。
error	终止程序，显示出错信息。
keyboard	暂时终止程序的运行，等待键盘输入的指令，在执行完输入的指令以后，只要键入return，又可以恢复程序的运行。
return	停止执行return所在的程序，回到指令窗口或调用它的主程序。
warning	显示警告信息，继续运行程序。

§1.3.6 数据输入与输出

MATLAB 常用 save 和 load 两个指令与外部环境交换数据。

1. 指令 save 将内存中的变量存入文件

语句格式：

save fname	将工作内存的变量存入二进制文件filename.mat中。
save fname X Y Z	存储变量X，Y，Z。可以用通配符*。
save fname X Y Z -ascii	存入8位的ASCII文件。
save fname X Y Z -ascii -double	存入16位的ASCII文件。
save fname X Y Z -ascii -double -tabs	列表式的文件。
save fname X Y Z -append	将新的变量加入到已有的文件中。

2. 指令load将文件数据读入内存

语句格式：

load filename	从文件filename.mat中读入数据。必须指明有关的路径。
load filename x, y, z,⋯	只读入指定的变量x，y，z，⋯。变量名也可以用通配符。
load filename.ext	读入ASCII码的分行形式的数据，文件中可以包含以% 开头的MATLAB的命令，变量名与文件同名。
load filename -ascii(or -mat)	按ASCII码或二进制读入文件数据。

指令load也可读入包含数值数据的文本文件。文本文件应该列成数据的表，同行中的各列用空格分开，每行中的元素个数相同。例如用任何编辑器编制如下数据表

```
16.0    3.0    2.0   13.0
 5.0   10.0   11.0    8.0
 9.0    6.0    7.0   12.0
```

并将它存入d盘的mag.dat文件。要将它读入工作空间，可键入命令

```
>> load d:\mag.dat
```

就会将数据文件读入内存并建立变量mag，它代表文件中的矩阵。

§1.4 作　图

§1.4.1 作图功能概述

这一节是本章中内容较多的一节，正是这些丰富多彩的作图功能更有力的展示了MATLAB的魅力。

无论在科研或教学上，为了更好地分析计算结果，都希望将计算结果可视化，而MATLAB完善与强大的作图功能，正好满足了这种需求。用语言作计算时，通常要在计算完成之后使用专门的作图软件去画图，而MATLAB的作图功能已经达到了作图软件的水平，能将计算结果直接作图输出，或者做成动画和电影输出，这就是使用MATLAB作计算的又一显著优点。

MATLAB可以用指令完成各种二维图和三维图，包括完成一些特殊图形，如对数图、复数图、极坐标图、矢量场图、等高线图等，可以用数据直接作图，也可以用函数作图，还可以对隐函数和参数函数作图。可以在二维图上用颜色表示三维数据，也可以在三维图上用切片和颜色来表示四维图形。对三维空间的线，面和体都能很容易地画出来。对于物理场中矢量场也有丰富的表现方式，如用流线图来表现矢量场整体的情况、用锥体图表示各点场强的大小与方向、用流管图中流管的粗细变化表现散度的变化、用流带图的扭转来表现各处旋度的变化等。

MATLAB可以对画好的图形进行再编辑，图形窗口不仅能显示图形，也能编辑图形。它可以对图形的全部属性进行编辑修改，如增加标题和说明性文字、可以像相机一样来改变观察图形的视角与位置、画线条和箭头、将图形放大缩小、改变线型、旋转三维图形等；也可以编辑图形对象的其他属性如颜色、位置、粗细等。

MATLAB有制作不同动画的功能，如电影动画与实时动画。还可以制作用户界面，使得编成的程序更方便实用。

MATLAB画出的图形不仅可以用MATLAB自身默认的图形文件格式fig文件存盘或打印输出，更可以用多达数十种的图形文件格式如eps，bmp，jpg，tif等输出，这些都是科研工作者投稿时常用的图形文件格式，而为了实现这些任务用户只要在图形窗口中的菜单File下选择Export，再找到相应的文件格式即可。

快捷地了解MATLAB的作图功能与指令的方法是浏览MATLAB的作图功能演示。在指令窗口键入demos，打开功能演示窗口，在左边的栏目中选择 MATLAB 目录，在这个目录下面的子目录Graphics，MATLAB/3-D Visualization 和 Gallery 都是作图功能演示，打开这些子目录，然后选择右边的各个程序，就可以看到作图功能的演示。这些演示实际上就是这些指令的使用格式，用户只要

将这些指令复制下来，将其中的参数改成自己的数据，就能画出类似的图形。所以MATLAB的作图功能不仅强大，而且使用简单，基本是将变量及函数值按规定格式输入即可，使用者并不需要把这些指令都记住，只要知道有哪些画图的指令，使用时查阅一些演示就行了。

MATLAB是在图形窗口中画图，任何作图指令都会自动打开一个图形窗口。如果已经有打开的图形窗口，则图形将画在这个窗口中，并默认用新图形替代旧图形。指令hold on可以在保留旧图形的基础上画出新图，而hold off则关闭这种功能。

也有专门用于打开图形窗口的指令，就是指令figure，使用时可以根据需要在指令后面的括号内填入必要的参数。

下面是一些常用指令：

figure	打开新作图窗口
figure(n)	打开第n个作图窗口
close figure(n)	关闭第n个作图窗口
subplot(m,n,p)	将窗口分成m×n个区,在第p区作图
hold on	在窗口中保留原图形,画上新图
hold off	关闭窗口保留原图形的功能

§1.4.2 二维曲线作图指令

在前面已经介绍过plot的用法，它是二维作图指令的一个典型例子。下面的指令都是画二维图形的指令:

plot	画一条或多条曲线	comet	画二维彗星曲线
bar	直方图	loglog	双对数坐标曲线
compass	原点出发的复数矢量图	pcolor	假彩色图
contour	在x y 面上的等高线图	polar	极坐标图
errorbar	误差棒图	quiver	矢量场图
ezplot	符号函数二维曲线	rose	统计频数扇块图
feather	沿x 轴分布的复数矢量图	semilogx	x 轴对数坐标曲线
fplot	数值函数二维曲线	semilogy	y 轴对数坐标曲线
fill	平面多边形填色	stem	火柴杆图
hist	统计频数直方图	stairs	阶梯图

这些指令用法基本相似，只要在指令后面输入相应的参数就可作出图形。所需要填写的参数可以用help系统查找。下面举两个例子。

例如polar是画极坐标图的指令，我们用它画出一个圆环在第一象限的图形。

图形由两段弧和两段直径组成，依次给出它们的极坐标，然后作图即可。对直径只要给起点与终点的坐标，而对于弧，则给出了10个点的坐标，这是为了使

曲线更光滑。所得图形为图1.11。

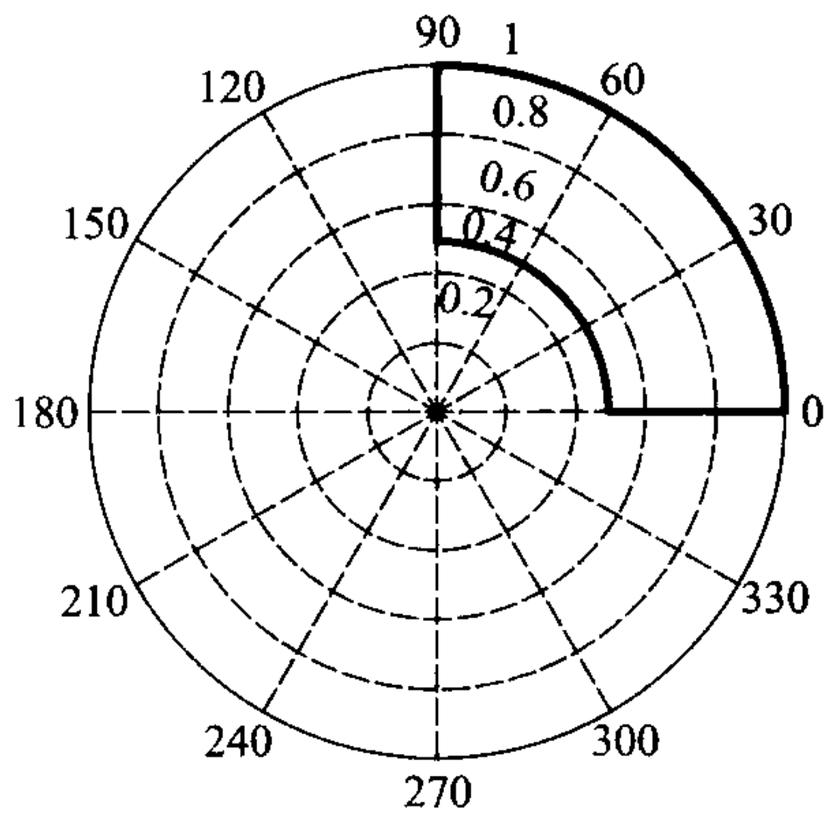


图 1.11　画圆环的四分之一图形

```
t=[0,0:pi/20:pi/2,pi/2:-pi/20:0]
r=[0.5,ones(1,11),0.5*ones(1,11)]
polar(t,r)
```

又如fplot是利用函数表达式直接画二维图形，不必要先计算函数值。更有趣的是，它绘图的数据点是用自适应法产生的，即在函数变化小的地方取较少的点，而在函数变化剧烈的地方取较多的点。这样图像就可以更好地反映函数的变化。

可以按下述方式运用指令：

fplot(FUN, LIMS, TOL)

其中FUN是作图函数，LIMS是变量及函数取值范围，TOL是相对误差，默认值为2e-3。如

```
>> fplot('sin(1/x)',[0.01 0.1],1e-3)
```

画出了函数sin(1/x)在[0.01,0.1]内的图像，其误差为10^{-3}。实际图像如图1.12，可以看出在变量值较小的地方，函数值变化较大，由于取点较多，所以图像仍可较好反映函数的变化。读者可以试用对变量x取等间距变化的值再用指令plot重画此图，就可看出两者的差别。

有一些二维的作图指令是以ez开头，它们其实是符号工具箱的作图指令，因此可以用函数直接画图，而无需先计算变量与函数的值，只需给变量的取值范围即可。这类指令有：

ezcontour	快速画等值线图
ezcontourf	快速画填色的等值线图
ezmesh	快速画网线图
ezmeshc	快速画网线及等值线图

ezplot3	快速画三维曲线图
ezpolar	快速画极坐标曲线图
ezsurf	快速画曲面图
ezsurfc	快速画曲面及等值线图

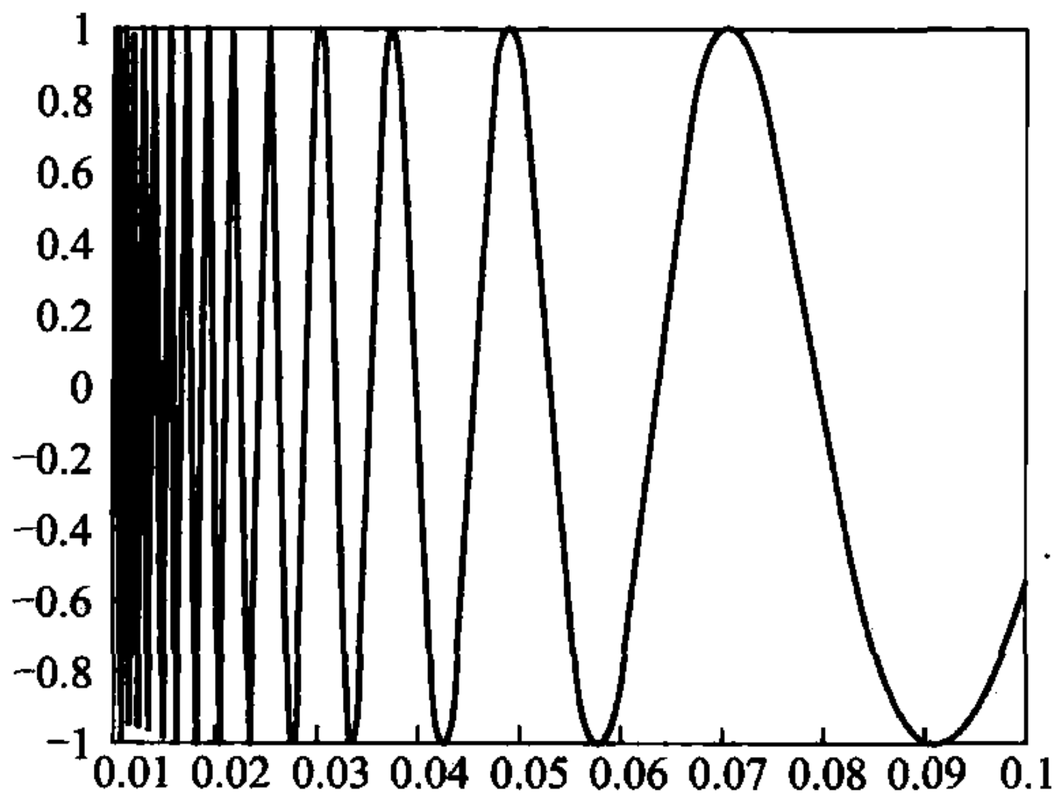


图 1.12 自适应法绘制的函数图像

例如ezplot就可以由函数表达式直接作图，而不必先算函数值,变量值的范围默认为$-2\pi < x < 2\pi$。用它画隐函数图形是很方便的。由help查询知道，如果要画$a < x < b$范围的函数图像，可按如下格式使用：

ezplot(f, [a,b])

下面是一些例子，所画图形都极简单，故不在此画出。

```
>>ezplot('cos(x)')                    %画-2*pi<x<2*pi的cos(x)的图像
>>ezplot('cos(x)', [0, pi])            %画0<x<pi的cos(x)的图像
>>ezplot('x^2+y^2-1',[-1.2,1.2]);      %用隐函数画圆,-1.2<x<1.2
>>ezplot('sin(t)','cos(t)')           %用参数函数画圆, -2*pi<t<2*pi
```

需要注意的是，对于用户自己建立的函数，其用法略有不同，例如：

```
>>f = @(x)cos(x)+2*sin(x);                          %建立用户函数f
>>ezplot(f)                              %用函数f画图, 注意没有单引号
```

§1.4.3 复数作图

用plot对复数作图有专门的规定。分两种情况，如果变量与函数都是复数，则作图时将虚部丢弃，只利用实部的数据作图。如果是对单个的复数作图。在这种特例下，作图指令plot表示用实部对虚部作图。因此plot(Z)表示plot(real(Z), imag(Z))，例如：

```
>> t = 0 : pi/10 : 2*pi;
>> plot(exp(i*t), '- o')
>> axis equal
```

画出了一个20条边的多边形，每个顶点有一个小圆圈。如图1.13所示。

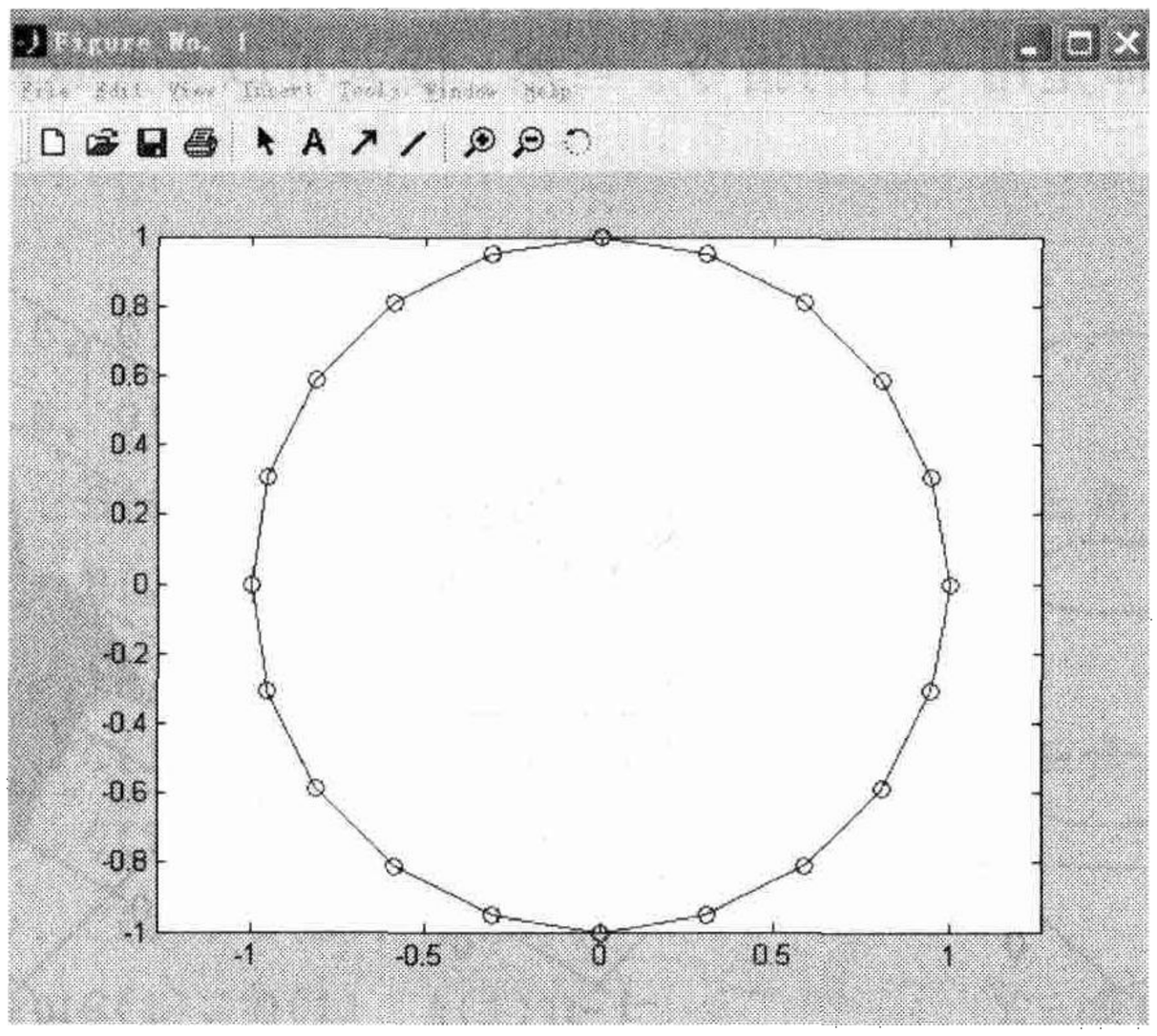


图 1.13　用复数画的多边形

又如，要画一个在任意位置的椭圆，利用复数的性质就很方便。先在原点画一个椭圆，然后旋转到需要的方向，对复数而言就是乘一个规定角度的指数项；然后将原点平移到新位置，对复数而言就是加一个复数。实际的程序如下。图1.14为所画图形。

```
the=linspace(0,2*pi,100);
A=2*cos(the); B=sin(the);
C=A+i*B;
C1=C*exp(i*pi/3);
C2=C1+(3.5+2i);
plot(C),
hold on
plot(C1)
plot(C2)
axis equal
```

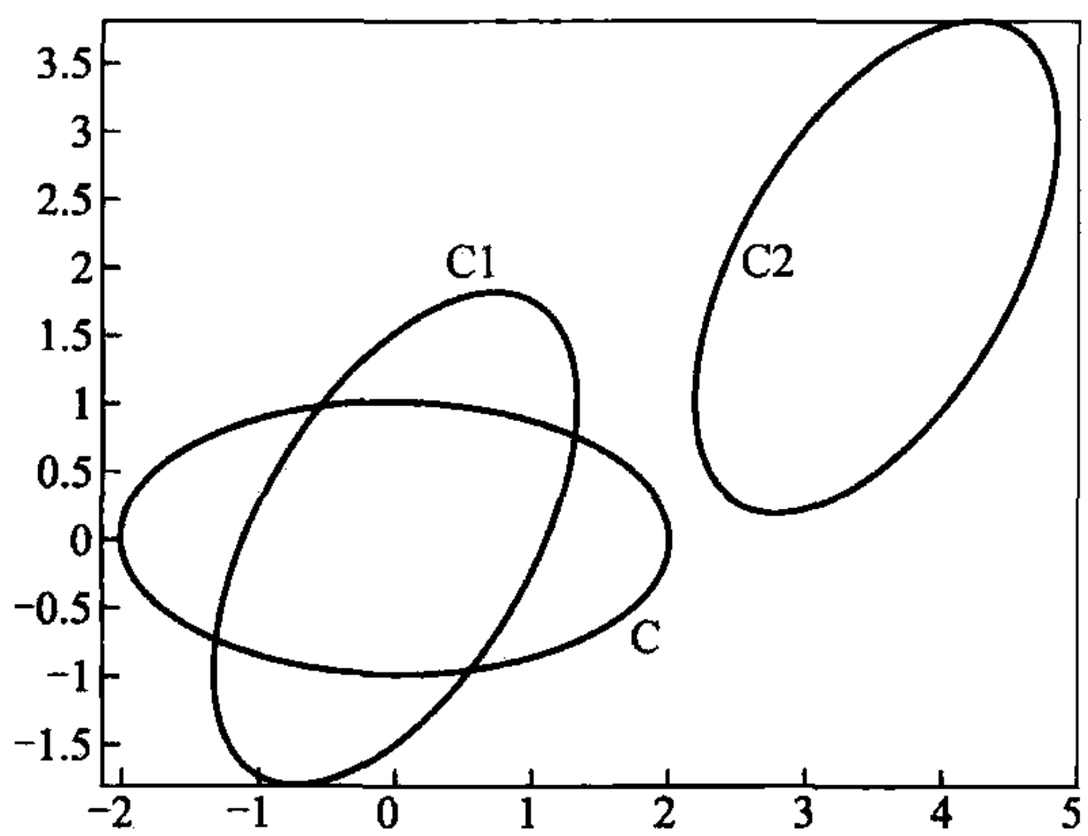


图 1.14　画椭圆

这里要注意的是，每次只能画一个椭圆，如果将语句写成plot(C1,C2)，画出来的将是一条直线。

§1.4.4　基本的三维图

三维图形包括了空间曲线、空间曲面和空间立体图形如球体、柱体等，

如图1.15所示。它们的画法指令各有不同。从下面例子会看到，空间曲线是利用参数方程来定曲线的点的三个坐标；而画空间曲面则要先建立数据网格才能作图；画旋转体是用专门的指令。当然，读者可以参照这些指令的程序编写自己的程序。

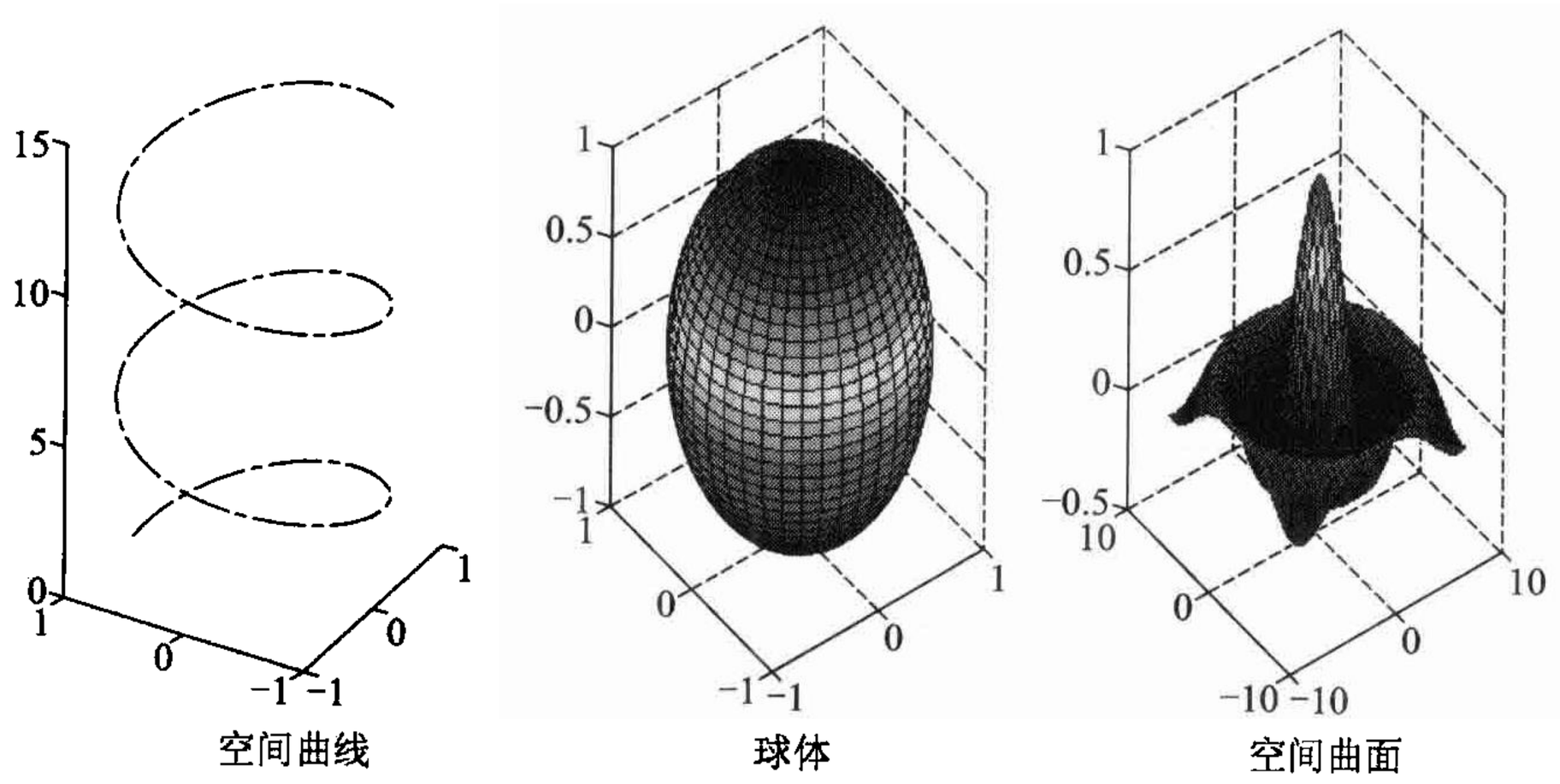


图 1.15

图1.15中三个图形的作图程序如下：

```
subplot(1,3,1),                                          %画三维曲线
t = 0 : 0.01 : 15;  x = sin(t);  y = cos(t);  z = t;
plot3(x, y, z, '-.')
view(-60, 18)                                            %选择视角
subplot(1,3,2)                                           %画旋转体图形
sphere(40);
subplot(1,3,3)                                           %画三维曲面
[X,Y]=meshgrid(-8:0.5:8);                                %构造数据网格
R = sqrt(X.^2 + Y.^2)+eps;   Z =sin(R)./R;               %构造函数
surf(X,Y,Z)
```

上述程序中的指令介绍如下。

1. 画空间曲线

画空间曲线的指令为plot3，用法如下：

plot3 (x,y,z,s)　　x,y,z 是矢量，表示点的空间坐标，s是线型

如果在上述程序中将指令plot3语句替换成comet3(x,y,z)，就会看光点沿空间曲线移动，产生动画效果。

利用 plot3 指令可以画空间图形，下面画一个四面体，顶点在 A (0,0,0)；

B (1,0,0)；C (0.5, 0.5, 0.5)；D (0.5, 0.5, 0)。下面的程序，程序中分三次画三个空间三角形 ABC，BCD 和 ACD，将它们的空间坐标依次列出，最后用plot3画图。图1.16为所画图形。

```
A=[0,0,0];                                          %四个顶点坐标
B=[1,0,0];
C=[0.5, 0.5, 0.5];
D=[0.5, 0.5, 0];
Xabc=[A(1), B(1), C(1), A(1)];                      %三角形ABC的顶点坐标
Yabc=[A(2), B(2), C(2), A(2)];
Zabc=[A(3), B(3), C(3), A(3)];
Xbcd=[B(1), C(1), D(1), B(1)];                      %三角形BCD的顶点坐标
Ybcd=[B(2), C(2), D(2), B(2)];
Zbcd=[B(3), C(3), D(3), B(3)];
Xacd=[A(1), C(1), D(1), A(1)];                      %三角形ACD的顶点坐标
Yacd=[A(2), C(2), D(2), A(2)];
Zacd=[A(3), C(3), D(3), A(3)];
plot3(Xabc, Yabc, Zabc,Xbcd, Ybcd, Zbcd,Xacd, Yacd, Zacd)
view(-44,-22)
```

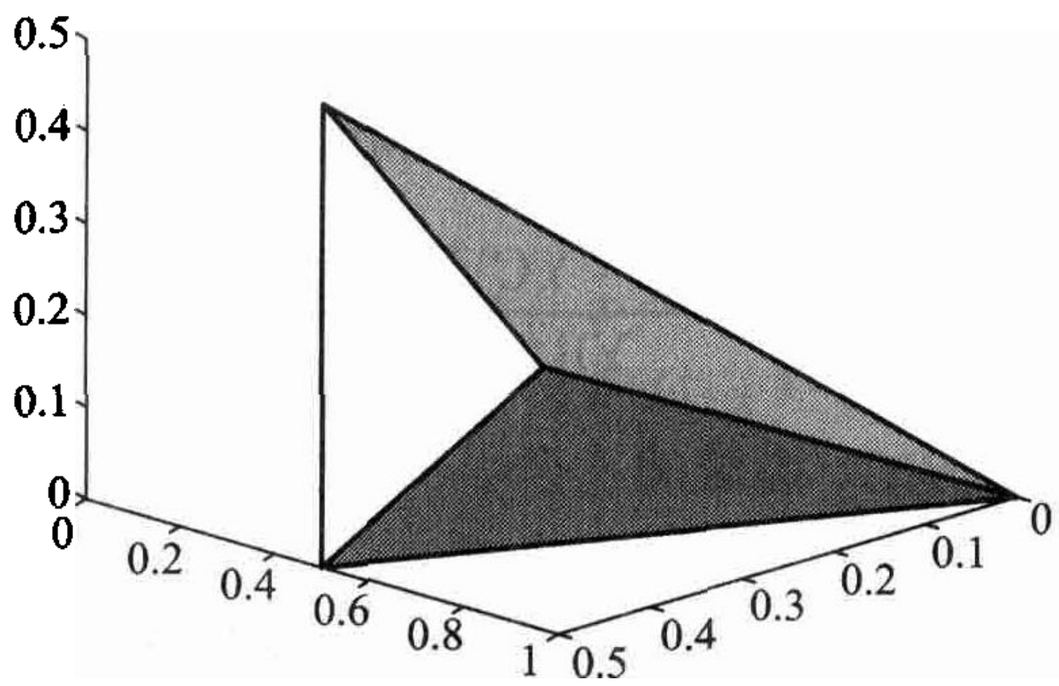


图 1.16　画四面体图形

将plot3的语句换成下面的语句，则会将四面体的三个面分别涂上绿色，红色和黄色。

```
hold on
fill3(Xabc, Yabc, Zabc,'g')
fill3(Xbcd, Ybcd, Zbcd,'r')
fill3(Xacd, Yacd, Zacd,'y')
```

程序中**view(az, el)**用来指明观察立体图形的视角，如图1.17所示。参数

az(Azimuth) 表示方位角，单位是度，取值为$-180 \sim 180$，计算起点是负y轴；el (Elevation) 表示俯视角，取值为$-90 \sim 90$，计算起点是xy平面。进行三维观察时的默认值是az=−37.5, el=30，进行二维观察时的默认值是az=0, el=90。在图形窗口中，用鼠标在工具栏中选择旋转图形的图标，再将鼠标指向图形，就会显示当前的视角。

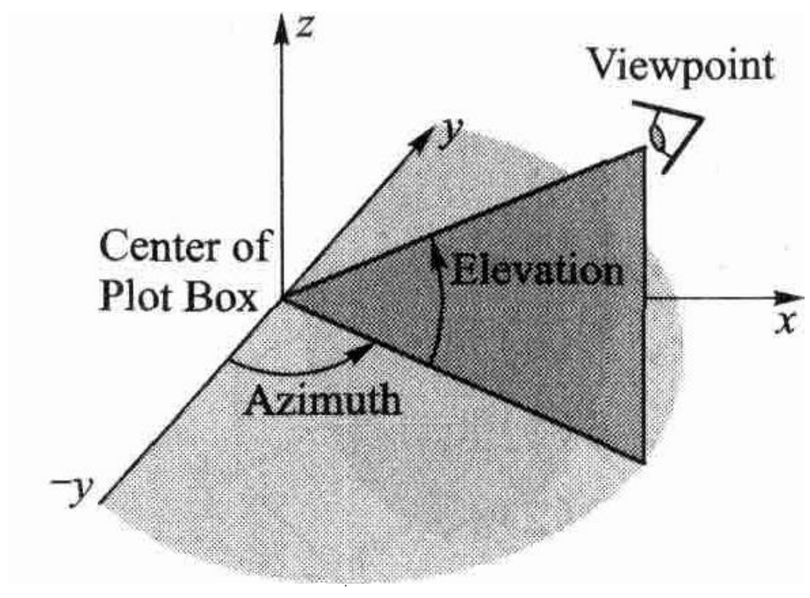


图 1.17 视角的示意图

2. 画旋转体表面

常用指令有：

sphere(N)	画半径为1，有N条经线的球。
cylinder(R,N)	画母线R旋转而成柱体，柱体上有N条母线。
ellipsoid(XC,YC,ZC,XR,YR,ZR,N)	画椭球，使其满足

$$\frac{(\mathrm{X}-\mathrm{XC})^2}{\mathrm{XR}^2}+\frac{(\mathrm{Y}-\mathrm{YC})^2}{\mathrm{YR}^2}+\frac{(\mathrm{Z}-\mathrm{ZC})^2}{\mathrm{ZR}^2}=1$$

椭球的球心在XC，YC，ZC，椭球的三个半长轴分别为XR，YR，ZR。

3. 画空间曲面

画空间曲面相对复杂一点，先要建立相应的数据网格，原因是：描述平面上所有的点，要给出点的坐标(x,y)，而x，y值都在一个平面内变化，所以描述x，y实际上是用了两个矩阵，即矩阵X和矩阵Y。这种矩阵的结构很特殊，X的每一列的值相同，Y的每一行的值相同，这种结构的两个矩阵就组成了MATLAB所谓的二维数据网格。在MATLAB中用指令meshgrid来生成矩阵网格。例如下面是x=1，2，3，y=4，5，6时生成的数据网格。

```
>> [X,Y]=meshgrid(1:3, 4:6)
X =  1     2     3        Y =  4     4     4
     1     2     3             5     5     5
     1     2     3             6     6     6
```

矩阵X和Y的对应元素给出一个点的坐标(x,y)。将这个值代入到表示空间曲面的函数中去，就对应曲面上一个空间点的高度z，而所有的z也组成了一个矩阵。

由此可见，画空间曲面会有三个步骤：构造数据网格，建立作图函数，用指令画图。画空间曲面的指令如下：

mesh	画网线图
meshz	画网线图再加基准平面
surfc	画表面图再加光照
meshc	画网线图再加等高线
surf	画表面图

在画空间曲面的时候已经用到指令surf，读者可以将其换成这里的其他指令以体会它们之间的区别。如前所述，读者还可以查看MATLAB中的演示程序来了解这些指令。

§1.4.5 四维数据的表现——切片函数slice

三元函数的值是第四维数据，比如物理场中的标量场在空间各点的值是三维空间坐标的值。我们表示它们的办法是用直角坐标表示三维空间坐标，而第四维数据即场的值用颜色表示，然后观察它的空间剖面图。使用的指令是切片函数slice。例如为了表现函数

$$v = x\mathrm{e}^{-x^2-y^2-z^2}$$

利用指令slice作了四个截面，分别是$x=5, x=15, y=15, z=10$。指令colorbar用以显示水平方向和垂直方向的颜色与数值的对应关系。图1.18为所画图形。

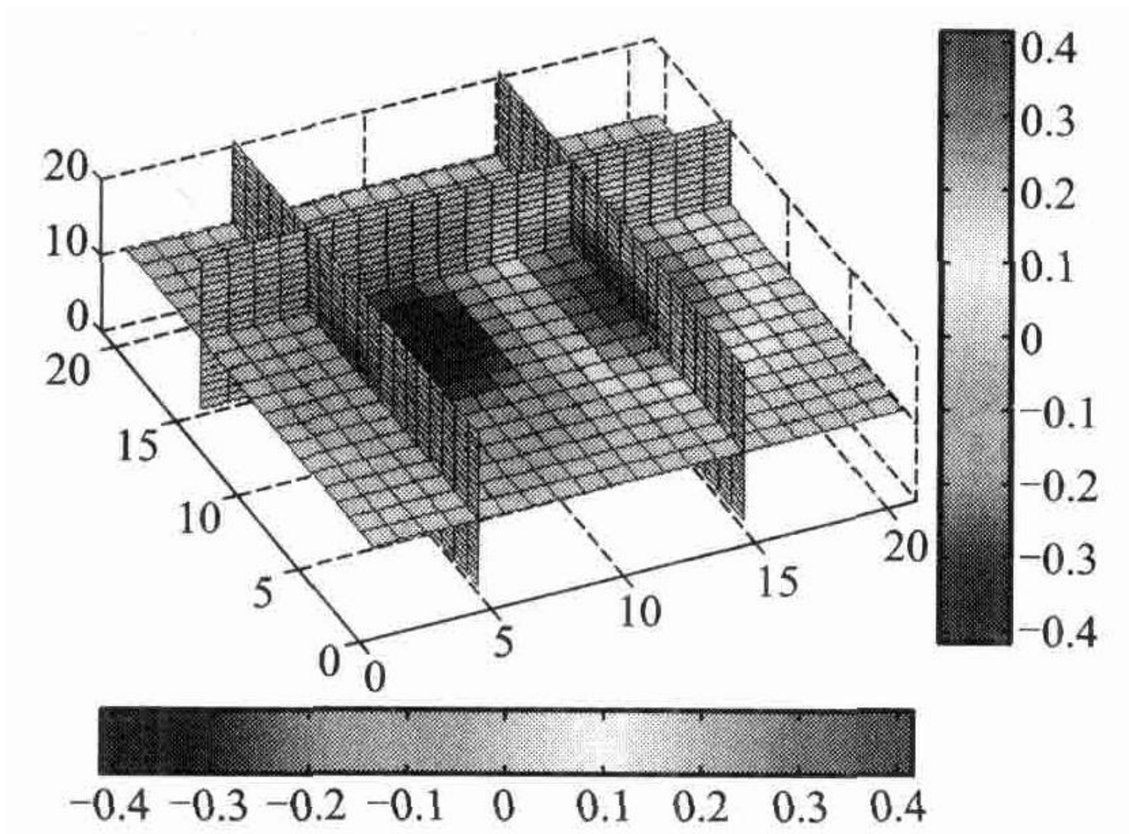


图 1.18　四维数据的截面图

```
[x,y,z] = meshgrid(-2:.2:2);
```

```
v=x.*exp(-x.^2-y.^2-z.^2);
slice(v,[5 15],15,10)
axis([0 21 0 21 0 21]);
hold on
colorbar('horiz')
colorbar('vert')
view([-25 65])
```

图形中由于没有给坐标轴标定范围，所以是以元素序号来标识坐标轴的。

§1.4.6 复变函数图形

复变函数的变量与函数值都是复数，所以是一种四维数据。MATLAB有以复数为变量画复函数图形的专门命令，方法是以x, y作为复变量所在的复数平面，以z轴表示复变函数的实部，以颜色表示复变函数的虚部。

如指令cplxgrid(m)在复变量平面的单位圆内绘制$(m+1)\times(2m+1)$极坐标的数据网格，指令cplxmap(u,f(u))绘制复变函数图形，指令cplxroot(n)绘制复数n次根的图形。如下面两个例子。

复数的平方u^2和方根$u^{1/2}$的画法如下。图形分别是图1.19和图1.20。

```
>> u=cplxgrid(20);
>>cplxmap(u,u.^2)
>>colorbar('vert')
```

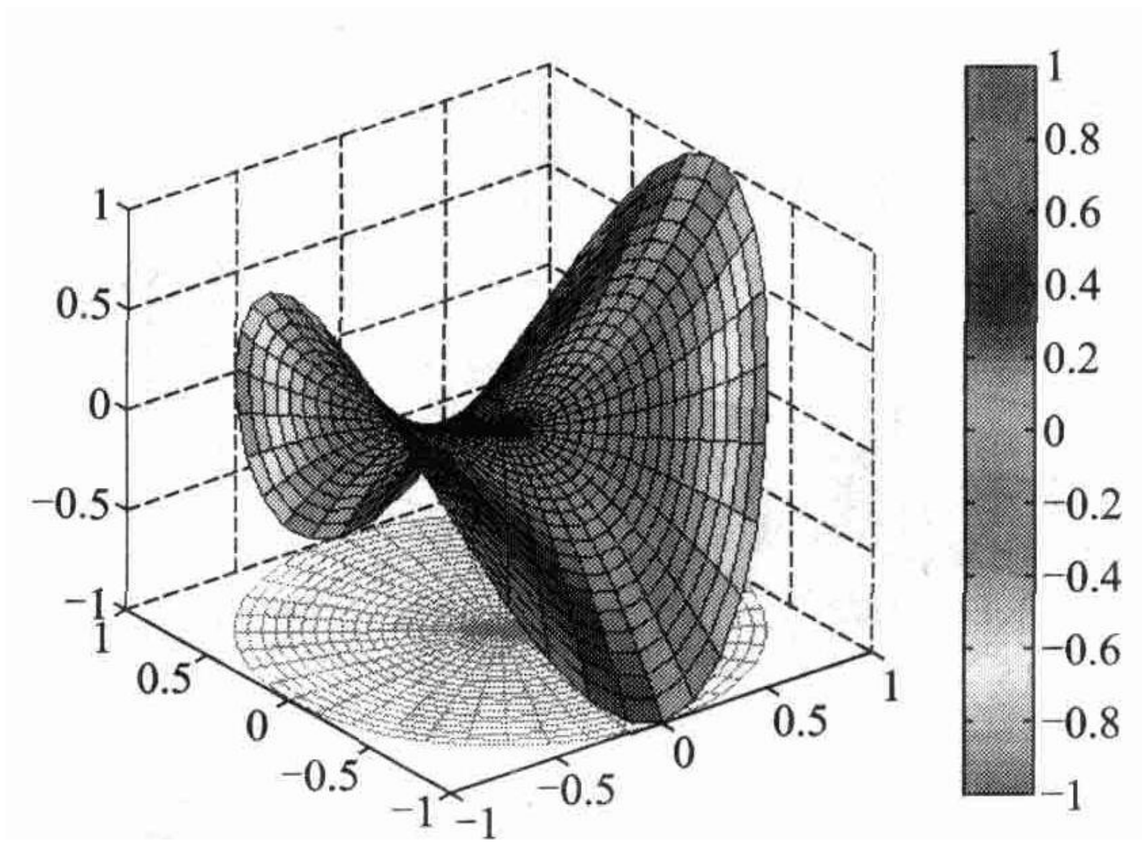


图 1.19 复变函数u^2的图形

```
>>u=cplxgrid(20);
>>cplxroot(2)
>>colorbar('vert')
```

原图为彩色,图中的色标显示了颜色与数据的对应。下面分析一下图1.19，x, y 平面上的单位圆实际就是$u = \mathrm{e}^{\mathrm{i}\theta}$的图形，$z$轴是$u^2$的实部，颜色是$u^2$的虚部，其值可以根据色标来判断。

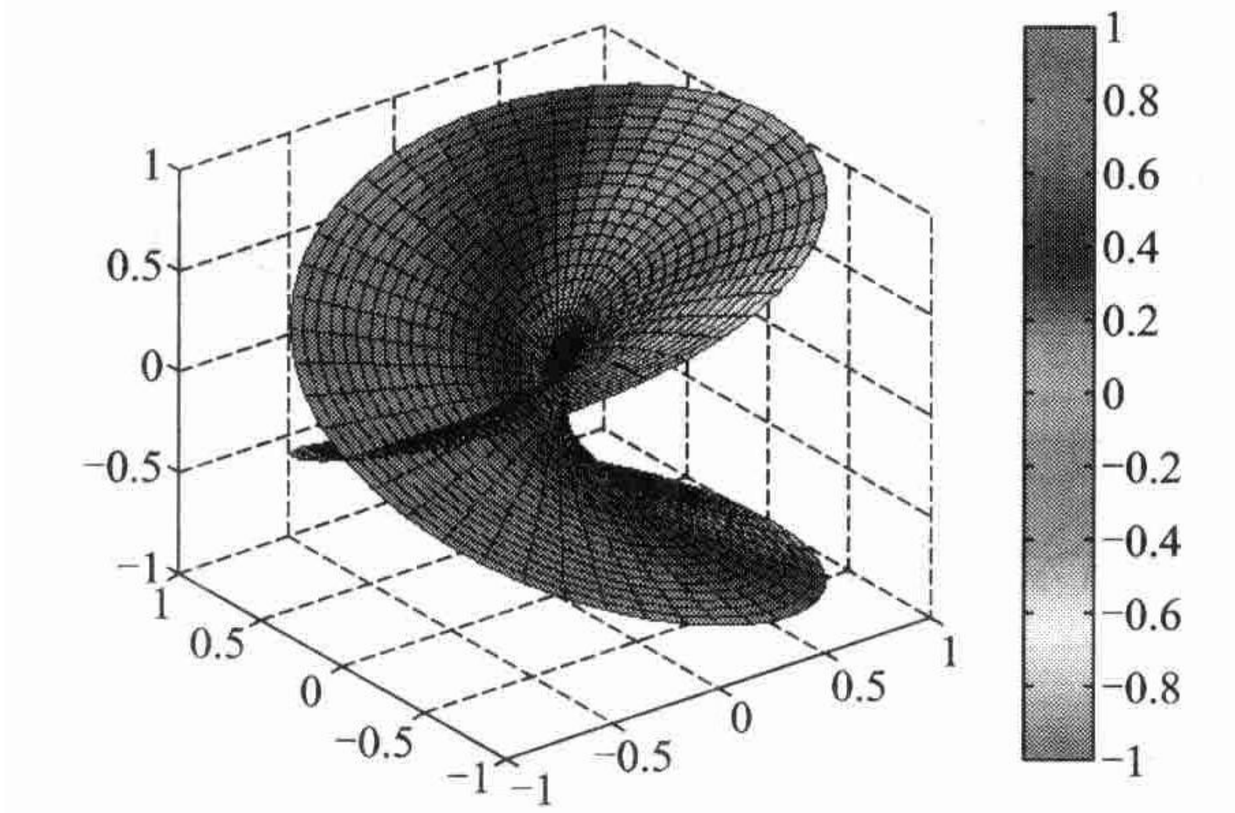


图 1.20　复变函数$u^{1/2}$的图形

为了分析函数实部与虚部的变化，令

$$(\mathrm{e}^{\mathrm{i}\theta})^2 = \cos 2\theta + \mathrm{i}\sin 2\theta$$

取$0 \leqslant \theta \leqslant 2\pi$，将$\cos 2\theta$和$\sin 2\theta$画在同一个图中，如图1.21所示。从图中可以看出，实部$\cos 2\theta$有两个极大和两个极小，两个极大在$\theta = 0$和$\theta = \pi$处，而u^2的图形也显示了两个极大和两个极小，两者是一致的。

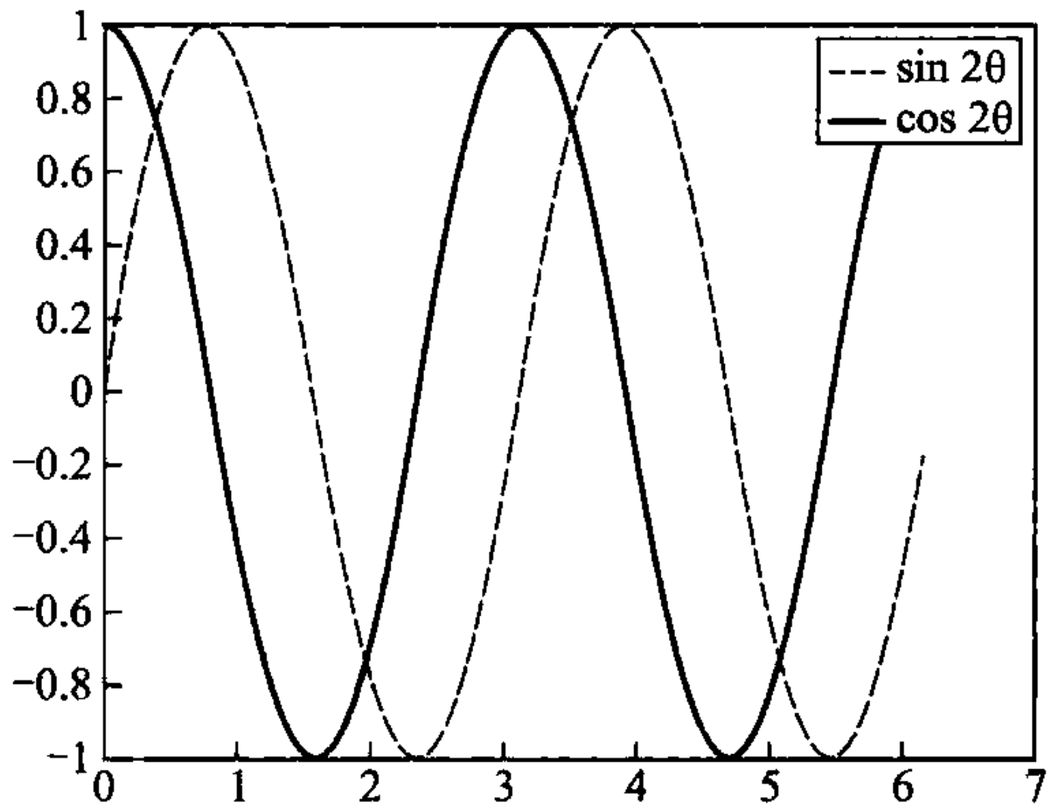


图 1.21　复变函数u^2的实部和虚部

同理，对复变函数$u^{1/2}$的图形可以作类似的分析，一个明显的结论是，在$0 \leqslant \theta \leqslant 2\pi$的范围，函数的实部的有两个值与其对应，显示一种两叶的结构。

§1.4.7 动画

§1.4.7.1 影片动画

将一幅一幅的图形制作好，放在内存中，然后像电影一样连续播放，称之为影片动画。所用到指令如下:

M=moviein(n)	定义矩阵M，以存储n帧画面，每列对应一帧
M(:, j)=getframe	依次将n帧画面存入M
movie(M,k,FPS)	将n帧画面演示k 次，每秒FPS帧

例15 行波的影片动画程序如下：

```
%wave1.m
t=0:pi/20:4*pi;  x=0:0.1*pi:4*pi; m=moviein(40);
for i=1:40
   y=sin(x-t(i));  plot(x,y);  axis([0,12,-1,1]);
   m(:,i)=getframe;
end
movie(m,3,30)
```

如果要将动画转为*.avi文件，可将程序改为

```
%wave1.m
t=0:pi/20:4*pi;  x=0:0.1*pi:4*pi;
mov=avifile('test.avi'); mov=close(mov);
m=moviein(40);
for i=1:40
   y=sin(x-t(i));     plot(x,y);   axis([0,12,-1,1]);
   m(:,i)=getframe;
end
movie(m,3,30)
movie2avi(m,'test.avi')
```

这样生成的文件在当前工作目录下，文件名是 test.avi。可用媒体播放器如 Windows Media Player 播放。

这种二维的动画可以用三维的图像来表示，程序如下，所得图形为图1.22。

```
t=0:pi/2:4*pi;
x=0:0.1*pi:6*pi;
[T,X]=meshgrid(x,t);
y=sin(X-T);
```

```
waterfall(T,X,y)
```

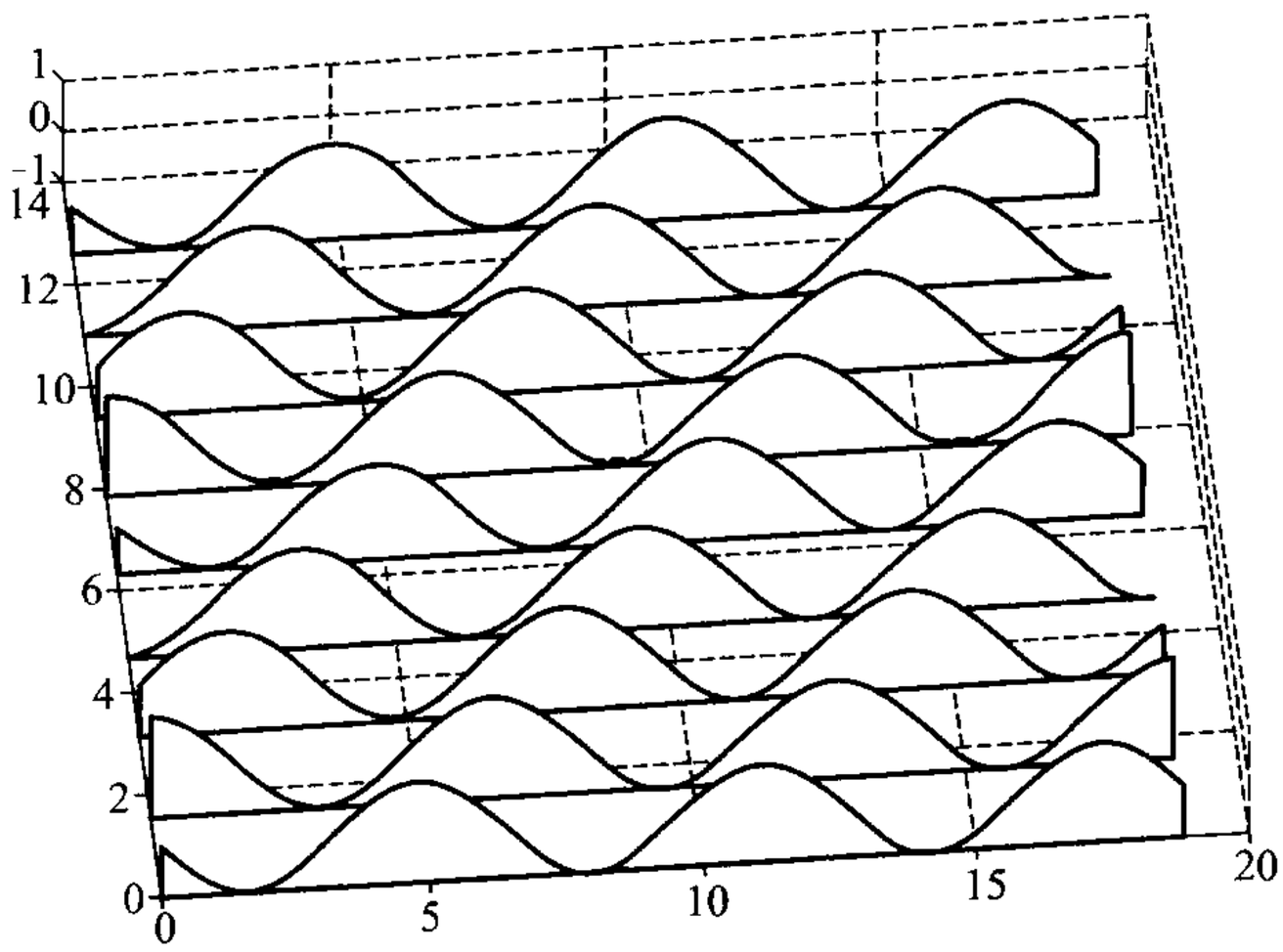


图 1.22　行波的图像

这里的waterfall指令用法与mesh相同，只是它不画出列元素之间的网线。图像表明，在横轴上的波形在纵轴的方向上是向右移动的。

§1.4.7.2　实时动画

实时动画的优点是保留没有改变的背景，只更新运动部分的图案，所以可以节约内存、提高运行速度。要想这样做，就不能只会对整幅图形进行处理，而必须将整幅图形分解成不同的组件，对每个组件给以标识（句柄），才能对图形中各个组件单独进行处理。正如可以用变量名调用变量一样，MATLAB是用图形句柄来控制图形对象。图像句柄是指定给图形对象的唯一的数字标识，它是在创建图形时建立的。下面画一条曲线，查看一下图形句柄所包含的图形属性，并通过改变其中的函数值及线型来改变图形。所画出两个图形如图1.23和图1.24所示。

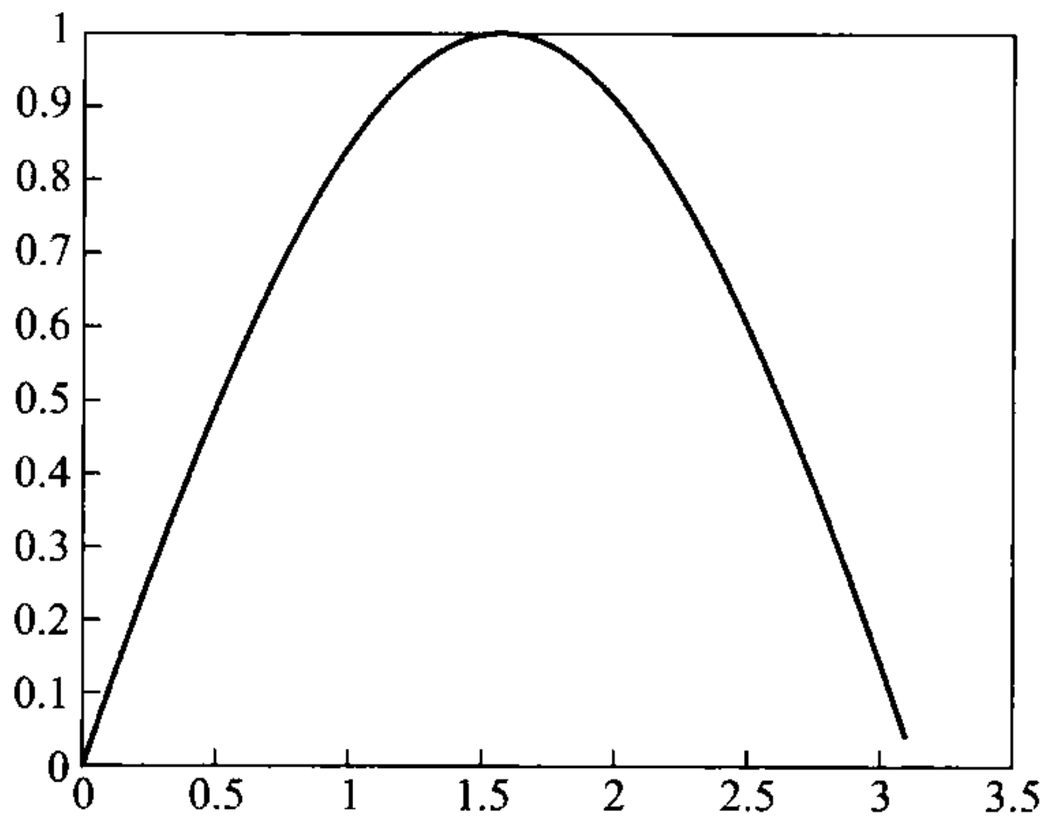


图 1.23　未修改的图形

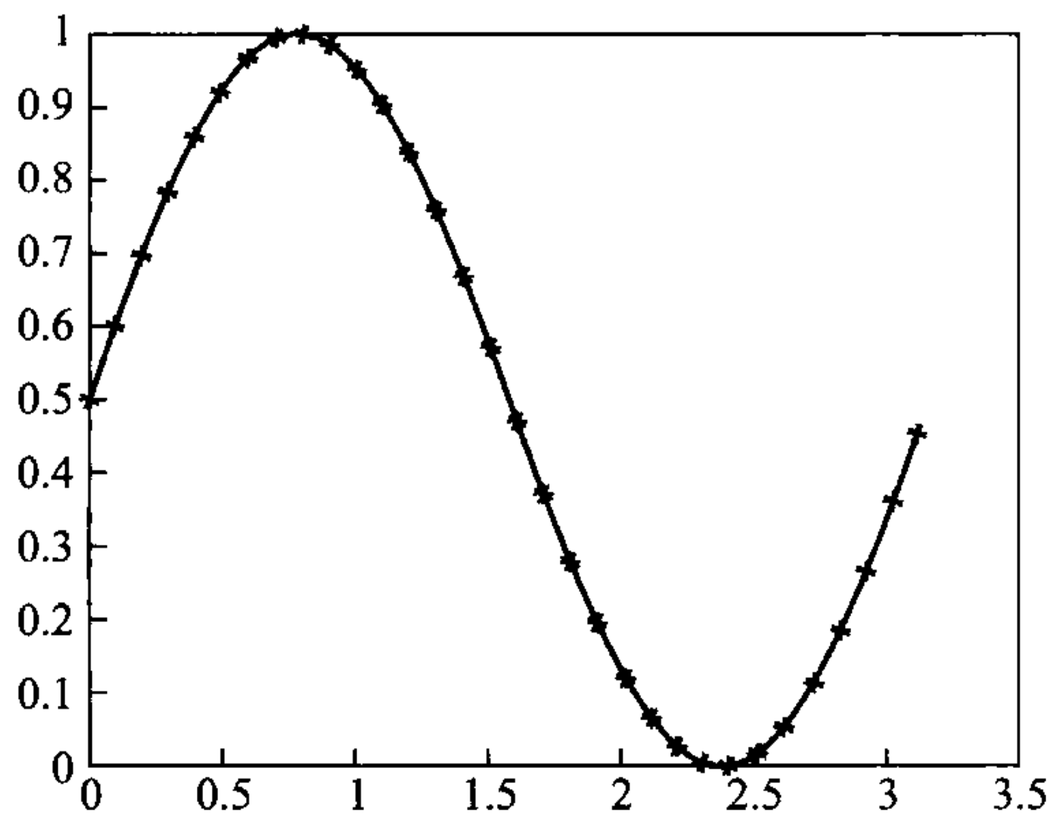


图 1.24 用图形句柄来修改的图形

```
>>x=0:0.1:pi;
>>y=sin(x);
>>h=plot(x,y)                                    %建立图形句柄
  h = 158                 %显示图形句柄的数字标识，其值由这次操作决定
>> get(h)                                        %通过图形句柄查看图形属性
            Color: [0 0 1]
        EraseMode: 'normal'
        LineStyle: '-'
        LineWidth: 0.5
           Marker: 'none'
       MarkerSize: 6
  MarkerEdgeColor: 'auto'
      MarkerFaceColor: 'none'
                XData: [1x32 double]
                YData: [1x32 double]
                ZData: [1x0 double]
          BeingDeleted: 'off'
          ButtonDownFcn: []
               Children: [0x1 double]
               Clipping: 'on'
             CreateFcn: []
             DeleteFcn: []
            BusyAction: 'queue'
```

```
    HandleVisibility: 'on'
             HitTest: 'on'
        Interruptible: 'on'
            Selected: 'off'
  SelectionHighlight: 'on'
                 Tag: ''
                Type: 'line'
       UIContextMenu: []
            UserData: []
             Visible: 'on'
              Parent: 157
         DisplayName: ''
           XDataMode: 'manual'
         XDataSource: ''
         YDataSource: ''
         ZDataSource: ''
>> set(h,'YData',sin(2*x),'Marker','*')   %改变图形的函数值及线型
>> drawnow                                 %画出新图
```

这里的get所显示的方式在MATLAB中称之为结构数组，也是一种特殊的数据形式。

关于句柄图形系统还需要了解以下内容：

1. 句柄图形的结构层次

句柄图形系统包含以下含义：一幅图的各个组成部分都是一个对象，每一个对象有一系列句柄来标识它，每一个对象都有可以设置和改变的属性。

图形对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其他对象的父对象。图形窗口是根对象的子对象，坐标轴的用户界面等对象是图形窗口的子对象。线条、文本、表面和图像等对象是坐标轴对象的子对象。这种层次关系如图1.25所示。一个父对象可以包含一个或多个子对象，如根屏幕可包含一个或多个图形窗口。

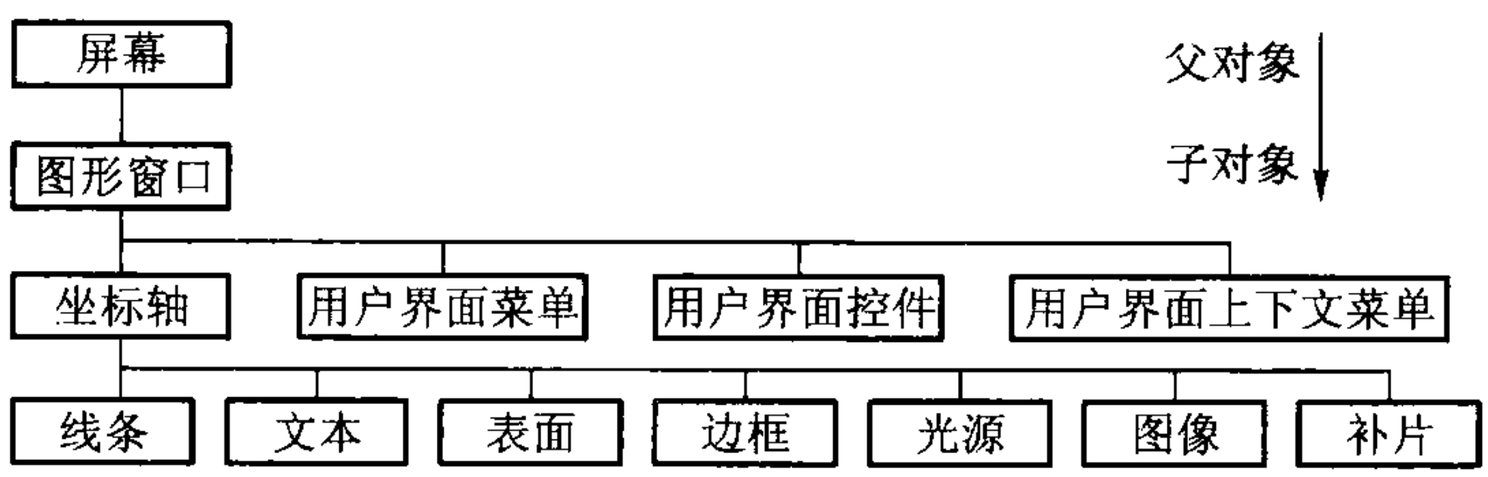


图 1.25　句柄图形系统的层次关系

2. 访问图形对象句柄

句柄实际上就是分配给每个对象的唯一的数字标识，它是在创建图形对象时自动建立的。访问图形对象句柄的指令有：

gcf	获取当前图形窗口的句柄
gca	获取当前坐标轴的句柄
gco	获取当前对象的句柄
gcbo	获取当前正在调用的对象的句柄
gcbf	获取包括当前正在调用的对象的图形的句柄
findobj	查找具有某一属性的图形对象的句柄
copyobj	复制图形对象
delete	删除图形对象

3. 设置图形对象属性

线条有粗细，字符的字体有大小，这些都叫做图形对象的属性。图形对象的属性包括属性名（字符串）和相应的数值，改变图形对象的某个属性的值就能改变图形对象。

(1) get　获取图形对象的属性值

语句格式为：

PropertyValue = get(h)

PropertyValue = get(H, 'PropertyName')

第一种格式的含义为获取图形句柄为h的图像属性，操作的结果是返回一个结构数组，结构数组的各个域名和赋值对应图形对象的各个属性和属性值。第二种格式的含义为获取图形句柄为H的图像属性名为PropertyName的属性值。

(2) set　设置图形对象的属性值

语句格式为：

set(H, '属性名', 属性值)

set(H, '属性名1', 属性值1, '属性名2', 属性值2, …)

第一种格式是对句柄为H的图形对象中指定属性名的属性值加以设置。第二种格式则可以同时设置多个属性值。

4. 用句柄图形做动画的方法

在做动画时要擦除旧对象，画出新对象，MATLAB用图形对象的 'EraseMode' 属性来在保持背景图案的条件下，完成这个任务。图形对象的'EraseMode'属性有四种：

normal	重绘整个显示区，它产生的图形准确但速度慢
none	不做任何擦除，直接在原图上绘图
xor	擦除旧对象的点并绘制新对象的点
background	把旧对象的颜色变为背景色
drawnow	刷新屏幕，更新图像

例16 行波的实时动画

```
%wave2.m
t=0:pi/20:4*pi;  x=0:0.1*pi:4*pi;  y=sin(x);
h=plot(x,y);                                  %将曲线的句柄命名为h
axis([0,12,-1,1]);                            %设定坐标轴的范围
set(h,'EraseMode','xor')                      %设定h的擦除属性为xor
for i=2:60
   y=sin(x-t(i));                             %计算新的y值（x值不变）
   set(h,'XData',x,'YData',y)                 %用新的x,y替代h中的旧值
   drawnow;                                   %重新画图
end
```

此动画演示很快，如果想放慢演示速度，可以在end之前加上pause(0.5)。

§1.5 画物理场的专用指令

物理场包括标量场与矢量场。对于二维的标量场，前面所介绍三维曲面的图形都可以应用，对于三维标量场，可以用颜色来表示各点场的大小，即前面介绍的四维数据的表示方法。此外对二维标量场，等值线也是常用的表示方法，而对于三维标量场，则可使用等值曲面来表示场的变化。对于矢量场，MATLAB也有很多有效的表现方法，详述如下：

§1.5.1 等值线表现二维标量场

二维标量场可用等值线来表现，比如指令contour画平面等值线，指令 contour3 画立体等值线，指令contourf画的等值线内填充了颜色。下面是两个例子，程序调用的函数peaks是MATLAB为演示程序所建立的一个函数，可以用edit peaks查看程序的内容。

画立体等值线和填色等值线的程序如下。图形分别为图1.26和图1.27。

```
>> z = peaks;
>> contour3(peaks)
```

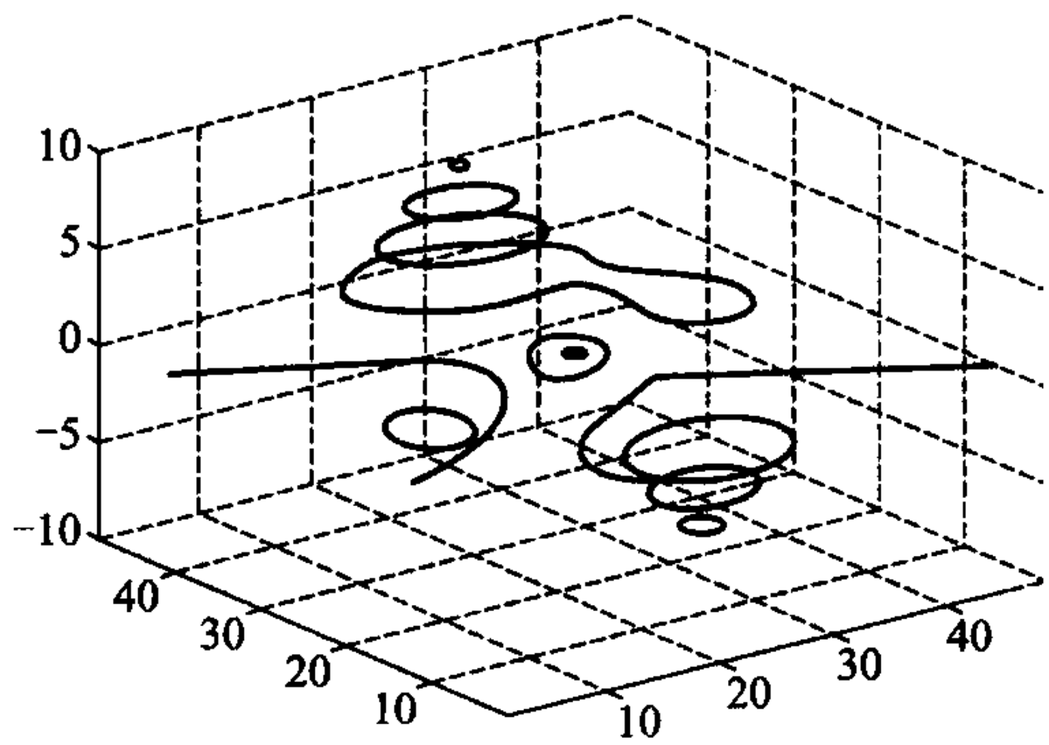


图 1.26 立体等值线图

```
>> z = peaks;
>> [c,h] = contourf(z); clabel(c,h), colorbar
```

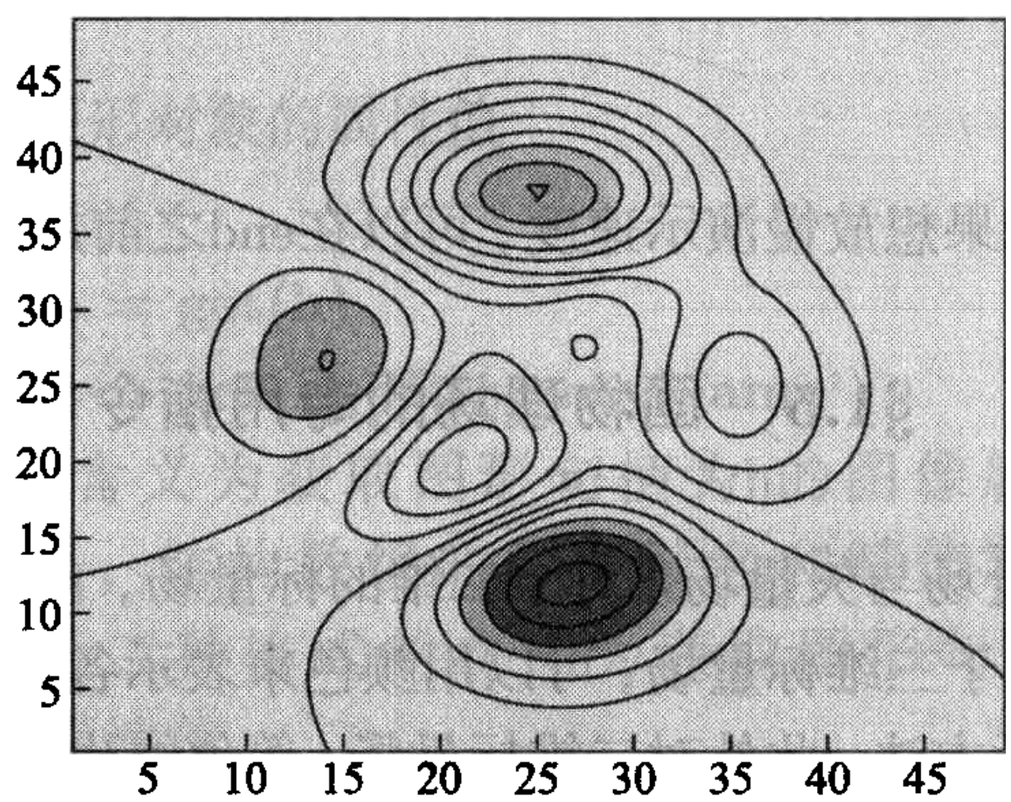


图 1.27 填色等值线图

§1.5.2 等值面表现三维标量场

三维标量场可以用等值面图来表现。图1.28画出的是一股水流喷到水箱中所形成的速度场的等势面。为了提高效果，还选择了一定的视角，加上了光照效果。为了能看到内部的等势面，可用指令alpha将外部的等势面画成半透明状。相关的操作指令如下：

```
[x y z v] = flow;                                  %获得三维速度场的数据
p = patch(isosurface(x, y, z, v, -3));             %为等值面准备光照
isonormals(x,y,z,v, p)                             %数据重新规一化
set(p, 'FaceColor', 'red', 'EdgeColor', 'none');
```

```
                                        %设置表面与边界的颜色
daspect([1 1 1])                        %坐标轴的比例
view(3)                                 %视角
axis tight;  grid on                    %加上格线
camlight; lighting phong                %加上光照
alpha(.5)                               %设置表面的透明度值
```

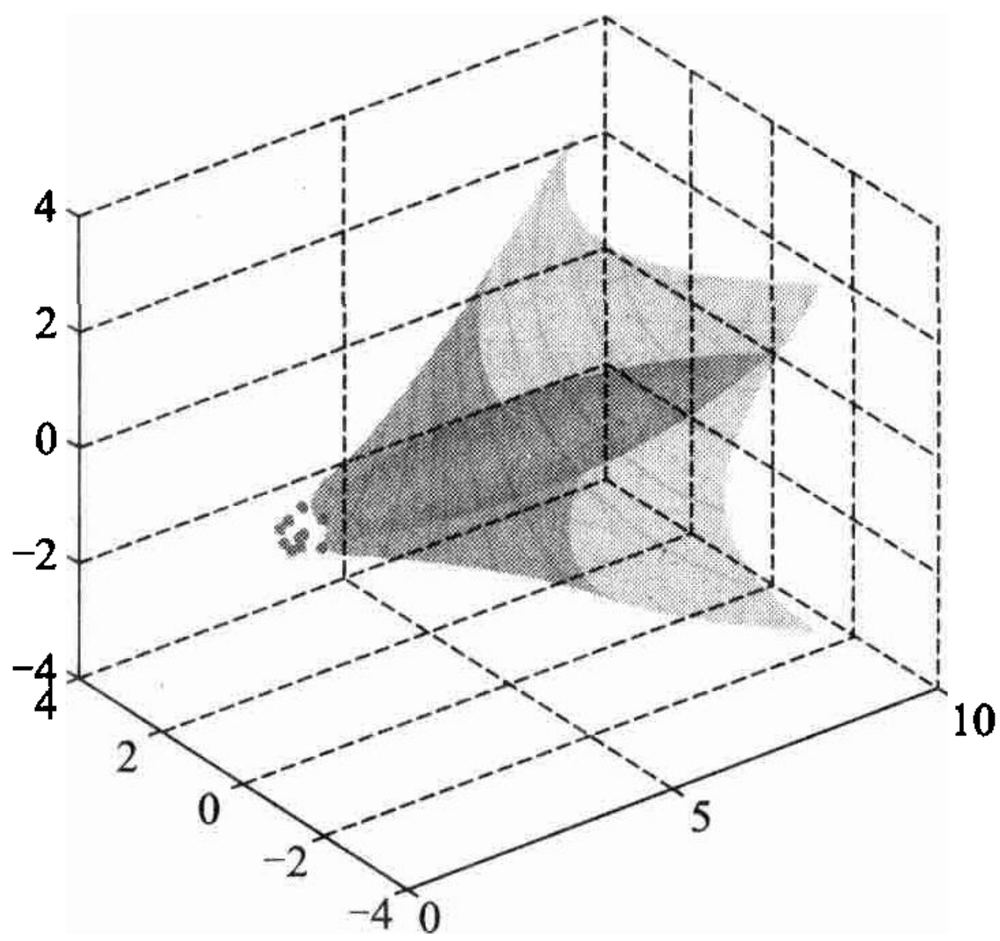


图 1.28 三维标量场的等值面图形

§1.5.3 用箭头、流线表现二维矢量场

1. 二维矢量场的箭头表示

指令 quiver 可以用箭头表示二维矢量场各点矢量场的大小与方向。在 quiver(x,y,u,v) 中x,y表示自变量，u,v表示矢量场的两个分量。如果需要，可以表示箭头的颜色，如下例所示。

例17 电量为$4\pi\epsilon_0$的正负电荷分别位于平面上的(0.2，0) 和(−0.2，0)处，形成一个电偶极子。下面的程序qtd.m画出了电偶极子的等势面（图1.29）。电偶极子的电势为Z，利用指令gradient 计算它在两个方向的梯度，指令contour是画等高线，在这里代表等势面，为了使等势线分布均匀，规定了画等势线的势能值。指令quiver是画出各点的电场的方向和大小。

```
%program  qtd.m
h=0.041; x=-0.3:h:0.3; y=-0.3:h:0.3;
[X,Y]=meshgrid(x,y);
Z=1./sqrt((X-0.2).^2.+Y.^2)-1./sqrt((X+0.2).^2+Y.^2);
[PX,PY]=gradient(-Z,h);
```

```
contour(x,y,Z,[-12,-8,-5,-3,-2,-1,-.5,-.1,.1,.5,1,2,3,5,8,12],
'b')
hold on
quiver(X,Y,PX,PY,'k')
```

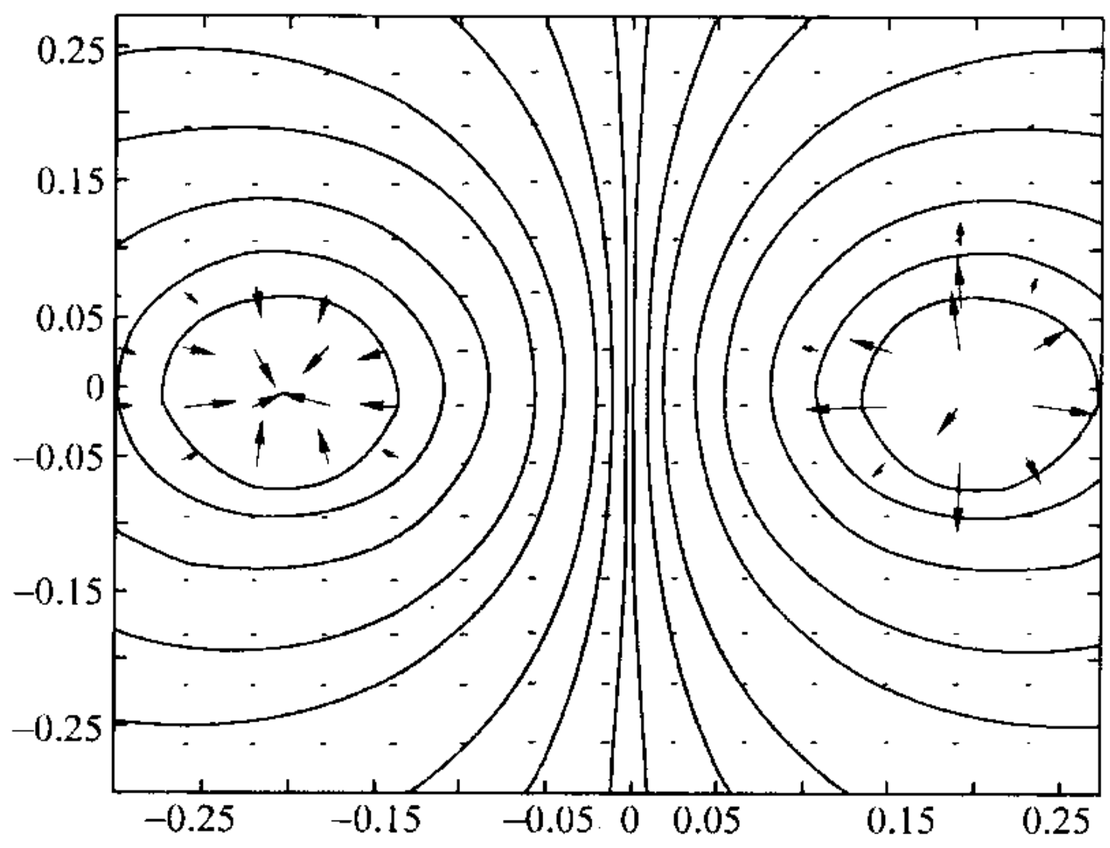


图 1.29 电偶极子的等势面

2. 二维矢量场的流线表示

Wind是大气中某处风速的数据，也是MATLAB内部的数据。我们可利用它来画二维矢量场的流线图和三维矢量场的空间流线图。所用的指令是streamline，其中三组参数的含义依次是空间坐标、矢量场的分量、流线起点。下面是平面流线图的程序，图像是图1.30。

```
load wind
zmax = max(z(:)); zmin = min(z(:));
streamslice(x,y,z,u,v,w,[],[],(zmax-zmin)/2)
```

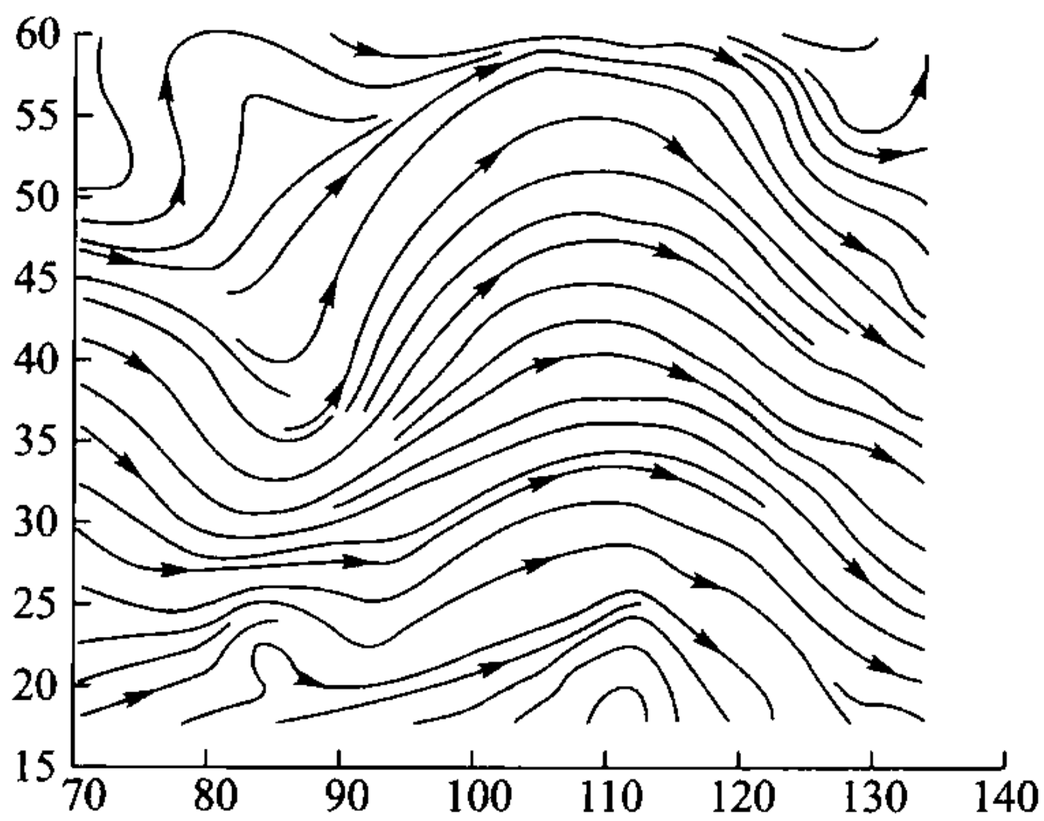


图 1.30 用流线表示的二维矢量场

§1.5.4 用箭头、流线、锥体、流管、流带表现三维矢量场

三维矢量场是比较难以可视化的内容，但是MATLAB提供了多种有效的方法。所用的方法有：

(1) 用箭头和锥体(Coneplot)表示各点的场的大小与方向；

(2) 用流线(Streamlines)表示矢量场，流线的切线表示该点场的方向，流线的密度反映场的大小；

(3) 用流管(Streamtubes)表示，流管的粗细反映矢量场的散度变化；

(4) 用流带(Sreamribbons)表示，流带的偏转反映矢量场的旋度变化。

查看Demos下面的Matlab/Valume Visualization的演示程序就能看到它们的应用。

1. 三维矢量场用箭头表示

可以用指令 quiver3 用箭头来表示三维矢量场的三个分量的大小与方向。quiver3 (x,y,z,,u,v,w,s) 中x,y,z 表示自变量，u,v,w表示矢量场在(x,y,z)点的三个分量，s表示调整箭头长短的参数，如果取零表示不调整箭头的长短。如下面的例子。

用三维曲线的三个切向分量画切线。所得图形为图1.31。

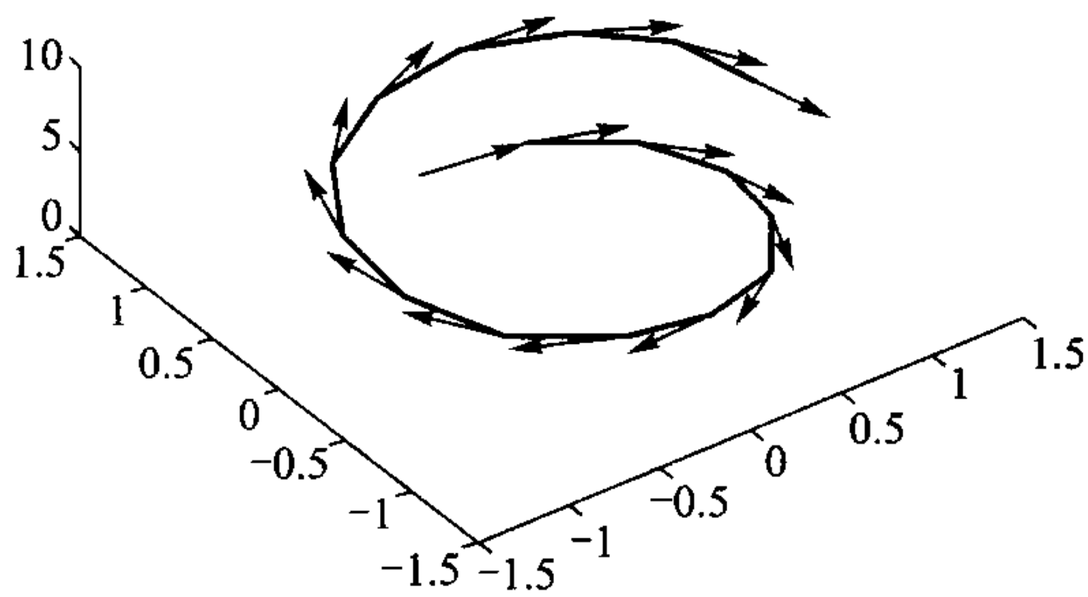


图 1.31 用箭头画三维曲线的切线

```
t = 0 : 0.5 : 8;
x = sin(t);
y =cos(t);
z = t;
plot3(x, y, z),
hold on
u=gradient(x);                                  %计算曲线三个方向梯度
v=gradient(y);
```

```
w=gradient(z);
quiver3(x,y,z,u,v,w,0)
view(-36,66)
```

用二维曲面的三个法向分量画法线。所得图形为图1.32。

```
[x,y] = meshgrid(-2:0.4:2,-1:0.3:1);
z = x .* exp(-x.^2 - y.^2);
[u,v,w] = surfnorm(x,y,z);                    %计算曲面的法向分量
quiver3(x,y,z,u,v,w);
hold on,
surf(x,y,z), hold off
```

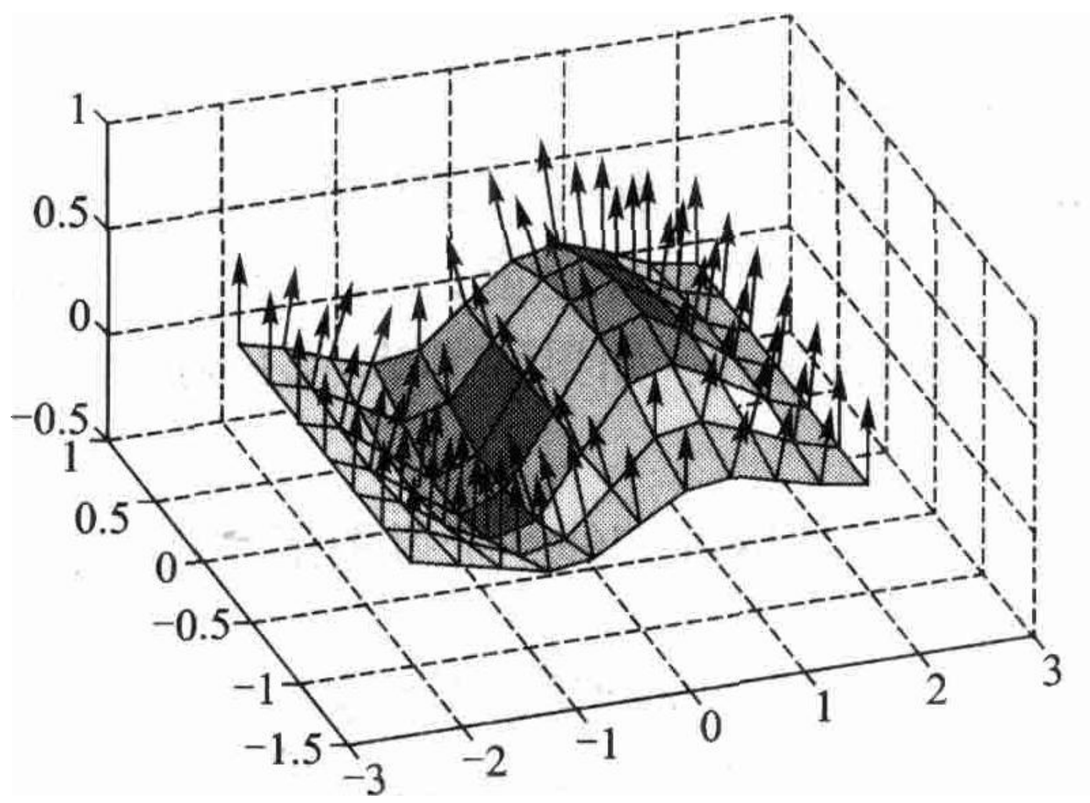


图 1.32 画二维曲面的法线

2. 三维矢量场用锥体表示

三维矢量场的表现手段相当丰富，但是要想取得好的表现效果，还要配合一定的观察方法。不难想象，观察三维图形时，随着观察点与视角的不同，会产生不同的观察效果。采用不同的光照，也会影响观察效果。MATLAB用相机的位置、运动及方位表示观察点与视角的变化，同时加上光照来提高观察效果。下面的程序就用到这种指令。所以先介绍一些有关相机的指令：

camdolly	移动相机和观察目标的相对位置
camlookat	用相机观察指定的对象
camorbit	使相机绕观察目标沿设定的轨道运动
campan	绕相机位置转动观察目标
campos	设置或获取相机位置
camproj	设置或获取投影方式（正交投影或透视投影）
camroll	绕观察轴转动相机

camtarget	设置或获取观察目标的位置
camup	设置或获取相机向上移动的数值
camva	设置或获取相机的观察角度
camzoom	放大或缩小相机所观察的场景
camlight	以相机为参照建立或移动光源

这些指令在图形窗口都有对应的图标和菜单。读者可任意画一个三维图形，在图形窗口用这些图标去操作图形以观察它们的作用。

以下程序中,画出了三维矢量场的锥体图、流线图、流管图及流带图，并将几种技术结合起来表示同一个矢量场。这些程序非常相似，我们只对其中一个程序作了注解。图1.33为所画锥体图形。

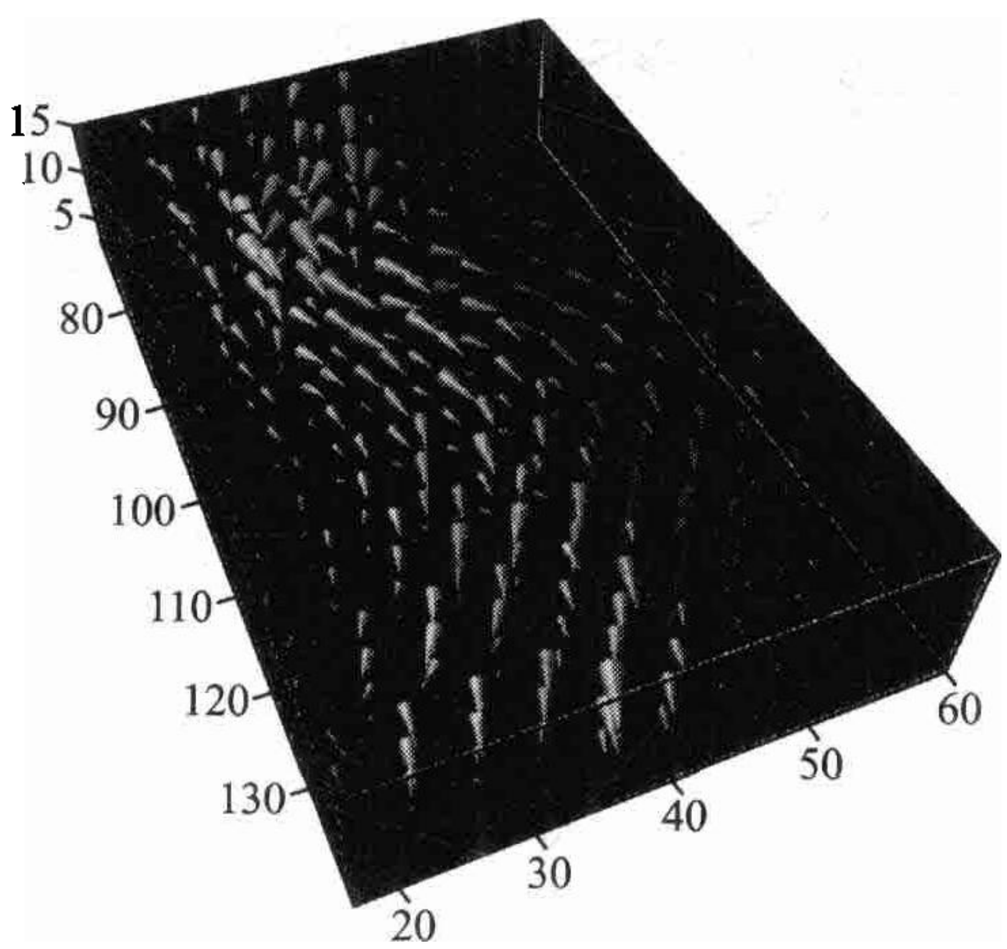


图 1.33 矢量场的锥体表示法

```
figure;
colordef(gcf,'black')                          %定义图形窗口背景颜色
cla                                  %删除当前坐标轴下所有可见的子对象的句柄
load wind                                                    %下载风速的数据
[cx, cy, cz] = meshgrid(linspace(71,134,10);
linspace(18,59,10),3:4:15);                                  %数据网格
daspect([1 1 1])                                        %数据在坐标轴上比例
h=coneplot(x,y,z,u,v,w,cx,cy,cz,y,3);     %画锥体图并建立图形句柄
set(h,'EdgeColor', 'none');
colormap(hsv);
box on;
axis tight
```

```
camproj perspective;                    %采用透视图投影法
camva(35);                              %照相机视角
campos([175 10 85]);                    %照相机位置
camtarget([105 40 0])                   %观察目标位置
camlight left;                          %照明灯光
lighting gouraud                        %体渲染的灯光效果
```

3. 三维矢量场用流线表示

程序如下。图1.34为所画流线图。

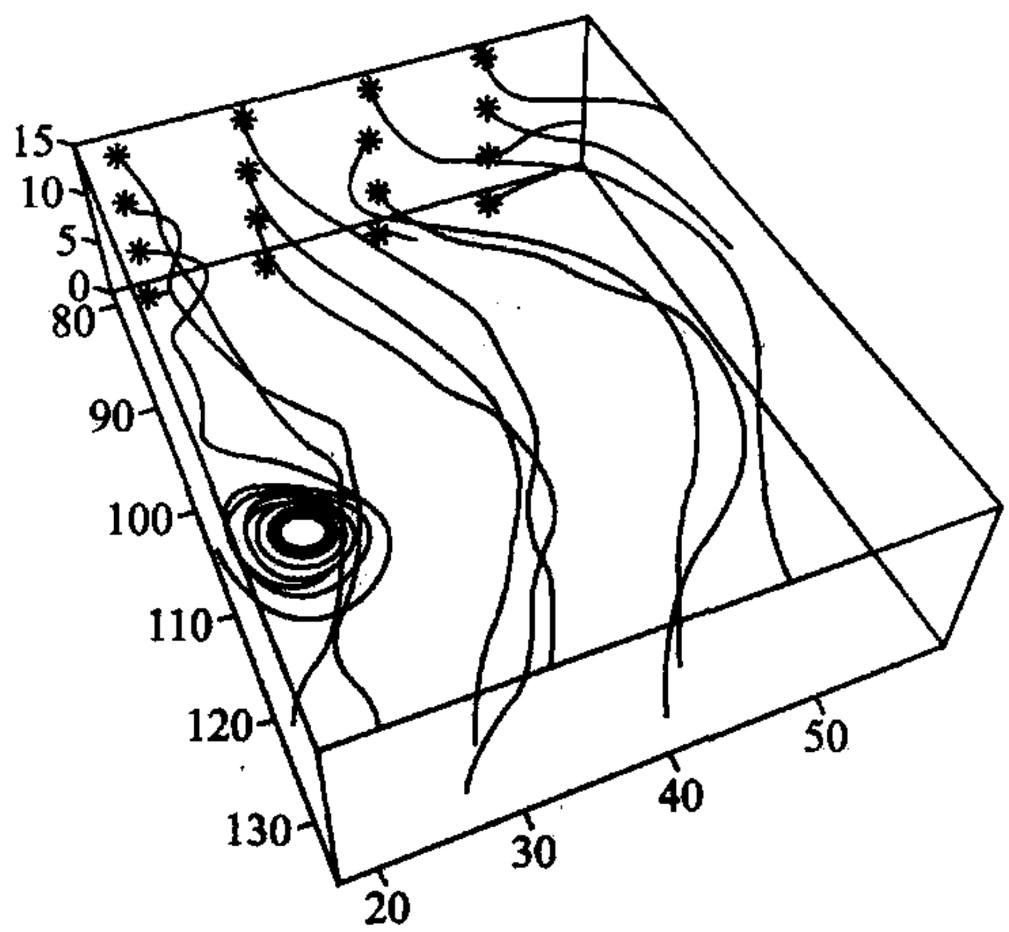


图 1.34 矢量场的流线表示法

```
%Streamlines of Wind Datafigure
figure
load wind
[sx sy sz] = meshgrid(80, 20:10:50, 0:5:15);
plot3(sx(:),sy(:),sz(:),'*r');                    %标出流线起点
h=streamline(x,y,z,u,v,w,sx,sy,sz);
set(h, 'Color', 'cyan');
daspect([1 1 1])
box on;
camproj perspective;
camva(32);
axis tight
campos([175 10 85]);
camtarget([105 40 0])
```

```
camlight left;
lighting gouraud
```

4. 三维矢量场用流管表示

程序如下。图1.35为所画流管图。

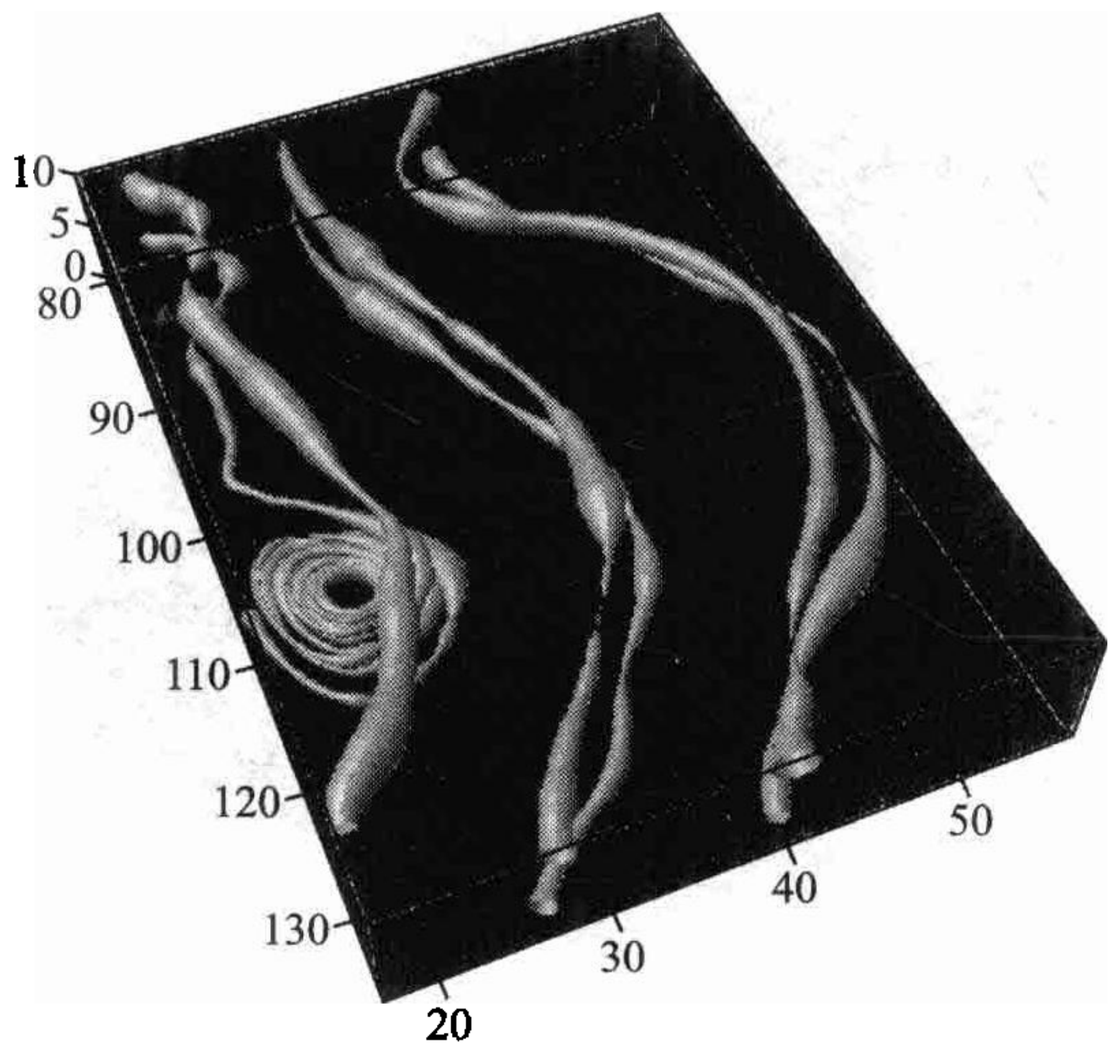


图 1.35 矢量场的流管表示法

```
% Streamtubes of Wind Data
%Useful for visualizing divergence of a vector fieldfigure;
figure
colordef(gcf,'black')
cla
load wind
[sx sy sz] = meshgrid(80, [20 30 40], [5 10]);
daspect([1,1,1]);
h=streamtube(x,y,z,u,v,w,sx,sy,sz);
set(h,'facecolor','cyan','edgecolor','none');
box on;
camproj perspective;
axis([70 138 17 60 2.5 16]);
axis tight
camva(28);
campos([175 10 95]);
camtarget([105 40 0])
```

```
camlight left;
lighting gouraud
```

5. 三维矢量场用流带表示

程序如下。图1.36为所画图形。

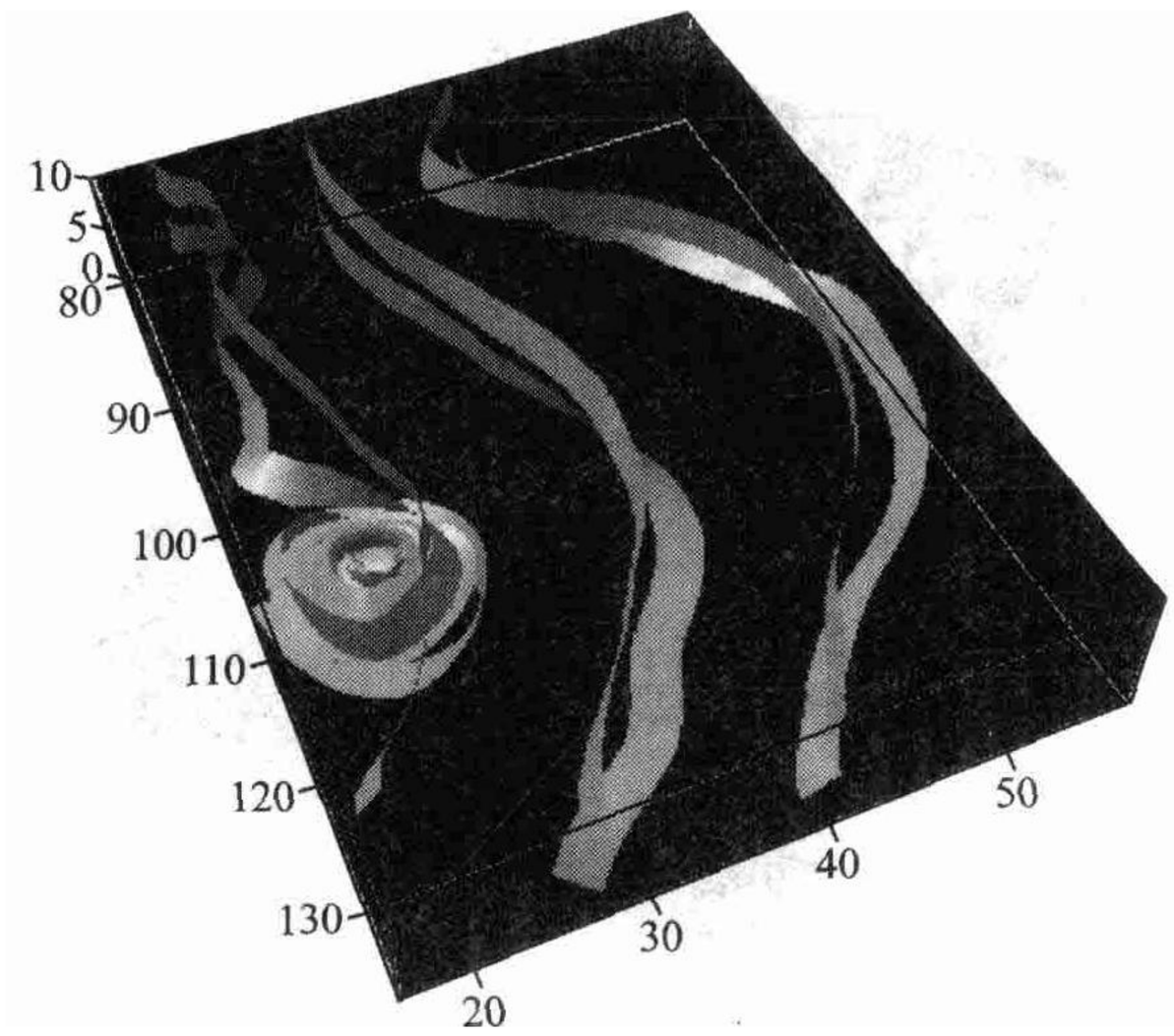


图 1.36 矢量场的流带表示法

```
%Streamribbons of Wind Data
%Useful for visualizing curl of a vector fieldfigure;
figure
colordef(gcf,'black')
cla
load wind
[sx sy sz] = meshgrid(80, [20 30 40], [5 10]);
daspect([1,1,1]);
h=streamribbon(x,y,z,u,v,w,sx,sy,sz);
set(h,'facecolor','cyan','edgecolor','none')
box on;
camproj perspective;
axis([70 138 17 60 2.5 16]);
axis tight
camva(28);
campos([175 10 85]);
camtarget([105 40 0])
```

```
camlight left;
lighting gouraud
```

6. 三维矢量场的综合表示

程序如下。图1.37为所画图形。

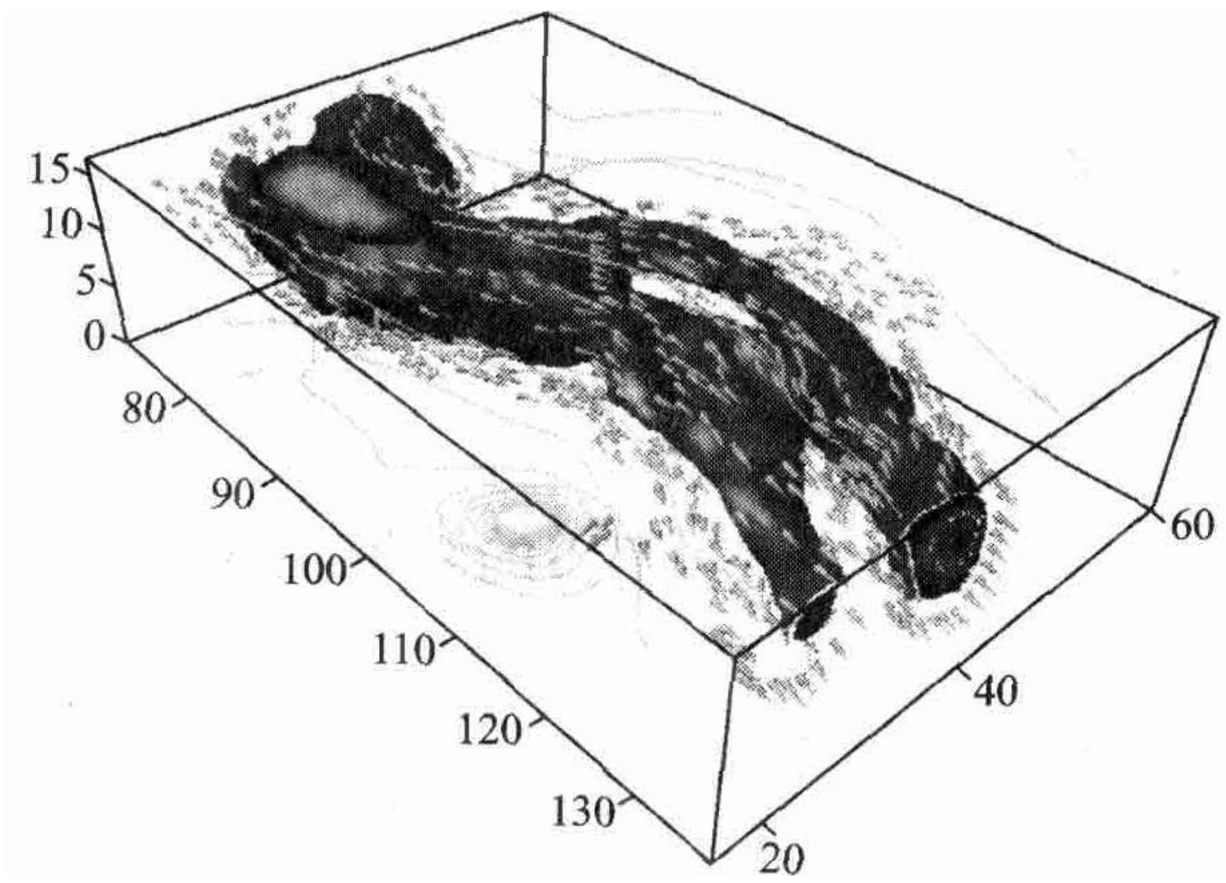


图 1.37 矢量场的综合表示法

```
%Isosurface, Isocaps, Coneplot, and Streamlines of Wind
Datafigure;
figure
colordef(gcf,'black')
cla
load wind
spd = sqrt(u.*u + v.*v + w.*w);
p = patch(isosurface(x,y,z,spd, 40));
isonormals(x,y,z,spd, p)
set(p, 'FaceColor', 'red', 'EdgeColor', 'none');
p2 = patch(isocaps(x,y,z,spd, 40));
set(p2, 'FaceColor', 'interp', 'EdgeColor', 'none')
daspect([1 1 1]);
[f verts] = reducepatch(isosurface(x,y,z,spd, 30), .2);
h=coneplot(x,y,z,u,v,w,verts(:,1),verts(:,2),verts(:,3),2);
set(h, 'FaceColor', 'cyan', 'EdgeColor', 'none');
[sx sy sz] = meshgrid(80, 20:10:50, 0:5:15);
h2=streamline(x,y,z,u,v,w,sx,sy,sz);
```

```
set(h2, 'Color', [.4 1 .4]);
colormap(jet)
box on
axis tight
camproj perspective;
camva(34);
campos([165 -20 65]);
camtarget([100 40 -5])
camlight left;
lighting gouraud
```

§1.6 图形窗口的编辑功能

图形窗口不仅能显示图形，还是一个功能全面的图形编辑器。在图形窗口中可用菜单或工具栏管理图形文件，编辑图形属性，加注文字，处理数据，旋转图形，改变观察图形的位置与视角(照相机功能)。当我们初步画好一个图形以后，可以利用这个工具更细致地观察图形，如改变光照效果且移动图形来进行观察，修改图形，添加文字甚至重画图形，并将修改的结果保存。

完整的图形窗口如图1.38所示，点击菜单View下面的全部子菜单，就能打开全部的图形窗口。窗口中间就是所画的图形，图形上面是菜单与工具栏，包括图形窗口工具栏(Figure Toolbar)，相机工具栏(Camera Toolbar)，作图工具栏(Plot Edit Toolbar)。图形左边是调色板(Figure Palette)，右边是曲线浏览器(Plot Browser)，下边是属性编辑器(Property Editor)，用以修改与处理图形。

图形窗口工具栏各个图标的功能依次是：

(1) 打开一个新的图形窗口。

(2) 打开保存的图形文件。

(3) 保存图形窗口现有的文件。

(4) 打印图形。

(5) 选择要编辑的图形属性。

(6) 放大图形。

(7) 缩小图形。

(8) 移动图形。

(9) 旋转图形。

(10) 显示光标处的曲线数据。

(11) 显示图形的颜色条。

(12) 插入曲线标识。

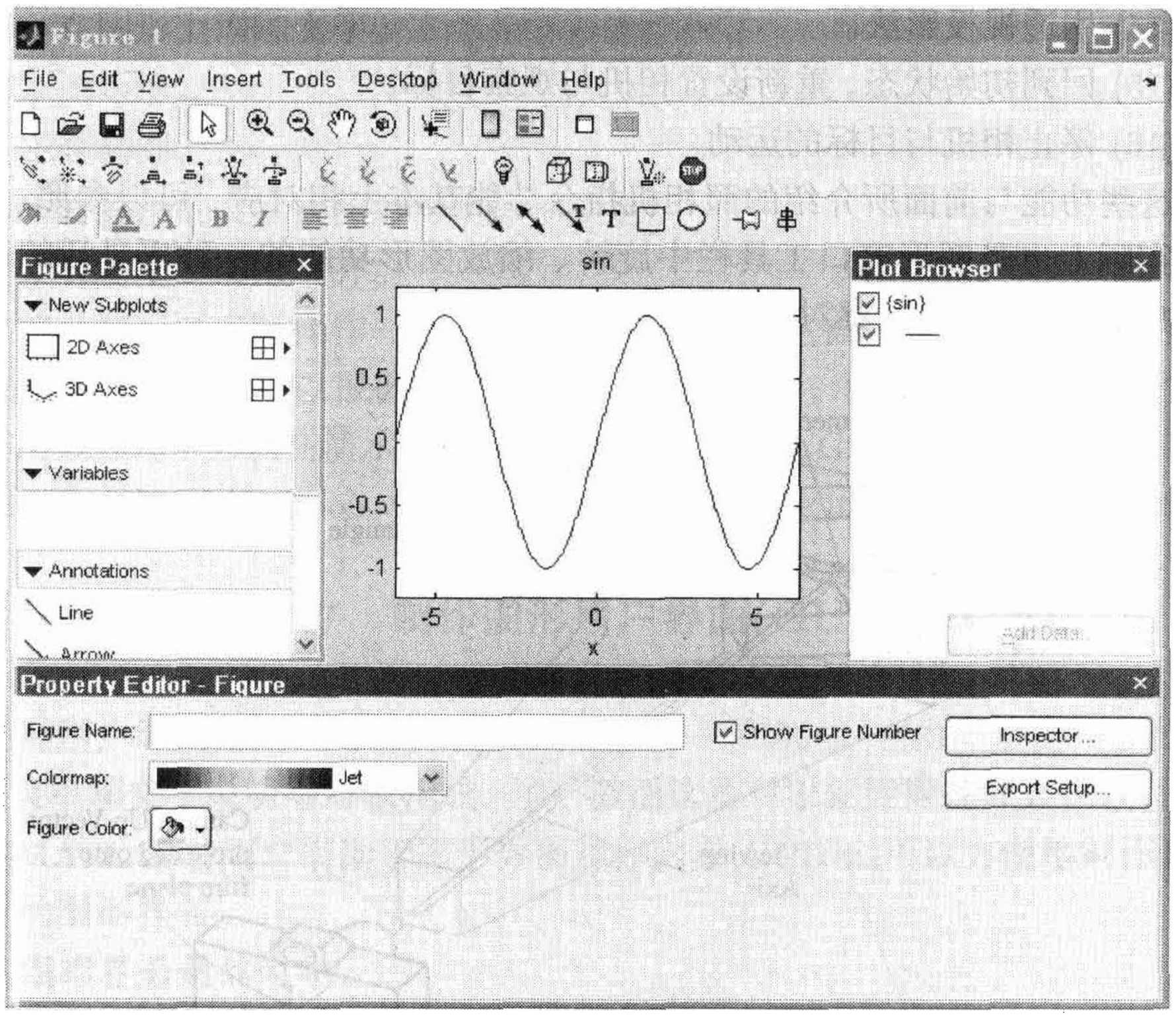


图 1.38 图形窗口的界面

(13) 不显示调色板，曲线浏览器和属性编辑器。

(14) 显示调色板，曲线浏览器和属性编辑器。

相机工具栏各个图标的功能依次是：

(1) 使相机绕观察目标沿设定的轨道运动。

(2) 以相机为参照建立或移动光源。

(3) 绕相机位置移动观察目标。

(4) 水平或垂直移动相机。

(5) 沿相机与观察目标连线移近或移远相机。

(6) 改变相机视角。

(7) 绕观察轴转动相机。

(8) 以x轴为主轴。

(9) 以y轴为主轴。

(10) 以z轴为主轴。

(11) 不确定主轴。

(12) 将灯光固定。

(13) 用正交投影法。

(14) 用透视投影法。

(15) 回到初始状态，重新设置相机与观察目标。

(16) 停止相机与目标的运动。

这些功能与前面所介绍的照相机指令功能基本上相对应，可以参照。这些功能实际上也是图形窗口工具栏中旋转、缩放图形功能的一种更灵活的运用。图1.39表现了相机与观察对象的坐标关系。

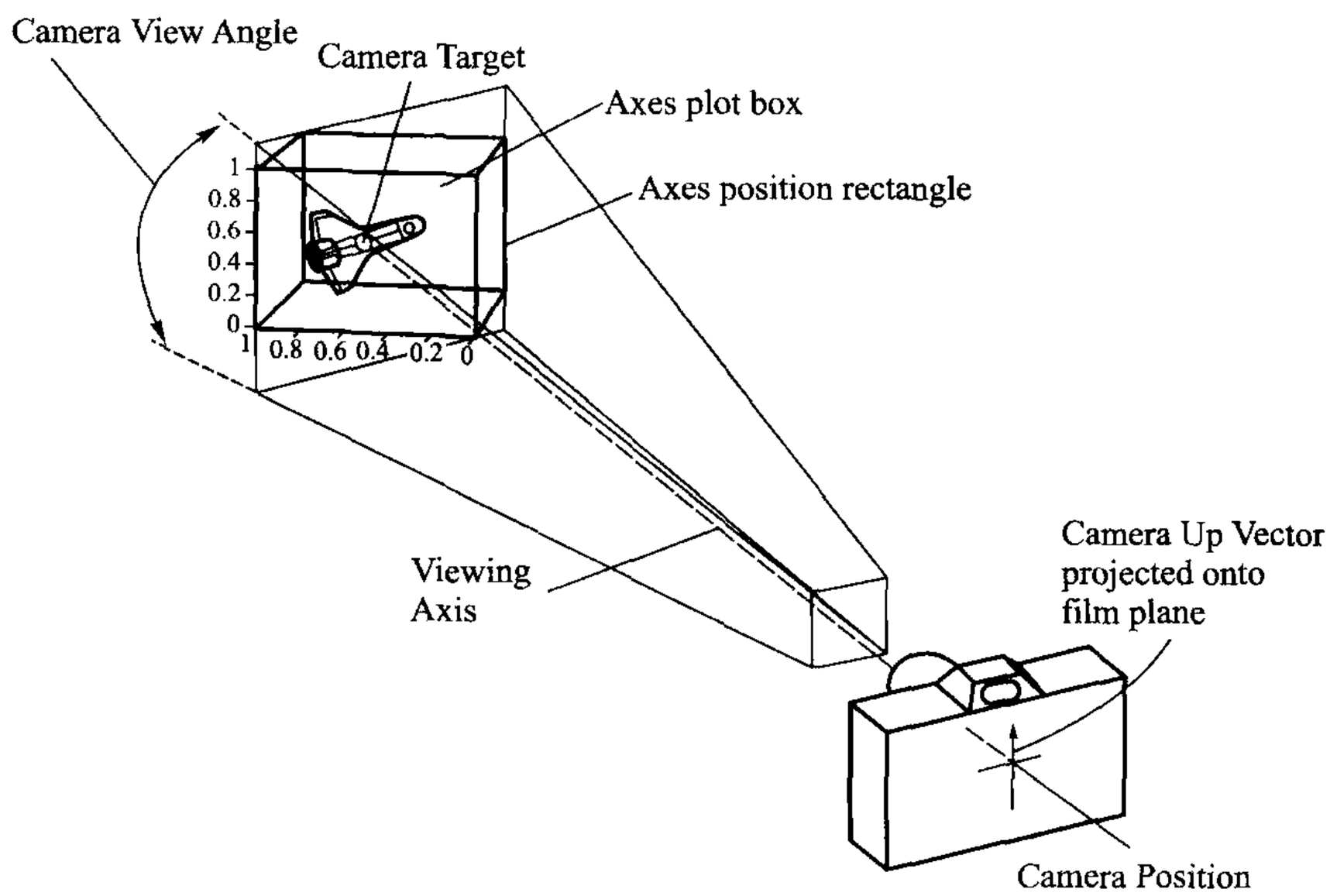


图 1.39 相机与观察对象的坐标关系

作图工具栏各个图标的功能依次是：

(1) 设置曲线表面颜色。

(2) 设置边线颜色。

(3) 设置文字色彩。

(4) 设置字体。

(5) 采用黑体字。

(6) 采用斜体字。

(7) 将文字左边对齐。

(8) 将文字居中对齐。

(9) 将文字右边对齐。

(10) 在图形上添加直线。

(11) 在图形上添加单箭头。

(12) 在图形上添加双箭头。

(13) 在图形上添加带箭头的文字。

(14) 在图形上添加文字。

(15) 画矩形。

(16) 画椭圆。

(17) 将添加的文字或图形与坐标轴锁定。

(18) 将多个子图形对齐。

这些功能只要实际操作一下，很容易掌握。而调色板、曲线浏览器和属性编辑器都是对话窗口，可以根据文字提示来操作。此外，在演示程序中有两个动画程序可以参考，分别在演示程序中的菜单MATLAB/Graphics/Interactive Plot Creation with the Plot Tools 和MATLAB/New Feature in Version 7/Graphics。

§1.7 制作图形用户界面(GUI)

图形用户界面是用户与计算机交流信息的一种方法，它由图形窗口、光标、按键、菜单和文字说明等对象构成。用户激活这些对象就能操控计算机。

在MATLAB中用户借助于图形界面制作向导(GUIDE)可以方便地制作自己所需要的图形界面。

1. 图形界面制作向导

键入guide，打开GUI窗口如图1.40所示。

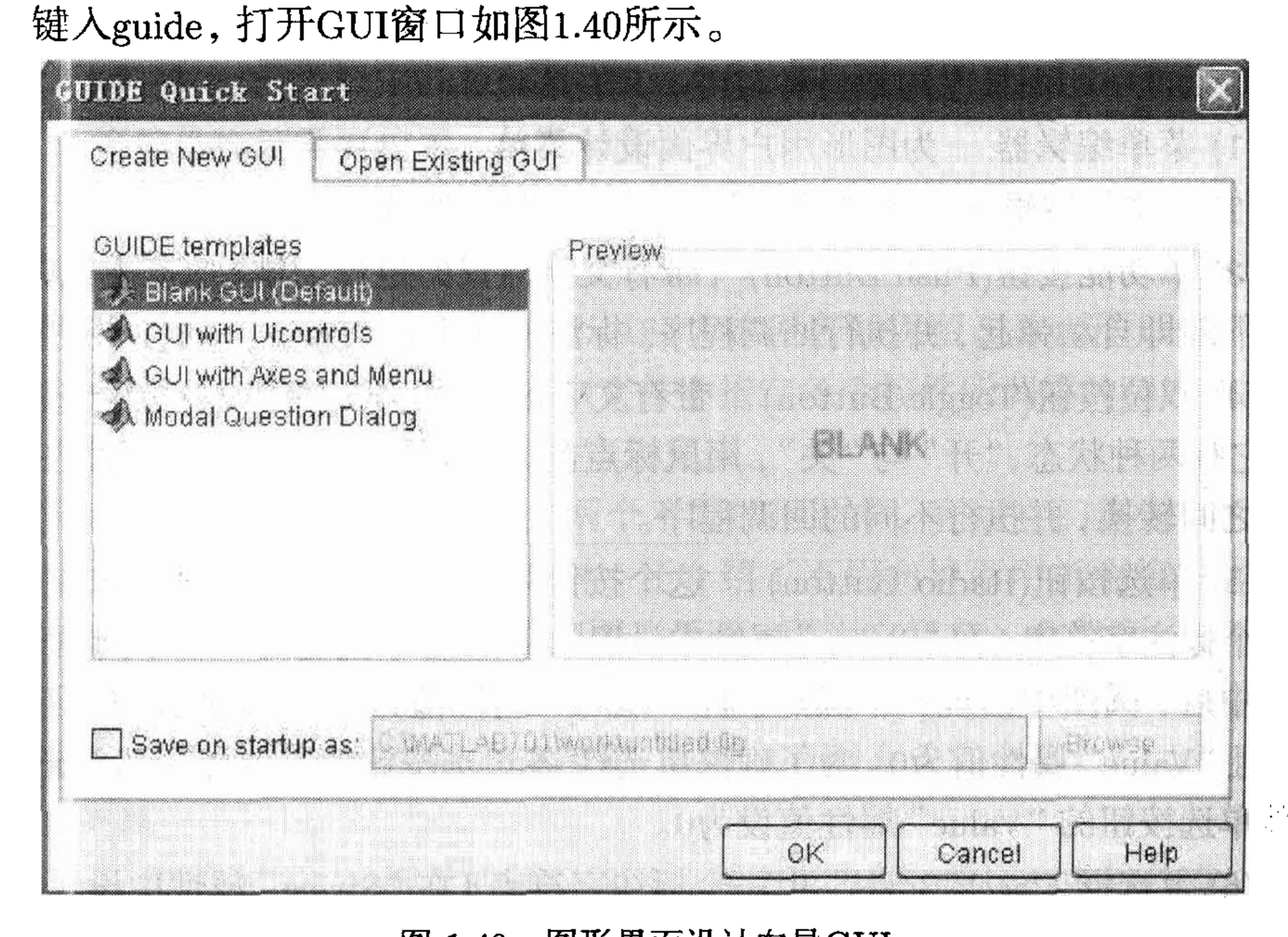


图 1.40 图形界面设计向导GUI

对话框中两项选择为：

Open Existing GUI　　打开已有的图形用户界面程序

Create New GUI　　建立新的图形用户界面程序

任意选择一个将打开新的对话框如图1.41所示。

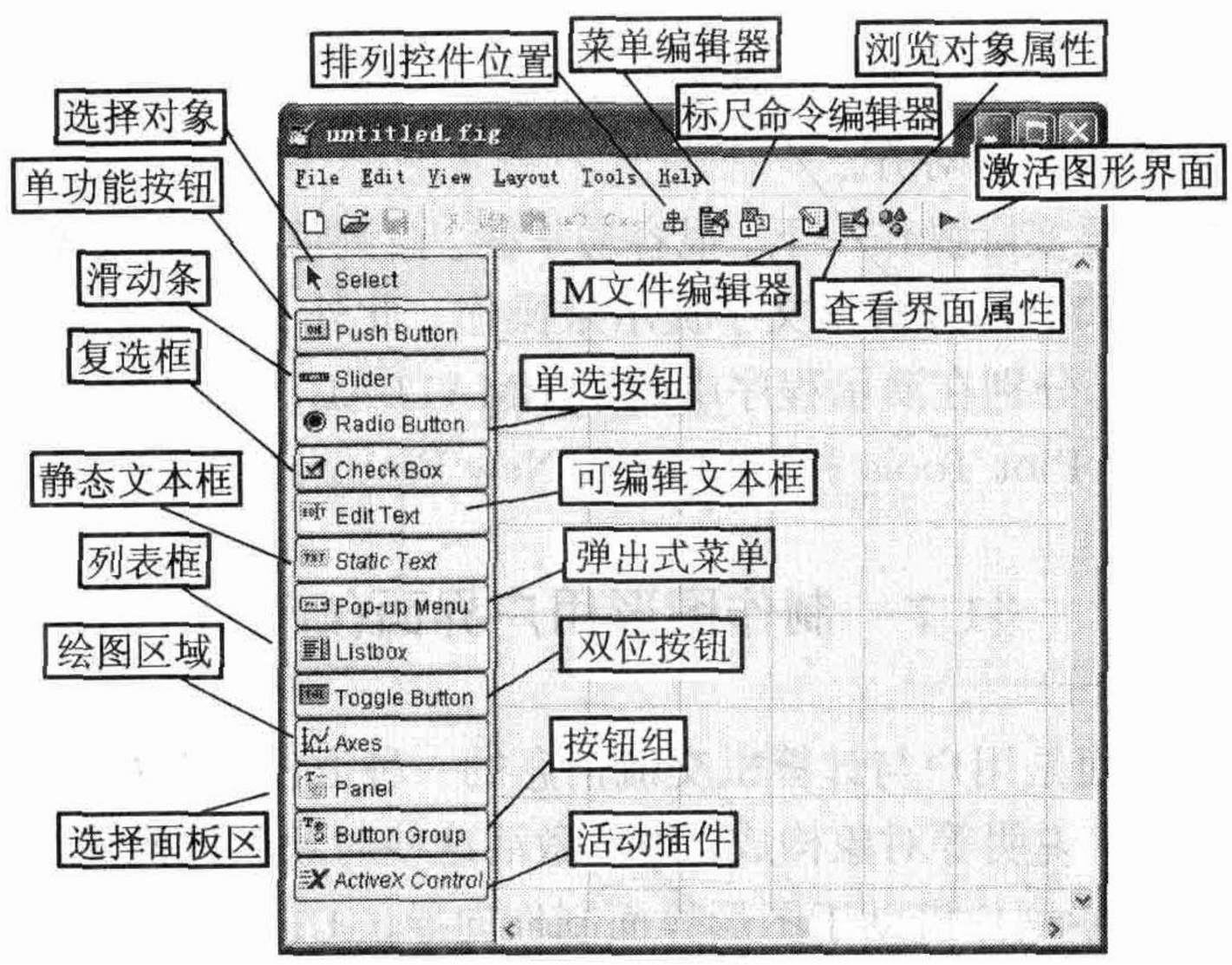


图 1.41　GUI的工作界面

2. 控件及工具栏功能介绍

对话框中的面板左边是各种控件，上边是菜单和工具栏。

(1) 菜单编辑器　为图形用户界面设计菜单。

(2) M文件编辑器　编辑图形用户界面所对应的M文件。

(3) 单功能按钮(Push Button)　带有文字标识的矩形，用鼠标点击，按钮会被按下随即自动弹起，并执行回调程序。标注的文本在“String”属性中设置。

(4) 双位按钮(Toggle Button)　带有文字标识的矩形，形状与单功能按钮相同。它有两种状态，“开”与“关”，用鼠标点击，按钮会被按下或者弹起，在两种状态之间转换，并执行不同的回调程序。

(5) 单选按钮(Radio Button)　这个按钮用于一组互斥的选项中选择一项。由一个标注字符串（在“String”属性中设置），和字符串左边的一个小圆圈组成。被选中后，圆圈中出现一个黑点，且“Value”属性值为1；若未被选中，圆圈为空，且“Value”属性值为0。为了确保单选性,各单选按钮的回调程序都要同时将其他单选按钮的“Value”属性值设为0。

(6) 复选框(Check Box)　由一个标注字符串（在“String”属性中设置），和字符串左边的一个方框组成。被选中后，方框中出现一个勾，且“Value”属性值

为1；若未被选中，方框为空，且“Value”属性值为0。复选框可用于多项互容的选项。

(7) 可编辑文本框(Edit Text)　可以动态地输入或显示文本字符串或数字。显示的字符在“String”属性中设置。

(8) 静态文本框(Static Text)　显示文本字符串的内容。该内容在“String”属性中设置，它不能用作字符或数据的输入。

(9) 滑动条(Slider)　有三个部分，滑动槽表示取值范围，槽内的滑块代表滑动条的当前值，滑动槽两端的箭头用于改变滑动条的值。滑动条的取值范围由“Max”和“Min”设定。可以用鼠标拖动滑块或者点击两边的箭头改变滑动条的值。

(10) 列表框(Listbox)　若干选项的列表，可以同时选择互容的多项，选项由“Min”和“Max”来设定，“Value”值为被选定的选项的序号，用鼠标选取一个选项，就会执行对应的回调程序。

(11) 弹出式菜单框(Pop-up Menu)　是一系列互斥的菜单，点击框边的箭头，会弹出一个菜单，用鼠标选取其中的一项，弹出的菜单会自动关闭，并将当前选定的项显示在框中，“Value”值就是该选项的序号。在“String”中设置各选项的字符串，不同选项的字符串用“|”隔开。

(12) 绘图区域(Axes)　程序所画的图形显示在这个区域。

控件的详细用法可以在help帮助界面下搜索得到。

3. 操作步骤

具体的操作步骤如下：

(1) 将控件拖到需要位置，布置对齐。

(2) 设定控件的有关参数。

(3) 点击工具栏中三角形按钮，激活程序。

此时将自动生成两个文件，一个以fig为扩展名的保留了界面图形的图形文件，另一个是对应的M文件。必须在M文件中指定各个控件所调用的程序或指令，界面才能工作。至此整个GUI的设计宣告完成。

下面将所有的控件分别运用到两个不同的例题中，利用它们作为模板，很容易编写自己要用的图形界面。此外，还介绍了不用向导而用指令编写这两个界面的程序。有的用户喜欢用指令编图形界面，因为它只有一个程序，在不同电脑中移植或改变程序所在目录，会有很多方便。限于篇幅，这里不再详述，对比两个程序，很容易理解各个指令的含义，也可以查阅有关的书籍。

4. 例题

例18　制作如图1.42所示的画双曲函数的图形界面。要用到的控件有，菜单编辑器、单选按钮、复选框、双位按钮。

图1.42的界面是用单选按钮选择画正弦双曲或余弦双曲，这是互斥的功能，所以只能画其中的一种曲线；用复选框决定是否加上标题；用双位按键来决定画网格或不画网格；用菜单来查看程序或退出界面。

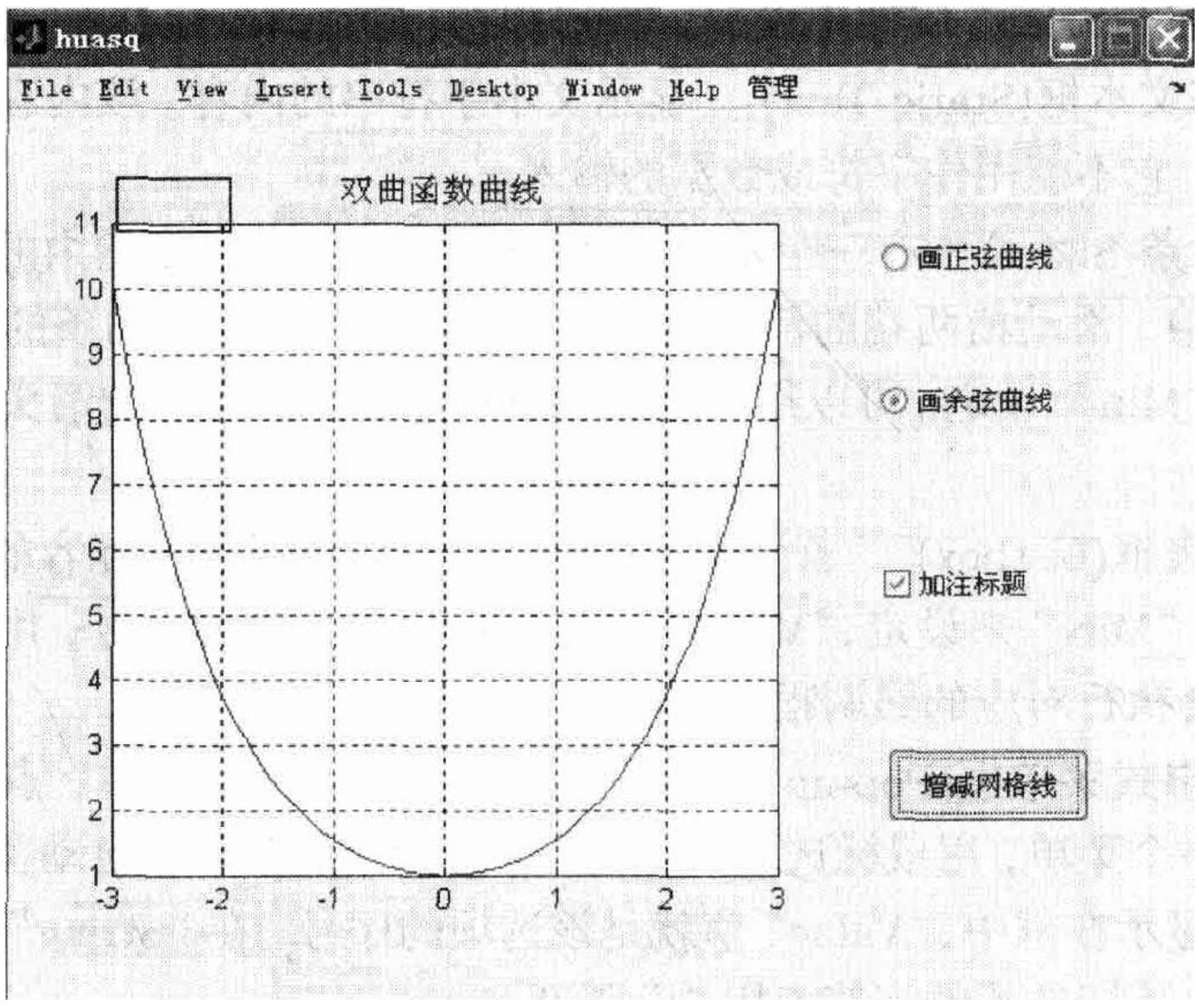


图 1.42 设计完成的图形界面

步骤一 建立按钮。

用向导面板按要求布置控件，如图 1.43 所示。方法是用鼠标点击要用的控件，然后将鼠标移动到右边，按下鼠标左键画出放置控件的位置，松开左键即可。

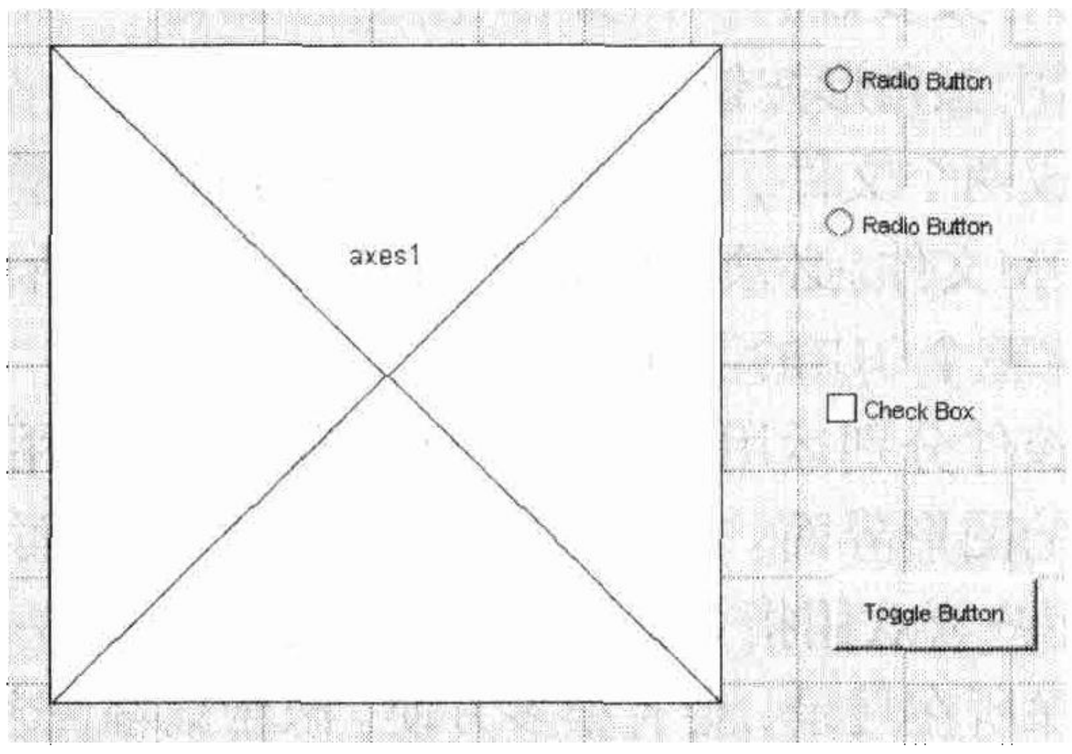


图 1.43 布置好的控件图

然后双击布置好的控件，打开Property Inspector对话框，设置各个控件的属性。如String是控件名称，FontSize 是字体大小。得到图1.44。

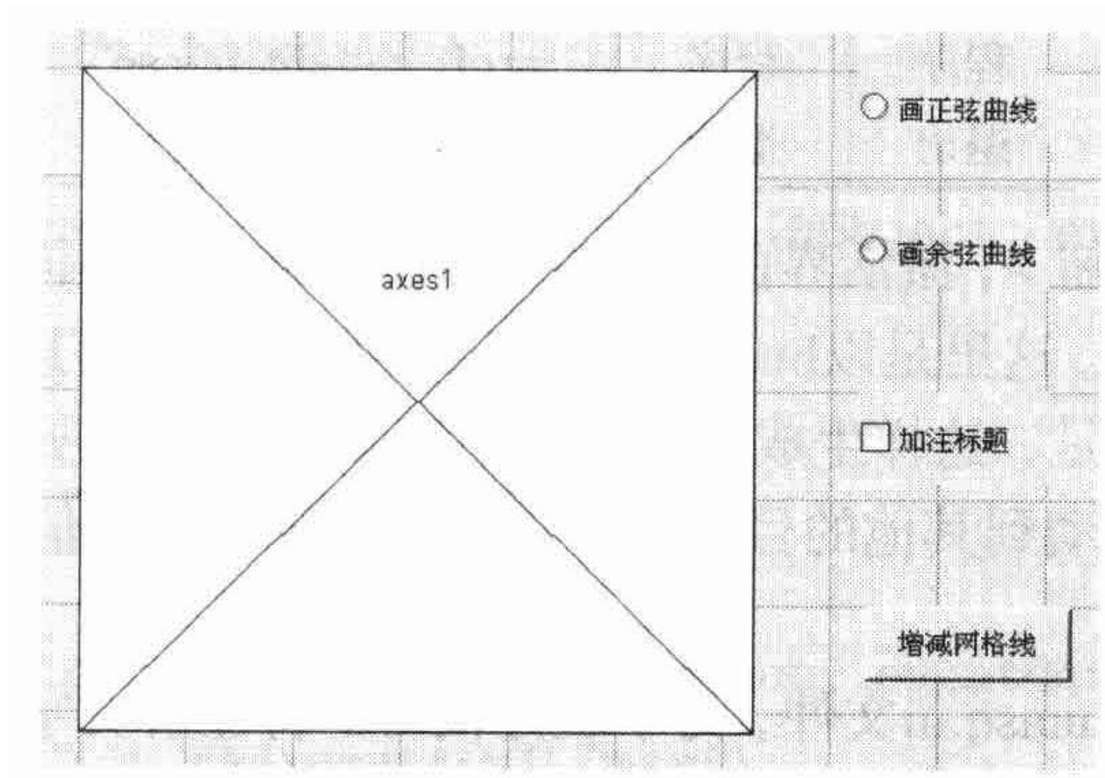


图 1.44 修改过控件属性的图形

步骤二 保留原菜单并增加新菜单。

点击“查看界面属性(Property Inspector)”按钮，将其中Menubar的属性由none 改为 figure，即保留了原来菜单。再建立新菜单，点击Tools/Menu Editor，打开菜单编辑器(图1.45)，用图标按钮建立一级菜单、二级菜单。如图填写一级菜单（管理），二级菜单（程序、退出），对每个菜单根据其功用填写有关的项目，如下表所示

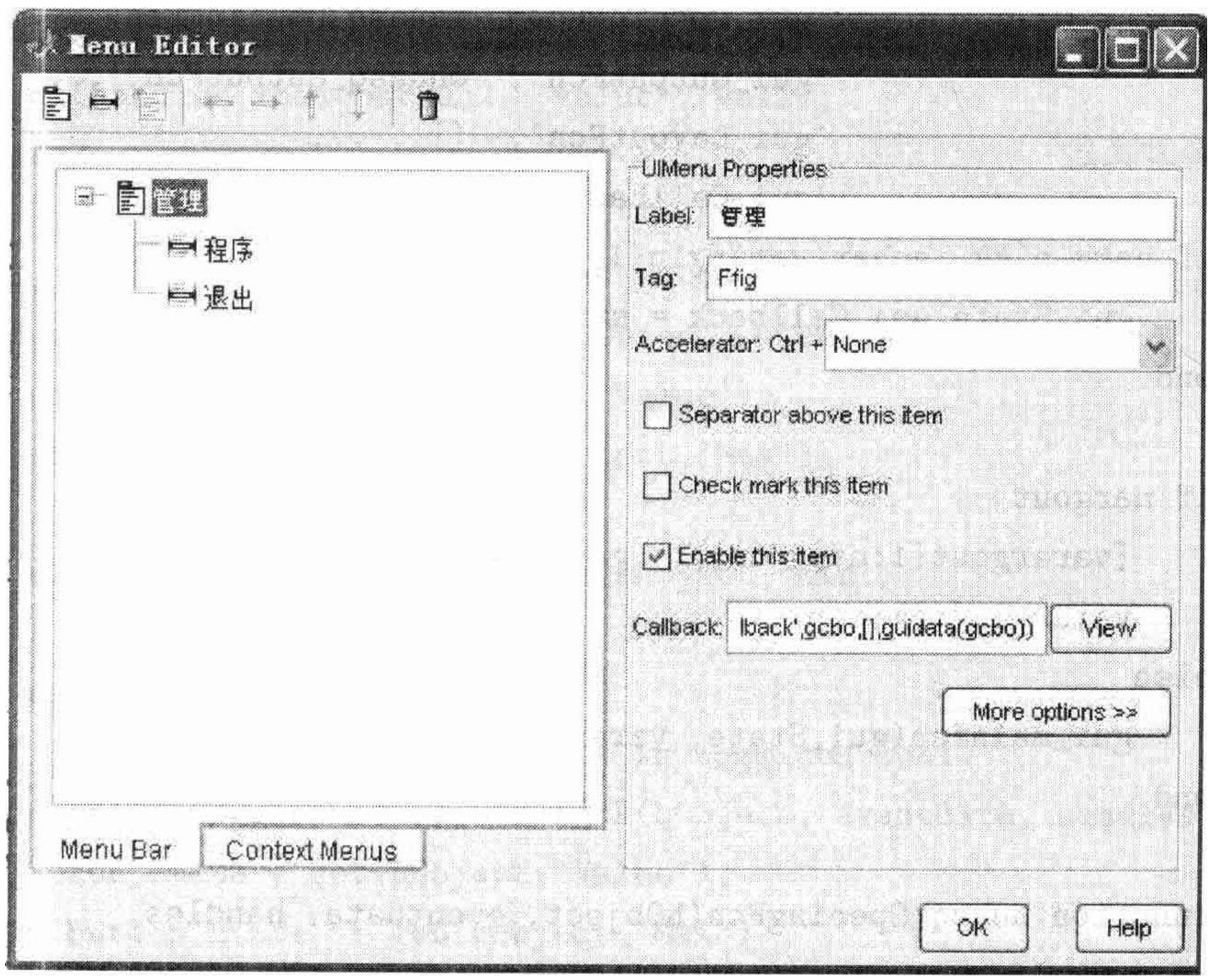


图 1.45 菜单编辑器的界面

菜单名	Label	Tag	Callback
程序	程序	prof	edit huasq
退出	退出	ex	close gcf

步骤三 点按钮 ▷，激活界面，自动生成两个同名但不同格式的文件。注意要自己填写文件名。这里是以huasq. fig 保存界面的图形信息，用huasq. m保存程序。需要注意的是，这样生成的M文件中还保存了文件所在目录的信息。如果将这两个文件移动到其他的目录下，必须手工改变这些信息才能使文件正常运行。

步骤四 编辑huasq.m文件，为了程序简练，将注解文字删去，设置各个按钮的回调函数如下面程序所示。使用时在指令窗口输入huasq，激活的图形界面如图1.42所示。

```
%界面一的M文件huasq.m
function varargout = huasq(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename,...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @huasq_OpeningFcn, ...
                   'gui_OutputFcn',  @huasq_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
    varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function huasq_OpeningFcn(hObject, eventdata, handles,
varargin)
```

```
handles.output = hObject;
guidata(hObject, handles);

function varargout = huasq_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;

function radiobutton1_Callback(hObject, eventdata, handles)
val1=get(handles.radiobutton1,'max');
set(handles.radiobutton1, 'value', val1);
val2=get(handles.radiobutton2,'min');
set(handles.radiobutton2, 'value', val2);
hsq1                                                    %画双曲函数
% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
val1=get(handles.radiobutton1,'min');
set(handles.radiobutton1, 'value', val1);
val2=get(handles.radiobutton2,'max');
set(handles.radiobutton2, 'value', val2);
hsq2                                                    %画双曲函数

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
 checkbox1_state=get(hObject,'Value');
 if  checkbox1_state==get(hObject,'Max');
     title('%双曲函数曲线')
 elseif  checkbox1_state==get(hObject,'Min');
     title('')
 end
% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
    grid on
```

```
elseif button_state == get(hObject,'Min')
    grid off
end

% ---------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)


%-----------------------------------------------------------------
function  hsq1                                          %画正弦双曲函数
t=-3:0.1:3;
y1=0.5*(exp(t)-exp(-t));
plot(t,y1)
%-----------------------------------------------------------------
function  hsq2                                          %画余弦双曲函数
t=-3:0.1:3;
y2=0.5*(exp(t)+exp(-t));
plot(t,y2)
```

例19 制作如下图1.46所示的画两个不同函数图形的界面。练习弹出式菜单、静态文本框、列表框、滑动条、可编辑文本框的用法。

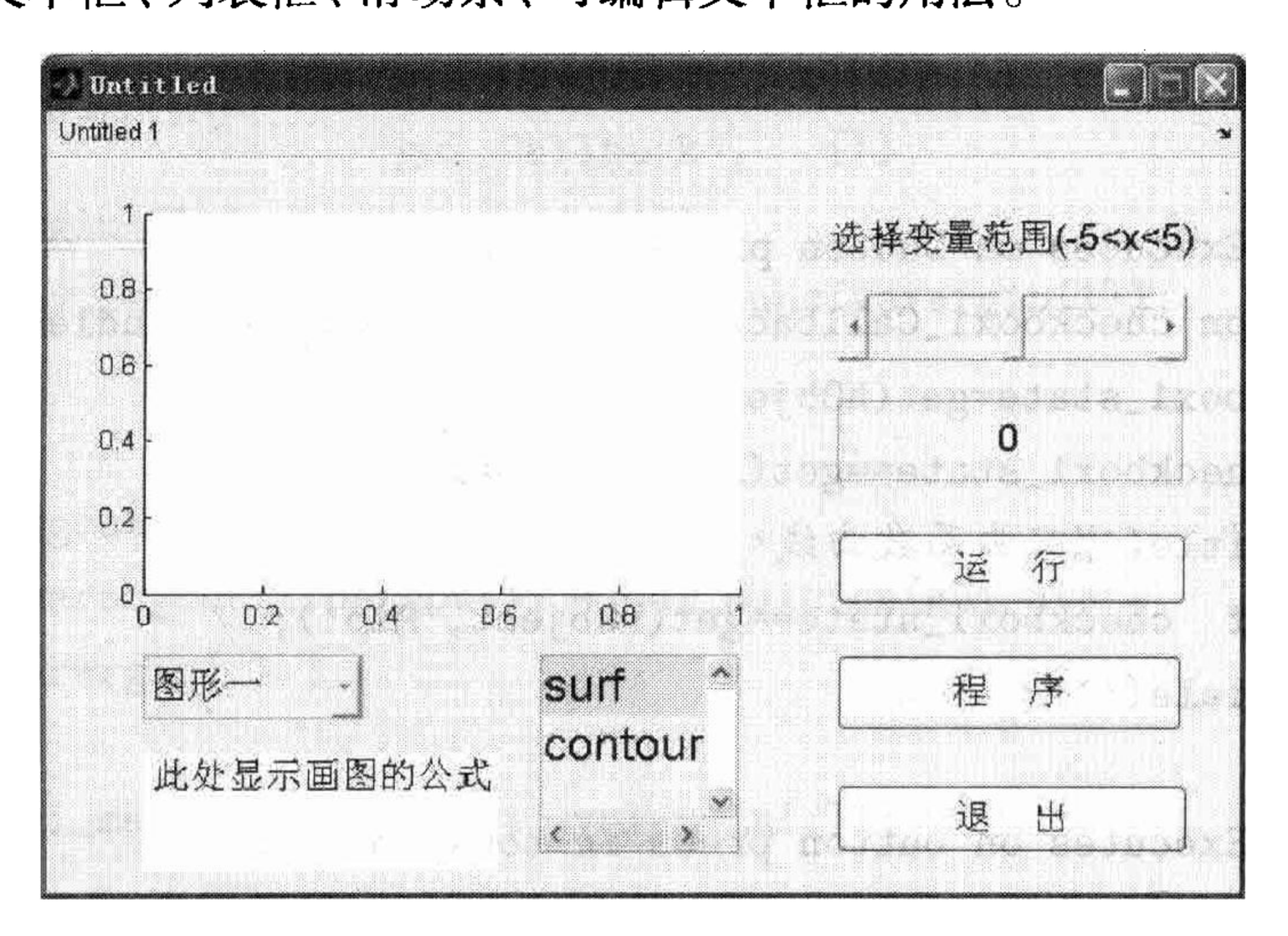


图 1.46 设计完成的图形界面

制作过程与前相同,设置的控件如图1.47所示。然后设置控件的属性，凡是在图形界面上出现的文字都是在guide中填写。由于弹出式菜单和列表框都是多选

项，设置时要注意，在输入文字“图形一图形二”以后，要点击它旁边的坐标，打开一个对话框，在对话框中用回车键将文字“图形二”移到下一行，这样才能保证它不会和文字“图形一”出现在一起。然后关闭对话框。设置列表框的方法相同。

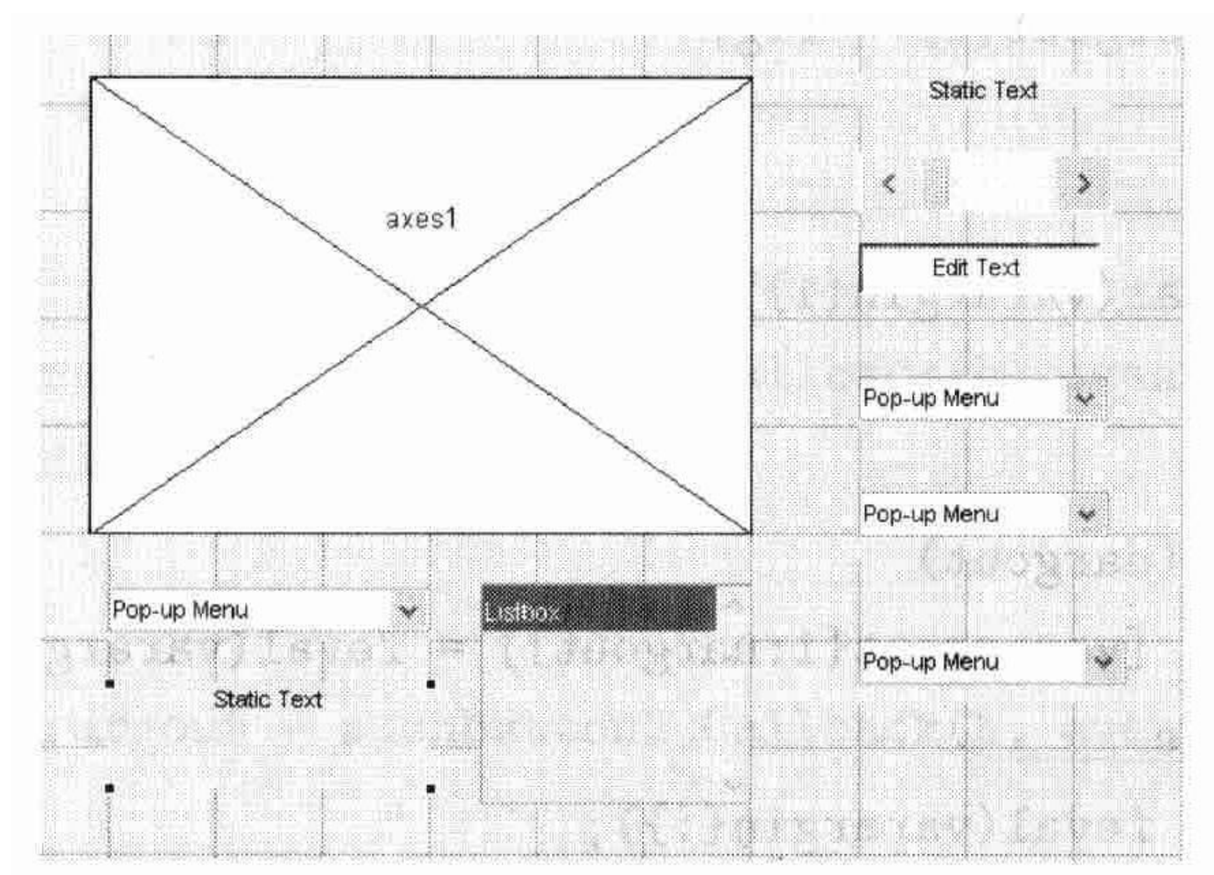


图 1.47 布置好控件的界面

各个控件的功能在M文件中完成。设计的思路是：

用弹出式菜单来选择不同的公式，选择值保存在popval中，指令switch根据popaval 的值来将所选择的公式显示在静态文本框中。

作图方式有surf和contour两种，在列表框中选择。选择值保存在vlist中。定义cell列阵来保存两个画图指令，根据vlist 的值来选择要用哪一个作图指令，并用eval来执行。

作图变量的变化范围用滑动条和可编辑文本框来设定，二者是联动的，使用其中任何一个都有相同的联动的方式。当其中一个改变以后，M文件中有相应命令来设定另一个的值。

所有选项设置好以后，就可以点击按钮“运行”来运行程序，图形会画在图形框中。另外还可用单功能按钮来查看程序和退出界面。

M文件中所有需要改写的语句都在下面。

```
%界面二的M文件
function varargout = huaT(varargin)

if nargin == 0

    fig = openfig(mfilename,'reuse');
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    handles = guihandles(fig);
```

```
        guidata(fig, handles);

        if nargout > 0
            varargout{1} = fig;
        end

    elseif ischar(varargin{1})

        try
            if (nargout)
                [varargout{1:nargout}] = feval(varargin{:});
            else
                feval(varargin{:});
            end
        catch
            disp(lasterr);
        end

    end

    function varargout = text2_Callback(h, eventdata, handles,
    varargin)

    function varargout = popupmenu1_Callback(h, eventdata, handles,
    varargin)

    function varargout = listbox1_Callback(h, eventdata, handles,
    varargin)

    % ---------------------------------------------------------------
    function varargout = slider1_Callback(h, eventdata, handles,
    varargin)
    val2=num2str(get(handles.slider1,'value'));   %与可编辑文本框联动
    set(handles.edit2, 'string', val2);
```

```
% ---------------------------------------------------------------
function varargout = edit2_Callback(h, eventdata, handles,
varargin)
val1=str2double(get(handles.edit2,'string'));        %与滑动条联动
set(handles.slider1, 'value', val1);

% ---------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles,
varargin) edit huaT

% ---------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles,
varargin)
close gcf

% ---------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles,
varargin)
popval=get(handles. popupmenu1,'value');        %从弹出式菜单获值
val1=str2double(get(handles.edit2,'string'));
                                                %从可编辑文本框中获值
[X,Y]=meshgrid(-5:0.2:val1); switch popval
      case  1
          set(handles. text2,'string','Z=X.*(-X.^2-Y.^2)');
          Z=X.*exp(-X.^2 - Y.^2);
      case  2
          str2='R=sqrt((2*X).^2+(2*Y).^2)+eps; Z=sin(R)./R;'
          set(handles. text2,'string',[str2,sprintf('\n ')]);
                                                %文字分行显示
          R=sqrt((2*X).^2+(2*Y).^2)+eps;
          Z=sin(R)./R;
end

vlist=get(handles.listbox1,'value');                %有两个值1,2
liststr={'mesh(X,Y,Z),','contour(X,Y,Z)'}; %用字符串定义作图语句
```

```
cla
eval(liststr{vlist});                                    %选取作图命令
hold on;                                                 %保持三维的作图方式
```

图形用户界面可以直接用指令编写，这时只要一个程序文件即可，有关的指令可以参看其他书籍。下面给出上述两个界面的编写程序。程序中基本是两个步骤，第一步是开启一个图形窗口，对窗口中一般属性进行统一设置。如果想知道一个图形窗口的默认属性，可以在开启窗口后，运行指令get(gcf)。第二步是在图形窗口中设置各种控件及其功能。

用指令编写的两个图形界面程序如下：

```
%界面一的程序
figure
%对当前图形窗口的属性进行设置
set(gcf,'numbertitle','off','name','%画双曲函数');
set(gcf,'unit','normalized','position',[ 0.2 0.2 0.6 0.5]);
set(gcf,'defaultuicontrolunits','normalized');
set(gcf,'defaultuicontrolfontsize',15);          %统一设置控件字体
axes('position',[ 0.074 0.12 0.6 0.75]);
t=-3:0.01:3;
%设置控件及其功能
h_radio1=uicontrol(gcf,'style','radio','value',0,'position',
[0.74 0.75 0.23 0.09],...
   'string','%画正弦双曲',...
   'callback',['set(h_radio2,''value'',0),''plot(t,sinh(t)),']);
h_radio2=uicontrol(gcf, 'style', 'radio', 'position', [0.74 0.57
0.23 0.09],...
   'string','%画余弦双曲',...
   'callback',['set(h_radio1,''value'',0)','plot(t,cosh(t)),']);
h_text=uicontrol(gcf,'style','text','visible','off',...
   'position',[0.22 0.9 0.3 0.06],'string','%双曲函数曲线');
vcheck=0;
h_check=uicontrol(gcf,'style','check',...
   'position',[ 0.74 0.39 0.23 0.09],...
   'string','%加注标题',...
   'callback',['vcheck=mod(vcheck+1,2);',...
       'if(vcheck==1),set(h_text,''visible'',''on'');',...
```

```
        'else,set(h_text,''visible'',''off''),end;']);
h_button=uicontrol(gcf,'style','toggle','position',[ 0.755 0.18
0.2 0.12],...
   'callback','grid','string','%画网格');
f = uimenu('Label','%管理');                              %建立菜单
    uimenu(f,'Label','%程序','Callback','edit huasqgui');
    uimenu(f,'Label','%退出','Callback','close gcf');
%界面二的程序
figure
clf reset
set(gcf,'menubar','none');
set(gcf,'numbertitle','off','name','Untitled');
set(gcf,'unit','normalized','position',[ 0.03 0.08 0.85 0.85]);
set(gcf,'defaultuicontrolunits','normalized');
set(gcf,'defaultuicontrolfontsize',14);
axes('position',[0.08 0.4 0.55 0.54]);
str='%此处显示画图的公式';
str1='Z=X.*(-X.^2-Y.^2)';
str2 ='R=sqrt((2*X).^2+(2*Y).^2) +eps; Z=sin(R)./R;';
h_popup=uicontrol(gcf,'style','popup',...
   'position',[0.06 0.275 0.33 0.05],...
   'max',2,...
   'string',['%图形一|图形二']);
h_list=uicontrol(gcf,'style','list',...
   'position',[ 0.43 0.05 0.22 0.274],'max',2,...
   'fontsize',14,...
   'string','surf|contour');
h_text1=uicontrol(gcf,'style','text',...
   'position',[ 0.69 0.85 0.285 0.07],...
   'horizontal','left',...
   'string','%选择变量范围(-5<x<5)');
h_text2=uicontrol(gcf,'style','text',...
   'position',[ 0.06 0.05 0.33 0.17],'max',3,...
   'backgroundcolor','w',...
   'string',str);
```

```
h_slider=uicontrol(gcf,'style','slider',...
   'min',-5,'max',5,'value',0,...
   'position',[0.69 0.7 0.285 0.08],...
   'callback',['w1=get(gcbo,''value'');',...
       'set(h_edit,''string'',num2str(w1))']);
h_edit=uicontrol(gcf,'style','edit',...
   'position',[ 0.69 0.55 0.285 0.08 ],...
   'backgroundcolor','w','string',0,...
   'callback',['w2=get(gcbo,''string'');',...
       'set(h_slider,''value'',str2double(w2))']);

h_push1=uicontrol(gcf,'style','push',...
   'position',[0.69 0.4 0.285 0.08],...
   'string','%开始','callback','huat1(h_popup,h_list,h_edit,
   h_text2)');
h_push2=uicontrol(gcf,'style','push',...
   'position',[0.69 0.25 0.285 0.08],...
   'string','%程序','callback','edit huat1');
h_push3=uicontrol(gcf,'style','push',...
   'position',[0.69 0.1 0.285 0.08],...
   'string','%退出','callback','close');
```

下面是界面二程序中调用的程序huat1.m:

```
function huat1(h_popup,h_list,h_edit,h_text2)
popval=get(h_popup,'value');
val1=str2double(get(h_edit,'string'));
[X,Y]=meshgrid(-5:0.2:val1);
switch popval
case  1
    set(h_text2,'string','Z=X.*(-X.^2-Y.^2)');
    Z=X.*exp(-X.^2 - Y.^2);
case  2
    str2='R=sqrt((2*X).^2+(2*Y).^2) +eps; Z=sin(R)./R;';
    set(h_text2,'string',[str2,sprintf('\n ')]);
   R=sqrt((2*X).^2+(2*Y).^2)+eps;
    Z=sin(R)./R;
```

```
end

vlist=get(h_list,'value');
liststr={'mesh(X,Y,Z),','hold on,contour(X,Y,Z),hold off'};
vv=zeros(1,2); vv(vlist)=1;

if vv(1)&vv(2)                                    %有两项选择的情况
        cla
        for k=1:2
            if vv(k)
                eval(liststr{k});hold on;
            end                                   %按列表选项影响图形
        end
        hold off
else                                              %有两项选择的情况
       cla
       for k=1:2
            if vv(k)
                eval(liststr{k});
            end                                   %按列表选项影响图形
        end
end
```

§1.8 浮点运算与数值计算的误差

使用MATLAB将0.1相加10次，得到的结果用16位数字的浮点数（long e格式）表示为

$$0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1=9.999999999999999e-001$$

而精确值应该是1。再计算下列式子：

```
a=4/3
b=a-1
c=3*b
d=1-c
```

得到的结果分别是：

```
a = 1.333333333333333e+000
b = 3.333333333333333e-001
c = 9.999999999999998e-001
d = 2.220446049250313e-016
```

本来d的精确值应该是0。为什么会出现这样的结果？这正是浮点数运算产生的现象。了解浮点数运算的一些知识，就能理解这种现象，并在以后的数值计算中注意它的影响。

进行加减乘除等算术运算就会用到实数，而实数的定义包含有连续、极限与无限等数学概念，这些概念在数值计算中并不适合。所以在MATLAB和大多数科学计算环境中使用的都是浮点算术体系，它是一个有限精度的有限数的集合，是分立的不连续的数，这会在计算时产生舍入(roundoff)、下溢(underflow)和上溢(overflow)等现象。因此，使用MATLAB时应该注意到这些缺陷。

由于计算机基本上是使用二进制，1985年IEEE标准委员会和美国国家标准局共同为二进制浮点数体系采用了ANSI/IEEE 754-1985号标准。MATLAB使用IEEE 的双精度格式，不过MATLAB7也能支持单精度浮点数运算。单精度格式可以节省存储空间，但是计算速度较慢。

非零的规范化的浮点数可以表示成

$$x = \pm(1+f)\cdot 2^e$$

其中e是指数，$0 \leqslant f < 1$称之为尾数，f最多只能用52位的二进制数表示，即有

$$0 \leqslant 2^{52} f < 2^{52}$$

指数e的取值范围是

$$-1022 \leqslant e \leqslant 1023$$

这种表示可以在某个固定长度的存储空间内表示定点数无法表示的更大范围的数。显然，f的有限取值造成了对浮点数精度上的限制，而e的有限取值造成了对浮点数取值范围的限制。也就是说，在浮点数运算体系里，实际使用的是有限个离散的数，而且这些数之间的间距并不相等。任何一个数如果不能正好用这个表达式表示，那么只好用一个能表示成这个表达式的近似的数来表示。

一个双精度浮点数存储于一个64位的字(word)中，其中52位存f，11位存e，1 位表示数的正负号。存储$e = +1023$时不必考虑正负号。因为e的负值的数值范围为1到$1022 = 2^{10} - 2$。e取0或取$1023 = 2^{10} - 1$的两个值被保留用来表示浮点数以外的值。后面会加以介绍。

浮点数的整个小数部分不是f而是$1 + f$，它占53位。然而首位的1并不需要存储，这样IEEE格式便将65位的信息打包成64位的一个字。

举一个较简单的例子，按照浮点数的表示，设存储 f 的位数用 t 表示，并取 $t = 3$，再假定指数e取$-4 \leqslant e \leqslant 3$，生成的浮点数如图1.48所示。

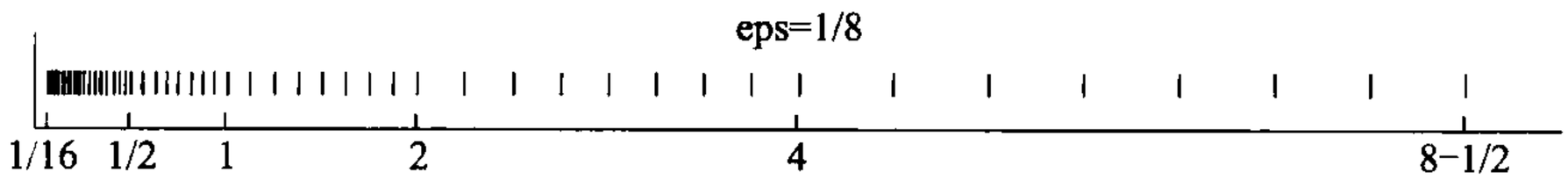


图 1.48 浮点数分布图

在每个二进制区间 $2^e \leqslant x \leqslant 2^{e+1}$ 中，数字按间隔 2^{e-t} 等距离排列。例如当 $e=0$且$t=3$时，1 和2之间的数的间隔为1/8，注意到f的取值为0,1,2,3,4,5,6,7,则其中的数为9/8,10/8,11/8,12/8,13/8,14/8,15/8。当e变大时，这个间隔也会变大。

在对数刻度显示下则可以看到每个二进制间隔中数的分布是一样的。图 1.49 就是$t=5, -4 \leqslant e \leqslant 3$ 生成的对数刻度的浮点数分布图。

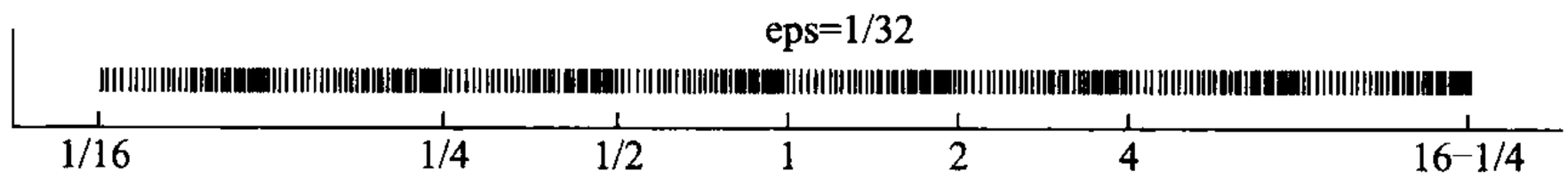


图 1.49 对数刻度的浮点数分布图

在MATLAB中有一个量叫eps，它是machine epsilon的缩写。它表示的是从1到下一个浮点数的距离。对于IEEE双精度系统

$$\text{eps} = 2^{-52}$$

换成十进制的近似数为2.2204×10^{-16}。eps/2或eps通常被称为舍入误差级别。当一个计算结果用最接近的浮点数来近似时，可能造成的最大相对误差为eps/2，而两个浮点数的最大相对间距为eps。不论何种情况，舍入误差级别大约是16位十进制数。

现在可以解释本节开头的例子。在MATLAB中要表示

$$t = 0.1$$

就必须进行舍入，因为用二进制表示十进制的分数1/10需要一个无穷级数，所以存储于t的数值并不精确地等于0.1。事实上，

$$\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \cdots$$

在第一项之后，后续项的系数按1,0,0,1重复出现，根据这个规律以4项为一组进行合并后，可得到一个基为16，或十六进制的序列。

$$\frac{1}{10} = 2^{-4} \times \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \frac{9}{16^4} + \cdots\right)$$

需要在二进制表达式的第52项或十六进制表达式的第13项截断这个无穷级数的小数部分，然后进行向上或向下舍入，才能得到1/10的浮点数近似值。因此

$$t_1 < \frac{1}{10} < t_2$$

其中

$$t_1 = 2^{-4} \times \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \cdots + \frac{9}{16^{12}} + \frac{9}{16^{13}}\right)$$

$$t_2 = 2^{-4} \times \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \cdots + \frac{9}{16^{12}} + \frac{10}{16^{13}}\right)$$

可以发现1/10更接近于t_2，因此$t = t_2$，即

$$t = (1 + f)2^e$$

其中

$$f = 1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \cdots + \frac{9}{16^{12}} + \frac{10}{16^{13}}$$

$$e = -4$$

可见，存于t中的数非常接近于0.1，但是不精确地等于0.1。这种差别有时很重要，例如计算0.3/0.1中并不精确等于3，因为实际的分子比0.3小一点，而实际的分母比0.1大一点。本节开头的计算表明，长度为t的10步并不精确的等于长度为1的一步。在MATLAB中是进行仔细处理以后才使得向量

$$0 : 0.1 : 1$$

最后一个元素精确地等于1。同样，在本节开头计算a, b, c, d时，也是由于计算中产生了舍入，所以最后结果d不等于零。

舍入误差级别eps有时被称为“浮点零”，但这是用词不当，因为有许多远小于eps的浮点数。最小的规范化的浮点数为$f = 0$，$e = -1024$，最大的浮点数是当f比1略小时且$e = 1023$。在MATLAB中这些数被称为realmin，realmax。它们和eps一起组成了这个标准体系。

	二进制	十进制
eps	2^{-52}	2.2204e-16
realmin	2^{-1022}	2.2251e-308
realmax	$(2-\text{eps})2^{1023}$	1.7977e+308

如果出现计算结果大于realmax的情况，称为上溢出。这个计算结果为一个特殊浮点数，称为无穷大（infinity）或Inf，表示为$f = 0$，$e = 1024$，并满足关系

1/Inf=0, Inf+Inf=Inf

还可能计算出一个在实数系统中从未定义的值,这个特殊值叫“非数”或NaN。出现这种情况的例子有0/0和Inf−Inf,在浮点数体系中表示为$e = 1024$和f非零。

如果出现计算结果小于realmin的情况,称为“下溢出”。这涉及IEEE标准中一个可选的但是有争议的方面。有很多计算机允许在realmin和eps*realmin之间有例外的非规范或次规范的浮点数。最小的正非规范数大约为0.494e-323,任何小于它的数都设为0。在没有定义非规范数的机器里,任何小于realmin的数都设为0。非规范数填充了0和最小正数之间的空隙。这样提供了处理下溢出的简洁办法,但对于MATLAB的计算,其重要性不大。采用$e = -1023$来表示非规范数,这样实际存储的指数e+1023为零。

对于整数,MATLAB也是用浮点数系统来处理。数3和数3.0在数学上是完全一样的,不过许多计算机编程语言采用不同的方式来表示它们,而MATLAB并不区分它们。有时人们会用浮点整数(flint)来描述值为整数的浮点数。只要计算结果不太长,浮点整数的计算不会有舍入误差。若结果不超过2^{53},浮点整数的加、减、乘、除是精确的,且结果仍为浮点整数。若涉及浮点整数的除法和平方根的计算,结果为整数时,也用浮点整数来表示。例如sqrt(363/3)的结果为11,计算中不会出现舍入误差。

在数值计算中,除了上面介绍的机器运算所带来的舍入误差之外,还会有其他一些误差来源,主要的有:

(1) 模型误差

当我们用一个数学模型去描述一个实际的物理系统时,总会忽略一些我们认为是次要的因素,这样的模型必然与实际系统有偏差。

(2) 观测误差

模型中使用的参数往往来自观测值,测量必然有误差,这些误差会带到计算中来。

(3) 计算方法带来的误差

数值计算的方法很多都是近似方法,例如计算无穷级数的和,可能只取若干项之和作为结果,这样也会产生误差。

(4) 计算过程中的舍入误差

例如我们前面介绍的浮点数计算中所产生的误差。

计算过程中这些误差可能积累与传播,最后会影响结果的准确性。所以在得到计算结果后,应该检查结果的可靠性。

§1.9 物理应用:Logistic模型的周期分岔与混沌现象

现在我们已经有能力用MATLAB来研究一些物理问题了。下面就是一个很著名的非线性物理中的问题。

Logistic模型是一个描述自然界生物种群中昆虫数目变化的数学模型，它由上一年的昆虫数目x_n及增殖系数μ预测次年的昆虫数目x_{n+1}。其数学表达式为

$$x_{n+1} = \mu(x_n - x_n^2) \qquad (0 < \mu < 4,\ 0 < x < 1)$$

下面研究这个模型反映出来一些生物繁殖的规律。

1. 数值迭代的不同结果

给定相同初值 $z_0 = 0.6$，取 $\mu = 2,\ 3.2,\ 3.45,\ 3.6$，进行迭代计算得到：

$\mu = 2$，x 趋于 0.5。结论：没有分岔；

$\mu = 3.2, x$ 趋于 0.51304，0.79946。结论：周期 2 分岔；

$\mu = 3.45, x$ 趋于 0.38282，0.50088，0.82694，0.875。结论：周期 4 分岔；

$\mu = 3.6, x$的取值没有明确的趋向。结论：趋于混沌。

对应的图形分别为图1.50至图1.53。

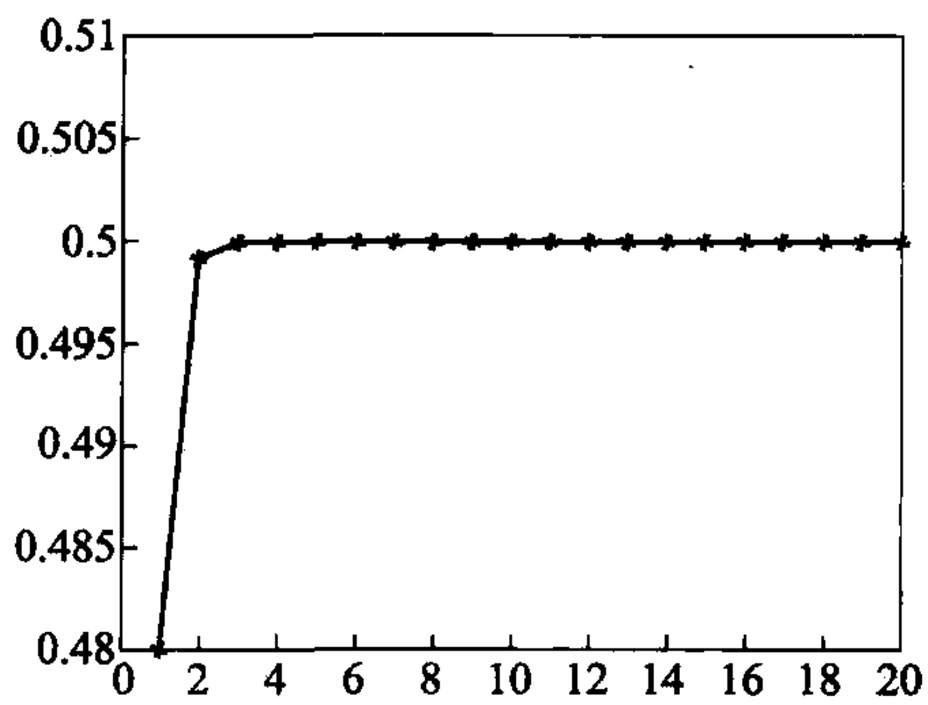


图 1.50 $\mu = 2$

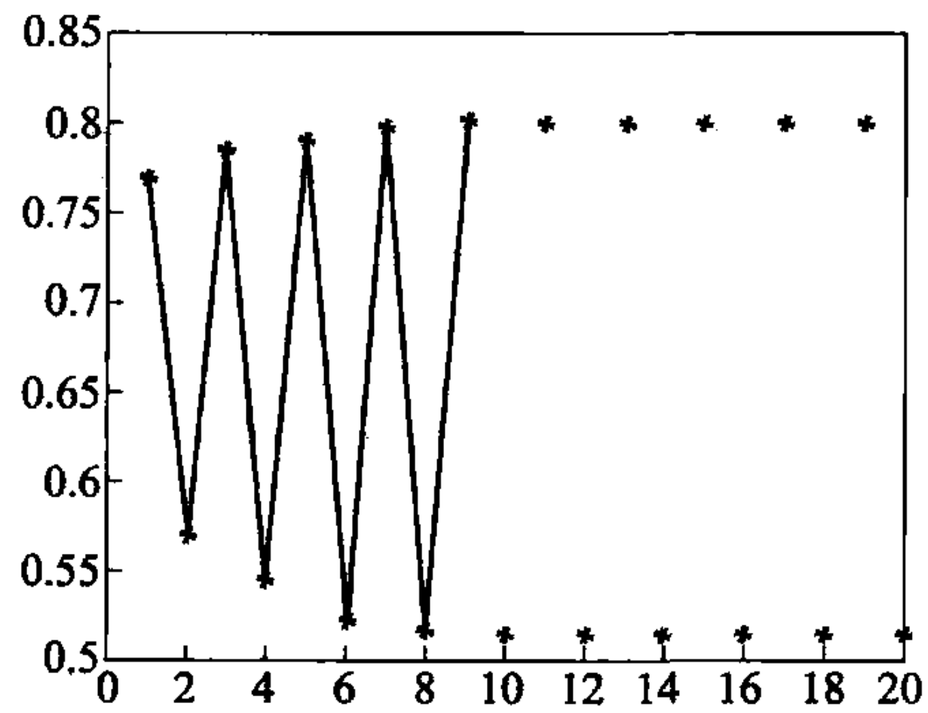


图 1.51 $\mu = 3.2$

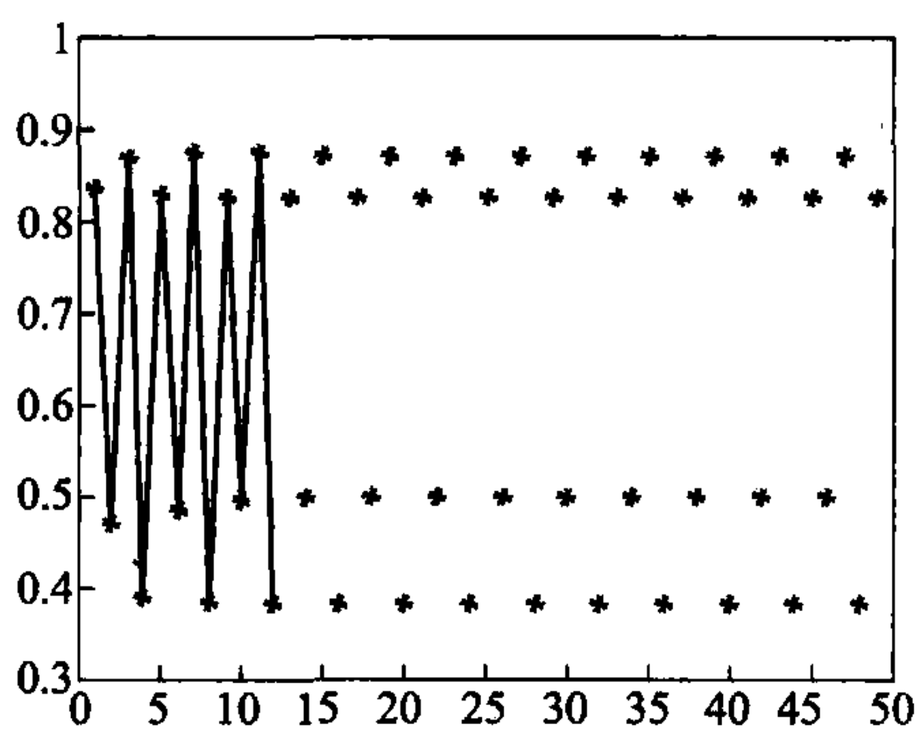


图 1.52 $\mu = 3.45$

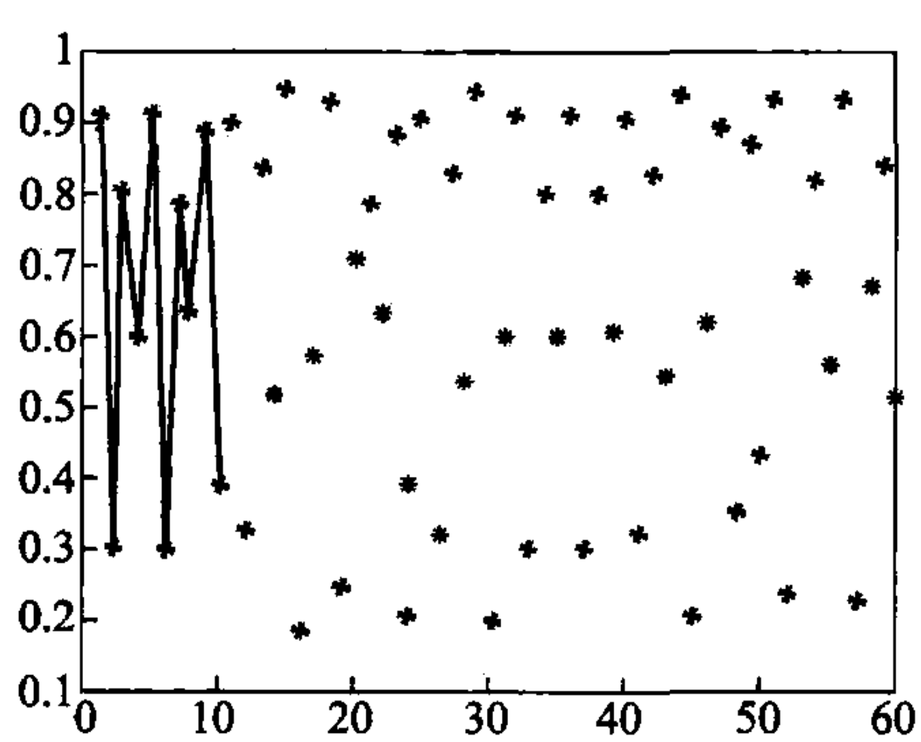


图 1.53 $\mu = 3.6$

改变初始值z_0，所看到的结果相似。

2. 费根鲍姆图

为了研究μ值对分岔现象的影响，可以取不同的初值对所有的μ进行计算。方法是给定x初值，对不同的μ值循环计算新的x，共循环250次，循环计算150次后开始画图。计算所得的x值是矩阵，行标对应循环次数，列标对应μ值。使用矢量化编程以后，对所有的μ值同时计算一次新的x值，得到矩阵x中的一列元素。所得图形如图1.54所示，叫做费根鲍姆（Feigenbaum）图。

在图形上反映的倍周期分岔现象是

$$3.000 < \mu < 3.4495 \text{，二周期循环；}$$
$$3.4496 < \mu < 3.5441 \text{，四周期循环；}$$
$$3.5441 < \mu < 3.5644 \text{，八周期循环；}$$
$$3.5644 < \mu < 3.5688 \text{，十六周期循环；}$$
$$\cdots\cdots\cdots\cdots$$

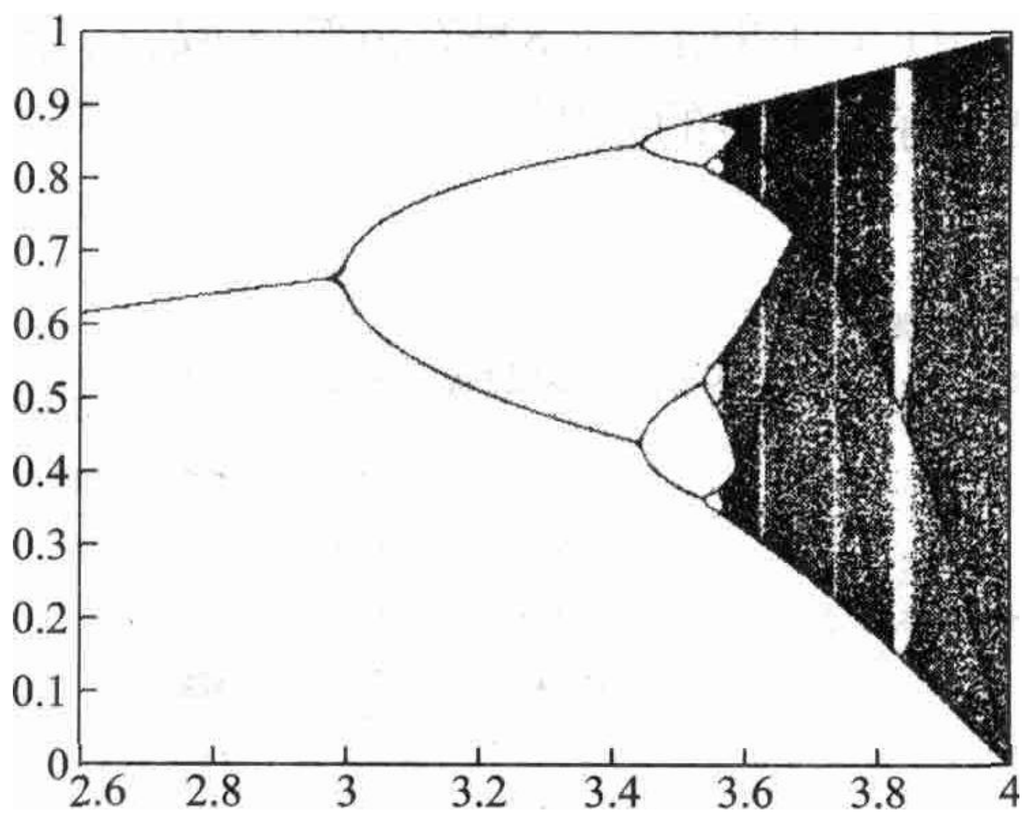


图 1.54 费根鲍姆图

如此一直分岔下去，直到进入混沌。

为了说明编程技巧，下面采用两种方法编程。两个程序不同之处在于是否保留所有的x值。在程序2中预先设置一个矩阵存放x的元素可以节约扩充矩阵元素的时间。在Workspace中可以查看这两个程序中x的区别，设置间断点以后运行程序，可以看到程序中各个语句的运行结果。

```
% bug1.m
u=2.6:0.001:4;   x=0.6;
for j=1:150,
   x=u.*(x-x.^2);
end
```

```
% bug2.m
u=2.6:0.001:4;
X=ones(250,1401);
X(1,:)=0.6*X(1,:);
for j=1:250
```

```
for i=1:100
   x=u.*(x-x.^2);
   plot(u,x,'r.')
   hold on;
end
```

```
X(j+1,:)=u.*(X(j,:)-X(j,:).^2);
end
plot(u,X(150:end,:),'r.')
```

程序1运行时间0.28秒。不保留旧的x值，而是直接用它画图，能节约内存。

程序2运行时间0.15秒，比程序1快。保留所有X值，每次计算的X值生成矩阵的一行元素，最后用矩阵X的后150行作图，程序2可读性强，占用较大内存。

运用编辑器窗口菜单下的Tolols/Open Profiler可以查看各部分程序的运行时间，便于有针对性地改进程序。从两个程序的分析报告（图 1.55 和图1.56）可以看到，程序1重复作图100次，所用时间与程序2全部运行时间相当。所以程序1比程序2更费时间。

混沌现象有个特点，就是初值的微小变化将引起结果的完全不同，它说明混沌现象的不可预测性。图1.57中的$\mu = 3.8$是在混沌区域，z_0分别取0.6001和0.6002，所得到迭代结果很快就有了明显的差别。

Profile Summary

Generated 23-Mar-2006 17:24:59 using real time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
BUG	1	0.469 s	0.094 s	
newplot	100	0.281 s	0.031 s	
newplot>ObserveAxesNextPlot	100	0.109 s	0.016 s	
gcf	200	0.094 s	0.094 s	
graphics\private\clo	1	0.094 s	0.016 s	
hold	100	0.094 s	0.063 s	
setdiff	2	0.063 s	0.063 s	
newplot>ObserveFigureNextPlot	100	0.031 s	0.031 s	
ancestor	100	0.031 s	0.016 s	
gca	200	0.016 s	0.016 s	
graphics\private\clo>find_kids	1	0.016 s	0.016 s	
axes (Opaque-function)	100	0.016 s	0.016 s	
display (Builtin-function)	3	0 s	0.000 s	
findall	1	0 s	0.000 s	

图 1.55 程序1的分析报告

Profile Summary

Generated 23-Mar-2006 17:40:54 using real time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
BUG	1	0.281 s	0.156 s	
newplot	1	0.125 s	0.016 s	
newplot>ObserveAxesNextPlot	1	0.078 s	0.000 s	
graphics\private\clo	1	0.078 s	0.000 s	
setdiff	2	0.078 s	0.062 s	
gcf	1	0.031 s	0.031 s	
ismember	1	0.016 s	0.016 s	
display (Builtin-function)	3	0 s	0.000 s	
newplot>ObserveFigureNextPlot	1	0 s	0.000 s	
gca	1	0 s	0.000 s	
graphics\private\clo>find_kids	1	0 s	0.000 s	
findall	1	0 s	0.000 s	
isappdata	2	0 s	0.000 s	
isfield	2	0 s	0.000 s	
datenummx (MEX-function)	2	0 s	0.000 s	
handle (Builtin-function)	23	0 s	0.000 s	

图 1.56 程序2的分析报告

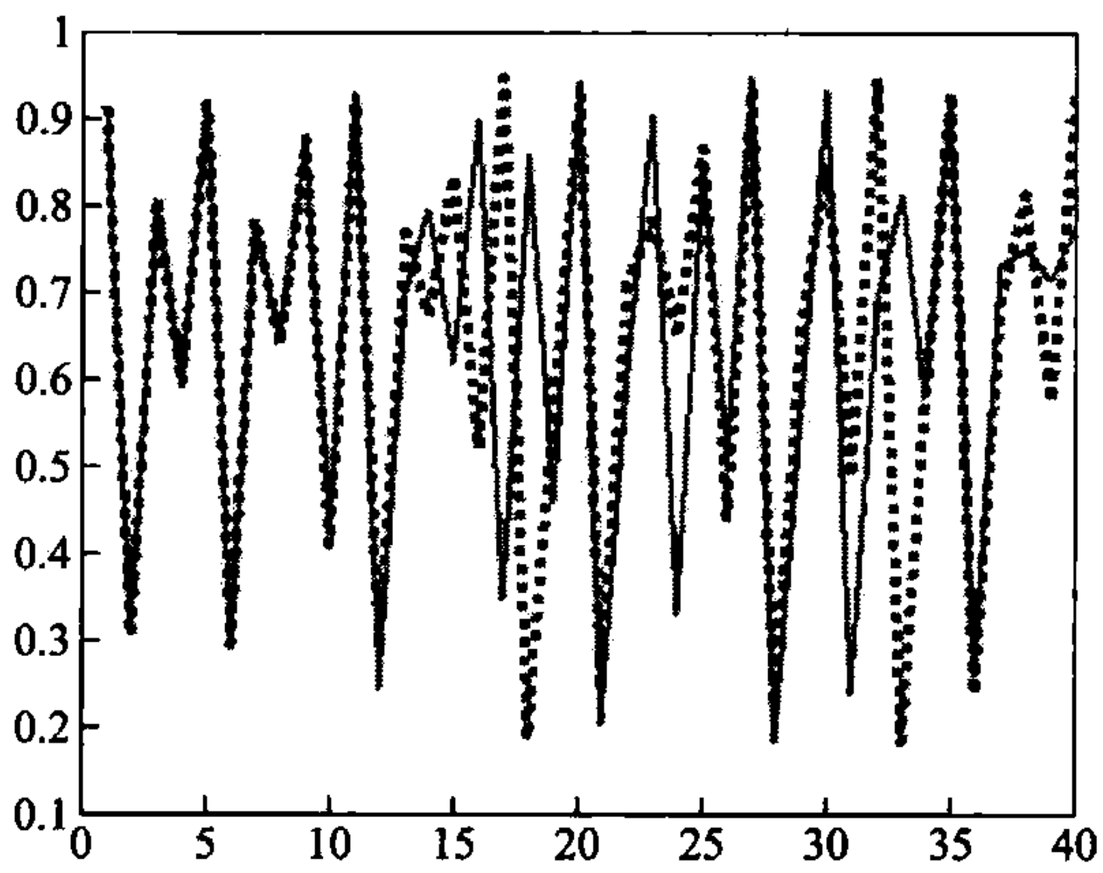


图 1.57 混沌现象对初值的敏感性

3. 费根鲍姆δ常数

利用分岔点的μ值，计算相邻分岔点的间距之比，所得的极限值叫费根鲍姆δ常数。

$$F_\delta = \lim_{n\to\infty} \frac{\mu_n - \mu_{n-1}}{\mu_{n+1} - \mu_n}$$

$$= 4.669201661\cdots$$

这是一个普适常数，所有的系统通过倍周期分岔进入混沌时，都会遵循这个规律。下表列出了分岔点的间距比值的变化情况。

分岔情况	分岔值μ	间距比值
$1 \to 2$	3	
$2 \to 4$	3.449487743	4.751466
$4 \to 8$	3.544090359	4.656251
$8 \to 16$	3.564407266	4.668242
$16 \to 32$	3.568759420	4.66874
$32 \to 64$	3.569691610	4.6691
$64 \to 128$	3.569891259	4.669
$128 \to 256$	3.569934019	4.669
$\vdots$	$\vdots$	$\vdots$
周期解$\to$ 混沌	3.569945672	4.669201661

4. 李雅普诺夫指数

对于一维映射$x_{n+1} = f(x_n)$，考虑从x_0及其相邻点y_0 出发的两个迭代过程，开始两点距离为$|x_0 - y_0|$，迭代一次两点距离为

$$|x_1 - y_1| = |f(x_0) - f(y_0)| \approx \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_0} |x_0 - y_0|$$

再作第二次迭代得

$$|x_2 - y_2| = |f(x_1) - f(y_1)| \approx \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_1} |x_1 - y_1| = \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_1} \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_0} |x_0 - y_0|$$

经过N次迭代得

$$|x_n - y_n| \approx \prod_{n=0}^{N-1} \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_n} |x_0 - y_0|$$

每次迭代的平均分离值为

$$\left(\prod_{n=0}^{N-1} \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_n}\right)^{1/n}$$

将上式取对数并令$n \to \infty$得到李雅普诺夫(Lyapunov)指数为

$$\lambda = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \ln \left|\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_n}$$

利用李雅普诺夫指数λ，计算出的相空间内初始时刻的两点距离将随时间(迭代次数)作指数分离：

$$|x_n - y_n| = |x_0 - y_0|\mathrm{e}^{n\lambda}$$

在一维映射中，λ只有一个值，而在多维相空间情况下一般有多个$\lambda_i(i=1,2,\cdots)$，而且沿相空间的不同方向，其λ_i值一般也不同。

下面画出一维映射λ随参数μ的变化图(图1.58)及其局部放大图（图1.59）。由图1.58中可以看到，对于稳定的周期n，有$\lambda<0$。对于倍周期分岔点，有$\lambda=0$。从$\mu=3.5699\cdots$开始产生混沌运动，由于敏感的初始条件，有$\lambda>0$。所以λ由负变正表明运动向混沌转变。

作图程序如下：

```
x=0.5;  u=3:0.0001:4;  y=0; N=300;
for j=1:N
   x=u.*(x-x.^2);
   df=log(abs(u-2*u.*x));
   y=y+df;
end
plot(u,y/N)
```

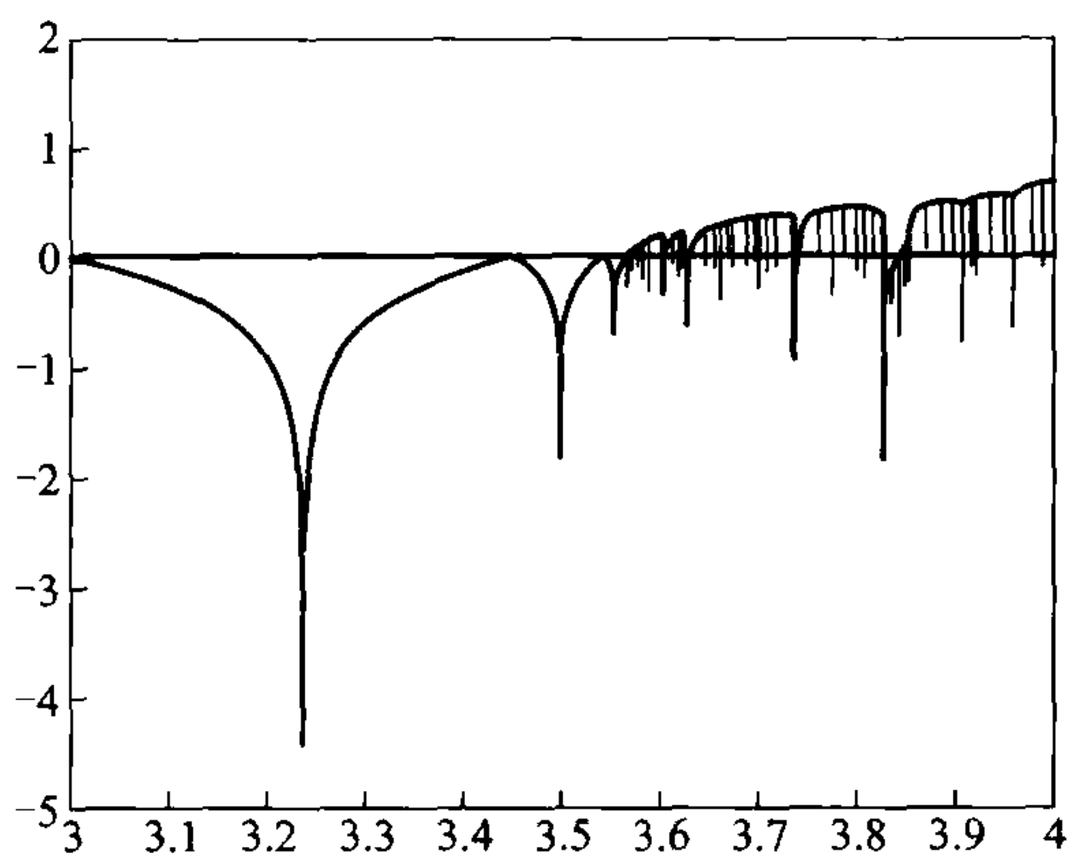


图 1.58 指数λ随参数μ的变化

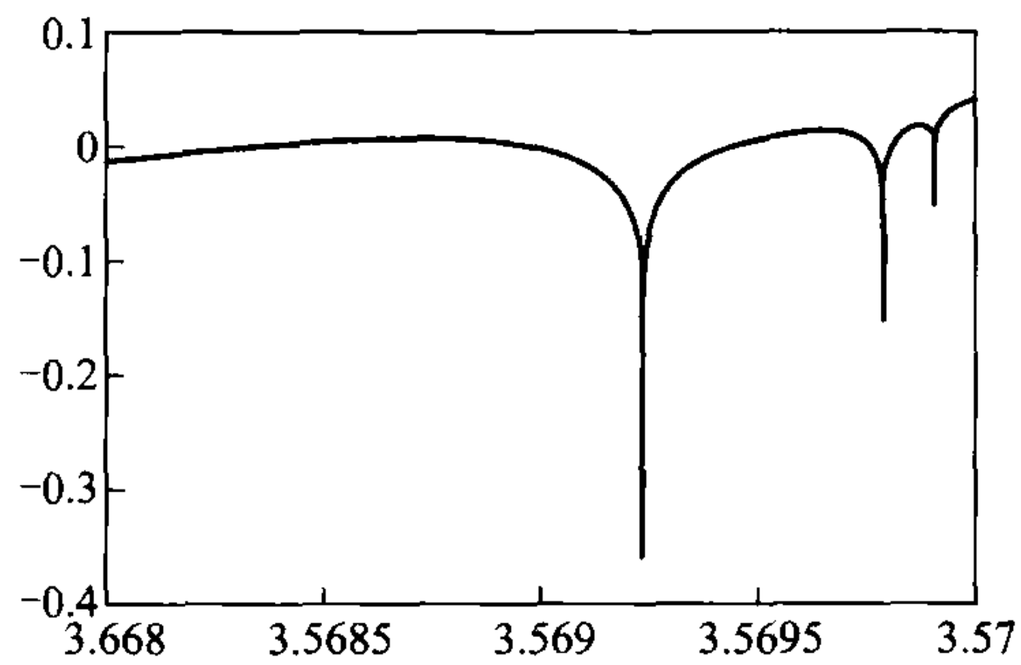


图 1.59 局部放大图

§1.10 练 习

1. 查看Start图标下的DEMO/MATLAB/Desktop environment演示，详细了解操作界面各窗口的功能。

2. 用help查看第 4 页中各个指令的用法。如help exist会显示exist的用法。

3. 以下是 MATLAB 的内部函数，请将它译成中文，并通过 help 系统学习其用法。

```
Trigonometric.
    sin       - Sine.
    sind      - Sine of argument in degrees.
    sinh      - Hyperbolic sine.
    asin      - Inverse sine.
    asind     - Inverse sine, result in degrees.
    asinh     - Inverse hyperbolic sine.
    cos       - Cosine.
    cosd      - Cosine of argument in degrees.
    cosh      - Hyperbolic cosine.
    acos      - Inverse cosine.
    acosd     - Inverse cosine, result in degrees.
    acosh     - Inverse hyperbolic cosine.
    tan       - Tangent.
    tand      - Tangent of argument in degrees.
    tanh      - Hyperbolic tangent.
    atan      - Inverse tangent.
    atand     - Inverse tangent, result in degrees.
    atan2     - Four quadrant inverse tangent.
    atanh     - Inverse hyperbolic tangent.
    sec       - Secant.
    secd      - Secant of argument in degrees.
    sech      - Hyperbolic secant.
    asec      - Inverse secant.
    asecd     - Inverse secant, result in degrees.
    asech     - Inverse hyperbolic secant.
    csc       - Cosecant.
```

```
    cscd        - Cosecant of argument in degrees.
    csch        - Hyperbolic cosecant.
    acsc        - Inverse cosecant.
    acscd       - Inverse cosecant, result in degrees.
    acsch       - Inverse hyperbolic cosecant.
    cot         - Cotangent.
    cotd        - Cotangent of argument in degrees.
    coth        - Hyperbolic cotangent.
    acot        - Inverse cotangent.
    acotd       - Inverse cotangent, result in degrees.
    acoth       - Inverse hyperbolic cotangent.

Exponential.
    exp         - Exponential.
    expm1       - Compute exp(x)-1 accurately.
    log         - Natural logarithm.
    log1p       - Compute log(1+x) accurately.
    log10       - Common (base 10) logarithm.
    log2        - Base 2 logarithm and dissect floating point
                  number.
    pow2        - Base 2 power and scale floating point number.
    realpow     - Power that will error out on complex result.
    reallog     - Natural logarithm of real number.
    realsqrt    - Square root of number greater than or equal to
                  zero.
    sqrt        - Square root.
    nthroot     - Real n-th root of real numbers.
    nextpow2    - Next higher power of 2.

Complex.
    abs         - Absolute value.
    angle       - Phase angle.
    complex     - Construct complex data from real and imaginary
                  parts.
    conj        - Complex conjugate.
```

```
imag      - Complex imaginary part.
real      - Complex real part.
unwrap    - Unwrap phase angle.
isreal    - True for real array.
cplxpair  - Sort numbers into complex conjugate pairs.

Rounding and remainder.
fix       - Round towards zero.
floor     - Round towards minus infinity.
ceil      - Round towards plus infinity.
round     - Round towards nearest integer.
mod       - Modulus (signed remainder after division).
rem       - Remainder after division.
sign      - Signum
```

4. 用 help ops 查看所有的运算符号并使用 help 系统学习不熟悉的算符的用法。

5. 输入a=exp(1)和A=π，查看在指令窗口、内存窗口、历史指令窗口所出现的变量，并比较e^{π} 和π^{e} 的大小。

6. 构造3×3的随机数矩阵A，求它的逆矩阵B，计算C=B*A和D=B.*A，然后计算C的方根E及E的绝对值。在每次运算后，查看在数据显示方式short, long, short e, long e, short g, long g 下这些结果显示的差别。

7. 利用MATLAB求当x=1,2,3,⋯,100时$y = x^2 + \sin x + \ln x$的值。

8. 用MATLAB绘制出简谐振动的位移曲线。简谐振动的运动学方程是$x = A\sin(\omega t + \phi)$ 。(要求：任意自取A，ϕ的值，但ω要取ω_1, ω_2两个值，画两条曲线，分别用红色和蓝色曲线表示。)

9. 计算下列复变函数

(1) 3+5i/(1+i); (2) sin (4+5i); (3) ln (3+2i); (4) lg(3+2i); (5) sh(3+2i).

10. 在指令窗口内输入A=[ ],B=zeros(3,3)和C=cell(3)在workspace中查看它们的各个元素是什么?

11. 生成一个3×3的幻方矩阵A。再做如下操作：

(1) 取矩阵 A 的上三角部分 a 和下三角部分 b；

(2) 求 A 的各列元素之和 c，各行元素之和 d，及各列的中值 e；

(3) 求 A 的对角矩阵 f 和转置矩阵 g；

(4) 对 A 进行以下操作：左右旋转得到 h，上下旋转得到 i，旋转 90°得到 j；

(5) 对 A 进行升序排列得到 k；

(6) 指出在运算中哪些是对各个元素进行计算，哪些是对整列元素进行计算，哪些是对整个矩阵进行计算。

12. 自己编写一个线性代数方程组，仿照教材中的例题用矩阵除法运算求解。

13. 用命令diag生成一个矩阵，要求对角元素是(1:3)。再思考一下如何生成下面的矩阵。

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{matrix} \quad 和 \quad \begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{matrix}$$

14. 已知质点的运动方程是$x = \sin\omega_1 t, y = \sin\omega_2 t$,用命令comet画出质点的运动轨迹图，要求频率比ω_1/ω_2分别是有理数和无理数。

15. 用plot用两种方法将两个三角形画在一个图形内。

(1) 实数法：顶点分别为A(−1,0)，B(0,1)，C(1,0)，用红色、线型为短划线，标志为'+'，线粗为3；

(2) 复数法：顶点分别为D(−1,0)，E(0,i)，F(1,0)，用绿色、线型为点线，标志为'o'，线粗为5。

16. 用指令comet3画出三维曲线

$$\begin{cases} x^2 + y^2 + z^2 = 64 \\ y + z = 0 \end{cases}$$

提示:改写为如下参数方程

$$\begin{cases} x = 8\cos t \\ y = 4\sqrt{2}\sin t \\ z = -4\sqrt{2}\sin t \end{cases}$$

17. 用fplot绘制曲线$\cos(\tan \pi x)$。

18. 用ezplot绘制曲线$\mathrm{e}^y + \dfrac{\cos x}{x} + y = 0$。

19. 用ezsurf绘制曲面

$$\begin{cases} x = \mathrm{e}^{-s}\cos t \\ y = \mathrm{e}^{-s}\sin t \\ z = t \end{cases} \qquad (0 \leqslant s \leqslant 8, 0 \leqslant t \leqslant 5\pi)$$

20. 用subplot画出椭球与旋转体。椭球心在(0,0,1)，三个半长轴依次为2,3,4。旋转体的母线为$1 + t^2$。

（旋转体参考画法为：`t=-1:0.1:1;[X,Y,Z]=cylinder(1+t.^2);surf(X,Y,Z)`）

21. 画曲面$z=x^2y+\dfrac{\sqrt{y}}{x}$ $(-5\leqslant x\leqslant 5,\ 0\leqslant y\leqslant 10)$的图像。

22. 分别画复变函数$\mathrm{e}^z, z^{1/5}$图形，注意观察多值函数的图形。

23. 画出四种本征函数系 $\sin(n\pi x/l)$, $\cos(n\pi x/l)$, $\sin[(n+1/2)\pi x/l]$, $\cos[(n+1/2)\pi x/l]$ 的图形，其中 n 取 1,2,3,4。再画这 4 个函数乘以时间因子 $\cos(n\pi at/l)$ 所得的驻波图形。

24. 两列行波分别从A、B两点相向而行，相遇后产生叠加，以动画表示。

25. 两个三角波分别从A、B两点相向而行，相遇后产生叠加，以动画表示。

26. 画出牛顿环的干涉图样。

27. 画出等容、等压、等温曲线。

28. 画出表面水波的干涉图样。

29. 用comet画摆轮线$x=t-R\sin t, y=R\cos t$ $(R<1)$。t为参数。

30. 画两端固定的弦的级数解图形，

$$u(x,t)=\sum_{n=1}^{\infty}\left(A_n\cos\frac{n\pi at}{l}+B_n\sin\frac{n\pi at}{l}\right)\sin\frac{n\pi x}{l}$$

其中的系数可以利用下面公式作积分计算：

$$A_n=\frac{2}{l}\int_0^l\varphi(\xi)\sin\frac{n\pi\xi}{l}\mathrm{d}\xi$$

$$B_n=\frac{2}{n\pi a}\int_0^l\psi(\xi)\sin\frac{n\pi\xi}{l}\mathrm{d}\xi$$

其中初位移φ和初速度ψ分别为

(1)

$$\varphi(x)=\begin{cases}\sin\dfrac{7\pi x}{l} & \left(\dfrac{3l}{7}\leqslant x\leqslant\dfrac{4l}{7}\right)\\ 0 & \left(x<\dfrac{3l}{7}\text{或}x>\dfrac{4l}{7}\right)\end{cases}$$

$$\psi(x)=0$$

(2)

$$\varphi(x)=0$$

$$\psi(x)=\begin{cases}1 & \left(\dfrac{3l}{7}\leqslant x\leqslant\dfrac{4l}{7}\right)\\ 0 & \left(x<\dfrac{3l}{7}\text{或}x>\dfrac{4l}{7}\right)\end{cases}$$

31. 编制一个程序，计算费根鲍姆δ常数。

第二章　迭代-分形图形

本章重点是训练编程能力，通过编程的内容也能学些一点有关分形的知识。

初学者往往觉得编程很神秘，普遍存在一种畏难情绪。但是编程是开展计算中的第一步，编程能力是很重要的基本功，这是不能省略也不能替代的学习过程。初学者一定要亲自动手，独立编程，才能培养出这种能力，仅仅看看别人编程是学不会的。熟能生巧，长期反复的实践才能锻炼出高水平的编程技巧。

在编程中要做到有足够的仔细与耐心，随时检查，避免犯一些低级的输入错误而浪费了时间，如混淆字母的大小写，混淆了分号与冒号，忽略汉字全角符号与英文半角符号的差异，在输入指令时的漏掉一个字母等不一而足。要细心阅读优秀的程序，从中学习吸收别人的经验；要勤学多练，从实践中锻炼编程的技能……

完成同一个计算工作，好的程序与差的程序在运算时间上可能相差好几倍甚至百倍，例如后面要讲树程序1 与树程序4的两个例子就是很好的对比。

优秀的程序起码应该具备以下几条：

首先，必须保证正确性，这是必要条件。这就要求模型正确，计算方法正确，计算结果与物理事实相符。为此可以用定性分析预测一下粗略的结果；尽量用解析方法对某个简单的特例计算出定量的结果与数值计算进行对比；改变参数以便对不同的结果进行对比，修改算法以验证计算的结果。在这个调试过程如何节省时间也是很重要的，一般来说，先减小计算量是很重要的，如果有循环计算，可以先减少循环的次数，如果变量是连续取值，可以先减少取值的数目等，这样可以很快看到程序运行的结果。等程序基本能运行了，再按实际需要恢复各个参数的正常值。

其次，好程序应该算得快，计算时间越短越好。所以编程时要做到，避免不必要或重复的计算，采用节省时间的算法。

最后，好程序应该是语法正确，语句简洁，条理清晰，可读性强，便于交流和移植。就是说，好的程序应该具有通用性，即类似的问题可以借用类似的程序。这就要求编程中做到：对程序中的关键语句加上注解，尤其对变量所代表的物理量要加以说明，不能让使用者去猜。应该注意使用模块化编程，即一段语句解决一个计算小问题。程序格式正确，排版规范，标点正确。还有就是应该将程序尽量集中编在一个文件中，比如用子函数文件将一些程

序吸收到主程序内以减少程序文件的数目。程序文件过多会使文件之间的关系混乱，可读性差。

§2.1 迭代与分形

计算机进行数值计算的一个基本特点就是要重复成千上万次的计算，这就是所谓的迭代计算，所以迭代计算是编程的最基本的技巧之一，在数值计算中可以说无处不在，在计算物理中也有广泛的应用，例如简单的重复的迭代可以绘制许多有趣的分形图形，这在非线性物理中常常用到。本节就通过介绍分形图形的画法来学习迭代计算的技巧，理解编程的要求。

初学者可能以为迭代计算就是运用循环语句如for，while等，没有什么可学的，其实问题并非如此简单。初学者往往由于没有掌握正确的迭代计算的技巧，编出的程序使用了不正确或不必要的循环计算，使程序运行极慢，浪费了大量的宝贵机时，还可能由于计算量太大以致计算工作无法完成。

事实上，迭代计算除了运用for循环或while循环来实现之外，还有一种方法是递归调用，就是程序反复地调用自己来完成迭代计算。而MATLAB作为一种专用软件，有一种特殊技巧叫矢量化编程，它有时可以替代for循环并且运算速度很快，前面介绍过的Logistic模型的计算就是一例。还有一种值得注意的有趣的现象是，运用物理思想来设计算法也能为迭代计算的编程提供思路，如本节所讲的利用分形图形的自相似的特点所设计的编程方法。由此可见，编程不仅仅是语句和技巧的运用，它与物理思想、模型设计、算法选择都是密切相关的。

§2.1.1 分形树

如图2.1所示，将一条线段三等分，在等分点上各画一条长度为原线段长度的三分之一的线段，该线段与原线段成固定的夹角（如$\pi/3$），就会得到一种像树枝一样的图形。

对这个图形中每一条线段都按照这种画法依次做下去，得到图2.2。

要注意的是，不管重复画多少次，尽管线段越来越密集，但永远不会填满整个面积，这是一种介于线段与面积之间的图形，也就是所谓的分形图形。而且按照这种画法，每个局部与整体都存在一种相似性，为了看清这点，可以利用MATLAB的图形放大功能。点击工具栏中放大镜以后，按住鼠标的右键在要放大的区域画一个方框，松开鼠标右键，方框的区域就会被放大，如图2.3和图2.4所示。

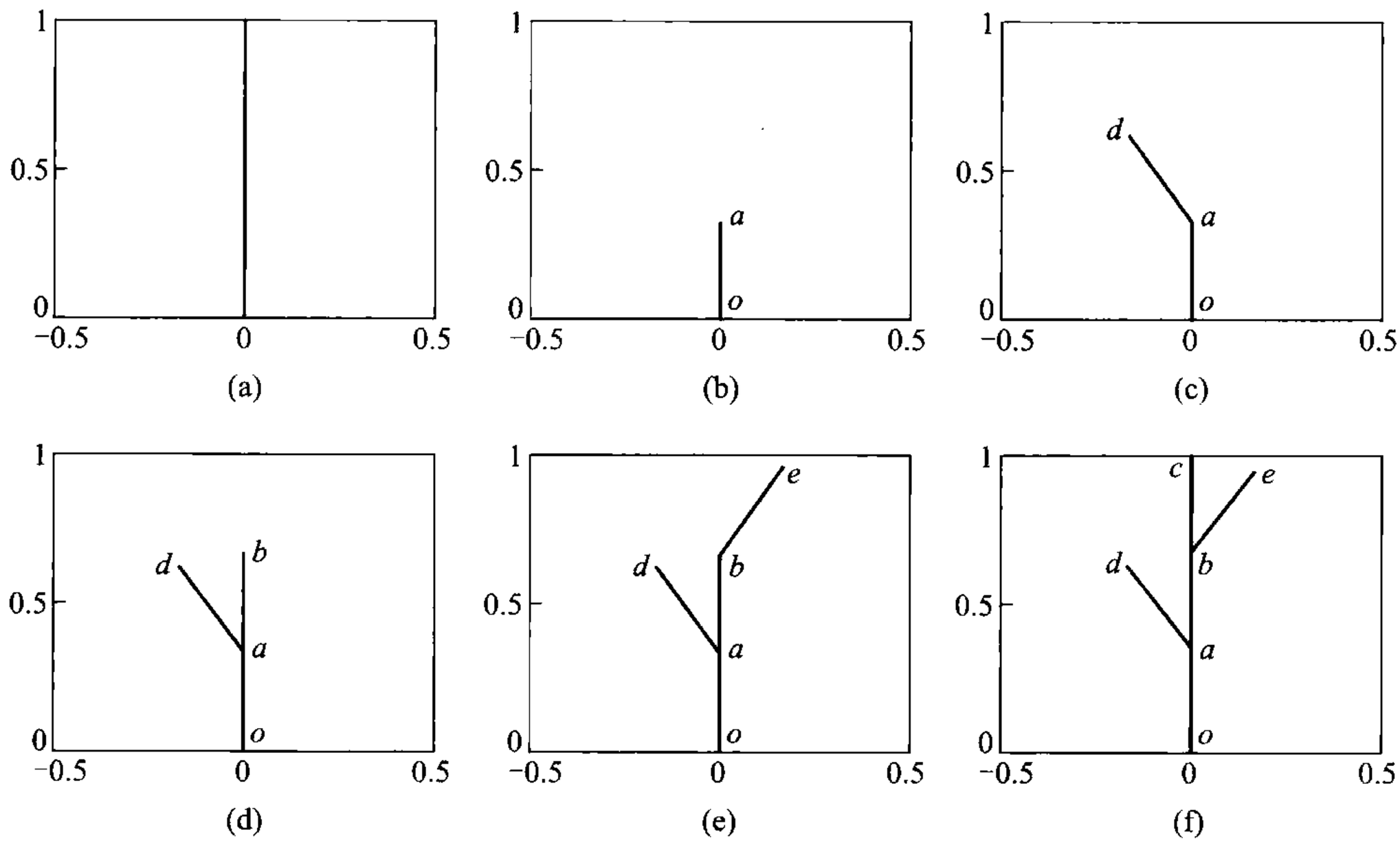


图 2.1 分形树的画法

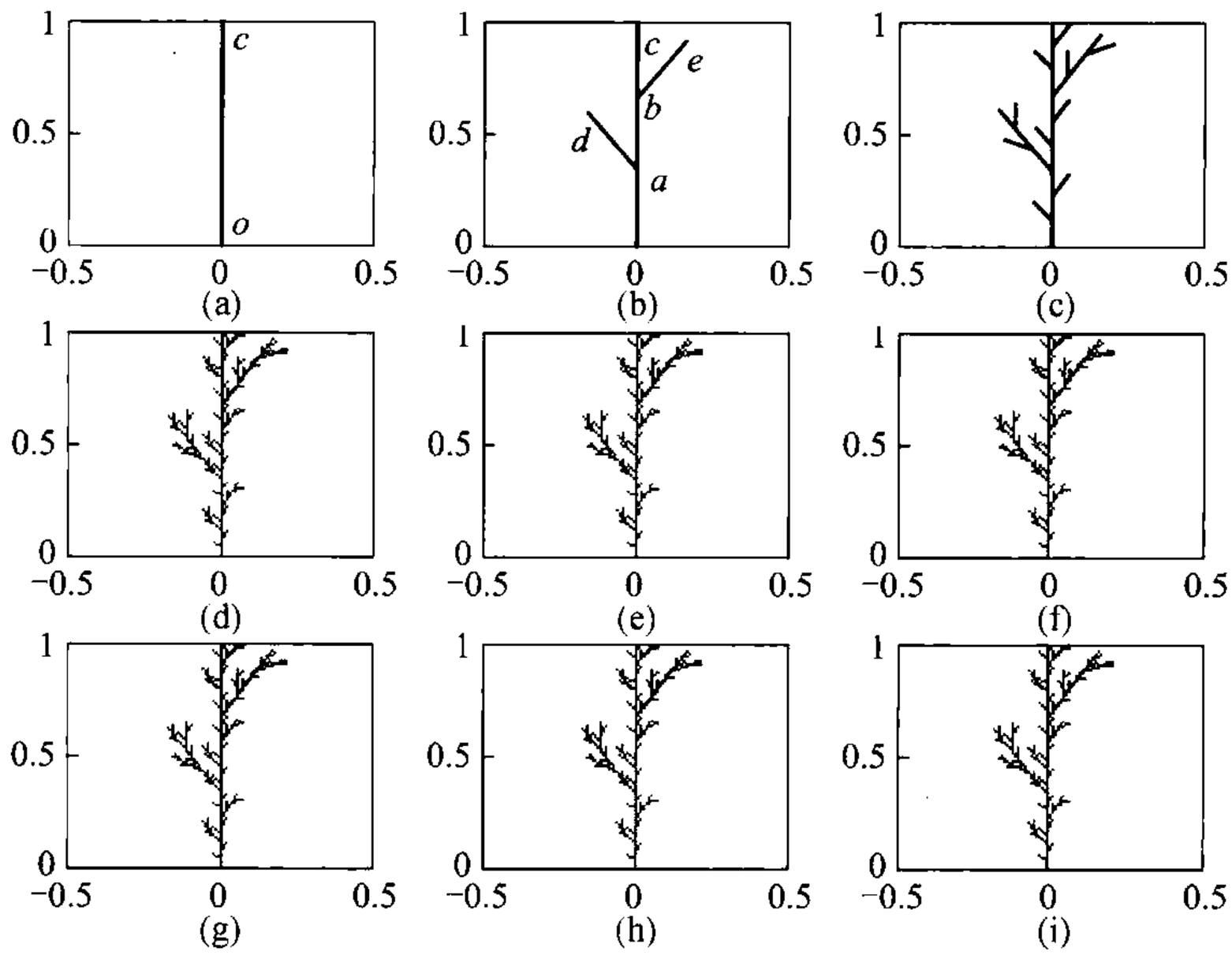


图 2.2 重复次数不同所画出的分形树

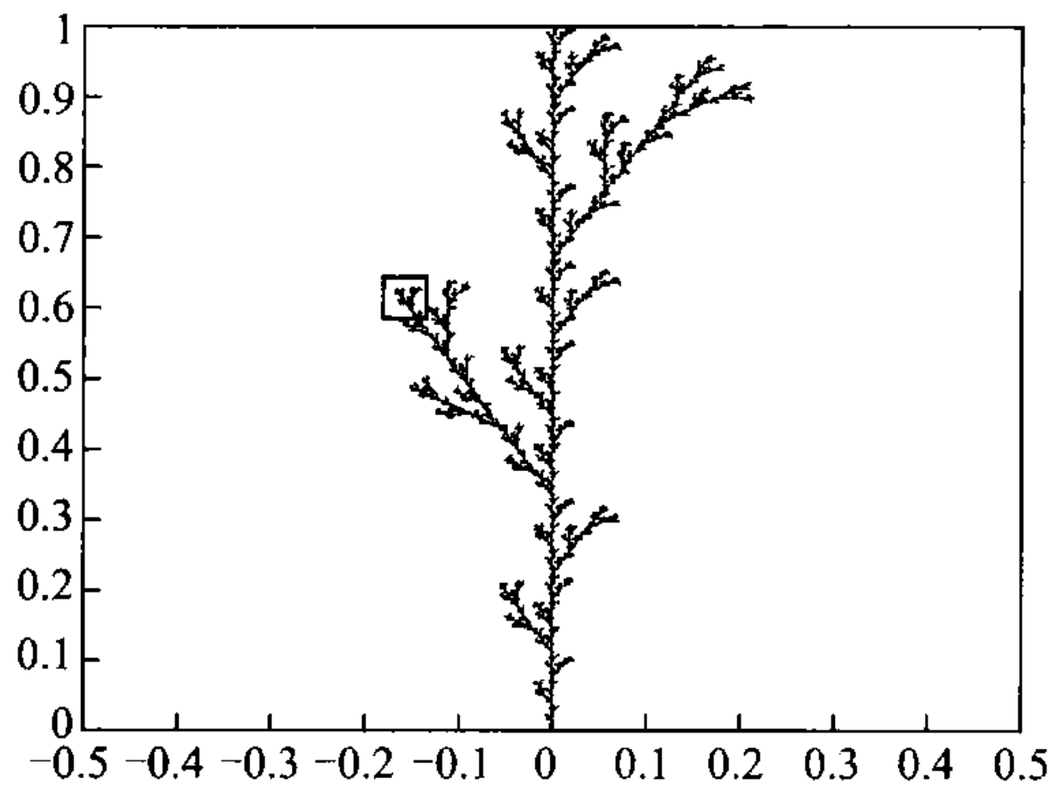


图 2.3 整体图形中小方框的位置

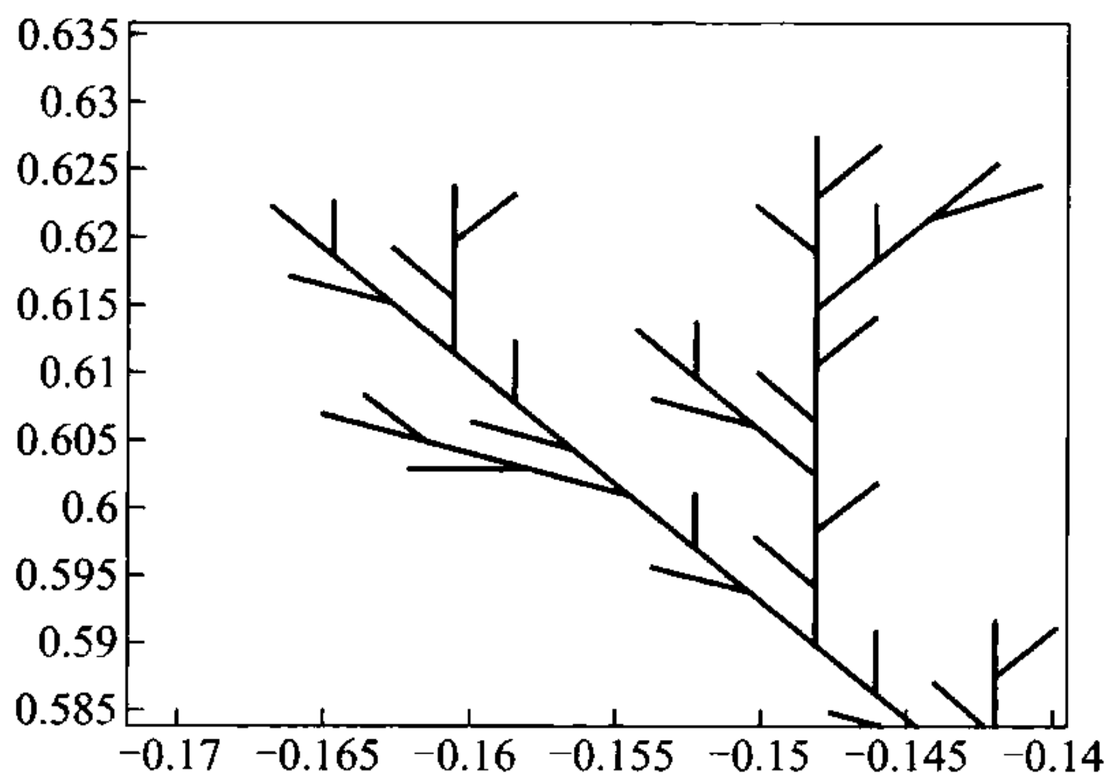


图 2.4 放大的小方框中的图形

下面有4种画分形树的程序。

1. 树程序1（运行时间120 s）

```
tic
u =[0, 0; 0, 1];                                  %线段oc端点的坐标
subplot(3,3,1),
plot(u(:,1),u(:,2)),                              %画oc
axis([-0.5,0.5,0,1])
theta=pi/6;                                       %夹角是下面用到参数
for n=1:8                                         %画分形树的循环次数
   uuu=[];
   for I=0:(length(u)/2-1)     %用上次分形树的线段画下一个分形树
       p1=(u(2*I+1,:)*2+u(2*I+2,:))/3 ;          % 线段的1/3处如a点
       p2=(u(2*I+1,:)+u(2*I+2,:)*2)/3;           % 线段的2/3处如b点
```

```
        lp=[cos(theta),-sin(theta);sin(theta),cos(theta)]*...
        [(u(2*I+2,1)-u(2*I+1,1));u(2*I+2,2)-u(2*I+1,2)]/3 ;
        lp=p1+lp';                                  % 向左转动的点如d点
        rp=[cos(theta),sin(theta);-sin(theta),cos(theta)]*...
        [(u(2*I+2,1)-u(2*I+1,1));u(2*I+2,2)-u(2*I+1,2)]/3;
        rp=p2+rp';                                  % 向右转动的点如e点
        uu=[u(2*I+1,:);p1;p1;lp;p1;p2;p2;rp;p2;u(2*I+2,:)];
                                                    %将点排序
          uuu=[uuu;uu];                             %将各线段所得数据排列起来
      end
      u=[uuu];
      subplot(3,3,n+1),plot(u(:,1),u(:,2)),                          %画新图
      axis([-0.5,0.5,0,1])
    end
  toc
```

学习一个新程序，应该先运行一下程序，看看最终显示的计算结果。然后按照前面介绍的调试程序的方法，设置几个中断点，分段或者逐句运行程序，看看每个语句的作用是什么。这样对程序的功能有个初步印象以后，再来仔细研究细节。比如对于这个程序，如果运行整个程序，就会画出题目中所示的图形。如果逐句运行，就会看到，先画出一条直线，再画出一个最简单的分形树，依次画下去，直到最后的图形出来。

下面进一步解释这个程序。本书的程序都相对简单，所以不画流程图。如果程序比较复杂，理解有困难，也可以自己画一个流程图帮助理解。

首先可以看到，程序的主要结构是两个for循环，第一个循环是画分形树的次数，第二个循环是对上一次画出的分形树中的各条线段作新的等分画图。

再看程序各个语句与指令的含义。程序中用到两个新的指令tic和toc，它们是用来记录程序运行的时间，tic是开始记录时间，toc是停止记录时间。

程序中画线段的指令是plot，其中plot的参数x,y分别对应各线段端点的x,y坐标。画一条线段必须有两个端点的坐标，端点坐标是运用解析几何的知识进行计算。

当已知线段的两个端点坐标为$(x_1,y_1),(x_2,y_2)$，那么在线段的1/3处的坐标是$(\frac{2x_1+x_2}{3},\frac{2y_1+y_2}{3})$，在线段2/3处的坐标是$(\frac{x_1+2x_2}{3},\frac{y_1+2y_2}{3})$。

坐标为(x,y)的点逆时针转动θ角以后的新坐标是

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

对于顺时针转动，要将公式中的θ换为$-\theta$。

参看图2.2可知，第一次是画线段oc，两个点的坐标是o(0,0)，c(0,1)，共有4个数据。第二次要画的线段是oa，ad，ab，be，bc，增加的新点是a，b，d，e，而o，c是原来的点。新点的坐标是a(0,1/3)，b(0,2/3)，即分别是oc的1/3和2/3处。而ad由ab向左转动$\pi/6$而得，be由bc向右转动$\pi/6$而得。

各次所计算出的线段端点的坐标依次存入二列的实数矩阵[x,y]内，供 plot 作图。为了使plot能连续作图，正确的排列端点的顺序是很关键的。各点排列次序为(o，a，a，d，a，b，b，e，b，c)，共10个点，这样能形成连续的如图2.5所示的一笔画。由于用plot作图，每两点都画出一条线段，实际画了 9 条线段，即 oa，aa，ad，da，ab，bb，be，eb，bc。而有用的只有5条，即oa，ad，ab，be，bc，在下一次画图时，将用到的线段也是这5条。多画的4条线段并不会影响图形的形状，因为 aa，bb 都是一个点，而 da 与 ad 重合，eb 与 be 重合。

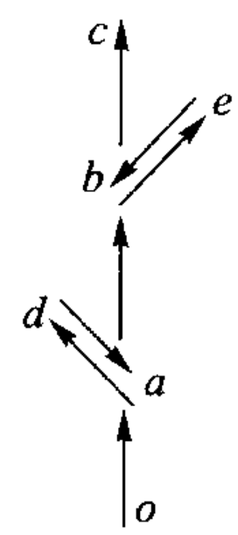


图 2.5 程序中作图中点的排列顺序

这样排列端点次序还有一个重要的用处是，下一次画图要用的线条数正好是这样排列出的端点数的一半，比如第三次画图要用5条线段，而排列出端点总数正好是10，所以程序中循环指标I的循环次数就是要用到线段数目，也就是端点总数的一半。取点时每两点取一条线段，比如10个点(o，a，a，d，a，b，b，e，b，c)中要取的线段是oa，ad，ab，be，bc共 5 条。这就是第二个循环中p1，p2 取点的方法。

由两个for循环运行一次形成的数据点数目按照如下所示规律增加

$$\begin{array}{lcccccc} x & *\cdots* & *\cdots* & *\cdots* & \cdots & *\cdots* & \cdots \\ y & \underbrace{*\cdots*}_{10} & \underbrace{*\cdots*}_{50} & \underbrace{*\cdots*}_{250} & \cdots & \underbrace{*\cdots*}_{5^m\cdot 2} & \cdots \end{array}$$

为了便于理解，将树程序1计算出的各点实数坐标列出如下，为了对比，将后面的树程序2中用复数计算的点的坐标也在此列出。

```
u =     x              y                    u =     x     +     iy
              0              0   %o点                0
              0        0.33333   %a点                0 +     0.33333i
              0        0.33333   %a点                0 +     0.33333i
        -0.16667       0.62201   %d点        -0.16667 +     0.62201i
              0        0.33333   %a点                0 +     0.33333i
              0        0.66667   %b点                0 +     0.66667i
              0        0.66667   %b点                0 +     0.66667i
        0.16667        0.95534   %e点        0.16667 +     0.95534i
              0        0.66667   %b点                0 +     0.66667i
              0              1   %c点                0 +           1i
```

这个编程过程体现了数值计算的基本特点，它包括了下面几个过程：

(1) 建立模型

这里就是题目所阐述的分形树的模型。

(2) 确定数值计算的方法

本题目中就是用解析几何的方法计算端点的坐标。程序1是用实数表示端点的坐标，后面的程序2是用复数表示端点的坐标，不同的方法会有不同的效率。在以后的内容还会学到一些数值计算方法，如数值微积分、数值解微分方程等。

(3) 编写程序

这一步就是要在计算机上实现数值计算的目的。比如在这个例子中计算端点坐标并不困难，但在计算机上要做到的是能自动地连续进行成千上万次的重复计算，就是要形成能自我复制的数据链，也就是要形成能连续进行推算的固定的逻辑关系，这就很复杂了。所以如何排列端点的次序，以保证循环的计算不会中断，所得数据又能让画图指令plot画出正确的图就成了编程的一个难点，在分析程序时主要讨论的就是这个问题。还有，第一个图画的是直线，它与以后的图形并不一致，还不属于分形图，所以程序中没有将它放到for循环中去，而是在开始循环计算之前，单独把它画出来，实际上这条线段的两个端点值是for循环的启动值。而以后for循环的运算则依赖每次循环结束后所得的所有线段端点值，也就是程序中矩阵u的值。

还要注意的是，变量名的选取最好与公式中所用的符号相同，或者取英文名的缩写，如这个程序中p1代表point1，p2代表point2，lp代表left point，rp代表right point。也可以取汉语拼音的缩写。对希腊字母可以用它们的英文注音，如这里用theta表示θ。总之变量取名应该好记好用，不宜太长。

在编程过程中，要随时备份，即将已经编好的部分另外存为一个文件，以防由于编程过程经常出现的失误，导致前功尽弃。每次修改已经编好的文件，最好也只用复制的文件进行修改，以保证原来的正确程序的完整性。这些虽然不是大问题，却是初学者常犯的毛病。

将上面的程序对比下面讲到的另外3个程序可以看到，虽然计算的目标相同，编程方法却大不相同。解决了数值计算方法不见得就是解决了编程方法，这是两种不同的技巧，学习时都应该加以重视，初学者有时不明白，已经学过数值计算，怎么还是不会编程？问题就在于此。其实就是学过计算机语言，也不等于就掌握了编程，这里还有一个熟练掌握与正确运用的问题。

(4) 分析计算结果

本题只是画图，所以结果不必再加以分析。但是上一章讲的Logistic模型，所用的程序比这里的程序更简单，但对结果的分析却很丰富，包括了周期分岔，混沌，以及Feigenbaum常数等内容。而用数值计算研究分析物理现象，主要的结论也是产生在这个步骤中。

2. 树程序2

这个程序思路基本与程序1相似，但是以复数表示点的坐标，则线段的长度可以用复数的加减来计算，而计算转动后的坐标可以运用复数z转动θ角所得新的复数是 $ze^{i\theta}$ 这一性质。对一个复数矩阵中的所有数据进行移动和转动，与计算其中一个元素的移动和转动是相同的。这些技巧已经在前面画椭圆时用过。程序2与程序1的作图方法相同，计算过程略快，程序更简洁，可读性更强，所以在计算平面上点的坐标时，复数表达是优先的选择，以后，本书的程序优先使用复数表示点的坐标。

```
u=[0,i]; subplot(3,3,1);  plot(u), axis([-0.5,0.5,0,1])
for n=1:7
        uuu=[];
        for I=0:length(u)/2-1
            diff=(u(2*I+2)-u(2*I+1))/3;
                                        %在第一次循环是oa的长度
            p1=u(2*I+1)+diff;
                                        %在第一次循环是计算a点坐标
            p2=p1+diff;
                                        %在第一次循环是计算b点坐标
            lp=p1+diff*(sqrt(3)/2+1/2*i);
                                        %在第一次循环是计算d点坐标
            rp=p2+diff*(sqrt(3)/2-1/2*i);
```

```
                                        %在第一次循环是计算e点坐标
            uu=[u(2*I+1);p1;p1;lp;p1; p2;p2;rp;p2;u(2*I+2)];
            uuu=[uuu;uu];
        end
        u=uuu;
        subplot(3,3,n+1);plot(u), axis([-0.5,0.5,0,1])
    end
```

3. 树程序3

仔细察看上述两程序会发现，在画新图时，有许多端点的坐标是上一次循环中已经计算过的，但在新的循环中又重新计算，这无疑是一种重复计算。前面讲过，避免重复计算是编好程序的基本要求，所以应该去掉这些重复计算。仔细研究以后可以发现，每当进行一次循环之后，表示线段的两个端点之间会插入8个数据点的坐标，例如在第一组图中，最初的线段是oc，循环以后画出第一个分形图，它有 o，a，b，c，d，e 共 6 个端点，同时数据也从2个扩充到10个。再进行一次循环后，又会在5条线段 oa，ab，bc，cd，de 中各插入4个端点，10个数据。这些新插入的点的坐标的计算方法都相同，数据排列的方法也相同。由此知道，只要按照这个规律，将新点插入旧点之中，就可以编出程序。程序中以old表示上一次循环所得的数据矩阵，以new表示本次循环的数据矩阵，按照上述规律，用10个数据去替换原来的2个数据，程序中用new 开头的10个语句就是完成这个工作。而p1,p2,lp,rp则是新插入的端点的数据。这样通过以矩阵元素赋值的方法计算新点坐标，减少一重循环，再次提高了运算速度。这也是矢量化编程的应用。

```
new=[0,i];  subplot(3,3,1);
plot(new);  axis([-0.5,0.5,0,1 ]);
for k=1:8
   old=new;
   n=length(old)/2-1;                          %上次生成的线段数目
   diff=(old(2:2:end)-old(1:2:end-1))/3;       %每条线段1/3的长度
   p1=old(1:2:end-1)+diff;
   p2=p1+diff;
   lp=p1+diff*(sqrt(3)/2+1/2*i);
   rp=p2+diff*(sqrt(3)/2-1/2*i);
     %以下的计算在第一次循环中计算10个端点o, a, a, d, a, b, b, e,
     %b, c的坐标
   new(1:10:10*n+1)=old(1:2:end-1);
```

```
                                        %各次循环分别计算第1,11,21,...个点
    new(2:10:10*n+2)=p1;
                                        %各次循环分别计算第2,12,22,...个点
    new(3:10:10*n+3)=p1;
                                        %各次循环分别计算第3,13,33,...个点
    new(4:10:10*n+4)=lp;
    new(5:10:10*n+5)=p1;
    new(6:10:10*n+6)=p2;
    new(7:10:10*n+7)=p2;
    new(8:10:10*n+8)=rp;
    new(9:10:10*n+9)=p2;
    new(10:10:10*(n+1))=old(2:2:end);                %第10,20,...点
    subplot(3,3,k+1),
    plot(new)
    axis([-0.5,0.5,0,1 ]);
end
```

4. 树程序4(运行时间0.21 s)

前面的程序根本的思路都是通过解析几何的方法计算所有的端点坐标，然后画图。下面的程序则利用了分形图形的自相似性，从而提供了一种新的编程思路。分形图形的整体与局部有一种相似性，各个局部也有相似性，在第一个分形图形 [图2.2 (b)] 中，如果把5个线段看成5个局部图形，则它们长度都相同，都是原线段的1/3，只是位置方向不同。换言之，将最初的线段[图2.2(a)]缩短到1/3，再摆放在5个不同的位置就是第一个分形图。按照这个思路，将第一个分形图，缩小到原来的1/3，再摆放到5个不同的位置就形成了第二个分形图形[图2.2(c)]，其他的分形图都可以按照这个方法生成，为了显示效果，图2.6表示的是第二次画图的过程。

编出来的程序如下：

```
u=[0,i];
subplot(3,3,1);
plot(u)
axis([-0.5,0.5,0,1])
for k=2:8
    m=u/3;                                          %将图形缩小到1/3
    uu=[m, ...                                      %放置缩小后的图形m
        i/3+m*(sqrt(3)*0.5+0.5i),...
```

```
                                        %将m在虚轴上向上移动1/3后向左旋转
    m+i/3,...                           %将m在虚轴上向上移动1/3
    2i/3+m*(sqrt(3)*0.5-0.5i),...
                                        %将m在虚轴上向上移动2/3后向右旋转
    m+2i/3];                            %将m在虚轴上向上移动2/3
subplot(3,3,k);
plot(uu)
axis([-0.5,0.5,0,1])
u=uu;
end
```

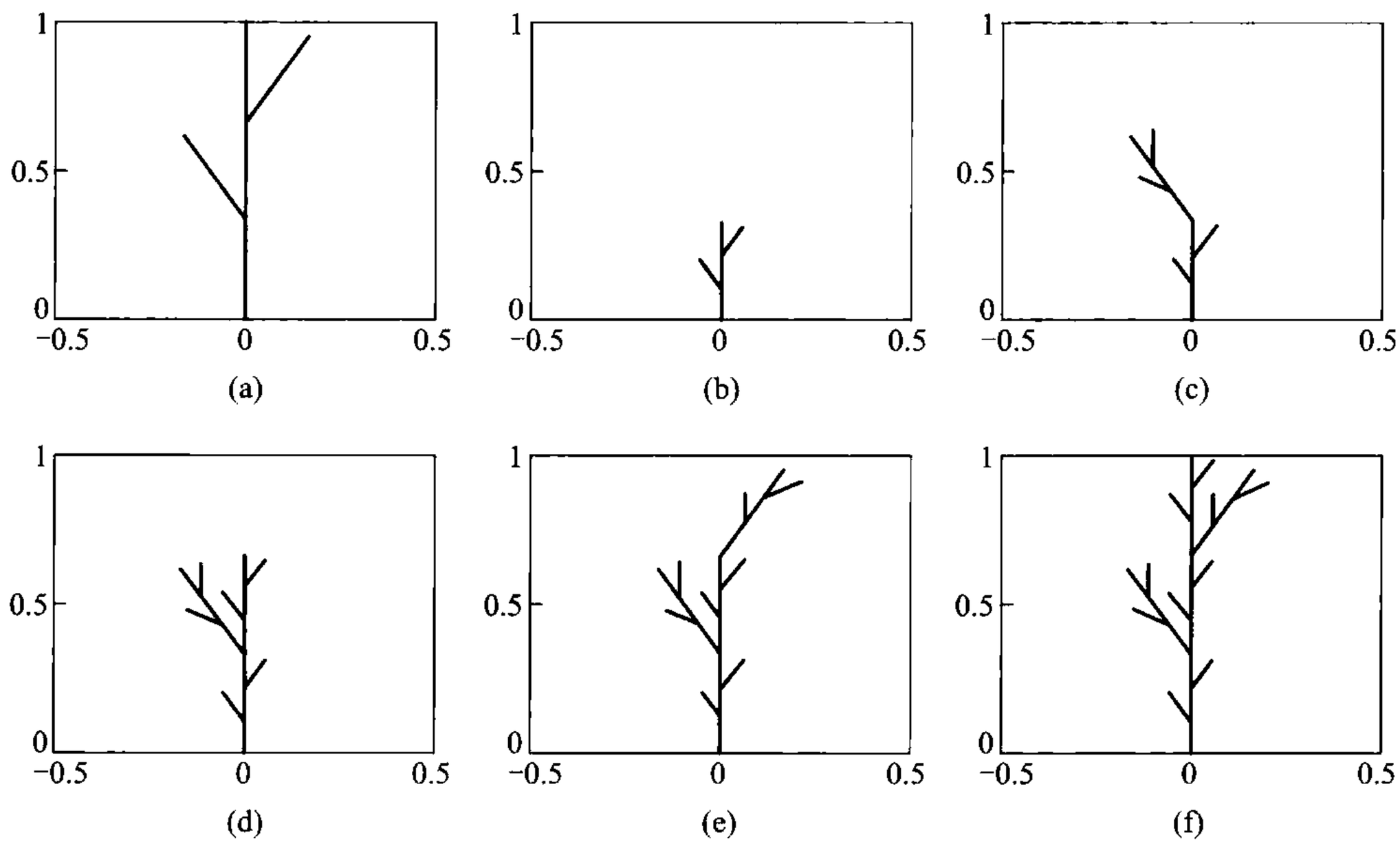


图 2.6 用自相似性画图的过程

这个程序的优点在于利用了分形图形的自相似性，这是物理思想在编程中的巧妙运用。尤其是复数的优点在此得到充分利用，因为图形的全部数据都是复数，所以复数的加减乘除都能利用。这样编出程序不仅优美好读，且运算速度大大提高，比第一个程序运算速度提高成百倍，试想如果要想通过提高CPU的速度达到同样目标，需要增加多少投资?

§2.1.2 Sierpinski三角形

如图2.7，连结等边三角形的三条边的中点，将三角形四等分，得到四个较小的三角形。舍去中间的小三角形，保留周围的三个，即有颜色的三个三角形[图2.7(b)]。

此后将这三个较小三角形按上述分割与舍去法则操作下去，得到一种介于线段与面之间的几何图形，或者说是一个被不断挖空的面积，它也是一种分形图形。

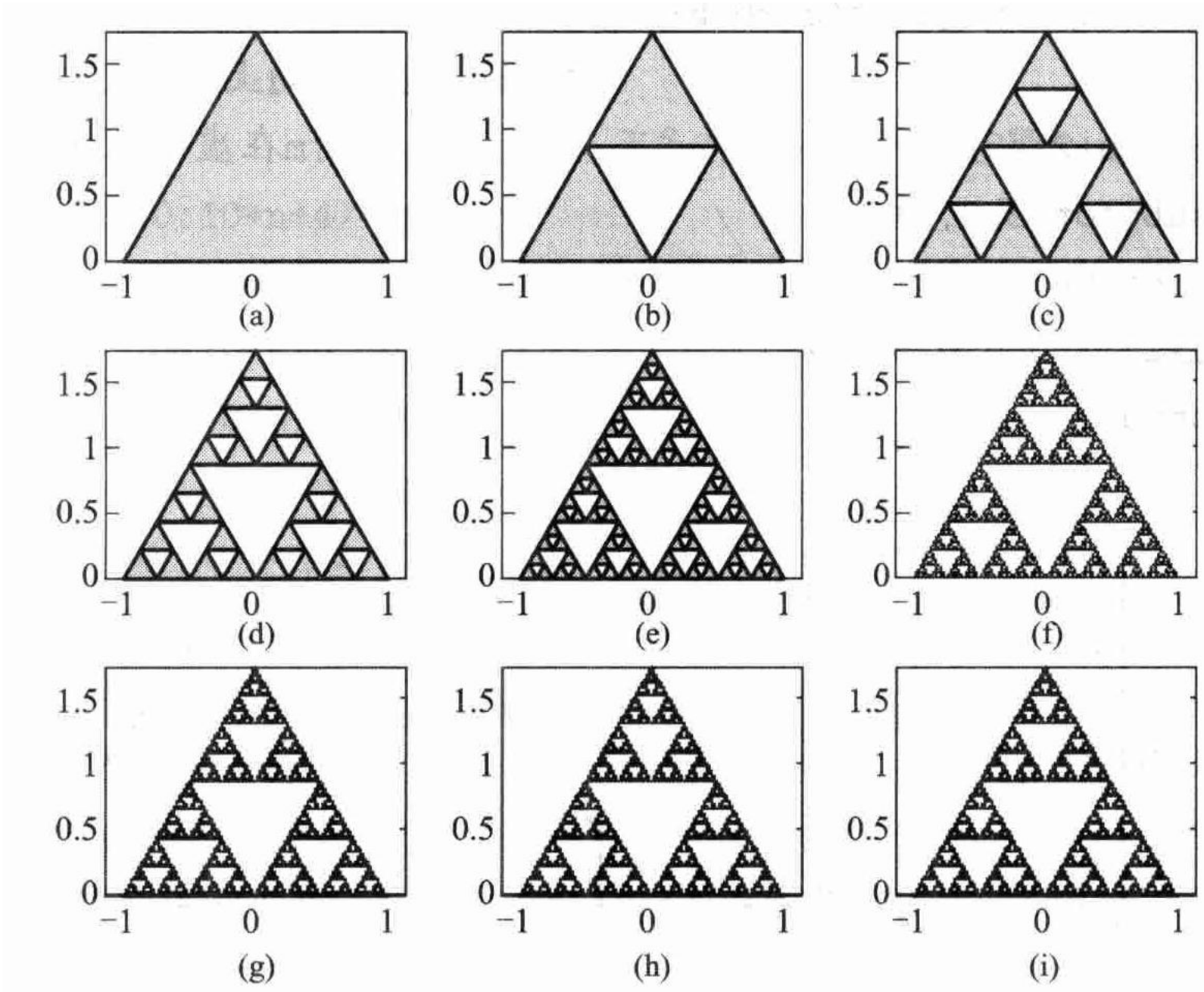


图 2.7 Sierpinski三角形

下面分析一下，如何求解这个问题。从图形可以看出，原始的三角形是最大的三角形，有三个顶点，第一次等分以后，保留3个小一些的三角形，即图2.7(b)中3个填了颜色的三角形，它们有9个顶点。第二次等分则会得到9个更小的三角形，27个顶点。如果作m次等分，将得到3^m个三角形，3^{m+1}个顶点。三角形数量的增加情况可以表示为

$$\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{*}_{1}\ \underbrace{***}_{3}}_{4}\ \underbrace{**\cdots**}_{9}}_{13}\ \underbrace{**\cdots**}_{27}}_{40}\ \cdots\ \underbrace{**\cdots**}_{3^m}}_{\frac{3^m-1}{2}}\ \cdots$$

再看这些顶点的数值，第一次等分后的9个顶点只有3个新的不同的取值，分别是原来3条边的中点。再加上原有的3个顶点的值，共有6个不同的值。第二次等分的27个顶点中，只有9个新的取值，分别是原来9条边的中点，再加上原有的6个值，共有15个不同的值。总之，新增加的不同的值总是原来的三角形的边的中点。

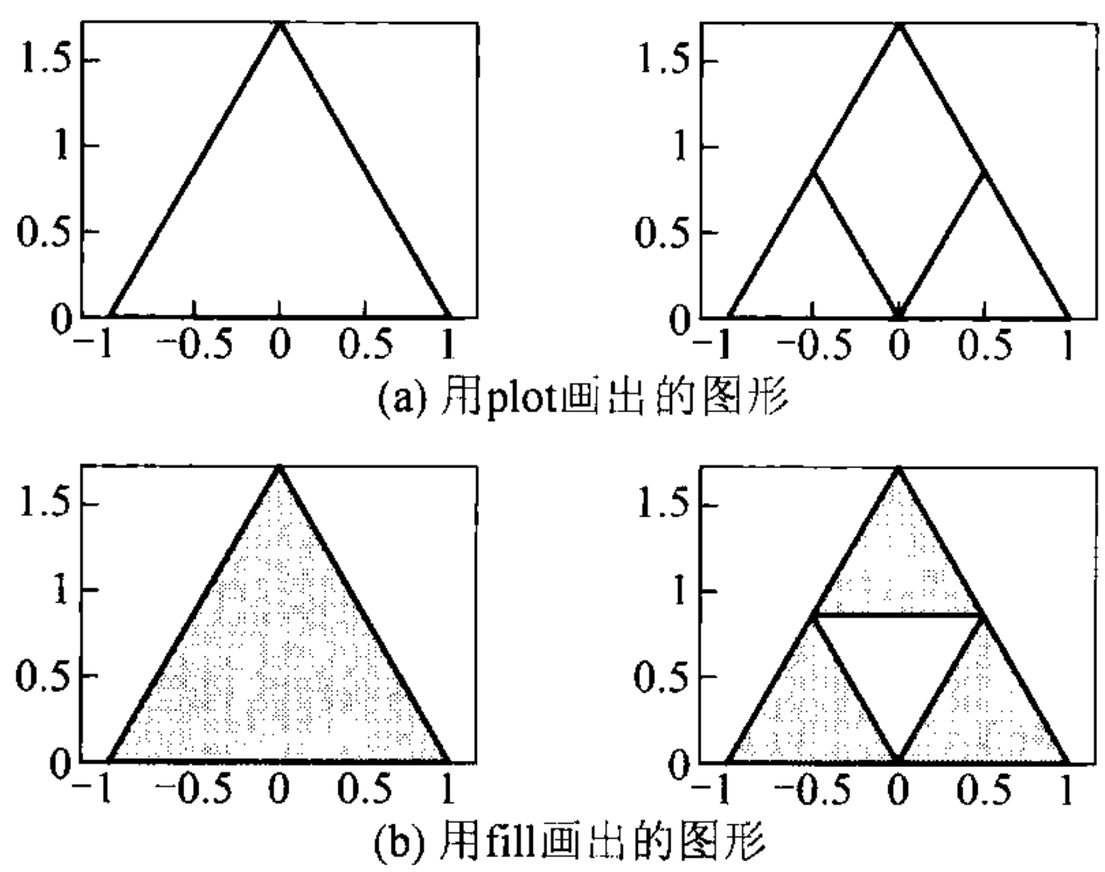
与树程序相仿，下面也有几个不同的程序来画这些图形。这说明，在以上程序中所采用编程方法在解决类似的问题中具有通用性。

1. 三角形程序1

这个程序与树程序2相似，用for循环语句作图，在给出原始三角形的顶点坐标后，再将三个中点的坐标加入进去，组成新的数据序列，然后用指令fill填色。计算中采用复数描述点的坐标。为了理解fill的填色效果，下面画出对同样的数据，用plot与fill两个指令所产生的效果，如图2.8所示。

下面的程序中，第一个循环表示作多少等分，第二个循环表示每次等分都要将已经存在的$\frac{3^m-1}{2}$个三角形各分成三个更小的三角形。最后只对最终生成的小三角形填色，它的数目是

$$\left(\frac{3^{m+1}-1}{2}\right)-\left(\frac{3^m-1}{2}+1\right)=3^m$$



(a) 用plot画出的图形

(b) 用fill画出的图形

图 2.8

```
for m=0:8                                   %最大的三角形等分的次数
     subplot(3,3,m+1)
     f=[-1;  sqrt(3)*i;   1];
                                  %用3×1的矩阵表示原始三角形的顶点坐标
old=f;
 for k=1:((3^m-1)/2);
                %每次循环都将上次循环的一个小三角形分为三个三角形
     new=[old(1), (old(1)+old(2))/2,(old(1)+old(3))/2;...
     (old(1)+old(2))/2, old(2),     (old(2)+old(3))/2;...
     (old(1)+old(3))/2, (old(2)+old(3))/2,    old(3)];
     f=[f,new] ;          %将新数据加入原矩阵后，矩阵增加3列
```

```
        old=f(:,k+1);%下一步取第k+1列元素(即第k+1个三角形)计算
    end
     ff=f(:,((3^m-1)/2+1):((3^(m+1)-1)/2)) ;
                                        %对最后生成的小三角形填色
     fill(real(ff), imag(ff),'g');
     axis equal;
end
```

2. 三角形程序2

这个程序与树程序3相似。三角形程序1也存在重复计算，应该想办法避免。下面的程序就是同时计算所有的新点的坐标，然后组成数据序列，再填色。指令numel是计算矩阵所有元素数目，对于3行的矩阵，元素数目除3就得到矩阵的列数。新矩阵将每个三角形等分为三个更小的三角形，也就是将每列扩充为3列，所以新矩阵的列数是3N。

```
new=[-1;  sqrt(3)*i;  1];
subplot(3,3,1);
fill(real(new),imag(new),'g');
axis equal;
for m=1:8
  old=new;                                  %调用上次循环所得的数据
  a=(old(1,:)+old(2,:))/2;                  %3个新的不同的数据
  b=(old(1,:)+old(3,:))/2;
  c=(old(2,:)+old(3,:))/2;
  N=numel(new)/3;                           % 计算原矩阵的列数
  new(1, 1:3:3*N)=old(1,:);                 %将新的数据插入形成新的矩阵
  new(1, 2:3:3*N+1)=a;
  new(1, 3:3:3*N+2)=b;
  new(2, 1:3:3*N)=a;
  new(2, 2:3:3*N+1)=old(2,:);
  new(2, 3:3:3*N+2)=c;
  new(3, 1:3:3*N)=b;
  new(3, 2:3:3*N+1)=c;
  new(3, 3:3:3*N+2)=old(3,:);
  subplot(3,3,m+1);                         %画图填色
  fill(real(new),imag(new),'g');
  axis equal;
```

```
end
```

3. 三角形程序3

这个程序与树程序4相似，利用了分形图形的自相似性。先将已有三角形图形的边长缩小到1/2，再移动后摆放两次，组成全部新图。这里要注意的是，不能将填好颜色的图形缩小，只能将描述三角形顶点的复数的数据矩阵进行缩小移动的运算。为了缩小移动后能正确填充颜色，必须正确地排列顶点数据的顺序。下面详细地分析程序中用的方法。

```
u=[0,1,0.5+0.5i*sqrt(3),0];                    %原始的三角形
subplot(3,3,1)
fill(real(u), imag(u),'g')
for k=1:8
    m=0.5*u;                                   %将数据缩小到1/2
    u=[m(1:end-k-1), ...                       %移动形成的3个三角形的数据
    m + 0.5,...                                %按画图要求排列
    m + 0.25+0.25*i*sqrt(3),...
    0];
    subplot(3,3,k+1)
    fill(real(u), imag(u),'g')
end
```

这个程序和三角形程序1的区别是fill的用法不同，程序中代表原始三角形的数据是4个，也就是如果用plot 画图，它画出的是三条边而不是像三角形程序1那样只画出两条边。所以这里 fill 是对一个封闭的图形填色，后面语句中的fill也是这种用法。那么缩小并移动后的数据是如何形成封闭的更小的三角形呢？看一下程序中k=1的情况就可以理解。取程序中的k=1，并将fill换成plot，画出的图形如图2.9所示。

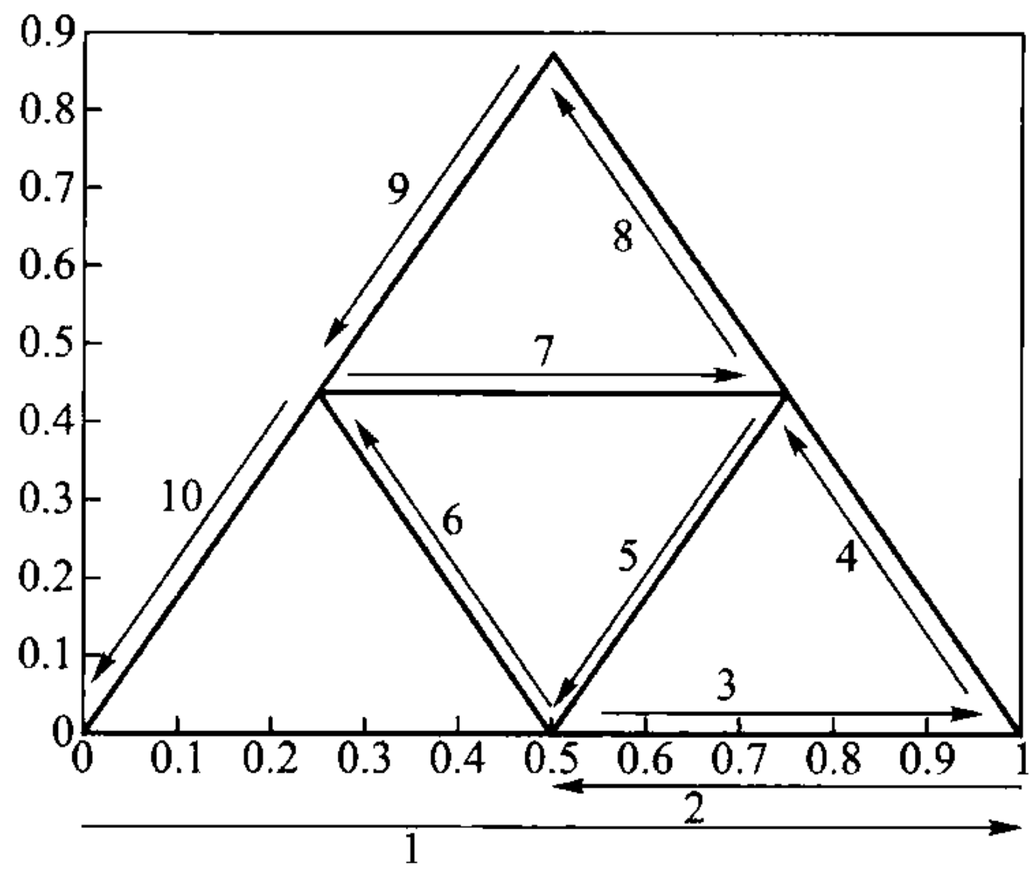


图 2.9 用自相似的性质画Sierpinski三角形

去掉中间的三角形，剩下的3个小三角形位置分别在左下，右下和上面。代表原始大三角形的矩阵是u，而矩阵m=0.5u，就是将大三角形缩小到1/2并且正好成为左下的小三角形。将左下小三角形右移0.5就是m+0.5，得到的是右下的小三角形。而上面的小三角形就是m + 0.25+0.25*i*sqrt(3)。虽然3个小三角形的顶点的数据都已经全部有了，如果直接将3个小三角形的数据连接起来，即令

```
u = [m, m+0.5,  m + 0.25+0.25*i*sqrt(3) ]
```

那么画出图形是错误的，如图2.10所示。

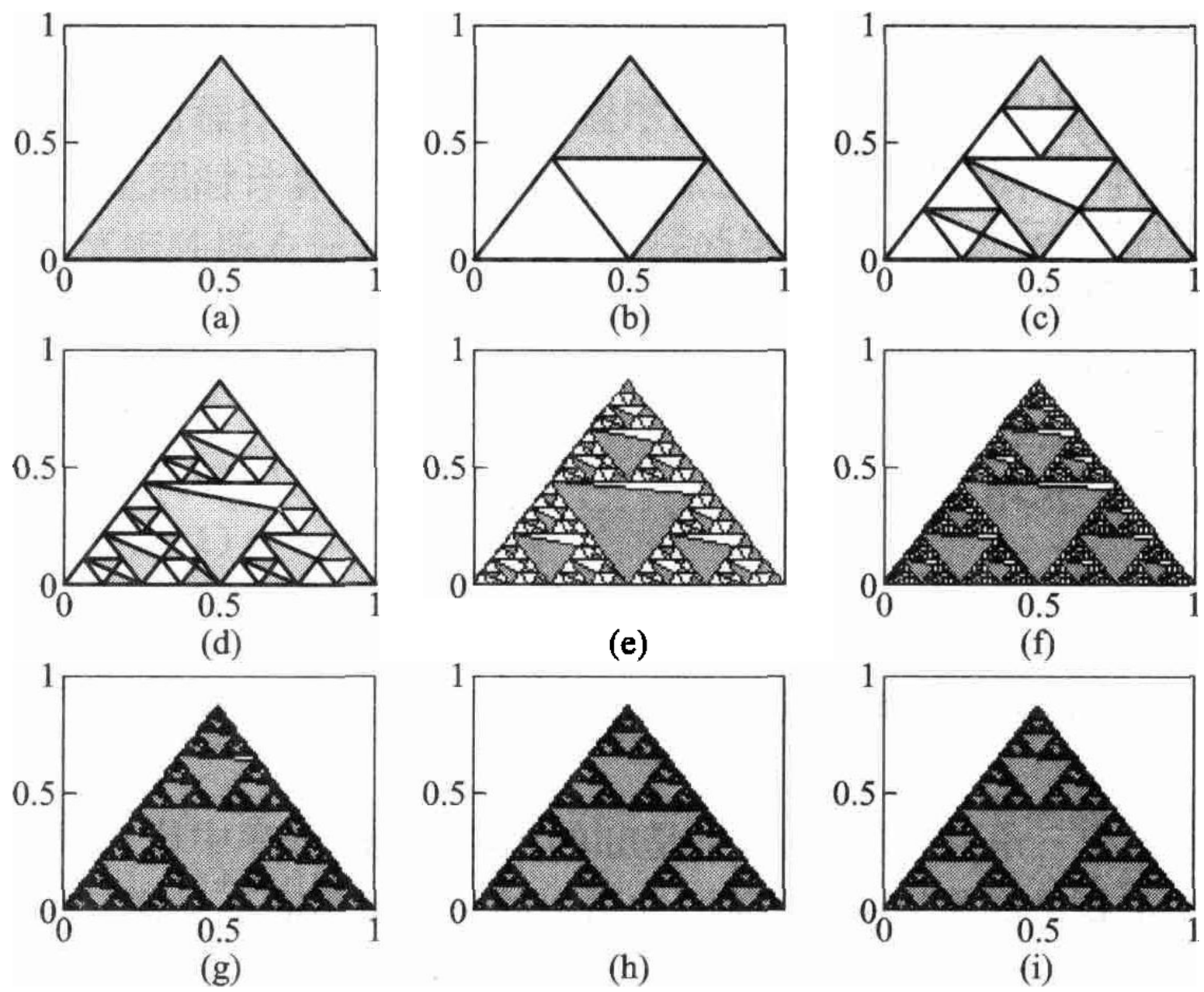


图 2.10 编程错误会画出错误的图形

而程序中的排列方法以k=1为例，所画的线段走向如图2.9所示，线段上标号代表画线的次序，这些线段正好组成了3个封闭的小三角形。所以指令fill能正确的填色。由于填色涉及fill的用法，数据排列的方法也不是唯一的，如果令

```
u = [m, m+0.5, 0, m + 0.25+0.25*i*sqrt(3), 0 ]
```

也能正确地填色，但是要将数据排列成与三角形程序1和2相同的顺序，则比较困难。如果有读者能做到，请告诉作者。

4. 三角形程序4

递归调用法也是编写迭代程序的一种技巧。递归调用就是程序调用自己，以此实现循环计算的功能。尽管计算n阶阶乘可以运用指令prod(1:n)，为了举例，下面编制一个示意性的递归程序来计算阶乘。

```
%program fact.m
function output=fact(n)
```

```
if n==1
    output=1;                                   %中止递归的条件
else
    output=n*fact(n-1);                         %递归调用的语句
end
```

调用时,输入fact(n)，例如计算5的阶乘就是fact(5)。

再用递归法画三角形的分形图形。编程的思路是，在一个大的三角形中，连接三条边的中点得到一个小三角形，将大的三角形填成绿色，将小三角形填成白色就完成一次分割。依次下去，可以得到需要的图形。为了说明这个思想，将下面程序中的第二个fill换成plot，也就是不把小三角形填成白色，那么就会在绿色的三角形内画出一个白边的小三角形。如图2.11所示。

这个程序思路独特，结构简单，易于理解，可读性强，缺点是运行速度较慢。调用程序时在指令窗口输入三角形三个顶点和迭代次数。程序用subplot设计成画6个图形，所以输入最多的迭代次数是N1=7。调用时在指令窗口输入如下内容：

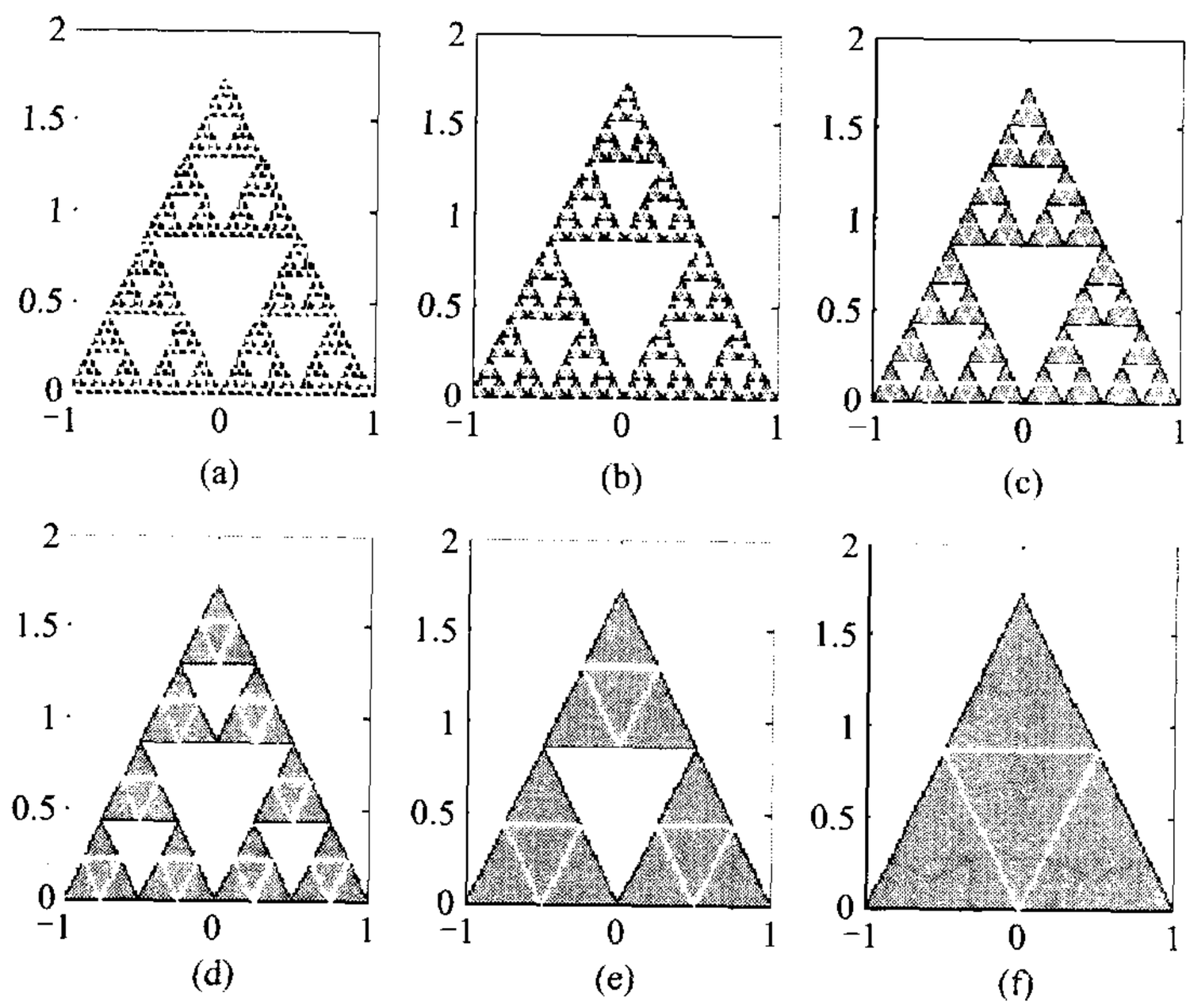


图 2.11 递归法所画的图形

```
N1=7;   A1=-1;   A2=sqrt(3)*i;   A3=1;     sjx(A1,A2,A3,N1)
```

实际程序如下：

```
function sjx(A1,A2,A3,N1)                          %函数文件的固定格式
triangle1=[A1,A2,A3,A1];                            %原始的大三角形
B1=(A1+A2)/2;  B2=(A1+A3)/2; B3=(A2+A3)/2;         %三角形的3个中点
```

```
triangle2=[B1,B2,B3,B1];                    %由3个中点连接的三角形
N2=N1-1;                                    %完成了一次迭代
if N2>0;                                    %继续新的迭代
  sjx(A1,B1,B2,N2);
                                            %递归调用后每次迭代又变成3个新的迭代
  sjx(B1,A2,B3,N2);
  sjx(B2,B3,A3,N2);
  subplot(2,3,N2)
  fill([real(triangle1)],[imag(triangle1)],'g');
                                            %将大三角形填成绿色
  hold on
  fill([real(triangle2)],[imag(triangle2)],'w')
                                            %将中间的小三角形填成白色
end
```

§2.1.3 科赫雪花曲线

将等边三角形的每条边三等分，将中间的一段去掉，代之以一个更小的等边三角形的两条边，依次做下去，将得到雪花分形图2.12，称为科赫(Koch)雪花曲线。

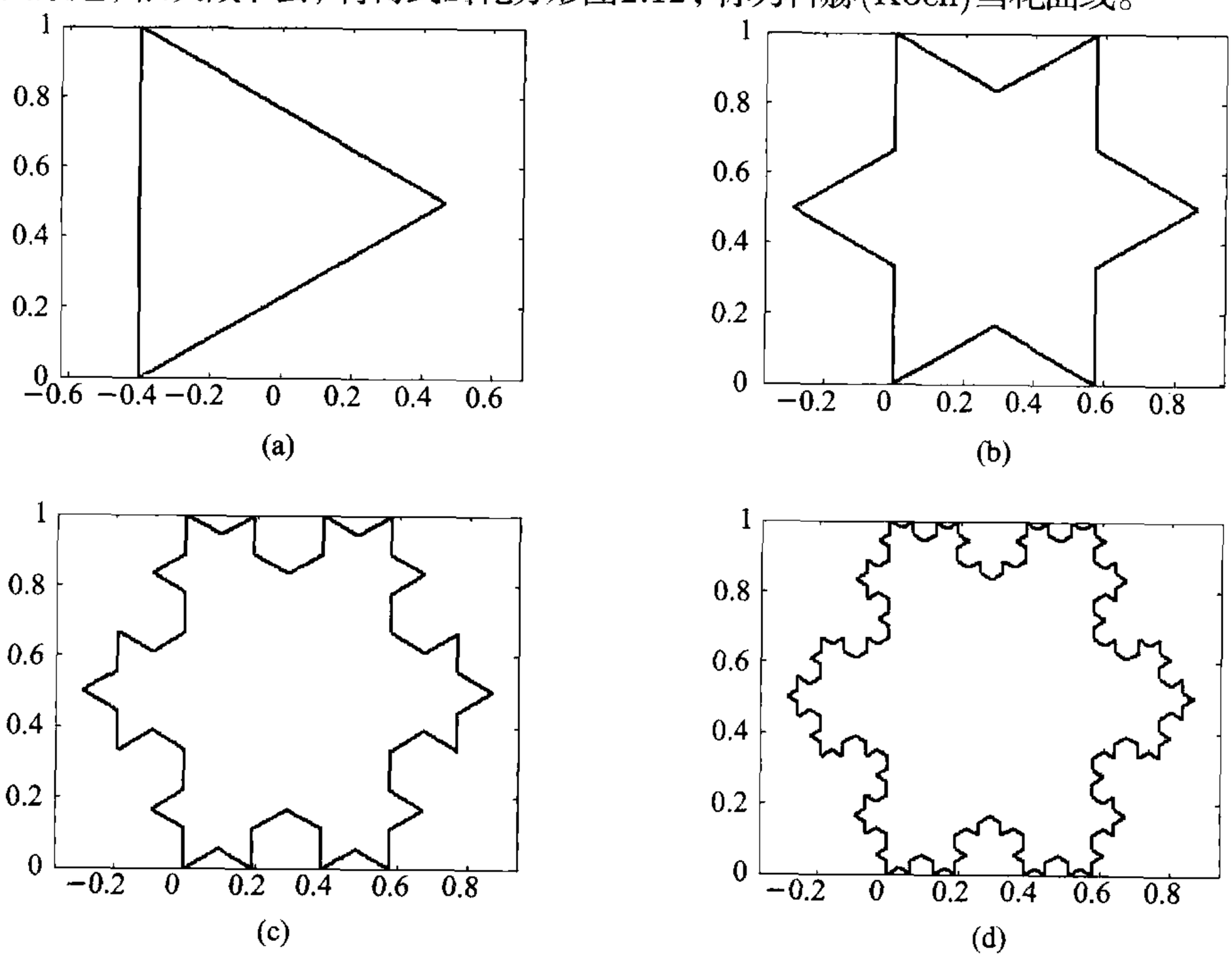


图 2.12 科赫雪花曲线

这里的程序与前面的两个例子有很多相似之处，尽管编程思路一致，但是也有差别。实际上，编程的技巧是讲不完的，不同的问题可能总有自己独特的技巧，多学习一些例子有助于启发思路，开阔眼界。这个例子再次说明了前面的编程方法的通用性。

1. 雪花程序1

这个图形的特点是所有线段首尾相接，本来一条线段要计算两个端点的坐标，在这里只需要计算起始点的坐标就可以。比如开始的三角形，三条边只有三个顶点，但要画出封闭的三角形，必须将起始点的坐标作为终点加入到所有数据最后。所以得到3+1=4个数据。经过第一次等分和重新画图以后，一条边成了四条线段，同时增加了三个端点。这三个端点分别位于原线段的1/3处，2/3处，以及将2/3处的点顺时针转动π/6 所得到的点。三条边得到12个数据，这时再把起始点坐标加入到最后作为终点坐标，才能画出封闭的图形，所以数据总数是13。依次下去，得到的数据数目序列是

$$
\begin{aligned}
4 &= 3+1 \\
13 &= 3\cdot 4+1 \\
49 &= 3\cdot 4\cdot 4+1 \\
&\cdots\cdots \\
&3\cdot 4^m+1
\end{aligned}
$$

每次算出新数据后，都是用它替代旧的数据画图，这点与前面的程序也有不同。

这些点的坐标的计算都和前面算过的相同，数据排列也简单，都是顺序排列成一个行向量，所以编出的程序比较简单，也是使用两个for循环结构。

```
new=[0,i,sqrt(3)/2+i/2,0];        %初始的等边三角形，画图用4个数据
subplot(3,3,1);
plot(new),
axis equal
for kkk=1:8;                                          %画图的次数
   old=new;
   n=length(old)-1;                                   %计算线段的数目
   for jj=0:n-1;                                      %对每条线段进行计算
      diff=(old(jj+2)-old(jj+1))/3;                   %线段1/3的长度
      new(4*jj+1)=old(jj+1);                          %起点坐标
      new(4*jj+2)=old(jj+1)+diff;                     %线段的1/3处
      new(4*jj+3)=new(4*jj+2)+diff*((1+sqrt(3)*i)/2);
```

```
                                                    %转动后的点的坐标
    new(4*jj+4)=old(jj+1)+2*diff;                   %线段的2/3处
end
new(4*n+1)=0;                                       %终点就是起点坐标
subplot(3,3,kkk+1),
plot(new),
axis equal;
end
```

2. 雪花程序2

去掉在雪花程序1中重复的计算，采用矢量化编程，所有的边上新点坐标同时计算,减少一重for循环，提高了运算速度。

```
new=[0,i,sqrt(3)/2+i/2,0];
subplot(3,3,1);
plot(new),
axis equal;
for k=1:8;
    old=new;
    n=length(old)-1;
    diff = (old(2:end) - old(1:end-1)) / 3;
    new(1:4:4*n-3) = old(1:end-1);
    new(2:4:4*n-2) = old(1:end-1) + diff;
    new(3:4:4*n-1) = new(2:4:4*n-2) + diff*(0.5+0.5*sqrt(3)*i);
    new(4:4:4*n) = old(1:end-1)+ diff + diff;
    new(4*n+1)=old(n+1);
    subplot(3,3,k+1);
    plot(new),
    axis equal;
end
```

3. 雪花程序3

在这个程序中虽然也是利用分形图形的自相似性画图，但是方法有较大差别。粗看起来，似乎利用自相似性画图不太可能。因为新图形并不是整个旧图形的缩小与移动，但是细看一下，不难发现，每个图形都可以看成是其中的一部分图形的复制与移动。那么就可以按照这样的思路来画图：画初始的等边三角形时先画出一条边，将它移动摆放在新位置就得到另外两条边。在画第二个图形时，也要先画好一条边。方法是将这

条边缩短到原来的1/3，然后按需要移动摆放三次，就形成了一条新的边。再将这条边移动摆放两次，就得到另外两条新边。这样全部图形还是利用自相似的特性完成了作图。

```
a=[0,i];                                    %画初始三角形的一条边a
b=a*(-0.5 - 0.86603i)+i;                    %利用第一条的移动来画第二条边b
c=(a-i)*(-0.5 + 0.86603i);                  %利用第一条边的移动画第三条边c
subplot(3,3,1),
plot([a,b,c]);
axis equal
for k=1:8                                                          %画8个图
  subplot(3,3,k+1)
  a1=a/3;
                                %将一条边缩短到1/3，并用它画新图第一条边a
  a=[a1,a1*(0.5 + 0.86603i)+i/3,(a1-i/3)*
  (0.5 - 0.86603i)+i*2/3,...a1+2*i/3];
  b=a*(-0.5 - 0.86603i)+i;                                  %画第二条边b
  c=(a-i)*(-0.5 + 0.86603i);                                %画第三条边c
plot([a,b,c])
axis equal
end
```

§2.1.4 L系统

前面的几种分形图形的画法有一个很明显的特点就是，它们都是由某个基本图形（生成元）反复迭代而生成的。例如分形树的生成元是一条线段，生成的方法是在线段的1/3和2/3处增加两段向不同方向转动的线段。雪花曲线的生成元也是一条线段，生成的方法是在1/3处增加两段折起的线段。而三角形分形图形的生成元则是一个三角形，生成方法是从中挖去由三条边的中点连成的三角形。这种作图规则就是所谓的林氏系统(通常称为L系统)，是用计算机模拟分形图形的最成功的系统之一。

L系统是Lindenmayer在1968年为模拟生物形态而设计的，后来Smith于 1984年、Prusinkiewicz 于1986年，分别将它应用于计算机图形学，引起生物学界和计算机界人士的极大兴趣，一时发表了许多论文和专著，在国内也有介绍L系统的著作和应用L系统的论文。

L系统实际上是字符串重写系统。所谓字符串指的是曲线，或者更准确地说是基本图形，重写则是按照一定的法则生成整个图形。L系统的精髓就是给出一

个起点（种子）和替换规则，连续利用该规则就能生成新一代系统。在专门介绍L系统的书中，是用一些专门的字符来表示种子和法则的，这样图形的生成过程就能表示成连续的字符式。

L系统的主要应用有两个方面：一是用这种方法生成那些规则比较明确的经典的分形图形，如前面的三角形分形图；二是用它可以模拟植物形态，特别是那些具有分枝结构的植物图形，如前面的分形树。

§2.2 复变函数迭代

前面介绍的具有自相似特征的分形图形都是实数迭代的例子，虽然在计算技巧上借助了复数进行运算以提高计算速度，但是它们并不是复数迭代的例子。下面再介绍用复数迭代生成分形图形的例子。这些例子不仅是复函数的运算的例子，而且是矢量化编程的优秀例子。

考虑复数迭代

$$Z_{k+1} = Z_k^m + C \qquad (k = 0, 1, 2, \cdots)$$

当$m = 2$时，如果令$C = C_0$(C_0为常数)进行迭代，所得的Z集合称之为Julia集。如果取$Z = Z_0$（Z_0为常数），对不同的C进行迭代，所得的Z集合称之为Mandelbrot集。当$m > 2$时，迭代的结果称之为广义Julia集和广义Mandelbrot集。

利用Julia集和Mandelbrot集可以画出许多极美丽的图形，它们不仅成为分形物理研究的对象，也是计算机图形学和计算机艺术的探讨对象。还出现了所谓的分形艺术，分形图形被用于制造邮卡、纺织面料的图案、服装设计和一些装饰用的工艺品等，甚至著名的门户网站google曾经有一段时间用分形图形做网站的图标，如图2.13所示。

图 2.13 网站google用过的图标

§2.2.1 Julia集

在迭代式中取$m = 2, C = 0$，则有$Z_{k+1} = Z_k^2$，显然$|Z_{k+1}| = |Z_k|^2$，那么有三种情况：

(1) 当$|Z_k| < 1$时，经过多次迭代，Z将趋于零且是稳定的；

(2) 当$|Z_k| > 1$时，经过多次迭代，Z将趋于无穷且是稳定的；

(3) 当$|Z_k| = 1$时，经过多次迭代，Z仍在单位圆上。

因此，复平面可以分成两个区域，一个是由轨迹趋于零的点和趋于无穷的点所组成的稳定集，另一个是单位圆上的点组成的集合，叫Julia集。

当$C \neq 0$时，为了划分Julia集，可以规定一个正的常数M和迭代次数N，如果Z进行N次迭代后，仍然有$|Z| \leqslant M$，则认为该点属于Julia集，对于趋于内部的点也是如此处理。如果经过若干次迭代后，出现$|Z| > M$，则认为该点已经逃离，并根据达到逃离时需要的迭代次数对起始点进行分类，用不同的颜色来表示它们，就会得到非常美丽的图案。

编程时，先在复平面上某个范围内按照一定的间隔取好所有点，得到由所有起始点组成的矩阵Z，再定义迭代式子中的常数C（在程序中是取$C = i$，读者可以自取需要的值），然后对全部Z点同时进行迭代，这样就实现了矢量化编程，同时用矩阵B来记录各初始点逃离时需要的迭代次数。最后用指令imagesc将B表示成颜色图形。指令imagesc的功能是用颜色表示标量矩阵。指令colormap则是规定颜色与数值的对应关系。增加取Z点的密度可以得到更精细的图形。

```
%画 Julia  set
C=i;                                  %迭代式中的常数，读者可以改变
V = linspace(-2.0,2.0,400);           %决定复数的范围及取点的数目
[Re,Im] = meshgrid(V);
Z = Re + i * Im;                      %复平面上所有起始点
B = 0;
for K = 1:100                         %迭代100次
    B = B+(abs(Z)<=2);                %当Z的绝对值大于2时,认为已经逃离
    Z = Z.*Z + C;
end;
imagesc(B);                           %表示成颜色图形
colormap(jet);
axis equal
```

下面是4幅有代表性的图形。

(1) $C = 0$, 复平面分为两个区域，边界是单位圆。如图2.14。

(2) $C = 0.3 + 0.5\mathrm{i}$，复平面分成许多大小不一的区域，边界内就是Julia集，这些区域的边界是很复杂的曲线，但这些大小不同的区域却很相似，将小区域放大一定倍数以后，就会变得和大区域形状一样。如图2.15。

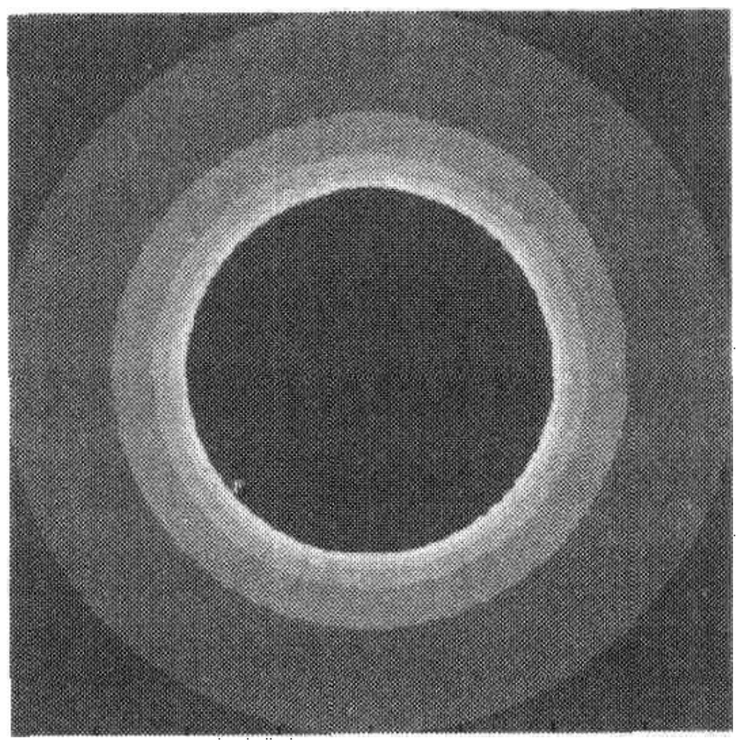


图 2.14 C=0

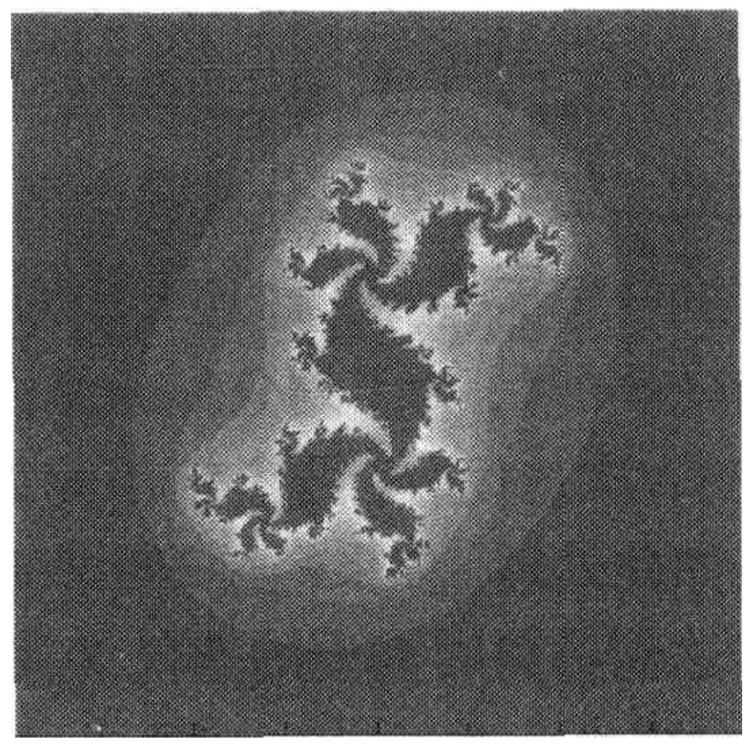


图 2.15 C=0.3+0.5i

(3) C =i，这时的Julia集是曲折的多技权的处处连通但处处不可微的曲线，图中没有闭合的环。如图2.16。

(4) $C = 0.11 + 0.66\mathrm{i}$，这时图形像一些细小的树叶，其实它是由一些点所组成。如图2.17。

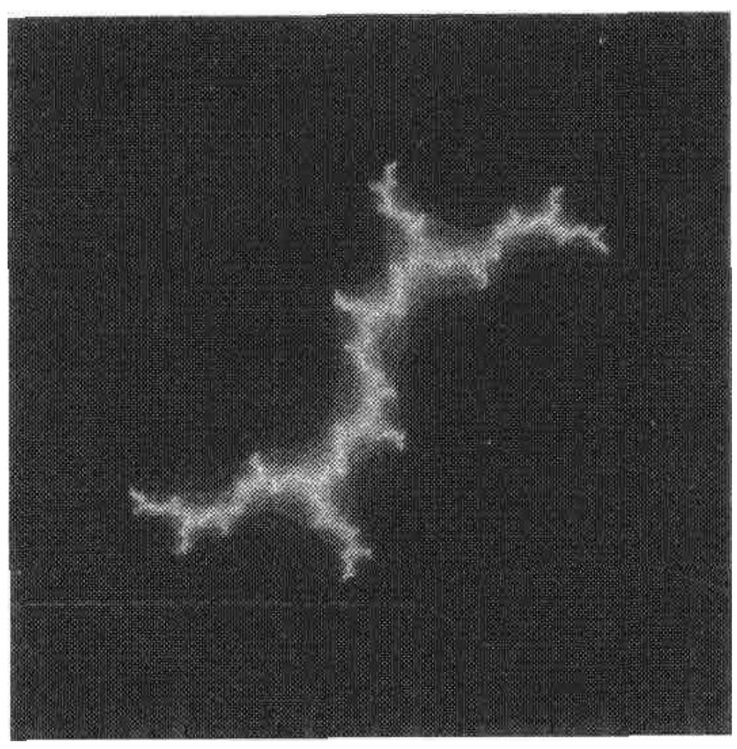


图 2.16 C=i

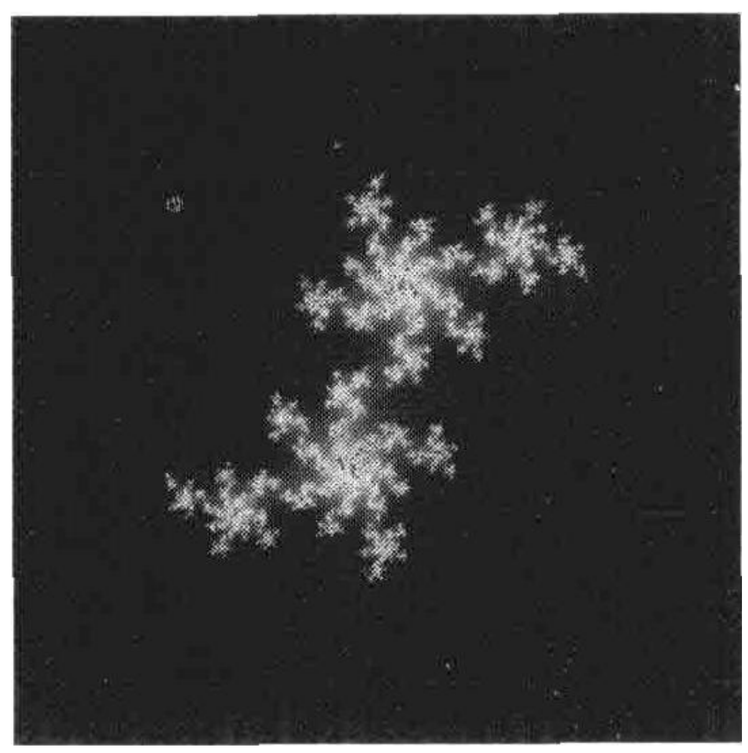


图 2.17 C=0.11+0.66i

这些例子表明，不同的C值可以得到不同的Julia集，C可以取无穷多个值，所以这种非线性的二次复函数的迭代可以得到无穷多个Julia集。

如果取$m = 2, 3, 4, 5$，得到的是广义的Julia集，图2.18至图2.21是几幅这样的图形，其中的迭代式分别是

$$Z = Z^2 + (0.32 + 0.5\mathrm{i})$$

$$Z = Z^3 + (0.2 + 1.05\mathrm{i})$$

$$Z = Z^4 + (0.4015 + 0.99\mathrm{i})$$

$$Z = Z^5 + (0.188 + 0.78603\mathrm{i})$$

这些图形除了有自相似的特征之外，还有一个特征是具有多叶的结构，如$m = 2,3,4,5$时，它们分别呈现两叶、三叶、四叶和五叶的结构。

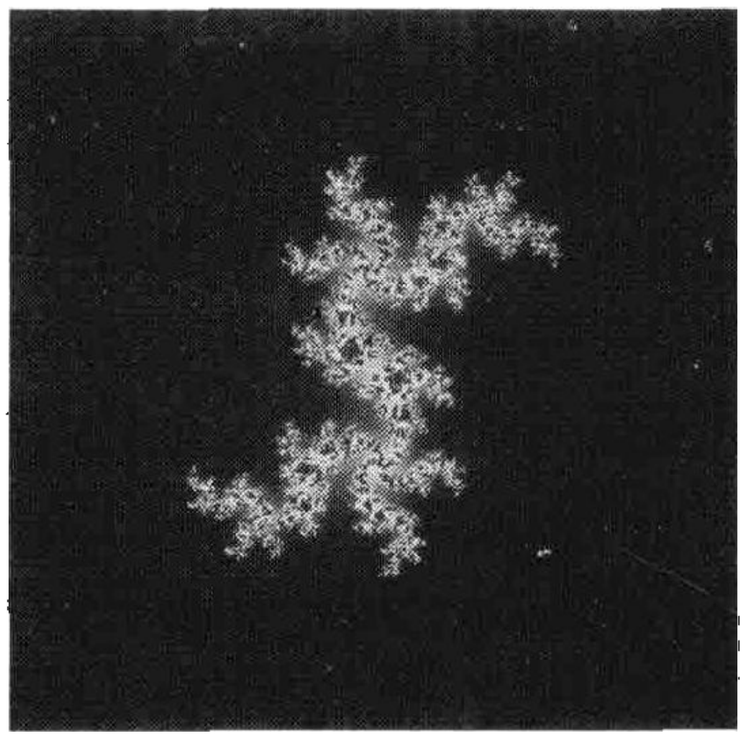


图 2.18　$Z = Z^2+(0.32+0.5i)$

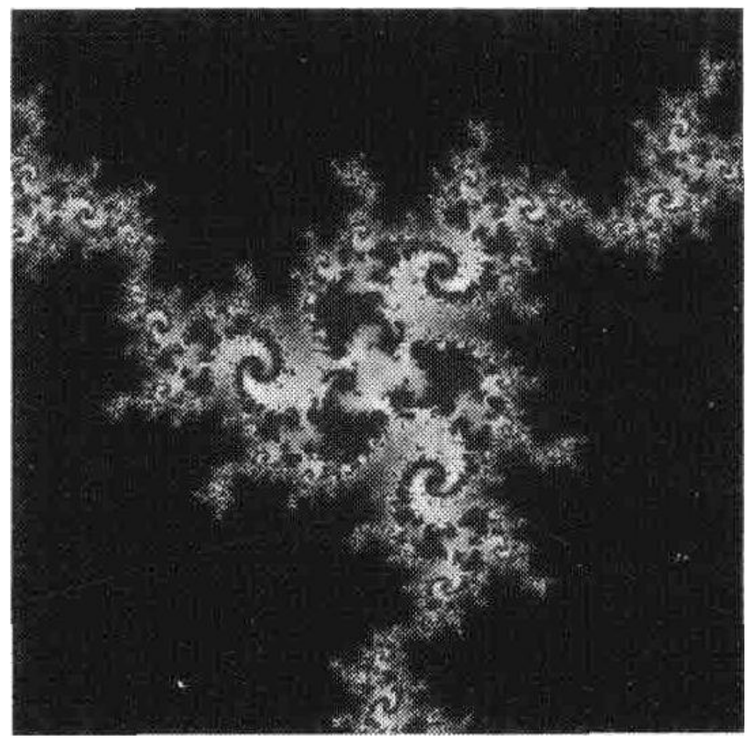


图 2.19　$Z = Z^3+(0.2+1.05i)$

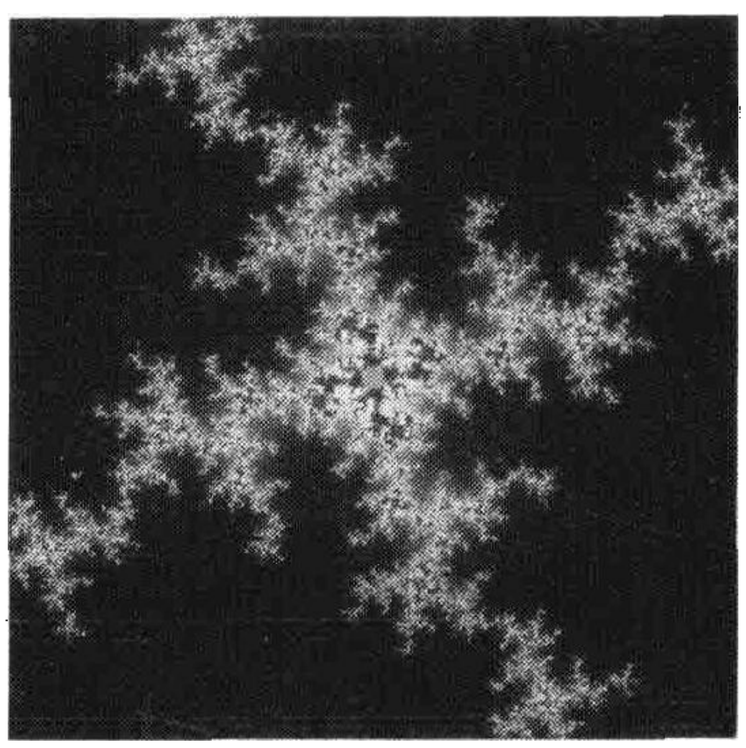


图 2.20　$Z = Z^4+(0.4015+0.99i)$

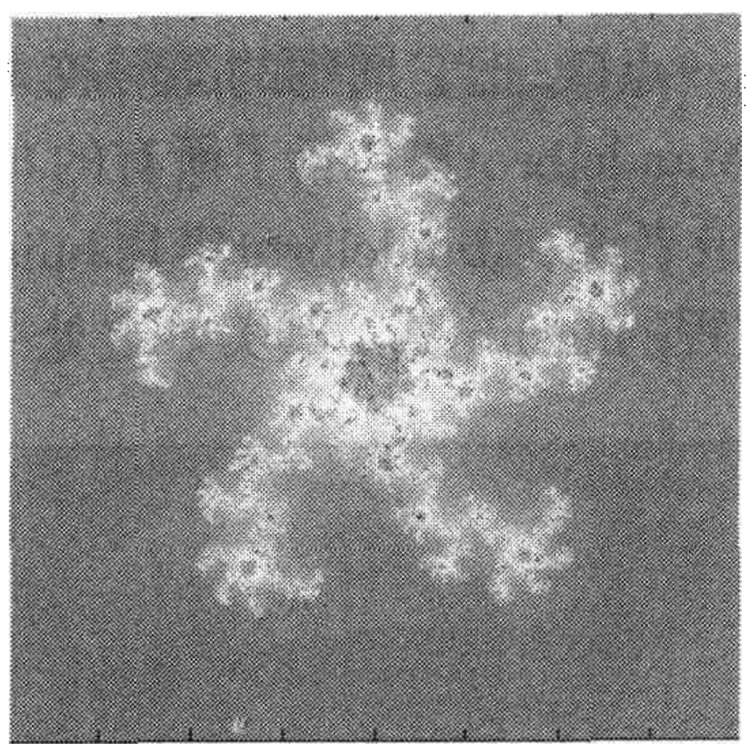


图 2.21　$Z = Z^5+(0.188+0.78603i)$

作为一种有趣的计算机实验，建议读者自己改变程序中的参数，画出一些自己的独特的分形图形。例如只要将程序中Z^2换成Z^3, Z^4, Z^5, Z^6就可以画出相应的广义Julia集。运行这些程序很容易，但是要找到合适的参数，画出一幅好图形，却要花费大量的时间与精力，例如在上面列出的画Julia集的图形，有的参数要到小数后面五六位，稍微大意就可能错过。复杂的问题可能对计算精度的要求还要更高。在实际问题中，为了得到一个有用的数据，可能要经过成百上千次计算甚至更多，而绝对不会是将书本上的程序输入计算机，在几秒钟以后就有了结果。这就是“计算机实验”或“计算机模拟”的实际含义。从这一点来说，初学者体会一下这种过程，对培养用计算机研究问题的耐心与细心是很必要的。读者还可以“分形艺术”作为搜索词，在互联网上搜索一些分形图形，一定会发现大量的美妙奇特、前所未见的图形，让你有不可思议的感觉，这对了解分形图

形也很有价值的。

§2.2.2 Mandelbrot集

在迭代式中，固定Z的起始点而改变C的取值进行迭代计算，得到的就是Mandelbrot 集。也要规定一个正的常数M和一定的迭代次数N，如果经过N次迭代，仍然有$|Z| \leqslant M$，则这个C值就属于Mandelbrot集。然后再根据达到逃离时所经过的迭代次数的不同对Mandelbrot集中的C加以分类并标示成不同的颜色就会得到很漂亮的分形图形。如图 2.22 是一张整体的图形，图 2.23 是局部的放大图。

粗看起来，图形很像乌龟的头与身子，细看又会发现，在乌龟的头与身子周边又长出了许多更小的像乌龟的图形，在放大的图形中看得更清楚。而且小的乌龟图形与大的乌龟图形非常相似。如果计算的结果足够精细，那么不断地放大小的图形可以发现，它们也有更细微的结构。事实上，将Mandelbrot集放大百万倍又会找到一个新的与原来的Mandelbrot集相似的图形。如果在Mandelbrot集中任选一点C，把它周围的图形放大，就会找到与这个C 值对应的Julia集。也就是说，Mandelbrot集包括了所有的Julia集，它是无穷多个Julia集的直观的图像目录表，或者说，Mandelbrot集是Julia集的缩微字典。作者曾经在网上搜寻到展现这个过程的flash电影，其美妙之处令人如入仙境。

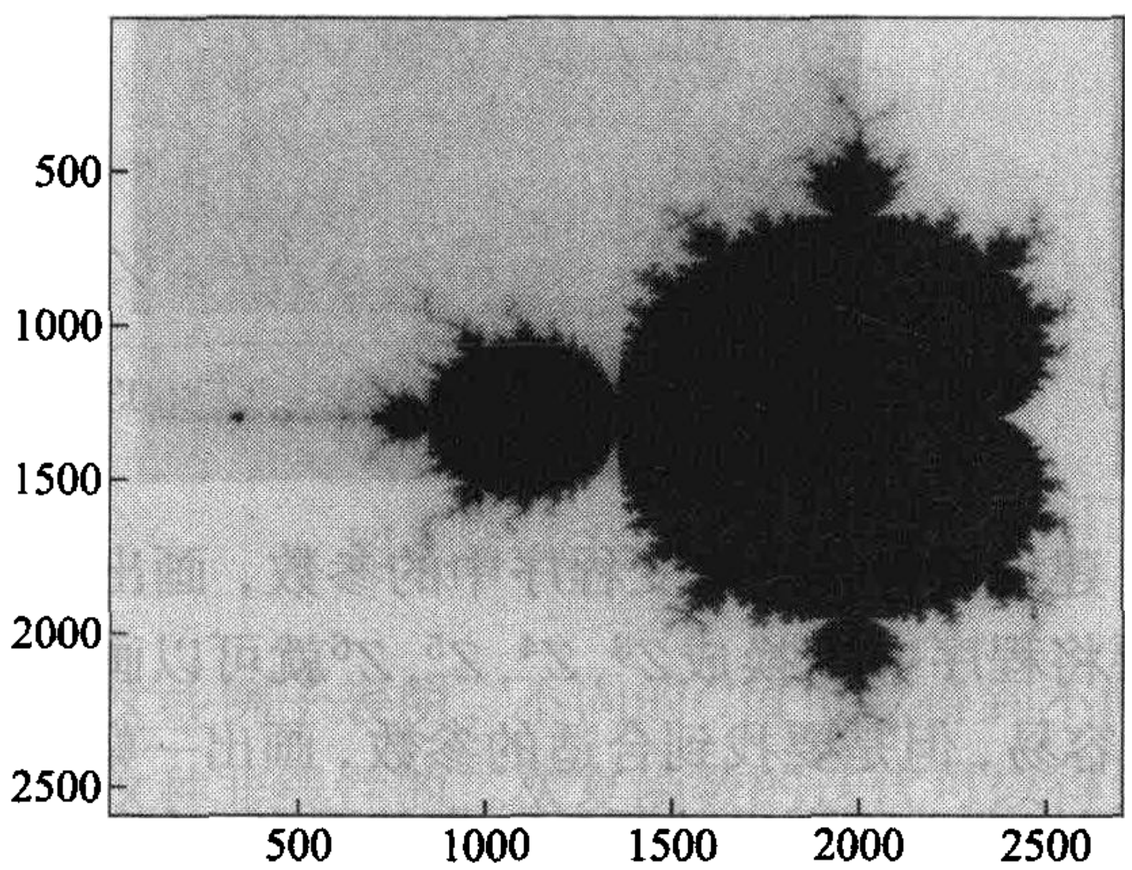


图 2.22 Mandelbrot集的图形

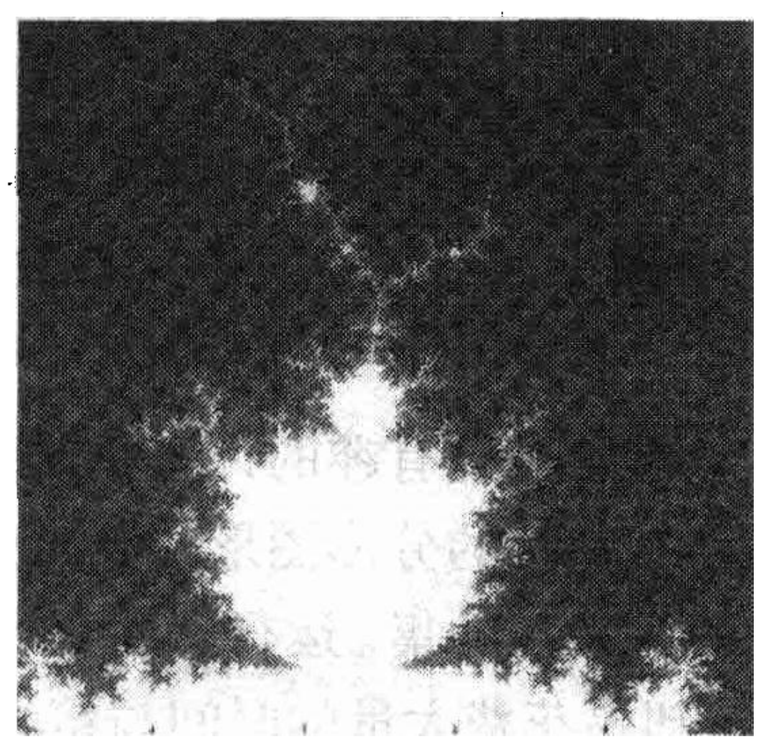


图 2.23 局部放大图

下面是画图的程序，这个程序与Julia集的程序非常相似，不同之处是复平面不是代表Z而是代表C，其余部分基本相同。

```
%画 Mandelbrot  set
x = linspace(-2.0,1.0,400);                    %x的范围与取点数目
y = linspace(-1.5,1.5,400);                    %y的范围与取点数目
```

```
[Re,Im] = meshgrid(x,y);                    % 定义常数矩阵C值
C = Re + i * Im;
B = 0;                                      % 矩阵B记录达到逃离时的迭代次数
Z = 0;                                      % Z的初值固定为0
for l = 1:50                                %迭代50次
    Z = Z.*Z + C;
    B = B + (abs(Z)<=2);                    %当Z的绝对值大于2时,认为已经逃离
end;
imagesc(B);                                 %绘图设置
colormap(jet);
axis equal
```

由于这个图形色彩层次不是很丰富，灰度图形可能比彩色图显示效果更好，可以将彩色图换成灰度图，方法是将语句colormap(jet)换成colormap(gray(256))即可。如果想看到更精细的图形，可以增加C的取点密度，比如将程序中取400个点换成1000个点。还有一个办法可以看到更高精度的图形，就是缩小计算的范围，例如将语句

```
x = linspace(-2.0,1.0,500);
y = linspace(-1.5,1.5,500);
```

换成下列语句

```
x = linspace(-0.4, 0.2,  500);
y = linspace(-1.2, -0.6, 500);
```

得到的就是上面的局部放大图2.23。

Mandelbrot集与Julia集并非只是一个计算机的作图游戏，事实上它们是分形理论研究的一个重要方向，下面摘录一段网上的文章供读者参考。

“动力系统中另一类分形集来源于复平面上解析映射的迭代。G. Julia 和 P. Fatou 于1918—1919年开创这一研究。他们发现，解析映射的迭代把复平面划分成两部分，一部分为Fatou集，另一部分为Julia集（J集）。他们在处理这一问题时还没有计算机，完全依赖于他们自身固有的想象力，因此他们的智力成就受到局限。随后50年间，这方面的研究没有得到什么进展。随着可用机算机来做实验，这一研究课题才又获得生机。1980年，B. Mandelbrot用计算机绘出用他名字命名的Mandelbrot集（M集）的第一张图来。1982年A.Douady构造了含参二次复映射f_c，其Julia集J（f_c）随参数C的变化呈现各种各样的分形图像，著名的有道迪兔子、圣马科吸引子等。同年，D. Ruelle得到J集与映射系数的关系，解决了解析映射集豪斯道夫维数的计算问题。L. Garnett得到J（f_c）集豪斯道夫维数的数值解法。1983年，M. Widom进一步推广了部分结果。Fatou 1926 年开始

整函数迭代的研究。1981年M. Misiuterwicz证明指数映射的J集为复平面，解决了Fatou提出的问题，引起研究者极大兴趣。随后发现超越整函数的J集与有理映射J的性质差异，1984年R. L. Devanney证明指数映射E_λ的J（E_λ）集是康托束或复平面，而J（f_c）是康托尘或连通集。

复平面上使J（f_c）成为连通集的点C组成M集即Mandelbrot集，H.Jurgens和H-O. Peitgen认为，M集的性质过去一直是并且将来继续是数学研究的一个巨大难题。通过将数学理论与计算机图形学实验加以融合，及Douady、H.Hubbard等人在这方面进行的基础性研究工作，在解决这一难题方面已取得重大进展，使人们加深了对M集的了解。Douady和Hubbard 1982 年证明M集是连通的和单连通的，人们猜测M集是局部连通的，目前每一张计算机图形都证实了这一猜测，但至今还没有人能给予证明。M是否为弧连通，目前尚不清楚。M集边界的维数也是值得研究的问题之一。

M集除了将J集分成连通与非连通的两类之外，还起着无穷个J 集的图解目录表作用，即把M集C点周围的图形放大就是与C点有关的J集的组成部分。但这一发现的数学严密性至今仍未确定，谭磊1985年证明了在每一个密休威茨点邻近M集与相关的J集之间存在着相似性。Jurgens等在M集的静电位研究中获得与自然形貌相似的分形图像。目前包括Jurgens等在内的很多研究人员都致力于借助计算机活动录像探索M集。其他一些分形集的研究工作正在取得进展。1990年Devanney通过数值实验观察到M集的复杂图形由许多不同周期的周期轨道的稳定区域共同构成。1991年黄永念运用他提出的代数分析法证明了这一事实，研究了M集及其广义情况周期轨道整体解析特性。

B. M. Barnsley 和 S. Demko 1985 年引入迭代函数系统，J集及其他很多分形集都是某些迭代函数的吸引集，用其他方法产生的分形集也可用迭代函数系逼近。1988年，H. A. Lauwerier 通过数值研究发现毕达哥拉斯树花是一迭代函数系的J集。1985 年 Barnsley 等研究含参数的函数系迭代动力系统，得到 M 集 D 并 D 与 M 在连通性上的差异。在一线性映射系迭代下，可以产生著名的分形曲线——双生龙曲线。1986年M. Mitzutani等对其动力系统进行了研究。”

§2.3 分形与分形维

前面介绍的几个图形都是分形图形，那么什么是分形呢?

分形(fractal)一词是Mandelbrot创造出来的。据他自己说，fractal 一词是 1975 年夏天的一个寂静夜晚，他在冥思苦想之余偶翻他儿子的拉丁文字典时，突然想到的。此词源于拉丁文形容词fractus，对应的拉丁文动词是frangere（“破碎”、“产生无规碎片”）。此外与英文的fraction（“碎片”、“分数”）及fragment（“碎片”）具

有相同的词根。在20世纪70年代中期以前，Mandelbrot一直使用英文 fractional 一词来表示他的分形思想。因此，取拉丁词之头，撷英文之尾的fractal，本意是不规则的、破碎的、分数的。Mandelbrot是想用此词来描述自然界中传统欧几里得几何学所不能描述的一大类复杂无规则的几何对象。例如，弯弯曲曲的海岸线、起伏不平的山脉，粗糙不堪的断面等。它们的特点是，整体上看极不规则或极不光滑，但是整体与局部又存在某种相似。Mandelbrot把这些部分与整体以某种方式相似的形体称为分形(fractal)。1975年，他创立了分形几何学(fractalgeometry)。在此基础上，形成了研究分形性质及其应用的科学，称为分形理论(fractaltheory)。

Mandelbrot的代表性的工作有：

1967年，在美国《科学》杂志上发表“英国的海岸线有多长”提出了分形的概念。

1973年，在法兰西学院讲课时，首次提出了分维和分形几何的设想。

1975年，用法文出版了分形几何的第一部著作《分形：形状、机遇和维数》。

1977年该书再次用英文出版。它集中了1975年以前Mandelbrot关于分形几何的主要思想，它将分形定义为豪斯道夫维数严格大于其拓扑维数的集合，总结了根据自相似性计算实验维数的方法。

1982年，出版《自然界的分形几何》，将分形定义为局部以某种方式与整体相似的集，重新讨论盒维数。

2004年，出版《分形与混沌：Mandelbrot集与其他》。这是一本论文集，收集了直到2003年的许多重要研究论文，其中有些是原始的未发表过的论文。

分形理论创新之处在于，基于传统欧几里得几何学的各门自然科学总是把研究对象想象成一个个规则的形体，而人类“熟悉”却无法描述的自然界许许多多真实的图形竟如此不规则和支离破碎，与欧几里得几何图形相比，拥有完全不同层次的复杂性。现代科学研究面对起伏蜿蜒的山脉、坑坑洼洼的地面、曲曲折折的海岸线、层层分叉的树枝、支流纵横的水系、翻腾变幻的浮云、地壳上的褶皱、密布人体周身的血管、满天闪烁的繁星、撕裂夜空的闪电、魔鬼般跳跃的火焰、船尾湍急的涡流、拍岸的惊涛与浪花、金属和非金属材料的断面、生物的大分子结构、分子光谱分布以及电磁波噪声分布等，急切要求得到精确和深入的解释。在这个传统欧几里得几何学无能为力的领域，分形理论脱颖而出，并得到迅速地发展，成了非线性科学研究中十分活跃的前沿和重要分支，还被升华成一种方法论与认识论。其启示是多方面的：一是分形整体与局部形态的相似，启发人们通过认识部分来认识整体，从有限中认识无限；二是分形揭示了介于整体与部分、有序与无序、复杂与简单之间的新形态、新秩序；三是分形从一特定层面揭示了世界普遍联系和统一的图景。

如前所述，分形图像具有自相似性并且可以迭代生成。分形理论认为，这两

条是在自然界普遍存在的。这种自相似性不仅表现在几何形态上，还可以在表现在结构、功能、信息、时间上。例如具有分形结构的物理系统有：动力系统的相空间（或子集）结构，一些系统的能谱，真实物理系统（浸透、晶体生长）。

在分形理论中，是用分形维数（简称分维），来定量描述分形结构自相似程度、不规则程度或破碎程度。

Mandelbrot这样解释引入分维的原因：

“为使分形几何有意义，我们不得不寻找一种方法，从数量的观点来表达形状的复杂性，就像欧几里得几何引用角度、长度、面积、曲率，以及用一维、二维、三维这些概念一样。

对于复杂的几何形体，普通维数的概念可能随尺度不同而改变。例如，直径10厘米的球用1毫米粗的细线做成。从远处看，球是一点。离10厘米远，线球是三维的。在10毫米处，它是一维线团。在1毫米处，每根线变成了圆柱体，整体又一次变成一维，如此等等，维数‘交叉’反复从一个值到另一个值。当球用有限数目像原子那么小的微物代表时，它变成零维。

对于分形，和普通维数（0，1，2，3）相对应的维数称为分形维数。通常，它们的维数值不是整数。”

对于不同的对象，分维有不同的计算方法，下面介绍几个分维的定义。

1. 豪斯道夫维数

取长度为l 的线段，放大2 倍后的长度$2l$。边长为l 的正方形，每边长放大2 倍的面积为$4 = l^2$。边长为l 的立方体，每边长放大2倍的体积为$8 = l^3$。如图2.24所示，结果整理如下：

一维图形（线段） $2^1 = 2$

二维图形（正方形） $2^2 = 4$

三维图形（立方体） $2^3 = 8$

归纳得：

$$L^D = K, \quad D = 1, 2, 3$$

取对数

$$D_f = D = \frac{\log K}{\log L}$$

这就是豪斯道夫维数。

推论：对于正规几何图形，分子为分母整除，D_f 为整数，是欧几里得维数。对非规则图形，分子与分母不是总可以整除，D_f 一般是分数，称为分维。

例1 以Sierpinski三角形为例，设想从一个小三角形开始，将每边扩大2 倍，得与之相似的大三角形，面积为小三角形4倍。将中间一个小三角形舍去，实际

面积为小三角形的3倍。按定义计算分维数

$$D_f = \frac{\log K}{\log L} = \frac{\log 3}{\log 2} = 1.5849$$

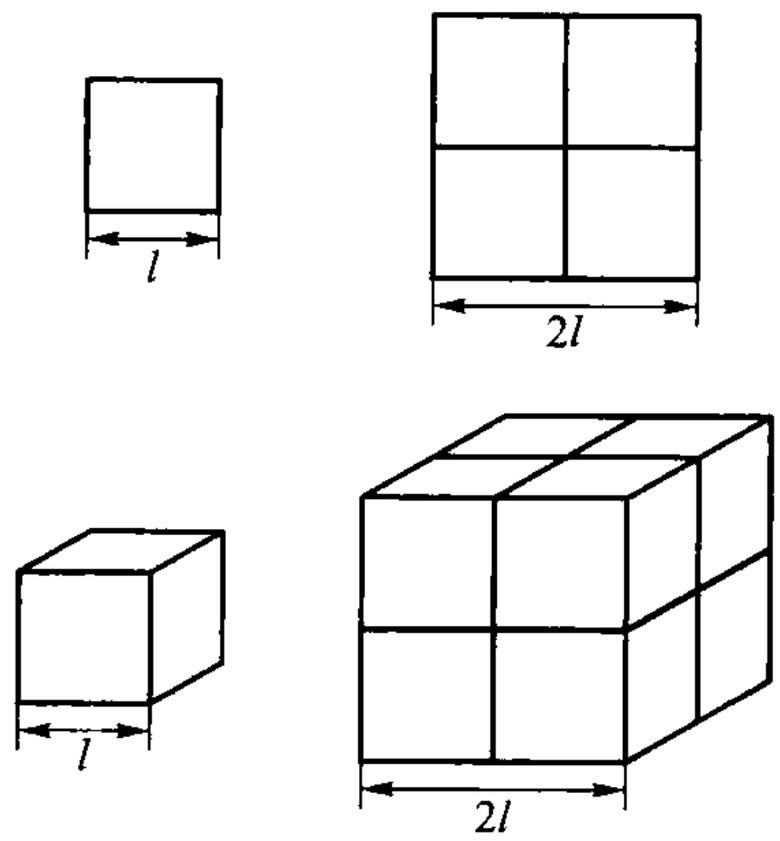


图 2.24 豪斯道夫维数的计算

得到Sierpinski三角形的维数是1.5849而不是2，所以它不是二维图形。分维的概念很好地描述了Sierpinski三角形与欧几里得三角形之间的区别.

2. 相似维数

如图2.25，把单位面积的正方形等分成九个小正方形，每个小正方形边长是原来长度的1/3，即

$$9 \times (1/3)^2 = 1$$

指数2显然为正方形的维数。该式表示局部与整体有相似关系。

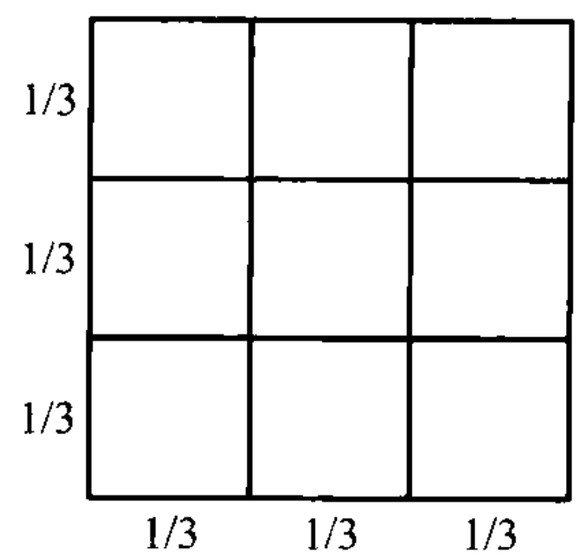


图 2.25 相似维数的计算

定义：假定某个几何体由N个局部组成，每个局部以相似比β 与整体相似，则客体的相似维数为

$$D_s = \frac{\log N}{\log(1/\beta)}$$

例2 边长为$2l$的正方体，四等分得边长l的四个小正方形。小正方形边长与原正方形边长之比为 $\beta=1/2$，局部与整体的相似比为 $b=l/2l=1/2$，相似维数 D_s 为

$$D_s=\frac{\log N}{\log(1/\beta)}=\frac{\log 4}{\log 2}=2=D_f$$

例3 科赫雪花曲线每条边由四个与整体相似的局部组成，每个局部的长度是整体长度的1/3，即相似比$\beta=1/3$，因此，科赫曲线相似维数

$$D_s=\frac{\log N}{\log(1/\beta)}=\frac{\log 4}{\log 3}=1.2618$$

3. 盒维数

前两个定义只适用于具有完全的自相似结构的图形，对于不是完全自相似的图形，可以用盒维数的定义。

盒维数的计算方法是：将分析对象置于一个网格中，每个网孔(也称盒子)的尺寸为s，数出包含了分析对象的网孔数目N。显然这个数目取决于网孔的大小s，所以记为$N(s)$。逐步缩小网孔的尺寸s，记录下对应的数目$N(s)$。将 $\log N(s)$ 和 $\log(1/s)$ 画在一张图中，得到一张用双对数坐标表示的曲线。如果它是直线，则它的斜率就代表分析对象的盒维数。

实际应用中，为了方便，通常是按1/2的比例依次缩小网孔，即将网格的一个网孔分成 4 个小网孔就得到下一张网格。而测量得到的数目就是 $N(2^{-k})$ $(k=0,1,2,\cdots)$. 其中最大的网孔的尺寸是$s=2^0=1$。在双对数图中，直线的斜率为

$$\frac{\log N[2^{-(k+1)}]-\log(2^{-k})}{\log 2^{k+1}-\log 2^k}=\log_2\frac{N[2^{-(k+1)}]}{N(2^{-k})}$$

上式右端用的是以2为底的对数，而左端可以用以任意数为底的对数。计算结果是相邻两次计算出的盒子数目之比然后取2为底的对数，所得斜率就是分形图形的盒维数。或者说，如果网格盒子的尺寸每次减半，所计算的盒子数目以2^D为因子增加，则分形维数是D。

盒维数在科学测量中应用最广，它的优势在于简单并且可用计算机计算，它只需要简单地统计盒子数目就行。它既适应于完全自相似的图形，也能用于非自相似的图形，甚至用于高维的图形。

以英国海岸线为例，图2.26将它画在两个网格中，网格的总宽度归一化为1，网孔的尺寸分别为1/24和1/32，这里没有按1/2的比例来缩小网孔。

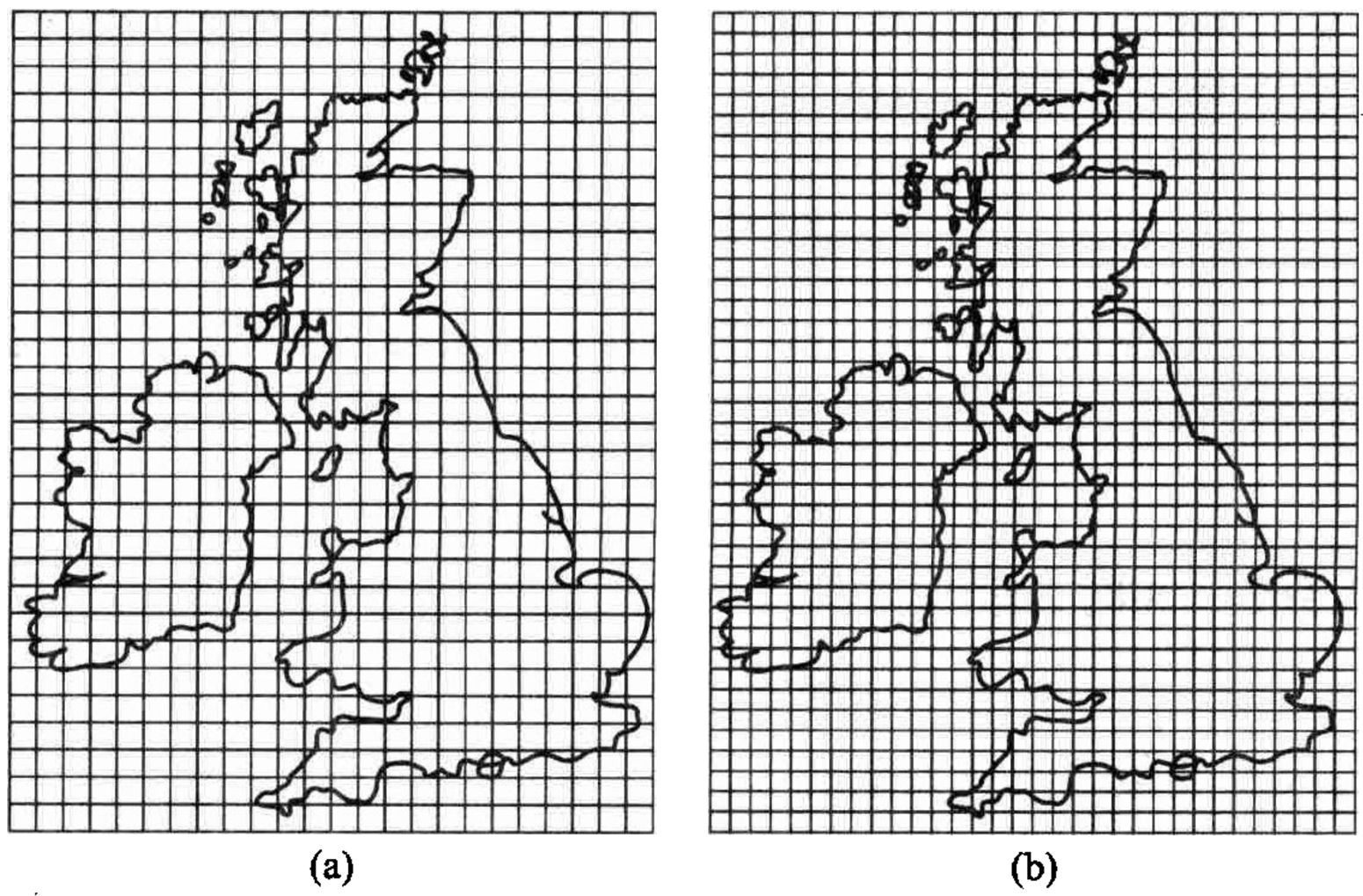


图 2.26 计算海岸线盒维数

与海岸线相交的盒子数分别是194和283。读者可以自己在图上数一下。由此得到双对数图中的直线斜率是

$$d=\frac{\log 283-\log 194}{\log 32-\log 24}\approx\frac{2.45-2.29}{1.51-1.38}\approx 1.31$$

这样测量出来斜率是一个确定的值，不会随系统的尺度而改变。所以分维是一个不随系统的尺度改变而改变的量。分形的概念将尺度这个自变量引入了物理学中，许多自然现象离开尺度来谈就毫无意义，例如海岸线的长度，以m为单位和以km 为单位来测量，其结果大不相同，如果以mm为单位，测量的结果差别更大。所以在多尺度系统中，物理量是随尺度而变化的，关键的问题是寻找该系统尺度变化时不会随尺度变化的特征量，分数维就是一个这样的不变量。

前面的分析都是从几何的意义来说明分维的概念，也可以从物理的意义上来说明分维的概念。

物体的质量与它的维度d存在比例关系，例如一维物体的质量与其长度成正比，二维物体的质量与其面积成正比，三维物体的质量与其体积成正比，将这种关系写成

$$\mu(l)\sim l^d$$

更一般的形式是将它写成与尺度变化之间的关系。例如，立方体的边长增加一倍，则质量增加为2^3倍，即有

$$\mu(\lambda l)=\lambda^d\mu(l) \tag{2.1}$$

再回到对Sierpinski三角形的研究。设最初的小三角形的边长为$L=l$，质量为$\mu(l)$。将小三角形的边长放大一倍即$L=2l$以后得到的Sierpinski三角形有3个小三角形，它的质量是最初的3倍，即

$$\mu(2l)=3\mu(l)$$

再将边长放大一倍得到边长为$L=4l$的Sierpinski三角形有9个小三角形，它的质量是最初的9倍，即

$$\mu(4l)=9\mu(l)$$

当边长扩大到$L=8l$时，有

$$\mu(8l)=27\mu(l)$$

所以质量与尺度变化之间的关系是

$$\mu(2^n l)=3^n\mu(l)$$

对比(2.1)式有

$$(2^n)^{d_f}=3^n$$

即

$$d_f=\frac{\ln 3}{\ln 2}\approx 1.5849...$$

这个结果与前面的讨论是一致的。

§2.4 练 习

1. 编写一个函数文件，求小于任意自然数n的斐波那契数列各项。斐波那契数列定义如下：

$$\begin{cases} f_1=1 & (n=1) \\ f_2=1 & (n=2) \\ f_n=f_{n-1}+f_{n-2} & (n\geqslant 3) \end{cases}$$

2. 将一段线段划分为三等份，然后将中间一段删去，留下来的二段分别重复上述操作，经过无限次操作后，所得到的点集称为康托尔集，画出康托尔集的图像。

3. 如图2.27，在一单位长度的线段上对其三等分，将中间段直线换成一个去掉底边的等边三角形，再在每条直线上重复以上操作，如此进行下去直到无穷，就得到分形曲线——科赫曲线。

4. 如图2.28所示，将一条线段4等分，将中间两段去掉，分别在上下代之以两个正方形的三条边，然后对每条边都重复这个操作，所得的曲线叫Minkowski“香肠”。编出画图程序。

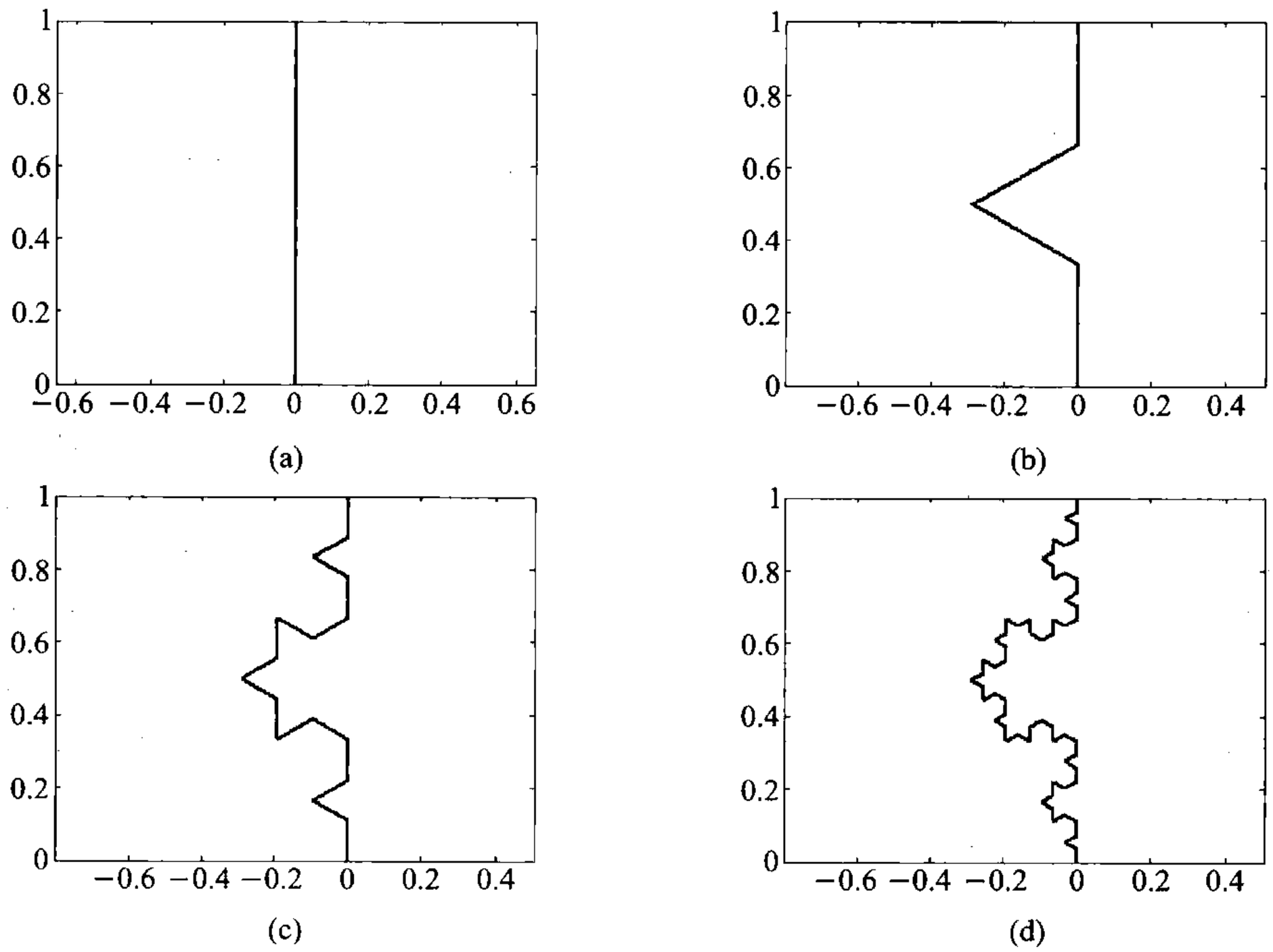


图 2.27　另外一种科赫曲线

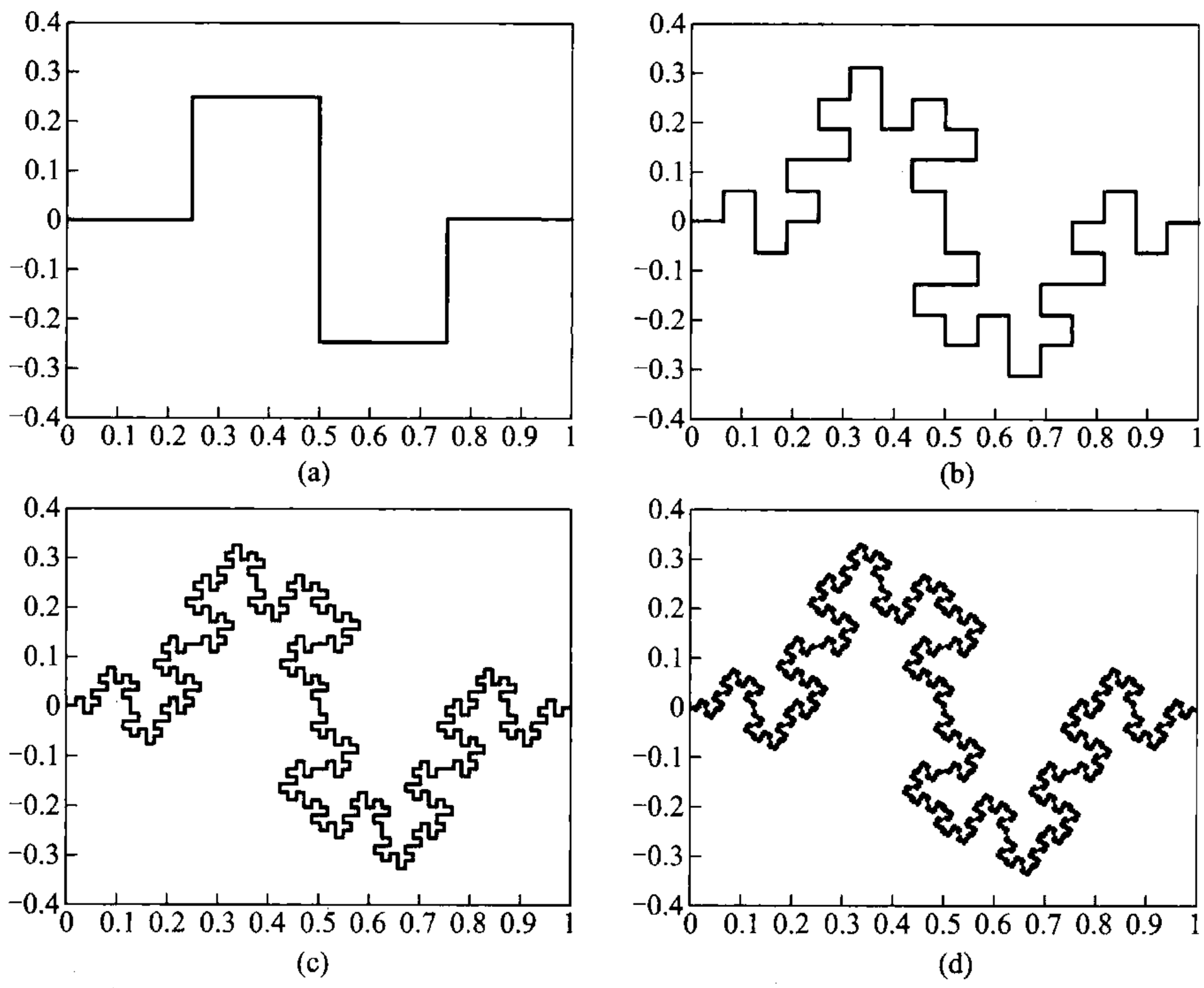


图 2.28　Minkowski“香肠”

第三章　数值微分与数值积分

§3.1　数值微分与数值积分

在数学分析中，求导数是先求出函数的导函数，再将变量值代入导函数中，得到该处的导数值；而求定积分则是先求出不定积分公式，再将积分上下限代入就得到定积分的值。这些计算都要通过公式推导即解析计算来完成。

数值微分与数值积分则不同，是近似的数值计算，不一定要求出导函数或不定积分公式，而是利用差分公式计算导数，用梯形公式或其他数值积分公式如辛普森公式等计算定积分。这样的方法非常实用，对一些不能求出不定积分公式的定积分也可能求得结果。

§3.1.1　数值微分的算法

下面推导数值微分计算中常用的几个公式，由泰勒展开公式

$$f(x+h)=f(x)+hf'(x)+O(h^2)$$

得出近似计算一阶导数的前差公式

$$f'(x)=\frac{f(x+h)-f(x)}{h}$$

类似地有后差公式

$$f'(x)=\frac{f(x)-f(x-h)}{h}$$

将两式平均，得出中心差分(平均值)公式

$$f'(x)=\frac{f(x+h)-f(x-h)}{2h}$$

将前差公式中的函数换成它的导函数，就能得到计算二阶导数的近似公式

$$\begin{aligned}f''(x)&=\frac{f'(x+h)-f'(x)}{h}\\&=\frac{f(x+h)-2f(x)+f(x-h)}{h^2}\end{aligned}$$

可以继续推导计算三阶导数以上的近似公式。在泰勒展开式中保留不同的项数将得到精度不同的公式。

§3.1.2 数值积分的算法

这里推导数值积分的近似计算公式。一维定积分$I=\int_a^b f(x)\mathrm{d}x$的几何意义是曲线下面所包围的面积，按照积分的定义，它是许多小曲边梯形面积之和的极限

$$I=\int_a^b f(x)\mathrm{d}x=\lim_{n\to\infty}\sum_{k=1}^{n} f(\xi_k)\frac{b-a}{n}$$

令$h=\dfrac{b-a}{n}$，得到下面两个近似计算定积分的近似公式，它们都叫矩形公式，都是用矩形来替代曲边梯形。但矩形的高不同，前者是用曲边梯形的左边作为矩形的高，后者是用曲边梯形右边的高作为矩形的高。

$$L_n=h\sum_{k=0}^{n-1} f_k,\qquad R_n=h\sum_{k=1}^{n} f_k$$

将两式平均得出近似计算定积分的梯形公式

$$T_n=h\sum_{k=1}^{n-1} f_k+\frac{h}{2}(f_0+f_n)$$

再推导精度更高的辛普森公式，令$h=\dfrac{a-b}{2n}$，以$2k$ 标记偶数节点($k=0,1,2,\cdots,n-1$)，第k段区间包含三个节点$(x_{2k},f_{2k}),(x_{2k+1},f_{2k+1}),(x_{2k+2},f_{2k+2})$，在这个区间内对中间的点$x_{2k+1}$作泰勒展开

$$\begin{aligned}f(x)&\approx f(x_{2k+1})+f'(x_{2k+1})(x-x_{2k+1})+\frac{1}{2!}f''(x_{2k+1})(x-x_{2k+1})^2\\&\approx f_{2k+1}+\frac{f_{2k+2}-f_{2k}}{2h}(x-x_{2k+1})+\frac{f_{2k+2}-2f_{2k+1}+f_{2k}}{2h^2}(x-x_{2k+1})^2\end{aligned}$$

用泰勒展开式代替原来的积分函数，在这个区间内积分得

$$\int_{x_{2k}}^{x_{2k+2}} f(x)\mathrm{d}x\approx\frac{h}{3}(f_{2k}+4f_{2k+1}+f_{2k+2})$$

上式对k求和得近似计算定积分的辛普森公式

$$S_n=\sum_{k=0}^{n-1}\frac{h}{3}(f_{2k}+4f_{2k+1}+f_{2k+2})$$

整理得

$$S_n=\frac{h}{3}\left(f_0+f_{2n}+2\sum_{k=1}^{n-1} f_{2k}+4\sum_{k=0}^{n-1} f_{2k+1}\right)$$

这些积分公式可以用一种更简单直观的方法来理解。令$h=b-a$表示一个区间的宽度，可以引入计算积分的中点法则

$$M=h\cdot f\left(\frac{a+b}{2}\right)$$

而梯形法则就是

$$T=h\cdot\frac{f(a)+f(b)}{2}$$

通过检验积分法则对多项式函数的效果，可以估计它的准确性。数值积分法则的阶数是指该法则不能准确计算的多项式的最低次数。若用一个p 阶的数值积分法则对一个光滑函数在长为h的小区间上求积分，那么泰勒级数分析表明，误差正比于h^p。前面提到的中点法则和梯形法则对x是常数和线性函数是完全准确的，但对于x的二次函数就不准确，因此相应有$p=1$和$p=2$。

这两种数值积分法则的准确性可以通过计算一个简单的积分来检验

$$\int_0^1 x^2\mathrm{d}x=\frac{1}{3}$$

中点法则计算的结果是

$$M=1\times\left(\frac{1}{2}\right)^2=\frac{1}{4}$$

梯形法则计算的结果是

$$T=1\times\left(\frac{0+1^2}{2}\right)=\frac{1}{2}$$

因此M的误差为1/12，而T的误差为$-1/6$。这两个误差的正负号相反，并且可能让人惊讶的是，中点法则的准确性竟然是梯形法则的两倍。

可以证明这种现象有普遍性。在小区间对光滑函数进行积分，M的准确性基本上是T的两倍，同时正负号相反。有了这样的误差估计，就可以将这两者加以结合得到新的积分法则。得到的新法则通常要比这两者任何一个都要准确。如果T的误差正好是M的误差的-2倍，那么求解下式

$$S-T=-2(S-M)$$

所得的S将给出积分的准确值。通常这个方程的解

$$S=\frac{2}{3}M+\frac{1}{3}T$$

是比M或T更准确的积分近似值。这样得到的积分法则叫辛普森法则。如果令$c=(a+b)/2$，则有

$$S=\frac{h}{6}[f(a)+4f(c)+f(b)]$$

这就是前面推导的辛普森公式的结果。用S可以准确地计算三次多项式的积分，但对四次多项式则不行，因此辛普森法则的p为4。

如果对区间$[a,c]$，$[c,b]$再运用辛普森法则。并令$d=(a+c)/2, e=(c+b)/2$就会得到对整个区间的一个新的数值积分公式

$$S_2=\frac{h}{12}[f(a)+4f(d)+2f(c)+4f(e)+f(b)]$$

这就是复合辛普森公式。这些法则积分的区别可以用图3.1来表示。

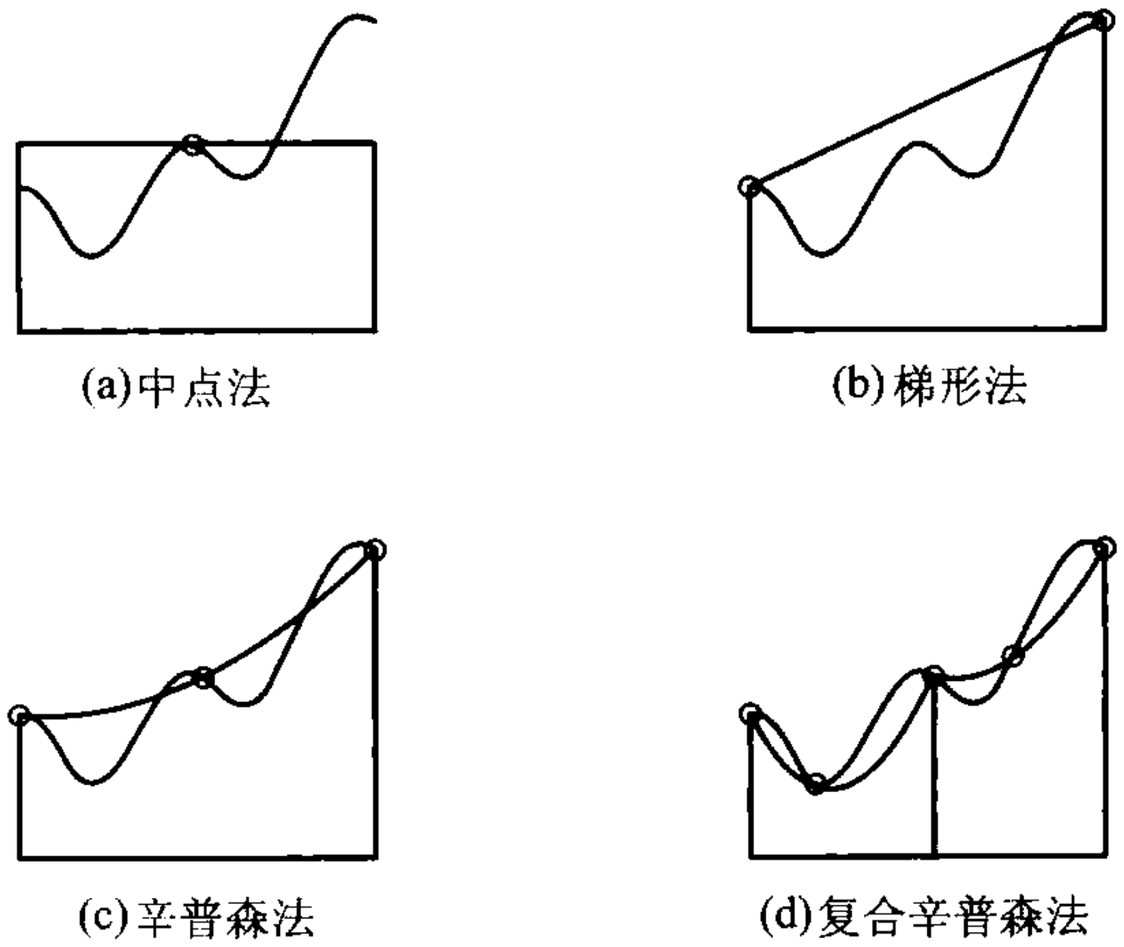


图 3.1 四种积分计算法则

S和S_2是同一个积分的近似值，因此它们之间的差可以用于估计误差

$$E=S_2-S$$

而且它们还可以结合起来，得到一个更加准确的近似值Q。由于这个两积分法则是四阶的，而S_2 的步长是S的一半，因此，S_2准确性大约是S的2^4倍。所以应该求解方程

$$Q-S=16(Q-S_2)$$

得到Q，其结果是

$$Q=S_2+(S_2-S)/15$$

这个公式叫Weddle法则，六阶牛顿–柯斯特法则，同时它也是龙贝格积分方法的第一步。也称为外推的辛普森公式，因为它对两个不同的h值使用辛普森法则，然后向$h=0$的极限情况进行外推。

§3.2 MATLAB指令

§3.2.1 差分运算(diff)

在数值计算中，差分$\Delta x = x_{n+1} - x_n$是微分$\mathrm{d}x$的近似，而差商是函数的差分除以自变量的差分$\dfrac{\Delta y}{\Delta x}$，是导数$\dfrac{\mathrm{d}y}{\mathrm{d}x}$的近似。

计算矩阵的差分的指令是diff，下面是它的用法举例。

```
>>a=[1,4,5,-2];
>>diff(a)
ans = 3 1 -7      %矢量的差分是后项减前项,结果会比原来少一个元素
>>b = [0.95013,        0.48598;
       0.23114,        0.8913;
       0.60684,        0.7621 ];
>> diff(b)                          %对矩阵求差分是对列矢量作差分
ans = -0.71899      0.40532
       0.3757      -0.1292
```

§3.2.2 梯度计算(gradient)

梯度是相邻两个差商值的平均(即用导数的中心差分式计算)。在端点则取其差分值。

一维矢量的梯度就是用中心差分式计算导数，在端点直接用差商作为近似值。自变量就是矩阵元素的编号，所以步长是1。如果自变量的步长不是1，则所得结果还要除以自变量的增量值才能得到导数值。矩阵的梯度是行向量与列向量两个方向的偏导数。

```
>> a=[1,  5,  4,  2,  7;
      2,  1,  8,  3,  4]
>> [px,py]=gradient(a)
px = 4  1.5 -1.5 1.5 5     %行向量方向的偏导数,第一个元素4=5-1,第
    -1  3    1  -2   1     %二个元素1.5=[(5-1)+(4-5)]/2, 其余类推
py = 1 -4  4  1 -3                          %列向量方向的偏导数
     1 -4  4  1 -3
```

§3.2.3 离散拉普拉斯算符(del2)

把矩阵U看成是在矩形网格上每一点赋值的函数$u(x,y)$，del2(U)的定义为

$$L = 0.25\Delta_2 u = \frac{1}{4}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{\mathrm{d}^2u}{\mathrm{d}y^2}\right)$$

L是一个与U大小相同的矩阵，Δ_2是二维拉普拉斯算符。矩阵内部的各个元素是

$$L_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - u_{i,j}$$

在矩形网格的边缘则使用三次方的外推法来使用这个公式。对有N个变量的多元函数$u(x,y,z,\cdots)$, del2(U)表示

$$L = \frac{1}{2N}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{\mathrm{d}^2u}{\mathrm{d}y^2} + \frac{\mathrm{d}^2u}{\mathrm{d}z^2} + \cdots\right)$$

下面的例子表明，对函数$U(x,y) = (x^2+y^2)$有$\Delta_2 U = 4$，而程序式4*del2(U)算得的结果也是4，可见两者是等效的。

```
[x,y] = meshgrid( -2:2, -3:3 );
U = x.*x + y.*y
U =
    13    10     9    10    13
     8     5     4     5     8
     5     2     1     2     5
     4     1     0     1     4
     5     2     1     2     5
     8     5     4     5     8
    13    10     9    10    13
V = 4*del2(U)
V =
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
```

§3.2.4 梯形积分(trapz)

指令trapz是用梯形公式计算定积分。对向量积分时是把向量的每个元素当作函数值，按照梯形公式，应该是相邻两个函数值之和除2再乘以自变量的增量，在没有说明自变量及其增量时，默认元素的编号就是自变量。而矩阵的梯形积分是对列矢量进行的，也可以根据需要指定积分的方向。

下面是一个示例：

```
>> a=[1, 5, 7, 2, 3];
>> trapz(a)
ans =  16          %向量的梯形积分即((1+5)+(5+7)+(7+2)+(2+3))/2=16
>>b = 0.95013       0.48598
                   %矩阵的积分默认为对列向量积分，得出的是行向量
      0.23114       0.8913
      0.60684       0.7621
>> trapz(b)
ans = 1.0096       1.5153
>> trapz(b,2)                %指定对行向量积分，得出的结果是列向量
ans =
      0.71806
      0.56122
      0.68447
```

§3.2.5 累计梯形积分(cumtrapz)

不定积分

$$I=\int_{x_0}^{x} f(t)\mathrm{d}t \qquad (x=x_1,x_2,\cdots)$$

在上限x连续取值时会得到一系列的结果。累计梯形积分相当于不定积分的数值计算，所得的结果是矢量，它的第n个元素是原矢量前n个元素的梯形积分。矩阵的累计梯形积分对列矢量进行，当然也可以指定对行向量作累计梯形积分。

例如接前例继续计算有：

```
>> cumtrapz(b)
ans =        0             0      %这里的最后一行就是trapz(b)的结果
      0.59063       0.68864
       1.0096        1.5153
```

§3.2.6 函数积分(quad,quadl)

如果已知积分函数，可以用指令quad和quadl计算定积分的数值，quad是用自适应辛普森法，也就是上述外推的辛普森法则，而quadl是用精度更高的自适应洛巴托(Lobatto)法，在数值计算中也称之为高斯积分法。与符号变量积分指令int不同，它不是进行定积分的公式运算，而是直接求出定积分的值。

所谓自适应积分就是可以根据积分精度的需要，自动调节积分取点的数目。如前所述，积分是计算各个小区间的面积和。小区间越小，积分值就越精确，但是这样就要增加取点的数目，也要增加计算量。为了兼顾这两者，自适应积分的做法是，首先取较大区间$[a,b]$计算区间内的积分，然后在区间内取一点C，再计算两个小区间$[a,c],[c,b]$内的积分，如果在误差范围内有

$$\int_a^b f(x)\mathrm{d}x = \int_a^c f(x)\mathrm{d}x + \int_c^b f(x)\mathrm{d}x$$

那么这个区间的积分就完成了，如果等式两边的差别超过误差范围，那么就用等式右边的两个积分取代左边的积分，同时再对两个小区间的积分继续重复上述过程，检验积分的精度，如此下去，直到达到要求。这样的算法可自动适应各种被积函数，在被积函数变化剧烈的地方，将区间划分为较密的子区间，在被积函数变化缓慢的地方，子区间就比较稀疏。

```
>> f=inline('exp(x.^(-3))','x')                    %建立积分函数
f = Inline function:
     f(x) = exp(x.^(-3))
>>format long                  %为了对比结果，要在屏幕上显示更多的数字
>> quad(f,0.5,0.6)
ans =  71.54649248133561     %两个指令进行计算的结果对比，精度不同
>> quadl(f,0.5,0.6)
ans =  71.54649247308223
```

在7.0以后的版本，使用@定义的匿名函数是首选，因为它可以生成更高效率的程序代码。例如上边的例子就是

```
>> f=@(x)exp(x.^(-3))
>> quad(f,0.5,0.6)
```

有时候使用函数文件更好，例如计算

$$\int_0^\pi \frac{\sin x}{x}\mathrm{d}x$$

这就是求MATLAB内部的Si函数在π处的值。写成函数文件是

```
function  f=sinc(x)
if x==0
   f=1,
else
   f=sin(x)./x
end
```

然后积分

```
>>quad(@sinc,0, pi)
ans=1.8519
```

这个结果与后面用符号工具箱所得的结果一样。注意在函数文件中变量的运算必须要用数组运算符号，也就是说，变量是以数组形式输入的。

对含有参数的积分也应该用函数文件，例如计算积分

$$\beta(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}\mathrm{d}t$$

这就是MATLAB内部的β函数，写成函数文件是

```
function  betaf(t,z,w)
f=t.^(z-1).*(1-t)^(w-1)
```

在函数文件中变量必须放在前面，参数放在后面，调用时要保持它们的顺序。

```
z=8/3, w=10/3, tol=1e-6
quad(@betaf,0,1,tol,z,w)
```

这里的tol是积分要达到的精度，这是指令quad的另一种用法。

§3.2.7 函数的二重积分(dblquad)

指令dblquad是计算已知函数的二重积分，积分时要按积分变量顺序指明两个变量的积分区间。如计算

$$\int_{\pi}^{2\pi}\mathrm{d}x\int_0^{\pi}\mathrm{d}y(y\sin x+x\cos y)$$

的操作如下

```
>> ff=inline('y.*sin(x)+x.*cos(y)','x','y')        %建立积分函数
>> dblquad(ff,pi,2*pi,0,pi)     %x,y的积分区间为(pi,2*pi),(0,pi)
ans =  -9.8696
```

§3.2.8 函数的三重积分(triplequad)

指令triplequad是计算已知函数的三重积分，用法与指令dblquad一样，积分时要按积分变量顺序指明三个变量的积分区间，还可规定积分精度。可以直接

将函数写在指令内，也可以另外编写一个函数文件。如计算

$$\int_0^{\pi} \mathrm{d}x \int_0^1 \mathrm{d}y \int_{-1}^1 \mathrm{d}z(y\sin x + z\cos x)$$

的操作如下

```
>>Q = triplequad(@(x,y,z) (y*sin(x)+z*cos(x)),0,pi,0, 1, -1, 1)
Q = 2
```

也可以先编写一个函数文件integrnd.m：

```
function f = integrnd(x, y, z)
     f = y*sin(x)+z*cos(x);
```

然后用函数句柄@调用函数

```
>>Q = triplequad(@integrnd, 0, pi, 0, 1, -1, 1)
```

§3.3 用符号工具箱计算微积分

MATLAB也能够求导函数和推导不定积分公式，这属于符号运算，要用到符号运算工具箱，这是借助加拿大滑铁卢大学研发的数学软件Maple的计算功能来实现的。有关的计算指令有：

diff	符号变量微分	taylor	泰勒展开
int	符号变量积分	symsum	求级数和
limit	求极限	simple	将表达式化简

下面是几个简单的例子，要详细了解它们的用法应该找更全面的参考资料。

```
>> syms x z                                    %指定符号变量
>> diff(log(x+1))                              %求函数的一阶导数
ans = 1/(x+1)
>> diff(sin(x^2),2)                            %求二阶导数
ans = -4*sin(x^2)*x^2+2*cos(x^2)
>> diff(x/(1+x^2),3)                           %求三阶导数
ans = 48/(1+x^2)^3*x^2-6/(1+x^2)^2-48*x^4/(1+x^2)^4
>> int(x/(1+x^2))                              %计算不定积分公式
ans = 1/2*log(1+x^2)
>>  int(z/(1+x^2),x)        %指定积分变量则其他符号变量成为参数
ans =z*atan(x)
>>  int(x*log(1+x),0,1)                        %计算区间(0,1)的定积分
ans = 1/4
```

```
>> int('sin(x)/x',0,pi)                    %如果没有定义变量x，可以加引号
ans =sinint(pi)                            %积分结果是Si函数
>> eval(ans)                               %用指令eval求Si函数在π这点的值
ans = 1.8519
>>  limit(sin(x)/x)                        %计算函数极限，默认以x=0为极限点
ans = 1
>> limit(1/x,x,0,'right')                  %求函数的右极限
ans = Inf
>> limit(1/x,x,0,'left')                   %求函数的左极限
ans = -Inf
>> limit((x-2)/(x^2-4),2)                  %求x=2处函数的极限
ans = 1/4
>>  taylor(exp(-x))                        %在零点求泰勒展开式，默认取前5项
ans = 1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
>> taylor(log(x),6,1)                      %在x=1处计算泰勒展开式前6项
ans =  x-1-1/2*(x-1)^2+1/3*(x-1)^3-1/4*(x-1)^4+1/5*(x-1)^5
>> symsum(k,0,n)                           %计算∑_{k=0}^{n} k
ans = 1/2*(n+1)^2-1/2*n-1/2
>>simple(symsum(k,0,n))                    %用指令simple化简上式的结果
ans =  1/2*n*(n+1)
>> symsum(1/k^2,1,Inf)                     %计算有无穷项的级数和
ans = 1/6*pi^2
```

§3.4 环形电流的磁场——物理场的可视化

环形电流的磁场是普通物理中的一个实验内容，但是它的计算涉及偏微分方程的解，在某些数学物理方法的教材中有相应的解法，本节介绍用计算机根据毕沙定律通过数值积分来求出环形电流的空间磁场分布。

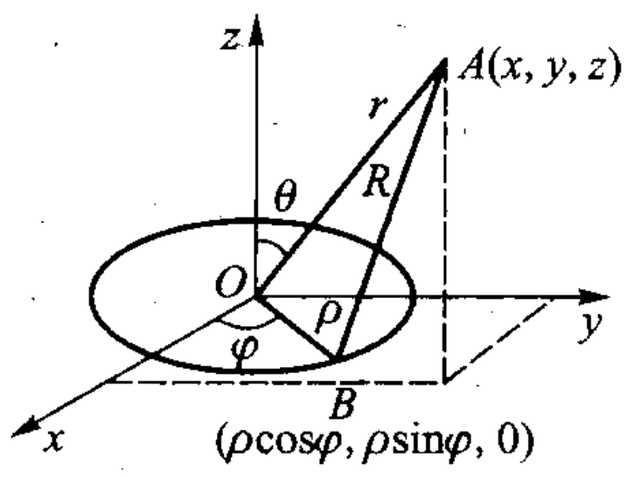


图 3.2 磁场计算示意图

如图 3.2 建立坐标系，在 B 点的电流密度是 λ 的圆环电流元在 A 点产生的磁场是

$$\mathrm{d}\boldsymbol{B} = \frac{\mu_0}{4\pi}\frac{\lambda}{R^3}\mathrm{d}\boldsymbol{l}\times\boldsymbol{R}$$

$$\mathrm{d}\boldsymbol{l} = -\rho\sin\varphi\mathrm{d}\varphi\boldsymbol{i} + \rho\cos\varphi\mathrm{d}\varphi\boldsymbol{j}$$

$$\boldsymbol{R}=(x-\rho\cos\varphi)\boldsymbol{i}+(y-\rho\sin\varphi)\boldsymbol{j}+z\boldsymbol{k}$$

代入得

$$\mathrm{d}\boldsymbol{B}=\frac{\mu_0\lambda}{4\pi R^3}[z\rho\cos\varphi\mathrm{d}\varphi\boldsymbol{i}+z\rho\sin\varphi\mathrm{d}\varphi\boldsymbol{j}+(\rho^2-y\rho\sin\varphi-x\rho\cos\varphi)\mathrm{d}\varphi\boldsymbol{k}\ ]$$

当A点在Oyz平面时，其坐标为$A(0,y,z)$，且有

$$\boldsymbol{R}'=(-\rho\cos\varphi)\boldsymbol{i}+(y-\rho\sin\varphi)\boldsymbol{j}+z\boldsymbol{k}$$

取上面$\mathrm{d}\boldsymbol{B}$式中的$x=0$，得到Oyz面上点$A(0,y,z)$产生的磁场。此时有

$$\int_0^{2\pi}\mathrm{d}B_x=\int_0^{2\pi}\frac{z\rho\cos\varphi\mathrm{d}\varphi}{(\rho^2+r^2-2y\rho\sin\varphi)^{3/2}}$$

$$=\frac{z\rho}{y\rho}\frac{1}{(\rho^2+r^2-2y\rho\sin\varphi)^{1/2}}\bigg|_0^{2\pi}=0$$

所以在画二维图形时，只要计算

$$\mathrm{d}\boldsymbol{b}=\frac{\mu_0\lambda}{4\pi R^3}[z\rho\sin\varphi\mathrm{d}\varphi\boldsymbol{j}+(\rho^2-y\rho\sin\varphi)\mathrm{d}\varphi\boldsymbol{k}\ ]$$

利用上式，可以编出下面的程序，程序中省略了常数 $\dfrac{\mu_0\lambda}{4\pi}$。图 3.3 为画出的图形。

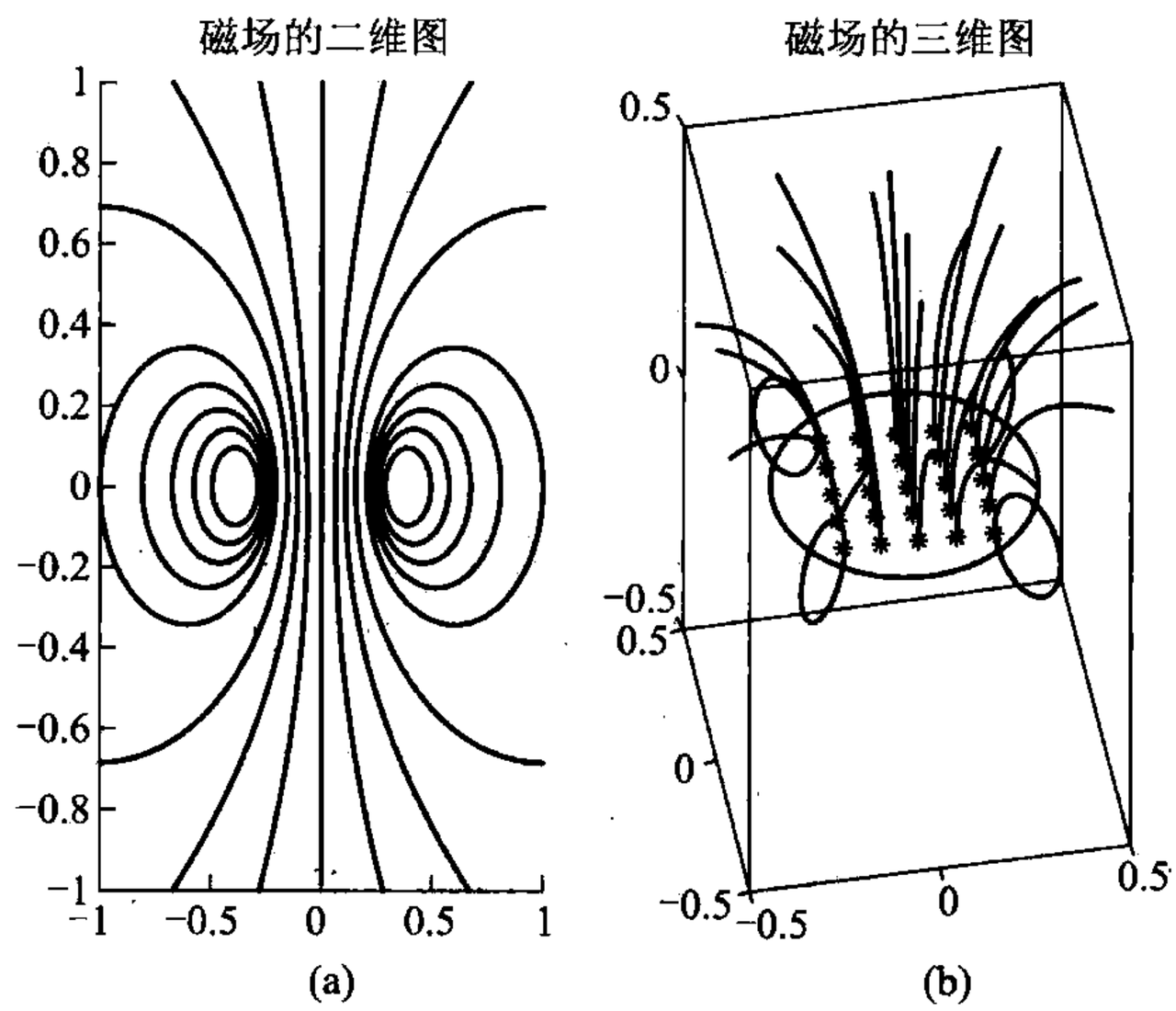


图 3.3 计算所得的磁场线分布图

```
%画二维图形
a=0.35;    the=0:pi/20:2*pi;                    %半径及θ变量
```

```
y= -1:0.04:1;    z= -1:0.04:1;                                  %y, z变量
[Y,Z,T]=meshgrid(y,z,the);                                      %三维数据网格
r=sqrt((a*cos(T)).^2+(Y-a*sin(T)).^2+Z.^2);
r3=r.^3;
dby =a*Z.*sin(T)./r3;     by=pi/40*trapz(dby,3);                %B的y分量
dbz =a*(a-Y.*sin(T))./r3; bz=pi/40*trapz(dbz,3);                %B的z分量
figure(1)
[bSY,bSZ]=meshgrid([0:0.05:0.2],0);
h1=streamline(Y(:,:,1),Z(:,:,1),by,bz,bSY,bSZ,[0.1,1000]);
h2=copyobj(h1,gca);
%用图形复制与旋转画其余的象限,代替下面三句
rotate(h2,[1,0,0],180,[0 0 0]);
h3=copyobj(allchild(gca),gca);
rotate(h3,[0,1,0],180,[0 0 0]);
title('磁场的二维图','fontsize',15);
for kk=1:4                                                      %曲率较大的场线要取不同的步长
[bSY,bSZ]=meshgrid(0.2+kk*0.02,0);
streamline(Y(:,:,1),Z(:,:,1),by,bz,bSY,bSZ,[0.02/(kk+1),4500]);
streamline(-Y(:,:,1),Z(:,:,1),-by,bz,-bSY,bSZ,[0.02/(kk+1),
4500]);
end
%以下画三维图形
[X,Y,Z]=meshgrid(-0.5:0.04:0.5);
r2=X.^2+Y.^2+Z.^2;
for k=1:81
   phi=pi/40*(k-1);    costh=cos(phi);    sinth=sin(phi);
   R3=(r2+a^2-2*a*(X*costh+Y*sinth)).^(3/2);
   Bx0(:,:,:,k)=a*Z*costh./R3;
   By0(:,:,:,k)=a*Z*sinth./R3;
   Bz0(:,:,:,k)=a*(a-X*costh-Y*sinth)./R3;
end
Bx=pi/40*trapz(Bx0,4);                                          %梯形积分
By=pi/40*trapz(By0,4);
Bz=pi/40*trapz(Bz0,4);
figure(2)
```

```
v=[-0.2,-0.1,0,0.1,0.2];
[Vx, Vy, Vz]=meshgrid(v,v,0);
plot3(Vx(:),Vy(:),Vz(:),'r*')
streamline(X,Y,Z,Bx,By,Bz,Vx,Vy,Vz,[0.01,2000]);
hold on;
axis([-0.5,0.5,-0.5,0.5,-0.5,0.5]);
view(-23,26);
box on;
title('磁场的三维图','fontsize',15);
t=0:pi/100:2*pi;
plot(a*exp(i*t),'r-','LineWidth',3);
```

程序中使用的指令streamline的作图精度与所选的步长有关，在语句中是最后两个参数，一个表示步长，另一个表示步数。适当选择这两个参数的值，才能保证所画的场线闭合，否则可能由于计算精度不够而产生的误差积累会导致场线不闭合。另外这个指令是沿着场强增加的方向去画场线，所以场线的起点应该选在场强较小的的地方，否则画的图形不美观。

指令copyobj是复制图像，rotate是旋转图像。

§3.5 分子振动的半经典量子化

本节计算双原子分子的振动能级。双原子分子中有两个原子核，两个核被环绕它们运动的电子束缚在一起。按照量子理论，两个核之间的距离不同，则分子体系的能量不同，而两个核之间的距离也是量子化的，这就是振动能级。在讨论振动能级时，采用了如下的简化模型：

1. 只考虑原子核的运动

认为两个原子核的运动只受与核间距离r有关的位势的影响而略去电子的运动。这个位势在远距离表现为吸引势(范德瓦耳斯相互作用)，而在近距离表现为推斥势(原子核的库仑相互作用和电子的泡利斥力)。描述这种位势的一个常用的形式是伦纳德–琼斯(Lennard-Jones)势或称6–12势：

$$V(r) = 4V_0\left[\left(\frac{a}{r}\right)^{12} - \left(\frac{a}{r}\right)^{6}\right]$$

其形状如图3.4(a)所示，极小点在$r_{\min} = 2^{1/6}a$处(a为玻尔半径)，深为V_0。

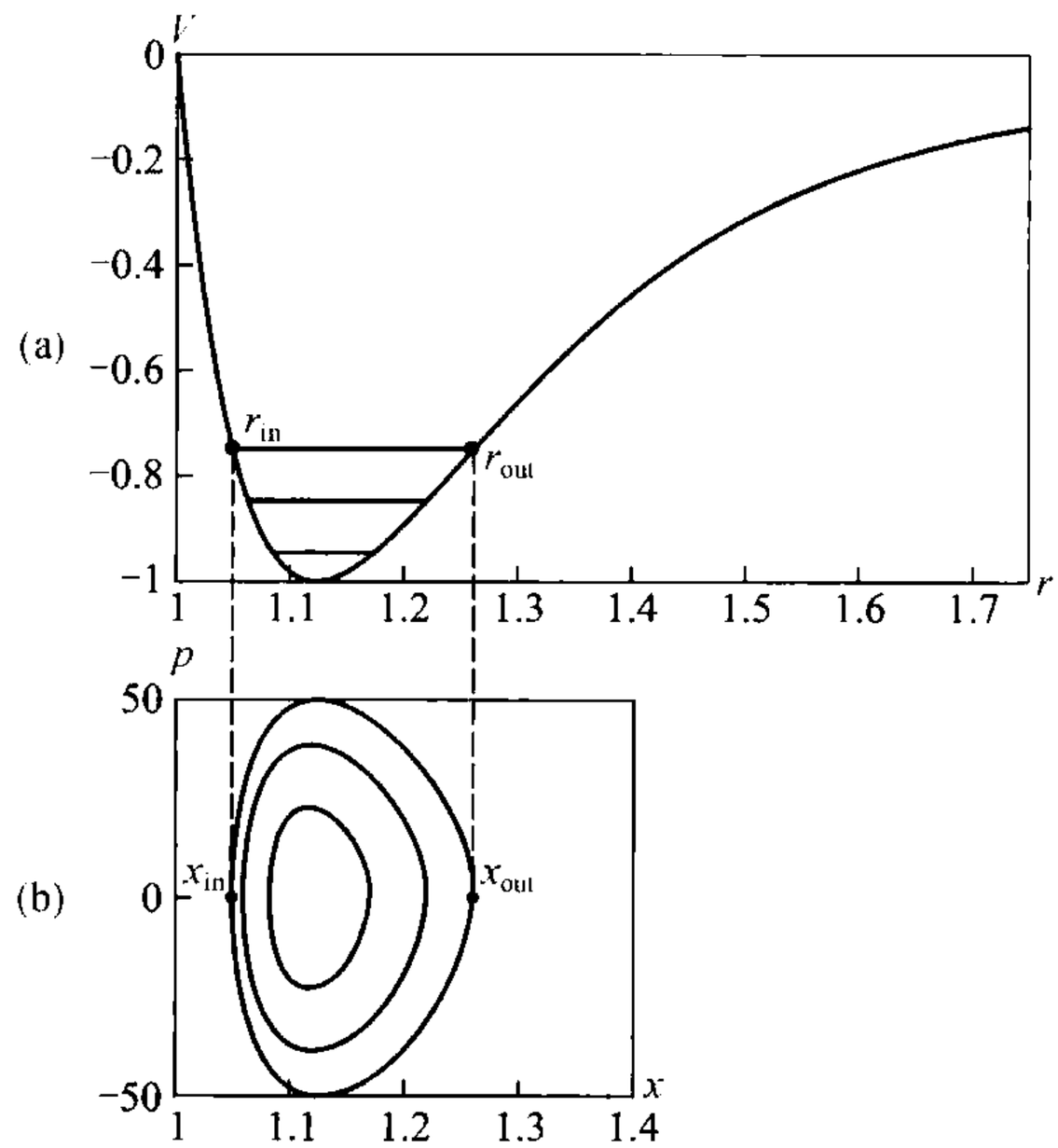


图 3.4

(a) 伦纳德–琼斯势和处于负能量的内外转折点。(b) 相空间中对应的轨道。

2. 转动与振动分开讨论

由于原子核的巨大质量，原子核的转动（类似于一个刚性哑铃）比振动（核间距离的变化）慢得多。

3. 振动方程

核的振动可由一维定态薛定谔方程

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2}+V(r)\right]\Psi_n=E_n\Psi_n \tag{3.1}$$

描述，其中E_n为振动能量，m是两个原子核的约化质量，$\Psi_n(r)$是波函数。

用第四章介绍的解微分本征值方程的方法可以求出这个方程的各个本征值和波函数。这里，我们是把问题看成是原子核在势场V中的经典运动，然后再应用玻尔的“量子化法则”来求量子化的振动能量。

两个原子核构成的原子核系统的总能量是

$$E=\frac{p^2}{2m}+V(r) \tag{3.2}$$

这里，p是原子核的相对运动的动量，上式可化成

$$p(r)=\pm[2m(E-V(r))]^{1/2} \tag{3.3}$$

在能量E给定以后，这是一条在相空间的闭合轨道，如图3.4(b)所示。

运动过程中，系统的动能和势能会相互转换。具体地说，就是原子核间的距离周期性地变化于图中的内转折点r_{in}和外转折点r_{out}之间，在核间距r减小到内转折点r_{in}时，动能为零，势能是斥力势，即$E = V(r_{\text{in}})$ 此时r开始增大；当r增大到外转折点r_{out}时，动能也为零，但势能是引力势，即$E = V(r_{\text{out}})$，所以，r又会开始变小。

对于束缚在势阱V中的原子核系统的经典振动，所有满足条件$-V_0 < E < 0$的运动都会发生，而量子化法则要求：只有在相空间轨道所包围的面积(以$\hbar$为单位) 是2π的半奇数倍时，那些运动才能稳定存在。即

$$\frac{1}{\hbar}\oint p(r)\mathrm{d}r = \left(n+\frac{1}{2}\right)2\pi \tag{3.4}$$

用E_n来表示第n条轨道对应的能量，再将(3.3)式代入，考虑到每个循环中振动都要经过每个r值两次(一次的p值为正，一次的p值为负)，得到

$$\frac{1}{\hbar}\int_{r_{\text{in}}}^{r_{\text{out}}}[2m(E-V(r))]^{1/2}\mathrm{d}r = \left(n+\frac{1}{2}\right)\pi \tag{3.5}$$

其中n是一个非负整数。在积分限即转折点r_{in} 和r_{out}上，被积函数为零。

为了计算方便，定义几个无量纲量

$$\varepsilon = \frac{E}{V_0},\quad x = \frac{r}{a},\quad \gamma = \left(\frac{2ma^2V_0}{\hbar^2}\right)^{1/2}$$

于是(3.5)式变为

$$\gamma\int_{x_{\text{in}}}^{x_{\text{out}}}[\varepsilon_n - v(x)]^{1/2}\mathrm{d}x = \left(n+\frac{1}{2}\right)\pi \tag{3.6}$$

其中

$$v(x) = 4\left(\frac{1}{x^{12}}-\frac{1}{x^6}\right)$$

是用V_0重新标度后的势能。(3.6)式可以看成是一个关于ε_n 的方程，求能级的过程实际上也就是找这个方程的零点。

γ这个量是本问题的量子本性的一个无量纲的测度。在经典极限下($\hbar$ 很小或m很大)，γ很大。若知道分子的转动惯量(可从分子转动的能量得知)和电离能(把分子分解为组成它的两个原子所需的能量)，就能从实验观测测定参量a和V_0，从而定出γ。对于H_2分子，$\gamma = 21.7$；而对于氘分子，$\gamma = 24.8$(一个质子换成氘核时，只是m改变，V_0不变)；对于由两个^{16}O核构成的质量大得多的O_2分子，$\gamma = 150$。这些值都相当大，它们表明，半经典近似适合于描述这种振动运动。

在（3.6）式中，有4个未知量就是$\varepsilon_n, n, x_{\text{out}}, x_{\text{in}}$，其中$x_{\text{out}}, x_{\text{in}}$是相轨迹上的转折点，满足

$$\varepsilon_n - v(x) = 0 \tag{3.7}$$

所以一共有3个方程，4个变量，是无法用解析方法求解的。但是这却是数值计算求解的一个精彩的例子。

数值计算是如何求解这个问题呢？简单地说，就是依赖于计算机的强大计算能力进行搜索比对。事实上，除了三个方程之外，还有一个规律是已知的，就是对应于$n = 1, 2, 3, \cdots$的能级$\varepsilon_1, \varepsilon_2, \varepsilon_3, \cdots$是从小到大排列的，并且都在$(-1, 0)$之间。那么，按照由小到大的顺序选定一个$n$值，例如先取$n = 1$，再选一个最小的$\varepsilon_1$的值，就可以由（3.7）式定出$x_{\text{out}}, x_{\text{in}}$。注意这个方程中的势能项是$-12$次的，所以方程有12个根，其中只有2个实根符合要求。再将求出的值代入（3.6）式，就可以算出积分的值，那么就能比较（3.6）式两边是否相等，这相当于找下述方程的零点：

$$f = \gamma \int_{x_{\text{in}}}^{x_{\text{out}}} [\varepsilon_n - v(x)]^{1/2} \mathrm{d}x - \left(n + \frac{1}{2}\right)\pi \tag{3.8}$$

如果不等，就按照计算精度的要求，稍稍增加ε_1的值，再进行检验，直到找出使方程（3.8）为零的解。依次做下去，就能找出所有的能级。而且每次求一个新能级之后，新的搜索就可以从这个能级开始向更高的能量值搜索，而不需要再从最低的能量值-1开始搜索，这样可以节省搜索的时间。

这个过程的计算量无疑是相当大的，靠人工计算是难以完成的，但有了计算机的帮助完成这种计算并不困难。所以计算机的速度越快，能完成的工作就越多。这个例子很好地表现了数值计算与解析方法在解决问题时的区别。

实际的程序如下，输出的计算结果有：各个能级的能量值，画出各个能级的势能曲线以及它们在相空间对应的轨道如图2.3所示。程序中$\gamma = 100$。

```
function nch3
global x1 x2
n=10;                                                    %计算10个能级
e1=-0.9;                                                 %E1的尝试值
for k=1:n
E(k)=fzero(@hhh,e1,optimset('tolx',1e-6),k);
e1=E(k)+0.1*abs(E(k));
xio(k,1:2)=[x1,x2];
end
E                                                        %输出能级值
subplot(2,1,1);                                          %画图程序
```

```
X=1:0.001:2;
Y=4*(X.^(-12)-X.^(-6));
plot(X,Y)
hold on
for k=1:n
plot([xio(k,1),xio(k,2)],[E(k),E(k)],'r')
end
subplot(2,1,2)                                          %画图程序
for k=1:n
   z=linspace(xio(k,2),xio(k,1),100);
   p=abs(100*sqrt(E(k)-4*(z.^(-12)-z.^(-6))));
   plot(z,p,z,-p)
   hold on
end
function [f2,x1,x2]=hhh(e,kk)                           %子函数文件
global x1 x2
vfun=inline('e-4*(x.^(-12)-x.^(-6))','x','e');
x1=fzero(vfun,1,optimset('tolx',1e-6),e);
x2=fzero(vfun,2,optimset('tolx',1e-6),e);
f2=(kk-0.5)*pi-100*quad(vfun,x1,x2,[],0,e);
```

程序中用到指令fzero是一个求单变量函数零点的指令，在第四章会作介绍。作为练习，可将势能改为抛物线势或有限深方势阱，计算相应的能级。

§3.6 练　　习

1. 用符号工具箱计算

$$\int_0^1 \frac{x^4(1-x)^4}{1+x^2}\mathrm{d}x$$

这个积分能用数值积分计算吗？如果能，应该如何做？

2. 误差函数$\mathrm{erf}(x)$的定义是

$$\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-x^2}\mathrm{d}x$$

用指令quad计算当$x=0.1,0.2,\cdots,1.0$时相应的erf函数值。将结果与MATLAB的内部函数$\mathrm{erf}(x)$的结果进行比较。

3. 已知室温下氢分子的最概然速率$v_\mathrm{p}=1578\ \mathrm{m/s}$，试求室温下氢分子的速率

(1) 在0~ v_{p} 内的分子数占分子总数的百分比；

(2) 在0~3.3v_{p}内的分子数占分子总数的百分比；

(3) 在3e4~3e8 内的分子数占分子总数的百分比。

注：麦克斯韦速率分布函数为

$$f(v)=4\pi v^2\left(\frac{m}{2\pi\kappa T}\right)^{3/2}\exp\left(-\frac{mv^2}{2\kappa T}\right)$$

以最概然速率$v_{\mathrm{p}}=\sqrt{\dfrac{2\kappa T}{m}}$表示为

$$f(v)=\frac{4}{\sqrt{\pi}}\left(\frac{v^2}{v_{\mathrm{p}}^3}\right)\exp\left(-\frac{v^2}{v_{\mathrm{p}}^2}\right)$$

4. 半径为a的均匀细圆环，环上带$4\pi\varepsilon_0 q$的电荷，求空间的电势与电场分布。要求以等势线、等势面表示电势，以箭头、场线或其他方式表示电场。

5. 亥姆霍兹线圈由两个相同的线圈组成，线圈之间的距离与线圈的半径相等，通过两个线圈的电流方向相同、大小相等，计算两个线圈之间的磁场。如果改变线圈间的距离，磁场将如何变化？如果两个线圈的半径不同，它们之间的磁场又将如何变化？编写相应的程序用动画表示这些变化。

6. 杂志SIAM News的2002年1/2期上刊登了牛津大学Nick Trefethen教授的一篇文章，题目是“A Hundred-dollar, Hundred-digit Challenge(100美元——100位数字的挑战)”。Trefethen教授所说的挑战包括10个数值计算问题，每个问题的答案都是一个实数。他要求每个答案的前10位有效数字，而计算出最多正确位数的人或研究组将得到100美元的奖金。来自25个国家的94支团队参加了这项计算工作，结果大大出乎Trefethen教授的预料。20支队伍得到了完美的100分，另外还有5支队伍得了99分。最近，一本相关的后续书（F.Bornemann, D.Laurie, S.Wagon, and J. Waldvogel, The SIAM 100-digit Challenge: A study in High-Accuracy Numerical Computing, SIAM, Philadelphia, 2004）已经出版。

Trefethen教授的第一个问题是计算

$$T=\lim_{t\to 0}\int_t^1 x^{-1}\cos(x^{-1}\log x)\mathrm{d}x$$

(1) 为什么我们不能简单地写几行程序，用某个MATLAB中的数值积分指令来计算这个积分？

下面是能计算出T的若干位有效数字的一种方法。将要求的程序表示为区间上的积分的无限和的形式，且每个区间上，被积函数的正负号不发生变化：

$$T=\sum_{k=1}^{\infty}T_k$$

其中

$$T_k = \int_{x_k}^{x_{k-1}} x^{-1} \cos x^{-1} \log x \mathrm{d}x$$

这里$x_0 = 1$,并且对$k > 0, x_k$为方程$\cos(x^{-1}\log x) = 0$顺序排列的根，其顺序为$x_1 > x_2 > \cdots$。换句话说，对$k > 0, x_k$满足方程

$$\frac{\log x_k}{x_k} = -\left(k - \frac{1}{2}\right)\pi$$

你可以用fzero来计算零点x_k。如果能使用符号工具箱，也可以用指令 lambertw 计算出x_k。对每个x_k ,可用quad或quadl计算数值积分T_k。由于T_k 为交替的正数和负数，所以这个级数的部分项和可能大于或者小于无限和的极限值。而且，相比于单个部分项和，两个相继部分项和的平均值更接近于最终结果。

(2) 在可接受的时间内,用这个方法尽可能准确地计算出T。尝试算出至少 4 位或 5 位准确的有效数字，也可以算出更多位。对于每个结果，估计出它的准确性。

(3) 研究在上述计算中，如何使用艾特肯(Aitken)的Δ^2加速方法：

$$\tilde{T} = T_k - \frac{(T_{k+1} - T_k)^2}{T_{k+1} - 2T_k + T_{k-1}}$$

注:这里的log是计算自然对数。

第四章　数据处理

§4.1　插　　值

假如通过实验与数值计算,得到未知函数$f(x)$的一些离散值,希望用已知函数$g(x)$逼近函数$f(x)$，这种函数逼近问题叫插值问题，$g(x)$称为插值函数。由于$g(x)$的选择不同，就产生不同的插值法。$g(x)$可以是代数多项式，三角多项式或有理函数等。

§4.1.1　插值分类

插值分类可以按插值范围分类或按插值方法分类。按插值范围分为内插与外插：

- 内插　在已知数据点范围之内插入数据。
- 外插（外推）　在已知数据点的范围之外插入数据。

按插值方法分为：

- 多项式插值　二点可以连成一条直线，三点可以确定一条抛物线，一般来说，$n+1$个数据点可以唯一地确定一个不高于n次的代数多项式，使其代表的曲线通过这$n+1$个数据点。

  插值多项式有：拉格朗日插值多项式，牛顿插值多项式，埃尔米特插值多项式等。
- 分段插值　如果数据点数目多(比如大于5)，得到的高次的插值多项式可能会与被插值的函数有很大的偏离。这时，可以将插值区间分成若干小区间，在每个小区间插入一个低次的多项式，称之为分段插值。
- 样条插值　分段插值生成的曲线在小区间内是光滑的，但在小区间的连接点则不一定光滑，即其导数不一定连续。样条插值法则要求生成的曲线在两个小区间的连接处也光滑。
- 曲线拟合　如果考虑到测量数据总会有误差，有时候并不要求插值函数通过数据点，只要求插值函数所代表的曲线按照某个判别标准（如最小二乘法，要求$\sum\limits_{i=0}^{n}[f_i-g(x_i)]^2$ 为最小）能最好地接近数据点，则所用方法叫曲线拟合。

§4.1.2 拉格朗日插值法

过两点$(x_0, y_0), (x_1, y_1)$的直线为

$$y = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$

将两点的坐标$(x_0, y_0), (x_1, y_1)$代入上式即可验证。上式改写成

$$L_1(x) = l_0(x) y_0 + l_1(x) y_1$$

其中

$$L_1(x) = y, \quad l_0 = \frac{x - x_1}{x_0 - x_1}, \quad l_1 = \frac{x - x_0}{x_1 - x_0}$$

$l_0(x), l_1(x)$显然是满足下列条件的一次函数

$$l_0(x_0) = 1, \qquad l_0(x_1) = 0$$

$$l_1(x_0) = 0, \qquad l_1(x_1) = 1$$

$l_0(x), l_1(x)$分别是对应节点x_0, x_1的线性插值基函数。这表示插值函数可以视为两个插值基函数的线性组合，其组合系数是对应点上的函数值y_0, y_1。这种形式的插值称为拉格朗日插值。

类似的方法可以求出二次插值函数。令

$$L_2(x) = l_0(x) y_0 + l_1(x) y_1 + l_2(x) y_2$$

$l_0(x), l_1(x), l_2(x)$ 分别是节点y_0, y_1, y_2的插值基函数，它们应该是不超过二次的多项式, 且满足条件

$$l_0(x_0) = 1, \quad l_0(x_1) = 0, \quad l_0(x_2) = 0$$

$$l_1(x_0) = 0, \quad l_1(x_1) = 1, \quad l_1(x_2) = 0$$

$$l_2(x_0) = 0, \quad l_2(x_1) = 0, \quad l_2(x_2) = 1$$

也就是

$$l_k(x_i) = \delta_{ik} \qquad (i, k = 0, 1, 2)$$

其中

$$\delta_{ik} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}$$

基函数可以这样构造，对于$l_0(x)$，因为x_1, x_2是其零点，所以$l_0(x)$应该含有因子$(x-x_1)(x-x_2)$，又因为$l_0(x)$是二次多项式，故可写成

$$l_0(x) = A(x-x_1)(x-x_2)$$

式中A为待定的系数，再由条件$l_0(x_0)=1$定出

$$A = \frac{1}{(x_0-x_1)(x_0-x_2)}$$

于是

$$l_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$$

同样可求得

$$l_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

最后得到

$$L_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2$$

显然，$L_2(x)$是一个不超过二次的多项式，且满足插值条件

$$L_2(x_0)=y_0,\ L_2(x_1)=y_1,\ L_2(x_2)=y_2$$

一般来说，给定函数$y=f(x)$在$n+1$个节点的函数值，可以求出一个次数不超过n次的插值多项式

$$L_n(x) = \sum_{k=0}^{n} l_k(x)y_k$$

$$l_k(x) = \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

且满足插值条件

$$L_k(x_i) = \delta_{ik} \quad (i,k=0,1,2,\cdots,n)$$

按照这种方法编写的程序为

```
function v = polyinterp(x,y,u)
n = length(x);
v = zeros(size(u));
for k = 1:n
```

```
    w = ones(size(u));
    for j = [1:k-1 k+1:n]
        w = (u-x(j))./(x(k)-x(j)).*w;
    end
    v = v + w*y(k);
end
```

程序中的x,y为已知的数据,u是要插入的数据。下面是一个应用这个程序的实例:

```
x = 1:6;
y = [16,18,21,17,15,12];
u= 0.75:0.05:6.25;
v=polyinterp(x,y,u)
plot(x,y,':',u,v,'-')
```

画出的图形如图4.1所示,圆圈是原有的数据点,实线是插值以后的曲线,也就是拉格朗日插值所得的曲线;虚线是直接将数据连成的折线,也叫分段线性插值法所得的结果。

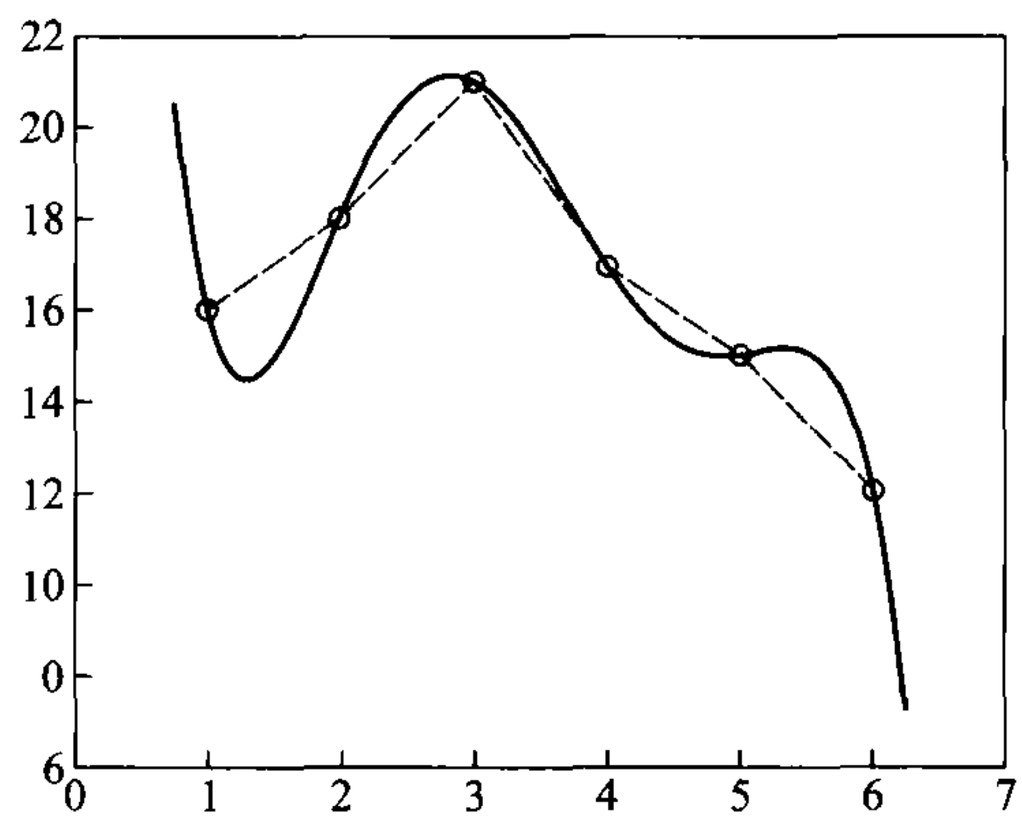


图 4.1　分段线性插值法与拉格朗日插值法的对比

显然,这两种方法的结果差距很大。实际上这是插值方法的两个极端,拉格朗日插值法得到的曲线有无限多阶导数,而分段线性插值法所得的折线是不光滑的,折线在数据点处发生转折。所以在实际使用中需要对它们进行改进。改进的思路有两个出发点,一个是将数据分成小区域进行插值,也就是常说的分段插值,另一个是设法使分段插值后的曲线在连接处保持一定的光滑,也就是使连接起来的曲线存在一阶甚至二阶导数,这就是埃尔米特插值法。根据对导数的限制条件不同,又有不同做法。在MATLAB中使用了pchip和spline两种方法。

§4.1.3 分段三次埃尔米特插值

将插值区间分为n个小区间，考虑第k个小区间，令区间长度为

$$h_k = x_{k+1} - x_k, \quad x_k \leqslant x \leqslant x_{k+1}$$

如果在区间内插入一个三次多项式

$$P(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

就需要4个条件来确定这4个系数。除了已知的小区间两个端点的函数值y_k, y_{k+1}外，还需要两个条件。假定区间端点的一阶导数是d_k, d_{k+1}，那么就可以用y_k, y_{k+1}, d_k，d_{k+1} 来表示这个多项式，其形式为

$$P(x) = \frac{3hs^2 - 2s^3}{h^3}y_{k+1} + \frac{h^3 - 3hs^2 + 2s^3}{h^3}y_k$$
$$+\frac{s^2(s-h)}{h^2}d_{k+1} + \frac{s(s-h)^2}{h^2}d_k$$

其中引入了区间内的局部变量$s = x - x_k$，并且令$h = h_k$。当$x = x_k$时有$s = 0$，当$x = x_{k+1}$ 时有$s = h$。所以它显然满足条件

$$P(x_k) = y_k, \quad P(x_{k+1}) = y_{k+1}$$
$$P'(x_k) = d_k, \quad P'(x_{k+1}) = d_{k+1}$$

这个表达式的推导在数值计算方法的教材中都有介绍，它也是通过构造插值基函数来求埃尔米特插值多项式，这里不再详述。

由于端点的导数值并不是已知的数据，所以就可以对它进行选择。如果要求相邻区间端点的一阶导数值相等，就可以保证插值后的曲线的一阶导数连续，这就是MATLAB中保形分段三次插值（pchip）的做法，pchip是“分段三次埃尔米特插值多项式(piecewise cubic Hermite interpolating polynomial)”的英文首字母缩写。如果在选择时，要求相邻区间的一阶和二阶导数都相等，那就是MATLAB中三次样条(cubic spline)插值函数。读者如果想了解具体的选择方法，可参阅MATLAB软件之父Cleve B. Moler所著的《MATLAB数值计算》。

这两种插值方法的区别可以用图4.2来表示。

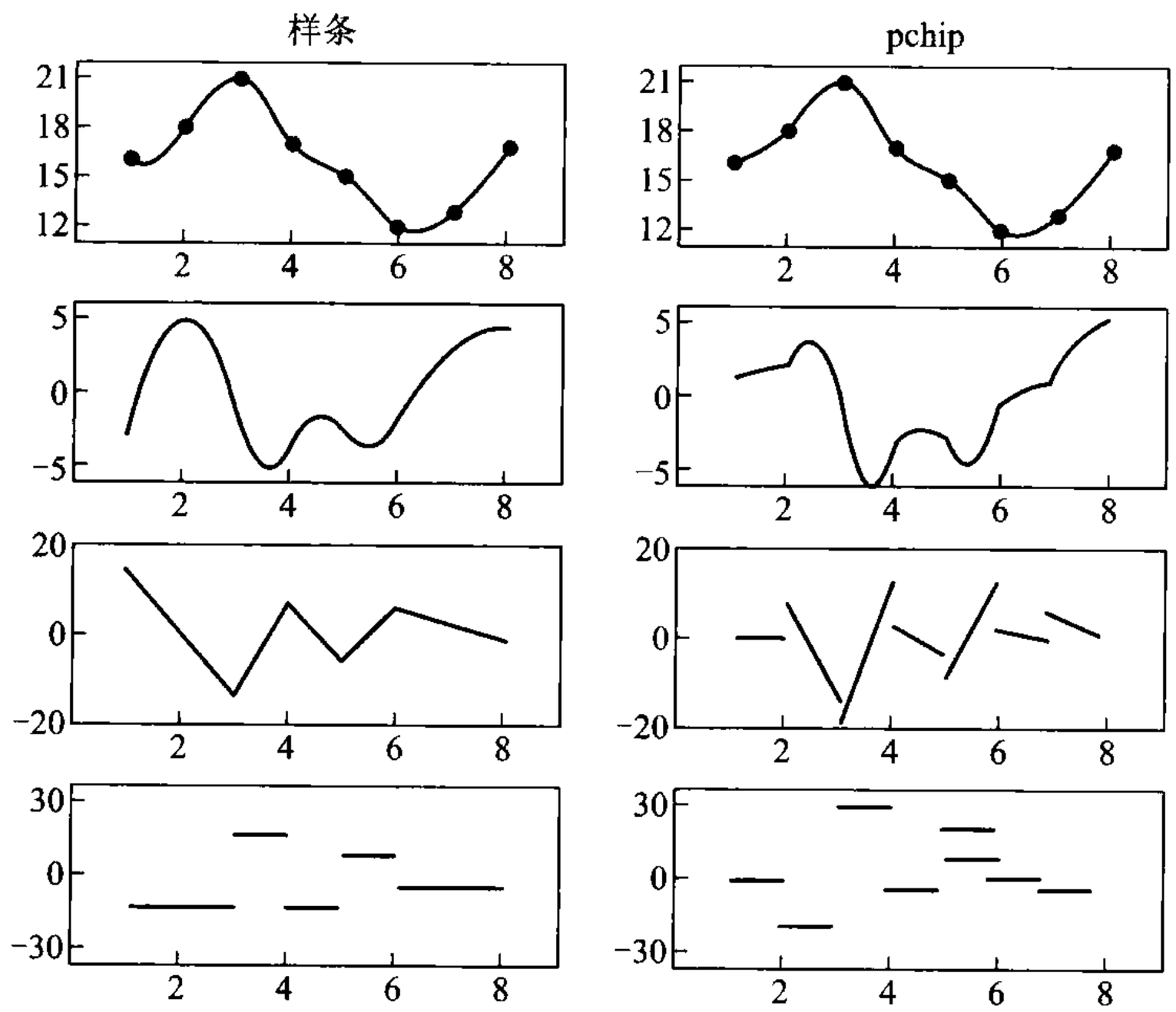


图 4.2　样条插值函数与pchip插值函数以及它们的一、二、三阶导数

§4.1.4　MATLAB插值指令

前面介绍的插值方法都可以用MATLAB的插值指令来实现，插值指令根据输入的参数来决定所使用的插值方法。其中的nearest方法是一种少见的方法，它在区间中以最接近的数据点的值作为插值的选择，所以图4.3(a) 中得到的插值曲线都是一些平行于横轴的线段。更方便的是，这些插值指令可以进行二维、三维以至更高维的插值。

具体指令为：

interp1(X,Y,XI,'method')	一维插值
interp1q(X,Y,XI)	快速一维插值
interp2(X,Y,Z,XI,YI)	二维插值
interp3(X,Y,Z,V,XI,YI,ZI)	三维插值指令
interpn(X1,X2,X3,...,V,Y1,Y2,Y3,...)	N维插值指令

各项参数的含义为：

X,Y,Z,V	已知的数据点与函数值
XI,YI,ZI	插入的数据点
method	选用的插值计算方法，含以下几种

nearest	最近邻插值
linear	线性插值
spline	分段三次样条插值
pchip	保形分段三次插值
cubic	与pchip相同
v5cubic	来自MATLAB5.0 版本的cubic插值法

例1 以正弦函数的10个值作为原始数据（小圆圈表示）。再用四种方法算出41个插值点并画曲线(如图4.3)。

```
x=0:10; y=sin(x);                                  %原始数据
xi=0:.15:10;                                       %插值点
yi1=interp1(x,y,xi,'nearest');
yi2=interp1(x,y,xi,'linear');
yi3=interp1(x,y,xi,'spline');
yi4=interp1(x,y,xi,'cubic');
subplot(2,2,1)
plot(xi,yi1,xi,yi1,'r.')
subplot(2,2,2)
plot(xi,yi2,xi,yi2,'r.')
subplot(2,2,3)
plot(xi,yi3,xi,yi3,'r.')
subplot(2,2,4)
plot(xi,yi4,xi,yi4,'r.')
```

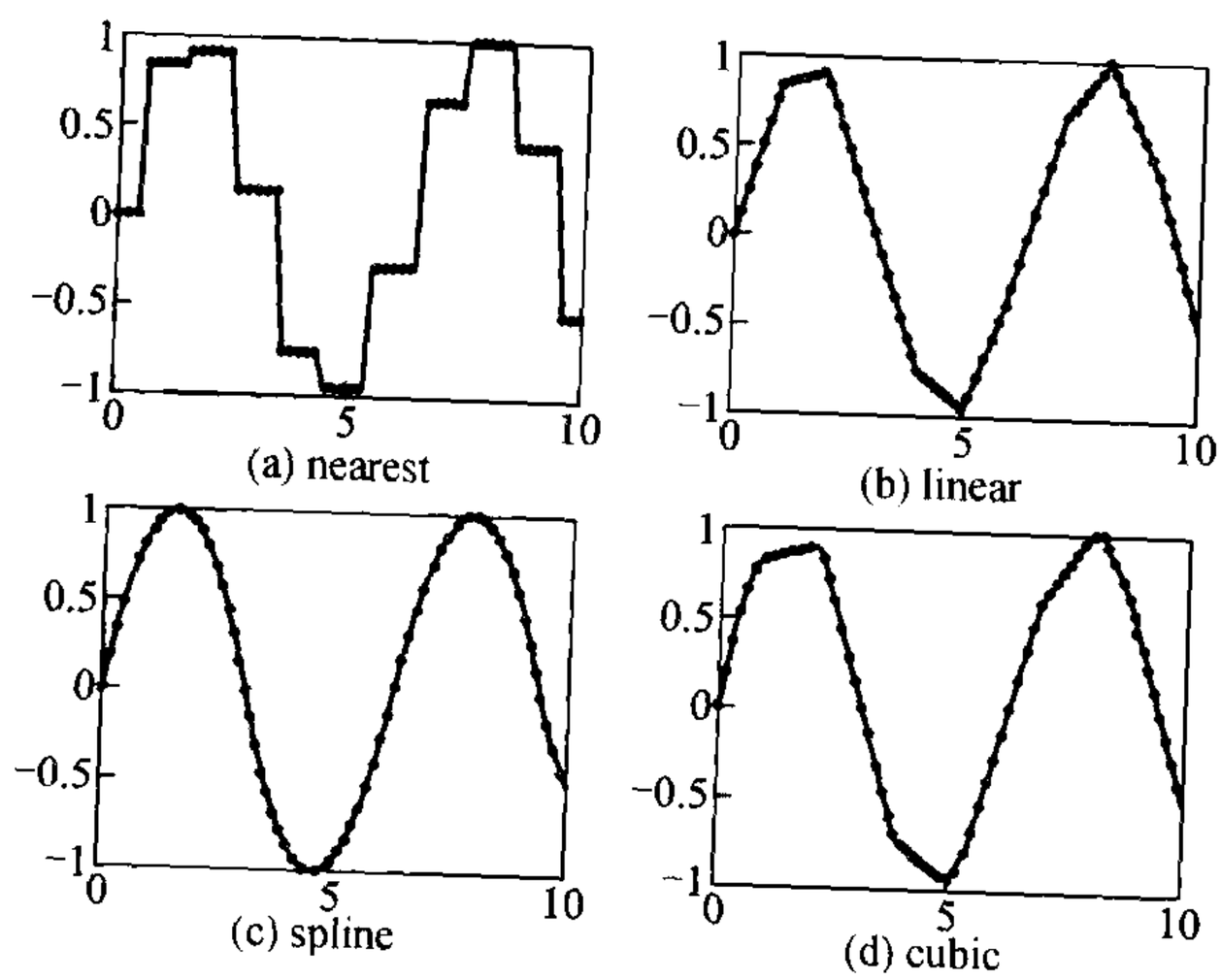


图 4.3 四种插值指令的应用

§4.2 曲线拟合

§4.2.1 曲线拟合的最小二乘法

曲线拟合也叫函数逼近，是另一类计算插值函数的方法。它只要求插值函数能合理地反映数据的基本趋势，并不要求函数能通过数据点。所以在前面所介绍的插值方法中，函数与数据点的偏差为零，而在曲线似合中，函数与数据点则可能存在偏差，因此可以用这个偏差来反映函数的逼近程度。

设在曲线拟合中，插值函数值$\varphi(x_i)$与n个原始数据点y_i的偏差为

$$\delta_i = \varphi(x_i) - y_i \qquad (i = 0, 1, 2, \cdots, n)$$

那么可以有以下几种判别准则：

(1) 使偏差的绝对值之和

$$\sum_{i=0}^{n} |\delta_i|$$

为最小；

(2) 使偏差的最大绝对值

$$\max_i |\delta_i|$$

为最小;

(3) 使偏差的平方和

$$\sum_{i=0}^{n} \delta_i^2$$

为最小。如果考虑到数据在精度上的差异，可以加上一定的权重因子，即要求加权的偏差的平方和

$$\sum_{i=0}^{n} \lambda_i \delta_i^2$$

为最小。

准则(1)的提出很自然也很合理，但使用不便。准则(2)所用的方法称为函数的最佳一致逼近。准则(3) 所用的方法称最佳平方逼近，也叫曲线拟合的最小二乘法，是最常用的方法。

§4.2.2 多项式拟合

设所求的插值函数$\varphi(x)$是一个m次多项式$P_m(x)$ $(m<n)$,

$$P_m(x)=a_0+a_1x+\cdots+a_mx^m=\sum_{j=0}^{m}a_jx^j$$

通过选取多项式的系数a_j，使得在x_i,y_i点有

$$\sum_{i=1}^{n}\delta_i^2=\sum_{i=1}^{n}[y_i-P_m(x_i)]^2=F(a_0,a_1,\cdots,a_m)$$

最小。由多元函数取极值的条件得方程组

$$\frac{\partial F}{\partial a_j}=-2\sum_{i=1}^{n}\left[y_i-\sum_{k=0}^{m}a_kx_i^k\right]x_i^j=0,$$

移项得

$$\sum_{k=0}^{m}a_k\left(\sum_{i=1}^{n}x_i^{k+j}\right)=\sum_{i=1}^{n}y_ix_i^j$$

即

$$\begin{cases}na_0+a_1\sum\limits_{i=1}^{n}x_i+a_2\sum\limits_{i=1}^{n}x_i^2+\cdots+a_m\sum\limits_{i=1}^{n}x_i^m=\sum\limits_{i=1}^{n}y_i\\ a_0\sum\limits_{i=1}^{n}x_i+a_1\sum\limits_{i=1}^{n}x_i^2+a_2\sum\limits_{i=1}^{n}x_i^3+\cdots+a_m\sum\limits_{i=1}^{n}x_i^{m+1}=\sum\limits_{i=1}^{n}y_ix_i\\ \cdots\cdots\cdots\cdots\\ a_0\sum\limits_{i=1}^{n}x_i^m+a_1\sum\limits_{i=1}^{n}x_i^{m+1}+a_2\sum\limits_{i=1}^{n}x_i^{m+2}+\cdots+a_m\sum\limits_{i=1}^{n}x_i^{2m}=\sum\limits_{i=1}^{n}y_ix_i^m\end{cases}$$

这是最小二乘拟合多项式的系数a_k应满足的方程组，称为正则方程组。

当最高幂次为1时就是最小二乘法的拟合直线，正则方程组简化为

$$\begin{cases}na_0+a_1\sum\limits_{i=1}^{n}x_i=\sum\limits_{i=1}^{n}y_i\\ a_0\sum\limits_{i=1}^{n}x_i+a_1\sum\limits_{i=1}^{n}x_i^2=\sum\limits_{i=1}^{n}y_ix_i\end{cases}$$

§4.2.3 多项式拟合的指令polyfit

MATLAB已经将多项式拟合开发成专用的指令，在指令的程序中使用了很先进的算法来求解正则方程。

polyfit (X,Y,N)对数据作最小二乘法的多项式拟合

polyval(P,X)　计算多项式的值

X,Y　数据点及其函数值

N　拟合多项式的最高幂次，返回值为按降幂排列的多项式系数

P　被求值的多项式系数，须按降幂排列

下面的例子是给定数据以后用polyfit进行多项式拟合（如图4.4）。

```
t=0:0.5:5;
s=1*t+0.2.*t.*t;                                    %给定二次曲线
s1=[0.5  -0.18  -0.01  0.13  0.1 ...
 0.31  -0.22  -0.31  0.2  0.4  -0.14];
ss=s+s1;                                            %设定误差值
P=polyfit(t,ss,2);                                  %二次曲线拟合
y=polyval(P,t);                                     %计算拟合函数的值
plot(t,y,t,ss,'r*')
```

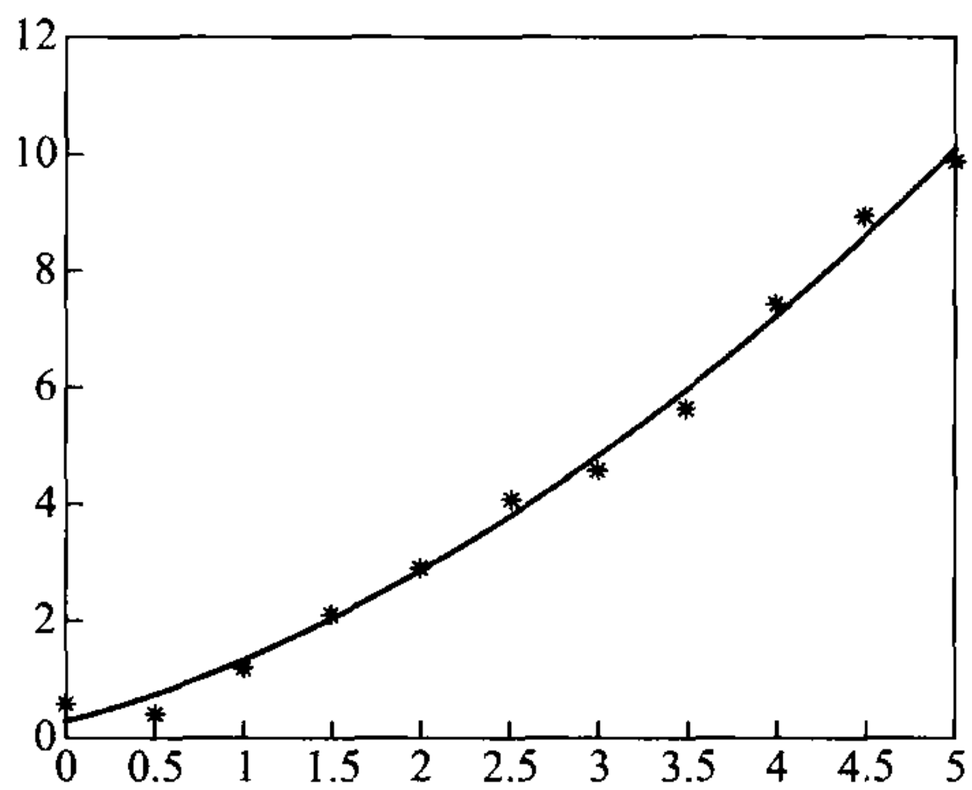


图 4.4　多项式拟合的曲线

例2　求下面数据表的最小二乘二次拟合多项式。

i	1	2	3	4	5	6	7	8	9
x_i	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	1
y_i	-0.2209	0.3295	0.8826	1.4392	2.0003	2.5645	3.1334	3.7601	4.2836

设所求的最小二乘二次拟合多项式是

$$P_2(x) = a_0 + a_1x + a_2x^2$$

此问题的正则方程组是

$$\begin{cases} 9a_0 + 0 + 3.75a_2 = 18.1732 \\ 0 + 3.75a_1 + 0 = 8.4842 \\ 3.75a_0 + 0 + 2.7656a_2 = 7.6173 \end{cases}$$

其解为$a_0 = 2.0034$, $a_1 = 2.2625$, $a_3 = 0.0378$，所求多项式为

$$P_2(x) = 0.0378x^2 + 2.2625x + 2.0034$$

用MATLAB验证结果

```
x=[-1 -0.75 -0.5 -0.25 0 0.25 0.5  0.75  1] ;
y=[-0.2209 0.3295 0.8826 1.4392 2.0003 2.5645 3.1334 3.7601
   4.2836];
yi=polyfit(x,y,2)
yi =  0.0379    2.2624    2.0034
```

§4.2.4 图形窗口的曲线拟合功能

在图形窗口中可以用菜单方式对数据进行拟合。它具有拟合快速，操作简便的优势，它的功能包括

- 数据的拟合包括样条拟合，保形拟合以及从1到10次的多项式拟合。
- 利用一组数据可以同时作出多种拟合曲线。
- 可以绘制残差图。
- 可以显示拟合曲线的表达式。
- 可用拟合方程和标准差来注释图形。
- 可以保存拟合曲线和计算结果。

下面用一个例子来讲解其用法。在指令窗口输入数据并画图

```
>> x = [1,2,3];  y = [7,3,9];  plot(x,y,'o')
```

然后在窗口的tools菜单中单击basic fitting 菜单选项，打开对话框(如图4.5)。在其中选择“Center and scale X data”单选框，可对数据进行中心化和规范化，数据经过处理后可提高计算准确性。在Check to displayfits on figure的下面选linear(线性方程拟合)和quadratic(二次方程拟合)，再选下面的Show equations（显示拟合方程）。选项Significant digits(有效数字)默认是2位，不必改变。再选Plot

risiduals（画出残差图），默认方式是用分区图和直方图形式。最后点出右下方的向右的箭头，将显示数字结果与图形如图4.6。

在图形中，原始数据用圆圈标出，线性方程的拟合显然不如二次方程的拟合效果好。方程的表达式也给出了，是

$$y = z + 6.3$$

$$y = 5z^2 + z + 3$$

其中$z = (x-2)/1$是经过中心化和规范化的变量。

这里给用户提供的拟合方法包括了前面所介绍的几种。如果我们再点击 Tools 下面的 Data statistics 就可以对数据进行统计分析，包括计算最大值、最小值、平均值、中值、标准差等。

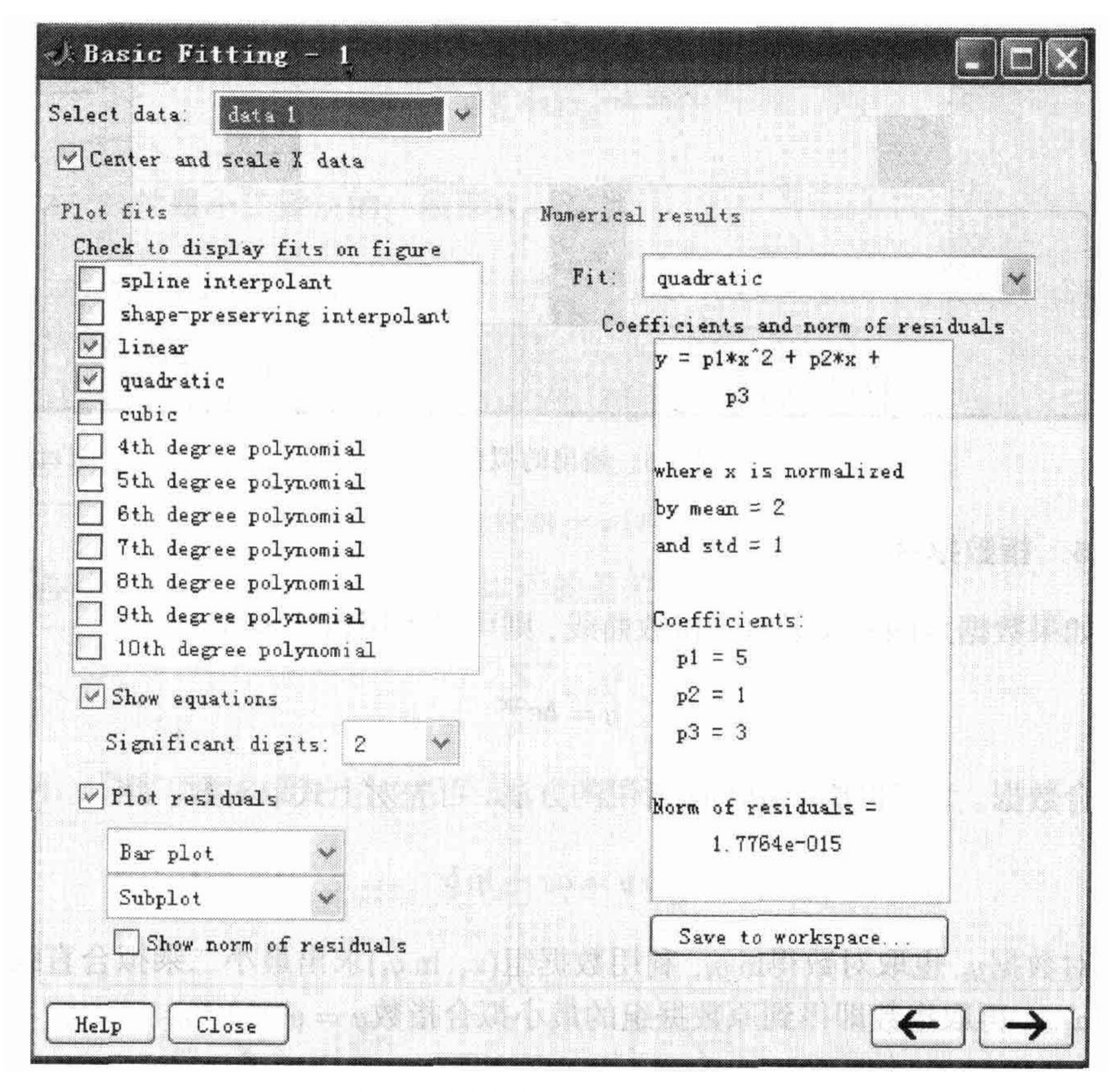


图 4.5 图形窗口的拟合功能演示

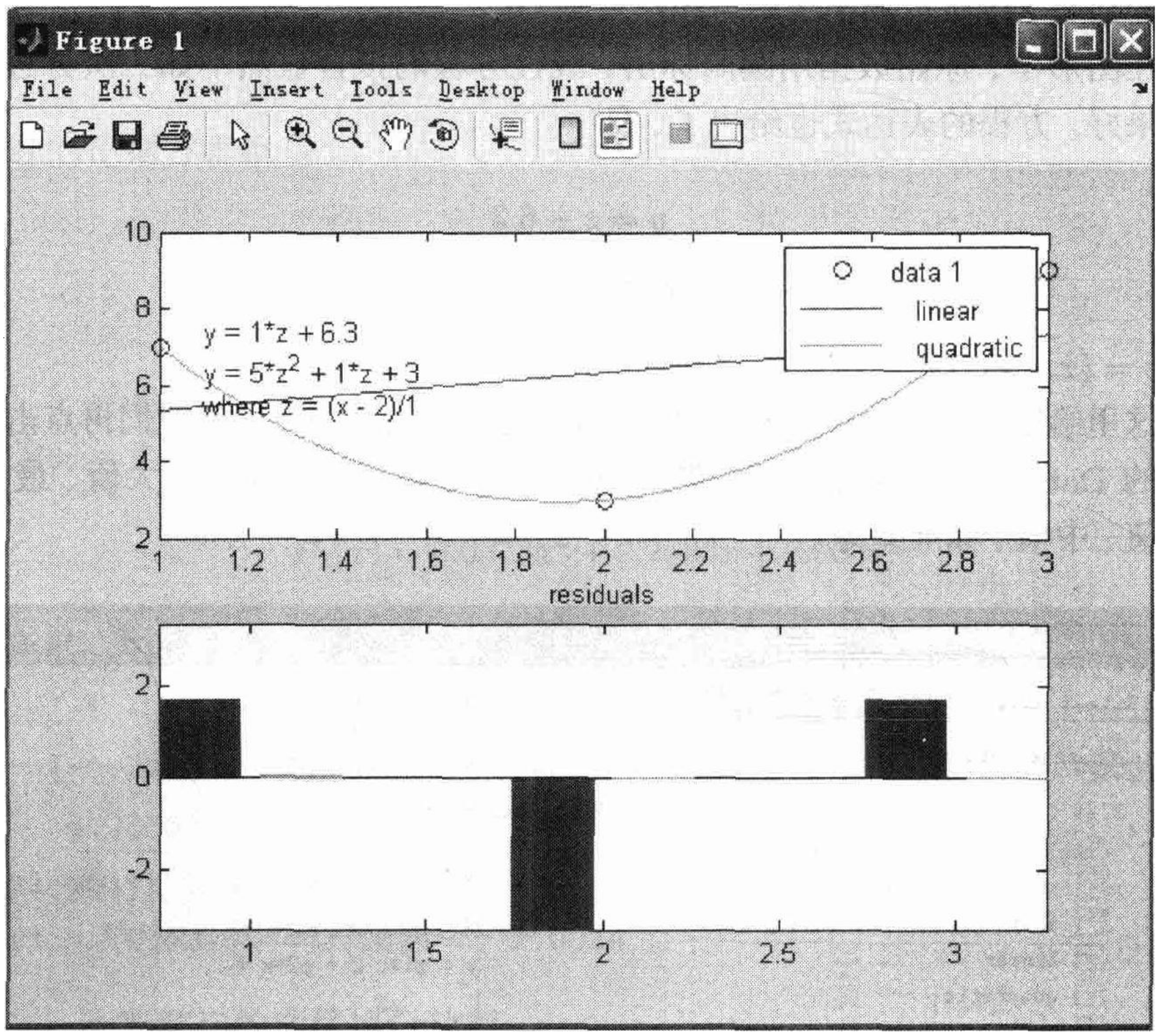


图 4.6 输出的拟合结果

§4.2.5 指数拟合

如果数据点的分布近似于指数曲线，则可考虑用指数函数

$$y = b\mathrm{e}^{ax}$$

去拟合数据。为了也能应用上面所说的方法，可先对上式取对数，得

$$\ln y = ax + \ln b$$

同时对数据y_i 也取对数得$\ln y_i$，利用数据组$(x_i, \ln y_i)$求出最小二乘拟合直线$\bar{y} = a_0 + a_1 x$，再取指数即得到原数据组的最小拟合指数$y = \mathrm{e}^{a_0 + a_1 x}$。

例3 设一发射源的发射强度公式具有指数形式$I = I_0\mathrm{e}^{-\alpha t}$。现有一组观测数据如下：

t_i	0.2	0.3	0.4	0.5	0.6	0.7	0.8
I_i	3.16	2.38	1.75	1.34	1.00	0.74	0.56

试用最小二乘法确定I_0与α。

全部求解过程都用MATLAB完成，操作步骤如下：

```
>> x=[0.2, 0.3,0.4,0.5,0.6,0.7,0.8];
>> I=[3.16,2.38 ,1.75,1.34,1.00,0.74,0.56];
>> y=log(I);
>> polyfit(x,y,1)
ans =
      -2.8883        1.7283
>> I0=exp(1.7283)
I0 = 5.6311
>> alpha=-(-2.8883)
alpha = 2.8883
```

得强度公式为

$$I = 5.6311\mathrm{e}^{-2.8883t}$$

§4.2.6 线性最小二乘法的一般形式

已知n个数据点(x_i, y_i)，取$[a,b]$上线性无关的函数组$\varphi_0(x), \varphi_1(x), \cdots, \varphi_m(x)$构成拟合函数

$$\varphi(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \cdots + a_m\varphi_m(x)$$

记偏差为

$$\delta_i = y_i - \varphi(x_i)$$

要求通过系数a_i的选择，使得加权以后的偏差的平方和

$$\sum_{i=1}^{n} \omega_i \delta_i^2$$

为最小。与多项式拟合的讨论相类似，上述问题的正则方程组为

$$\sum_{i=1}^{n} \omega_i [y_i - \varphi(x_i)]\varphi_j(x_i) = 0 \qquad (j = 0, 1, 2, \cdots, m)$$

即

$$\sum_{k=0}^{m} a_k \left[\sum_{i=1}^{n} \omega_i \varphi_k(x_i)\varphi_j(x_i)\right] = \sum_{i=1}^{n} \omega_i y_i \varphi_j(x_i)$$

如果记

$$(\varphi_k, \varphi_j) = \sum_{i=1}^{n} \omega_i \varphi_k(x_i)\varphi_j(x_i)$$

$$(y,\varphi_j)=\sum_{i=1}^{n}\omega_i y_i\varphi_j(x_i)$$

则方程组可以写成矩阵形式

$$\begin{pmatrix}(\varphi_0,\varphi_0) & (\varphi_0,\varphi_1) & \cdots & (\varphi_0,\varphi_m)\\ (\varphi_1,\varphi_0) & (\varphi_1,\varphi_1) & \cdots & (\varphi_1,\varphi_m)\\ \vdots & \vdots & & \vdots\\ (\varphi_m,\varphi_0) & (\varphi_m,\varphi_1) & \cdots & (\varphi_m,\varphi_m)\end{pmatrix}\begin{pmatrix}a_0\\ a_1\\ \vdots\\ a_m\end{pmatrix}=\begin{pmatrix}(y,\varphi_0)\\ (y,\varphi_1)\\ \vdots\\ (y,\varphi_m)\end{pmatrix}$$

由$\varphi_j(x)$的线性无关性可导出上述方程组的系数矩阵非奇异，从而保证了方程组有唯一的解存在。

为了方便，通常选取$\varphi_j(x)$满足

$$(\varphi_j,\varphi_j)=\sum_{i=1}^{n}\omega_i\varphi_j^2(x_i)>0$$

$$(\varphi_k,\varphi_j)=0 \qquad (k\neq j)$$

这其实就是选取正交函数族。则方程组的系数矩阵成对角型，正则方程组很好解，其解为

$$a_j=\frac{(y,\varphi_j)}{(\varphi_j,\varphi_j)}=\frac{\sum_{i=1}^{n}\omega_i y(x_i)\varphi_j(x_i)}{\sum_{i=1}^{n}\omega_i\varphi_j^2(x_i)}$$

而最小二乘函数为

$$\varphi(x)=\sum_{j=0}^{m}a_j\varphi_j(x)=\sum_{j=0}^{m}\frac{(y,\varphi_j)}{(\varphi_j,\varphi_j)}\varphi_j(x_i)$$

§4.3 曲线拟合工具箱

在MATLAB中有三个工具箱与本章的内容相关，它们是：

曲线拟合工具箱：包括了各种常用的拟合方法，下面会详细介绍。

样条工具箱：提供了三次插值样条函数，分段多项式样条（piecewise polynomial）函数，B（basic spline function）样条函数以及其他一些操作。

最优化工具箱：收集了求解各类优化问题的程序，其内容涵盖线性规划，二次规划，非线性规划，最小二乘问题，非线性方程求解，多目标决策，最小值与最大值问题以及半无限问题等的优化问题。

本节简单介绍曲线拟合工具箱的一些基本用法。

曲线拟合工具箱的功能比图形窗口的曲线拟合功能更强大也更方便。用指令cftool打开曲线拟合工具箱操作界面如图4.7所示。

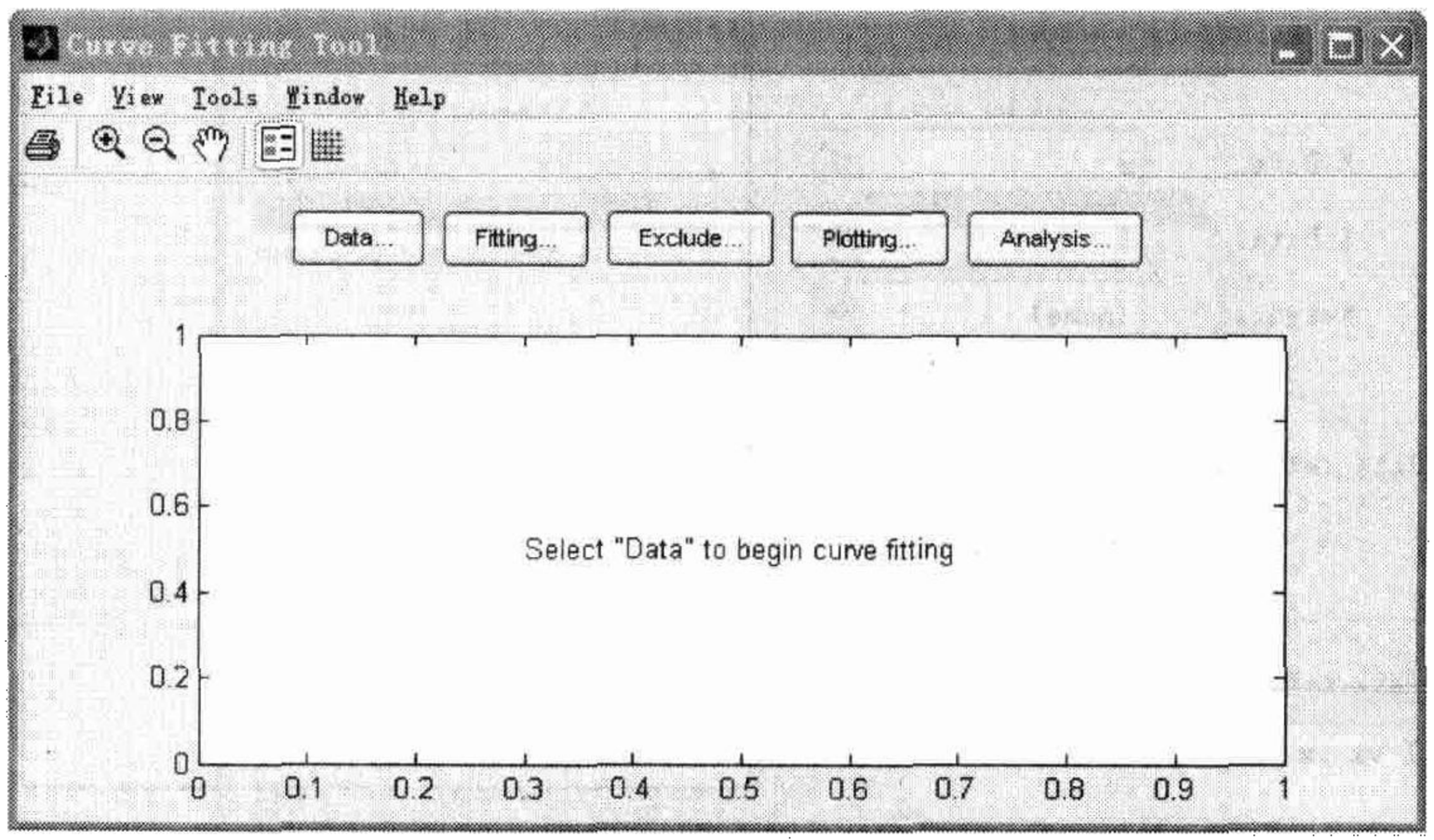


图 4.7 曲线拟合工具箱的工作界面

图形窗口中五个按钮的功能是：

- Data 按钮：可输出，查看和平滑数据；
- Fitting按钮：可拟合数据，比较拟合曲线和数据集；
- Exclude按钮：可从拟合曲线中排除特殊的数据点；
- Plotting按钮：在选定区间后，单击按钮，可以选择原始数据和拟合的数据作图；
- Analysis按钮：对拟合进行分析，如满意度、偏差分析等。

首先我们用曲线拟合工具箱来重做一遍前面的指数拟合的例3。步骤如下

在指令窗口输入数据

```
>> x=[0.2, 0.3,0.4,0.5,0.6,0.7,0.8];
>> I=[3.16,2.38 ,1.75,1.34,1.00,0.74,0.56];
```

打开曲线拟合窗口，点击Data按钮，打开对话框如图4.8，如图选择各个选项，就能在Preview下面的方框中的看到数据的散点图如图4.8右边所示。再在Data sets下面的框中选择I vs. x，然后选择框下面的View，就能看到数据列表如图4.9右边所示。

选择Fitting按钮，打开对话框，如图4.10选择各个选项，

从Results下面的框中可以看到拟合的指数形式是

$$f(x) = a * \exp(b * x), \qquad a = 5.636, \qquad b = -2.891$$

这个结果与例3相同，同时画出拟合曲线如图4.11。

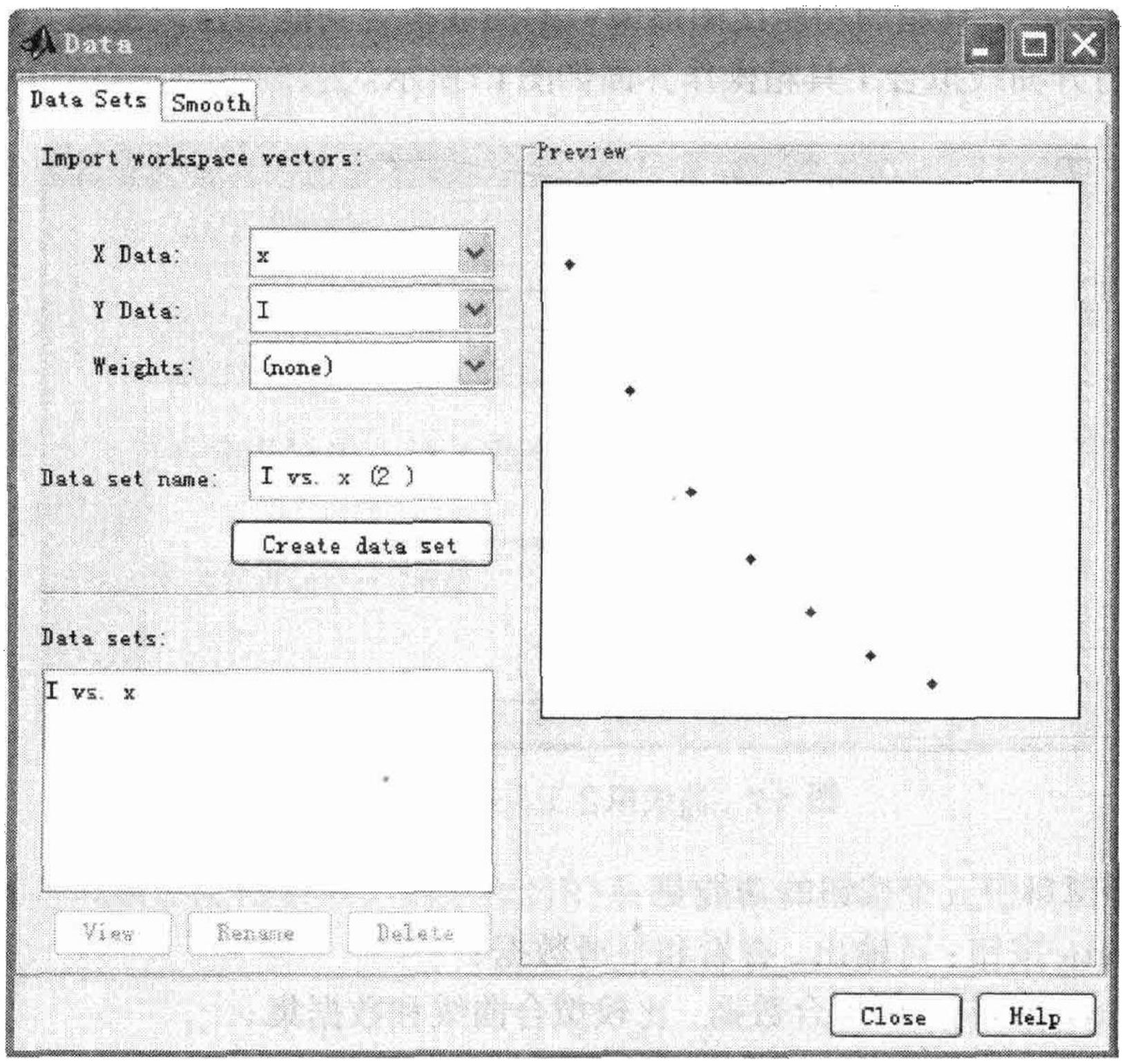
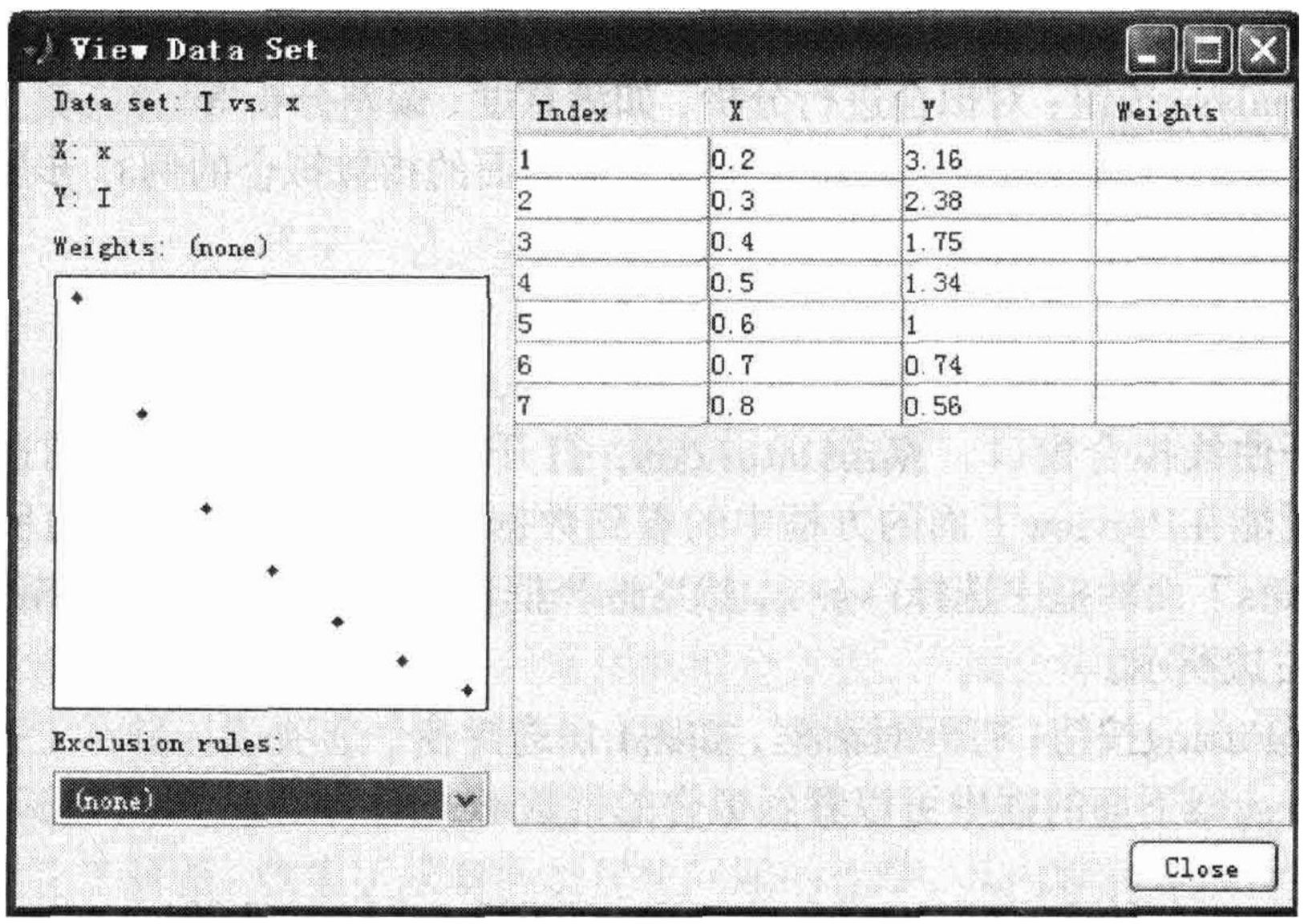


图 4.8 Data对话框



Index	X	Y	Weights
1	0.2	3.16	
2	0.3	2.38	
3	0.4	1.75	
4	0.5	1.34	
5	0.6	1	
6	0.7	0.74	
7	0.8	0.56	

图 4.9 散点图与数据表

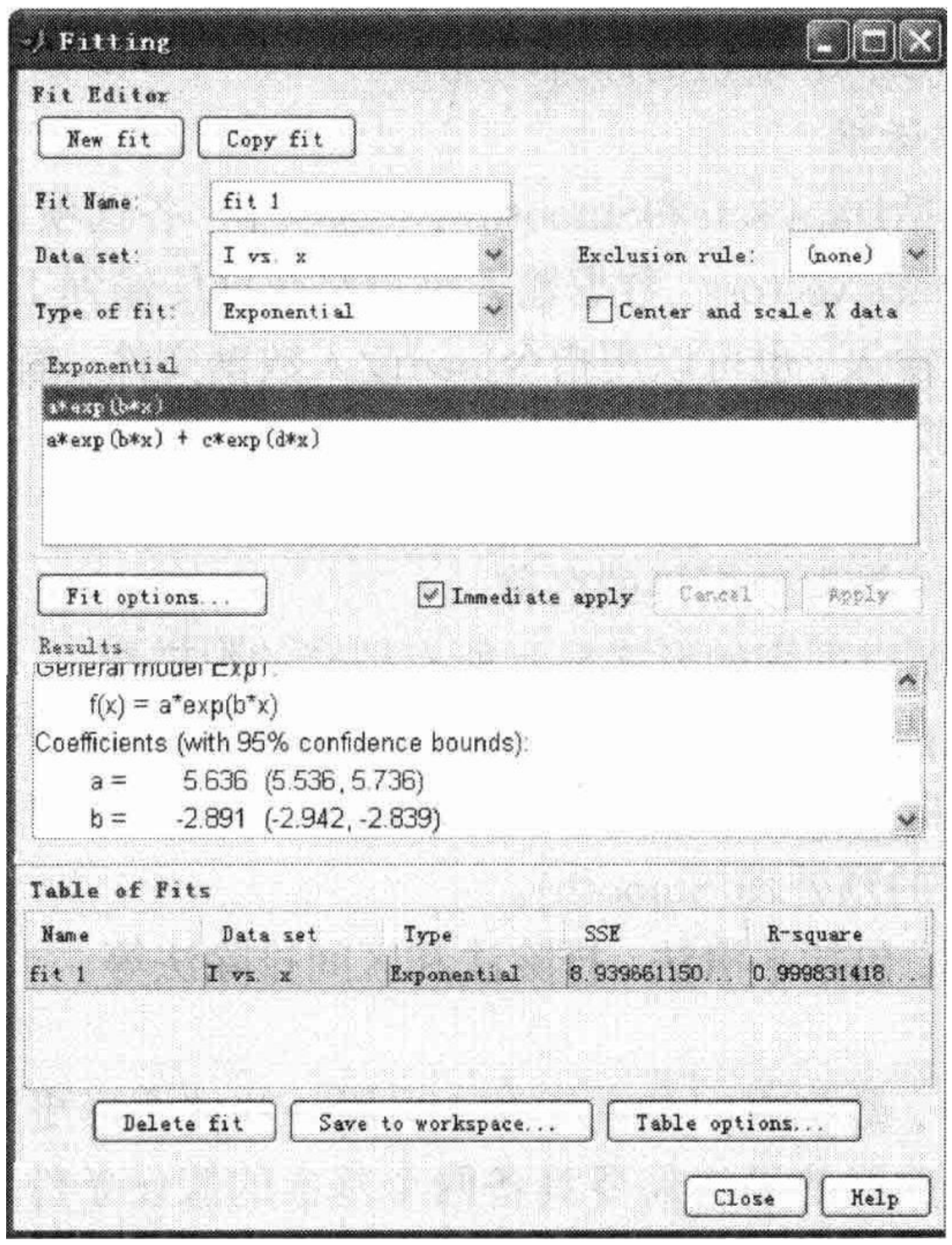


图 4.10 Fitting对话框

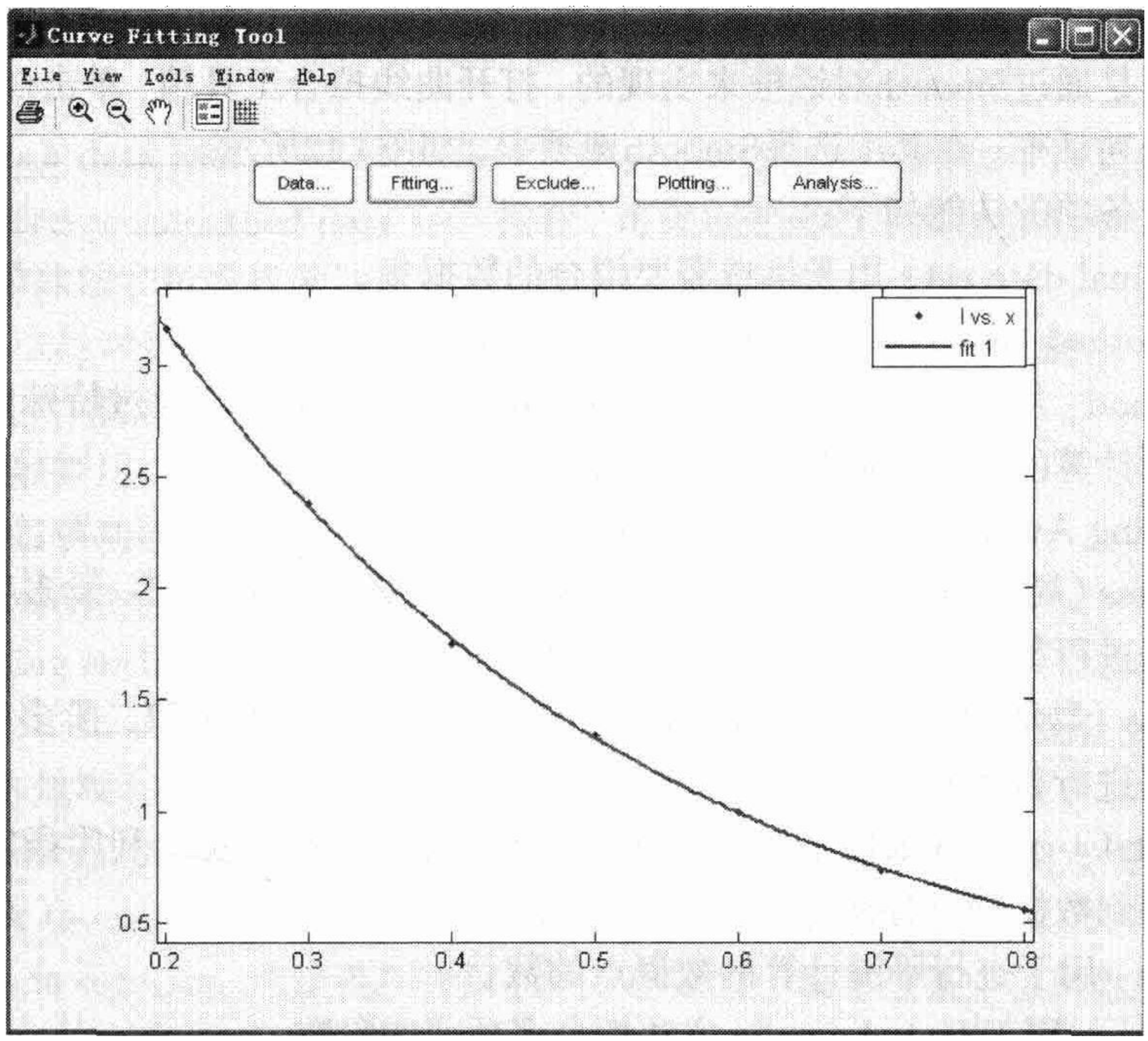


图 4.11 拟合的曲线

看过这个例子之后，对工具箱的用法已经基本了解，下面再介绍一下其中Data，Fittingt和Eclude按钮的详细功能。

1. Data按钮对话框

对话框里面包括Data sets和Smooth两个选项卡，各选项说明如下：

Import workspace vectors：数据输入是指将数据传输到工作内存中，可以在指令窗口通过键盘输入，也可以用指令load读入数据文件。输入后，将变量名放到X data，函数名放入Y data，权重放入Weights。

Preview：对所选向量进行图形化预览。

Data set name：设置数据集的名称。

Data sets：选项以列表的形式显示所有拟合的数据集。

在曲线拟合之前必须对数据进行预处理，去除异常值、不定值和重复值，以减少人为误差，提高拟合的精度。数据预处理的内容包括两部分，即数据输入查看(Preview)和数据的预处理(Smooth)。

数据的预处理，包括平滑法、排除法和区间排除法等。

(1) 平滑法

在处理数据时，数据有时杂乱无章，采用一些使图形更平滑的算法，可以提高的拟合效果。曲线的平滑需要具备两个基本的假设条件：预测数据和相应数据之间本质上是平滑的；曲线平滑的结果是得到平滑的数据，它应该最能反映原始数据的本质特征。曲线平滑的目的是试图估计每个响应数据的均值分布。平滑数据是通过Smooth对话框来实现的，打开曲线拟合工具箱，单击Data按钮，打开Data对话框，在其中选择Smooth选项卡，如图4.12所示。

其中各项的功能如下：

Original data set：用于挑选需要拟合的数据集。

Smoothed data set：用来平滑数据的名称。

Method：用于选择平滑数据的方法，每一个相应数据用通过特殊的曲线平滑方法所计算的结果来取代。平滑数据的方法包括：

Moving Average（用移动平均值进行交换）；

Lowess（局部加权散点图平滑数据，采用线性最小二乘法和一阶多项式拟合得到数据进行替换）；

Loess（局部加权散点图平滑数据，采用线性最小二乘法和二阶多项式拟合得到数据进行替换）；

Savitzky-golay（采用未加权的线性最小二乘法过滤数据，利用指定阶数的多项式得到数据进行替换）。

Span：用于进行平滑计算的数据点的数目。

Degree：用于Savitzky-golay方法拟合多项式的阶数。

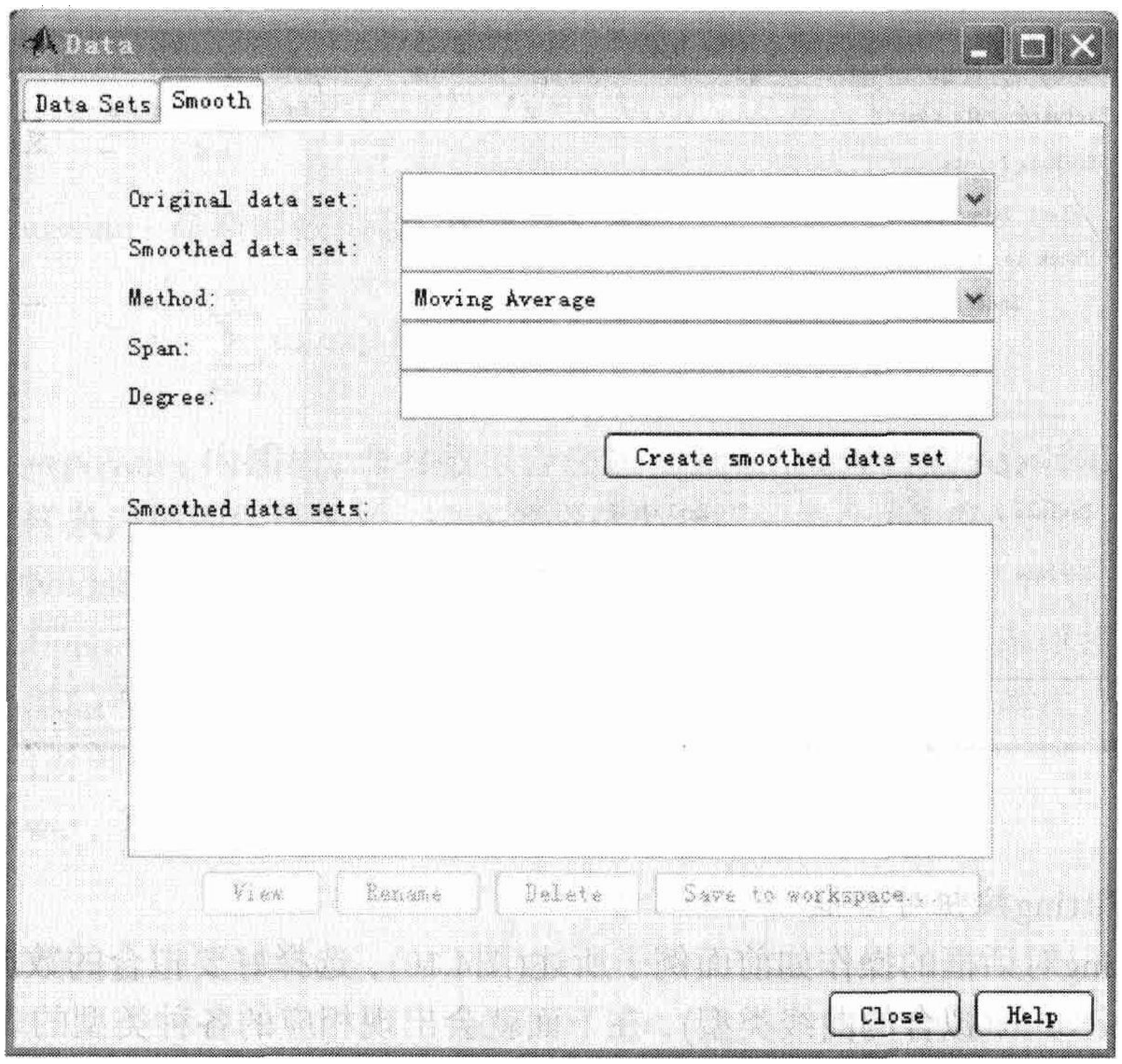


图 4.12 Smooth对话框

Smooth data sets：对于所有平滑数据进行列表。可以增加平滑数据集，通过单击“Create smoothed data set”按钮，可以创建经过平滑的数据集。

(2) 排除法和范围排除法

排除法是对数据中的异常值进行排除。范围排除法是在排除超出一定范围的数据，它可能是由于系统误差导致偏离正常值的异常值。

排除数据用Exclude对话框来实现。如图4.13所示：

其中各项功能如下：

Exclusion rule name：所使用的排除法则的名称。

Existing exclusion rules：现有的排除法则列表。

Select data set：选择需要排除的数据集。当选择好数据集以后就会出Exclude graphically选项和数据集列表。

Exclude graphically：允许以图形的形式除去异常值，排除个别的点用‘×’标记。

Check to exclude point：在数据表中打钩来选择要排除的数据。

Exclude sections：选定范围排除数据，比如大于或小于某个值的数据。

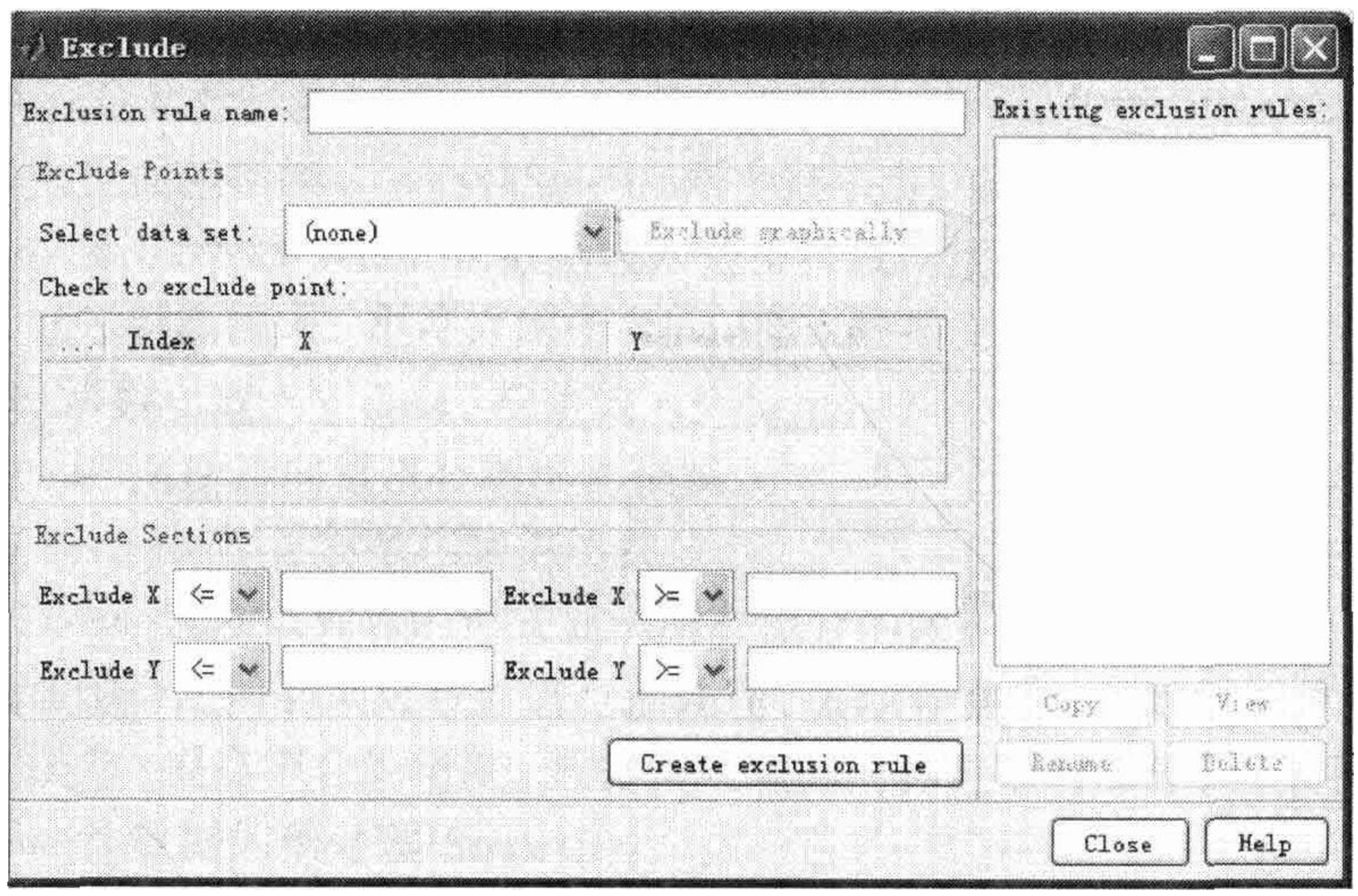


图 4.13 Exclude对话框

2. Fitting按钮对话框

Fitting对话框的操作如前面例子所述(图4.10)，选择好要拟合的数据集，再选择Type of fit(拟合的曲线类型)，在下面就会出现相应的各种类型的表达式或文字描述。选好以后点击Immediate apply即可。

在框中的Fit Editor下面的选项有：

New fit：开始进行新的曲线拟合，默认是用线性多项式拟合数据。

Copy fit：如果要选择不同的曲线拟合方法，可以用“Copy fit”保存已经拟合的结果。

Fit name：当前拟合曲线的名字。单击“New fit”按钮时系统会产生默认的文件名。

Data set：当前的数据集。

Exclusion rule：排除异常值的规则，在数据预处理前建立的文件名。

Center and scale X data：对数据进行中心化和规范化处理。

在Type of fit后面包括工具箱中的拟合函数的类型，具体有：

- Custom equations：自定义拟合的线性或非线性方程。
- New equations：使用“Custom equations”按钮前，必须先单击“New equations”按钮选择合适的方程。
- Exponential：指数拟合，包括两种形式：

$$a\exp(bx)$$

$$a\exp(bx)+c\exp(dx)$$

- Fourier：傅里叶级数拟合，最多可以取到8项

$$a_0+\sum_{k=1}^{n} a_k\cos(kxw)+b_k\sin(kxw) \quad (n=1,2,\cdots,8)$$

- Gaussian：高斯函数拟合，最多取8项：

$$\sum_{k=1}^{n} a_k\exp\left(-\frac{x-b_k}{c_k}\right)^2 \quad (n=1,2,\cdots,8)$$

- Interpolant：内插法，非参数拟合法之一，包括线性内插、最近邻内插、三次样条内插和保形内插，这些都是前面介绍的指令的功能。
- Polynomial：多项式，包括一阶至九阶多项式。
- Rational：有理函数，两个多项式之比，即分子与分母都是多项式，分子与分母的形式要分别选择。分子可以是常数和1阶到5阶的多项式，分母可以是1阶到5阶多项式。
- Power：幂函数拟合，包括两种形式：

$$y=ax^b$$

$$y=ax^b+c$$

- Smooth spline：平滑样条内插法，非参数拟合法之一，平滑样条内插法是对数据进行平滑处理后，再用平滑后的数据来拟合，是曲线拟合工具箱的方法。参数是0产生一个分段的线性多项式拟合，参数是1产生一个分段三次多项式拟合。
- Sum of sin function：正弦函数级数拟合，与傅里叶级数相似，最多可选择8个正弦函数。
- Weibull：两个参数的韦伯分布，表达式是

$$y=abx^{b-1}\exp(-ax^b)$$

韦伯分布由瑞典物理学家Wallodi Weibull于1939年引进，是可靠性分析及寿命检验的理论基础。

这些拟合方法包括参数拟合和非参数拟合两种。参数拟合能够将全部数据拟合成一个数学表达式，但是有些情况下，只能进行分段拟合以求得到一个光滑的曲线，这种方法叫非参数拟合。在以上的方法中，非参数拟合的方法包括内插法和平滑样条内插法，其余都是参数拟合法。

还有一些选项的说明如下：

Apply：采用上述所选各种方法进行拟合。

Immediate apply：在选择一个拟合形式后立即输出结果并保存。

Results：罗列进行拟合的各种参数，包括拟合的形式、置信区间大于95及显示拟合效果好坏的各种参数。

Fit options：包括一些拟合方法，如线性拟合、非线性拟合，以及其它选项。

在Table of Fits下面的框中会列出所有的拟合结果。

§4.4 求方程的零点

§4.4.1 求单调连续函数$f(x)$的实数根的算法

1. 对分法

已知函数$f(x)$在区间$[a,b]$单调连续，设$f(a)<0, f(b)>0$，则a,b间必有一实根。

记中点$x_0=(a+b)/2$，若$f(x_0)=0$，则x_0为所求实根。

若$f(x_0)<0$，则令$a_1=x_0, b_1=b$，若$f(x_0)>0$，则令$b_1=x_0, a_1=a$。这时根必在$[a_1,b_1]$中，它的长度为原区间的一半。

令$x_1=(a_1+b_1)/2$，再重复上面的过程，得到新的区间$[a_2,b_2]$又缩小了一半。一直进行下去，可得到一区间系列

$$[a,b],\quad [a_1,b_1],\quad [a_2,b_2],\cdots,[a_n,b_n]$$

当$n\to\infty$，区间$[a_n,b_n]$的长度$b_n-a_n=(b-a)/2^n$将趋于零，且区间的中点

$$x^*=\lim_{n\to\infty}x_n=\lim_{n\to\infty}\frac{a_n+b_n}{2}$$

是方程根的近似值。可根据需要的精度来决定计算次数，此方法的缺点是收敛速度慢，优点是可靠。只要找到了一个函数值改变了符号的区间，也就是有实根存在的区间，这个方法就能按照设定的精度找到逼近这个实根的浮点数。

2. 切线法

用对分法得到方程$f(x)=0$的一个根的近似值x_n以后，可用切线法来提高速度。

将$f(x)$在x_n处展开成泰勒级数

$$f(x)=f(x_n)+f'(x_n)(x-x_n)+f''(x)(x-x_n)^2/2!+\cdots+O(x^3)$$

取一级近似得

$$f(x_n)+f'(x_n)(x-x_n)=0$$

设$f'(x_n) \neq 0$，以x_{n+1}代替x得到牛顿迭代序列为

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

从图4.14可以看出，x_{n+1}是过点$(x_n, f(x_n))$的切线

$$y - f(x_n) = f'(x_n)(x - x_n)$$

与x轴的交点，所以叫切线法。

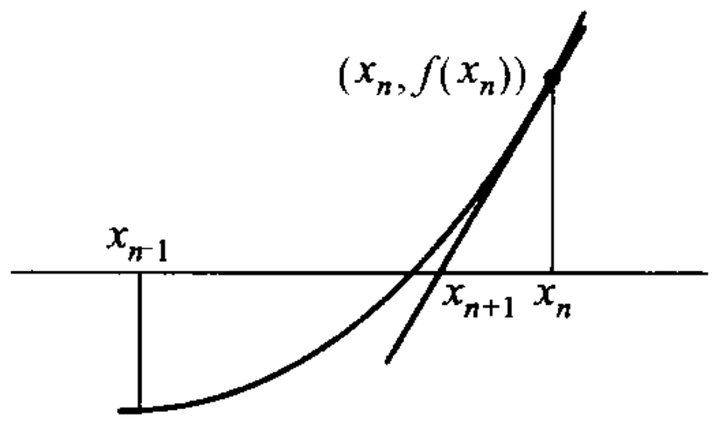


图 4.14 切线法

这个方法效率比对分法高，但是要求函数必须是光滑的，也即存在一阶导数，另外还要求初始猜测解要十分靠近准确解，否则可能会收敛得十分慢，甚至会不收敛。

3. 弦割法

用差分公式近似代替上式中的导数

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

得到的递推公式是

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

这就是弦割法的公式，它通过x_n和x_{n-1}求出x_{n+1}。从图4.15看出，它是用过点$(x_{n-1}, f(x_{n-1}))$, $(x_n, f(x_n))$ 的直线与x轴的交点来求根。它的优点与缺点和牛顿法一样，但是不需要直接计算$f'(x)$。

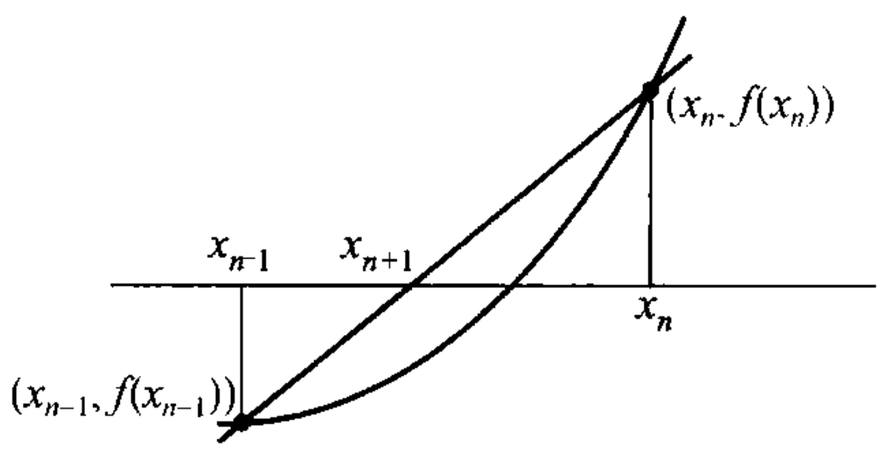


图 4.15 割线法

4. 求函数为某个值的算法

有时候并不是求函数$f(x)$的零点，而是求函数值为某个值时，比如η所对应的自变量的值。这时可以建立一个函数

$$F(x) = f(x) - \eta$$

然后再求$F(x)$的零点。

§4.4.2 MATLAB解方程的指令

MATLAB有多个解方程的指令，要分情况加以应用。

1. 求单变量函数的零点fzero

指令fzero是求单变量函数$f(x)$的零点。它使用的算法叫zeroin算法，是结合了对分法、割线法以及其他方法，如逆二次插值法等的一种综合方法。这个算法简单而可靠，能一直将方程的限制在不断缩小的区间中。在迭代过程中，如果可以就采用快速收敛的方法，否则就用速度较慢的但是非常安全的方法如对分法来求下一步的近似解。

指令的使用格式为：

x=fzero(fun, x0)

x=fzero(fun,x,options,p1,p2,⋯)

[x,fval]=fzero(⋯)

各参量的意义为：

x, fval	零点及对应的函数值，没有零点则返回NaN
fun	求零点的单变量函数，函数值须是实数
x0	猜测的初始值或搜寻零点的区间[x0(1), x0(2)]
options	控制算法的优化选项，[ ]是默认值

函数fzero使用时要输入两个基本参数，即要求解的函数以及初始的猜测解或者求解的区间。由于它把函数当作参数用，所以称之为函数的函数（function functions）。一个函数作为参数传递给另一个函数，有三种方法可用：

(1) 函数句柄

就是在一个内部函数名，或者是由M文件定义的函数名之前加上@。例如

```
>>x = fzero(@sin,3)
x = 3.1416
```

是求内部函数sin$x$在3附近的零点。又如建立一个带参数的函数文件myfun.m，然后用函数句柄求解：

```
function ff = myfun(x,a)
ff= cos(a*x);
```

```
>> a=2;                                    %先给出参数a的值再求根
>> x = fzero(@(x) myfun(x,a),0.1)          %调用时要注明自变量
x =  0.7854
```

对于带参数的内部函数也可以用这种方法调用，如besselj是求贝塞尔函数值的内部函数，下面计算零阶贝塞尔函数在$[0,\pi]$ 之间的零点：

```
>> fzero(@(x)besselj(0,x),[0,pi])
ans = 2.4048
```

(2) 匿名函数

```
>> fzero(@(x)5*x^5+3*x^4-4*x.^3+x.^2+7*x+9,1)
ans = -1.3127
```

在这里，要求零点的函数表达式直接作为参数写入，而没有给这个函数定义一个名称，所以叫匿名函数。

(3) 内嵌对象

这是MATLAB7版本之前采用的一种定义函数的简单方法，它不需要生成新的文件。

```
>>  f=inline('5*x^5+3*x^4-4*x.^3+x.^2+7*x+9');
>> fzero(f,1)
ans =  -1.3127
```

现在推荐使用匿名函数，因为它能生成更高效率的程序代码，操作如下：

```
>> f=@(x)5*x^5+3*x^4-4*x.^3+x.^2+7*x+9
f =  @(x)5*x^5+3*x^4-4*x.^3+x.^2+7*x+9
>> X=fzero(f,1)
X = -1.3127
```

带参数的函数也能这样定义：

```
>> B=@(x,n,y)besselj(n,x)-y
B =  @(x,n,y)besselj(n,x)-y
>> B(1,0,0.5)
ans = 0.2652
```

用fzero调用方法如下：

```
>> fzero(@(x)B(x,0,0.5),[0,2])
ans = 1.5211
```

需要注意，fzero只能求实根，而且一次只能求一个根。如果是给定求根区间，一定要求区间两端的函数值符号相反，也就是一定有根存在这个区间内。

2. 求多项式的零点roots(c)

与指令fzero(c)不同的是，指令roots可以同时求出多项式的全部零点。这里的例

子中使用的多项式与上面的相同，目的是为了对比两个指令在功能上的差别。

指令roots(c)对多项式求零点，c为各幂次项的系数。如多项式为

$$c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$$

则c的表达式为多项式系数按降幂排列

$$c = [c_n,\ c_{n-1},\ \cdots,\ c_1,\ c_0]$$

下面求多项式

$$5x^5 + 3x^4 - 4x^3 + x^2 + 7x + 9 = 0$$

的零点。操作如下

```
>> roots([5,3,-4,1,7,9])
ans = 0.95299 +    0.76221i
      0.95299 -    0.76221i
      -1.3127
     -0.59662 +    0.75154i
     -0.59662 -    0.75154i
```

3. 解非线性方程组

在第一章开头曾经介绍过指令solve，它可以解各种方程，包括符号方程，其实它就是符号工具箱中的指令。符号工具箱中还有一个指令fsolve，可以解多变量的非线性方程组，它的用法如下：

x = fsolve(fun, x0)

[x,fval]=fsolve(fun,x0,⋯)

各参量意义为：

x, fval	零点的位置与对应的函数值
fun	求解的非线性方程组
x0	猜测的初始解

下面用它解方程组：

$$\begin{cases} 2\cos x + \sqrt{y} - \ln z = 7 \\ 2^x + 2y - 8z = -1 \\ x + y - \cosh z = 0 \end{cases}$$

首先要编写函数文件qg.m，将方程中的x, y, z看作矢量$\boldsymbol{X}$的三个分量，函数值$\boldsymbol{Y}$各个分量代表的含义是

$$\begin{cases} Y(1) = 2\cos x + \sqrt{y} - \ln z - 7 \\ Y(2) = 2^x + 2y - 8z + 1 \\ Y(3) = x + y - \cosh z \end{cases}$$

求解时，要求输出函数值为零时的变量的值，也即零点的$\boldsymbol{X}$值和函数值$\boldsymbol{Y}$。

```
% program qg.m
function Y = qg(X)
Y = [2*cos(X(1)) + sqrt(X(2)) - log(X(3)) - 7;
     2^X(1) + 2*X(2) - 8*X(3) + 1;
     X(1) + X(2) - cosh(X(3))]
```

任意猜测一个初始解x_0：

$$x_0 = [\ 0.5+0.5\mathrm{i},\ 0.5+0.5\mathrm{i},\ 0.5+0.5\mathrm{i}\ ]$$

求解过程如下：

```
>> {[X,Y] = fsolve ('qg', [0.5+0.5i, 0.5+0.5i,0.5+0.5i])
X = -0.014456 +      1.601i
     1.0567   -      1.7481i
     0.44424  -      0.3262i
Y = -1.2777e-011   +1.2092e-011i
     1.5534e-012   +5.4401e-013i
     1.199e-014    +1.3936e-013i
```

该指令默认的精度是1e-6，所以$\boldsymbol{Y}$的三个分量与零的偏差的绝对值已经达到精度要求。

§4.4.3 搜寻函数的极小值

MATLAB中有一个指令fminbnd，可以寻找实函数$f(x)$的局部极小值。这里介绍一下它的用法与算法。

先定义一个函数，并画出它大致的图形如图4.16，知道它存在一个极小值

```
>> f =@(x)-1./((x-.3).^2+.01)-1./((x-.9).^2+.04)+6
f = @(x)-1./((x-.3).^2+.01)-1./((x-.9).^2+.04)+6
>> x=-1:0.01:2;
>> y=f(x);
>> plot(x,y)
```

然后在区间[0,0.4]中搜寻极小值。

```
>> fminbnd(f,0,0.4)
   ans =  0.30038
```

如果在区间中只有一个极小值，指令 fminbnd 就能找到它。如果在区间中有多个极小值，指令 fminbnd 只能找出其中的一个，而且还不能肯定是一定是最小值。

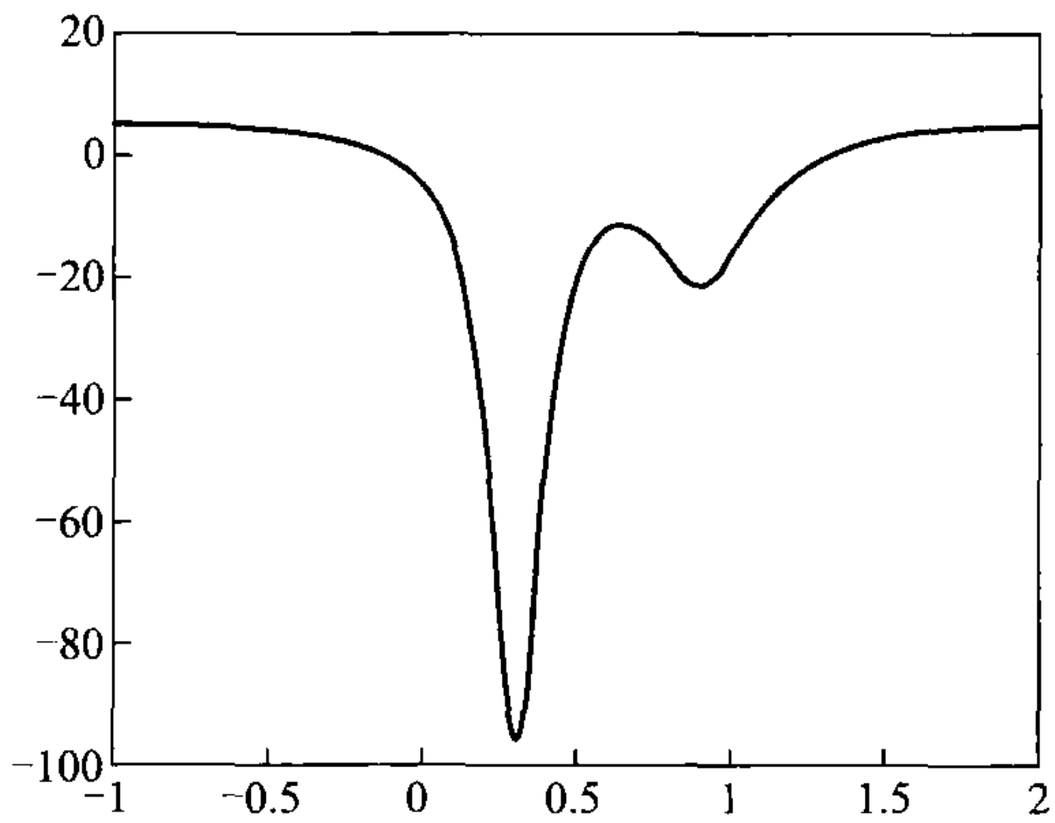


图 4.16 搜寻曲线极小值

对于多变量的单值函数，MATLAB 是用指令 fminsearch 搜寻它的局部极小值。

指令fminbnd使用的算法是黄金分割法与抛物线插值法，介绍如下：

寻找极值的工作与求零点的工作紧密相关，但并不相同。这里不能使用对分法，因为在区间$[a,b]$中，即使知道$f(a), f(b), f((a+b)/2)$的值，也无法决定丢弃哪一半区间而保证剩下的区间一定包含有最小值。

如果将区间分成三等分，令$h=(b-a)/3$，则$u=a+h$和$v=b-h$将区间分成三等分。如果$f(u)<f(v)$，就可以用v代替b，使区间缩小三分之一，同时保证剩下的区间一定包含极小值。不过由于u是在剩下区间的中点处，所以已经计算出来的$f(u)$在下一步骤中毫无用处。

如果采用黄金分割法（golden section）来决定取点的位置，就能克服这个缺点。它的做法如图4.17所示，其中$a=0, b=1$。令$h=\rho(b-a)$，其中ρ 不是1/3而是比1/3大一点的量。点$u=a+h$和$v=b-h$ 将区间分成三个不等长的部分。如果有$f(u)<f(v)$，就可以用v代替b，这个过程与前面的做法相同。

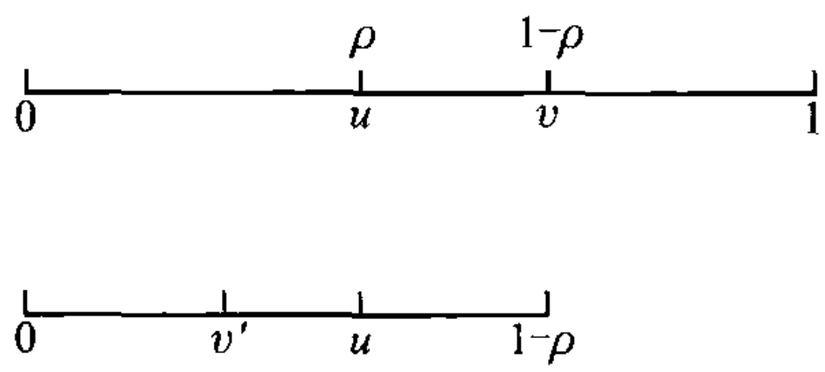


图 4.17 用黄金分割法选点

但是如果要求按照ρ确定的u点能够使得计算出来的$f(u)$在下一步骤中也能

使用，那么这个ρ值就要满足如下比例关系

$$\frac{\rho}{1-\rho}=\frac{1-\rho}{1}$$

即

$$\rho^2-3\rho+1=0$$

解方程得出

$$\rho=(3-\sqrt{5})/2=2-\phi\approx 0.382$$

其中$\phi=(1+\sqrt{5})/2$就是所谓的黄金分割比。而另外一个新点u'的位置是$\rho(1-\rho)$。

使用黄金分割法搜索，区间的长度随每步计算以$\phi-1\approx 0.618$的比例缩小，经过若干步分割以后达到eps的量级即2^{-52}，就有

$$\frac{-52}{\log_2(\phi-1)}\approx 75$$

即经过约75步计算可达到IEEE双精度浮点数计算的舍入误差的大小。

还有一种抛物线插值方法也可以用到这里。经过几次计算以后，就会得到区间内三个不同的点及其函数值。如果由这三个点插值产生的抛物线的最小值在这个区间内，那么下一个点通常是选这个极小值而不是选黄金分割点。黄金分割与抛物线插值相结合，提供了一个一维化问题的可靠而有效的求解方法。

最优化搜索过程停止判据的设置是需要技巧的。在$f(x)$的极小值点，一阶导数$f'(x)$为零。因此在最小值点附近，$f(x)$近似于一个没有一次项的二次函数

$$f(x)\approx a+b(x-c)^2+\cdots$$

函数极小值发生在$x=c$处，并且其值为$f(c)=a$。如果x靠近c，比如$x\approx c+\delta$而δ很小，那么

$$f(x)\approx a+b\delta^2$$

当计算函数值时，x的微小改变会被平方。如果$a\neq 0, b\neq 0$,且大小差不多，那么停止判据就选$\sqrt{\text{eps}}$，因为x的任何更小的改变都不会影响$f(x)$的值。但如果a,b有不同的数值，或者a,c 中有一个近似于零，那么就使用eps乘以区间长度比用$\sqrt{\text{eps}}$更合适。

§4.5 快速傅里叶变换

§4.5.1 离散傅里叶变换

令$\omega_n=\mathrm{e}^{-\mathrm{i}2\pi/n}$，它就是1的$n$次根，即$\omega_n^n=1$。

N个数据点$x_l(l=0,1,\cdots,N-1)$ 的离散傅里叶变换是

$$y_m=\sum_{k=0}^{n-1}x_k\omega_n^{mk}\qquad(m=0,1,2,\cdots,n-1)$$

用矩阵表示为

$$y=F_nx$$

在这里$n\times n$的矩阵F的矩阵元F_{mk}记为

$$\{F_n\}_{mk}=\omega_n^{mk}$$

当$n=4$时，所得的矩阵是

$$F_4=\begin{pmatrix}1&1&1&1\\1&\omega^1&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{pmatrix}=\begin{pmatrix}1&1&1&1\\1&-\mathrm{i}&-1&\mathrm{i}\\1&-1&1&-1\\1&\mathrm{i}&-1&-\mathrm{i}\end{pmatrix}$$

利用$\omega_n^1=-\mathrm{i},\omega_n^2=-1,\omega_n^3=\mathrm{i},\omega_n^4=1$，得到

$$\frac{1}{n}\begin{pmatrix}1&1&1&1\\1&\omega^{-1}&\omega^{-2}&\omega^{-3}\\1&\omega^{-2}&\omega^{-4}&\omega^{-6}\\1&\omega^{-3}&\omega^{-6}&\omega^{-9}\end{pmatrix}\begin{pmatrix}1&1&1&1\\1&\omega^1&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{pmatrix}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

由此知，F_4的逆F_4^{-1}是

$$F_4^{-1}=\frac{1}{n}\begin{pmatrix}1&1&1&1\\1&\omega^{-1}&\omega^{-2}&\omega^{-3}\\1&\omega^{-2}&\omega^{-4}&\omega^{-6}\\1&\omega^{-3}&\omega^{-6}&\omega^{-9}\end{pmatrix}$$

以此类推，可得到相应的逆变换是

$$x=F_n^{-1}F_nx=F_n^{-1}y$$

即有

$$x_k=\frac{1}{n}\sum_{m=0}^{n-1}y_m\omega_n^{-mk},\qquad k=0,1,\cdots,n-1$$

离散傅里叶变换把输入的数据x分解为它的基本频率的组合，也就是变换后所得的y的各个分量。由表达式

$$\omega_n^{-mk}=\left(\mathrm{e}^{-\mathrm{i}\frac{2\pi}{n}}\right)^{-mk}=\mathrm{e}^{\mathrm{i}k\frac{2\pi}{n}m}$$

可知，m/n相当于频率，$m2\pi/n$相当于圆频率。这是因为，与傅里叶变换中的表达式$e^{i2\pi vt}$相比，如果将离散数据看成是从连续的信号中按照时间间隔τ取样所得，则第k个点所对应的时间为$k\tau$，取n个点的时间为$n\tau$，它相当于一个周期的时间，所以基频为$1/n\tau$，倍频为$m/n\tau$，将它们代入$e^{i2\pi vt}$得到$e^{i2\pi\frac{m}{n}k}$。另外，y_m/n相当于各频率的振幅，常把振幅的平方称之为功率谱。其中有两个特殊的分量，一个是y_0，即零频分量，是一个常数，其值为x的各个分量之和，还有一个是$y_{n/2}$，正好是采样频率的一半，叫做尼奎斯特频率(Nyquist frequency)，是由取样频率所表达的最高频率。由于离散傅里叶变换在频率上有周期性，所以除零频及尼奎斯特频率以外的频率是两两共轭的。

§4.5.2 傅里叶级数与傅里叶积分

离散傅里叶变换将N个的离散的数据变换成相同长度的离散的频谱系列。还有另外三种傅里叶变换，即傅里叶级数、傅里叶积分与离散时间傅里叶变换。顺便将它们列在下面以方便读者对它们进行比较。

傅里叶级数将变量范围为无限的连续的周期函数变换成无限的离散的傅里叶频谱序列。傅里叶积分是将变量范围为无限的连续的非周期函数变换成无限的连续的傅里叶频谱序列。离散时间傅里叶变换将一个无限长数据序列变换为周期函数。

1. 傅里叶级数

定义在$[-l,l]$上的任意平方可积函数$f(x)$可展成傅里叶级数

$$f(x)=C_0+\sum_{n=1}^{\infty}\left(C_n\cos\frac{n\pi x}{l}+D_n\sin\frac{n\pi x}{l}\right)$$

其中傅里叶系数C_0,C_n,D_n分别为

$$C_0=\frac{1}{2l}\int_{-l}^{l}f(x)\mathrm{d}x$$

$$C_n=\frac{1}{l}\int_{-l}^{l}f(x)\cos\frac{n\pi x}{l}\mathrm{d}x\qquad(n\neq 0)$$

$$D_n=\frac{1}{l}\int_{-l}^{l}f(x)\sin\frac{n\pi x}{l}\mathrm{d}x\qquad(n\neq 0)$$

在$x=\pm l$的边界上，$f(x)$的傅里叶级数与$f(x)$在边界上的值$f(l),f(-l)$满足如下关系

$$\left[f(x)=C_0+\sum_{n=1}^{\infty}\left(C_n\cos\frac{n\pi x}{l}+D_n\sin\frac{n\pi x}{l}\right)\right]_{x=\pm l}=\frac{1}{2}[f(l)+f(-l)]$$

因此，$f(x)$的傅里叶级数在边界上并不一定收敛于$f(x)$，但是当$f(l)=f(-l)$时，$f(x)$ 的傅里叶级数在边界上收敛于$f(x)$。

$f(x)$也可以展成如下复数形式的傅里叶级数

$$f(x)=\sum_{n=-\infty}^{\infty} C_n \mathrm{e}^{\mathrm{i}\frac{n\pi x}{l}}$$

$$C_n=\frac{1}{2l}\int_{-l}^{l} f(x)\mathrm{e}^{-\mathrm{i}\frac{n\pi x}{l}}\mathrm{d}x$$

即使$f(x)$是实函数，其复数形式的傅里叶级数的系数也可能是复数。

这两种形式的傅里叶级数是等价的，可以根据需要选用。

2. 傅里叶积分

若定义在$(-\infty,\infty)$上的函数$f(x)$满足

(1) 在任一有限区间上连续或有有限个第一类间断点，并且有有限个极大值、极小值；

(2) $\int_{-\infty}^{+\infty}|f(x)|\mathrm{d}x$为有限值，则$f(x)$可以展成如下形式的傅里叶积分

$$f(x)=\int_0^{\infty} A(\omega)\cos\omega x\mathrm{d}\omega+\int_0^{\infty} B(\omega)\sin\omega x\mathrm{d}\omega$$

其中

$$A(\omega)=\frac{1}{\pi}\int_{-\infty}^{\infty} f(\xi)\cos\omega\xi\mathrm{d}\xi$$

$$B(\omega)=\frac{1}{\pi}\int_{-\infty}^{\infty} f(\xi)\sin\omega\xi\mathrm{d}\xi$$

这是实数形式的傅里叶积分，还有复数形式的傅里叶积分，形式为

$$G(\omega)=\frac{1}{2\pi}\int_{-\infty}^{\infty} g(x)\mathrm{e}^{-\mathrm{i}\omega x}\cdot\mathrm{d}x$$

$$g(x)=\int_{-\infty}^{\infty} G(\omega)\mathrm{e}^{\mathrm{i}\omega x}\cdot\mathrm{d}\omega$$

可将两式写成对称形式

$$F(\omega)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)\mathrm{e}^{-\mathrm{i}\omega x}\cdot\mathrm{d}x$$

$$f(x)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{\mathrm{i}\omega x}\cdot\mathrm{d}\omega$$

这两个式子分别叫傅里叶变换式和傅里叶逆变换式。

3. 离散时间傅里叶变换

令x_k为无限长离散数据序列，足标k包括所有正负整数。

离散时间傅里叶变换得到的是一个复数周期函数

$$X(\mathrm{e}^{\mathrm{i}\omega}) = \sum_{k=-\infty}^{\infty} x_k \mathrm{e}^{\mathrm{i}k\omega}$$

这个序列则表示为

$$x_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\mathrm{e}^{\mathrm{i}\omega})\mathrm{e}^{-\mathrm{i}k\omega}\mathrm{d}\omega, \qquad k = \cdots, -1, 0, 1, \cdots$$

对无限长的连续函数按照间隔b取样，就得到无限长的离散的函数。按照取样定理，它的傅里叶变换不同于原来的连续函数的傅里叶变换，是一个以$1/b$为周期的函数，在每个周期内，则是原来的连续函数的傅里叶变换的频谱。如果连续函数有一定的带宽s_c，只要$s_c \leqslant 1/2b$，就可以用一个周期的频谱去代替原来的连续函数的频谱，反之，如果$s_c \geqslant 1/2b$，则各个周期的频谱会发生重叠，导致高端的频谱失真。而N个数据的离散傅里叶变换所得的N个离散的频谱就是无限长的离散的函数的频谱在一个周期[0,1/b] 上N个等分点上的值。

§4.5.3 快速傅里叶变换的MATLAB指令

快速傅里叶变换(FFT)是离散傅里叶变换(DFT)的快速算法。

N个数据点的离散傅里叶变换公式在MATLAB中的形式如下:

$$X(k) = \sum_{n=1}^{N} x(n)\mathrm{e}^{\mathrm{j}2\pi(k-1)\frac{n-1}{N}}$$

$$x(n) = \frac{1}{N}\sum_{k=1}^{N} X(k)\mathrm{e}^{-\mathrm{j}2\pi(k-1)\frac{n-1}{N}}$$

由于在MATLAB中的指标n, k不能为0, 所以公式形式略有变化。这里圆频率是$2\pi(k-1)/N$，各频率的振幅是$X(k)/N$。

1. 一维的FFT及其逆变换

指令及其语句格式如下：

fft(x)

fft(x, N)

在这里，x是待变换的数据，如果输入数据的长度是2的整次幂，则FFT自动采用以2为基数的快速算法，否则就采用较慢的算法。如果x是矩阵，FFT是对每一列进行。N表示进行N个点的FFT，如果N大于x的长度，则用零补充；如果x大于N的长度，则把多余的部分去掉。

逆变换ifft的格式与fft的格式相同。

2. 二维的FFT及其逆变换

指令及其语句格式如下：

fft2(x)

在这里，x是要变换的二维矩阵，如果它是一维的，则按一维进行变换。逆变换ifft2的格式与fft2的格式相同。

n维的FFT及其逆变换指令fftn, ifftn 的格式分别与fft2, ifft2的格式相同。

下面是一维傅里叶变换的例子。

```
>> X = [ 4, 0, 3, 6, 2, 9, 6, 5];                    %原始数据
>> Y=fft(X)
Y = 35                                               %对应零频分量
-5.0711 + 8.6569i
-3 +        2i
9.0711 + 2.6569i
-5                                                   %对应尼奎斯特频率
9.0711 - 2.6569i                                     %以下频率的振幅与上面的共轭
-3 -        2i
-5.0711 - 8.6569i
```

FFT最多的用法是用于分析寻找隐藏在有噪声系列中的信号的频率分量。下面是两个实例。

例4 考虑一个取样频率为 1 000 Hz 的数据。建立一个时间轴，范围从 0 到 0.25 s。步长为 1 ms。然后构造一个包含频率为50 Hz与120 Hz的正弦波的信号。

```
t = 0:0.001:0.25;
x = sin(2*pi*50*t) + sin(2*pi*120*t);
```

加上一些随机的偏差作为噪声以形成信号y，并画出它的图形如图4.18。

```
y = x + 2*randn(size(t));
plot(y(1:50))
title('Noisy time domain signal')
```

显然,很难从这张图上识别出它的频率分量，这就是为什么频谱分析如此流行的原因。对噪声信号y作离散傅里叶变换是很容易的。

```
Y = fft(y,256);
```

利用复共轭(conj)计算功率谱密度以测定不同频率的能量。以前面127个分量画出频率轴（256个点的其余的分量与它们对称）。如图4.19。

```
Pyy = Y.*conj(Y)/256;
```

```
f = 1000/256*(0:127);
plot(f,Pyy(1:128))
title('Power spectral density')
xlabel('Frequency (Hz)')
```

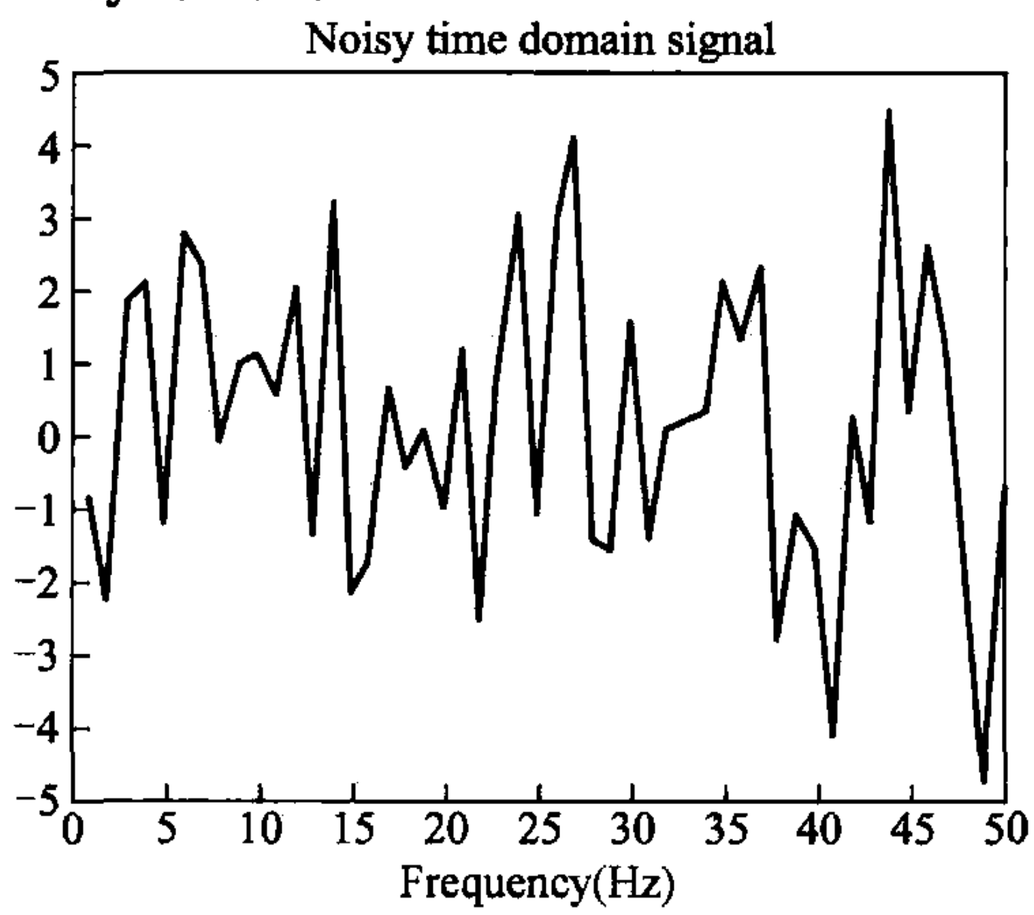


图 4.18 有噪声的信号

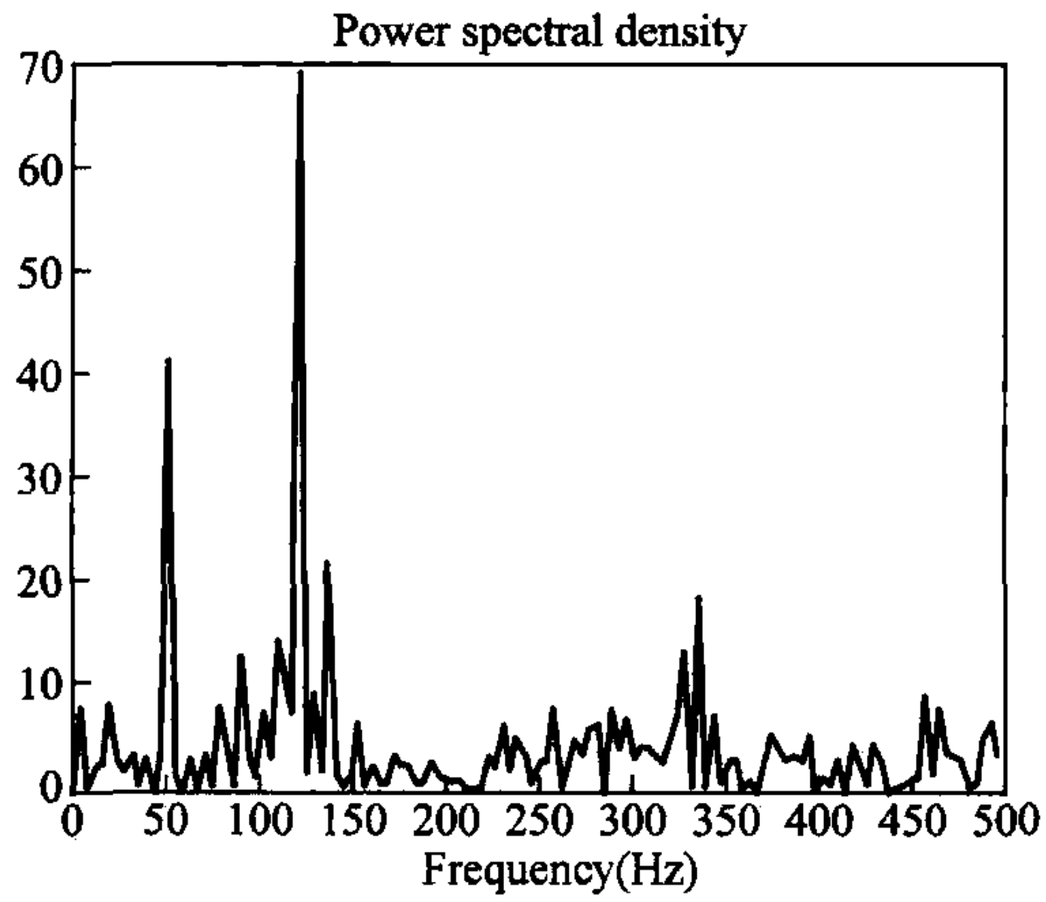


图 4.19 频率分布图

将图形放大只保留到200Hz，如图4.20。注意在50 Hz与120 Hz处的峰值。这就是原始信号的频率。

```
plot(f(1:50),Pyy(1:50))
title('Power spectral density')
xlabel('Frequency (Hz)'
```

例5 对函数$x = \sin(30\pi t) + \sin(80\pi t)$作快速傅里叶变换。

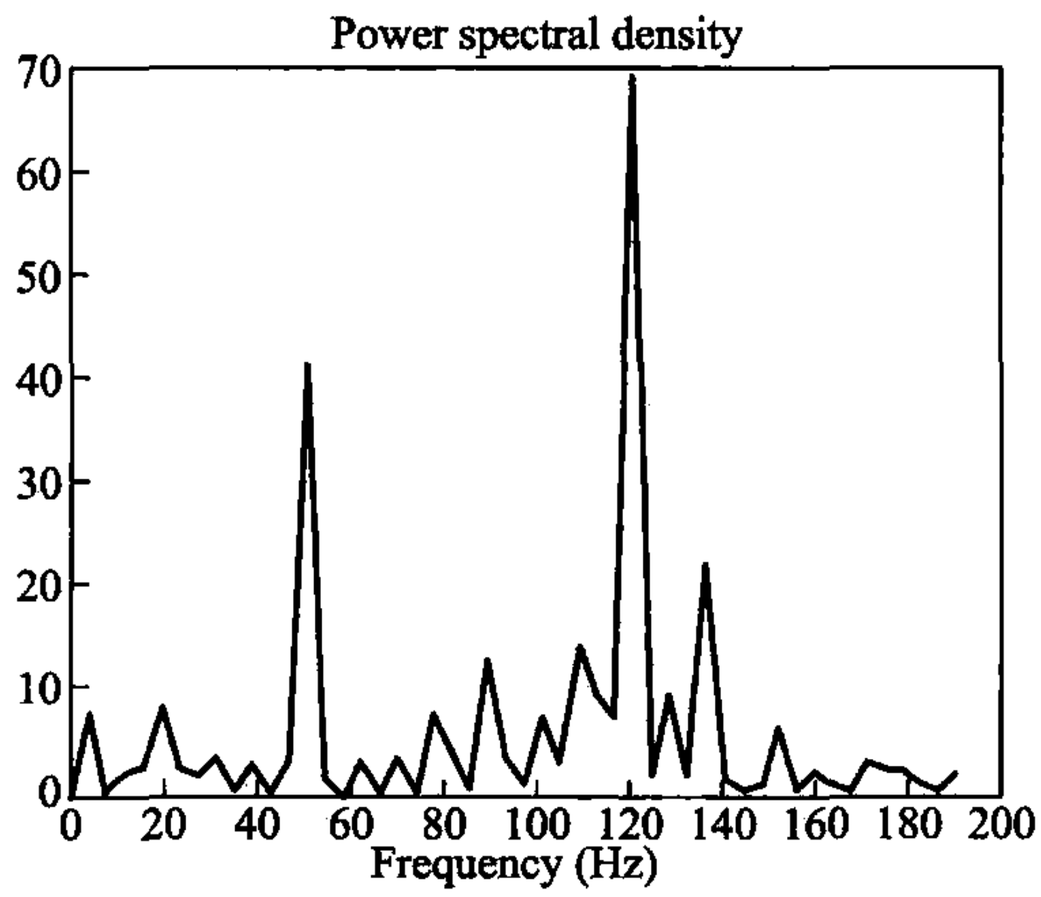


图 4.20 局部放大的频率分布图

```
t = 0:0.01:9.99;
x = sin(2*pi*15*t) + sin(2*pi*40*t);
y=fft(x);
m = abs(y);
f =(0:numel(y)-1)'*100/numel(y);
plot(f,m)
grid on
ylabel('Abs. Magnitude')
xlabel('Frequency [Hertz]')
```

指令numel是计算列阵中的元素总数。图4.21为画出的图形。

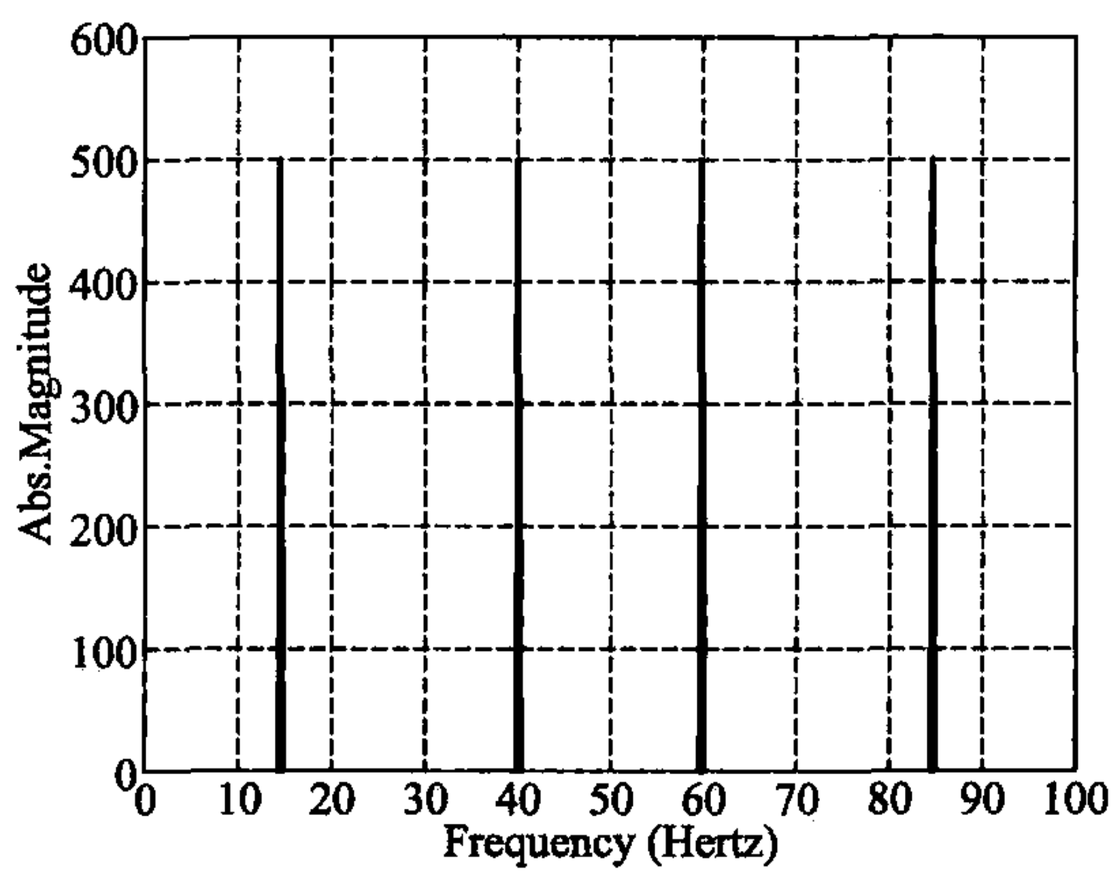


图 4.21 用FFT得到的频率分布图

例6 分析太阳黑子活动的周期性。

通过分析三百年左右的太阳黑子活动的记录资料可以发现每过11年就会达到一个极大值。图4.22是Wolfer数图，它记录的是太阳黑子的数目与面积，天文学家已经积累了近三百年的数据资料。图4.23是放大的前50年的数据图形。

```
load sunspot.dat
year=sunspot(:,1);
wolfer=sunspot(:,2);
plot(year,wolfer)
title('Sunspot Data')
plot(year(1:50),wolfer(1:50),'b.-');
```

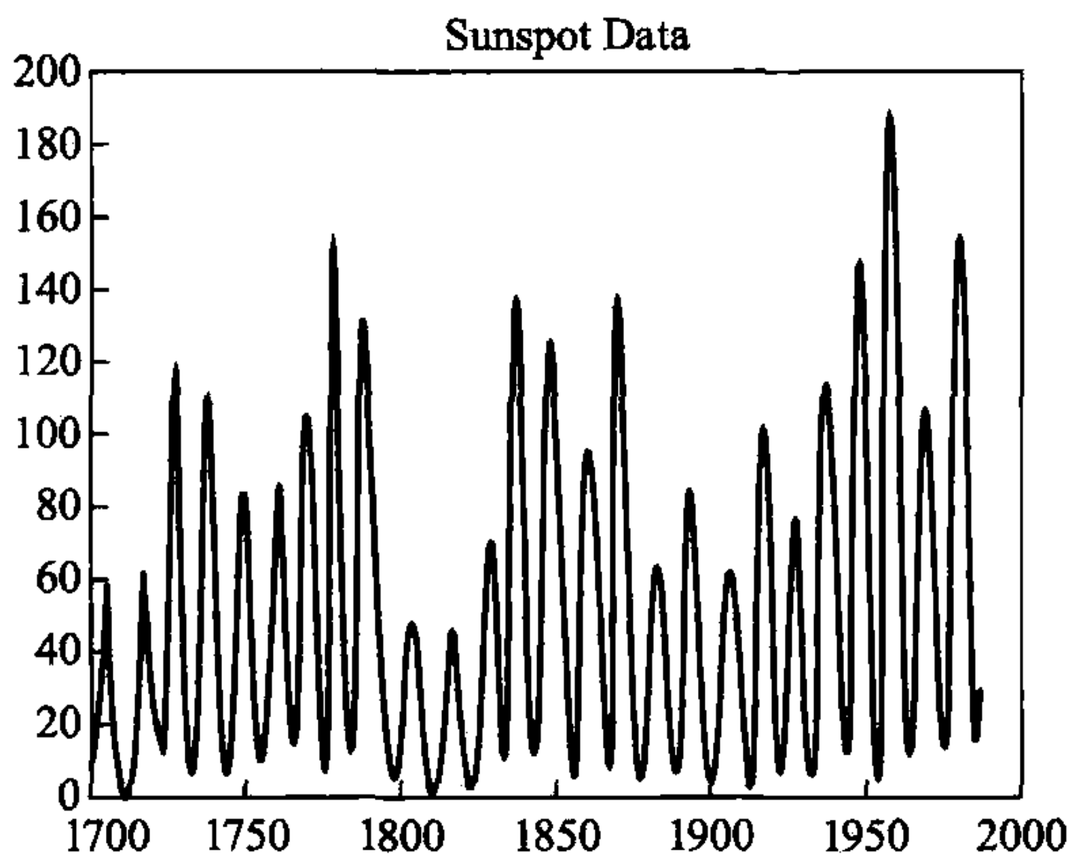


图 4.22 300年的太阳黑子数据

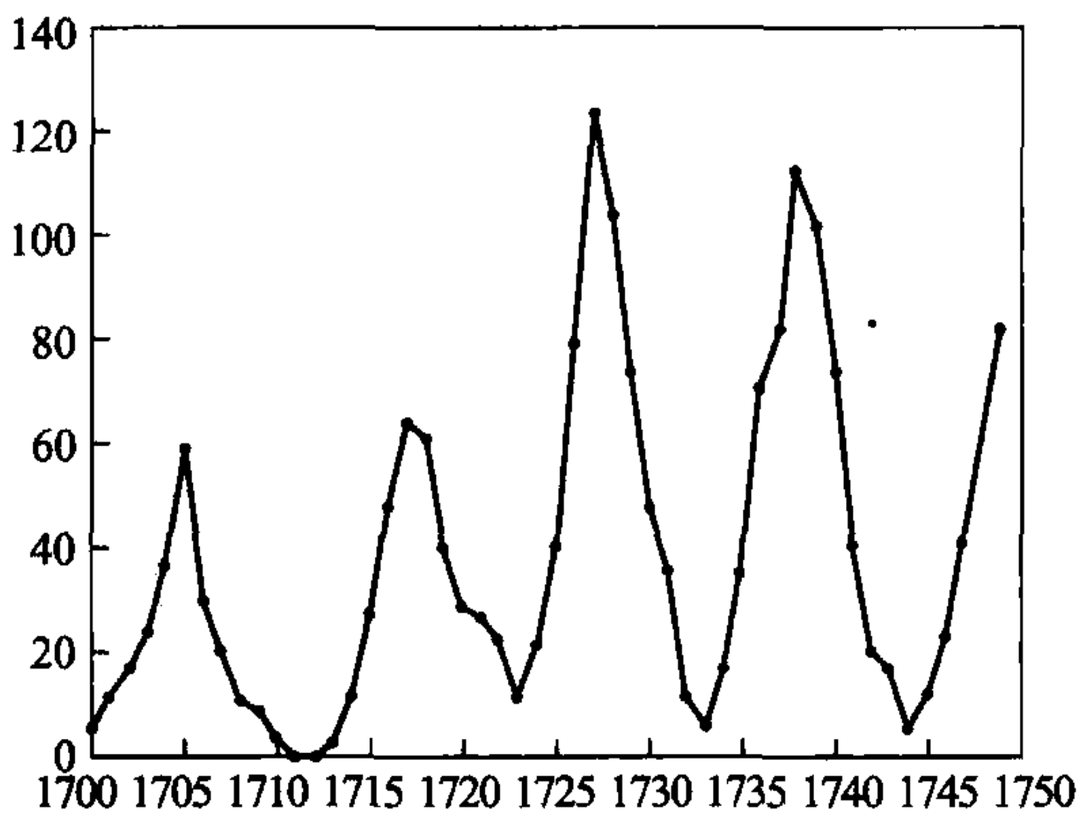


图 4.23 50年的太阳黑子数据

信号处理的基本工具是FFT。用FFT对太阳黑子的数据处理如下。其中第一个分量Y，Y(1)，是所有数据之和，可以删除。

```
Y = fft(wolfer);          Y(1)=[];
```

傅里叶系数Y在复平面上的分布图4.24是很漂亮的，但难以看出其含义。我们需要更有用的方式来考察这些数据。

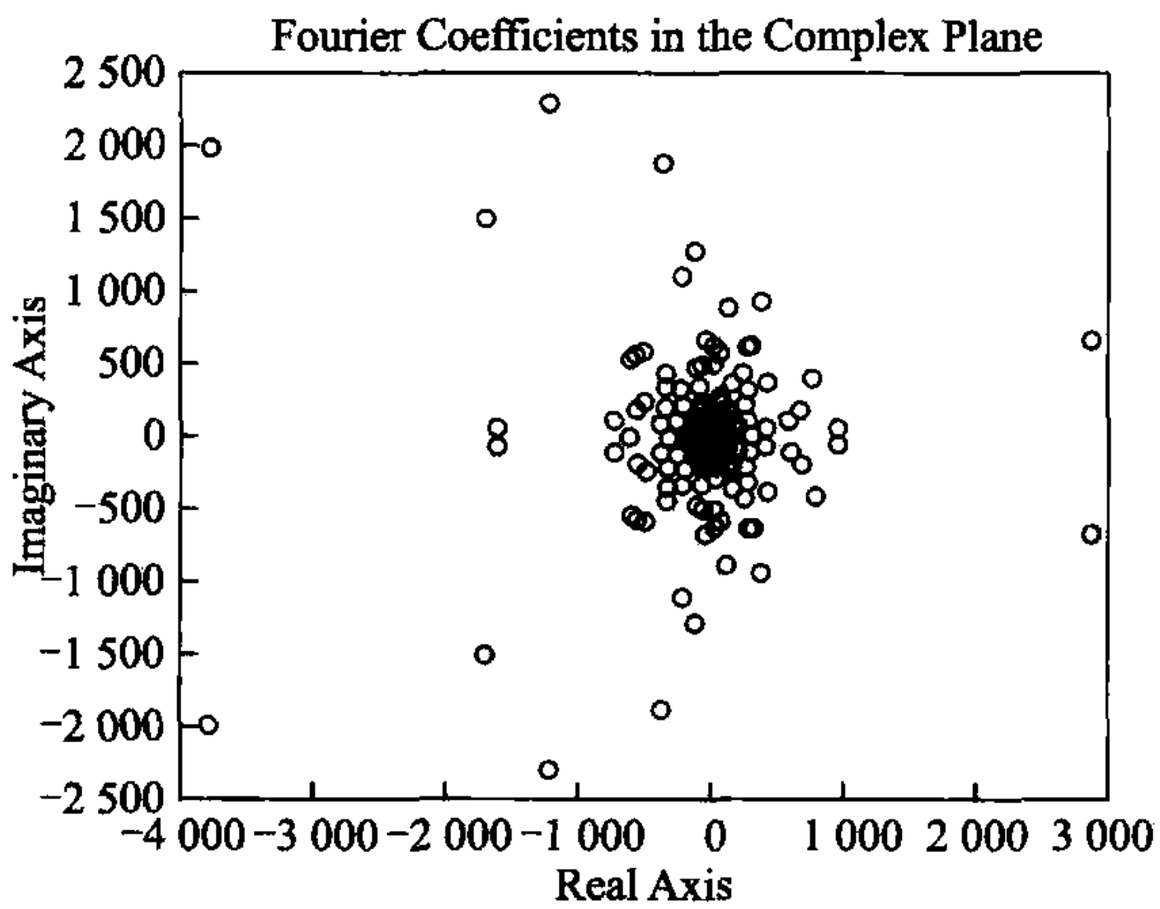


图 4.24

```
plot(Y,'ro')
title('Fourier Coefficients in the Complex Plane');
xlabel('Real Axis');
ylabel('Imaginary Axis');
```

Y的复振幅的平方叫功率，功率相对于频率的图形叫周期图，如图4.25。由于取样频率是每年一次，按照信号处理理论，尼奎斯特频率是取样频率的一半，故为1/2。

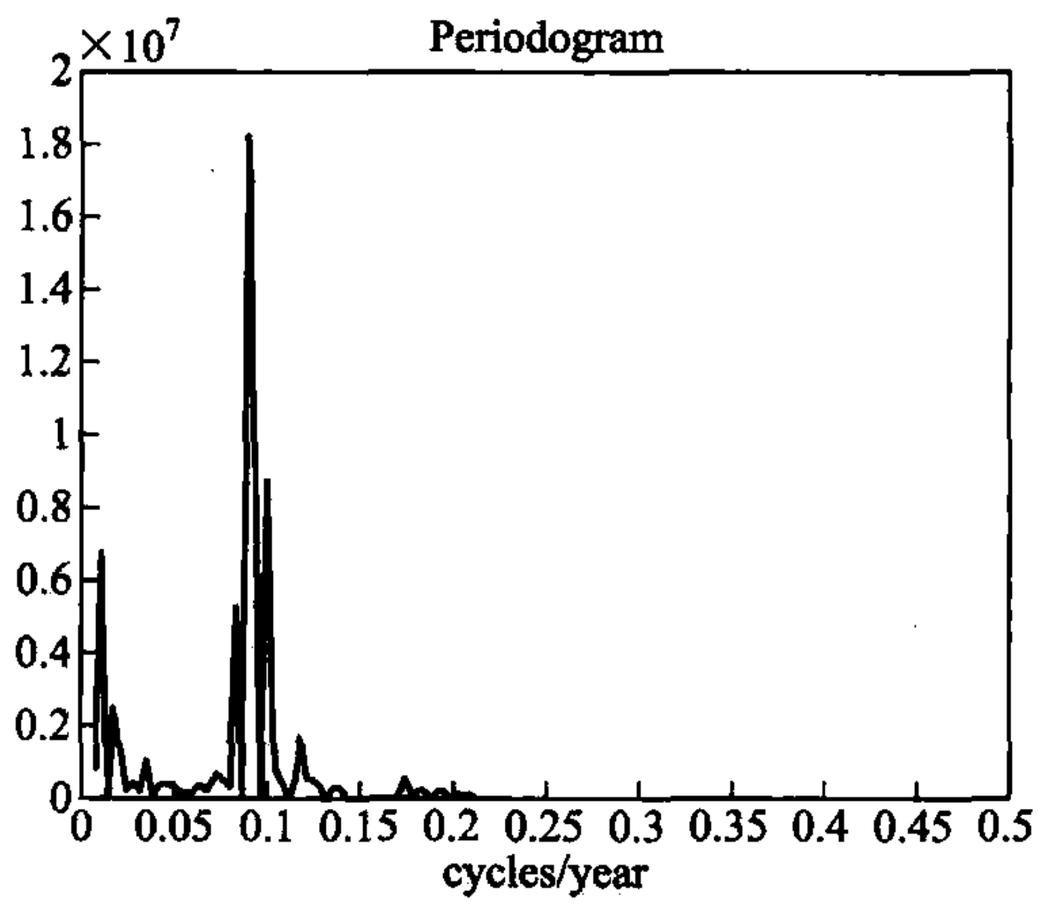


图 4.25 太阳黑子的周期图

```
n=length(Y);
power = abs(Y(1:floor(n/2))).^2;
nyquist = 1/2;
freq =(1:n/2)/(n/2)*nyquist;
plot(freq,power)
xlabel('cycles/year')
title('Periodogram')
```

图上用的标度cycles/year还是不方便，改用years/cycle并用来估计一个周期的长度。

```
plot(freq(1:40),power(1:40));          xlabel('cycles/year')
```

为了方便，我们画出功率相对周期的图形（period=1./freq）(图4.26)。正如所预料的，存在一个明显的长度的为11年的周期。

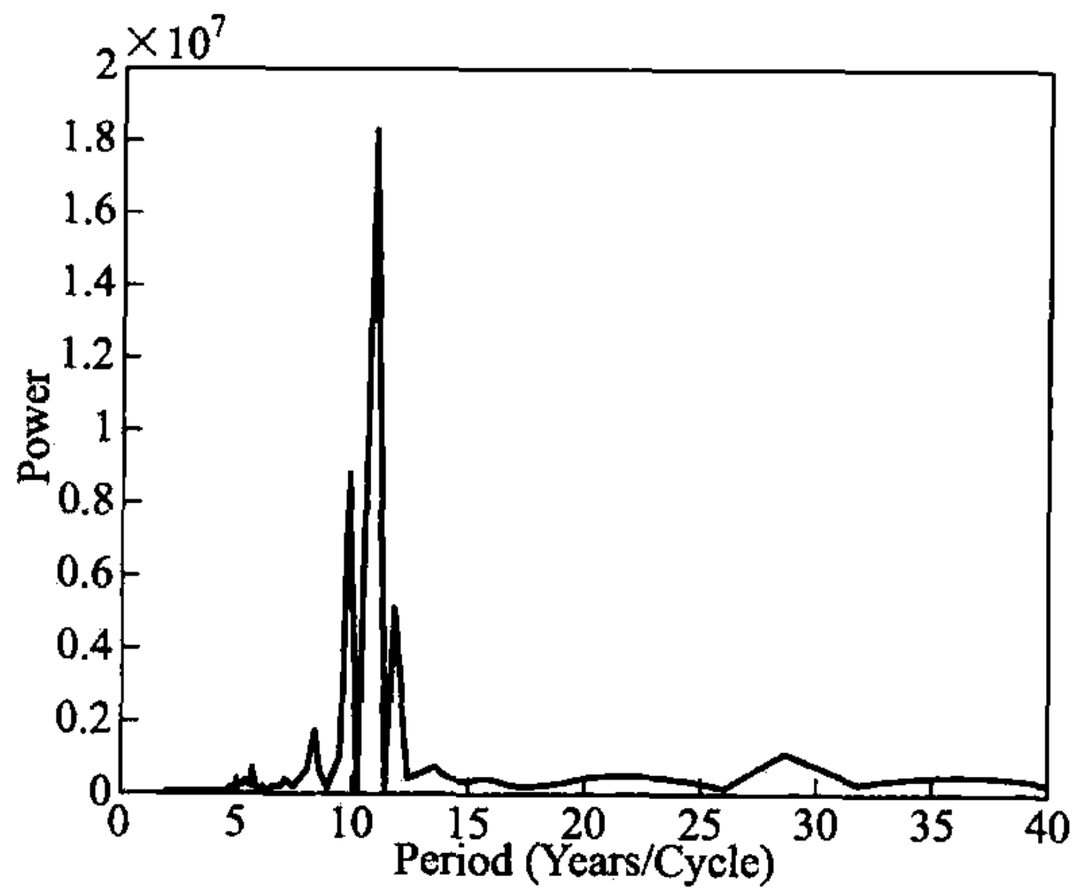


图 4.26 太阳黑子的功率谱

```
period=1./freq;
plot(period,power);
axis([0 40 0 2e+7]);
ylabel('Power');
xlabel('Period (Years/Cycle)');
```

最后，我们通过挑选强度最大的频率来更精确地确定一下周期的长度。有一个红点标志了这个点（图4.27）。

```
hold on;
index=find(power==max(power));
mainPeriodStr=num2str(period(index));
```

```
plot(period(index),power(index),'r.', 'MarkerSize',25);
text(period(index)+2,power(index),['Period = ',mainPeriodStr]);
hold off;
```

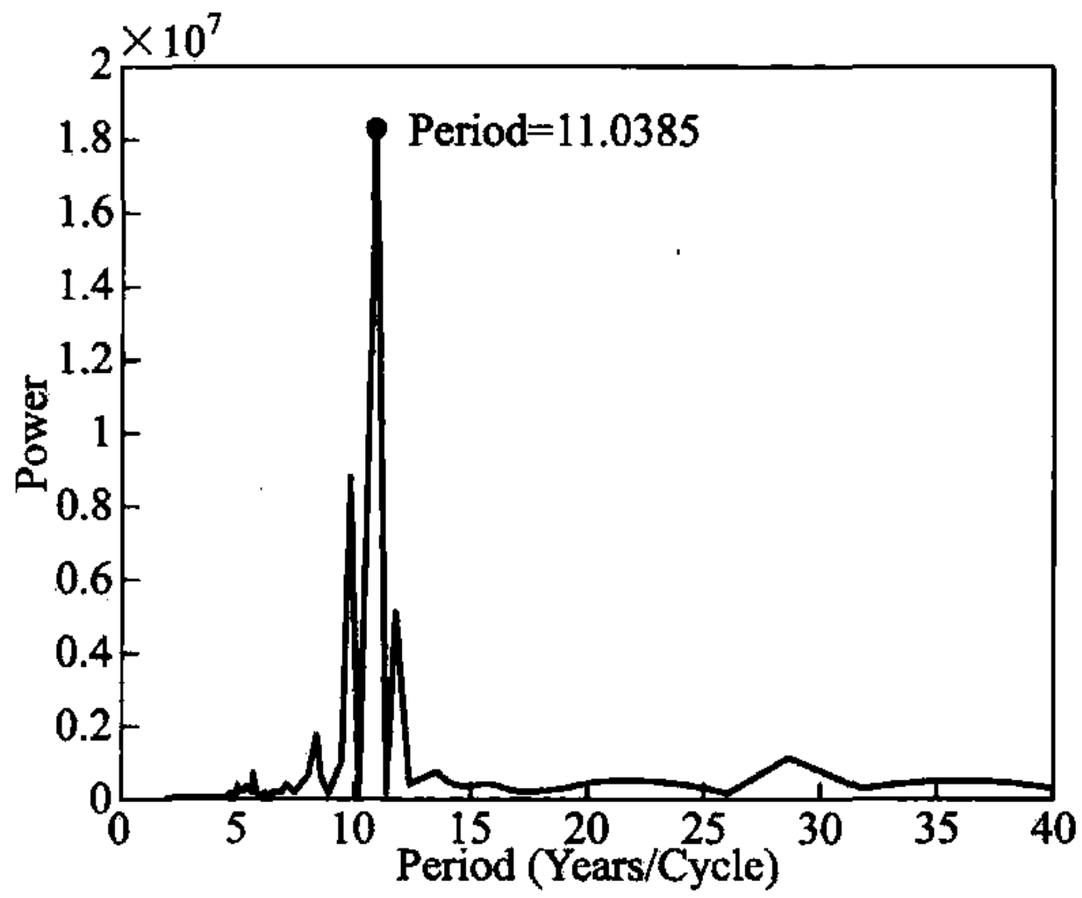


图 4.27 强度最大的频率对应的周期

§4.5.4 快速傅里叶变换的算法

本节介绍离散傅里叶变换的快速算法，也即快速傅里叶变换(FFT)。

利用离散傅里叶变换的某种对称性和允余性，可以发展一种很有效的计算离散傅里叶变换的算法。下面以$n=4$ 为例加以说明。

$$y_m = \sum_{k=0}^{3} x_k \omega_n^{mk} \qquad (m=0,1,2,3)$$

写出来就是

$$\begin{aligned}
y_0 &= x_0\omega_n^0 + x_1\omega_n^0 + x_2\omega_n^0 + x_3\omega_n^0 \\
y_1 &= x_0\omega_n^0 + x_1\omega_n^1 + x_2\omega_n^2 + x_3\omega_n^3 \\
y_2 &= x_0\omega_n^0 + x_1\omega_n^2 + x_2\omega_n^4 + x_3\omega_n^6 \\
y_3 &= x_0\omega_n^0 + x_1\omega_n^3 + x_2\omega_n^6 + x_3\omega_n^9
\end{aligned}$$

注意到

$$\omega_n^0 = \omega_n^4 = 1, \quad \omega_n^2 = \omega_n^6 = -1, \quad \omega_n^9 = \omega_n^1$$

可以将上式重新组合为

$$\begin{aligned}
y_0 &= (x_0 + \omega_n^0 x_2) + \omega_n^0 (x_1 + \omega_n^0 x_3) \\
y_1 &= (x_0 - \omega_n^0 x_2) + \omega_n^1 (x_1 - \omega_n^0 x_3) \\
y_2 &= (x_0 + \omega_n^0 x_2) + \omega_n^2 (x_1 + \omega_n^0 x_3) \\
y_3 &= (x_0 - \omega_n^0 x_2) + \omega_n^3 (x_1 - \omega_n^0 x_3)
\end{aligned}$$

可以看到，现在只要8次加法和6次乘法，而不是原来的$(4-1)\times 4=12$次加法和$4^2=16$次乘法。其实考虑$\omega_n^0=1$，需要的乘法次数还会更少，不过我们这里主要是为了说明一般的算法。现在计算4点的DFT已经变成了2点的对奇数数列和偶数数列的变换。这个性质一般都成立，如果n是偶数，则n点的DFT可能分解成两个长度为一半的DFT。

看一下前几个傅里叶矩阵，这种特征更清楚，

$$\boldsymbol{F}_1=1,\quad \boldsymbol{F}_2=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},\quad \boldsymbol{F}_4=\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -\mathrm{i} & -1 & \mathrm{i}\\ 1 & -1 & 1 & -1\\ 1 & \mathrm{i} & -1 & -\mathrm{i}\end{pmatrix}$$

引入置换矩阵$\boldsymbol{P}_4$和对角矩阵$\boldsymbol{D}_2$:

$$\boldsymbol{P}_4=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix},\qquad \boldsymbol{D}_2=\begin{pmatrix}1 & 0\\ 0 & -\mathrm{i}\end{pmatrix}$$

可得

$$\boldsymbol{F}_4\boldsymbol{P}_4=\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & -\mathrm{i} & \mathrm{i}\\ 1 & 1 & -1 & -1\\ 1 & -1 & \mathrm{i} & -\mathrm{i}\end{pmatrix}=\begin{pmatrix}\boldsymbol{F}_2 & \boldsymbol{D}_2\boldsymbol{F}_2\\ \boldsymbol{F}_2 & -\boldsymbol{D}_2\boldsymbol{F}_2\end{pmatrix}$$

对照一下，可以看出，前面交换以后的结果，其实就是$\boldsymbol{F}_4\boldsymbol{P}_4\boldsymbol{x}$。可见,$\boldsymbol{F}_4$可以排列成用$\boldsymbol{F}_2$组成的方块。只要点数是偶数，这种分层分解的步骤就可以逐层进行下去。通常，$\boldsymbol{P}_4$ 是交换矩阵，它将$\boldsymbol{F}_n$的偶数列放到奇数列之前。而

$$\boldsymbol{D}_{n/2}=\mathrm{diag}\left(1,\omega_n,\cdots,\omega_n^{(n/2)-1}\right)$$

因而，把$\boldsymbol{F}_n$应用于长度为n的数列，只要将$\boldsymbol{F}_{n/2}$分别作用于它的偶数列和奇数列，然后再用$\boldsymbol{D}_{n/2}$标度一下。用这种递归分解与解决的方法来计算DFT就叫FFT。

§4.6 练　　习

以下题目可用曲线拟合工具箱完成。

1. 用一次多项式拟合下列数据

x	-1.0	-0.75	-0.50	-0.25	0	0.25	0.50	0.75	1.00
y	-0.2209	0.3295	0.8826	1.4392	2.0003	2.5645	3.1334	3.7061	4.2836

2. 用二次多项式拟合下列数据

x	-3	-2	0	3	4
y	18	10	2	2	5

3. 令$x=-0.2:0.1:1$, $y=2x^5-3x^3-x^2-9$，利用产生的数据作多项式拟合，看看结果如何。如果对这些作保形内插法拟合，比较两条拟合曲线。注意在多项式拟合时会得到多项式的数学表达式，而保形拟合不产生数学表达式。再保持x不变，在原来的y值加上随机偏差0.1*rand(1,13)，再作一次多项式拟合和保形拟合，看看结果有什么不同。

4. 已知方形波数据为

$$x=0:0.1:2,$$

$$y=[0,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0,0,0,0,0,0,0,0,0,0]$$

分别用曲线拟合中8个不同的傅里叶级数去拟合它们，看看结果如何。再用保形内插法对它拟合，结果又是如何。最后用快速傅里叶变换和反变换的指令对这组数据作运算，并画出图形，再看看结果如何。

5. 用指数公式拟合数据

x	1	2	3	4	5	6	7	8
y	15.3	20.5	27.4	36.6	49.1	65.6	87.87	117.6

6. 令$x=0:0.1:2, y=1/(7-x^2+x)$，再对它们作有理函数拟合，比较拟合结果与原来函数的差别，再用保形内插法对数据作拟合，看看效果如何。

7. 自己编写一个指数拟合的练习。

8. 求方程$4\cos x=\mathrm{e}^x$的根，要求保留三位小数的有效数字：

(1) 用对分法编程，取$x_0=\pi/4, x_1=\pi/2$；

(2) 用牛顿法编程，取$x_0=\pi/4$；

(3) 用弦割法编程，取$x_0=\pi/4, x_1=\pi/2$；

(4) 用指令fzero计算，取$x_0=\pi/4, x_1=\pi/2$。

比较到达同样精度时前三种方法所需要的计算次数。

9. 已知$y=x^4-4x^3-6x^2-16x+4$，在$-1<x<4$的区间内寻找方程的零点、极小值及其对应的变量值。

10. 求函数$f(x,y)=x^2+25y^2-2x+50y+25$的极小点与极小值，寻找从初始点$x_0=2, y_0=3$开始。

11. 解代数方程组

$$\begin{pmatrix} 3.3330 & 15920 & -10.333 \\ 2.2220 & 16.710 & 9.6120 \\ 1.5611 & 5.1791 & 1.6852 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 15913 \\ 28.544 \\ 8.4252 \end{pmatrix}$$

12. 在学习§5.9.2的弹簧摆以后，完成下列计算：

(1) 画出弹簧摆在r与θ方向的位移曲线；

(2) 对画出的两条位移曲线进行快速傅里叶分析，找出它们振幅最大的频率。当位移幅度从小到大增加时，频谱有何变化?

第五章　解常微分方程

本章介绍常微分方程的初值问题，边值问题与本征值问题的数值解法。还介绍了 MATLAB 解常微分方程的指令。

§5.1　龙格–库塔法

首先讨论常微分方程的初值问题，常用的基本方法是龙格–库塔法。

§5.1.1　基本思想

1. 数值法解常微分方程

一阶的常微分方程的初值问题可以表示成

$$\begin{cases} \dot{y} = f(t, y) \quad (a \leqslant t \leqslant b) \\ y(t_0) = y_0 \end{cases}$$

这里$\dot{y}$表示一阶导数。

取步长为$h = (b-a)/n$，将区间分成n个子区间：

$$a = t_0 < t_1 \cdots < t_i < \cdots < t_n = b$$

解方程的目的就是要求出在一系列离散点上的近似的$y(t_i)$，

由中值定理，得

$$\frac{y(t_{i+1}) - y(t_i)}{h} = \dot{y}(t_i + \theta h) \quad (0 < \theta < 1)$$

将它改写为

$$y(t_{i+1}) = y(t_i) + hk_{\text{ave}}$$

平均斜率为

$$k_{\text{ave}} = f(t_i + \theta h, y(t_i + \theta h))$$

这个递推公式表明，如果能确定平均斜率，就能从$y(t_i)$求出后面的$y(t_{i+1})$。在后面的各种方法中，实质就是用不同的方法确定不同精度的平均斜率。

2. 欧拉公式

欧拉法是最简单的一种方法，也是其他方法的基础。

取$k_{\text{ave}} = f(t_i, y_i)$，即以$t_i$点的斜率作为平均斜率，得到欧拉公式

$$y_{i+1} = y_i + hf(t_i, y_i)$$

如果取t_i, t_{i+1}两点斜率的平均值作为平均斜率，则得改进的欧拉公式为

$$y_{i+1} = y_i + \frac{h}{2}(K_1 + K_2)$$

其中K_1和K_2分别为t_i 和t_{i+1} 两点的斜率值。

由于t_{i+1} 的斜率K_2还是未知值，所以采用一种叫预报法的方法来计算。先用欧拉法求得$y(t_{i+1})$的预报值

$$\overline{y}_{i+1} = y_i + hK_1 = y_i + hf(t_i, y_i)$$

再用预报值计算t_{i+1}处的斜率值

$$K_2 = f(t_{i+1}, \overline{y}_{i+1}) = f(t_{i+1}, y_i + hf(t_i, y_i))$$

可得

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{2}(K_1 + K_2) \\ &= y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_i + hf(t_i, y_i))] \end{aligned}$$

这就是改进的欧拉公式，通常写成

$$\begin{cases} y_{i+1} = y_i + \dfrac{h}{2}(K_1 + K_2) \\ K_1 = f(t_i, y_i) \\ K_2 = f(t_i + h, y_i + hK_1) \end{cases}$$

由此可见，为了提高精度，可以多取几点的斜率值作加权平均当作平均斜率k_{ave}，这就是龙格–库塔法的基本思想。

一般的显式龙格–库塔法可以写成

$$y_{i+1} = y_i + h\sum_{m=1}^{N} \lambda_m K_m$$

式中

$$K_1 = f(t_i, y_i)$$

$$K_m = f(t_i + \alpha h, y_i + h\sum_{j=1}^{m-1} \beta_{mj} K_j), \qquad m = 2, 3, \cdots, N$$

其中的λ_m，α_m，β_{mj} 都是常数。确定这些常数的原则是，使局部截断误差关于h的阶数尽可能的高。

§5.1.2 二阶龙格–库塔法

如果在改进的欧拉公式中将边界点t_{i+1}替换为区间中内的某个点

$$t_{i+p} = t_i + ph \qquad (0 < p \leqslant 1)$$

还是将它的斜率记为K_2。利用欧拉公式得$y(t_{i+p})$的预报值

$$\overline{y}_{i+p} = y_i + phK_1$$

求得

$$K_2 = f(t_{i+p}, \overline{y}_{i+p})$$

平均斜率则取

$$k_{\text{ave}} \approx \lambda_1 K_1 + \lambda_2 K_2$$

其中λ_1，λ_2是待定的系数。

这样得到的公式为

$$\begin{cases} K_1 = f(t_i, y_i) \\ K_2 = f(t_i + ph, y_i + phK_1) \\ y_{i+1} = y_i + h(\lambda_1 K_1 + \lambda_2 K_2) \end{cases}$$

现在是任务是如何选取参数λ_1，λ_2，p，能使公式的局部截断误差关于h 的阶尽可能高。对上式加以计算并以$\widetilde{y}_{i+1}$表示计算结果，得到

$$\begin{aligned} K_1 &= f(t_i, y_i) = \dot{y}(t_i) \\ K_2 &= f(t_i + ph, y_i + phK_1) \\ &= f(t_i, y_i) + ph\dot{f}(t_i, y_i) + O(h^2) \\ &= \dot{y}(t_i) + ph\ddot{y}(t_i) + O(h^2) \\ \widetilde{y}_{i+1} &= y(t_i) + h(\lambda_1 + \lambda_2)\dot{y}(t_i) + \lambda_2 ph^2 \ddot{y}(t_i) + O(h^3) \end{aligned}$$

将它与$y(t_{i+1})$在t_i处的泰勒展开式

$$y(t_{i+1}) = y(t_i) + h\dot{y}(t_i) + \frac{h^2}{2}\ddot{y}(t_i) + O(h^3)$$

对比可见，若要局部截断误差达到$O(h^3)$, 则要求有

$$\lambda_1 + \lambda_2 = 1, \quad \lambda_2 p = \frac{1}{2}$$

这里有三个参数的两个方程，它可以有无穷组解，都统称为二阶龙格–库塔公式。当我们选取

$$\lambda_1 = \lambda_2 = 1/2, \quad p = 1$$

时就得到前面的欧拉公式。如果取中点$t_{i+1/2}$的斜率为K_2，即取

$$p = 1/2, \quad \lambda_1 = 0, \quad \lambda_2 = 1$$

就得到变形的欧拉公式（中点公式）为

$$\begin{cases} y_{i+1} = y_i + hK_2 \\ K_1 = f(t_i, y_i) \\ K_2 = f(t_i + \dfrac{h}{2}, y_i + \dfrac{h}{2}K_1) \end{cases}$$

§5.1.3 三阶与四阶龙格–库塔法

三阶的龙格–库塔公式是以三个点t_i，t_{i+p}，t_{i+q}的斜率值K_1，K_2，K_3的加权平均作为k_{ave}，所得的公式形式为

$$y_{i+1} = y_i + h(\lambda_1 K_1 + \lambda_2 K_2 + \lambda_3 K_3)$$

假定K_1，K_2仍取上节的形式，第三个点t_{i+q}的取法是

$$t_{i+q} = t_i + qh, \quad 0 < p \leqslant q \leqslant 1$$

然后利用K_1，K_2来预报$y(t_{i+q})$得到

$$\begin{aligned} \overline{y}_{i+q} &= y_i + qh(rK_1 + sK_2) \\ K_s &= f(t_{i+q}, \overline{y}_{i+q}) \\ &= f(t_{i+q}, y_i + qh(rK_1 + sK_2)) \end{aligned}$$

所得的公式形式为

$$\begin{cases} y_{i+1} = y_i + h(\lambda_1 K_1 + \lambda_2 K_2 + \lambda_3 K_3) \\ K_1 = f(t_i, y_i) \\ K_2 = f(t_i + ph, y_i + phK_1) \\ K_3 = f(t_i + qh, y_i + qh(rK_1 + sK_2)) \end{cases}$$

适当地选择其中的参数，可使局部截断误差为$O(h^4)$。为此，只要利用泰勒展开并采用类似于上一节的推导方法，可以得到七个参数所满足的五个条件：

$$\begin{cases} r + s = 1 \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \lambda_2 p + \lambda_3 q = \dfrac{1}{2} \\ \lambda_2 p^2 + \lambda_3 q^2 = \dfrac{1}{3} \\ \lambda_3 pqs = \dfrac{1}{6} \end{cases}$$

它有无穷组解，都称为三阶龙格–库塔公式。下面就是其中的一种：

$$\begin{cases} y_{i+1} = y_i + \dfrac{h}{6}(K_1 + 4K_2 + K_3) \\ K_1 = f(t_i, y_i) \\ K_2 = f(t_i + \dfrac{h}{2}, y_i + \dfrac{h}{2}K_1) \\ K_3 = f(t_i + h, y_i + h(-K_i + 2K_2)) \end{cases}$$

经典的龙格–库塔法是一个四阶的方法。计算公式是：

$$\begin{cases} y_{n+1} = y_i + \dfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = hf(t_i, y_i) \\ k_2 = hf(t_i + \dfrac{h}{2}, y_i + \dfrac{k_1}{2}) \\ k_3 = hf(t_i + \dfrac{h}{2}, y_i + \dfrac{k_2}{2}) \\ k_4 = hf(t_i + h, y_i + k_3) \end{cases}$$

经典的龙格–库塔法的总体截断误差是$O(h^4)$。

§5.2 变步长的龙格–库塔法

以上介绍的龙格–库塔法都是定步长的，单从每一步看，步长越小，截断误差就越小；但随着步长的减小，不但引起计算量的增大，而且可能导致舍入误差的严重积累，因此同积分的数值计算一样，微分方程的数值解法也要注意合理选择步长。

下面以标准的四阶龙格–库塔公式为例加以讨论。从节点t_i出发，先以某个步长h求出一个近似值，记为$y_{i+1}^{(h)}$，由于标准四阶公式的局部截断误差为$O(h^5)$，故有

$$y(t_{i+1}) - y_{i+1}^{(h)} \approx ch^5$$

这里系数c与$y^{(5)}(t)$在$[t_i, t_{i+1}]$ 内的值有关。然后将步长折半，即取$h/2$为步长，从t_i 跨两步到t_{i+1}，再求得一个近似值$y_{i+1}^{(\frac{h}{2})}$，每跨一步的局部截断误差是$c\left(\dfrac{h}{2}\right)^5$，因此有

$$y(t_{i+1}) - y_{i+1}^{(\frac{h}{2})} \approx 2c\left(\frac{h}{2}\right)^5$$

可见，步长折半以后，误差大约减少为原来的$\frac{1}{16}$，即有

$$\frac{y(t_{i+1})-y_{i+1}^{(\frac{h}{2})}}{y(t_{i+1})-y_{i+1}^{(h)}}\approx\frac{1}{16}$$

由此可得下列事后估计式

$$y(t_{i+1})-y_{i+1}^{(\frac{h}{2})}\approx\frac{1}{15}(y_{i+1}^{(\frac{h}{2})}-y_{i+1}^{(h)})$$

这样，可以通过检查步长折半前后两次计算结果的偏差

$$\Delta=|y_{i+1}^{(\frac{h}{2})}-y_{i+1}^{(j)}|$$

来判断所选取的步长是否合适。具体可分以下两种情况：

(1) 对于给定的精度ε，如果$\Delta>\varepsilon$，就反复将步长折半计算，直至$\Delta<\varepsilon$为止，这时取步长折半后的“新值”$y_{i+1}^{(\frac{h}{2})}$作为结果。

(2) 如果$\Delta<\varepsilon$，就反复将步长加倍，直至$\Delta>\varepsilon$为止，这时取步长加倍前的“旧值”作为结果。

这种通过步长折半或加倍的计算的方法就叫变步长方法。表面上看，为了选择步长，每一步的计算量似乎增加了，但从总体上考虑往往是合算的。

§5.3 常微分方程组的初值问题

高阶的常微分方程可以化成一阶的常微分方程组，例如谐波振荡器的二阶的常微分方程

$$\frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2}=-x(t)$$

令

$$y=\begin{pmatrix}x(t)\\ \frac{\mathrm{d}x(t)}{\mathrm{d}t}\end{pmatrix}=\begin{pmatrix}y_1(t)\\ y_2(t)\end{pmatrix}$$

可以表示成

$$\dot{y}=\begin{pmatrix}y_2(t)\\ -y_1(t)\end{pmatrix}$$

前面求解一阶常微分方程的初值问题的解法很容易推广到一阶常微分方程组的情形，只要将y和右端的函数f理解为列向量。下面用标准的四阶龙格-库塔公式为例加以说明。

一阶常微分方程组初值问题的向量形式为

$$\begin{cases} \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}) \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \end{cases}$$

其中

$$\boldsymbol{y} = (y_1, y_2, \cdots, y_n)^{\mathrm{T}}$$

$$\boldsymbol{f} = (f_1, f_2, \cdots, f_n)^{\mathrm{T}}$$

这里上角T表示转置，再定义$\boldsymbol{f}(t, \boldsymbol{y})$为右端函数，写成分量的形式就是：

$$\begin{cases} \dot{\boldsymbol{y}}_m = \boldsymbol{f}_m(t, y_1, y_2, \cdots, y_n) \\ y_m(t_0) = y_{m0} \quad (m = 1, 2, \cdots, n) \end{cases}$$

其相应的标准四阶龙格–库塔公式为

$$y_{m,i+1} = y_{m,i} + \frac{h}{6}(K_{m1} + 2K_{m2} + 2K_{m3} + K_{m4}) \tag{5.1}$$

式中

$$\begin{cases} K_{m1} = f_m(t_i, y_{1i}, \cdots, y_{ni}) \\ K_{m2} = f_m(t_i + \frac{h}{2}, y_{1i} + \frac{h}{2}K_{11}, \cdots, y_{ni} + \frac{h}{2}K_{n1}) \\ K_{m3} = f_m(t_i + \frac{h}{2}, y_{1i} + \frac{h}{2}K_{12}, \cdots, y_{ni} + \frac{h}{2}K_{n2}) \\ K_{m4} = f_m(t_i + h, y_{1i} + hK_{1s}, \cdots, y_{ni} + hK_{n3}) \end{cases} \tag{5.2}$$

$$(m = 1, 2, \cdots, n; \quad i = 0, 1, \cdots)$$

计算顺序规定为：先利用第i步节点值y_{1i}，y_{2i}，$\cdots$，y_{ni} 按 (5.2) 式计算

$$K_{m1} \to K_{m2} \to K_{m3} \to K_{m4}$$

然后代入 (5.1) 式即可求得下一步的节点值

$$y_{1,i+1}, y_{2,i+1}, \cdots, y_{n,i+1}$$

对于$m = 2$的特殊情形，上述公式具体化为以下形式

$$\begin{cases} \dot{y} = f(t, y, z), \quad y(t_0) = y_0 \\ \dot{z} = g(t, y, z), \quad z(t_0) = z_0 \end{cases}$$

$$\begin{cases} y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ z_{i+1} = z_i + \frac{h}{6}(L_1 + 2L_2 + 2L_3 + L_4) \end{cases}$$

式中

$$\begin{cases} K_1 = f(t_i, y_i, z_i) \\ L_1 = g(t_i, y_i, z_i) \\ K_2 = f(t_i + \frac{f}{2}, y_i + \frac{f}{2}K_1, z_i + \frac{f}{2}L_1) \\ L_2 = g(t_i + \frac{f}{2}, y_i + \frac{f}{2}K_1, z_i + \frac{f}{2}L_1) \\ K_3 = f(t_i + \frac{f}{2}, y_i + \frac{f}{2}K_2, z_i + \frac{f}{2}L_2) \\ L_3 = g(t_i + \frac{f}{2}, y_i + \frac{f}{2}K_2, z_i + \frac{f}{2}L_2) \\ K_4 = f(t_i + h, y_i + hK_3, z_i + hL_3) \\ L_4 = g(t_i + h, y_i + hK_3, z_i + hL_3) \end{cases}$$

§5.4 用 MATLAB 解常微分方程

上面介绍的方法在MATLAB中都可以找到现成的指令，在使用中无需再花费时间去编程，只需要按照规定的格式调用即可。事实上，MATLAB解常微分方程的指令的功能远远超过上面介绍的内容，这些指令有：

ode45 解非刚性问题，中等精度，使用龙格–库塔法的四阶、五阶算法。大部分情况下适用，是解决问题的首选。

ode23 解非刚性问题，低精度，使用龙格–库塔法的二阶、三阶算法。适用于容差较大或适度刚性问题。

ode113 解非刚性问题，从低到高的精度，使用 PECE (Adams-Bashforth-Moulton) 法。适用于容差要求严格，或者常微分方程函数计算代价较大的情况。

ode15s 解刚性问题，低到中等精度，使用可变阶次的数值微分(NDFs)算法。在ode45计算很慢（由于刚性）或有质量矩阵时使用。

ode23s 解刚性问题，低精度，使用修正的罗森布罗克（Rosenbrock）公式。适用于对误差要求不高的情况或者有质量矩阵的情况。

ode23t 解刚性问题，低精度，用自由内插法的梯形法则。适用于要求结果无数值衰减的适度刚性问题。

ode23tb 解刚性问题，低精度，使用TR-BDF2方法。适用于对误差要求不高的刚性问题，或者含有质量矩阵的情况。

常微分方程分“刚性的”和“非刚性的”，刚性的是指其雅可比矩阵的特征值相差悬殊。两者对解法中步长选择的要求不同。后面将作更详细的说明。

以ode45为例，其使用的语句格式如下，其余指令用法与此相似：

[T, Y] = ode45(odefun, tspan, y0)

[T, Y] = ode45(odefun, tspan, y0, options)

[T, Y, TE, YE, IE] = ode45(odefun, tspan, y0, options)

sol = ode45(odefun,[t0,tf],y0,...)

其含义为：

odefun	求解的常微分方程的函数句柄
tspan	单调递增（减）的积分区间[t0, tfinal]或[t0, t1, ···, tfinal]
y0	初始条件矢量。矢量元素的排列顺序与函数中的元素顺序一致
options	用odeset建立的优化选项，如用默认值则不必输入
T, Y	T是输出的时间列矢量、矩阵Y的每个列矢量是解的一个分量
TE, YE, IE	见odeset中'events' 的说明，是事件函数的输出值
sol	以结构数组的形式表示解，用指令deval可以确定在[t0,tf]中任何一点的解

为了易于理解这些指令的用法，下面先介绍一个使用指令解方程的简单例子。

例 1 求解常微分方程

$$\ddot{x} = -x$$

初始条件为$\dot{x}\,|_{t=0} = 0, x|_{t=0} = 2$。这个方程可以理解为一个谐振子运动的方程。

令

$$y = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

将原方程用方程组表示并写成列向量或矩阵形式，这两种形式都可以用来编写表示常微分方程的函数。

$$\dot{y} = \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

在指令窗口输入

```
>> F=@(t,y)[y(2);  -y(1)]
>> ode45(F, [0, 10], [0, 2])
```

方程即被解出并输出图 5.1。

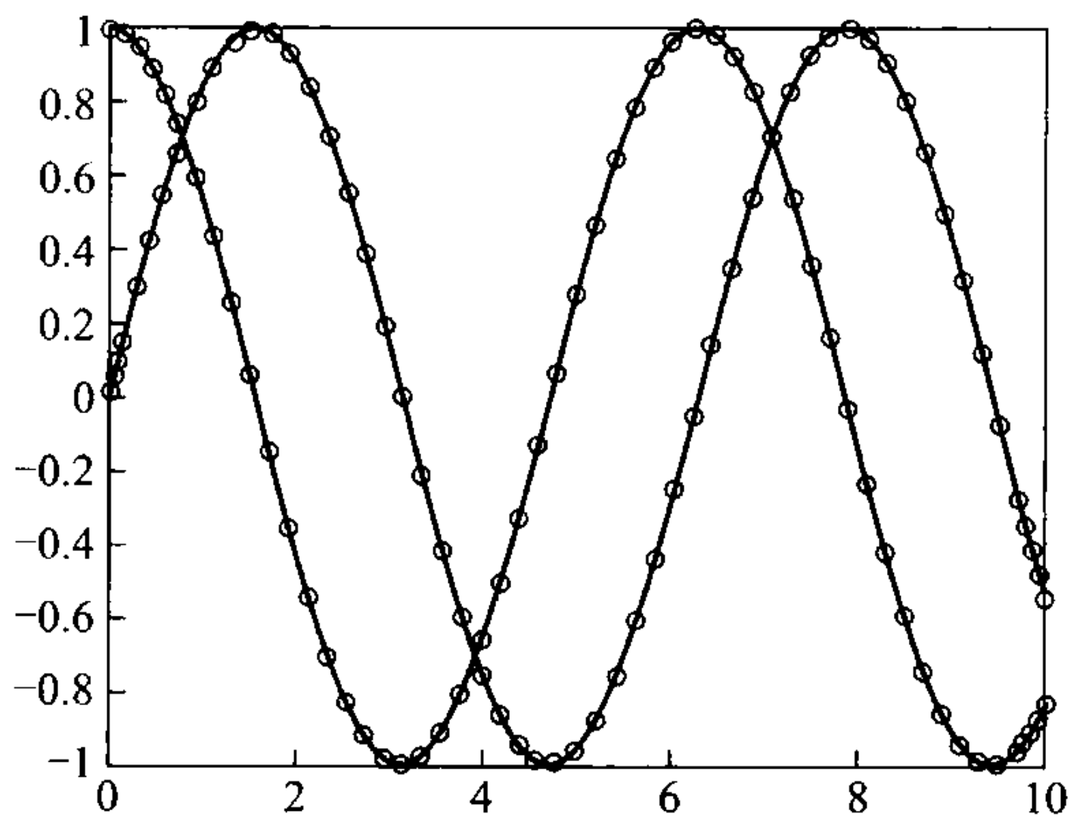


图 5.1 谐振子运动的位移与速度

在这里，第一句是将常微分方程编成匿名函数，使用的是列向量的形式；第二句是用指令求解方程并按照默认的设置输出位移与速度的图形。第一句还可以使用矩阵形式来编写匿名函数：

```
F=@(t,y)[0, 1;  -1, 0]*y
```

完整的解题过程包含三个步骤，就是将常微分方程编写为函数，设置求解条件，调用指令求解。分述如下：

1. 将微分方程表示为函数

在上面例子中是用匿名函数来表示常微分方程，这是MATLAB7的新功能，在MATLAB6中是用内嵌函数来编写，也有两种方式：

```
inline('[y(2);-y(1)]','t','y')
inline('[0, 1;  -1, 0]*y ]','t','y')
```

更复杂的方式是使用子函数文件来编写常微分方程，在主程序中使用函数句柄调用这个子函数，这种方法的好处是在MATLAB的各个版本都能使用。所解的常微分方程必须改写成一阶常微分方程组，然后编写成如下列格式的子函数文件，它与主程序同在一个文件内：

function ydot = odefun(t,y)

ydot = [在括号内插入t 和/或y的函数]

在这里，t 是标量，y 是列矢量，ydot 是列矢量 dy/dt。

子函数文件中的第一行的等号后面的odefun是用户命名的子函数文件名。主程序文件也必须是函数文件，也就是第一行要以function开头，才能在其中使用子函数文件。而前面两种方法是可以用脚本文件来编写程序，这是很大的区别。

前面的例子可以编写成如下程序：

```
function xzz
ode45(@F,[0,10],[0,2]);
```

```
function ydot=F(t,y)
ydot=[y(2);
     -y(1) ];
```

无论在哪一种方式中，变量 t，y 的顺序不能改变，t 必须是第一个变量，即使它没有显式地出现在方程中。函数 y 是个矢量，其中的各个分量的顺序在所有出现的地方都相同，例如作为初始条件输入 y_0时，各个分量的顺序也与它相同。

2. 给定解方程的条件与要求

在例子中我们是使用指令中默认的条件来解方程，事实上，这些条件是可以根据用户的要求来改变的。指令odeset用来建立或改变解方程的条件和要求，指令odeget则用来获取“选项结构数组(options structure)”中的信息，也就是现在使用的解方程的条件与要求。关于结构数组参见前面的介绍。在MATLAB中，指令默认的求解条件和要求可以用指令odeset显示，下面是全部可设置的参数及其默认值（括号{}中的值）：

```
          AbsTol: [ positive scalar or vector {1e-6} ]
          RelTol: [ positive scalar {1e-3} ]
     NormControl: [ on | {off} ]
       OutputFcn: [ function ]
       OutputSel: [ vector of integers ]
          Refine: [ positive integer ]
           Stats: [ on | {off} ]
     InitialStep: [ positive scalar ]
         MaxStep: [ positive scalar ]
             BDF: [ on | {off} ]
        MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
        Jacobian: [ matrix | function ]
        JPattern: [ sparse matrix ]
      Vectorized: [ on | {off} ]
            Mass: [ matrix | function ]
MStateDependence: [ none | weak | strong ]
        MvPattern: [ sparse matrix ]
     MassSingular: [ yes | no | {maybe} ]
     InitialSlope: [ vector ]
           Events: [ function ]
```

键入help odeset可查看全部参数的说明，这些选项可分为以下几类：

分类目录	选项名称
误差控制	RelTol, AbsTol, NormControl
解的输出	OutputFcn, OutputSel, Refine, Stats
雅可比矩阵	Jacobian, JPattern, Vectorized
步长	InitialStep, MaxStep
质量矩阵和DAEs	Mass, MStateDependence, MvPattern, MassSingular, InitialSlope
事件控制	Events
ode15s专用指令	MaxOrder, BDF

下面只对其中的几个参数作简单说明。

RelTol：相对误差，默认值为1e-3。

AbsTol：绝对误差，默认值为1e-6。

OutputFcn：输出方式，默认值为'odeplot'，其他可选项有：

odeplot：按时间顺序画出全部变量的解；

odephas2：二维相空间中两个变量的图形；

odephas3：三维相空间中三个变量的图形；

odeprint：打印输出。

为了满足解微分方程的某些特殊要求，可以用odeset修改它。指令odeset用法语句格式如下：

options = odeset('name1', value1, 'name2', value2, ⋯)

options = odeset(oldopts, 'name1', value1, ⋯)

options = odeset(oldopts, newopts)

odeset

各项符号的含义为：

options	"选项结构数组"的名称
'name1',value1，'name2',value2	选项中的参数 1，2 及其取值
oldopts,newopts	旧的选项，新的选项
optimfunction	使用默认的选项

第一种格式是指定各个参数的取值，对不指定取值的参数，取默认值。在不引起混淆的情况下，参数名可以只键入前面的几个字母，也不必区分大小写，如用'abst'表示'AbsTol'。但数值的输入必须格式正确，否则仍然采用默认值。

第二种格式使用了原来的优化选项，但对其中的参数1等指定了新值。

第三种格式合并了两个优化选项 oldopts 和 newopts，重复部分取 newopts 的指定值。

3. 调用指令求解并输出结果

首先要根据方程的性质来选用前面所介绍的指令。刚性方程的内容会在下

面介绍，本书的例子都可以用ode45来求解。调用指令时要严格按照前面介绍的语句格式代入各种参数，同时要注意选择解的输出方式。例如下面改变了输入的时间变量的表达形式，并以结构数组形式输出方程的解。

```
>> sol=ode45(F,[0:0.1:10],[0,1])
sol = olver: 'ode45'
    extdata: [1x1 struct]
          x: [1x19 double]
          y: [2x19 double]
      stats: [1x1 struct]
      idata: [1x1 struct]
```

在输出的结构数组中查找数据要使用指令deval,

```
>> ss=deval(sol,5.15)
    ss = -0.90562
          0.42439
```

如果这个数据是解中原来没有的，就像这里的$t = 5.15$，那么MATLAB会自动用插值的方法计算它。

如果不采用默认的求解条件，就要把自己设置的求解条件代入。例如要输出相图，就要改变原来的输出的设置，操作如下：

```
>> options=odeset('OutputFcn','odephas2');
>> ode45(F,[0,10],[0,1],options)
```

这时输出的图形如图 5.2 所示。

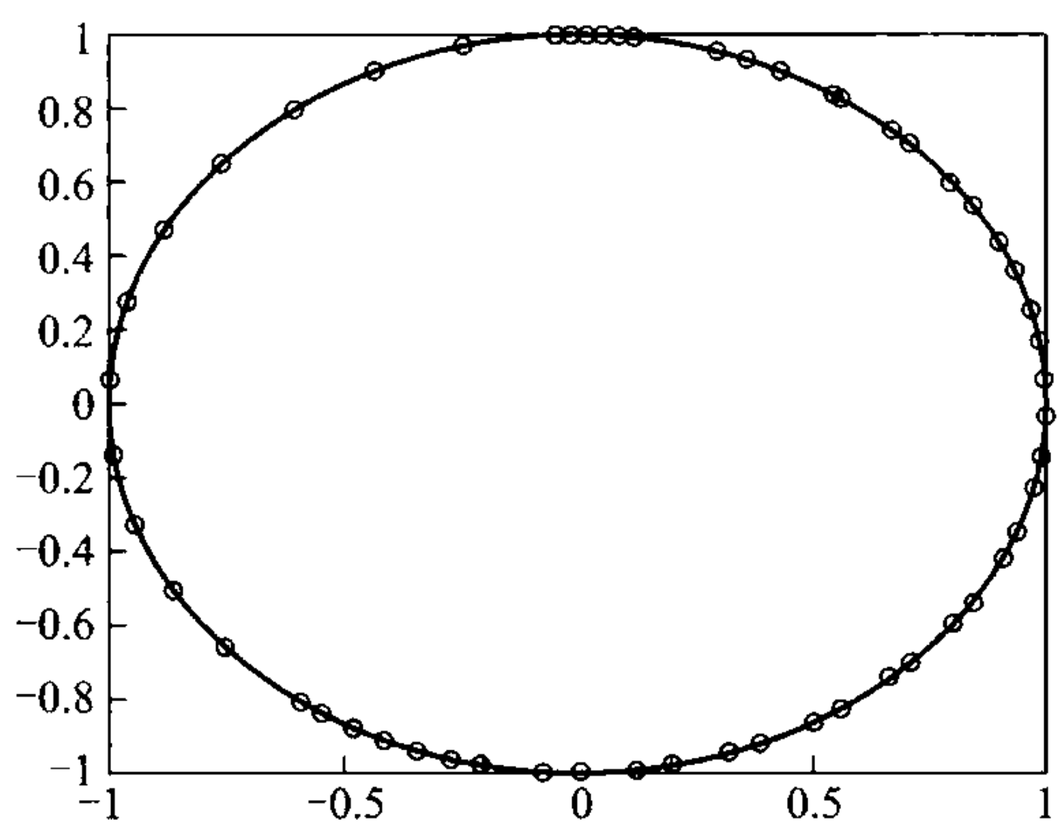


图 5.2 谐振子的相图

要查看结构数组sol中某个域比如域x的内容，可键入sol.x。

MATLAB的指令不仅可以解单变量的常微分方程，也可以解多变量的常微分方程组。指令的用法基本相似，就是将各个变量的各阶导数按顺序排列成一

个列向量，并且在程序中始终保持这个顺序。下面就是这样一个例子，它是受空气阻尼的抛体运动，虽然问题不复杂，但它是有两个变量的方程组。

例 2 研究受空气阻尼的抛体运动。抛体质量为m，初速度为$\boldsymbol{v}_0$，所受空气阻力的大小R与速率v的n次方成正比，即$R = bv^n$，其中b是阻尼系数。

以地面为参考系，以抛出点为原点O建立直角坐标系Oxy，Ox沿水平方向，Oy竖直向上。质点受重力和空气阻力作用，而空气阻力包括三种情况。质点的运动微分方程可统一表示为

$$m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = -m\boldsymbol{g} - b|v|^p\boldsymbol{v}$$

空气阻力的三种情况分别对应方程中参数值为

$$b = [0,\ 0.1,\ 0.1],\quad p = [0,\ 0,\ 1]$$

令 $y(1) = x$，$y(2) = \mathrm{d}x/\mathrm{d}t$，$y(3) = y$，$y(4) = \mathrm{d}y/\mathrm{d}t$，将方程写成一阶微分方程组

$$\begin{cases} \dfrac{\mathrm{d}y(1)}{\mathrm{d}t} = y(2) \\ \dfrac{\mathrm{d}y(2)}{\mathrm{d}t} = -\dfrac{by(2)}{m}(y^2(2) + y^2(4))^{p/2} \\ \dfrac{\mathrm{d}y(3)}{\mathrm{d}t} = y(4) \\ \dfrac{\mathrm{d}y(4)}{\mathrm{d}t} = -g - \dfrac{by(4)}{m}(y^2(2) + y^2(4))^{p/2} \end{cases}$$

再用指令ode45解常微分方程。主程序为：

```
function znxp
global  m d e                                   %用全局变量传递参数
m=1;b=[0,0.2,0.2];p=[0,0,1];                     %传递给方程的参数
px=[4.6;4.5;4.5];                                %说明文字的x坐标
py=[3.5;1.8;0.4];                                %说明文字的y坐标
strdd{1}='无阻尼';
strdd{2}='阻力正比于v';
strdd{3}='阻力正比于v^2';
figure
for i=1:3                                       %三次循环计算三种条件下的运动
d=b(i);  e=p(i);                                 %选定传递参数的值
[t,y]=ode45(@znxpfun,[0:0.01:10],[0,3,0,5]);
H{i}=max(y(:,3))                                 %轨迹最高点
```

```
T{i}=t(find(y(:,3)==H{i}))          %用关系运算找出到达最高点时间
vx0{i}=y(end,2)                     %到最高点速度
subplot(2,1,1)                      %分区画图
axis([0 6 -70 2]);                  %确定坐标范围
hold on                             %保留图形
xlabel('x');    ylabel('y');        %x,y轴的标注
comet(y(:,1),y(:,3));               %粒子的空间轨迹
subplot(2,1,2)
axis([0 10 0 4]),     hold on
xlabel('t'),    ylabel('dx/dt')
text(px(i),py(i),strdd{i});
                                    %给定文字起始位置并调用基元列阵加注文字
comet(t,y(:,2))                     %速度随时间变化图
end
function ydot=znxpfun(t,y)          %待解的常微分方程
global m d e
    ydot=[y(2);
    -d/m*y(2)*(y(2).^2+y(4).^2)^(e/2);
    y(4);
    -9.8-d/m*y(4)*(y(2).^2+y(4).^2)^(e/2)];
```

画出的图形如图 5.3。所用的程序有如下特色：

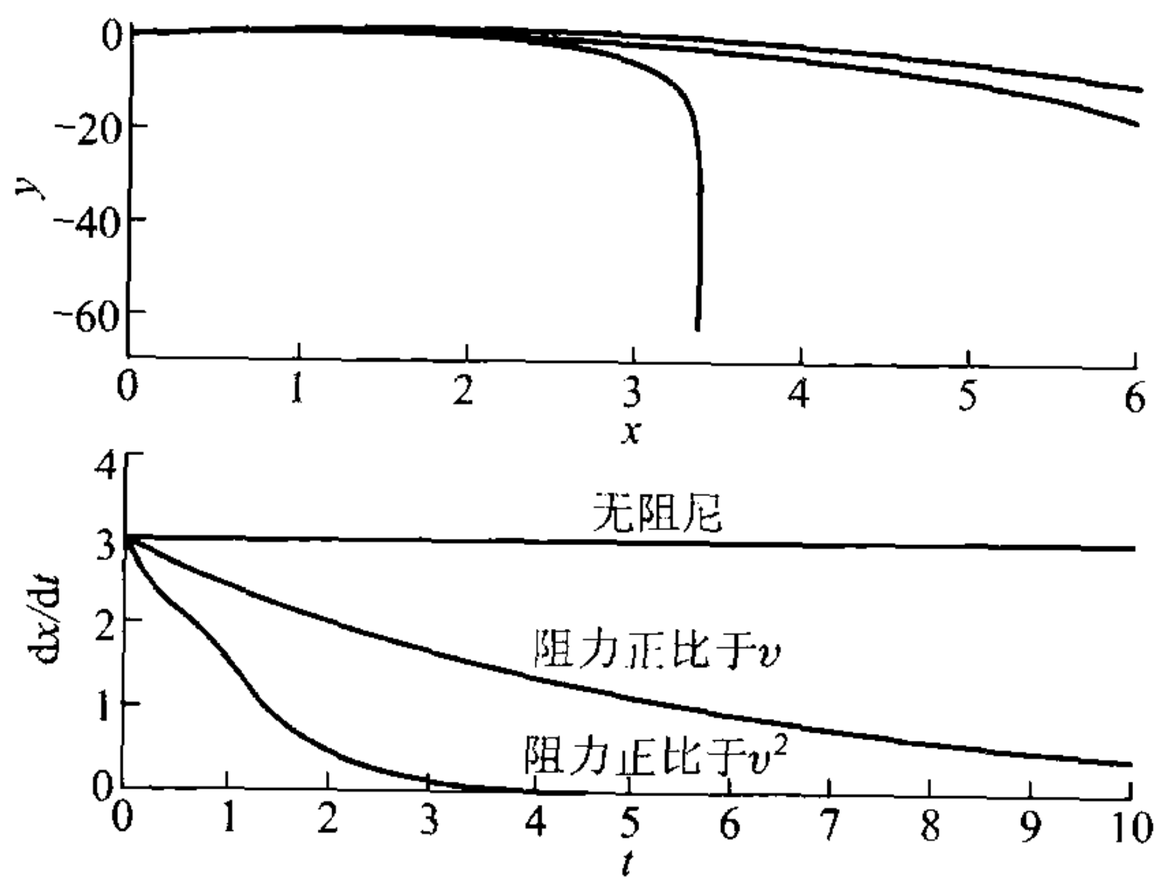


图 5.3 有阻尼的抛体运动

(1) 用参量传递改变计算问题。

(2) 运用关系运算查找数据。

(3) 运用cell矩阵进行图形标注。

(4) 分区作图。

程序中用到了基元列阵。基元列阵也是一种MATLAB特有的数据形式。

§5.5 刚性问题

如果在求解一个问题时，其解的产生变得越来越慢，而变量变动很小时解的变化却很快，以至于数值计算必须采用小步长才能获得满意的结果，这样的问题称之为刚性问题。

刚性问题实际上是个效率问题，如果不在乎计算时间的多少，可以不理会刚性的影响。非刚性的方法也能求解刚性问题，只是多花些时间。这里借用Moler书中的例子来说明刚性问题的计算。

火焰蔓延的模型是一个例子。当点燃一根火柴时，火焰迅速增大到一个临界体积，然后维持这一体积不变。此时，火球内部燃烧耗费的氧气与其表面的氧气达到了一种平衡。简化的模型是

$$\begin{aligned}&\dot{y} = y^3 - y^2\\&y(0) = \delta\\&0 \leqslant t \leqslant 2/\delta\end{aligned}$$

标量$y(t)$代表了火球的半径，y^2和y^3项源自面积和体积。其中关键的参数是比较“小”的初始半径 δ，这里求解所用的计算时间和 δ 成正比。在 δ 不是很小时，这个问题还不算非常刚性。例如取$\delta = 0.01$，相对误差为10^{-4}。求解过程如下：

```
delta = 0.01;
F=inline('y^2 - y^3', 't',  'y');
opts=odeset('RelTol',1.e-4);
ode45(F,[0 2/delta],delta,opts);
```

输出的解的图 5.4 从$y = 0.01$开始，以一个适度增大的速度增长，直到时间 t 达到100，也就是$1/\delta$，然后增加很快，达到一个很接近1的值，并保持不变。

把δ减小两个数量级，问题表现出刚性性质：

```
delta=0.0001;
ode45(F,[0 2/delta],delta,opts);
```

这时解的绘图过程变得很慢。图 5.5 是把图 5.4 放大的结果，可以看到解的误差范围在10^{-4} 左右波动，但是很难达到稳定。

这个问题在开始时并不是刚性的，它只是在解接近稳态时变为刚性。这是由于稳态解过于严格，任何接近$y(t) = 1$的解，都会很快地增大或缩小。

解决刚性问题的办法是改变数值计算方法，所有刚性方法都是隐式的，每步都要调用MATLAB的矩阵运算，求解联立的线性方程组，来预测解的变化，所

以要做更多的工作，才能有把握取较大的步长。

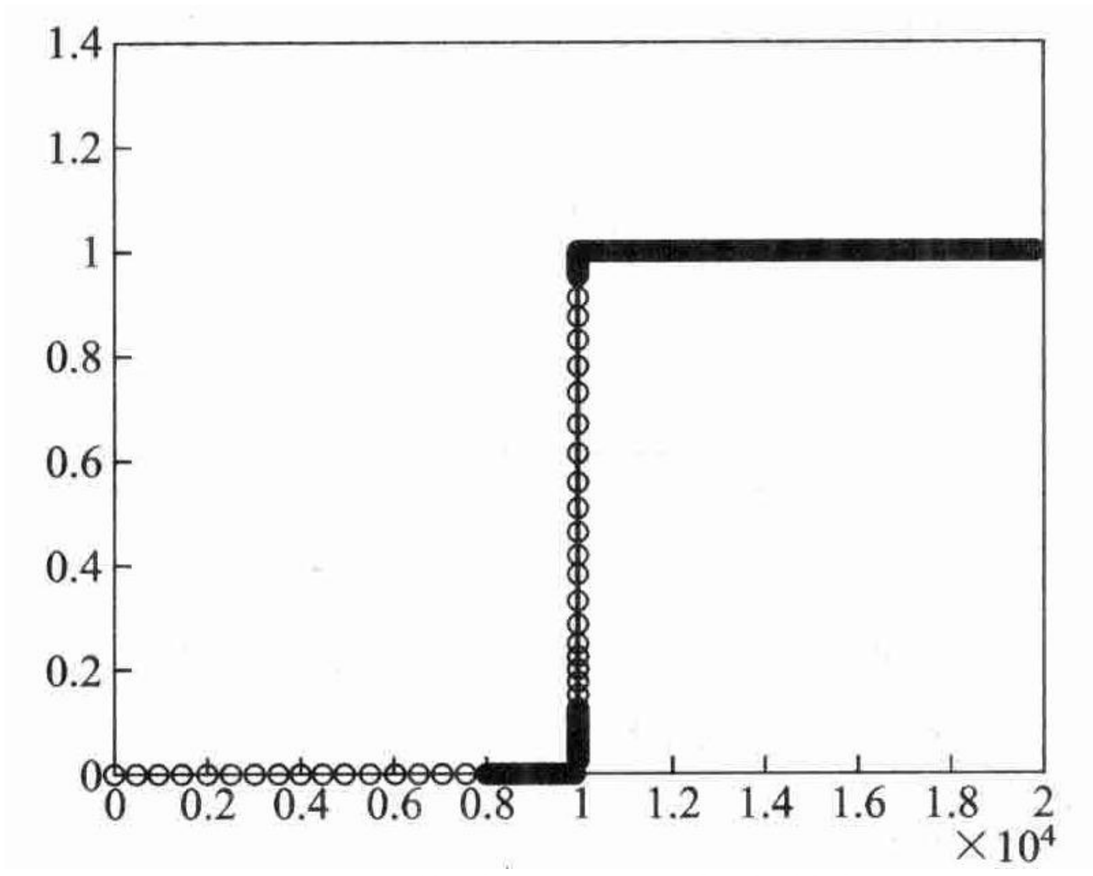


图 5.4 指令 ode45 求出的解

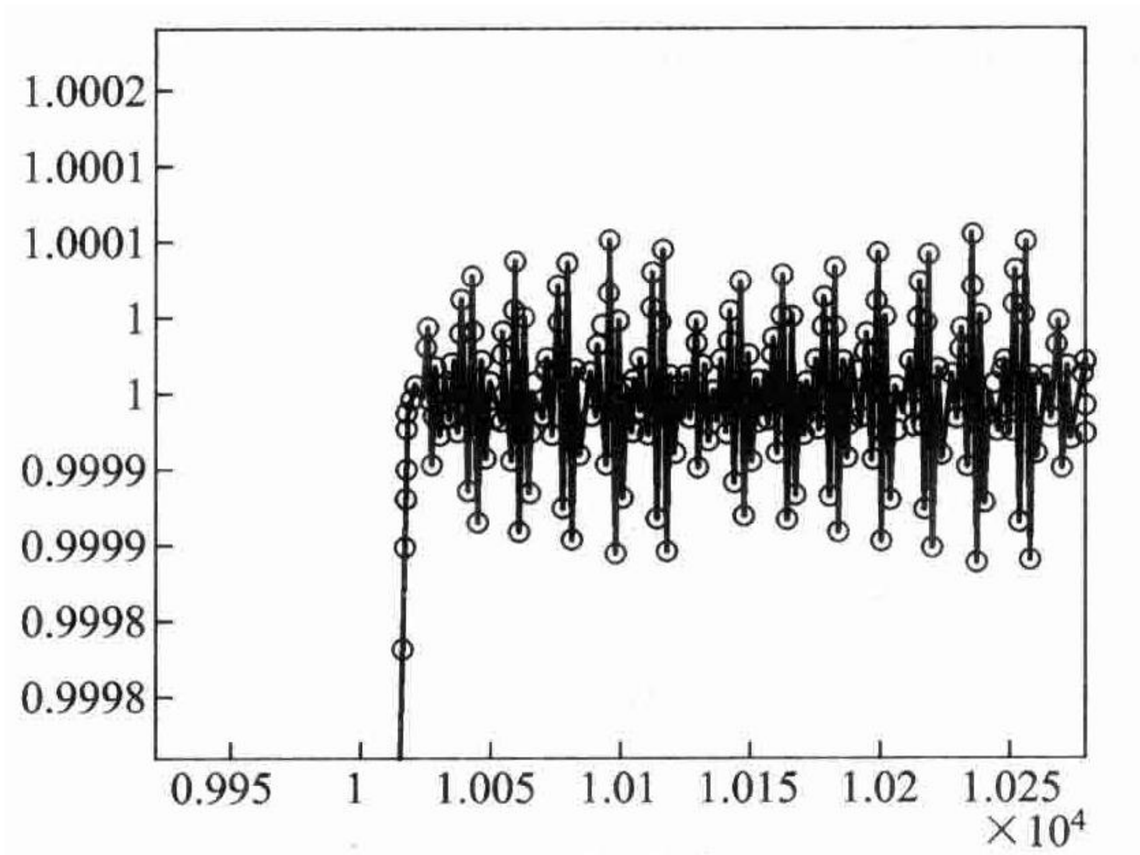


图 5.5 局部放大图

现在改用 ode23s 来解火焰问题，“s”是 stiff（刚性）的意思。

```
delta = 0.0001;
ode23s(F,[0  2/delta], delta, opts);
```

图中显示了计算的结果 (图 5.6) 和放大的图形 (图 5.7) 的细节。可以看出，ode23s 使用的步数比 ode45 大大减少，ode23s 只用了 90 步和 412 个函数求值，而 ode45 用了 3040 步和 20179 个函数求值。刚性也会影响到作图，ode45 的绘图文件比 ode23s 的大得多。

想象一下，从山上长途步行回来，行走在一个两边都是陡峭斜坡的狭窄的峡谷中。显式的方法简单地通过局部的斜率来寻找下降的方向。但是跟随两边轨迹的斜率会使你反复往返于峡谷的谷底，就像ode45那样。虽然最终还是可以回到家，但一定是天黑之后很久才到家。而隐式的方法使你紧盯山路并预测每步该如何走，于是少走了很多路，总的来说，这种额外的专心是非常值得的。

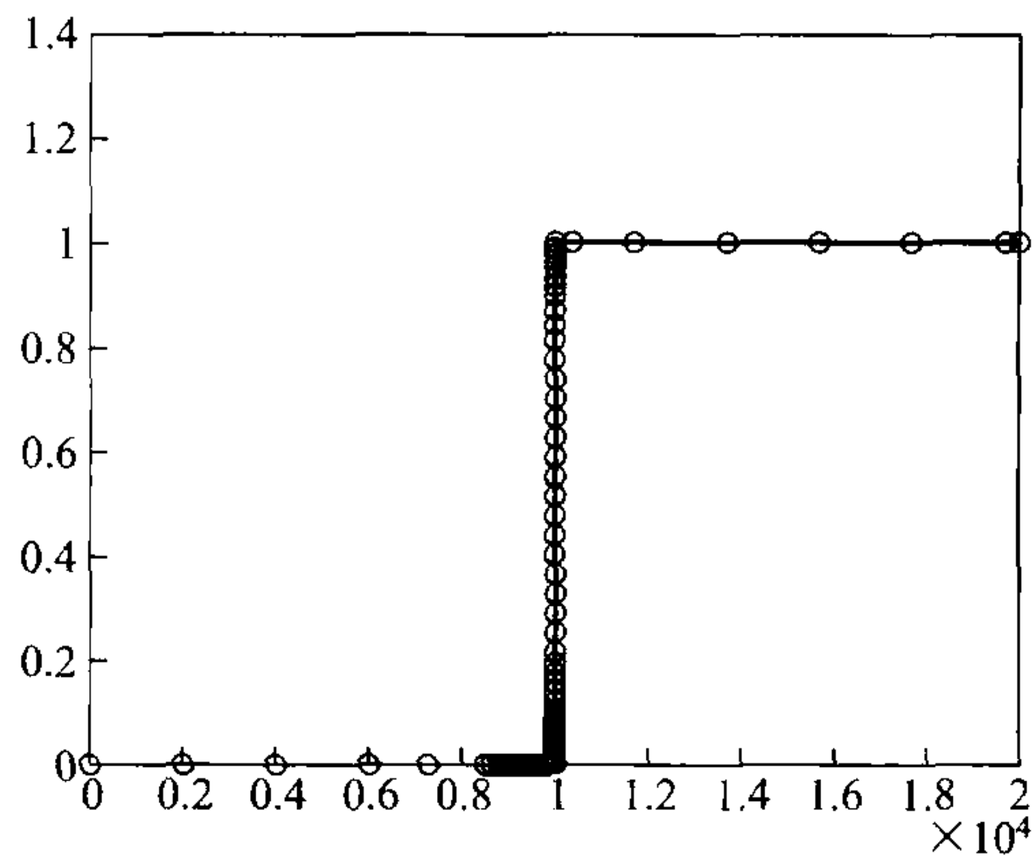


图 5.6 指令 ode23s 求出的解

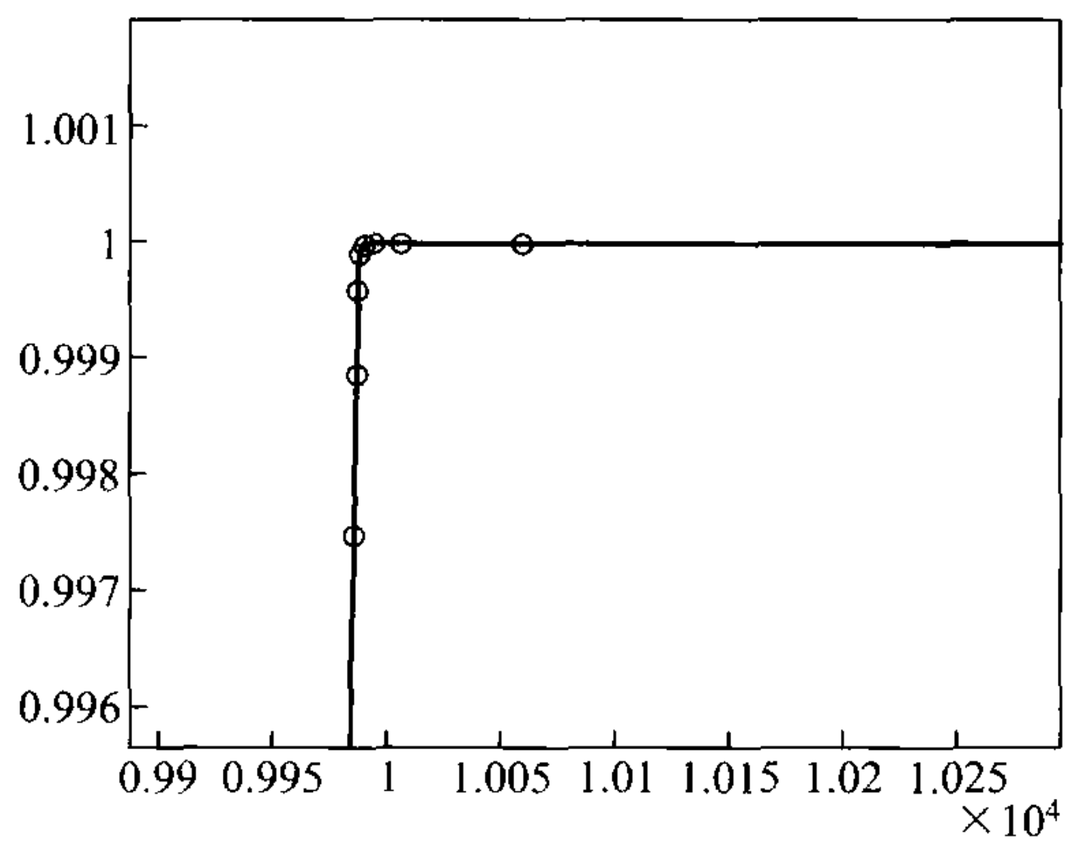


图 5.7 局部放大图

火焰问题是有精确解的，它涉及兰伯特 (Lambert) W 函数 $W(z)$。这个微分方程是可分离变量的，积分一次得到 $y(t)$ 一个隐式方程

$$\frac{1}{y}+\log\left(\frac{1}{y}-1\right)=\frac{1}{\delta}+\log\left(\frac{1}{\delta}-1\right)-t$$

这个方程可以解出y。可以证明这个火焰模型的精确解析解是

$$y(t)=\frac{1}{W\left(a\mathrm{e}^{a-t}\right)+1}$$

其中$a=\dfrac{1}{\delta}-1$。兰伯特W函数 $W(z)$ 是下面方程的解

$$W(z)\mathrm{e}^{w(z)}=z$$

用符号工具箱求解结果如下，图形如图 5.8 所示。

```
>> y=dsolve('Dy=y^2 - y^3', 'y(0 ) =0.01');
>> y=simplify(y);
>> pretty(y)

                1
 ----------------------------
 lambertw(99 exp(99-t))+1
>> ezplot(y,0,200)
```

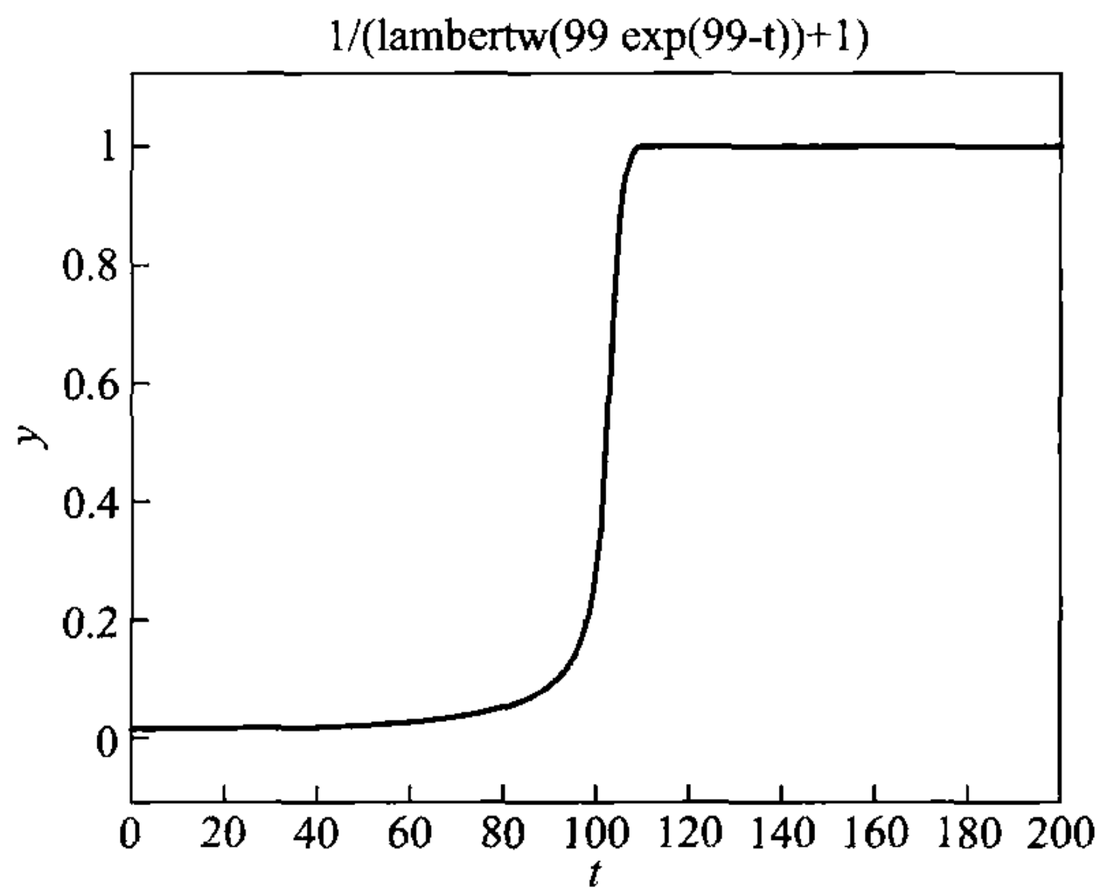


图 5.8 符号工具箱求出的解

如果稍微减小初始值1/100并适当扩大时间段t的范围，则画出的图形中过渡区域会变得更窄。

§5.6 事 件

到目前为止，我们都是在一个已知的时间段$0 \leqslant t \leqslant t_{\text{final}}$求解问题，但是有时候，在开始计算时$t_{\text{final}}$还不能确定，例如小球落地以后弹跳问题，就不能事先确定小球到达地面的时间。MATLAB有一个功能叫事件（events），可以用来解决这类问题。

举一个例子，在重力作用下小球下落的过程中存在空气阻力，求它落到地面的时间。简化以后的模型是

$$\ddot{y} = -1 + \dot{y}^2$$

初始条件为$y(0) = 1$，$\dot{y}(0) = 0$。求$y(t) = 0$时t的值等于多少。方程可以写成如下函数文件

```
function  ydot = f(t, y)
```

```
ydot = [y(2);  -1 + y(2)^2];
```

还要建立一个函数$g(t,y)$来检测$y(t)=0$发生的时间。

```
function  [gstop, isterminal,direction] = g(t, y)
gstop = y(1);
isterminal = 1;
direction = [ ];
```

这个函数是仿照MATLAB内部建立的函数events的格式建立的。调用ode45时用opts将'events'选项设为'on'，表示启动事件开关，用它来控制积分中断的时刻。而'events'的值从子函数文件$g(t,y)$中输出。在子函数文件中，判断事件发生的变量是高度为零，所以取gstop = y(1)，direction表示该变量变化的方向，当direction取1表示判断事件的变量从负值增加到零，取−1则从正值减小到零，取0则不考虑方向。参数isterminal取1表示一旦gstop为零，就中断积分，如取0则不中断积分。y的两个分量分别是高度和速度，程序中的设置是当高度减小到零时，积分终止，即isterminal = 1，而direction不取任何值表示不必考虑变化方向。这些取值列表如下：

取值 / 指令	−1	0	1
direction	由正变到零	不考虑方向	由负变到零
isterminal		不中断积分	中断积分

这两个函数文件建立以后，用下面的程序去调用：

```
opts=odeset('events',@g);                          %启动事件函数
y0 = [1; 0];                                       %初值
[t,y,tfinal] = ode45(@f, [0, Inf], y0, opts );
tfinal                                             %输出物体落地的时间
plot(t,y(:,1),'-',[0,tfinal],[1 0], 'o')
axis([-0.1  tfinal+0.1  -0.1  1.1])                %坐标范围
xlabel('t')                                        %标示坐标变量
ylabel('y')
title('Falling body')                              %标注图名
text(1.2,0,['tfinal='num2str(tfinal)])
                                                   %将数值变成字符用以标注落地时间
```

运行结果输出的落地时间为1.6585，图形如图 5.9 所示。

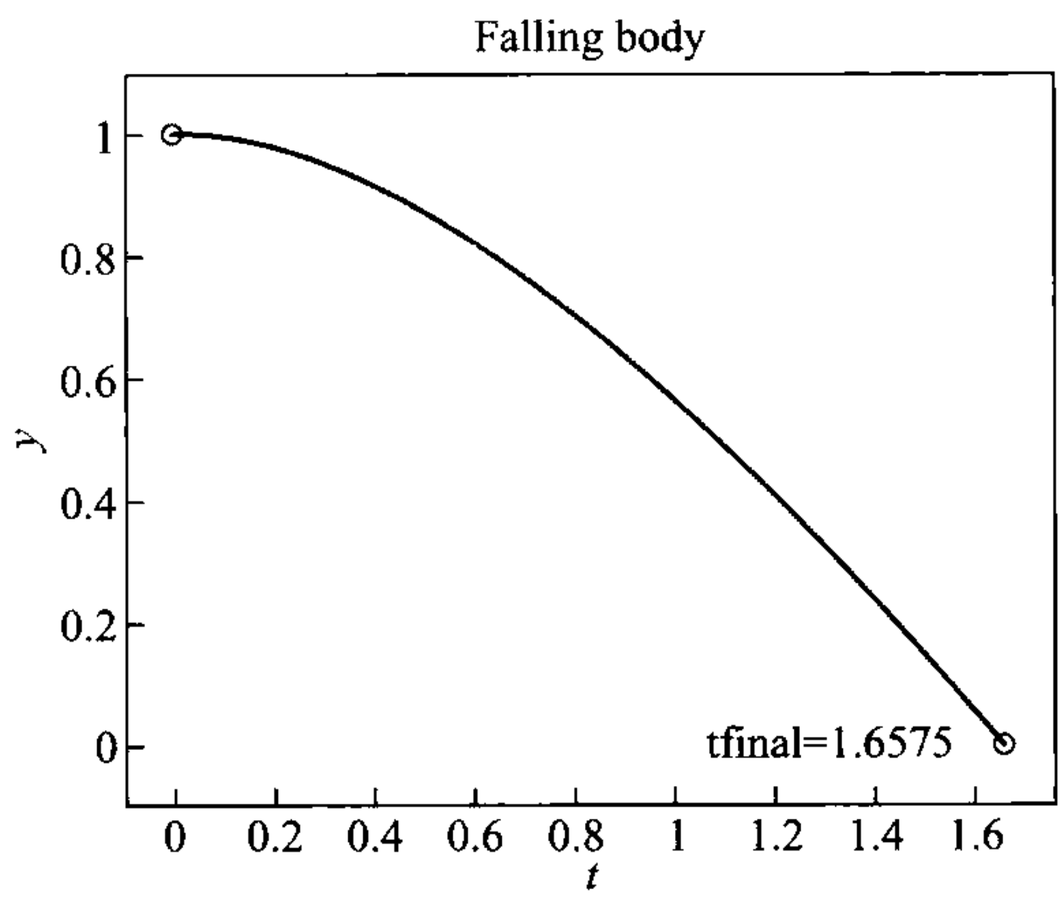


图 5.9 用 events 计算小球落地的时间

MATLAB 有一个程序演示小球落地弹跳逐渐衰减的过程，运行指令是 ballode。程序就用到了事件函数 events。程序编写规范，值得借鉴，其中用到了逻辑函数 ishold，还随时对图形句柄的属性加以改动，对深入了解 MATLAB 的功能很有帮助，列在下面供有兴趣的读者参考。

例 3 小球以某个初速度从地面跳起，落地后发生非弹性碰撞，速度减至原来的 90%，计算它经过 10 次弹跳的运动轨迹。

程序如下，画出图形如图 5.10 所示。

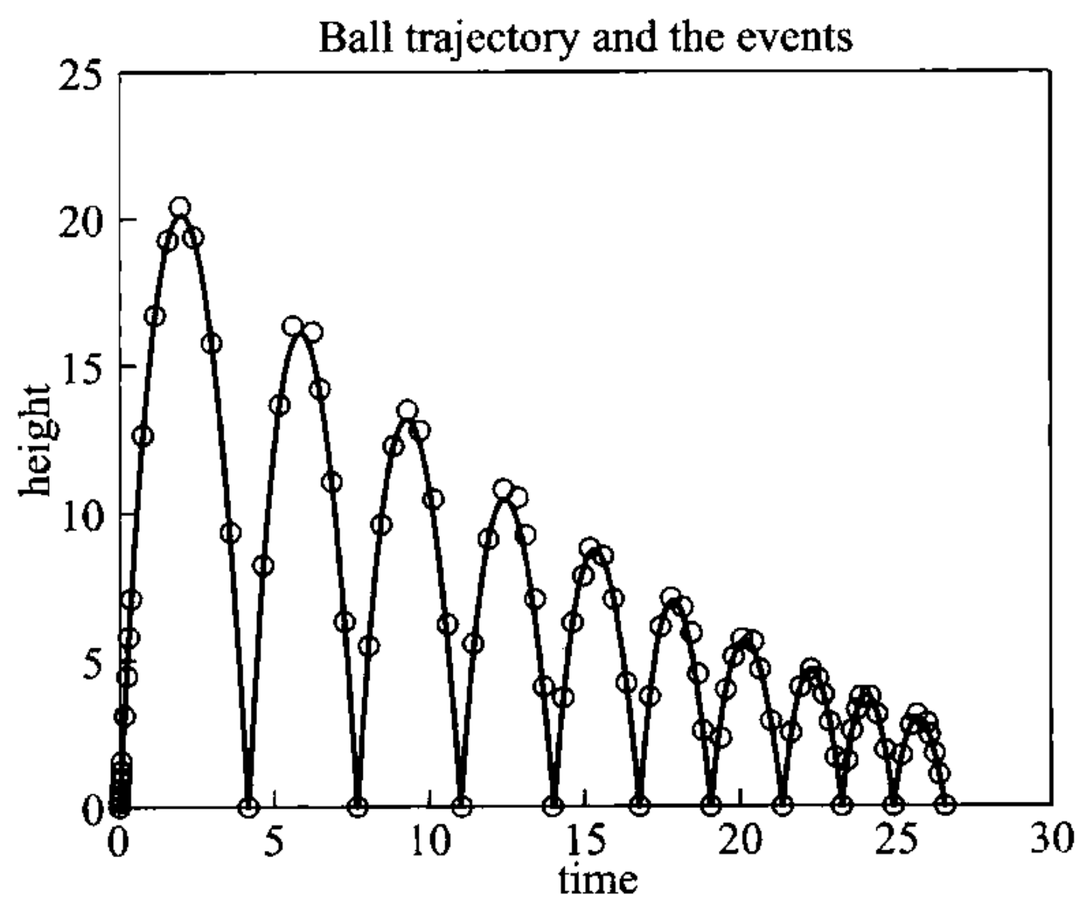


图 5.10 小球在地面的弹跳运动

```
function  ballode
%  BALLODE Run a demo of a bouncing ball.
%  This is an example of repeated event location, where the
%  initial conditions are changed after each terminal event.
%  This
```

```
% demo computes ten bounces with calls to ODE23.  The speed of
% the ball is attenuated by 0.9 after each bounce.
% The trajectory is plotted using the output function ODEPLOT.
%
% See also ODE23, ODE45, ODESET, ODEPLOT, @.
% Mark W. Reichelt and Lawrence F. Shampine, 1/3/95
% Copyright 1984-2001 The MathWorks, Inc.
%  $ Revision:  1.16 $ $Date:  2001/04/15 12:02:54 $
tstart = 0;  tfinal = 30;  y0 = [0; 20];  refine = 4;
options = odeset('Events',@events,'OutputFcn',@odeplot,
'OutputSel',1,···'Refine',refine);
figure;
set(gca, 'xlim', [0 30], 'ylim', [0 25]);
box on
hold on;
tout=tstart; yout=y0.'; teout=[ ]; yeout=[ ]; ieout=[ ];
for i = 1:10
    % Solve until the first terminal event.
    [t,y,te,ye,ie] = ode23(@f,[tstart tfinal],y0,options);
    if ~ishold
      hold on
    end
    % Accumulate output.  This could be passed out as output
    % arguments.
    nt = length(t);
    tout = [tout; t(2:nt)];  yout = [yout; y(2:nt,:)];
    teout = [teout; te];  % Events at tstart are never reported.
    yeout = [yeout; ye];  ieout = [ieout; ie];
    ud = get(gcf, 'UserData');
    if ud.stop
      break;
    end
    % Set the new initial conditions, with .9 attenuation.
    y0(1) = 0;
    y0(2) = -.9*y(nt,2);
```

```
    % A good guess of a valid first timestep is the length of the
    % last valid timestep, so use it for faster computation.
    % 'refine' is 4 bydefault.
    options = odeset(options,'InitialStep',t(nt)-t(nt-refine),
    'MaxStep',t(nt)- t(1));
    tstart = t(nt);
end
plot(teout,yeout(:,1),'ro')
xlabel('time');
ylabel('height');
title('Ball trajectory and the events');
hold off
odeplot([ ],[ ],'done');
% ------------------------------------
function dydt = f(t,y)
dydt = [y(2); -9.8];
% ------------------------------------
function [value,isterminal,direction] = events(t,y)
% Locate the time when height passes through zero
% in a decreasing direction and stop integration.
value = y(1);       % detect height = 0
isterminal = 1;     % stop the integration
direction = -1;     % negative direction
```

程序中用options将'events'选项设为'on'，表示启动事件开关，用它来控制积分中断的时刻。在这个程序中，y的两个分量分别是高度和速度，判断事件的变量是高度，即子函数文件中取value = y(1)，程序中的设置是当高度由正值减小到零时，积分终止，即isterminal = 1，direction = −1，这时就完成一次落地的计算，然后在主程序中重新给出新的初始条件，即考虑非弹性碰撞后开始新的一次弹跳。

§5.7 误 差

初值问题在数值求解过程中的误差有两个来源：离散化误差和舍入误差。

离散化误差是微分方程和数值方法的一种性质。如果算术计算可以在无限精度下完成，那么结果只会存在离散化误差。舍入误差是和计算机硬件及程序相关的，通常情况下远远小于前者，除非需要非常高的精度，一般不需要考虑。

离散化误差可以从全局和局部两个方面分析，假定计算中忽略舍入误差。局部离散化误差是在一步计算中所产生的误差，用方程来表示就是，如果t_n时刻的计算值为y_n，根据下列方程

$$\dot{u}_n = f(t, u_n)$$

$$u_n(t_n) = y_n$$

可以计算出t_{n+1}时刻的值y_{n+1}，局部离散化误差就是

$$d_n = y_{n+1} - u_n(t_{n+1})$$

而全局离散化误差是从t_0处的初始条件开始，计算到某时刻的计算值y_n，与该处的理论值$y(t_n)$之差

$$e_n = y_n - y(t_n)$$

当$f(t, y)$ 实际与y无关时，很容易看出局部与全局离散化误差的区别。此时解只是一个简单的积分

$$y(t) = \int_{t_0}^{t} f(\tau)\mathrm{d}\tau$$

按照矩形积分公式

$$\int_{t_0}^{t_N} f(\tau)\mathrm{d}\tau \approx \sum_{n=0}^{N-1} h_n f(t_n)$$

在子区间上局部离散化误差是

$$d_n = h_n f(t_n) - \int_{t_n}^{t_{n+1}} f(\tau)\mathrm{d}\tau$$

而全局离散化误差是整个误差

$$e_n = \sum_{n=0}^{N-1} h_n f(t_n) - \int_{t_0}^{t_N} f(\tau)\mathrm{d}\tau$$

在这种特殊情况下每步子积分过程都是独立的，也就是可以按照任意次序求和，所以此时全局误差为局部误差的简单累加

$$e_n = \sum_{n=0}^{N-1} d_n$$

但在实际微分方程中，$f(t, y)$和y相关，所以每一步计算的误差都和前面的结果有关。因此，在全局误差与局部误差的关系与微分方程的稳定性密切相关。对于仅有一个标量变量的方程，如果偏导数$\partial f/\partial y$为正，那么解$y(t)$随着t的增大而增大，全局误差比局部误差的累加和要大。反之，如果$\partial f/\partial y$为负，全局误差要比

局部误差之和小。如果$\partial f/\partial y$ 的符号不定，或者对非线性方程$\partial f/\partial y$是一个变化的矩阵，那么全局误差e_n和d_n的累加和之间的关系非常复杂，难以预测。

在MATLAB的指令中，只尝试对局部误差进行控制。因为对全局误差进行控制，会使求解程序非常复杂，运行代价高昂。

衡量一个数值方法精度的基本概念是阶数（order），它是通过把数值方法应用到具有光滑解的问题时所获得的局部误差来定义的。一个方法具有p阶精度是指，存在一个常数C满足

$$|d_n| \leqslant Ch_n^{p+1}$$

数值C可能和定义微分方程的函数的偏导数以及求解区间长度有关，但它应该独立于步数n和步长h。上面的不等式常表示成

$$d_n = O(h_n^{p+1})$$

以欧拉法为例来说明：

$$y_{n+1} = y_n + h_n f(t_n, y_n)$$

假设局部解$u_n(t)$具有连续的二阶导数，那么在点t_n处用泰勒展开得到

$$u_n(t) = u_n(t_n) + (t - t_n)u_n'(t_n) + O((t - t_n)^2)$$

使用定义$u_n(t)$的微分方程和初始条件有

$$u_n(t_{n+1}) = y_n + h_n f(t_n, y_n) + O(h_n^2)$$

$$d_n = y_{n+1} - u_n(t_{n+1}) = O(h_n^2)$$

于是可以得出$p = 1$,即欧拉法具有一阶精度。按照MATLAB中对常微分方程求解程序的命名规则，这种使用欧拉法自身，固定步长，没有误差控制的方法应当称为ode1。

现在考虑在固定点$t = t_f$处的全局离散化误差。随着对精度要求的提高，步长h会减小。为了使到达t_f 处需要的步数N增大，可以简单地认为

$$N = \frac{t_f - t_0}{h}$$

其中h是平均步长，此外，全局误差e_N可以表示成N个局部误差的加权和，权重因子是描述方程稳定性的因子。这些因子与步长没有太大的关系，从而可以近似认为如果局部误差为$O(h^{p+1})$，那么全局误差为$N \cdot O(h^{p+1}) = O(h^p)$。这就是为什么使用$p+1$而不是使用$p$作为指数来定义阶数的原因。

对于欧拉法，$p=1$，即步长减小为$1/2$，会使误差降低为原来的$1/2^{p+1}=1/4$，但使到达t_f所需要的步数增大两倍，进而全局误差只减小到原来的$1/2^p=1/2$。可见，对于高阶的方法，平滑解的全局误差会以大得多的倍数降低。

这里需要指出的是，在讨论常微分方程的数值解法时，“阶数”一词有多种不同的含义。微分方程的阶数，是出现的阶导数的指数，比如$\mathrm{d}^2y/\mathrm{d}t^2=-y$就是一个二阶的微分方程。而方程系统的阶数通常指方程的个数。例如

$$\begin{cases}\dot{y}=2y-yz\\ \dot{z}=-z+yz\end{cases}$$

是一个二阶系统。数值方法的阶数，则是前面所讨论的含义，即全局误差表达式中出现的步长的倍数。

检查数值方法的阶数的一种方法是，当$f(t,y)$为t的多项式且与y无关时，该数值方法的表现。如果对t^{p+1}保持精确，但对t^p不精确，那么其阶数不会超过p（阶数可能小于p是因为该方法对一些普通的函数可能不像对多项式那样精确）。欧拉法仅在$f(t,y)$为常数时才精确，但$f(t,y)=t$时不行，所以欧拉法的阶数不会超过1。

现代计算机使用IEEE双精度浮点数进行计算，计算过程中的舍入误差只在需要很高精度或积分区间非常长的情况下才变得重要。如果积分区间的长度为$L=t_f-t_0$，且一步计算中的舍入误差为ε，那么经过N步长为$h=L/N$的计算后，误差最坏的情况类似

$$N\varepsilon=\frac{L\varepsilon}{h}$$

对于一个全局离散化误差为Ch^p的方法来说，总的误差为

$$Ch^p+\frac{L\varepsilon}{h}$$

可以看出，为了让舍入误差与离散化误差可比，需要有

$$h\approx\left(\frac{L\varepsilon}{C}\right)^{\frac{1}{p+1}}$$

以这样的步长完成计算需要的步数大约为

$$N\approx L\left(\frac{C}{L\varepsilon}\right)^{\frac{1}{p+1}}$$

如果 $L=20$，$C=100$，$\varepsilon=2^{-52}$，对于不同阶数 p 的需要的步数为

p	N
1	4.5×10^7
3	5 647 721
5	37 285
10	864

这里p值为欧拉法、MATLAB指令ode23，ode45的阶数，以及变阶ode113s方法中阶数的典型取值。从上面的数值可以看出，对于低阶的方法，为了使最坏情况下的舍入误差和离散化误差可比，需要的步数是非常大的，在实际中几乎不可能达到。而且如果假设每步引起的误差随机变化，则需要更多的步数。相比之下，变阶数的多步法ode113s则完全能够满足考虑舍入误差时的精度要求。

§5.8 性 能

下面通过一个例子来考察这几种解法在实际中的表现。谐波振荡器的微分方程是

$$\ddot{x}(t) = -x(t)$$

取初始条件是$x(0) = 1$，$\dot{x}(0) = 0$，求解区间为$0 \leqslant t \leqslant 10\pi$，区间长度为周期解的5倍，所以全局误差可以简单地通过解的初值与终点值之差来计算，因为理论上有$x(0) = x(10\pi)$，所以全局误差近似和局部误差成正比。

下面的程序使用odeset来设置相对和绝对误差，并设置了细化等级以便算法在每一步都产生一个输出。

```
y0=[1 0];
for k=1:13
    tol= 10^(-k);
    opts=odeset('reltol',tol,'abstol',tol,'refine',1);
    tic
    [t,y]=ode23(@harmonic,[0 10*pi],y0,opts);
    time=toc;
    steps=length(t)-1;
    err=max(abs(y(end,:)-y0));
end
```

微分方程harmonic.m是

```
function ydot=harmonic(t,y)
ydot=[y(2); -y(1)];
```

分别用ode23，ode45，ode113运行上面的程序三次，图 5.11(a) 中显示了三个全局误差和要求的误差的变化关系。从图中可以看出，实际的误差和要求的误差的轨迹非常吻合。对于ode23，全局误差是要求误差的36倍，ode45是4倍，而ode113 则是在1和45倍之间变化。

图 5.11(b) 显示了需要的步数，结果和前面的模型非常吻合。令τ表示容差为10^{-k}，ode23方法需要的步数为$10\tau^{-1/3}$，符合一个三阶方法的特性。而ode45需

要的步数为$9\tau^{-1/5}$，也符合一个5阶方法的特性。对于ode113，图中反映出解的本身比较平缓，所以该方法大部分情况下都是其最大阶数13。

图 5.11(c) 显示了程序在一台800兆主频的奔腾III笔记本电脑上的运行时间，单位是秒。对这个问题当容差为10^{-6}或更大时，ode45是最快的方法，而对于较小的容差，不过只是一个实验，其中问题的解是一个非常平滑而稳定的函数。

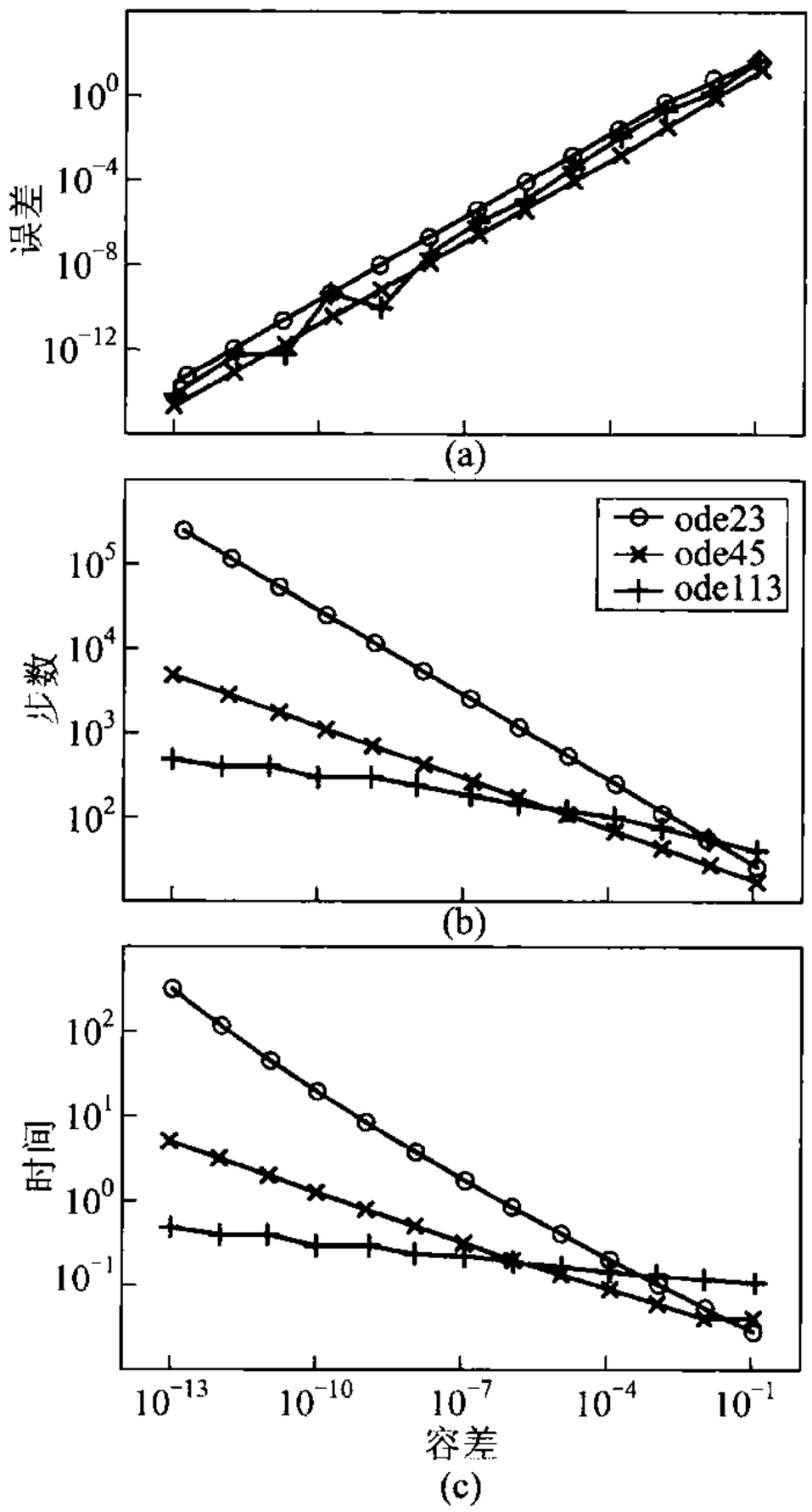


图 5.11 常微分方程求解指令的性能比较

§5.9 物理应用

§5.9.1 刚体绕瞬心的转动方程

在《大学物理》的创刊号（1982 年第一期）上，两位学生以“方言”为笔名发表了一篇论文，提出正确的刚体绕瞬时转动中心（简称瞬心）的转动方程应该是

$$\frac{\mathrm{d}}{\mathrm{d}t}(I_{\mathrm{s}}\dot{\boldsymbol{\theta}}) = \sum_i \boldsymbol{r}_i \times \boldsymbol{F}_i - \frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t} \times \boldsymbol{p} \tag{5.3}$$

式中$\boldsymbol{p}=\sum m_i\boldsymbol{v}_i$，$I_s$是刚体对瞬心的转动惯量，$\boldsymbol{R}$ 为瞬心的位矢，$\boldsymbol{r}_i\times\boldsymbol{F}_i$ 为所有外力对瞬时轴的力矩。

而此前在某些中外力学教材中介绍的公式是

$$\frac{\mathrm{d}}{\mathrm{d}t}(I_s\dot{\boldsymbol{\theta}})=\sum_i\boldsymbol{r}_i\times\boldsymbol{F}_i \tag{5.4}$$

与方言的公式相比，公式（5.4）缺少了第二项，因而是错误的。此后《大学物理》收到 32 篇稿件约 10 万字对此进行了热烈讨论，《大学物理》发表了其中的一部分，这些文章提出了多种论证方法并对公式的意义作了深入的分析，确认了新公式（5.3）是正确的，讨论有力地推动力学的教学研究工作。在庆祝《大学物理》创刊二十年的《大学物理》优秀论文评选中，方言的文章获得了特等奖。

虽然这个公式的正确性已经不容置疑，但是如何去展现公式所描述的运动图像仍然很有趣。在实验验证难以实现的情况下，计算机模拟自然就成了一个最佳的选择。计算机模拟的基本做法是，数值求解系统的运动微分方程，利用所得的结果计算两个公式中各个物理量，再代入公式加以对比。

公式（5.3）中的第二项如果为零，则两个公式没有区别，但是在实际的运动过程中，公式（5.3）中的 $\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}\times\boldsymbol{p}$ 项可能有 4 种情况：$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}$ 为零；$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}\times\boldsymbol{p}$为零；$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}\times\boldsymbol{p}$不为零；$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}\times\boldsymbol{p}$ 在一段时间为零，在另一段时间不为零。最后一种情况最为有趣，似乎很难遇到，下面就是这种例子。

长为$2l$质量为m的木杆斜靠在光滑的墙壁与地面之间，在重力作用下作下滑运动。如图 5.12 建立坐标系。木杆的运动要分成两个阶段来讨论。

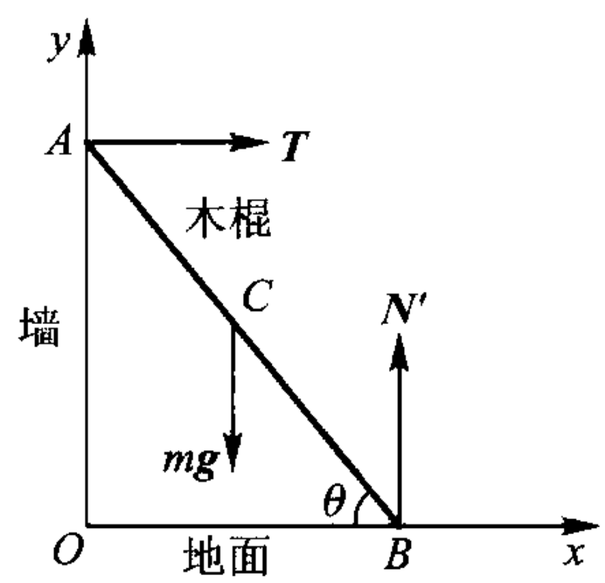


图 5.12 沿墙壁下滑的木杆

第一阶段，杆沿墙壁下滑但没有离开墙壁，杆受的力有3个：重力$m\boldsymbol{g}$，地面的支撑力$\boldsymbol{N}$，墙面推力 $\boldsymbol{T}$。运动微分方程如下：

$$\begin{cases} \ddot{x}_c = l\dot{\theta}^2\cos\theta - \dfrac{3}{4}g\sin\theta\cos\theta \\ \ddot{y}_c = -l\dot{\theta}^2\sin\theta - \dfrac{3}{4}g\cos^2\theta \\ \ddot{\theta} = -\dfrac{3g}{4l}\cos\theta \end{cases}$$

杆的瞬心是过木杆两端且与两端的速度方向（沿墙面与地面）垂直的直线的交点，显然$\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t} \neq \mathbf{0}$；但是通过计算得到$\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t} \times \boldsymbol{p} = \mathbf{0}$。

第二阶段，杆离开了墙壁继续运动，杆受的力变成2个，重力$m\boldsymbol{g}$和地面支撑力$\boldsymbol{N}$，杆的运动微分方程为

$$\begin{cases} \ddot{x}_c = 0 \\ \ddot{y}_c = \dfrac{l\dot{\theta}^2\sin\theta\cos^2\theta - g\cos^2\theta}{\dfrac{l}{3} + \cos^2\theta} - l\dot{\theta}^2\sin\theta \\ \ddot{\theta} = \dfrac{l\dot{\theta}^2\sin\theta\cos\theta - g\cos\theta}{\dfrac{l}{3} + l\cos^2\theta} \end{cases}$$

此时可算得$\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t} \neq \mathbf{0}$ 且有$\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t} \times \boldsymbol{p} \neq \mathbf{0}$。

数值计算的步骤如下，取$m = 5$，$l = 8$，$\theta_0 = 2\pi/3$。

第一阶段的初始条件取为

$$\begin{cases} x_c(0) = -l\cos\dfrac{2\pi}{3}, \quad \dot{x}_c(0) = 0 \\ y_c(0) = l\sin\dfrac{2\pi}{3}, \quad \dot{y}_c(0) = 0 \\ \theta(0) = \dfrac{2\pi}{3}, \quad \dot{\theta}(0) = 0 \end{cases}$$

求解第一阶段的运动时用指令events中止计算，中止的条件是$\boldsymbol{T} = \mathbf{0}$，即$\ddot{x} = 0$，得到

$$l\dot{\theta}^2\cos\theta - \frac{3}{4}g\sin\theta\cos\theta = 0$$

在计算结束时输出中止时的数据作为第二阶段的初始条件。

求解第二阶段的运动时也用指令events中止计算，中止的条件是$y_c = 0$。

求解过程中可以得到质心坐标x_c，y_c，质心速度分量$\dot{x}_c$，$\dot{y}_c$，转角及角速度θ，$\dot{\theta}$。由此计算出瞬心坐标

$$x_s = x_c - \frac{\dot{y}_c}{\dot{\theta}}, \quad y_s = y_c + \frac{\dot{x}_c}{\dot{\theta}}$$

根据平行轴定理，求得刚体绕瞬心的转动惯量为

$$I_s = I_c + mr_{sc}^2 = I_c + m[(x_s - x_c)^2 + (y_s - y_c)^2]$$

将这些数据画成图形，包括木杆下落的运动模拟，质心运动轨迹，瞬心运动轨迹，瞬心位矢随时间的变化，并将有关的数据分别代入方言的公式（5.3）与过去教材中的旧公式（5.4），将两个公式等号两边的计算结果画在一张图内进行对比。可以看到，方言的公式能够全面地描述系统的运动而旧公式只能在某些特例下反映系统的运动，所以是片面的。这个例子表明，计算机模拟能够弥补理论与实验的欠缺，直观地展现各种抽象的运动规律，深化对公式和概念的理解。

下面是两个阶段运动的程序与图像，两个阶段的时间前后相连的，读者可以将两个图合并在一个图像中观察。这只要在画完第一个图后，运行指令 hold on，再画第二个图即可。

木杆运动第一阶段的图像为图 5.13，程序如后面所列。运行以后输出的TE是停止运算的时间，它将作为第二阶段画图的时间坐标的起点，输出的 y_end 是结束运算时的函数值，它将作为第二阶段的初始条件。

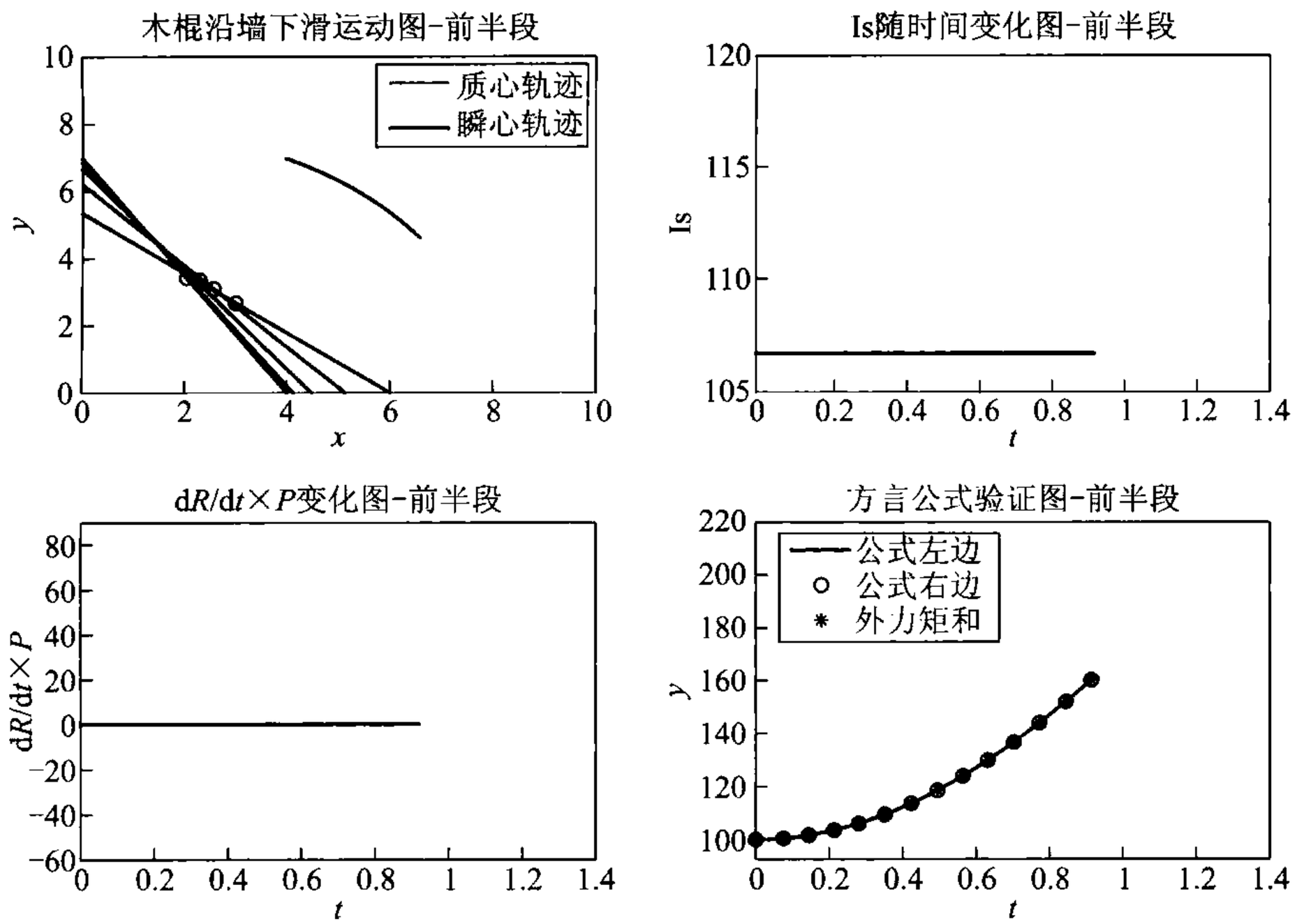


图 5.13 木杆运动的第一阶段

```
function mga                                    %木杆运动第一阶段
M=5;  w=0.00;  l=4;  g=9.81;
                                  %质量M,初速度w,杆长度l,重力加速度g
Ic=M*l^2/3;                                     %对质心的转动惯量
dt=0.001;     tspan=[0:dt: 2];
y0=[l*cos(pi/3),0,l*sin(pi/3),0,(2*pi/3),0];%初始条件
```

```
options=odeset('Events',@events);                    %开启事件判断功能
[T,Y,TE]=ode45( @fy4, tspan, y0, options);

N=length(T)-1;    t=T(2:N);                          %解方程中的时间变量
xc=Y(2:N, 1) ;   v_xc=Y(2:N, 2);                     %质心坐标与速度
yc=Y(2:N, 3);    v_yc=Y(2:N, 4);
fy=Y(2:N, 5);     v_fy=Y(2:N,  6);                   %转角与速度
xs=xc-v_yc ./ v_fy;    ys=yc+v_xc ./ v_fy;           %瞬心坐标

subplot(2,2,1)
plot(xc,yc,'r-', xs,ys,'k-'  );                      %质心轨迹和瞬心轨迹
axis([-0,10,0,10]);
xlabel('x');     ylabel('y');

title('木棍沿墙下滑运动图-前半段');
legend('质心轨迹','瞬心轨迹',1);

hold on
for k=2:200:N;                                       %木棍的位置
        x_a = xc(k) + l*cos(fy(k));  x_b=xc(k) - l*cos(fy(k));
        y_a = yc(k) +  l*sin(fy(k))  ;   y_b=0;
        plot([x_a,  x_b],    [y_a , 0],  xc(k),yc(k),'o');
end
dxs=gradient(xs,dt);                                 %求瞬心随时间的变化率
dys=gradient(ys,dt);
p=1 :  200 : N ;
quiver(xs(p),ys(p),dxs(p),dys(p));
                                                     %取部分数据作瞬心变化率的矢量图

subplot(2,2,2);
Is=Ic+M*((xs-xc).^2+(ys-yc).^2);                     %转动惯量
plot(t,Is);
axis([0,1.4,105,120]);
xlabel('t');    ylabel('Is');

title('Is随时间变化图-前半段');
```

```
subplot(2,2,3);
px=M.*v_xc; py=M.*v_yc;
dd=(dxs.*py - dys.*px);                                  %dR/dt×P 项
plot(t, dd);
axis([0,1.4,-60,90]);
xlabel('t');     ylabel('dR/dt    \times  P');

title('dR/dt   \times   P变化图-前半段');

subplot(2,2,4);
axis([0,1.4,90,220]);
hold on;

title('方言公式验证图-前半段');

xlabel('t');     ylabel('y');
V=1: 70 :N;       tt=t(V);                              %取部分数据画图
dfr=M*g.*(xs-xc);      dfr=dfr(V);                 %外力矩和(右边第一项)
left   = dfr-dd(V);                                  %方言公式的等号右边
dfl=gradient(Is.*v_fy,dt);
right = dfl(V);                                      %方言公式的等号左边
plot(tt,left,   tt,right,'O',     tt, dfr ,'*')

legend('公式左边','公式右边','外力矩和', 2);

TE                                                   %输出木棍脱离墙的时间
y_end=Y(end,:)                                       %输出终点数据

function f=fy4(t,y);
 l=4;     g=9.81;
f=[y(2);
   -3*g*cos(y(5))*sin(y(5))/4+l*cos(y(5))*y(6)^2;
   y(4);
   -3*g*cos(y(5))^2/4-l*sin(y(5))*y(6)^2;
   y(6);
```

```
    -3*g*cos(y(5))/(4*l)];

function [value,isterminal,direction]=events(T,Y)
l=4;    g=9.81;
value= l*Y(6).^2-3/4*g*sin(Y(5));
isterminal=1;    direction=0;
```

木杆运动第二阶段的图像为图 5.14，程序如下。两个程序的主要差别在于所解的方程不同。这个程序的初始条件是由第一个程序计算结束时输出的。

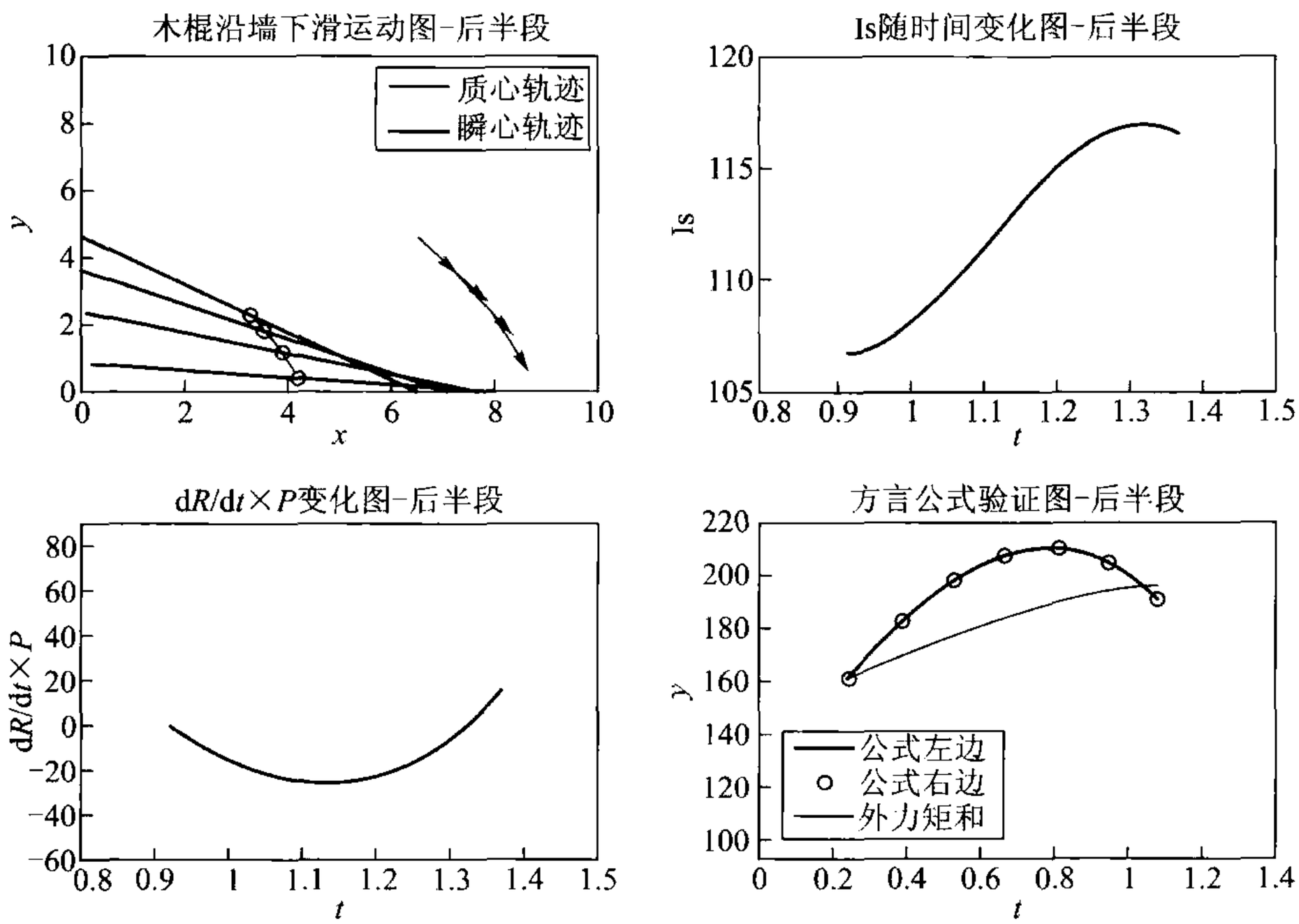


图 5.14　木杆运动的第二阶段

```
function mgb                                          %木杆运动的第二阶段
M=5; w=0.00; l=4; g=9.81;        %质量M,初速度w,杆长度l,重力加速度g
Ic=M*l^2/3;                                           %对质心的转动惯量
dt=0.001;      tspan=[0:dt:2];
y0=[3.2660,2.3799,2.3094,-3.3656,2.5261,1.0305];
options=odeset('Events',@events);                     %开启事件判断功能
[T,Y,TE]=ode45(@fy5,tspan,y0,options);                          %解方程

N=length(T)-1;    t=T(2:N);                           %解方程中的时间变量
xc=Y(2:N, 1) ;   v_xc=Y(2:N, 2);
```

```
yc=Y(2:N, 3);     v_yc=Y(2:N, 4);                    %质心坐标与速度
fy=Y(2:N, 5);      v_fy=Y(2:N,  6);                  %转角与速度
xs=xc-v_yc./v_fy;  ys=yc+v_xc./v_fy;                 %瞬心坐标

subplot(2,2,1)
plot(xc,yc,'r-', xs,ys,'k-'  );                      %质心轨迹和瞬心轨迹
axis([-0,10,0,10]);
xlabel('x');
ylabel('y');

title('木棍沿墙下滑运动图-后半段');
legend('质心轨迹', '瞬心轨迹',1);

hold on
for k=2:130:N;                                       %木棍的位置
        x_a = xc(k) + l*cos(fy(k)); x_b = xc(k) - l*cos(fy(k));
        y_a = yc(k) +  l*sin(fy(k));  y_b=0;
        plot([x_a,  x_b],    [y_a , 0],  xc(k),yc(k),'o');
end
dxs=gradient(xs,dt);
dys=gradient(ys,dt);                                 %求瞬心随时间的变化率
p=1 :  130 : N ;
quiver(xs(p),ys(p),dxs(p),dys(p));
                                                     %取部分数据作瞬心变化率的矢量图

subplot(2,2,2);
Is=Ic+M*((xs-xc).^2+(ys-yc).^2);                     %转动惯量
plot(t+0.9193,Is);
axis([0,1.4 ,105,120]);
xlabel('t');    ylabel('Is');

title('Is随时间变化图-后半段');

subplot(2,2,3);
px=M.*v_xc;      py=M.*v_yc;
```

```
dd=(dxs.*py - dys.*px);                                  %dR/dt×P 项
plot(t+0.9193, dd);
axis([0,1.4 ,-60,90]);
xlabel('t');    ylabel('dR/dt  \times  P');

title('dR/dt   \times   P变化图-后半段');

subplot(2,2,4);
axis([0,1.4 ,90,220]);
hold on;
xlabel('t');    ylabel('y');

title('方言公式验证图-后半段');

V=1: 70 :N;     tt=t(V)+0.9193;                          %取部分数据画图
dfr=M*g.*(xs-xc);
dfr=dfr(V);                                              %外力矩和(右边第一项)
left   = dfr-dd(V);                                      %方言公式的等号右边
dfl=gradient(Is.*v_fy,dt);
right = dfl(V);                                          %方言公式的等号左边
plot(tt,left,  tt,right,'O', tt, dfr)

legend('公式左边','公式右边','外力矩和',3);

function f=fy5(t,y);
g=9.81;     l=4;
f=[y(2);
      0;
   y(4);
   (l*sin(y(5))*cos(y(5))^2*y(6)^2-...
         cos(y(5))^2*g)/(1/3+cos(y(5))^2)-l*sin(y(5))*y(6)^2;
   y(6);
   (l*sin(y(5))*cos(y(5))*y(6)^2-cos(y(5))*g)/(1/3*l+...
   l*cos(y(5))^2)];
```

```
function [value,isterminal,direction]=events(T,Y)
value=Y(3);
isterminal=1;    direction=-1;
```

如果在第一个程序的每个子图的语句中分别加上hold on，并且在运行第一个程序后不要关闭图形窗口，接着运行第二个程序，就可以画出一个完整的木杆下落的图形及公式中各个量随时间变化的图形。

§5.9.2 弹簧摆运动

计算机模拟可以直观表现物理系统的运动状态，对于理解抽象的物理公式和研究系统的运动带来很多方便，所以在科研中有广泛的应用。例如发射神五、神六号登月舱的科研人员不仅计算了飞船的轨道，而且将它们的运动做成了动画。当这些动画在中央电视台播放时，全国人民都能了解飞船运行的状态。

这里我们借助于MATLAB，在研究物理系统的运动的同时，也用动画模拟它们的运动。由于这些运动模拟是采用真实的从求解运动方程得到的数据，而不是flash动画中想象的情景，所以这种模拟就更有价值。

如图 5.15 所示，设质量为m的摆锤挂在劲度系数为k，原长为l_0的轻弹簧上，弹簧的另一端悬挂于固定点O，系统静止自然下垂时弹簧长度为$l=l_0+\dfrac{mg}{k}$，系统可在过O点的竖直平面内自由摆动，试研究摆锤的运动。要求作出弹簧摆的模拟运动图像并画出摆锤的运动轨迹，并由此学习作模拟动画的技巧，如在模拟摆锤运动的同时画出摆锤的运动轨迹，用正弦曲线表示弹簧，并且通过极坐标变换来表示运动中的弹簧。

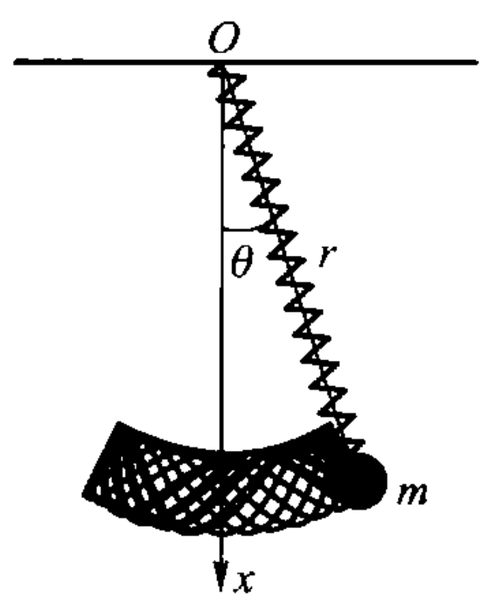


图 5.15 弹簧摆的运动

对题目分析如下，系统自由度为2。以O为极点，竖直向下的Ox轴为极轴，在图中建立极坐标系。r为质点m到O点距离，θ为Ox轴与弹簧间的夹角。则系统的拉格朗日函数为

$$L=T-V=\frac{1}{2}m\left[\left(\frac{\mathrm{d}r}{\mathrm{d}t}\right)^2+r^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2\right]-\left[-mgr\cos\theta+\frac{1}{2}k\left(r-l_0\right)^2\right]$$

由拉格朗日方程可求出系统的运动微分方程为

$$\begin{cases} \dfrac{\mathrm{d}^2 r}{\mathrm{d}t^2} = r\left(\dfrac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 + g\cos\theta - \dfrac{k}{m}\left(r - l + \dfrac{mg}{k}\right) \\ \dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\dfrac{2}{r}\dfrac{\mathrm{d}r}{\mathrm{d}t}\dfrac{\mathrm{d}\theta}{\mathrm{d}t} - \dfrac{g}{r}\sin\theta \end{cases}$$

上式未作小摆角近似，因此可用以研究弹簧摆的大摆角运动，但是难以求出解析解。由数值计算结果所画出弹簧摆的大摆角运动轨迹，可发现它的运动情况很复杂，不作数值计算是无法想象的。

令$y_1 = r, y_2 = \dfrac{\mathrm{d}r}{\mathrm{d}t}, y_3 = \theta, y_4 = \dfrac{\mathrm{d}\theta}{\mathrm{d}t}$，则微分方程成为

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = y_1 y_4^2 + g\cos y_3 - \dfrac{k}{m}\left(y_1 - l + \dfrac{mg}{k}\right) \\ \dfrac{\mathrm{d}y_3}{\mathrm{d}t} = y_4 \\ \dfrac{\mathrm{d}y_4}{\mathrm{d}t} = -\dfrac{2}{r}y_2 y_4 - \dfrac{g}{y_1}\sin y_3 \end{cases}$$

程序中解微分方程的过程比较简单，主要的技巧是在作模拟动画上。有两点值得注意，一是模拟弹簧的运动，另一点是在模拟弹簧运动的同时画出轨迹图。

主程序的文件名是thb，子程序文件为 thbfun，两个程序共用的变量 L，m，k，g 用指令global来传递。

```
function thb
global  L m k g
theta0=pi/10                                    %初始角度,可设不同的值
m = 1; k = 80;  g = 9.8;
L0 = 1;                                         %L0为弹簧原来长度
L = L0 + m*g/k;                                 %L为弹簧静止时长度
[t,u1] = ode45(@thbfun,[0:0.005:15],[L0 0 theta0 0]);
[y1,x1] = pol2cart(u1(:,3),u1(:,1));            %将极坐标换为直角坐标
y1 = -y1;

figure
ymax = max(abs(y1));
axis([-1.2  1.2  -1.2*ymax  0.2]);              %设置坐标范围
axis off
```

```
title('弹簧摆', 'fontsize', 14)
hold on;
R = 0.055 ;                                          %设置弹簧半径
yy = -L0:0.01:0;
xx = R*sin(yy./L0*30*pi);                  %用正弦曲线表示垂直位置的弹簧
[a,r] = cart2pol(xx,yy);                       %将弹簧直角坐标换成极坐标
a=a+theta0;
                                  %通过极角的转动将垂直位置的弹簧转到初始位置
[xx,yy]=pol2cart(a,r);          %将初始位置弹簧的极坐标换回直角坐标
line([-1 1],[0 0], 'color','r','linewidth',2)                  %画横杆
ball = line(xx(1),yy(1),'color','r','marker','.',...
   'markersize',70,'erasemode','xor');                         %画摆球
ball2 = line(xx(1),yy(1),'color',[0.5 0.51 0.6],
'linestyle','-',...
   'linewidth',1.3,'erasemode','none');          %画摆球的运动轨迹
spring = line(xx,yy,'color','g','linewidth',2,...
   'erasemode', 'xor');                                         %画弹簧
pause(0.5)

for i = 1 : length(t)                                  %模拟弹簧摆的运动
   yy = -u1(i,1) : 0.01 : 0;
   xx = R*sin(yy./u1(i,1)*30*pi);
   [a,r] = cart2pol(xx,yy);
   a = a + u1(i,3);
   [xx,yy] = pol2cart(a,r);
   set(ball,'XData',x1(i),'YData',y1(i));
   set(ball2,'XData',x1(i),'YData',y1(i));
   set(spring,'XData',xx,'YData',yy);
   drawnow;
end

function F = thbfun(t,u)
global  L m k g
F = [u(2);
    u(1)*u(4)^2 + g*cos(u(3)) - k/m*(u(1) - L + m*g/k);
```

```
u(4);
-2*u(2)*u(4)/u(1) - g*sin(u(3))/u(1)];
```

§5.9.3 圆锥陀螺运动

MATLAB的可视化功能十分强大，充分开发这种功能，对科研和教学都十分有益。本节研究拉格朗日-泊松情况下的圆锥陀螺运动，即圆锥陀螺在重力场中的运动。陀螺的运动十分复杂，微分方程组的形式也很复杂，虽然在教科书中都会介绍这个问题，理论上也是可以求出解析解，但是在教材中几乎都不介绍它求解的过程，因为实在太复杂。但是数值求解却并不困难。更有趣的是，如果做成模拟系统运动的动画，不仅可以直观地表现陀螺的自转、章动和进动，还可以定量地观察初始条件对陀螺运动的影响。

这里主要是学习如何用动画演示圆锥陀螺在重力场中的自转、章动和进动。改变程序中的初始条件，可以表现陀螺的不同运动形态，实现了对物理系统模拟的效果。

程序中有些技巧很值得注意，如使用坐标变换函数来画动画，用指令 cylinder 画圆锥以表现陀螺的形状。

首先建立描述陀螺的模型。如图 5.16 所示，设匀质圆锥陀螺的顶点为固定点O，高为h，圆锥底面半径为R，密度为ρ，则其质量为$m=\dfrac{\pi}{3}\rho hR^2$，质心在动力对称轴上，距固定点$O$的距离为$l=\dfrac{3}{4}h$。以陀螺的固定点$O$为原点建立静止坐标系$O\xi\eta\zeta$，以$O$点为原点，选取陀螺的对称轴为$Oz$轴，取节线为$Ox$轴，建立与刚体半固连的主轴坐标系$Oxyz$，该坐标系只随陀螺进动和章动，不随陀螺自转。

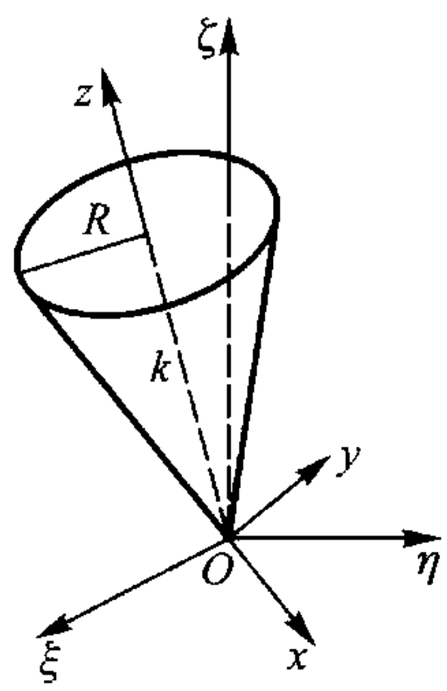


图 5.16 圆锥陀螺的运动模型

陀螺的拉格朗日函数为$L=T-V$，动能和势能分别为

$$T=\frac{1}{2}\left(I_x\omega_x^2+I_y\omega_y^2+I_z\omega_z^2\right)$$

$$V=mgl\cos\theta$$

角速度的投影为

$$\begin{cases} \omega_x = \dfrac{\mathrm{d}\theta}{\mathrm{d}t} \\ \omega_y = \dfrac{\mathrm{d}\varphi}{\mathrm{d}t}\sin\theta \\ \omega_z = \dfrac{\mathrm{d}\varphi}{\mathrm{d}t}\cos\theta + \dfrac{\mathrm{d}\psi}{\mathrm{d}t} \end{cases}$$

式中θ，φ，ψ分别为陀螺的章动角、进动角和自转角。由此得

$$L = \frac{1}{2}\left[I_x\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 + I_y\left(\frac{\mathrm{d}\varphi}{\mathrm{d}t}\sin\theta\right)^2 + I_z\left(\frac{\mathrm{d}\varphi}{\mathrm{d}t}\cos\theta + \frac{\mathrm{d}\psi}{\mathrm{d}t}\right)^2\right] - mgl\cos\theta$$

其中

$$I_x = I_y = \rho\int_0^h \mathrm{d}z\int_0^{2\pi}\mathrm{d}\theta\int_0^{\frac{zR}{h}}(z^2 + r^2\sin^2\theta)r\mathrm{d}r = \rho\frac{\pi}{5}h^3R^2 + \rho\frac{\pi}{20}hR^4$$

$$I_z = \rho\int_0^h \mathrm{d}z\int_0^{2\pi}\mathrm{d}\theta\int_0^{\frac{zR}{h}} r^3\mathrm{d}r = \rho\frac{\pi}{10}hR^4$$

运动微分方程为

$$\begin{cases} \dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = (1-a)(\dfrac{\mathrm{d}\varphi}{\mathrm{d}t})^2\sin\theta\cos\theta - a\dfrac{\mathrm{d}\varphi}{\mathrm{d}t}\dfrac{\mathrm{d}\psi}{\mathrm{d}t}\sin\theta + bg\sin\theta \\ \dfrac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} = (a-2)\dfrac{\mathrm{d}\theta}{\mathrm{d}t}\dfrac{\mathrm{d}\varphi}{\mathrm{d}t}\cot\theta + \dfrac{a}{\sin\theta}\dfrac{\mathrm{d}\theta}{\mathrm{d}t}\dfrac{\mathrm{d}\psi}{\mathrm{d}t} \\ \dfrac{\mathrm{d}^2\psi}{\mathrm{d}t^2} = \dfrac{\mathrm{d}\theta}{\mathrm{d}t}\dfrac{\mathrm{d}\varphi}{\mathrm{d}t}[\sin\theta - (a-2)\cot\theta\cos\theta] - a\dfrac{\mathrm{d}\theta}{\mathrm{d}t}\dfrac{\mathrm{d}\psi}{\mathrm{d}t}\cot\theta \end{cases}$$

其中

$$a = \frac{I_z}{I_x} = \frac{2}{4h^2/R^2+1};\quad b = \frac{ml}{I_x} = \frac{5\rho h}{4h^2+R^2}$$

令$y_1 = \theta$, $y_2 = \dfrac{\mathrm{d}\theta}{\mathrm{d}t}$, $y_3 = \varphi$, $y_4 = \dfrac{\mathrm{d}\varphi}{\mathrm{d}t}$, $y_5 = \psi$, $y_6 = \dfrac{\mathrm{d}\psi}{\mathrm{d}t}$，得

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = (1-a)(y_4)^2\sin y_1\cos y_1 - ay_4y_6\sin y_1 + bg\sin y_1 \\ \dfrac{\mathrm{d}y_3}{\mathrm{d}t} = y_4 \\ \dfrac{\mathrm{d}y_4}{\mathrm{d}t} = (a-2)y_2y_4\cot y_1 + \dfrac{a}{\sin y_1}y_2y_6 \\ \dfrac{\mathrm{d}y_5}{\mathrm{d}t} = y_6 \\ \dfrac{\mathrm{d}y_6}{\mathrm{d}t} = y_2y_4[\sin y_1 - (a-2)\cot y_1\cos y_1] - ay_2y_6\cot y_1 \end{cases}$$

这个微分方程的解法很简单，主要的问题是如何作出动画。从欧拉角的知识中可知，如果以圆锥的顶点为原点建立一个静止坐标系，再以圆锥的顶点为原点建立一个与刚体固连的动坐标系，则圆锥表面上一点的坐标x_0，y_0，z_0在动坐标系中是一个常数，而这点在静止坐标系中的坐标x，y，z则可以通过欧拉角表示的坐标变换来得到，从几何关系可以得出，这个坐标变换关系是

$$\begin{cases} x = x_0(\cos\psi\cos\varphi - \sin\psi\cos\theta\sin\varphi) \\ \qquad +y_0(-\sin\psi\cos\varphi - \cos\psi\cos\theta\sin\varphi) + z_0\sin\theta\sin\varphi \\ y = x_0(\cos\psi\cos\varphi + \sin\psi\cos\theta\cos\varphi) \\ \qquad +y_0(-\sin\psi\sin\varphi + \cos\psi\cos\theta\cos\varphi) + z_0(-\sin\theta\cos\varphi) \\ z = x_0\sin\psi\sin\theta + y_0\cos\psi\sin\theta + z_0\cos\theta \end{cases}$$

若将从微分方程中解得的欧拉角随时间变化的规律$\theta = \theta(t)$，$\varphi = \varphi(t)$，$\psi = \psi(t)$代入，就得出此点相对静止坐标系的运动规律。在主程序中，按照这个公式建立了一个子函数来计算运动中圆锥表面上各点在静止坐标系中的坐标。

下面给出参考程序。主程序的文件名是tlyd.m。其中包含两个子函数文件，一个是用来文件对陀螺的坐标进行变换，文件名为zbbh，另一个是用来建立陀螺的运动微分方程，文件名为tlydfun。在主函数文件与子函数文件tlydfun之间用指令global来传递共用的变量a，b，g。程序还给出了章动角、进动角和自转角的位移、速度和加速度随时间演变的图形。

```
function tlyd
R = 1; h = 2;  p = 1;    g = 9.8;                          %陀螺参数
a = 2/(4*(h/R)^2+1);   b = 5*p*h/(4*h^2+R^2);
thi=pi/6;  phi = 0;  psi = 0;                              %初始条件
[t,u] = ode45(@tlydfun,[0 :0.02:10],[thi phi psi 0 0 75]);
axis([-1  1 -1 1 0 1])
grid on
title('%拉格朗日-泊松情况下的圆锥陀螺运动');
hold on;
[x0,y0,z0]=cylinder(0:.05:.5,60);                  %陀螺侧面的数据网格
Q=linspace(0,2*pi,60);                             %以下4句是陀螺上表面数据
cx0=0.5*cos(Q);
cy0=0.5*sin(Q);
cz0=ones(1,60);
[x,y,z] = zbbh(x0,y0,z0,thi,phi,psi);              %求陀螺新位置的数据
[cx,cy,cz] = zbbh(cx0,cy0,cz0,thi,phi,psi);
                                                   %求陀螺盖新位置的数据
```

```
xm0=0;   ym0=0;    zm0=1.2;                                  %中轴线数据
[xm,ym,zm]=zbbh(xm0,ym0,zm0,thi,phi,psi);                    %中轴线
h3=plot3([0,xm],[0,ym],[0,zm],'r','linewidth',5);
h1=surf(x,y,z);                                          %画初始位置的陀螺
h2=fill3(cx,cy,cz,'m');                                      %给陀螺加盖
h4=plot3(cx(1),cy(1),cz(1),'b.','markersize',20);
pause(0.5);
 for i = 1 : length(u);
   [x,y,z] = zbbh(x0,y0,z0,u(i,1),u(i,2),u(i,3));
                                                   %求陀螺新位置的数据
   [cx,cy,cz] = zbbh(cx0,cy0,cz0,u(i,1),u(i,2),u(i,3));
                                                   %求陀螺新位置的数据
   cn=[xm,ym,zm];
    [xm,ym,zm]=zbbh(xm0,ym0,zm0,u(i,1),u(i,2),u(i,3));
                                                            %中轴线
    set(h1,'xdata',x,'ydata',y,'zdata',z);          %画新位置的陀螺
    set(h2,'xdata',cx,'ydata',cy,'zdata',cz);
    set(h3,'xdata',[0,xm],'ydata',[0,ym],'zdata',[0,zm]);
    set(h4,'xdata',cx(1),'ydata',cy(1),'zdata',cz(1));
    plot3([cn(1),xm],[cn(2),ym],[cn(3),zm]);        %留下中轴轨迹
    drawnow;
 end
%计算加速度
u(:,7) = u(:,5).^2.*sin(u(:,1)).*cos(u(:,1)).*(1-a) - ...
   a.*u(:,5).*u(:,6).*sin(u(:,1)) + b.*g.*sin(u(:,1));
u(:,8) = u(:,4).*u(:,5).*cot(u(:,1)).*(a-2) + ...
   a.*u(:,4).*u(:,6)./sin(u(:,1));
u(:,9) = u(:,4).*u(:,5).*(sin(u(:,1))-
   cot(u(:,1)). ... *cos(u(:,1)).*(a-2)) - ...
   a.*u(:,4).*u(:,6).*cot(u(:,1));
text_s = {'%章动角-t','进动角-t','自转角-t', ...
      '%章动角速度-t','进动角速度-t','自传角速度-t', ...
      '%章动角加速度-t','进动角加速度-t','自转角加速度-t'};
figure('Units','normalized',    'Position', ...
[0.05 0.10 0.90 0.80], ...
```

```
    'Tag','curve',    'NumberTitle','off', 'Name', ...
    '%欧勒角特性曲线');
for i = 1 : 3
  for j = 1 : 3
     axes('Position',[0.05+0.32*(j-1)
     0.04+0.32*(3-i) 0.27 0.24], ...
        'XGrid','on','YGrid','on');
     uicontrol('Units','normalized', ...
        'Position',[0.11+0.32*(j-1)...
        0.28+0.32*(3-i) 0.15 0.04], ...
        'Style','text',     'BackgroundColor',...
        [0.8 0.8 0.8], ...
        'String',text_s{(i-1)*3+j},   'FontSize',12);
     plot(t,u(:,(i-1)*3+j));
  end
 end
%-----------坐标变换函数-----------------------------
function [x,y,z] = zbbh(x0,y0,z0,thi,phi,psi)
[ph,r] = cart2pol(x0,y0);
ph = ph +psi -phi ;
[x,y] = pol2cart(ph,r);
[th,r] = cart2pol(y,z0);
th = th + thi;
[y,z] = pol2cart(th,r);
[ph,r] = cart2pol(x,y);
ph = ph  + phi;
[x,y] = pol2cart(ph,r);

function y = tlydfun(t,u)
g = 9.8;    R = 1; h = 2;  p = 1;
y = [u(4);
   u(5);
   u(6);
   u(5)^2*sin(u(1))*cos(u(1))*(1-(R/h)^2/2) - ...
      (R/h)^2/2*u(5)*u(6)*sin(u(1)) + 5*p*g*sin(u(1))/4/h;
   u(4)*u(5)*cot(u(1))*((R/h)^2/2-2) + (R/h)^2/2*u(4)...
```

```
    *u(6)/sin(u(1));
u(4)*u(5)*(sin(u(1))-cot(u(1))*cos(u(1))*((R/h)^2/2-2))...
    - (R/h)^2/2*u(4)*u(6)*cot(u(1))];
```

运行程序所得图形如图 5.17，图 5.18 所示。

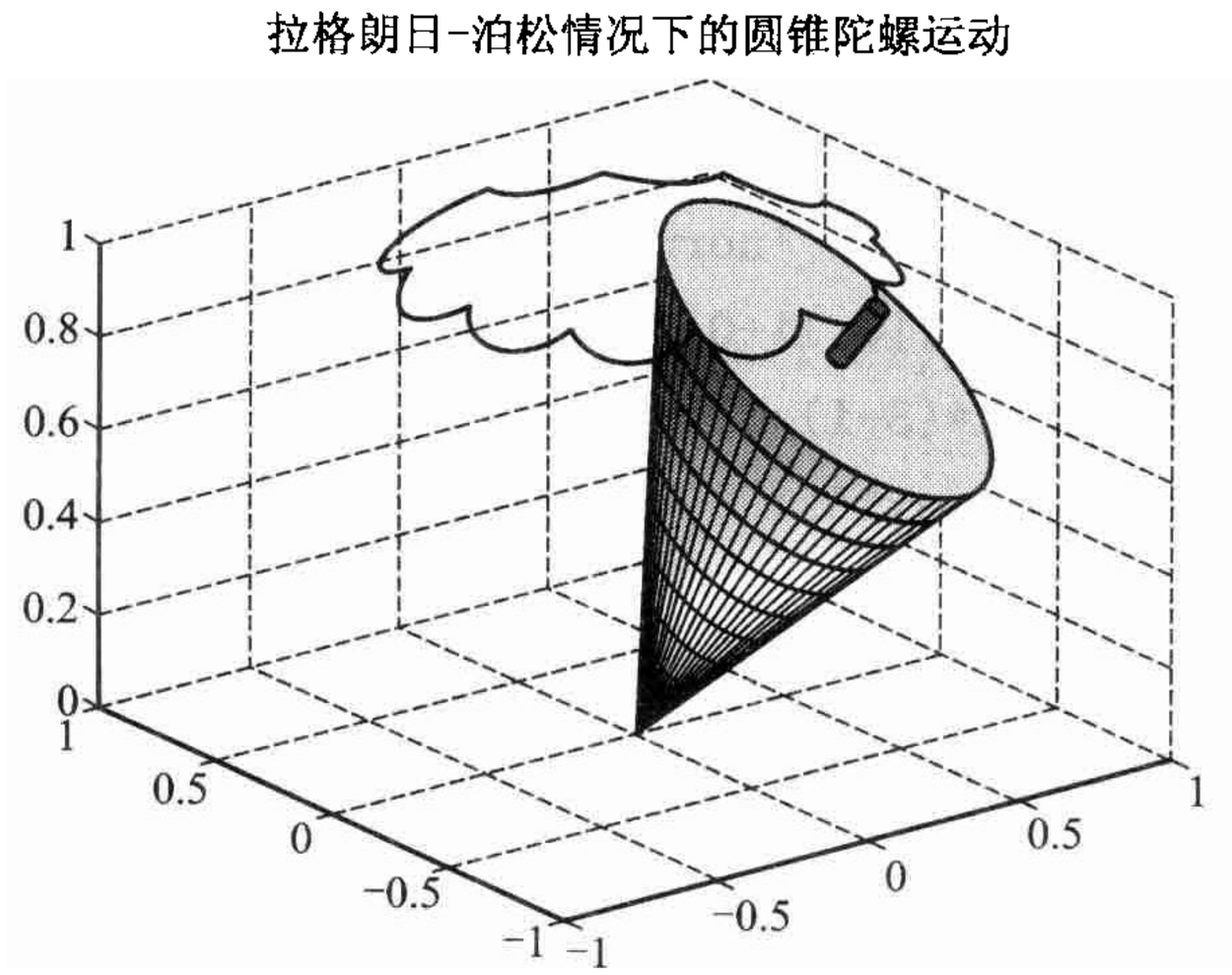


图 5.17 陀螺运动的动画模拟

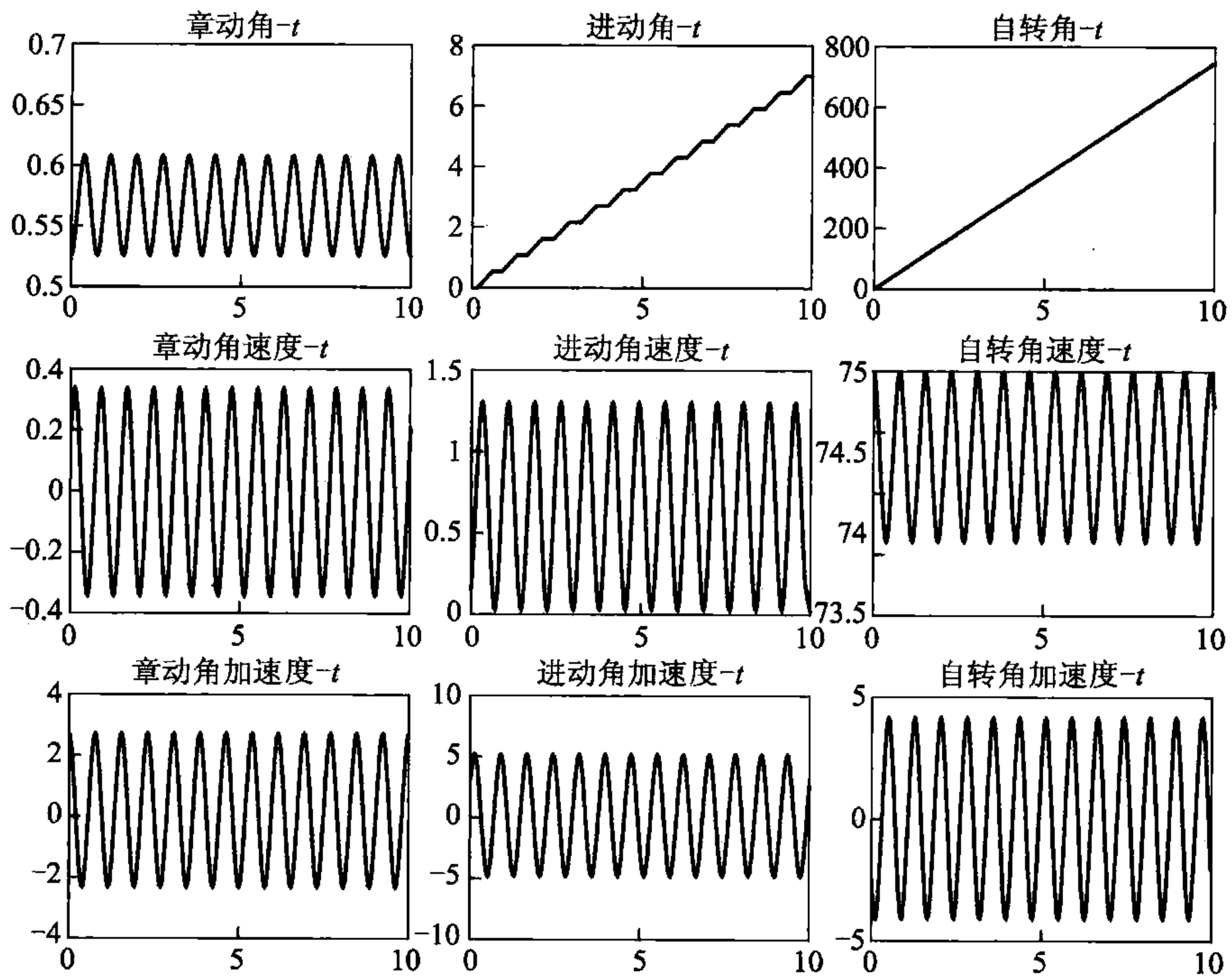


图 5.18 陀螺运动中章动角、进动角和自转角的变化

常微分方程在理论物理的教学中有大量的应用，在《理论力学计算机模拟》和《数学物理方程的MATLAB解法与可视化》两本书中有丰富的实例，有兴趣的读者可以参考。

§5.10 边值问题和本征值

本节讨论二阶常微分方程的边值问题和本征值问题。二阶常微分方程

$$y'' = f(x, y, y'), \quad a \leqslant x \leqslant b \tag{5.5}$$

的边值问题有以下三类边值条件：

第一类边值条件：$y(a) = \alpha$，$y(b) = \beta$。

第二类边值条件：$y'(a) = \alpha$，$y'(b) = \beta$。

第三类边值条件：$y'(a) - \alpha_0 y(a) = \alpha_1$，$y'(b) + \beta_0 y(b) = \beta_1$。

微分方程(5.5)附加上第一、第二、第三类边值条件，分别称为第一、第二、第三类边值问题。

§5.10.1 边值问题的直接积分

边值问题通常是在两个边界上各有一个边界条件，如果在同一个边界上有两个边界条件，就可以把问题作为初值问题来解。把边值问题化为初值问题的基本做法是，在任一边界上补充一个猜测的边界条件，按照初值问题来解方程，所得的解通常不会满足另一端的边界条件，需要改变猜测的边界条件或者对所得的解加以某些修正，重新解方程，直到找出解为止。这里的关键是，如何去改变猜测的边界条件，所用的方法有多种，下面将介绍几种。

作为边值问题的一个具体实例，考虑在电荷分布为

$$\rho(r) = \frac{1}{8\pi} \mathrm{e}^{-r} \tag{5.6}$$

的情形下所产生的静电势Φ。静电势Φ满足泊松方程

$$\nabla^2 \Phi = -4\pi\rho \tag{5.7}$$

对于球对称的ρ 和Φ，这个方程简化为

$$\frac{1}{r^2} \frac{\mathrm{d}}{\mathrm{d}r} \left[r^2 \frac{\mathrm{d}\Phi}{\mathrm{d}r} \right] = -4\pi\rho \tag{5.8}$$

作代换

$$\Phi(r) = r^{-1} \phi(r) \tag{5.9}$$

上述方程化为

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}r^2} = -4\pi r\rho \tag{5.10}$$

这种电荷分布的总电荷是

$$Q = \int \rho(r)\mathrm{d}^3r = \int_0^\infty \rho(r)4\pi r^2\mathrm{d}r = 1 \tag{5.11}$$

这个问题的精确解是

$$\phi(r) = 1 - \frac{1}{2}(r+2)\mathrm{e}^{-r} \tag{5.12}$$

由此可得静电势$\Phi = r^{-1}\phi$。可以看出这个解的行为特性是：当$r = 0$时它等于零，当r足够大时，$\phi \to 1$，它相当于$\Phi \to r^{-1}$，即单位电荷产生的库仑势。解析解的图形见图 5.19(a)。

现在把这个问题当作一个初值问题来解。以原点作为起点，向外对泊松方程积分。要开始计算，除了要知道$\phi(0) = 0$还必须知道ϕ_1的值，(或者等价地，知道$\mathrm{d}\phi/\mathrm{d}r = \Phi$ 在$r = 0$ 的值)。而ϕ_1正是要求的函数，所以是未知的。下面将会讨论在一般的情形下处理的办法，这里先用解析解(5.12) 求$\phi_1 = \phi(r = h)$的值。这就是下面的程序中的第一段所计算的ϕ_0。程序中积分区间为$0 \sim R$，故要输入R的值，R可取$0 \sim 30$。

```
function bzwt
r=0:0.1:20;
s= - 0.5*r.*exp(-r);                                    %电势
exact = 1 - 0.5* (r + 2).* exp(-r);
dexact = exact(2)/0.1;
[x,y0]=ode45(@bzfun,r,[0,dexact]);
figure(1)
plot(x,y0(:,1),  r,exact,':')
[x,y1]=ode45(@bzfun,r,[0,dexact+0.01]);
m=(y1(end,1)-y1(end-10,1))/(x(end)-x(end-10));
phi=y1(:,1)-m*x;
figure(2)
plot(x,y1(:,1) ,x,phi,'-.')
function ydot=bzfun(x,y)
ydot = [ y(2);   - 0.5*x.*exp(-x) ];
```

从图 5.19 上看，解析解的曲线和数值解的曲线符合得相当好。但是，如果不知道ϕ_1的值的话，问题就会复杂起来，为了说明这一点，假定程序中的$\phi(2)$有

一点误差，所以将语句$\phi(2) = \text{exact}(2)$ 改为$\phi(2) = \text{exact}(2) + 0.001$，这时会发现，随着$r$的增加，数值解$\phi_1$的结果与解析解的差别变得很大。

产生这个结果的原因是：方程(5.10)对应的齐次方程

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}r^2} = 0$$

的解可以加到方程(5.10)的任何一个特解上以得出另一个解。齐次方程的两个线性独立的解为

$$\phi \sim r; \quad \Phi \sim \text{常数}$$

和

$$\phi \sim \text{常数}; \ \Phi \sim r^{-1}$$

方程(5.10)在渐近区域内（此时ρ为零，方程是齐次的）的通解可以写成这两个函数的线性组合，但是后一个处于劣势的解才是物理上有意义的解。因为r 大时的位势由$\Phi \to 1/r$ 给出,规定原点Φ值时不够精确或积分过程中的任何数值舍入误差都能引入$\phi \approx r$解的一个小成分，它最终将在r大时占支配地位。

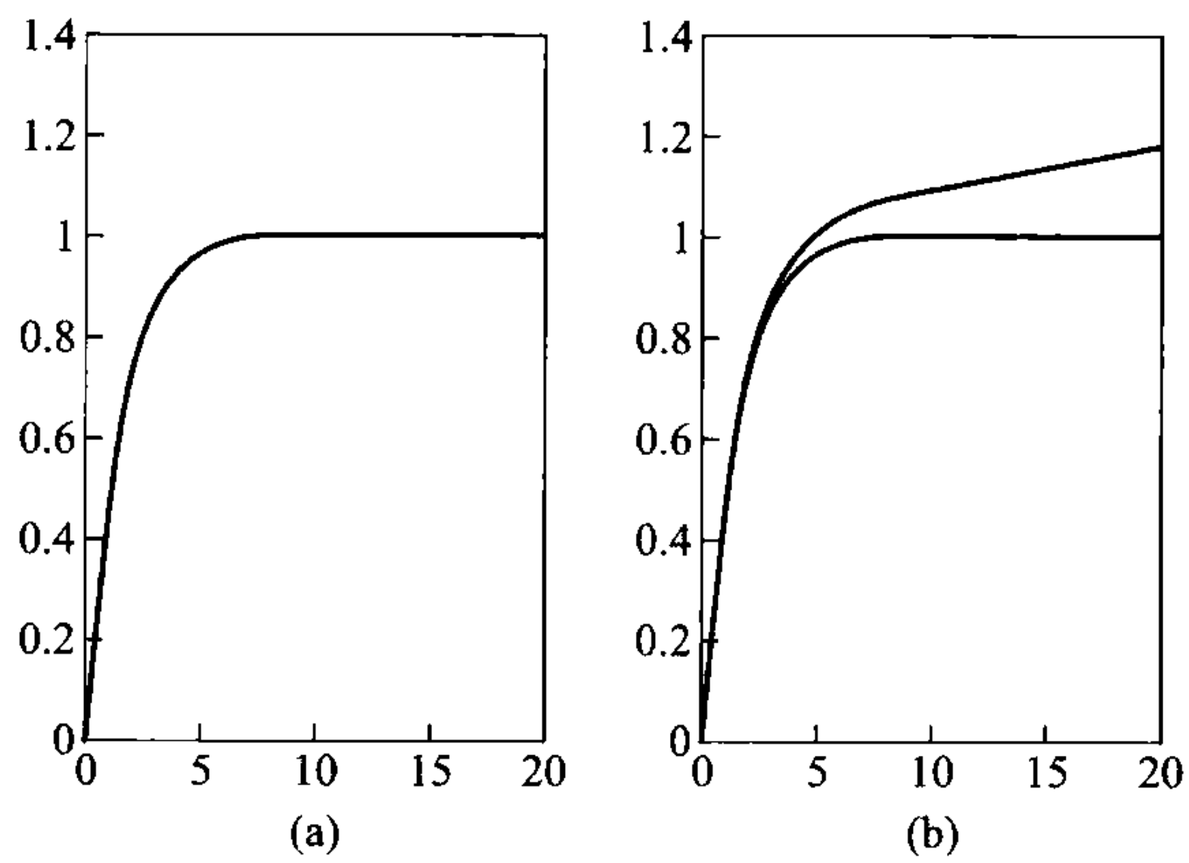


图 5.19

(a) 解析解的曲线与用精确解作启动值的数值解的曲线，两条曲线重合很好。

(b) 启动值具有一定误差的数值解曲线和经过修正以后的数值解曲线。可以看出，经过修正以后，数值解又变得与解析解一样。

解决这个问题的方法就是从数值结果中减去齐次方程的“坏的”、非物理解的一个倍数以保证渐近区域内的物理行为。很容易看出，“坏的”结果对大的r表现为一种线性关系，所以可以将最后10个点拟合于直线

$$\phi = mr + b$$

然后从数值解中减去mr以保证解在大的r时有正确的行为。具体的做法是将下列两行程序：

```
m=(phi(R/0.1)-phi(R/0.1-10))/(10*0.1);
phi=phi-m*r;
```

加入到原程序的end 语句之后。这时可以看到新的结果ϕ_2与精确结果的两条曲线实际上已完全重合。图 5.19(b) 显示了这种结果。

在这个例子中，由于能够在解中直接找出产生误差的“成分”，所以只要在尝试解中简单地减去引起误差的成分就得到了正确的解而不必经过多次的尝试。

§5.10.2 打靶法求弦振动方程本征值

本征值问题就是方程中含有参数的边值问题，只有当参数取某些特定值，这个方程才有解。打靶法解本征值问题的做法是：先尝试一个本征值，然后将微分方程作为初值问题求解。如果所得的解不满足边界条件，就改变尝试本征值，再解方程，重复这个过程，直到能找到一个本征值，在这个本征值下生成的解和边界条件的误差小于预定的容许误差。

下面用打靶法解弦振动方程。考虑一根两端固定的密度均匀的绷紧的弦，描述它的微小振动的方程经过分离变量以后的本征值方程为

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}x^2} = -k^2\varphi; \quad \varphi(0) = \varphi(1) = 0$$

这里$0 < x < 1$是沿着弦的标度化后的坐标，φ 是弦的横向位移，k 是本征值。由数学物理方法的知识可知，它的解析解是

$$k_n = n\pi, \quad n = 1, 2, \cdots$$

$$\varphi_n \sim \sin n\pi x$$

具体的解法是：由小到大选取一个试验的 k 值，从 $x = 0$ 出发向前积分，初始条件为

$$\varphi(0) = 0, \quad \varphi'(0) = t$$

数t是任意选取的，因为要解的问题是齐次的，并且对解如何归一化并没有规定。在积到$x = 1$时，一般将得到一个不为零的φ值，因为试验本征值并不是一个真正的本征值，于是需要重新调整k并且再度积分，重复这个过程，直到在规定的容许误差限内找到$\varphi(1) = 0$为止。这样就找到了一个本征值和对应的本征函数。

求使$\varphi(1)$为零的k值问题就是以前讨论过的求根的问题。可以用对分法来做。为了提高效率，可以在用对分法确定了一个本征值的近似位置之后，再用弦割法来确定更精确的本征值。

实际的程序如下，程序中m为所求本征值的个数。

```
function tt=ch5x2(m)
global k
```

```
k=1;tol=1e-8;
for n=1:m
    dk=k/15;   k=k+dk;
    [x,phi]=ode45(@ch5x2fun,[0,1],[0,1e-3]);
    oldphi=phi(end,1);
    dphi=oldphi;
    while abs(dphi)>tol
          k=k+dk;
          [x,phi]=ode45(@ch5x2fun,[0,1],[0,1e-3]);
          dphi=phi(end,1);
          if dphi*oldphi<0
              k=k-dk; dk=dk/2;
          end
    end
    plot(x,phi(:,1))
    hold on
    kk(n)=k;
end
kk
function yy=ch5x2fun(x,phi)
global k
yy=[phi(2);-k^2*phi(1)];
```

要求出 3 个本征值可用$m=3$调用这个程序，得到的本征值是 3.1416，6.2832，9.2153。相应的图形如图 5.20 所示。

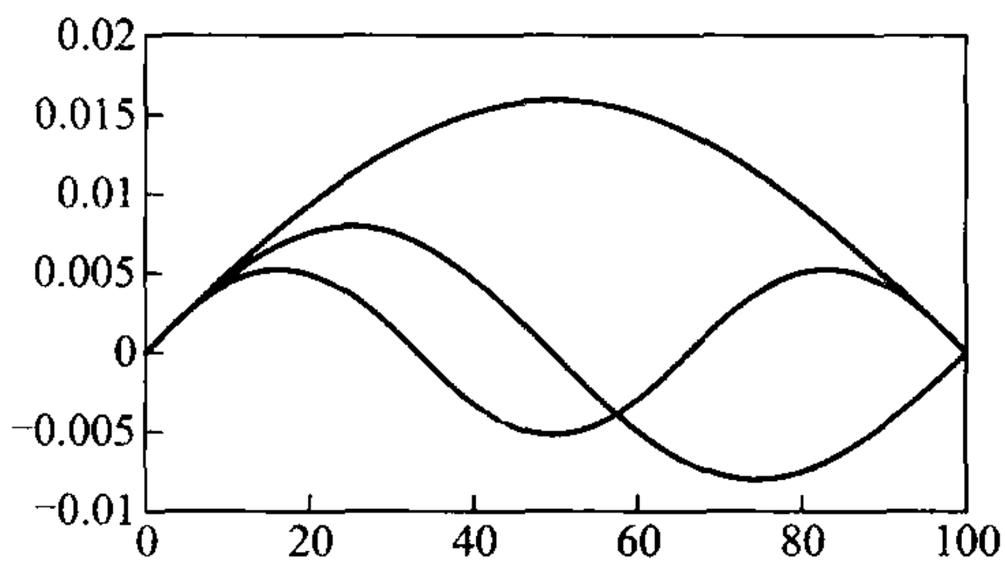


图 5.20 打靶法求得的本征函数

§5.10.3 一维薛定谔方程的定态解

解本征值问题的更有意义的例子是，用打靶法求在一维势阱$V(x)$中运动的质

量为m的粒子的量子力学定态。$V(x)$的形状大致如图 5.21 所示：在$x = x_{\min}$和$x = x_{\max}$两点位势变为无穷大 (或者说，在这两点有“刚性壁”)，在这两点之间则是一个势阱。粒子的定态薛定谔方程和边界条件是

$$\frac{\mathrm{d}^2\psi}{\mathrm{d}x^2} + k^2(x)\,\psi(x) = 0$$

$$\psi(x_{\min}) = \psi(x_{\max}) = 0$$

它是本征方程的形式，其中

$$k^2(x) = \frac{2m}{\hbar^2}\left[E - V(x)\right]$$

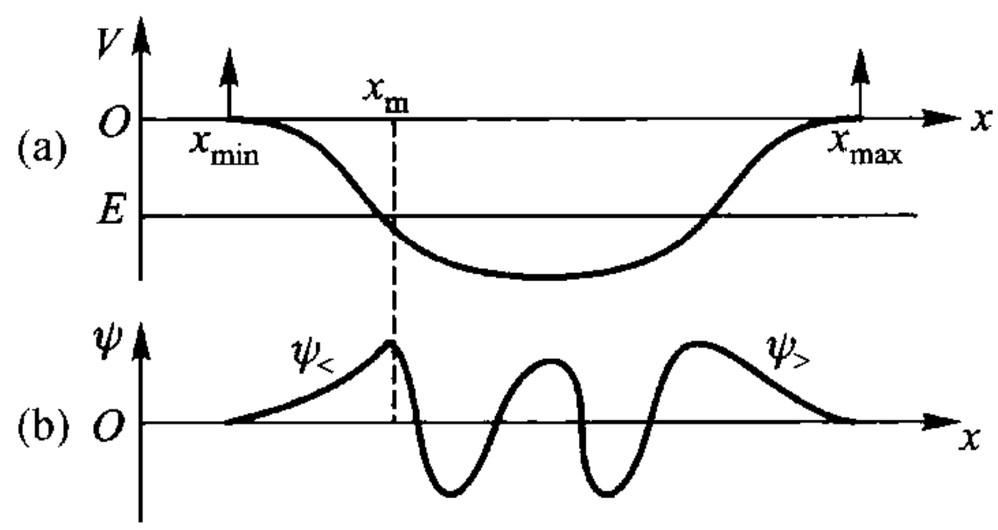


图 5.21

(a) 一维薛定谔方程的位势示意图。(b) 薛定谔方程在一个任意的能量$E < 0$上的解$\psi_<$和$\psi_>$用左转折点为匹配点。当E是一个本征值时，导数在匹配点上是连续的。

设要求的是粒子在势阱内的束缚态，那么能量本征值$E < 0$，预期本征函数在经典物理学容许的区域($E > V(x)$)内，波函数是振荡的，在经典物理学禁戒的区域($E < V(x)$)内，波函数呈指数行为。

求解时，取一个负的试验能量值。从$x_{\min}$出发朝更大的x积分。得到解$\psi_<$，它在经典禁戒区域内按指数方式增长，然后越过左转折点在经典容许的区域内振荡[见图 5.21 (b)]。如果越过转折点后继续积下去，那么积分将变成数值上不稳定的，因为即使在一个精确的本征值上(这时$\psi_<(x_{\min}) = 0$)，也可能混入不想要的指数增长的解的成分。作为一条普遍规则，进入一个经典禁戒区的积分限可能是不准确的。因此比较明智的做法是，在每一个能量上通过由$x_{\max}$ 向较小的x积分产生另一个解$\psi_>$，为了判断这个能量是不是一个本征值，可以在一个接合点x_{m}上比较$\psi_<$和$\psi_>$，x_{m}要这样选择，使得没有一个积分是不准确的 (x_{m}的一个方便的选择是左转折点)。由于$\psi_<$和$\psi_>$二者都满足一个齐次薛定谔方程，它们的归一化总可以这样选择，便得两个函数在x_{m}上相等。这时如果它们的微商在x_{m}也相等，就宣告了这个能量是本征值；也就是说，两个解要光滑地接合，这是这类问题的解析解所要求的。于是

$$\left.\frac{\mathrm{d}\psi_<}{\mathrm{d}x}\right|_{x_{\mathrm{m}}} - \left.\frac{\mathrm{d}\psi_>}{\mathrm{d}x}\right|_{x_{\mathrm{m}}} = 0$$

如果我们用这些微商在x_m和$x_m - h$两点的最简单的有限差分近似来代替它们,那么与上式等价的一个条件是

$$f = \frac{1}{\psi}\left[\psi_<(x_m - h) - \psi_>(x_m - h)\right] = 0$$

因为归一化的选择已经保证了$\psi_<(x_m) = \psi_>(x_m)$。上式中的量$\psi$是方括号中的差值的一个方便的标尺,可以这样来定它使得f的量级为1。它可以是x_m点上的ψ值,或者是$\psi_<$或$\psi_>$的最大值,注意如果不存在转折点 (例如,如果对于一个像图 5.21 所示那样的位势而$E > 0$),那么x_m 可以选在任何地方;而如果有多于两个转折点,那么就必须把三个或更多的齐次解 (每一个在不同区域中是准确的) 拼在一起。

下面程序求解了一个一维位势的定态。假定位势的形式为$V(x) = V_0\nu(x)$。其中无量纲函数$\nu(x)$具有一个极小值-1 (这只要对能量进行适当的线性标度总可以保证)。若坐标由一个物理长度a标度,那么粒子的定态薛定谔方程可以写成

$$\left[-\frac{1}{\gamma^2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \nu(x) - \varepsilon\right]\psi(x) = 0$$

其中$\gamma = \left[\frac{2ma^2V_0}{\hbar^2}\right]^{1/2}$是系统的经典力学本性的一个无量纲的测度,$\varepsilon = E/V_0$是无量纲能量。因此所有的本征值都满足$\varepsilon > -1$,这个程序的位势的函数形式是解析类型的抛物线位阱 (如取$\nu(x) = x^2/2 - 1$);用一个初始的试验能量、能量步长和γ值去对若干个态进行搜寻。对于每个试验本征值,子程序向前和向后积分薛定谔方程;两个解在最左边的转折点上进行接合,在这一点上,波函数的行为由振荡变为指数变化 (如果没有这样的转折点,则在x_{max}附近进行接合)。求得结果以后,试验波函数以图形方式显示,找到一个本征值之后,可输入下一个试验本征值,有时还要适当调整能量步长。

下面是用MATLAB编写的程序。在程序中,$\gamma = 50$,并使波函数的一阶导数在转折点连续,波函数的连续性则通过在转折点的调整来实现。用指令end求出在转折点的x的坐标。作图时安排成在每行画3 幅图,为了在同一个屏幕上把所有的图形都画出来,用指令fix确定将图形排成几行。

用MATLAB指令编写的程序如下:

```
function wlch5
global e1
n=9; eold=-1; olddpsi=0.5; tol=1e-6;
for k=1:n
   de=2*tol;e1=eold+abs(eold)/70;
   while abs(de)>tol
```

```
        xturn=-sqrt(2*(e1+1));kk=(-1)^(k+1)*0.0001;
        [x1,u1]=ode45(@wlch5fun,[-1 xturn],[0 kk]);
        [x2,u2]=ode45(@wlch5fun,[1 xturn],[0 -0.0001]);
        dpsi=u1(length(x1),2)-u2(length(x2),2);
        de=-dpsi*de/(dpsi-olddpsi);
        olddpsi=dpsi; eold=e1; e1=e1+de;
    end
  e(k)=eold;
  subplot(3,3,k);plot(x1,u1(:,1),x2,u2(:,1))
  end
  e
  diff(e)

  function yy=wlch5fun(x,psi)                                    %ga=50
  global e1
     yy=[psi(2);50^2*(-e1-1+x^2/2)*psi(1)];
```

程序求得的前四个能级为 -0.9859, -0.9576, -0.9293, -0.9010，这些能级之间的间距相等，这正是线性谐振子的基本性质。相应的波函数如图 5.22 所示：

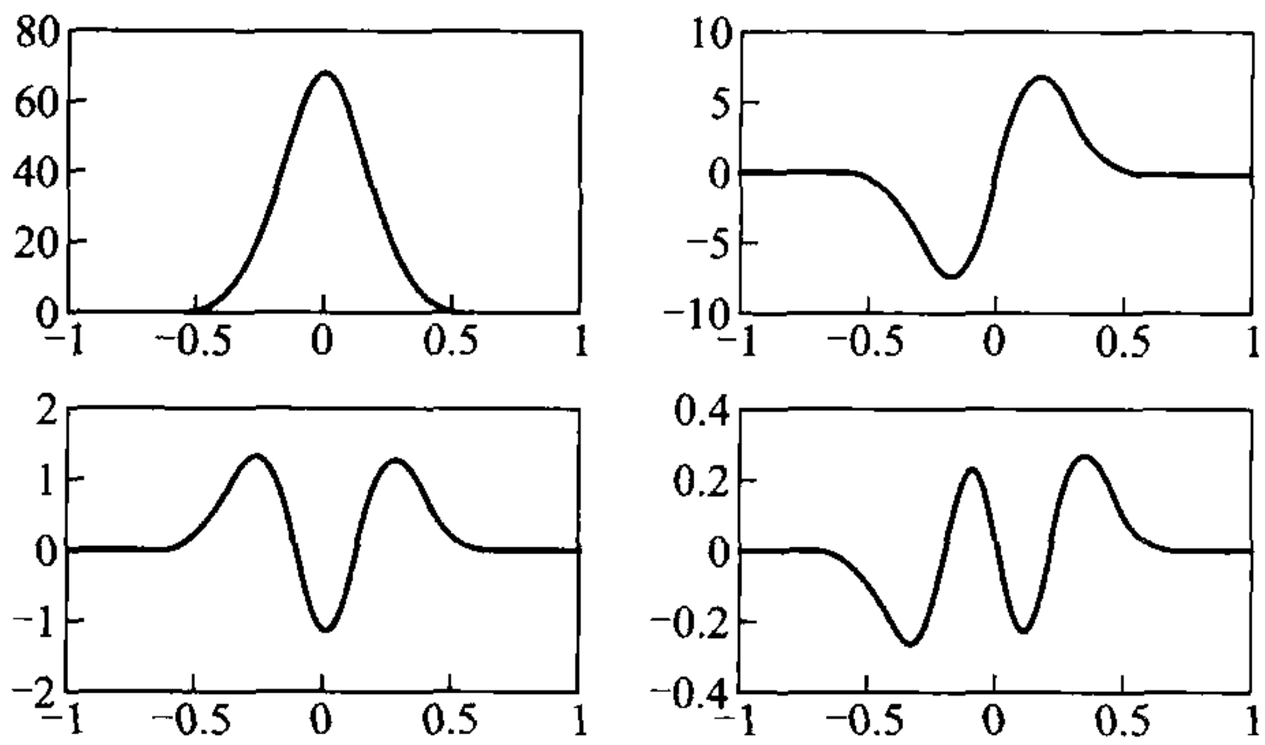


图 5.22 一维谐振子的前四个波函数

§5.11 用指令 bvp4c 解边值问题与本征值问题

在MATLAB中也有求解一维的边值问题或本征值问题的指令，这个指令就是bvp4c，这个指令的功能比前面所介绍的内容丰富得多。这里介绍一下bvp4c的用法。使用bvp4c会用到三个函数，即用odefun定义要求解的方程，用bcfun设置边界条件，用solinit给定一个初始的猜测解。

§5.11.1 用指令 bvp4c 解一维本征值问题

1. 指令 bvp4c 的用法

常微分方程的两点边值问题可能没解，可能有多个解，也可能有无限个解。在使用指令求解时，必须给出一个猜测解。猜测解的好坏会严重影响求解过程。

使用 bvp4c 的语句格式有三种：

sol = bvp4c(odefun,bcfun,solinit)

sol = bvp4c(odefun,bcfun,solinit,options)

sol = bvp4c(odefun,bcfun,solinit,options,p1,p2...)

语句中各个量的含义为：

odefun	由被求解的微分方程表示的函数$f(x,y)$。
bcfun	表示边界条件的函数$bc(y(a),y(b))$。
solinit	用以设置初始的猜测解的结构数组。
options	可选的积分参数。可以用bvpset来建立的结构数组。
p1,p2...	可选参数。是已知的参数，由算法指令将它们传递给odefun,bcfun，以及所有由用户在options中指定的函数。

下面会对 odefun，bcfun，solinit 作更详细的说明。

2. bvp4c 三种用法的比较

三种语句中的输入量不同，下面作一说明。

(1) **sol = bvp4c(odefun,bcfun,solinit)**

该语句在区间 $[a,b]$ 求解形式为

$$y' = f(x,y)$$

的常微分方程组，要求解满足两端点边界条件

$$bc(y(a),y(b)) = 0$$

bvp4c 也能解带有未知参数的形式为

$$y' = f(x,y,p)$$

$$bc(y(a),y(b),p) = 0$$

的方程，这里p代表参数。边界条件必须能够确定 p，并且首先要对未知参数给定一个初始的猜测值，而 bvp4c 所解得的结果在 sol.parameters 中。

bvp4c 所得的解及其一阶导数在区间 $[a,b]$ 上是连续的，用指令 deval 及 bvp4c 可以求得区间 $[a,b]$ 中任何一点 xint 的值sxint = deval(sol,xint)。

由 bvp4c 返回的解 sol 是一个结构数组，其中包含以下的域：

sol.x	bvp4c 使用的变量的数据点
sol.y	与数据点 sol.x 对应的近似 y 值
sol.yp	与数据点 sol.x 对应的近似 y' 值
sol.parameters	由bvp4c 返回的未知参数的值
sol.solver	bvp4c 中使用的算法

结构数组 sol 的命名可以是任意的，bvp4c 会自动建立它的域名 x，y，yp，parameters 和 solver。

(2) **sol = bvp4c(odefun,bcfun,solinit,options)**

用法与 (1) 相同，对积分属性的要求不是默认的，而是由 options 指定的，这是由指令 bvpset 所建立的结构数组。详情参看 bvpset 的帮助文件。附带说明一下，指令 bvpset 是进入或修改 bvp4c 解方程的各种选项 (option structure)，指令 bvpget 是获得用 bvpset 所建立的各种选项的属性。

(3) **sol = bvp4c(odefun,bcfun,solinit,options,p1,p2...)**

将已知的参量 p1，p2，$\cdots$，传递给 odefun，bcfun，和所有用户在 options 指定的函数。采用 options = [ ] 表示不改变原来的设置。

3. 微分方程函数文件 odefun

求解的微分方程用函数文件 odefun 来设定，要求写成一阶的微分方程组

$$y' = f(x, y)$$

的形式，它的写法与解常微分方程时用的函数文件相同，有以下几种：

dydx = odefun(x,y)

dydx = odefun(x,y,p1,p2,...)

dydx = odefun(x,y,parameters)

dydx = odefun(x,y,parameters,p1,p2,...)

其中，x 是一个标量，y 是一个列矢量，parameters 是代表未知量的矢量，p1，p2，$\cdots$ 是未知的参数。输出量 dydx 是列矢量。

4. 边界条件函数文件 bcfun

边界条件是函数或者其导数在边界上的值，有两种类型

$$bc(y(a), y(b)) = 0$$

$$bc(y(a), y(b), p) = 0$$

它是用函数文件 bcfun 来设定的。它的形式为：

res = bcfun(ya,yb)

res = bcfun(ya,yb,p1,p2,...)

res = bcfun(ya,yb,parameters)

res = bcfun(ya,yb,parameters,p1,p2,...)

其中，ya 和 yb 是分别对应 $y(a)$ 和 $y(b)$ 的列矢量。parameters 是未知参量的列矢量。而 p1，p2，$\cdots$ 是已知参量。输出 res 是列矢量。

5. 指令 bvpinit 的用法

指令 bvpinit 建立微分方程的初始猜测解 solinit。这是一个结构数组，结构数组可以任意命名，但是它的域名必须是 x，y 和 parameters。

指令 bvpinit 的语句格式有：

solinit = bvpinit(x,v)

solinit = bvpinit(x,v,parameters)

solinit = bvpinit(x,v,parameters,p1,p2...)

solinit = bvpinit(sol,[anew bnew])

solinit = bvpinit(sol,[anew bnew],parameters)

solinit 是具有如下域名的结构数组，它本身可以任意命名，但它包含的域名必须是 x，y 和 parameters。几个域名的含义是：

x	变量初始的有序的数据点。边界条件由 a = solinit. x(1) 和 b = solinit. x(end)确定。
y	一个猜测的初始解，solinit.y(:,i) 是这个猜测的初始解在节点 solinit. x(i) 的值。
parameters	可选项，它是矢量，给出未知参量一个初始的猜测值。

下面对这几种语句格式加以解释。

(1) **solinit = bvpinit(x,v)**

在一个通用的环境中生成 bvp4c 要用的初始猜测值v，而 x 是初始指定的数据矢量。如果是在区域 $[a,b]$ 上解边值问题，则有 $x(1)=a$ 和 $x(\text{end})=b$。bvp4c 将在求解中使用这个数据，一般来说，猜测值取 x = linspace(a,b,10) 就够了。然而在复杂的情形下，在方程的解变化剧烈的地方必须要多放置一些数据点。各个 x 必须是排序的而且各不相同，所以如果 $a<b$，则有 $x(1)<x(2)<\cdots<x(\text{end})$；而对 $a>b$，则大小的顺序也相反。

v是一个猜测解。它可以是矢量或函数。

如果是矢量，则对于解的每个分量，bvpinit 将复制矢量对应的分量作为所有网格点上常数猜测值。也即 $v(i)$ 是解的第 i 个分量 $y(i,:)$ 在 x 网格上所有点的猜测值。

如果是函数则对于给定的数据点网格，函数将返回一个矢量，它的元素是解的对应分量的一个猜测值。例如，如果对于 $[a,b]$ 之中的任何 x，yfun(x) 的返回解 $y(x)$ 的一个猜测值，则可以使用语句 solinit = bvpinit(x,@yfun)。

(2) **solinit = bvpinit(x,v,parameters)**

这个格式使用了未知参数。当使用参数矢量时是对所有未知参数提供一个

猜测值。

(3) **solinit = bvpinit(x,v,parameters,p1,p2...)**

传递附加的参量给猜测的函数 v，这时未知的参数已经给定。这种格式只有在 v 函数时才能使用。

(4) **solinit = bvpinit(sol,[anew bnew])**

从区域 $[a,b]$ 上的解 sol 生成的在新区域 [anew bnew] 的一个猜测解 solinit。新的区域必须大于原有的区域。可以是 anew$\leqslant a<b\leqslant$ bnew 或者 anew $\geqslant a>b\geqslant$ bnew。原来的解是外推到新的区域的。如果在 sol 中含有参数，则它们会复制到 solinit 中。

(5) **solinit = bvpinit(sol,[anew bnew],parameters)**

用法与 (3) 相同，但是使用猜测的参数以替代未知的参数。

§5.11.2 有两个解的边值问题

边值问题可能有多重解，初始猜测的一个目的就是指定我们想要的解。下面的二阶常微分方程

$$y''+|y|=0$$

有两个解析解满足边值条件

$$y(0)=0,\quad y(4)+2=0$$

我们用 bvp4c 来求这两个解，这个问题说明不同的初始猜测值会导致不同的解。

引入 $y_1=y, y_2=y'$，将方程改写为

$$\begin{cases} y_1'=y_2 \\ y_2'=-|y_1| \end{cases}$$

这就是 bvp4c 编程所需要的如下形式 $y'=f(x,y)$，$bc(y(a),y(b))=0$。它们分别对应下面程序中的子函数文件 twoode 和 twobc。

在程序中，语句

```
solinit = bvpinit(linspace(0,4,5),[1 0]);
```

在 [0,4] 之间 5 个等间距的点组成了一个初始的变量数据网格，而猜测值则是

$$y_1(x)=1,\quad y_2(x)=0$$

解方程的语句是：

```
sol = bvp4c(@twoode,@twobc,solinit);
```

以下的语句是求出在 100 个等间距上的点的数值解并作图：

```
x = linspace(0,4);
y = deval(sol,x);
```

```
plot(x,y(1,:));
```

使用另一个猜测，将得到另外一个解：

```
solinit = bvpinit(linspace(0,4,5),[-1 0]);
```

完整的程序如下：

```
function twobvp
% One solution is obtained using an initial guess of y1(x)=1,
%y2(x)=0
solinit = bvpinit(linspace(0,4,5),[1 0]);
sol = bvp4c(@twoode,@twobc,solinit);
x = linspace(0,4);
y1 = deval(sol,x);
figure;
plot(x,y1(1,:));
xlabel('x');
ylabel('y');

% The other solution is obtained using an initial guess of
  y1(x)=-1,
% y2(x)=0
solinit = bvpinit(linspace(0,4,5),[-1 0]);
sol = bvp4c(@twoode,@twobc,solinit);
y2 = deval(sol,x);

% Plot both solutions
figure;
plot(x,y1(1,:),x,y2(1,:));
xlabel('x');
ylabel('solution y');
title('A BVP with two solutions');

function res = twobc(ya,yb)
res = [    ya(1)
        yb(1) + 2 ];

function dydx = twoode(x,y)
dydx = [    y(2)
         -abs(y(1)) ];
```

程序画出的图形为图 5.23。

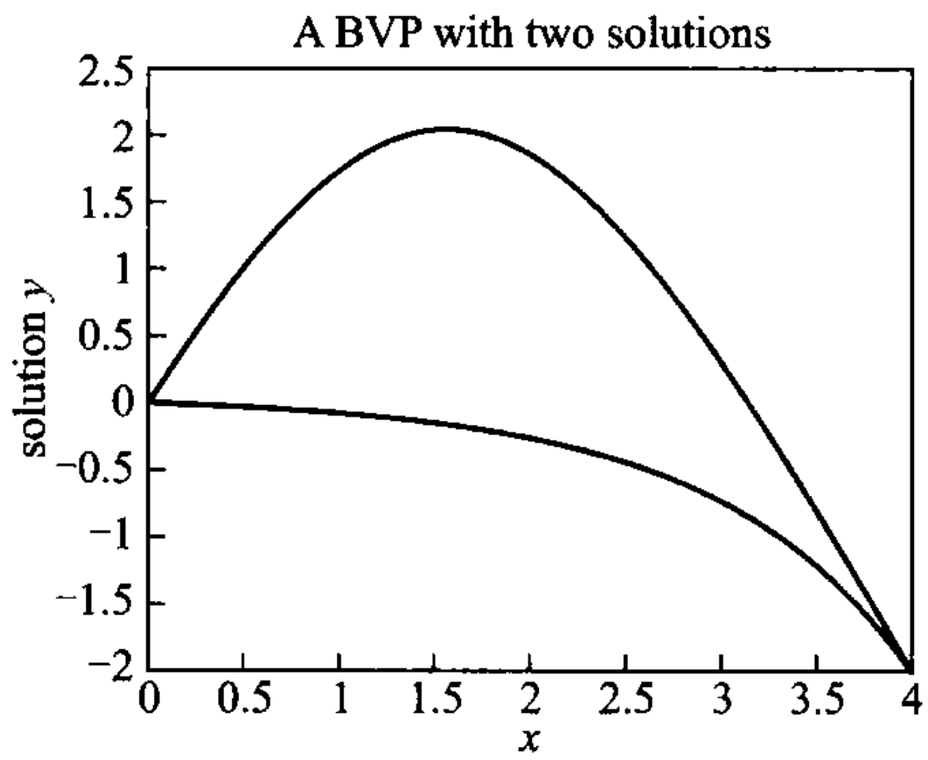


图 5.23 有两个不同解的边值问题

§5.11.3 马蒂厄方程的本征值

我们要计算的是马蒂厄 (Mathieu) 方程

$$\begin{cases} y'' + [(\lambda - 2q\cos 2x)]y = 0 \\ y'(0) = 0, \quad y'(\pi) = 0 \end{cases}$$

的第 4 个本征值。其中 $q = 5$。

要用专门的编程来计算施图姆-刘维本征值问题。而 BVP4C 只能求出最接近猜测值的本征值。可以通过使猜测的函数具有与第 4 个本征函数一样的定性的行为来使做到这一点。

所求得的本征函数会差一个常数因子，所以增加一个规范化的条件 $y(0) = 1$ 来得到一个特定的函数。用 bvp4c 所得到的网格作图并不能产生一条光滑的曲线。解 $S(x)$ 是连续的，其导数也是连续的。为了得到光滑的曲线，可以用 deval 来计算所增加的点上的数据。

在程序中，子函数 mat4ode 是转换成一阶常微分方程组的微分方程，子函数 mat4bc 是边界条件。由于出现了未知参量，这些函数都必须接受三个输入参量，即使是某些量可能没有用到。猜测的初始结构数组 solinit 由子函数 bvpinit 来构造。任何一个初始的猜测解都必须由函数 mat4init 所决定的形式来提供。我们选择 $y = \cos 4x$ 是因为它和它的导数满足边界条件并且有正确的定性行为 (正确的符号改变次数)，bvpinit 用以在 10 个等间隔的点上产生一个初始的猜测值。在调用 bvpinit 时，第三个量 ($\lambda = 15$) 给出了本征值的初始猜测值。

bvp4c 返回的解是结构数组 sol，本征值在域名 sol.parameters 下。这里，域 sol.parameters 返回的值 $\lambda = 17.097$，而图形则是与此本征值相联的本征函数的图形。程序及其画出的图形如下 (图 5.24)：

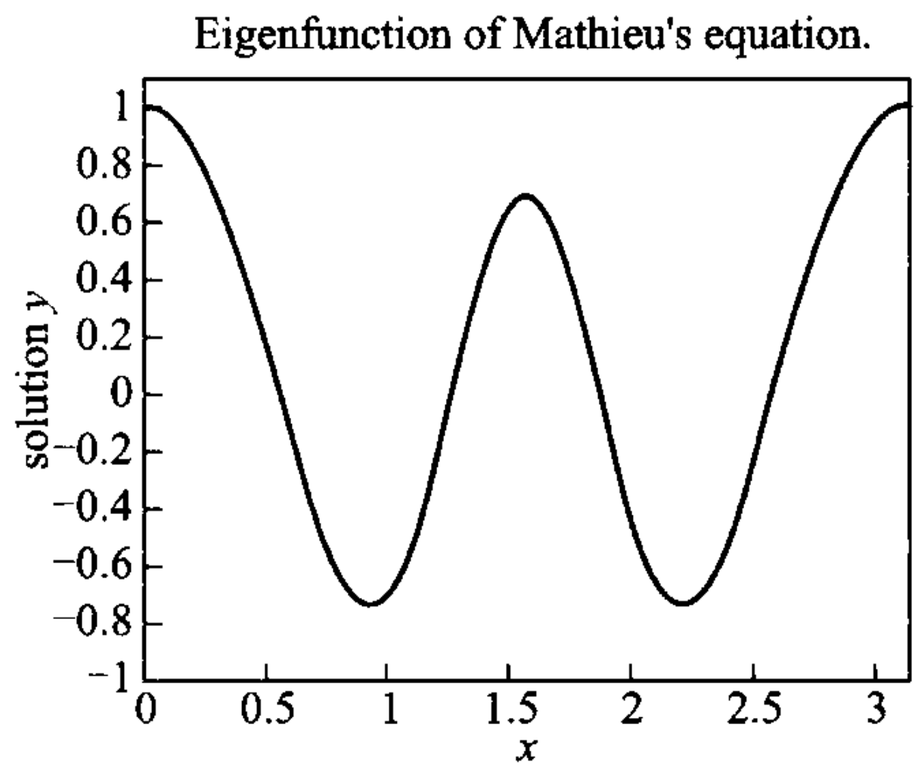


图 5.24 马蒂厄方程本征函数图

```
function mat4bvp
lambda = 15;
solinit = bvpinit(linspace(0,pi,10),@mat4init,lambda);

sol = bvp4c(@mat4ode,@mat4bc,solinit);

fprintf('The fourth eigenvalue is approximately...
     %7.3f.\n',sol.parameters)
     xint = linspace(0,pi);
     Sxint = deval(sol,xint);
     figure;
     plot(xint,Sxint(1,:));
     axis([0 pi -1 1.1]);
     title('Eigenfunction of Mathieu''s equation.');
     xlabel('x');
     ylabel('solution y');

     function dydx = mat4ode(x,y,lambda)
     q = 5;
     dydx = [                y(2)
              -(lambda - 2*q*cos(2*x))*y(1) ];

     function res = mat4bc(ya,yb,lambda)
     res = [  ya(2)
              yb(2)
```

```
        ya(1)-1 ];

function yinit = mat4init(x)
yinit = [   cos(4*x)
          -4*sin(4*x) ];
```

§5.11.4 艾登方程

要解的方程是艾登 (Emden) 方程，形式如下

$$y'' + \frac{2}{x}y' + y^5 = 0 \quad (0 \leqslant x \leqslant 1)$$

艾登方程来自于一个气体球的模型，这是一个有奇异项的方程。由于对称性而简化成上述常微分方程。$x = 0$ 是系数 $2/x$ 的奇点，但是，对称性意味 $y'(0) = 0$，因此，根据这个边界条件，当 $x \to 0$ 时，$(2/x)y'(x)$ 是有定义的。而另一端的边界条件是 $y(1) = \sqrt{3/4}$，所以这个边值问题有解析解。

指令 bvp4c 能解形式为

$$y' = \frac{S}{x}y + f(x, y)$$

的具有奇异性的问题。矩阵 S 必须是常数，并且 $x = 0$ 的边界条件必须与必要条件 $Sy(0) = 0$ 自洽。S 是作为 odeset 中的属性 'SingularTerm' 的值传递给 bvp4c 的。在其他方面，这个问题的解法与没有 $(S/x)y$ 的方程一样。

求解时，定义$y(1) = y$，$y(2) = y'$，将方程写成一阶方程组，而S=[0 0; 0 -2]。即

$$\begin{pmatrix} y'(1) \\ y'(2) \end{pmatrix} = \frac{1}{x}\begin{pmatrix} 0 & 0 \\ 0 & -2 \end{pmatrix}\begin{pmatrix} y(1) \\ y(2) \end{pmatrix} + \begin{pmatrix} y(2) \\ y^5(1) \end{pmatrix}$$

由于 $x = 0$ 的边界条件为 $y(2) = 0$，所以有

$$Sy(0) = \begin{pmatrix} 0 \\ -2y(2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

在程序中，guess 是一个满足边界条件的猜测常数，它表示猜测的解是

$$y(1) = \sqrt{3}/2, \quad y(2) = 0$$

最后在对计算结果画图 5.25 时，与这个问题的解析解 truy 作了对比。程序如下：

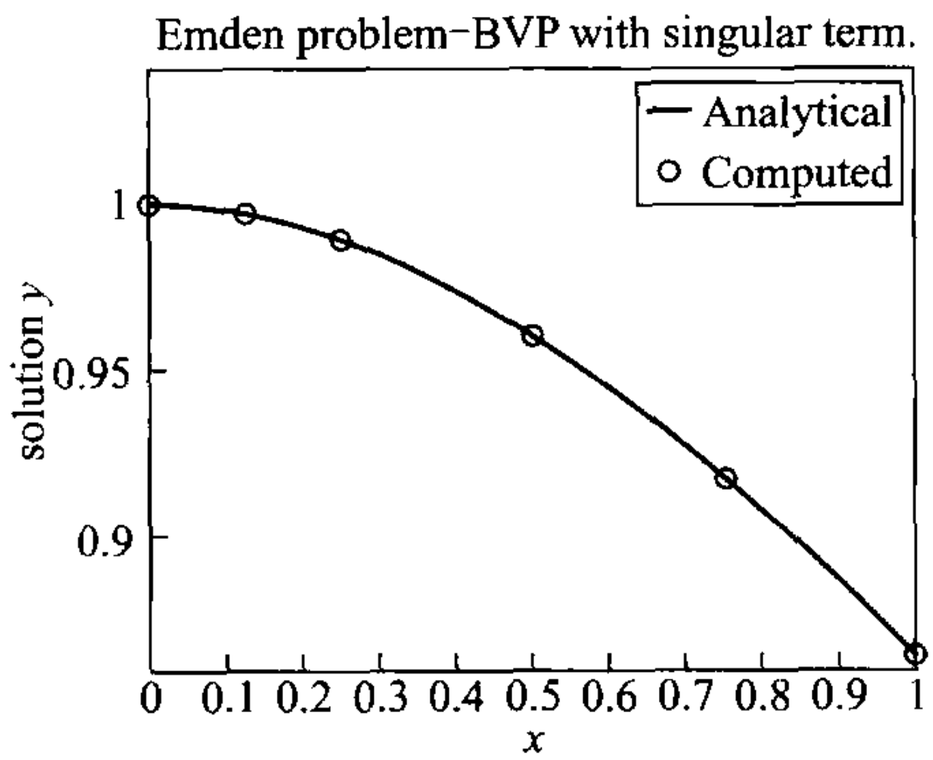


图 5.25 艾登方程图形

```
function emdenbvp

S = [0  0
     0 -2];
options = bvpset('SingularTerm',S);

guess = [sqrt(3)/2; 0];
solinit = bvpinit(linspace(0,1,5),guess);

sol = bvp4c(@emdenode,@emdenbc,solinit,options);

x = linspace(0,1);
truy = 1 ./ sqrt(1 + (x.^2)/3);

figure;
plot(x,truy,sol.x,sol.y(1,:),'ro');
title('Emden problem -- BVP with singular term.')
legend('Analytical','Computed');
xlabel('x');
ylabel('solution y');

% ---------------------------------------------------------------

function dydx = emdenode(x,y)
dydx = [  y(2)
```

```
            -y(1)^5 ];

% ----------------------------------------------------------------

function res = emdenbc(ya,yb) res = [ ya(2)
        yb(1) - sqrt(3)/2 ];
```

§5.11.5 Falkner-Skan 边值问题

Falkner-Skan 边值问题是解在薄板上的黏性的不可压缩的层流。方程是

$$\begin{cases} f''' + ff'' + \beta(1-(f')^2) = 0 \\ f(0) = 0, \quad f'(0) = 0 \\ f'(\infty) = 1 \end{cases}$$

其中 $\beta = 0.5$。这个问题是在不断地改变边界点。

这个问题的求解区域是 $0 \leqslant x < \infty$。在求解时，无穷远的边界是被代之以有限远点，不断地改变这个边界点，使它在足够大时能够收敛。并且从收敛的一致性验证结论的可靠性。在初值条件函数文件中，从一个足够大的边界点开始，求出一个解，然后改变边界点，使解也得到不断地伸展。

代替无穷远边界点的第一个猜测值是 3，它能满足边界条件也能得到很好的收敛。作为对比，还给出了原始文献中结果 $f''(0) = 0.92768$。程序如下，图 5.26 是解对应的图形。

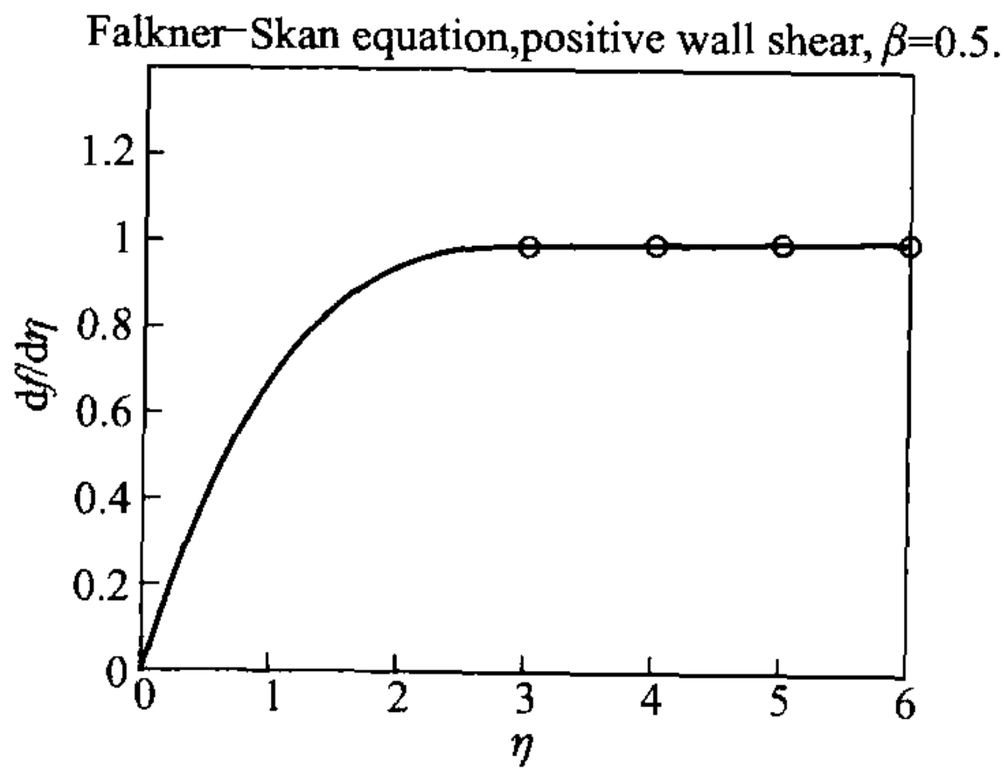


图 5.26 Falkner-Skan 边值问题

```
function fsbvp
infinity = 3;
maxinfinity = 6;

solinit = bvpinit(linspace(0,infinity,5),[0 0 1]);
```

```
sol = bvp4c(@fsode,@fsbc,solinit);
eta = sol.x;
f = sol.y;
fprintf('\n');
fprintf('Cebeci \& Keller report that f''''(0) = 0.92768.\n')
fprintf('Value computed using infinity = %g is...
        %7.5f.\n',infinity,f(3,1))

figure
plot(eta,f(2,:),eta(end),f(2,end),'o');
axis([0 maxinfinity 0 1.4]);
title('Falkner-Skan equation, positive wall shear,\beta = 0.5.')
xlabel('\eta')
ylabel('df/d\eta')
hold on
drawnow
shg

for Bnew = infinity+1:maxinfinity

  solinit = bvpinit(sol,[0 Bnew]); %Extend the solution to Bnew.
  sol = bvp4c(@fsode,@fsbc,solinit);
  eta = sol.x;
  f = sol.y;

  fprintf('Value computed using infinity = %g is %7.5f.\n',Bnew,
  f(3,1))
  plot(eta,f(2,:),eta(end),f(2,end),'o');
  drawnow

end
hold off

% ----------------------------------------

function dfdeta = fsode(eta,f)
beta = 0.5;
```

```
dfdeta = [ f(2)
       f(3)
          -f(1)*f(3) - beta*(1 - f(2)^2) ];

% ---------------------------------------------------------------

function res = fsbc(f0,finf)
res = [f0(1)
       f0(2)
       finf(2) - 1];
```

§5.11.6 在 $x=0$ 处有突变的问题

所解的方程是

$$\begin{cases} ey'' + xy' = -e\pi^2\cos\pi x - \pi x\sin\pi x & (-1 \leqslant x \leqslant 1) \\ y(-1) = -2, \quad y(1) = 0 \end{cases}$$

其中 $0 < e \ll 1$，e 是各个子函数共用的一个参数，bvp4c 要把它的值传递给所有的函数，如 shockODE，shockBC，shockJac and shockBCJac。

解在 $x=0$ 处有一个突变，本例介绍一种通过网格选择的技巧来解决这个问题。本例表明，在数值解上很困难的问题却可以借助于连续性来成功地解决。在这个问题中，解析的偏导数很容易求出，可以用它来帮助解题。

将方程写成一阶的常微分方程组，然后用 shockODE 和 shockBC 对它和边界条件分别编程。它们的偏导数分别用 shockJac 和 shockBCJac 编程并且用 options 传递给算法指令。选项 Vectorized 提示算法指令，微分方程是矢量化的，也即 shockODE([x1 x2 ...]，[y1 y2 ...]，e)，返回的是 [shockODE(x1,y1,e) shockODE(x2,y2,e) ...]。这种编程会提高程序的效率。

程序如下，图 5.27 是解的图形。

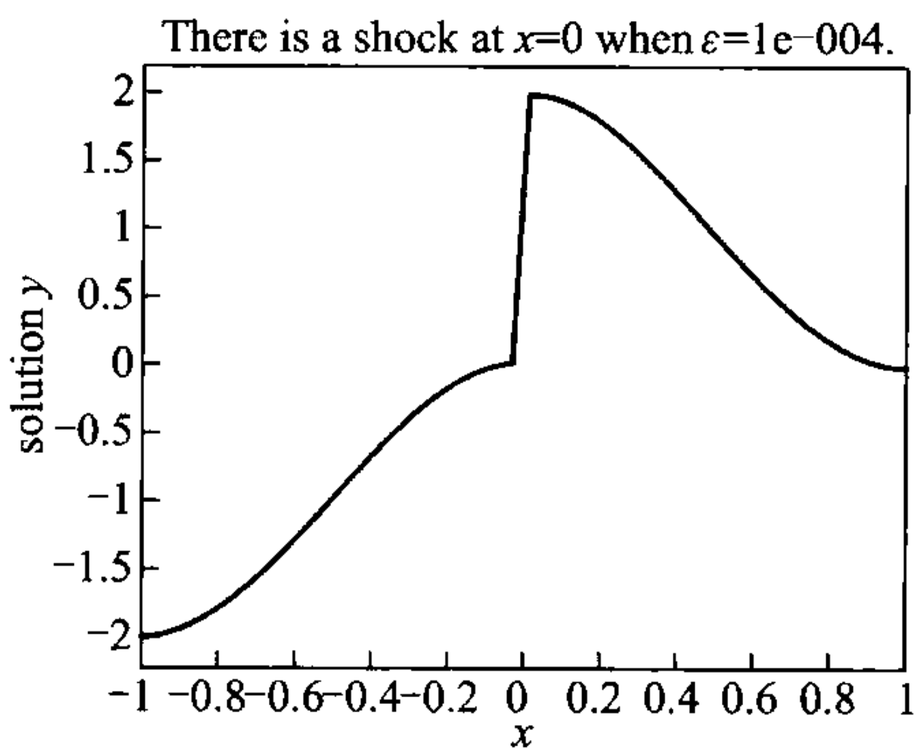


图 5.27 在 $x=0$ 处有突变的问题的解

```
function shockbvp
options = bvpset('FJacobian',@shockJac,'BCJacobian',...
          @shockBCJac,'Vectorized','on');

sol = bvpinit([-1 -0.5 0 0.5 1],[1 0]);     %猜测初始网格与初始解
for i=2:4                    %由连续性得到的当e = 1e-2, 1e-3, 1e-4时的解
    e = e/10;
    sol = bvp4c(@shockODE,@shockBC,sol,options,e);
end

figure;
plot(sol.x,sol.y(1,:));
axis([-1 1 -2.2 2.2]);
title(['There is a shock at x = 0 when \epsilon...
=' sprintf('%.e',e) '.']);
xlabel('x');
ylabel('solution y');
   function dydx = shockODE(x,y,e)            %计算矢量化的ODE函数
pix = pi*x;
dydx = [ y(2,:)
         -x/e.*y(2,:)  - pi^2*cos(pix) - pix/e.*sin(pix) ];
function res = shockBC(ya,yb,e)                          %计算边界条件
res = [ ya(1)+2
        yb(1) ];

function jac = shockJac(x,y,e)                           %计算雅可比行列式
jac = [ 0 1
0 -x/e ];

function [dBCdya,dBCdyb] = shockBCJac(ya,yb,e)
                                                  %计算边界条件的偏导数
dBCdya = [ 1 0
           0 0 ];
dBCdyb = [ 0 0
           1 0 ];
```

§5.12 练 习

1. 研究质点在平方反比引力场中的运动，例如行星绕太阳的运动。设质量为m_0的质点位于力心且固定不动，质量为m的质点在m_0产生的引力场中运动，当m与m_0相距r时，质点所受万有引力为$F = Gm_0m/r^2$，G为引力常量。要求：

(1) 当质点总能量大于、等于和小于零时，画出质点在平方反比引力场中的运动轨迹。

(2) 当质点总能量小于零且保持不变时，改变角动量的大小，画出质点相应的运动轨迹。

(3) 学习将极坐标变换成直角坐标的方法以及利用对称性画曲线的方法。

2. 研究平方反比斥力场中粒子的运动。以α粒子在重核场中的运动为例，设重核位于力心且固定不动，α粒子的质量为m，它到重核的距离为r，所受到库仑斥力为$F = k/r^2$，k为由库仑定律确定的常量。要求:

(1) 画出α粒子在不同初始条件下的轨道，通过改变初始条件研究影响散射角的因素。

(2) 学习根据解决问题的需要来选择坐标系，本题就是选择直角坐标系而不是极坐标系。

3. 研究水星近日点的进动。由于广义相对论对万有引力定律的修正，引起水星运动轨道的进动，水星的空间轨道不再是闭合的椭圆轨道。广义相对论对万有引力的修正可以归结为在原来的运动方程中增加一个小的修正项ε/r^4，其中$\varepsilon = 3Gm_0mh^2/c^2$是小量，$G$为万有引力常量，$m_0$为太阳质量，$m$为水星质量，$c$为真空中的光速，$h$为水星掠面速度的两倍。要求:

画出水星运动轨道。验证只要质点在有心力场中所受的力与平方反比引力有微小偏离，其轨道就不是闭合椭圆，从而证明广义相对论对万有引力定律的修正将引起椭圆轨道进动。

4. 小球与弹簧的碰撞。本题是练习事件函数events的应用。

如图 5.28 所示，地面上有垂直放置的一弹簧，弹簧上端固定有一个木块。有一个小球从空中自由下落，与木块发生弹性碰撞。设小球及木块均只沿垂直方向运动，木块保持平动且顶面与地面平行，研究小球与木块运动。小球质量为m_1，木块质量为m_2。弹簧劲度系数为k。要求画出小球与木块的位移曲线并模拟两物体的运动状况。计算时，小球与弹簧发生碰撞时刻由events来判断，此时小球离地面的高度与弹簧的高度相等，由于碰撞过程十分短暂，可以忽略碰撞过程中二者高度的变化，认为二者遵循动量守恒、机械能守恒。在其他的时间，小球与弹簧的运动微分方程遵循牛顿定律。

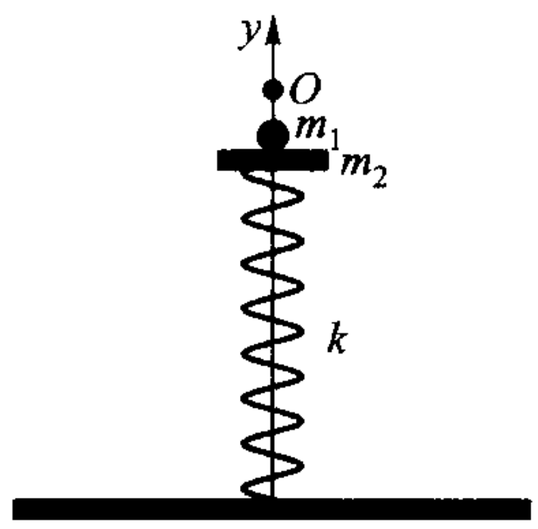


图 5.28　习题 4 的示意图

将弹簧水平放置并固定好，让在桌面上作水平运动的小球与它碰撞，重新研究小球与弹簧的运动。

5. 三质点系统的微振动。研究两个弹簧连接三个质点组成的一维振动系统的运动。如图 5.29 所示，其中弹簧的劲度系数均为k，中间的质点的质量为m_0，两端质点的质量都为m。要求：

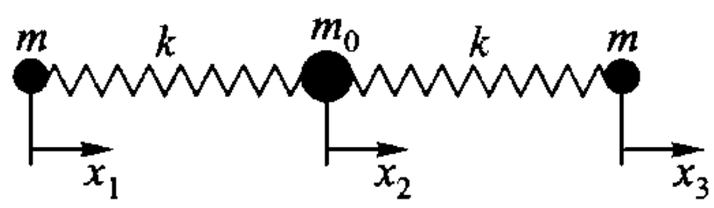


图 5.29

(1) 用三种不同的方法即矩阵方法、快速傅里叶变换法和拉普拉斯变换法分别求出振动系统的简正频率。从而证明这三种方法所得的结果是相同的。本题将这三种方法放在一起，通过对比，读者可以了解它们之间的内在联系。

(2) 将拉普拉斯变换法得出的系统的运动微分方程的解析解做成动画模拟。

(3) 学习快速傅里叶变换法和拉普拉斯变换法的用法。

6. 推导本章中陀螺的运动微分方程组和变换关系。对不同的初始条件同时画出描述陀螺的章动角、进动角和自转角的位移，速度和加速度随时间演变的图形。

7. 推导本章中木杆沿光滑墙面下滑的两个阶段的运动微分方程。

8. 用下面几个运动系统验证方言公式：

(1) 用轻质吊杆挂在天花板上的圆盘在重力的作用下摆动 $\left(\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}=\mathbf{0}\right)$。

(2) 在圆弧形轨道上作无滑动滚动的圆盘 $\left(\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}\times\boldsymbol{p}=\mathbf{0}\right)$。

(3) 上抛的哑铃，哑铃在作抛体运动的同时还有绕质心的平面转动 $\left(\dfrac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}t}\times\boldsymbol{p}\neq\mathbf{0}\right)$。

9. 误差函数的定义为

$$\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-x^2}\mathrm{d}x$$

也可以视为微分方程

$$y'(x)=\frac{2}{\sqrt{\pi}}\mathrm{e}^{-x^2},\quad y(0)=0$$

的解。在区间 $0\leqslant x\leqslant 2$ 用ode23求解这个方程，并将结果与MATLAB的内部函数erf(x)比较。

10. 常微分方程

$$\dot{y}=-1000(y-\sin t)+\cos t,\qquad y(0)=1$$

在区间 $0\leqslant t\leqslant 1$ 上是略为刚性的。

(1) 自己推导出方程的精确解，也可以用指令dsolve求解。

(2) 利用指令ode23求解，需要多少步?

(3) 利用指令ode23s求解，需要多少步?

(4) 将两种解绘在同一幅图上，分别用“x”和“o”来表示它们。

(5) 将图像放大，仔细观察图像变化剧烈的地方，比较这两种方法的步长大小。

(6) 再观察图像变化缓慢的地方，再比较这两种方法的步长，哪种方法的步长更大，也即是哪种方法的计算效率更高?

11. Lotka-Volterra捕食者-被掠食者模型是数学生态学中的一个经典模型。考虑一个简单的生态系统。包含有无限食物供给的兔子和依靠捕食兔子为生的狐狸群体。这个系统可以通过两个非线性的一阶微分方程来建模：

$$\frac{\mathrm{d}r}{\mathrm{d}t}=2r-\alpha rf,\qquad r(0)=r_0$$

$$\frac{\mathrm{d}f}{\mathrm{d}t}=-f+\alpha rf,\qquad f(0)=f$$

其中 t 为时间，$r(t)$为兔子的数量，$f(t)$为狐狸的数量，α为一个正的常数。如果$\alpha=0$，这两个种群没有关系，兔子很好地生存，而狐狸因饥饿而死亡。如果$\alpha>0$，狐狸捕食与自身数量成比例的兔子，这种捕食会造成兔子数量的减少和狐狸数量的增加。

这个非线性系统的解无法写成已知的其他函数的组合，只能通过数值求解。可以证明该系统的解有周期性，其周期取决于初始条件。也就是说，对任意的$r(0)$和$f(0)$，都存在一个时间$t=t_p$，此时这两个种群的数量都等于初始值。因此对所有的t有

$$r(t+t_p)=r(t),\quad f(t+t_p)=f(t)$$

(1) 取$r_0=300$，$f_0=150$以及$\alpha=0.01$，求系统的解。结果应该是$t_p\approx 5$。绘制出$r(t)$，$f(t)$，$\dot{r}(t)$图形。

(2) 计算并画出$r_0 = 15$，$f_0 = 22$以及$\alpha = 0.01$时的解，计算出的周期应该是$t_p \approx 6.62$。

(3) 计算并画出$r_0 = 102$，$f_0 = 198$以及$\alpha = 0.01$时的解，计算出周期t_p，可以通过误差试算或事件函数。

(4) 点$r_0 = 1/\alpha$，$f_0 = 2/\alpha$是一个稳定的平衡点，如果初值为此值，那么种群的数量不会变化。如果初值不等于此值但比较接近该平衡点，那么种群数量不会发生大的变化。令

$$u(t) = r(t) - 1/\alpha, \quad v(t) = f(t) - 2/\alpha$$

那么 $u(t)$ 和 $v(t)$ 满足另外一个非线性方程组。但如果忽略 uv 项，方程组变为线性的。试问这个线性方程组是什么？它的周期解是多少？

12. 已经提出了多种对 Lotka-Volterra 捕食者–被掠食者模型的修改模型，以便更精确地反映自然中的实际情况。比如，如下的修改就是为了防止兔子数量无限的增长：

$$\frac{\mathrm{d}r}{\mathrm{d}t} = 2\left(1 - \frac{r}{R}\right)r - \alpha r f, \quad r(0) = r_0$$

$$\frac{\mathrm{d}f}{\mathrm{d}t} = -f + \alpha r f, \quad f(0) = f_0$$

其中t为时间，$r(t)$为兔子的数量，$f(t)$为狐狸的数量，α 和R都是正的常数。由于α为正常数，当$r \geqslant R$时$\dfrac{\mathrm{d}r}{\mathrm{d}t}$ 为负，因此兔子的数量就永远不会超过R。对于$\alpha = 0.01$，比较原始模型和$R = 400$时的修改模型的特性。在比较的过程中，取$r_0 = 300$，$f_0 = 150$在50个时间单位上求解方程，并绘制四个不同时间的图：

(1) 原始模型狐狸数量和兔子数量相对于时间的图；

(2) 修改模型狐狸数量和兔子数量相对于时间的图；

(3) 原始模型狐狸数量相对于兔子数量的图；

(4) 修改模型狐狸数量相对于兔子数量的图。

对于所有的图，标记所有的曲线和坐标轴并给出标题，对后两个图，设置纵横比以便保证 x 和 y 坐标轴上同样增量对应的长度相同。

13. 计算有限深方势阱内能级，并与量子力学教材中的解析解对比。

14. 用指令 bvp 计算谐振子的最低 2 个能级。

第六章　混　　沌

混沌现象的研究是非线性物理的重要内容，数值计算则是非线性物理的主要研究工具之一，其中大量地涉及常微分方程的求解与研究。所以，适当地介绍一些这方面的内容，能更深入地理解如何借助常微分方程来研究物理现象。对混沌感兴趣的读者可以参看下列文献：

[1] Hao,B.L.,Chaos II ,World Scientific,Singapore,1990.
列出了 2244 篇学术论文清单并收录了Feigenbaum, Hénon, Lorenz 和 May 等人的原始论文。

[2] Zhang Shu-yu,Bibliography on Chaos——Directions in Chaos Vol.5,World Scientific, Singapore, 1991.
列出了超过 7000 篇参考文献的目录。

本章通过单摆，倒摆，自激振荡及描述天气动力学的洛仑茨方程等物理模型来介绍非线性物理，其中内容涉及：

(1) 不动点(平衡点)的分类；
(2) 吸引子类型及其频谱特征；
(3) 分岔的类型；
(4) 混沌运动的判别方法，分析方法及通向混沌的几条道路。

§6.1　单摆——从周期运动到混沌

§6.1.1　单摆的动力学方程

如图 6.1 所示，单摆摆长为l，小球的质量为m，相对于铅垂线的角位移为θ，所受阻力与速度成比例，阻尼系数为r，受到圆频率为ω_d的外驱动力F的作用，则运动方程为

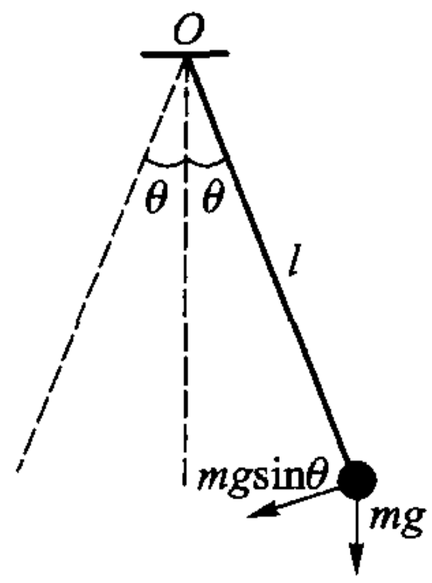


图 6.1　单摆的运动

$$ml\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+rl\frac{\mathrm{d}\theta}{\mathrm{d}t}+mg\sin\theta=F\cos\omega_d t$$

公式两边除以mg，并引入$\omega_0=\sqrt{g/l}$，ω_0是单摆在没有阻尼没有外力时作简谐振动的频率，也就是固有频率。则方程为

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}(\omega_0 t)^2}+\frac{r}{m\omega_0}\frac{\mathrm{d}\theta}{\mathrm{d}(\omega_0 t)}+\sin\theta=\frac{F}{mg}\cos\omega_d t$$

这时在公式中的无量纲量有 (以方括号表示量纲)

$$[\omega_0 t]=T^{-1}\cdot T$$

$$\left[\frac{r}{m\omega_0}\right]=\frac{MT^{-1}}{MT^{-1}}$$

$$\left[\frac{F}{mg}\right]=\frac{MLT^{-2}}{MLT^{-2}}$$

$$[\omega_d t]=T^{-1}\cdot T$$

注意 θ (以弧度计算) 是一个无量纲量。如果令

$$\omega=\frac{\omega_d}{\omega_0},\quad \tau=\omega_0 t,\quad \beta=\frac{r}{2m\omega_0},\quad f=\frac{F}{mg}$$

得到无量纲化运动方程是

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}\tau^2}+2\beta\frac{\mathrm{d}\theta}{\mathrm{d}\tau}+\sin\theta=f\cos\omega\tau$$

公式无量纲化相当于对公式中的物理量选择新的基本量度单位，不妨说是“选择了一个新单位制”。公式中原来用的基本单位是 m（质量）、L（长度）、T（时间），重新选择的 3 个互相独立的基本量度单位是 ω_0(频率量度单位，其倒数可用来作时间量度单位)、mg(力的量度单位)、$m\omega_0$（阻尼系数量度单位)，再以新的单位来量度公式中的各个量，则得出的β，f，ω，τ都是纯数。

新方程比旧方程简单，计算中涉及的都是纯数，不必再考虑物理量的单位。在公式无量纲化时，通常是选择公式中物理量的特征量作为量度单位，如这里就是用单摆的固有频率ω_0作频率单位来量度外驱动力的频率，它的倒数正好可以作时间量度单位（实质上是用固有频率的周期来量度时间），用杆受到的重力作用的量度单位，最后根据公式变换所得的表达形式来确定阻尼系数的量度单位。又如在前面的分子振动能级的计算中，是用势阱的深度作能量单位，用玻尔半径作长度单位，也是这个道理。这样量度所得的数值比较适中，不会太大也不会太小。这一点是十分重要的，它避免了在计算中去处理太大或者太小的数。

再解释一下公式无量纲化的意义。一个物理公式实际上一个物理规律的数学描述，其中涉及的物理量是确定的有限的。这些物理量都是用同一个单位制来量度，而单位制中的基本量是已经规定好了的，它们的量度单位就作为基本单位，其他物理量（叫作导出量）的单位是由基本单位组合而成的，也叫“导出单位”。可是在研究实际问题的许多方程中，基本量并不一定是会显性地出现，出现最多的往往是导出量甚至是导出量的组合。导出量的单位（导出单位）已经是基本单位的组合，而导出量的组合又涉及“导出单位”的组合，如单摆方程中的$\frac{r}{m\omega_0}$和$\frac{F}{mg}$，这样公式中的单位的表示变得十分复杂，同时也会给数值计算带来很多麻烦，因为选择不同的单位所得到的物理量的数值是不同的，计算中要时时注意，否则就会出错。那么能不能有一种方法，可以不涉及物理量单位的选择又能正确地进行计算呢？这就是公式无量纲化的目的。

试想一下，如果选用公式中实际出现的导出单位甚至是导出单位的组合作为新的基本量度单位，那么在原来的单位制中，这些导出量及导出量组合就是无量纲化的。例如在国际单位制中，单摆方程中的F与mg的量纲相同，实际上，力的单位就是由质量与加速度的单位导出的，以mg来量度F得到的就是纯数，在新单位制中是无量纲的。所以在上面的无量纲化过程中，mg就是新选的力的基本量度单位，也可以简单地说是选$mg=1$，因为如果用mg去量度mg 结果当然是1。按照这种解释，单摆的运动方程在上述无量纲化过程选用的“新单位制”就是$mg=1$，$\omega_0=1$，$m\omega_0=1$。新单位制简化了公式形式，在计算中也不必考虑公式中物理量单位的换算。由此也可以理解，在量子场论中使用自然单位制时，常常简单地说取$c=1$，$\hbar=1$是什么含义了。

还要注意的是，新的基本量度单位必须相互独立，且数目必须与旧的基本单位数目相同。这样就保证了旧的基本单位也能表示成新的基本单位的组合。例如在自然单位制中，除了选$c=1$，$\hbar=1$之外还可以根据问题的需要再选择一个新的基本单位，常见的选择是能量或者质量。当然不同的选择得出的方程形式是有差别的。

如果要将公式无量纲化以后的计算结果换算回到原来的单位制中，只要将物理量的值再乘以相应新的基本单位就可以。比如在本例中如果$f=2$，那么在国际单位制中，它的值就是$2mg$。

§6.1.2 周期运动

§6.1.2.1 无阻尼无驱动情况

此时有$\beta=0$，$f=0$，方程是

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+\sin\theta=0$$

1. 单摆周期的解析解

方程两边乘以 $\mathrm{d}\theta$ 再积分，并注意到单摆在最大摆角 θ_0 处的速度 $\mathrm{d}\theta/\mathrm{d}\tau=0$，得

$$\left(\frac{\mathrm{d}\theta}{\mathrm{d}\tau}\right)^2=2(\cos\theta-\cos\theta_0)$$

即

$$\frac{\mathrm{d}\theta}{\mathrm{d}\tau}=2\sqrt{\sin^2\frac{\theta_0}{2}-\sin^2\frac{\theta}{2}}$$

分离变量两边再积分，并设 $\tau=0$ 时有 $\theta=0$，得单摆运动到 θ 角的需要的时间为

$$\tau=\frac{1}{2}\int_0^\theta\frac{\mathrm{d}\theta}{\sqrt{\sin^2\frac{\theta_0}{2}-\sin^2\frac{\theta}{2}}}$$

引入$k=\sin\dfrac{\theta_0}{2}$得

$$\tau=\frac{1}{2}\int_0^\theta\frac{\mathrm{d}\theta}{\sqrt{k^2-\sin^2\frac{\theta}{2}}}$$

摆角从0到最大摆角θ_0所需要的时间为周期T的1/4，即

$$T=2\int_0^{\theta_0}\frac{\mathrm{d}\theta}{\sqrt{k^2-\sin^2\frac{\theta}{2}}}$$

再作变量变换

$$\sin\frac{\theta}{2}=k\sin\varphi,\qquad\frac{1}{2}\cos\frac{\theta}{2}d\theta=k\cos\varphi d\varphi$$

其范围为$0\leqslant\theta\leqslant\theta_0$，$0\leqslant\varphi\leqslant\pi/2$，所以

$$\mathrm{d}\theta=\frac{2k\cos\varphi\mathrm{d}\varphi}{\sqrt{1-k^2\sin^2\varphi}}=\frac{2\sqrt{k^2-\sin^2\frac{\theta}{2}}}{\sqrt{1-k^2\sin^2\varphi}}\mathrm{d}\varphi$$

最后得出的周期为

$$T=4\int_0^{\pi/2}\frac{\mathrm{d}\varphi}{\sqrt{1-k^2\sin^2\varphi}}=4K(k)$$

其中 $K(k)$ 是第一类椭圆函数。不要忘记这是无量纲化以后的计算结果，在国际单位制中的周期应该是

$$\frac{T}{\omega_0}=T\sqrt{\frac{l}{g}}$$

它的展开式是

$$T\sqrt{\frac{l}{g}}=2\pi\sqrt{\frac{l}{g}}\left(1+\frac{1}{2^2}\sin^2\frac{\varphi_0}{2}+\frac{1}{2^2}\frac{3^2}{4^2}\sin^4\frac{\varphi_0}{2}+\cdots\right)$$

如果当括号内的第二项小于千分之一时即可忽略,则最大摆角可取φ_0小于0.06325弧度，约3.6°。这时单摆运动的周期基本与初始角度无关，由摆长l和重力加速度g决定。在 MATLAB 中是用指令 ellipke 来计算椭圆函数，其用法是

[K,E] = ELLIPKE(M)

其中M就是这里的k^2，K，E分别是第一、二类椭圆函数。定义是

$$K(k^2)=\int_0^{\pi/2}(1-k^2\sin^2\varphi)^{-1/2}\mathrm{d}\varphi$$

$$E(k^2)=\int_0^{\pi/2}(1-k^2\sin^2\varphi)^{1/2}\mathrm{d}\varphi$$

这个公式可以计算任意最大摆角φ_0所对应的周期。

2. 单摆周期的数值解法

上面推导解析解时，假定单摆的运动是周期运动，这一点可以由数值解来验证。用ode45解无外力无阻尼的无量纲化方程。就能得到单摆的数值解。下面的程序以初位移分别为π/7，π/3 、初速度为零求解方程并画出了两条位移曲线如图 6.2。实线是小摆角，虚线是大摆角。可以看到，它们都是周期运动，初位移就是最大摆角，摆角增大周期也增长。

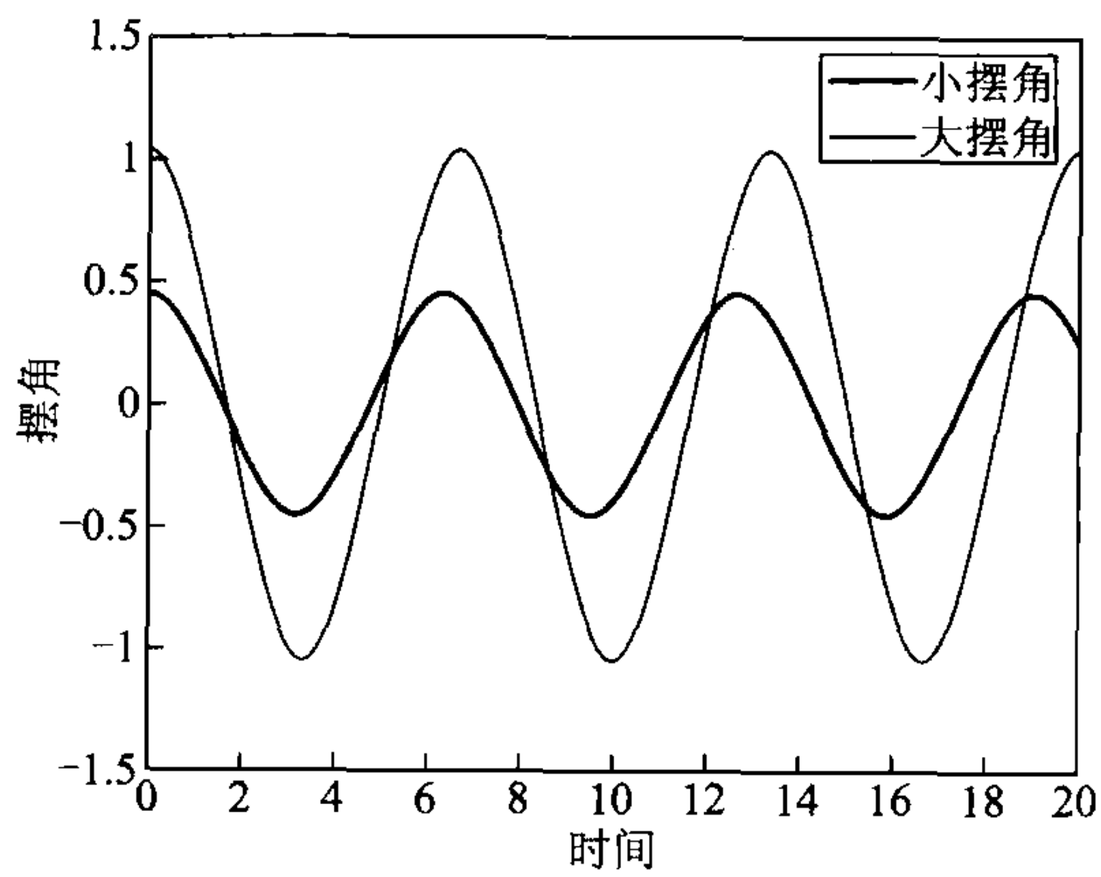


图 6.2 单摆的不同周期的位移曲线

```
function djdb
[t1,w1]=ode45(@f,[0,6*pi],[pi/10,0],[]);
[t2,w2]=ode45(@f,[0,6*pi],[pi/3,0],[]);
```

```
plot(t1,w1(:,1),'-',t2,w2(:,1),'k:' );
xlabel('t');
ylabel('\theta');

function ydot=f(t,y)
ydot=[y(2);    -sin(y(1))];
```

下面的程序画出周期与摆角的对应关系 (图 6.3)，其中用events指令来判断单摆从最大摆角θ到$-\theta$ 的时刻，此时$\dot{\theta}=0$，所需要的时间正好是周期的1/2。最大摆角的取值从0.02 到$\pi-0.02$，共选取了40个点。

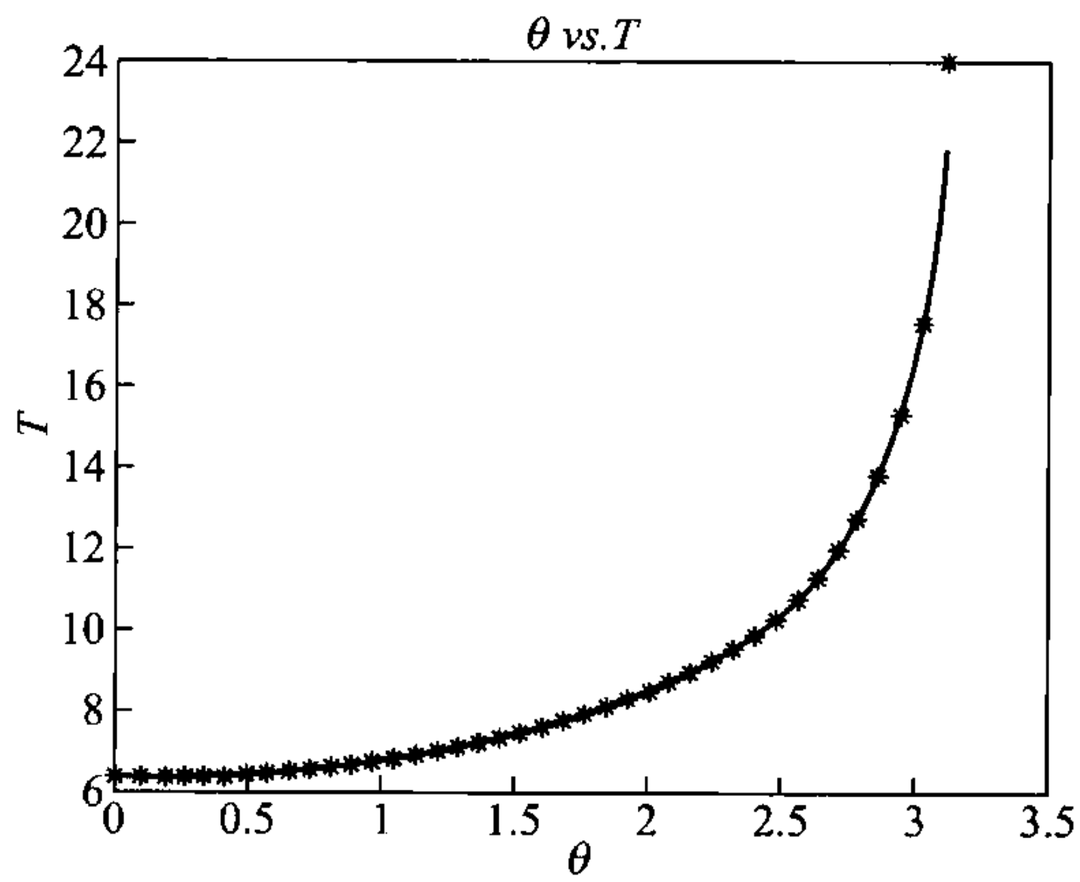


图 6.3 单摆的周期与摆角的关系

为了对比，将前面得到的用椭圆函数表示的解析解的结果也画在同一个图内，并用星号表示相应的数据点的值。

```
function djdb2
theta=linspace(0.02,pi-0.02,40);                    %最大摆角
T=[];
ops=odeset('Events',@events);
for k=1:40 ;                                        %不同摆角的周期
[t,u]=ode45(@f,[0,40],[theta(k),0],ops);
T=[T,2*t(end)];                                     %将周期存储在T
end

k2=sin(theta./2).^2;                                %解析解
[K,E]=ellipke(k2);
```

```
plot(theta,T,theta, 4*K,'*')
title('\theta   vs.  T');
xlabel('\theta'); ylabel('T');

function ydot=f(t,y)
ydot=[y(2);          -sin(y(1))];

function [value,isterminal,direction]=events(t,y)
value=y(2);    isterminal=1;    direction=1;
```

从图 6.3 可以看出，当最大位移小于 3.62° 时，周期曲线近似平行于水平轴，简谐振动的周期也近似不变。当最大位移大于 3.62° 小于 180° 时，是周期运动。当最大摆角接近 180° 时，周期趋于无穷大，实际上，$\varphi = \pi$ 时就是一个不稳定平衡点。

3. 相图

我们已经研究了单摆的角位移θ、角速度$\dot{\theta}$与时间t的关系，三者构成了一个三维空间，而我们的研究只是在t-θ 平面和 t-$\dot{\theta}$ 平面上进行的，其实在 θ-$\dot{\theta}$ 平面上研究单摆的运动也是一种重要的方法，这就是相图研究法，是法国数学家庞加莱 (Poincaré) 于 19 世纪末提出来的。θ-$\dot{\theta}$平面叫作相平面。相平面上的一个点表示了系统在某一时刻状态（摆角与角速度）称为相点，而相点的连续变化形成的轨迹叫相轨线，它描述了系统的运动过程。

如果通过考查微分方程的系数及其本身的结构画出了相图中的轨线，然后通过研究系统的相轨线了解了系统的动力学特性，那么不解微分方程也可能了解系统的运动情况。通过相图分析系统的运动是一种几何分析方法，所以也叫几何动力学。下面就用相图来重新研究单摆的运动。

将方程乘以 $\mathrm{d}\theta$ 作积分，得

$$\frac{1}{2}\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 - \cos\theta = C \tag{6.1}$$

对比能量守恒方程

$$\frac{1}{2}m(l\dot{\theta})^2 + mgl(1-\cos\theta) = E$$

可知方程 (6.1) 的左边第一项对应单摆动能，左边第二项对应势能V（相差常数），势能曲线是余弦函数$V = -\cos\theta$，右边积分常数C对应单摆总能。这是一个关于θ，$\dot{\theta}$隐函数方程，可以用它画出无阻尼无驱动单摆的相图如图 6.4。

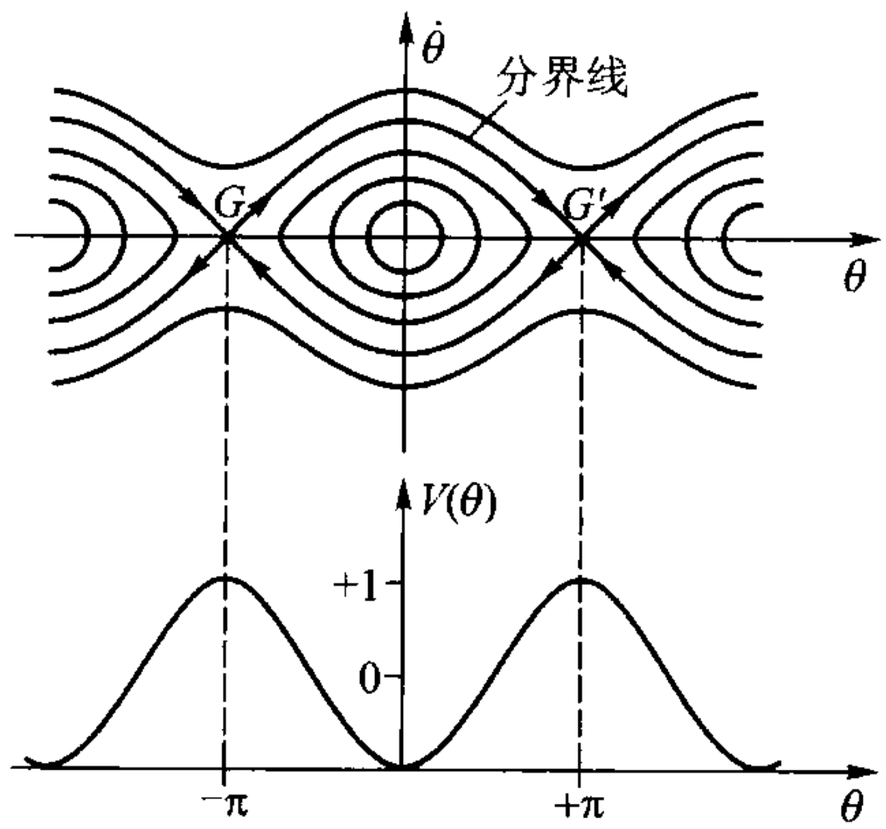


图 6.4 无阻尼无驱动时单摆的相图

使用 ezplot可以直接画隐函数图形，但要将隐函数方程写成等号右边为零的形式。为了对比，将势能曲线也画出。

```
ezplot('0.5*y^2-cos(x)+0.8')
hold on
ezplot('0.5*y^2-cos(x)+0')
ezplot('0.5*y^2-cos(x)-0.6')
ezplot('0.5*y^2-cos(x)-1')
ezplot('0.5*y^2-cos(x)-1.4')
```

势能变化范围为$-1 \leqslant V \leqslant 1$，单摆的总能$E$可以在这个范围变化也可以大于势能的最大值1。作图时选择了$E = 1$，$E < 1$，$E > 1$ 三种情况。现在虽然没有解方程，也可以通过相图来分析单摆的运动状态。

(1) $E < 1$

在相图上，这些轨线都是闭合的轨道，这反映单摆在作周期性的往返的运动。单摆的总能量越大，轨道包围的面积越大，反映出运动的周期越长，这与前面的摆角相对周期所画的图形的结论是一致的。能量小的时候轨道的形状近似于圆；能量逐渐增加，轨道变成椭圆；能量再增加，轨道形状发生畸变，两头变尖了，能量越大，畸变越大。

(2) $E > 1$

在相图上，轨线是上下都有的波浪线，它不是闭合曲线而是逃逸的，反映摆角是单向的增大或减小，这表明单摆是在旋转，旋转的方向取决于初速度的方向，可以顺时针方向也可以反时针方向。单摆的总能量越大，波浪线离θ轴越远。

(3) $E = 1$

在相图上，轨线是闭合轨线与波浪轨线的分界线。两种轨线是交叉的，将相

图划分成不同的区域。在方程 (6.1) 中令$E=1$得到分界线方程

$$\dot{\theta}=\pm 2\cos\frac{\theta}{2} \tag{6.2}$$

对它积分，取初始条件为$t=0$, $\theta=0$还能解出角位移

$$\theta=4\arctan \mathrm{e}^{t}-\pi \tag{6.3}$$

将 (6.3) 式代入 (6.2) 式

$$\begin{aligned}\dot{\theta}&=\pm 2\cos(2\arctan \mathrm{e}^{t}-\pi/2)\\&=\pm 2\sin(2\arctan \mathrm{e}^{t})\\&=\pm\frac{2}{\cosh t}\end{aligned} \tag{6.4}$$

推导时使用了三角函数公式

$$\sin 2\alpha=\frac{2\tan\alpha}{1+\tan^2\alpha}$$

画出 (6.4) 式的图形如图 6.5，它对时间的关系与孤立子（soliton）的波函数一样。

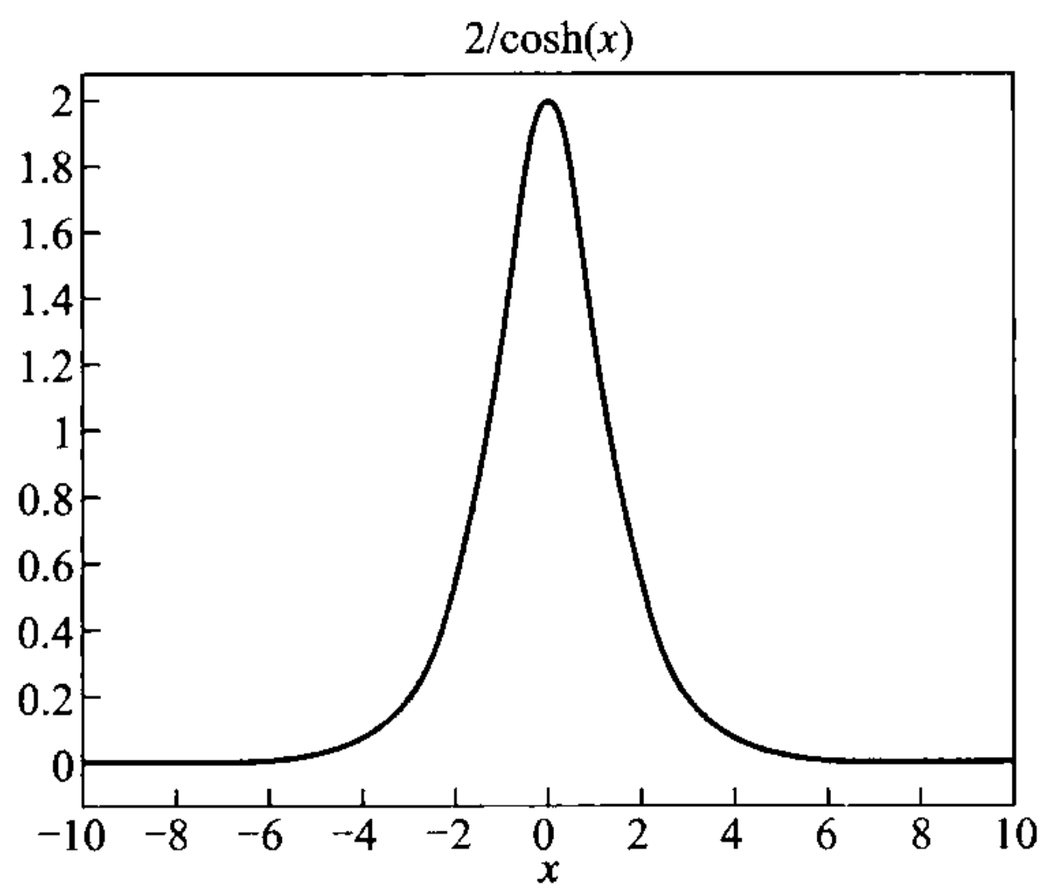


图 6.5 公式 (6.4) 对应的曲线

相图横坐标 θ 是以 2π 为周期的，摆角 $\pm\pi$ 是单摆的同一个倒立位置，把相图上 G 点与 G' 点重叠一起时，就把相平面卷缩成一个柱面，所有相轨线都将呈现在柱面上，如图 6.6 所示。周期性的摆动被限制在分界线内，而旋转运动则绕着圆柱不断延伸。

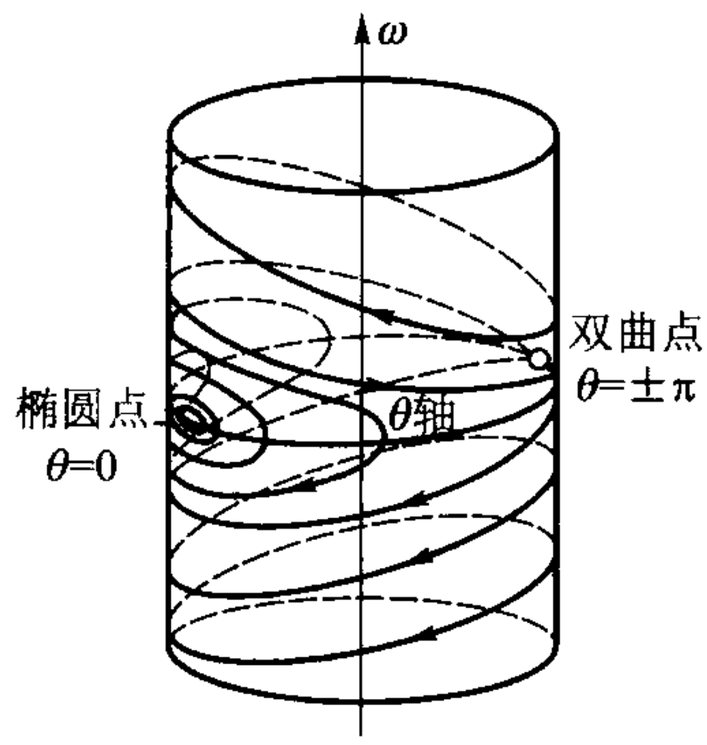


图 6.6 柱面上的相轨线

§6.1.2.2 有阻尼无驱动情况

此时 $\beta \neq 0$, $f = 0$, 方程是

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + 2\beta\frac{\mathrm{d}\theta}{\mathrm{d}t} + \sin\theta = 0$$

有阻尼时，由于能量被消耗，单摆振动的振幅会逐渐变小，其变化如图 6.7 所示有阻尼单摆运动的相图如图 6.8。

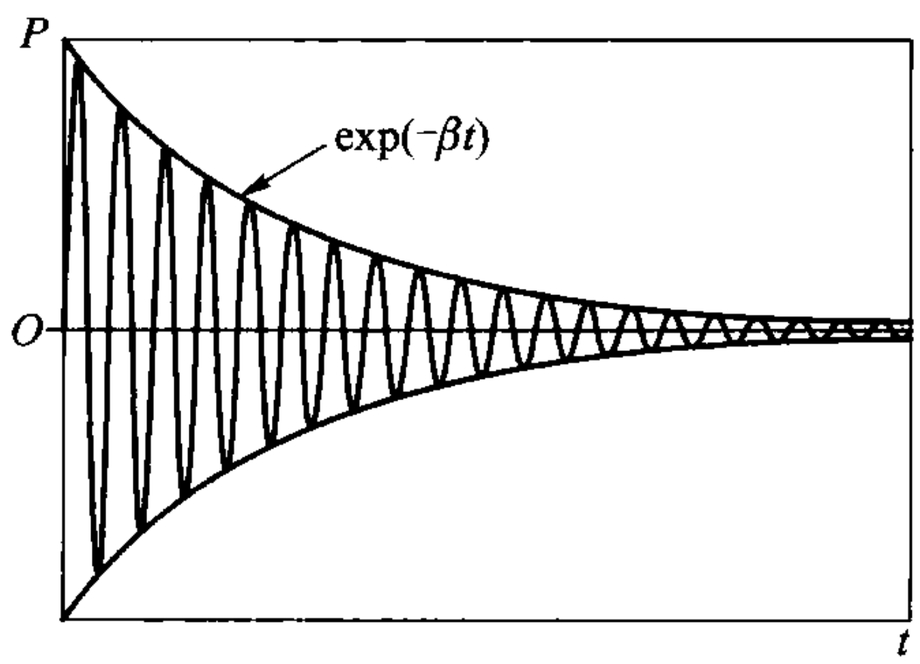


图 6.7 有阻尼无驱动的单摆运动

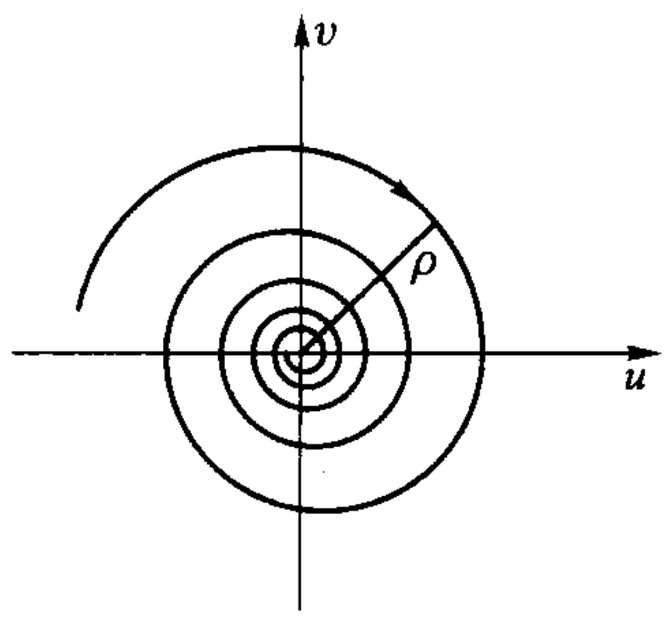


图 6.8 有阻尼无驱动的单摆运动的相图

能量耗散使相轨线矢径对数衰减，轨线不能再保持为封闭的曲线，如椭圆等。无论从哪点出发，经若干次旋转后趋向坐标原点，原点称为吸引子，它把相空间的点吸引过来。画出相应的势能曲线与相图如图 6.9。

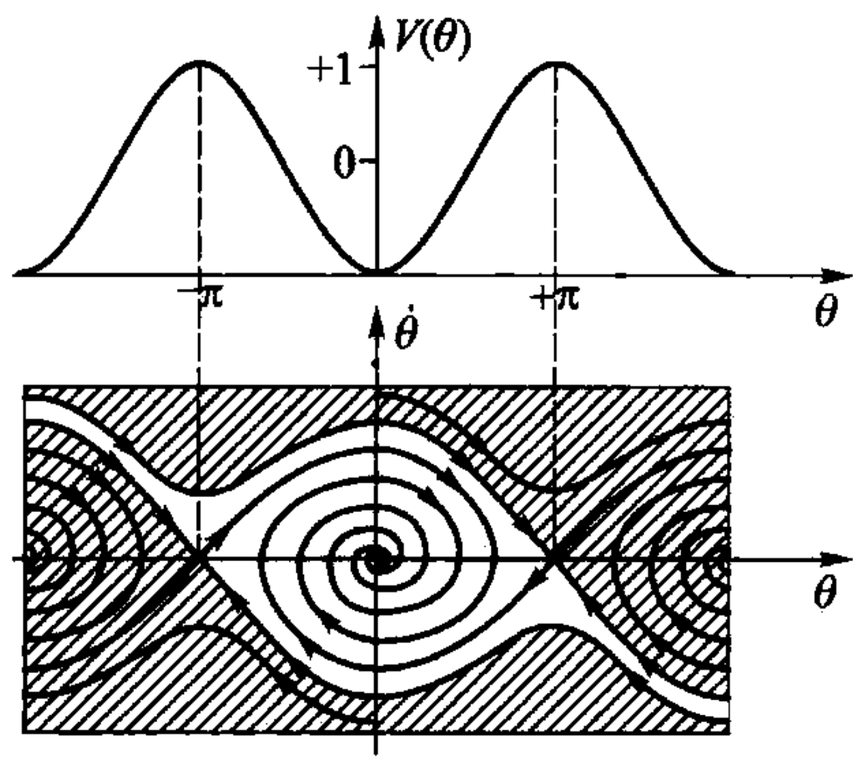


图 6.9 有阻尼无驱动的单摆运动的势能曲线与相图的对应关系

如果单摆从倒立开始运动，由于能量耗散回不到原有高度。因此，从点 G 出发的轨线到不了G' 点，分界线被破坏了。而在原来分界线以内的所有中间区域的相点都会流向坐标原点。

§6.1.2.3 不动点分类

通常把与外界有能量和物质交换的开放和远离平衡态的系统称之为耗散系统，反之则是保守系统。所以有阻尼的单摆是一个耗散系统。保守系统在相空间中的体积是守恒的，它只会随时间改变形状而不会改变大小。但耗散系统与此不同，如果跟随相空间中一个小体积中 ΔV 中所有点的演化轨迹时，ΔV 的体积会随着时间增大而缩小，最后这些点的轨迹会聚到一个低维的空间集即所谓的吸引子。初始条件不同，耗散系统可以演化到不同的吸引子。所有落到同一个吸引子上的出发点的集合称为吸引域。保守系统由于相体积守恒而不具有吸引子的吸引域。

吸引子的维数一般要比原始相空间低，这是由于耗散过程中，消耗了大量小尺度的运动模式，因而使得确定性系统长时间行为的有效自由度减少。如果系统最终剩下一个周期运动，则称该系统具有极限环吸引子。二维以上的吸引子，表现为相空间相应维数的环面。只有耗散系统中的混沌才会产生奇异吸引子。

如上所述，画出相轨线可以大致了解系统的运动状况，相图分析更重要的作用是可以分析解的稳定性以及稳定性发生变化的方式。例如一维的运动方程

$$\ddot{x} + \gamma\dot{x} + cx = 0$$

可写成一阶的常微分方程组

$$\begin{cases} \dot{x} = y \\ \dot{y} = -\gamma\dot{x} - cx \end{cases} \tag{6.5}$$

两式相除得

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{-\gamma\dot{x} - cx}{y}$$

这是相轨线方程，当 $\mathrm{d}y/\mathrm{d}x$ 为已知时，可用它画相轨线。当$\dot{x} = 0$，$\dot{y} = 0$时，相点的运动速度为零，这样的点叫不动点，也叫奇点，它就是系统的平衡点。下面研究不动点的分类及局域稳定性。

设 x_0，y_0是方程组的一个解（对不动点就是$x_0 = 0$，$y_0 = 0$），给它加上一个微扰δx，δy，再代入原方程组得

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}(x_0 + \delta x) = y + \delta y \\ \dfrac{\mathrm{d}}{\mathrm{d}t}(y_0 + \delta y) = -\gamma(y + \delta y) - c(x_0 + \delta x) \end{cases} \tag{6.6}$$

写成矩阵形式

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -c & -\gamma \end{pmatrix}\begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \tag{6.7}$$

按照雅可比矩阵J的定义，对于方程组

$$\begin{cases} \dot{x}_1 = f_1(x_1, x_2) \\ \dot{x}_2 = f_2(x_1, x_2) \end{cases}$$

有

$$J = \begin{pmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{pmatrix}$$

所以式 (6.7) 的系数矩阵就是方程组 (6.5) 的雅可比矩阵 J，

$$J = \begin{pmatrix} 0 & 1 \\ -c & -\gamma \end{pmatrix}$$

它的本征方程是

$$\begin{pmatrix} -\lambda & 1 \\ -c & -\gamma - \lambda \end{pmatrix} = 0$$

即

$$\lambda^2 + \gamma\lambda + c = 0$$

得本征值为

$$\lambda_{1,2}=\frac{-\gamma\pm\sqrt{\gamma^2-4c}}{2}$$

按照下式可以求出两个特征向量

$$\begin{pmatrix}0 & 1\\ -c & -\gamma\end{pmatrix}\begin{pmatrix}\xi_1\\ \xi_2\end{pmatrix}=\lambda_{1,2}\begin{pmatrix}\xi_1\\ \xi_2\end{pmatrix}$$

令

$$\Lambda=\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}$$

如果能找出两个线性无关的特征向量，可以由它们组成变换矩阵V，使得

$$J=V\Lambda V^{-1}$$

引入线性变换

$$\begin{pmatrix}\delta x\\ \delta y\end{pmatrix}=V\begin{pmatrix}u_1\\ u_2\end{pmatrix}\tag{6.8}$$

将它们代入方程组 (6.7) 得

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}u_1\\ u_2\end{pmatrix}=\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}\begin{pmatrix}u_1\\ u_2\end{pmatrix}$$

从中可解出u_1，u_2都是指数函数，写成$\exp(\lambda_1,t)$，$\exp(\lambda_2,t)$，再由式 (6.8) 知，方程组 (6.7) 的解的一般形式是u_1，u_2的线性叠加

$$\begin{pmatrix}\delta x\\ \delta y\end{pmatrix}=\begin{pmatrix}a_1\exp(\lambda_1,t)+a_2\exp(\lambda_2,t)\\ a_3\exp(\lambda_1,t)+a_4\exp(\lambda_2,t)\end{pmatrix}$$

指数函数的形式一般为 $\mathrm{e}^{\mu+\mathrm{i}\sigma}$，即指数有实部与虚部，可分以下几种情况:

如果 $\mu=0$，本征值是纯虚数，运动是振荡型的；

如果 $\sigma=0$，本征值是实数，运动将单调衰减或增大；

如果 $\mu\neq0$，$\sigma\neq0$则运动同时有增减和振荡两种运动。

$\mu=0$对应的不动点叫椭圆型不动点，其余的叫双曲型不动点。按照局域的性质，不动点分类如下：

当$\gamma^2-4c\geqslant0$ 时，λ_1，λ_2是一对实数，此时不动点分为：

(1) $\lambda_1<0$，$\lambda_2<0$，解是随t单调衰减的，轨线将流向不动点，如图 6.10(a) 所示，在不动点周围所有方向都是局域稳定的，不动点为稳定结点；

(2) $\lambda_1>0$，$\lambda_2>0$，解是随t单调增大的，轨线从不动点流出，如图 6.10(b) 所示，所以在不动点周围所有方向都是局域不稳定的，不动点为不稳定结点；

(3) $\lambda_1 < 0$，$\lambda_2 > 0$或$\lambda_1 > 0$，$\lambda_2 < 0$，不动点附近有一个不稳定方向，如图 6.10(c) 所示，不动点为鞍点。

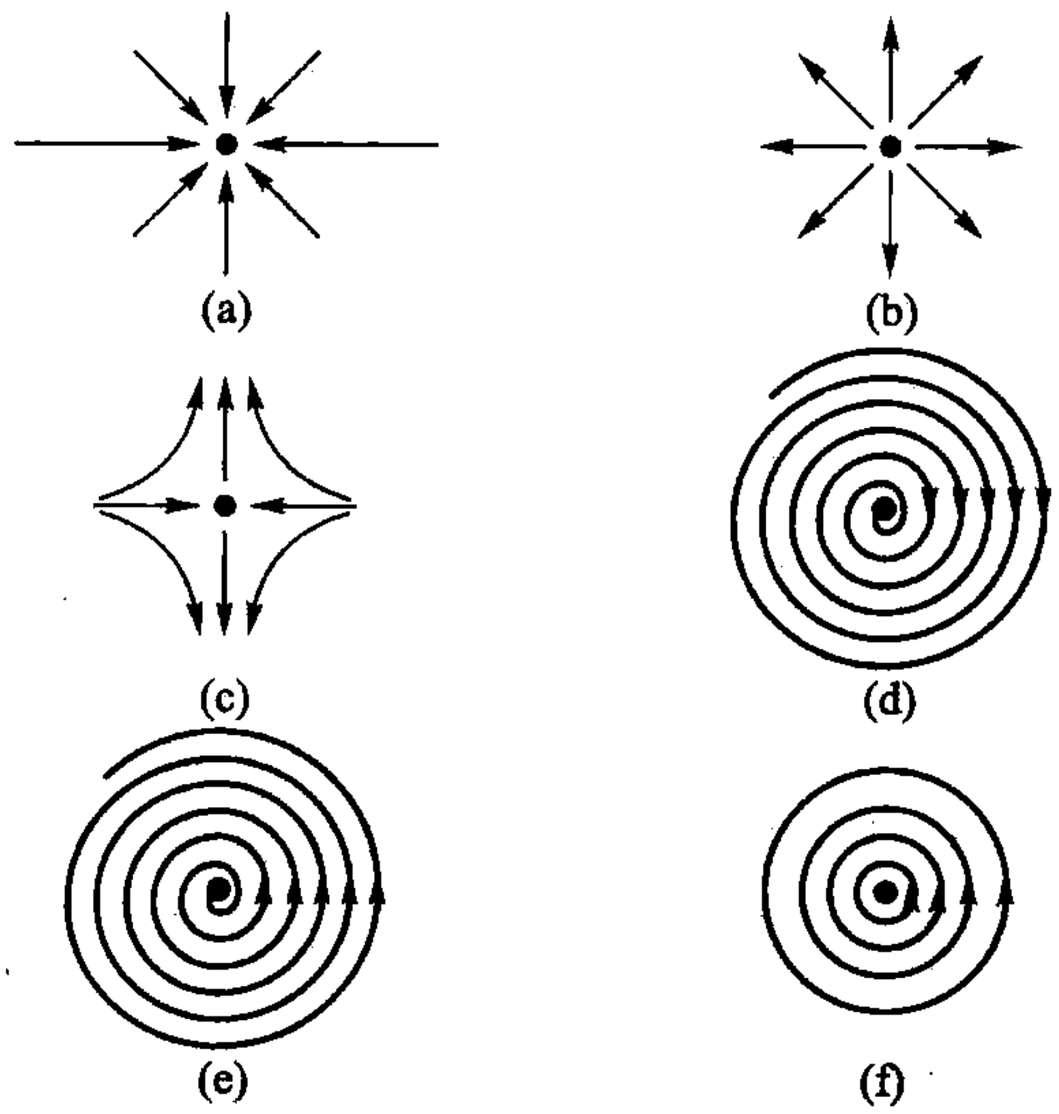


图 6.10 不动点的分类

当 $\gamma^2 - 4c < 0$ 时，λ_1，λ_2 是一对共轭复数，特征值具有虚部意味着解有三角函数的部分，这使得轨线发生旋转，而特征值的实部γ则会影响轨线运动的方向。不动点是根据实部分类：

(1) $\gamma < 0$时不动点为稳定焦点，轨线在旋转的同时也向不动点收缩，如图 6.10(d) 所示；

(2) $\gamma > 0$时不动点为不稳定焦点，轨线在旋转的同时还会从不动点向外逃逸，如图 6.10(e) 所示；

(3) $\gamma = 0$ 时不动点为中心点，轨线围绕不动点形成闭合曲线，如图 6.10(f) 所示。

在 γ-c 平面上画出这些不动点位置如图 6.11。

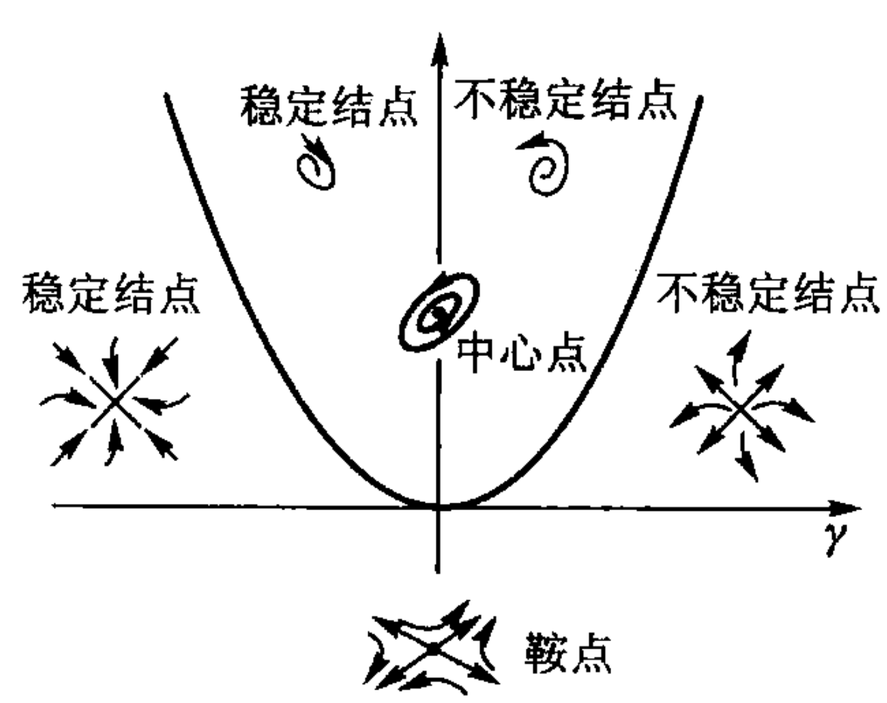


图 6.11 参数 γ-c 平面内不动点的位置

例如，对于无阻尼无驱动的单摆，不动点有 3 个：$\theta = 0, -\pi, +\pi$；$\dot{\theta} = 0, 0, 0$，雅可比矩阵是

$$J = \begin{pmatrix} 0 & 1 \\ -\cos\theta & 0 \end{pmatrix}$$

它的本征方程和本征值是

$$\lambda^2 + \cos\theta = 0, \qquad \lambda_{1,2} = \pm\sqrt{-\cos\theta}$$

可见，$\theta = 0, \dot{\theta} = 0$是中心点，也是椭圆型不动点即相图中的$O$点。$\theta = -\pi, +\pi$；$\dot{\theta} = 0, 0$，都是鞍点，即相图中的 G，G'点，也是双曲型不动点。

顺便介绍一下同宿轨道和异宿轨道。鞍点的轨线可以流出也可以流出，从一个鞍点到另一个鞍点的轨线称为异宿轨道,如无阻尼无驱动时相图中从G到G'的轨线。它也是分界线（也叫分型线），是两种类型轨线分界。从一个鞍点到自己本身的轨线就叫同宿轨道。如将相图绕在圆筒上，G和G' 合为一点，原来的异宿轨道就成了同宿轨道。

更一般的定义是，同宿轨道(homoclinic orbit)是动力系统中当$t \to \pm\infty$ 时趋向同一状态的一种轨道。一般把$t \to -\infty$和$t \to +\infty$时的极限状态分别称为该轨道的α极限集和ω极限集。如果$t \to -\infty$ 和$t \to +\infty$分别趋向各自不同的状态的轨道则叫异宿轨道(heteroclinic orbit)。

对于有阻尼无驱动的单摆，不动点也有3个：$\theta = 0, -\pi, +\pi$；$\dot{\theta} = 0, 0, 0$，雅可比矩阵是

$$J = \begin{pmatrix} 0 & 1 \\ -\cos\theta & -2\beta \end{pmatrix}$$

它的本征方程和本征值是

$$\lambda^2 + 2\beta\lambda + \cos\theta = 0, \qquad \lambda_{1,2} = -\beta \pm \sqrt{\beta^2 - \cos\theta}$$

对于空气阻力有$\beta > 0$，若$\beta < 1$，$\theta = 0$，$\dot{\theta} = 0$是稳定焦点，若$\beta \geqslant 1$，$\theta = 0$，$\dot{\theta} = 0$是稳定结点，这时单摆是单调衰减的，没有震荡。而$\theta = -\pi, +\pi$；$\dot{\theta} = 0, 0$还是鞍点。

下面再讨论三维的情形。三维的自治系统(指方程中不显含t)一般形式为

$$\begin{cases} \dot{x} = f_1(x, y, z) \\ \dot{y} = f_2(x, y, z) \\ \dot{z} = f_3(x, y, z) \end{cases}$$

平衡点是

$$f_1(x_0, y_0, z_0) = f_2(x_0, y_0, z_0) = f_3(x_0, y_0, z_0) = 0$$

若方程组的雅可比矩阵的三个本征值是λ_1，λ_2，λ_3，可仿照上面对二维自治系统的分析对它进行分类如下：

吸引子	所有本征值的实部小于零	指标为0
鞍点	有1个本征值的实部大于零	指标为1
鞍点	有2个本征值的实部大于零	指标为2
排斥子	有3个本征值的实部大于零	指标为3

在三维相空间中的平衡点的情况如图 6.12 所示。图中的 I，R 分别为本征值的虚部与实部，第一行到第四行依次是指标为 0 的吸引子，为1的鞍点，为2的鞍点，为3的排斥子。在后面讨论洛伦茨方程时会用到这里对三维情形的平衡点的分类。

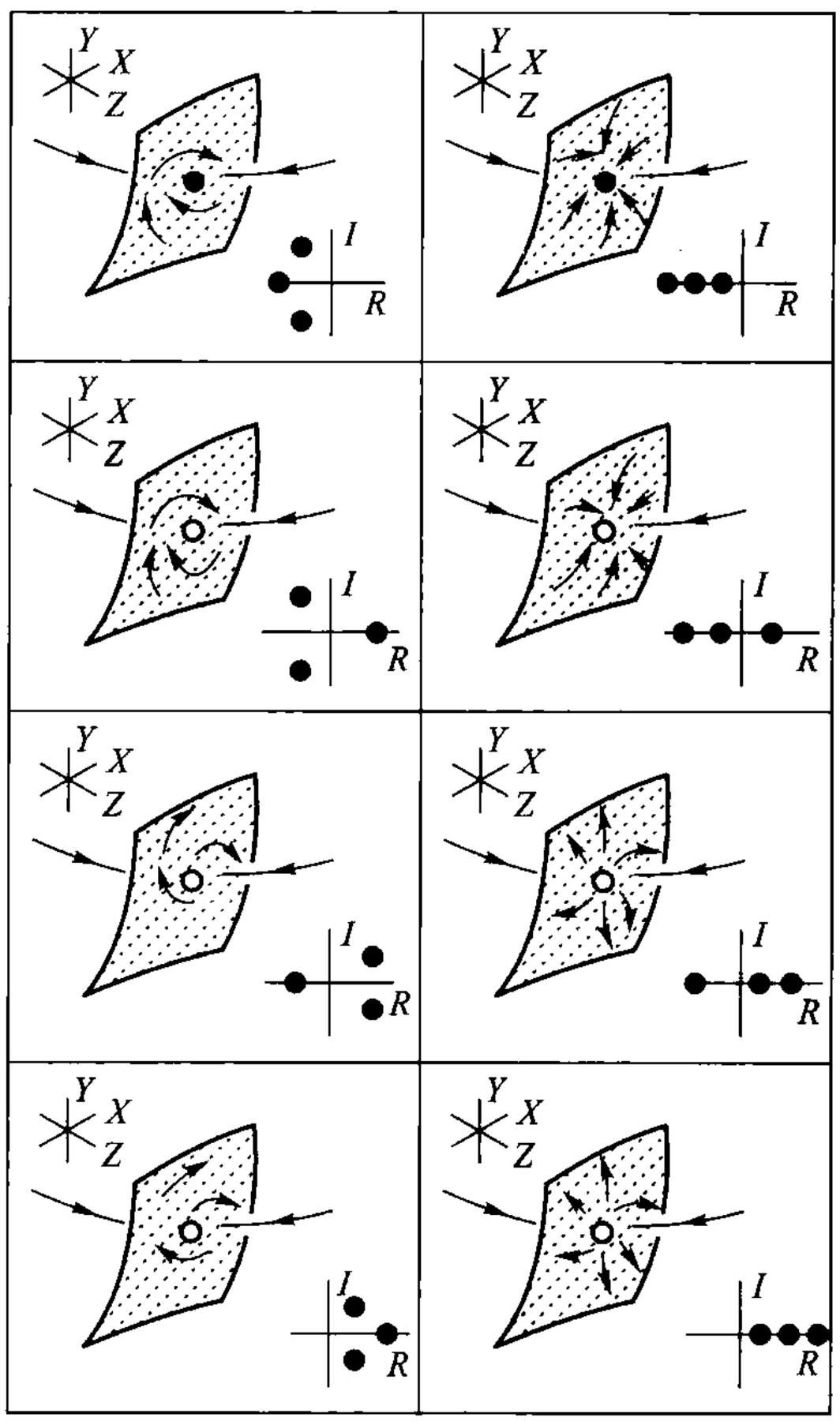


图 6.12 三维相空间中平衡点的分类

§6.1.3 有阻尼有驱动的情况——耗散系的混沌

此时有$\beta \neq 0$，$f \neq 0$(下面的计算取$\beta = 1/4$，$\omega = 2/3$)，方程是

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+2\beta\frac{\mathrm{d}\theta}{\mathrm{d}t}+\sin\theta=f\cos\omega t$$

将它改写成一阶常微分方程组，由于方程中显含t，所以称为非自治方程。为了得到自治的方程组（方程组中不显含t），引入$\varphi=\omega t$，得

$$\begin{cases}\dfrac{\mathrm{d}\theta}{\mathrm{d}t}=p\\ \dfrac{\mathrm{d}p}{\mathrm{d}t}=-2\beta\dfrac{\mathrm{d}\theta}{\mathrm{d}t}-\sin\theta+f\cos\varphi\\ \dfrac{\mathrm{d}\varphi}{\mathrm{d}t}=\omega\end{cases}$$

三个变量θ，p，φ 组成 3 维相空间，给定初始条件就可以通过下面的程序画出来，如图 6.13 所示。

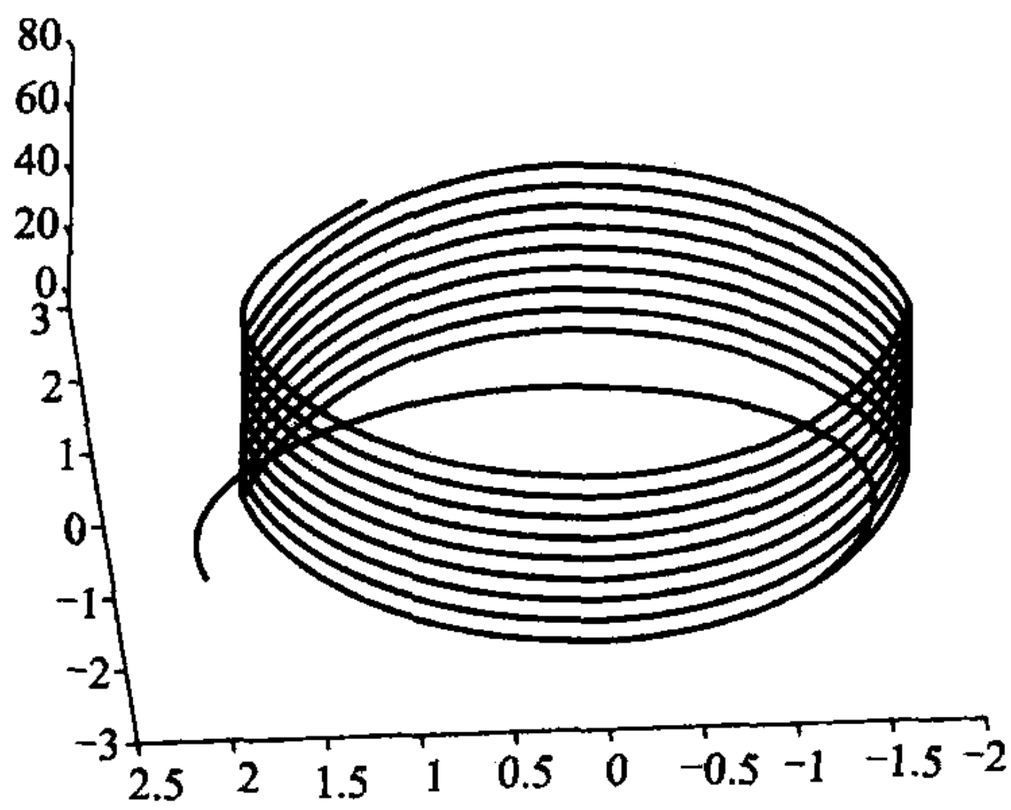


图 6.13 三维相图

```
function dbyd                                          %画三维相图
global a f u
u=2/3; a=0.25; ZQ=3*pi; f=0.8;
[T, Y]=ode45(@dby,[0:ZQ/100:10*ZQ],[-0.8,2,u]);
plot3(Y(:,1),Y(:,2),Y(:,3))
view(-95,60)
function ydot=dby(t,y)
global a f u
ydot=[y(2);
  -sin(y(1))-2*a*y(2)+f*cos(y(3));
  u];
```

相角 φ 有周期性，把 $2n\pi$ 和 $2(n+1)\pi$ 平面连接起来，相空间形成一个轮胎面。原来的圆形轨线成了在轮胎面上的环绕线，如图 6.14。

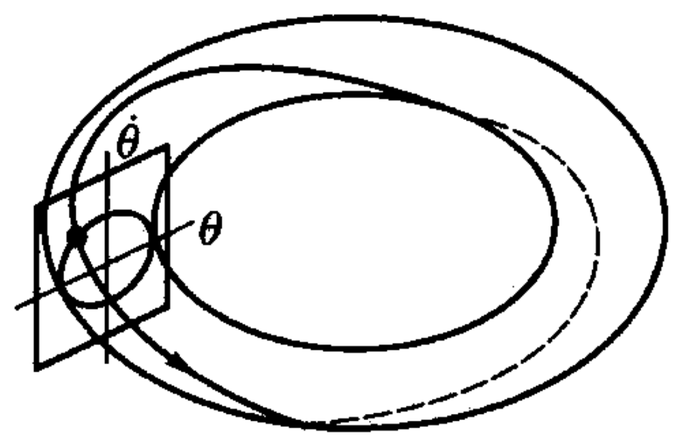


图 6.14 轮胎面上的相轨线

在轮胎面上作截面（庞加莱截面），轨线穿过它会留下一个点，它对应φ取某个常位相值时，在θ，$\dot{\theta}$平面上的相点。庞加莱截面对于观察多维相空间的轨线很有帮助。下面利用它研究有阻尼有驱动的单摆的运动。

§6.1.3.1 极限环吸引子

当驱动力较小时，如$f=0.8$，从不同的初始条件出发,经过长时间运动最终都落到同一个椭圆上，称之为极限环。

图 6.15(a) 的初始条件是 $\theta=-2$，$\mathrm{d}\theta/\mathrm{d}t=2$，轨线由外向内旋转到极限环。

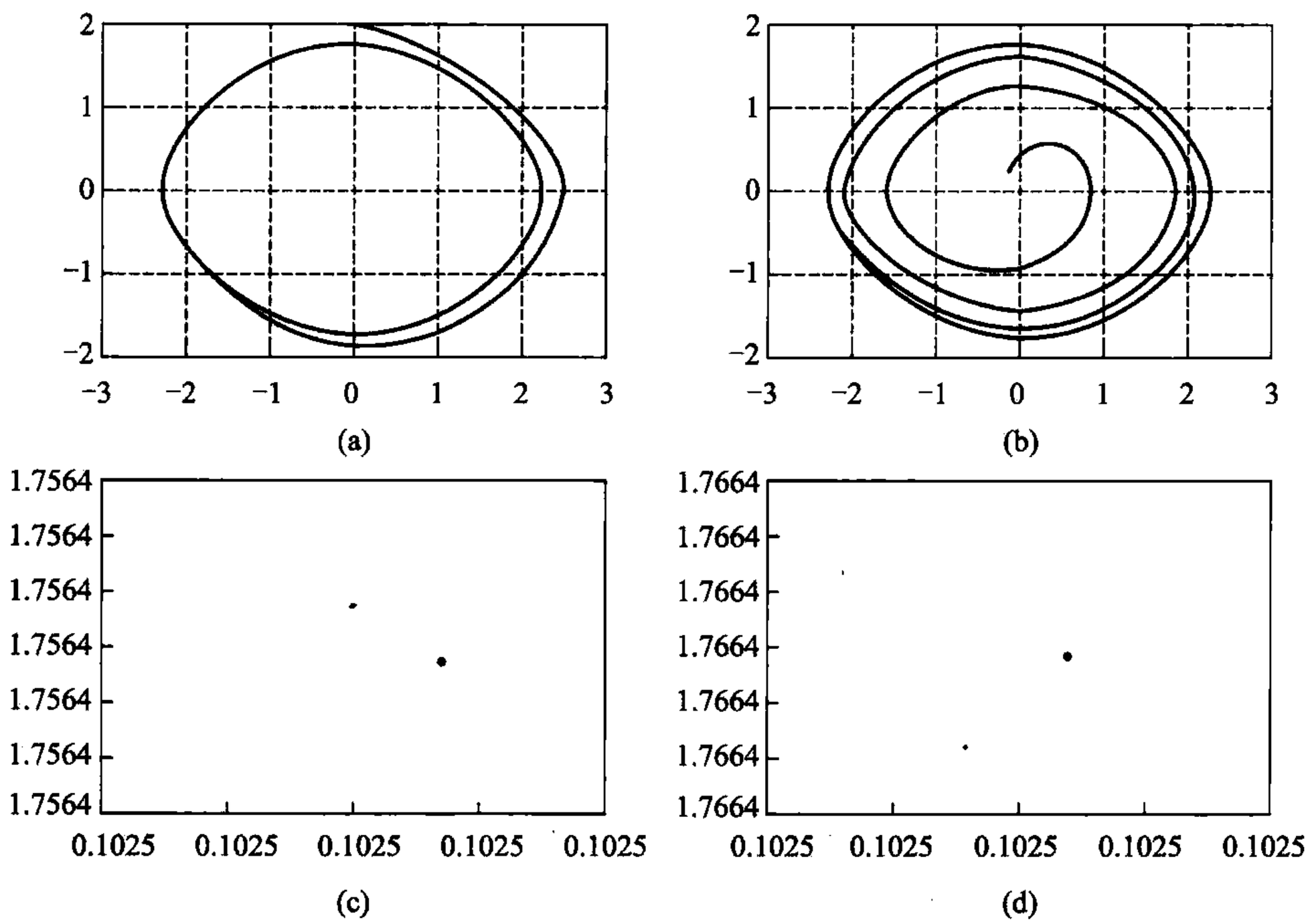


图 6.15 极限环吸引子

图 6.15(b) 的初始条件是$\theta = -0.1$，$\mathrm{d}\theta/\mathrm{d}t = 0.2$，轨线由内向外旋转到极限环。

图 6.15(c) (d) 是对应的庞加莱截面图。

§6.1.3.2 对称性破缺

当 $f = 1.03$，从不同的初条件出发，得到不同的蛋形的吸引子，它们左右反射对称，原来的左右对称性被破坏如图 6.16 所示。

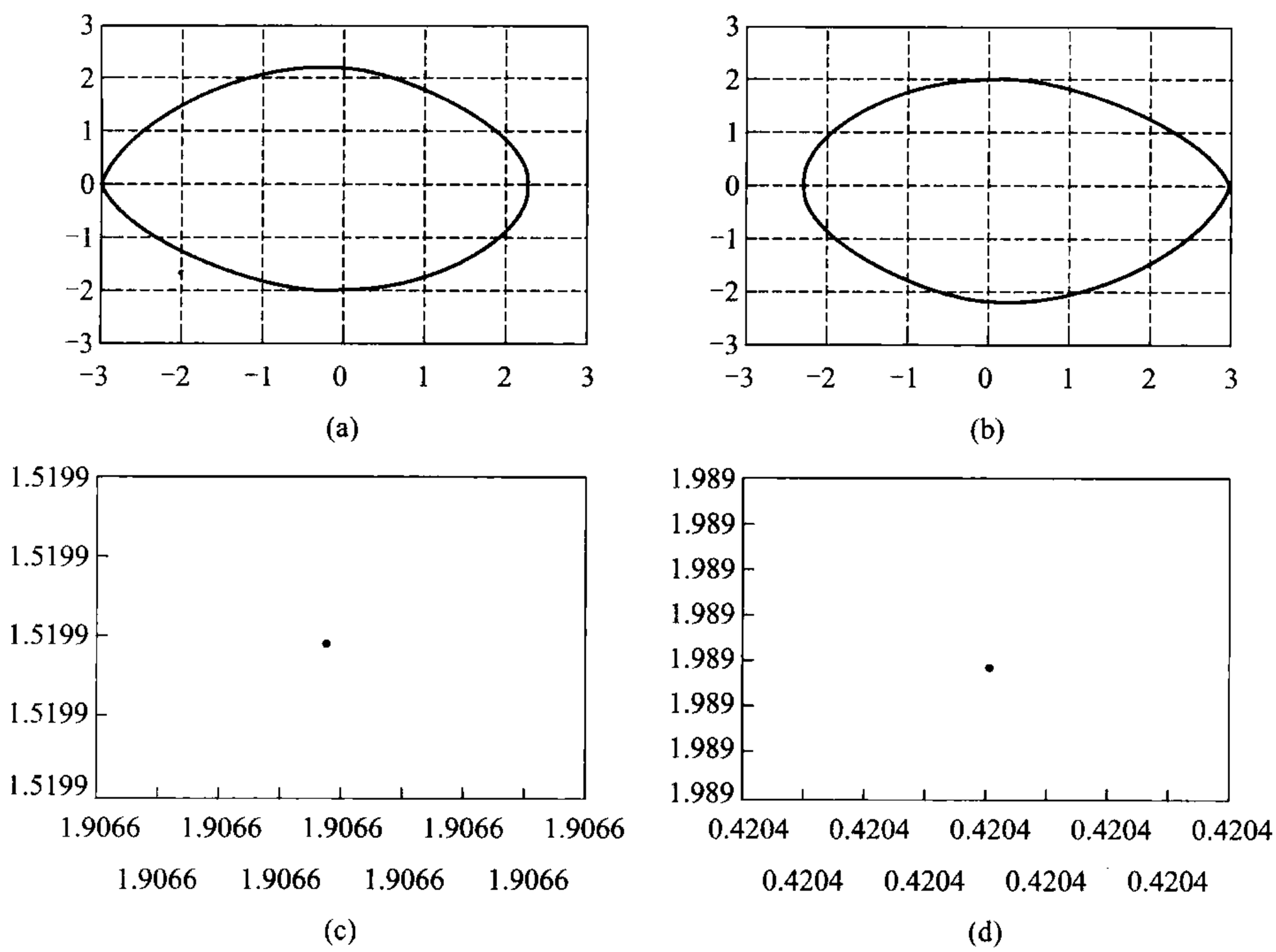


图 6.16 对称性破缺

图 6.16(a) 的初始条件是 $\theta = -0.1$，$\mathrm{d}\theta/\mathrm{d}t = 2$，在庞加莱截面 [图 6.16(c)] 仍然是一个点。

图 6.16(b) 的初始条件是 $\theta = -0.8$，$\mathrm{d}\theta/\mathrm{d}t = 2$，在庞加莱截面 [图 6.16(d)] 仍然是一个点。

§6.1.3.3 倍周期分岔与混沌

继续增加驱动力f，单摆的周期运动 (图 6.17) 发生了分岔，出现了周期 2 (图 6.18)，周期 4 (图 6.19) 的运动，最后趋于混沌 (图 6.20)。计算中取 $\theta = -0.8$，$\mathrm{d}\theta/\mathrm{d}t = 2$。周期 1 取 $f = 1.0584$，周期 2 分岔的 $f = 1.0726$，周期 4 分岔的 $f = 1.08829$，出现混沌取 $f = 1.095$。图 6.21 是相位对于f画图，图 6.22 是速度对于f画图。这两个图形都出现了通过倍周期分岔走向混沌的现象。

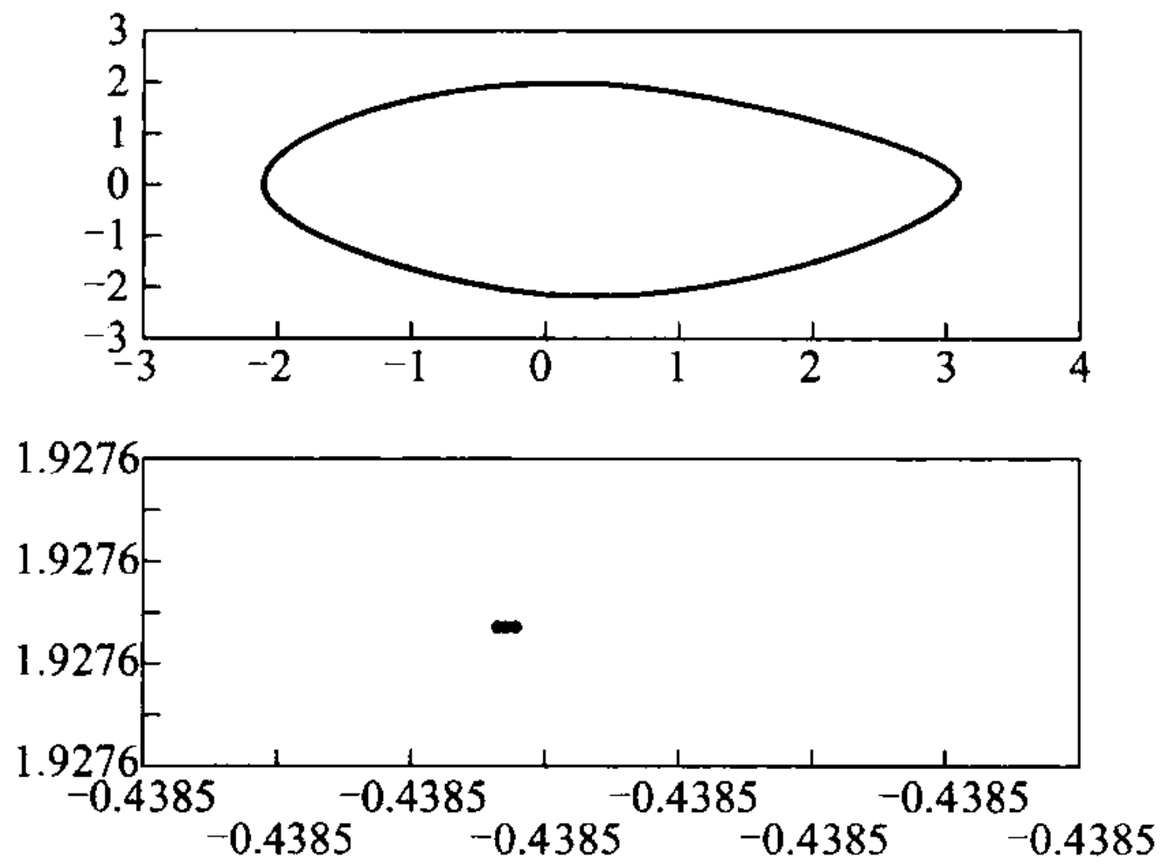


图 6.17 周期 1 的运动

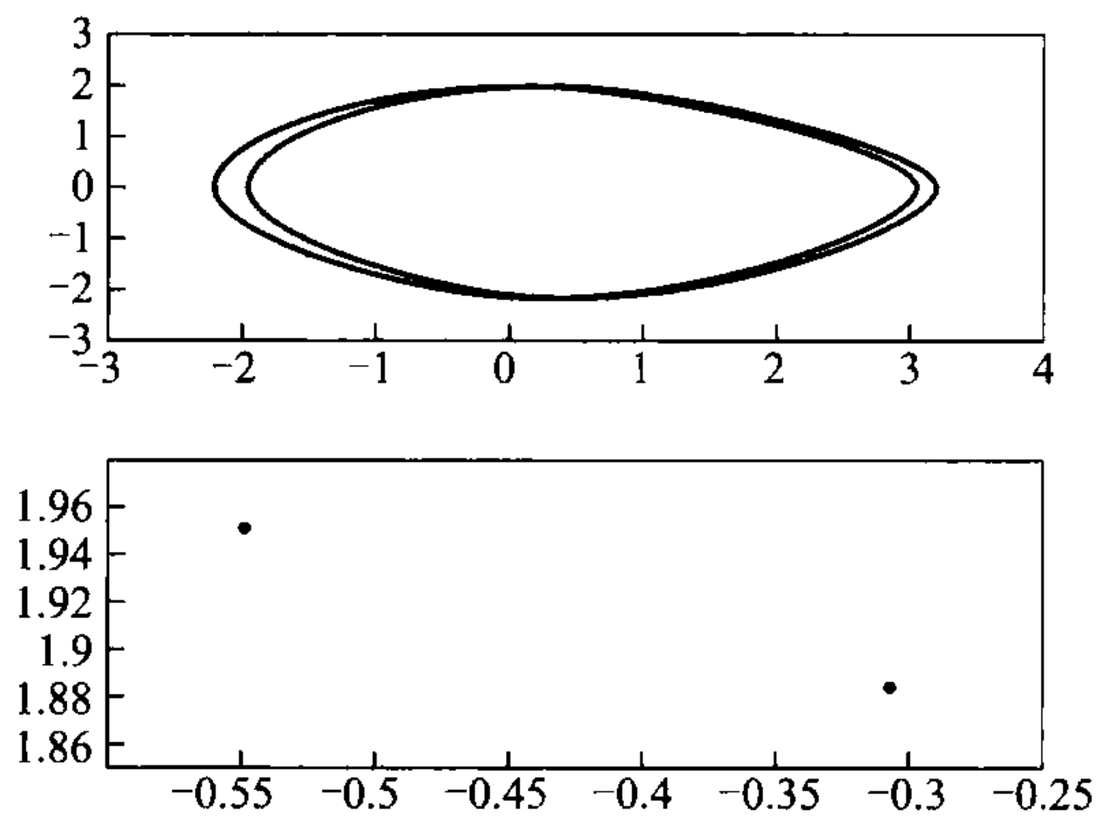


图 6.18 周期 2 的运动

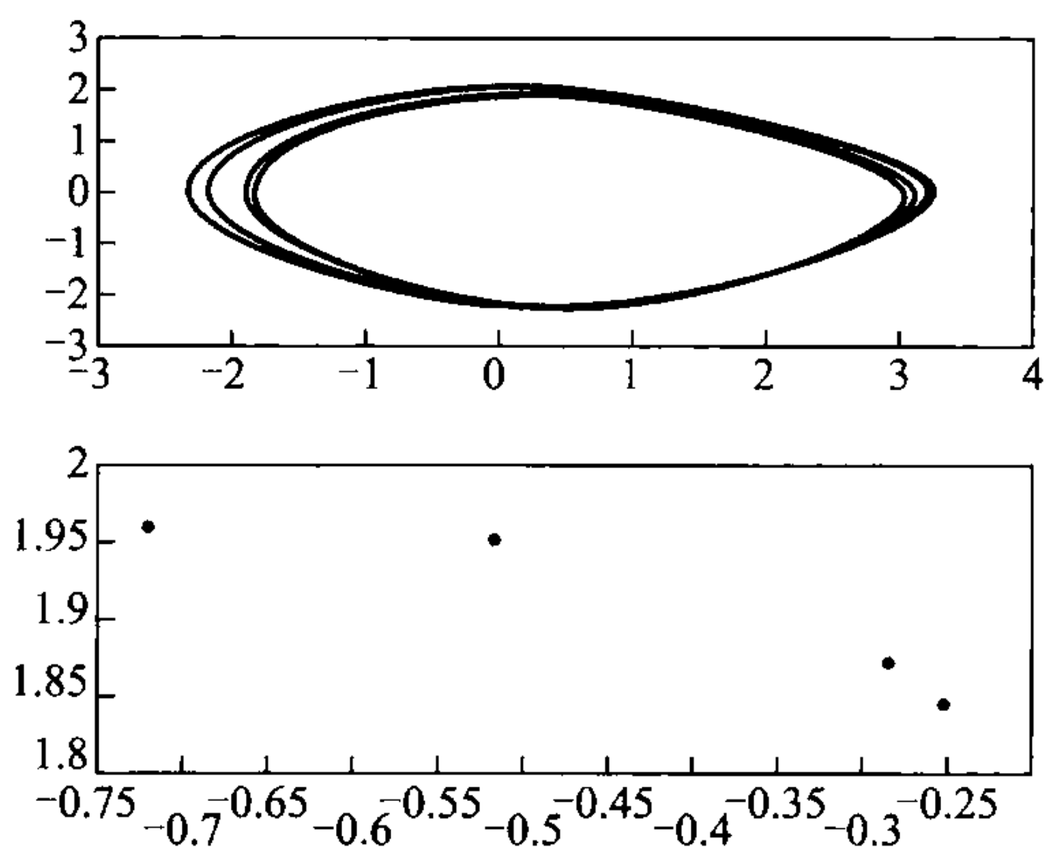


图 6.19 周期 4 的运动

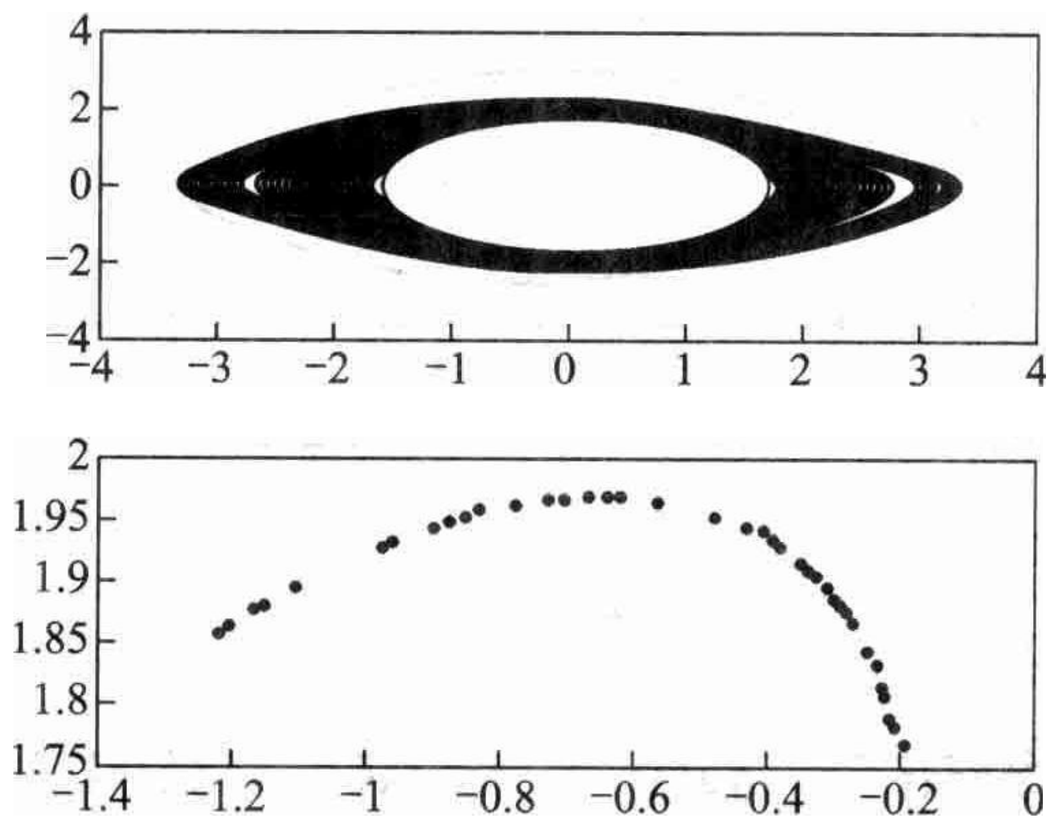


图 6.20 混沌运动

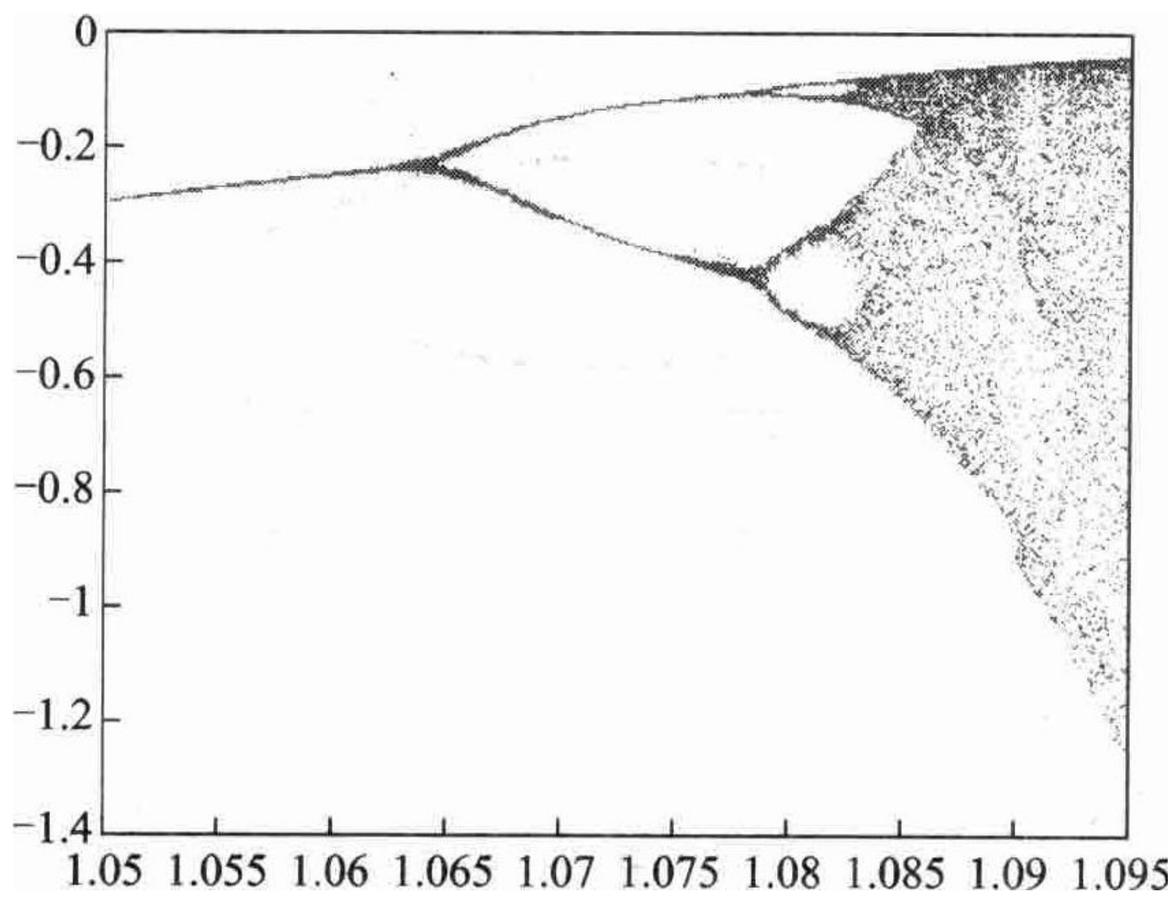


图 6.21 相位与 f 的关系

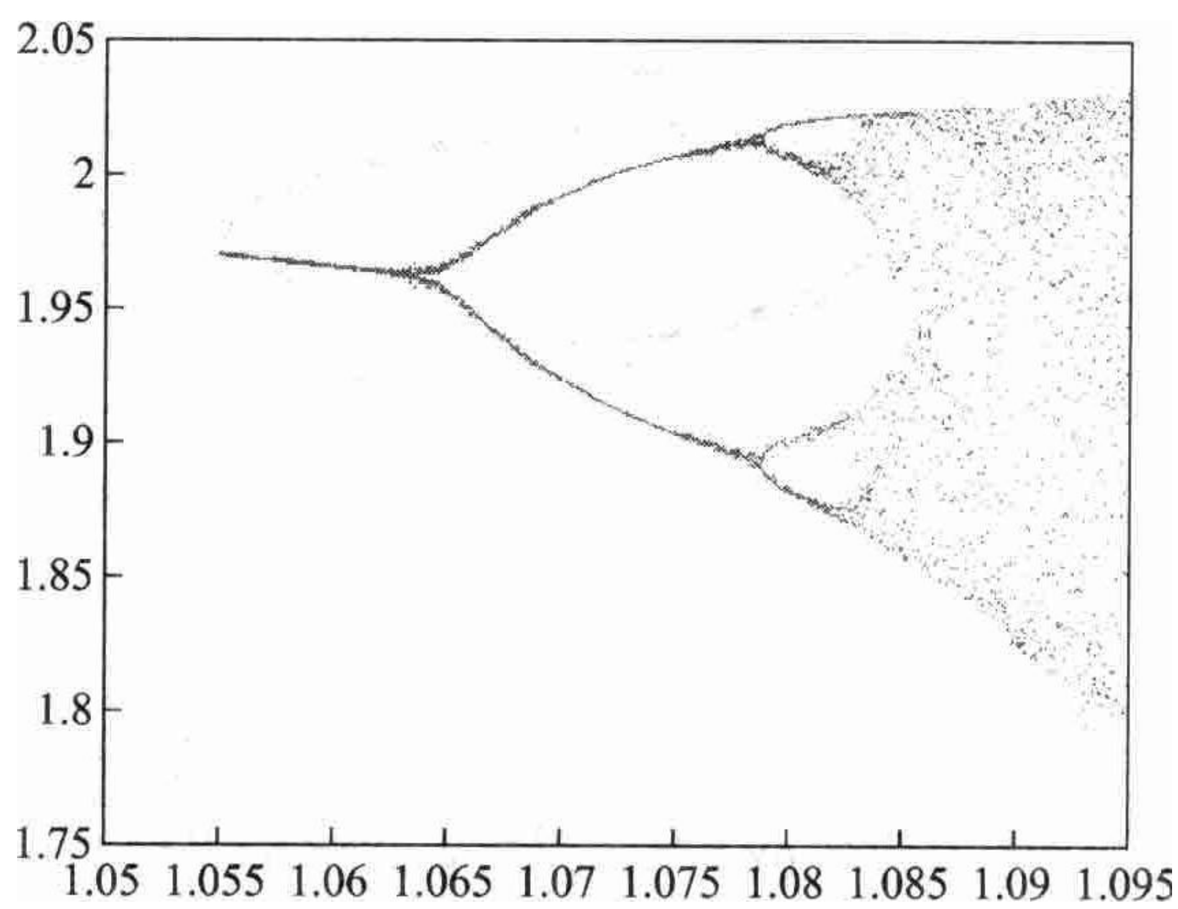


图 6.22 速度与 f 的关系

下面是相应的作图程序。

```
function dbyd
global a f u
u=2/3; a=0.5; ZQ=3*pi; f=1.0584;
[T, Y]=ode45(@dby,[0:ZQ/100:30*ZQ],[-0.8,2]);

subplot(2,1,1)
plot(Y(2500:end,1),Y(2500:end,2))
subplot(2,1,2)
xx=[];  yy=[];
for j=2500:100:3000
xx=[xx,Y(j,1)]; yy=[yy,Y(j,2)];
end
plot(xx,yy,'.r')

function ydot=dby(t,y)
global a f u
ydot=[y(2);
-sin(y(1))- a*y(2) + f*cos(u*t) ];
```

下面的程序画出周期分岔的相图。

```
function dbyd4
global a f u
u=2/3;  a=0.5;  ZQ=3*pi; ff=1.05:0.00005:1.095;
FF=[];  YY=[];
for k=1:length(ff)
     f=ff(k);
     [T, Y]=ode23(@dby,[0:ZQ/100:30*ZQ],[-0.8,2]);
     FF=[FF,f];   YY=[YY;Y(2501:100:end,2)'];
 end
plot(FF,YY,'r.','markersize',1)

function ydot=dby(t,y)
global a f u
ydot=[y(2);
-sin(y(1))- a*y(2) + f*cos(u*t) ];
```

§6.2 倒摆与达芬方程

§6.2.1 倒摆的运动方程

倒摆强迫振动的实验装置如图 6.23 所示。倒摆是一个倒立的摆，其摆锤质量为m，轻质杆长为 l，倒摆的底座以微小的幅度绕其中心作简谐运动 $\varphi = A\cos\Omega t$ (A,Ω为已知常数)，φ角表示底座的垂线对铅垂线的偏离，θ表示杆对底座的垂线的偏离。假设$\varphi \ll \theta$，所以θ可近似表示杆对铅垂线的偏离；弹簧产生力矩为$-c\theta$ (c为常量)；空气阻力为$-\beta l\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}+\frac{\mathrm{d}\varphi}{\mathrm{d}t}\right)\approx -\beta l\frac{\mathrm{d}\theta}{\mathrm{d}t}$ (β为阻尼系数)。改变运动阻尼，可以演示运动状态从周期解通过周期分岔到混沌的变化，同时进一步熟悉同宿轨道、周期解和混沌解等概念。

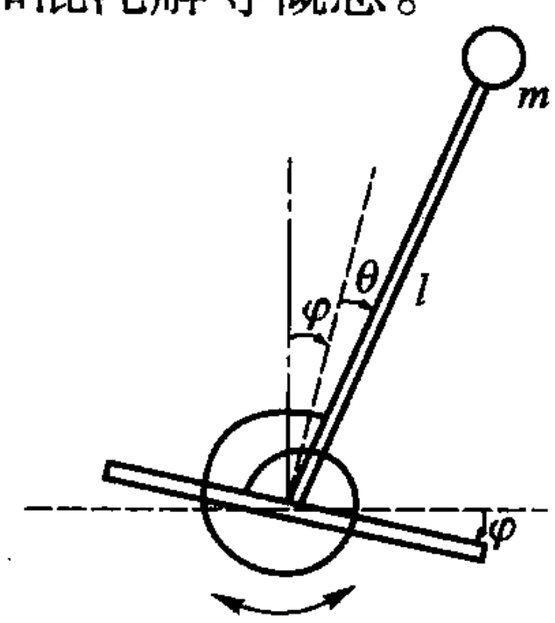


图 6.23 倒摆的运动模型

先导出倒摆强迫振动的运动微分方程：重力产生力矩为

$$mgl\sin(\theta+\varphi)\approx mgl\left(\theta-\frac{\theta^3}{6}\right) \tag{6.9}$$

根据质点的角动量定理，倒摆的运动微分方程为

$$ml^2\left(\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2}\right)=-c\theta+mgl\sin(\theta+\varphi)-\beta l^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}+\frac{\mathrm{d}\varphi}{\mathrm{d}t}\right) \tag{6.10}$$

由于$\frac{\mathrm{d}\varphi}{\mathrm{d}t}\ll\frac{\mathrm{d}\theta}{\mathrm{d}t}$, $\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2}\ll\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}$，所以上式可合理近似为

$$ml^2\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+\beta l^2\frac{\mathrm{d}\theta}{\mathrm{d}t}+(c-mgl)\theta+\frac{1}{6}mgl\theta^3=ml^2\Omega^2A\cos\Omega t \tag{6.11}$$

我们仅在 $c<mgl$ 条件下进行研究。

为了对方程进行无量纲化，将方程 (6.11) 进行简化，并用大写的 T 表示时间，于是得到

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}T^2}+\frac{\beta}{m}\frac{\mathrm{d}\theta}{\mathrm{d}T}-\frac{mgl-c}{ml^2}\theta+\frac{g}{6l}\theta^3=A\Omega^2\cos\Omega T \tag{6.12}$$

设

$$\Omega_0^2 = \frac{mgl - c}{ml^2}$$

Ω_0 具有角频率的量纲。它的倒数给出问题中的一个时间尺度T_0。其次，在无驱动力时系统具有3个平衡位置：$\theta = 0$为不稳定的平衡位置；还有两个稳定平衡位置

$$\theta_0 = \pm\sqrt{6 - \frac{6c}{mgl}}$$

θ_0 的数值给出问题中角度的一个尺度（在一个问题中时间的参考尺度，空间的参考尺度往往不止一个）。现取θ_0作为角度的量度单位，取$T_0(= 1/\Omega_0)$作为时间的量度单位，则无量纲的角度变量、无量纲的时间变量分别为

$$x = \frac{\theta}{\theta_0}, \quad t = \frac{T}{T_0}$$

对 (6.12) 式进行变量变换，得

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + \frac{\beta}{m\Omega_0}\frac{\mathrm{d}x}{\mathrm{d}t} - x + x^3 = \frac{A}{\theta_0}\left(\frac{\Omega}{\Omega_0}\right)^2 \cos\frac{\Omega}{\Omega_0}t$$

现分别引入无量纲的阻尼系数、无量纲的角频率和无量纲的驱动力的振幅

$$\delta = \frac{\beta}{m\Omega_0}, \quad \omega = \frac{\Omega}{\Omega_0}, \quad f = \frac{A}{\theta_0}\left(\frac{\Omega}{\Omega_0}\right)^2$$

于是可得无量纲方程

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + \delta\frac{\mathrm{d}x}{\mathrm{d}t} - x + x^3 = f\cos\omega t \tag{6.13}$$

这就是著名的强迫达芬 (Duffing) 振动方程。

设$x = y_1$，$\mathrm{d}x/\mathrm{d}t = y_2$，则方程(6.13)化为两个一阶方程

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = -\delta y_2 + y_1 - y_1^3 + f\cos\omega t \end{cases}$$

进行数值计算时可选取参数$\delta = 0.26$，$f = 2$，$\omega = 2$。

§6.2.2 倒摆的混沌运动

下面用波形图、相图、傅里叶频谱图和庞加莱截面图研究系统的运动，可以看出，不同的方法得出的结论是统一的，相互印证的。而动画模拟则能直观的给出一个图像。

§6.2.2.1 无阻尼无驱动情形

这时方程成为 $\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}-x+x^3=0$，积分得

$$\frac{1}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2+\frac{1}{2}\left(\frac{1}{2}x^4-x^2\right)=E$$

所以势能是 (图 6.24)

$$V=\frac{1}{2}\left(\frac{1}{2}x^4-x^2\right)$$

平衡点有：$x=0$ (不稳定平衡点)，$x=\pm1$ (稳定平衡点)。
奇点有：鞍点(0,0)；中心型奇点(1,0)和(−1,0)。

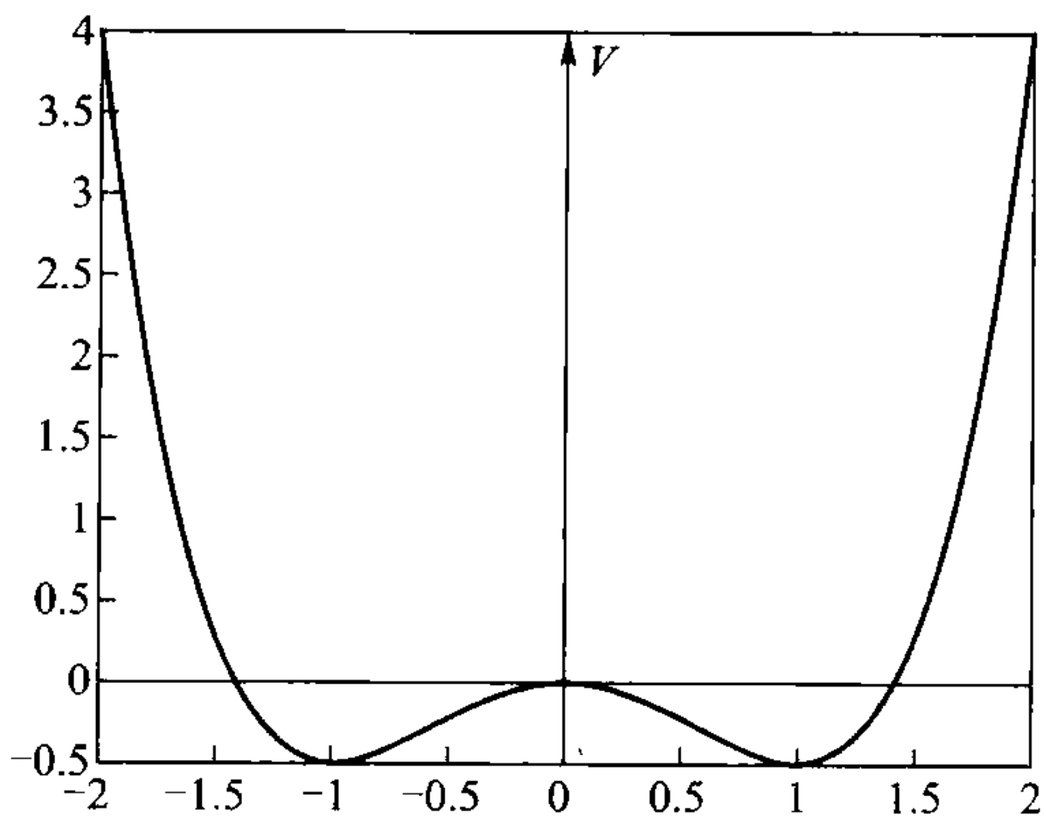


图 6.24 无阻尼无驱动时倒摆的势能曲线

在图 6.25 中，根据能量分成三种运动。

(1) $-0.25<E<0$，绕$x=\pm1$两个闭轨道，代表在稳定平衡位置附近的周期振动。(图中$E=-0.1$，-0.2)。

(2) $E=0$，对应∞型轨道，是同宿轨道，同宿点(0,0)。

(3) $E>0$，绕$x=0$，±1三个平衡位置的闭轨道，也是周期运动。图中$E=0.2$。

画图指令与画出图形如下 (图 6.25)：

```
>>ezplot('y^2+x^4/2-x^2',[-1.6,1.6])
>>hold on
>>ezplot('y^2+x^4/2-x^2+0.2')
>>ezplot('y^2+x^4/2-x^2+0.4')
>>ezplot('y^2+x^4/2-x^2-0.4')
```

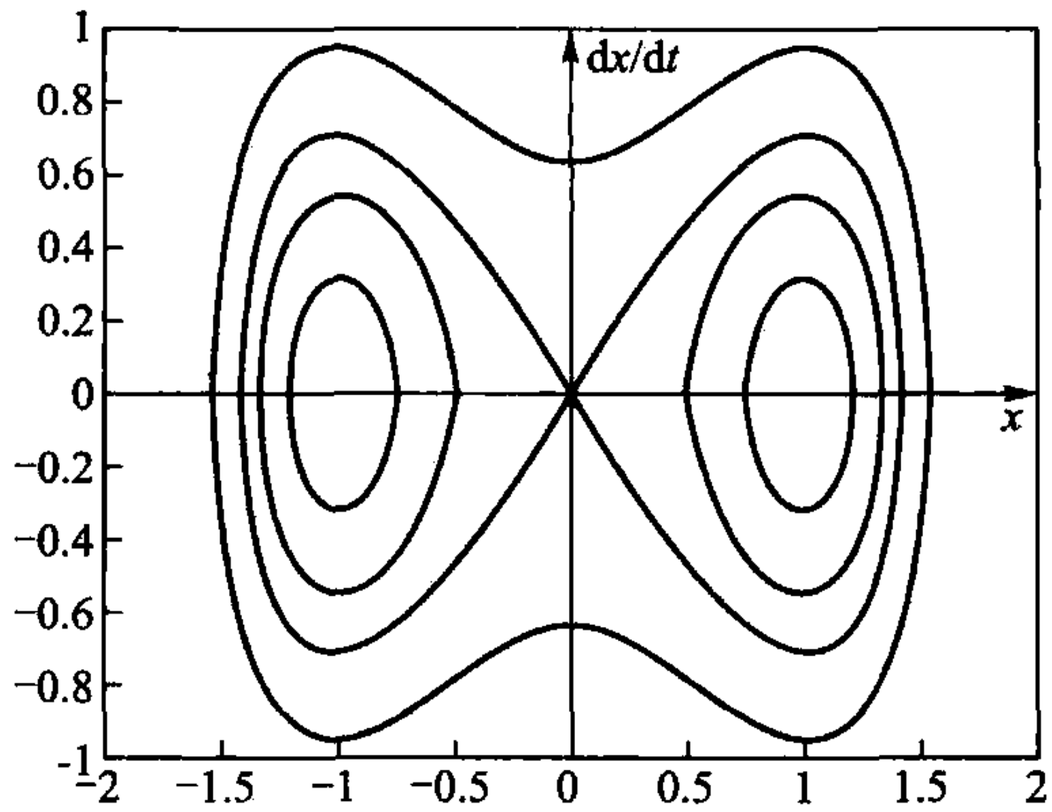


图 6.25 无阻尼无驱动时倒摆运动的相图

§6.2.2.2 有阻尼无驱动情形

这时的方程成为

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}+k\frac{\mathrm{d}x}{\mathrm{d}t}-x+x^3=0$$

奇点：原点是鞍点，坐标 $x=\pm1$ 处是两不动点，是吸引子。

整个相平面被分隔成两个区域，不同区的相点分别流向这两个不动点(图 6.26)。

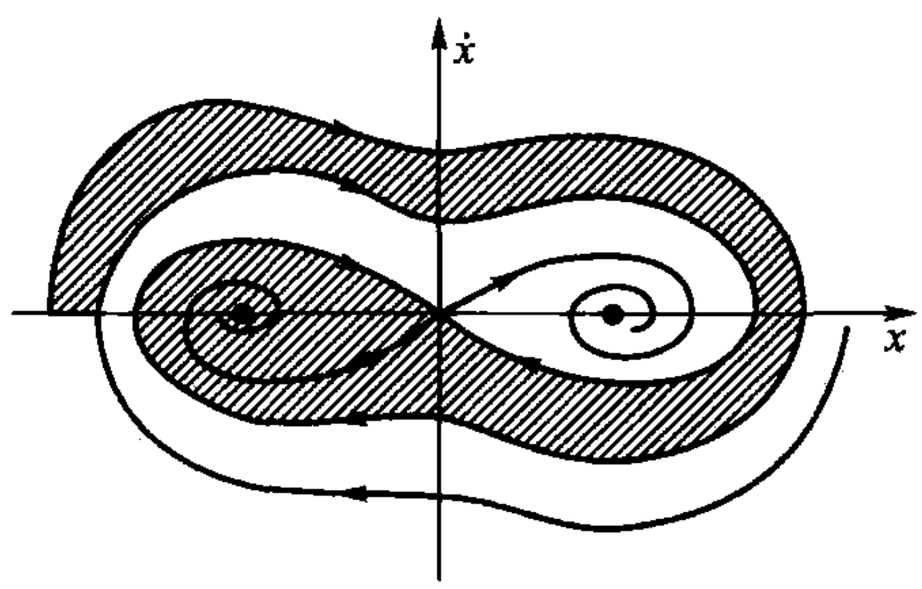


图 6.26 有阻尼无驱动时的相图

§6.2.2.3 有阻尼有驱动情形

这时的方程成为

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}+k\frac{\mathrm{d}x}{\mathrm{d}t}-x+x^3=f\cos\omega t$$

阻尼消耗能量，外部驱动补充能量，系统的运动状态解有周期解或混沌解。为了掌握运动的整体情况，先画系统的终态解随阻尼系数 k 变化的分岔图如图 6.27。

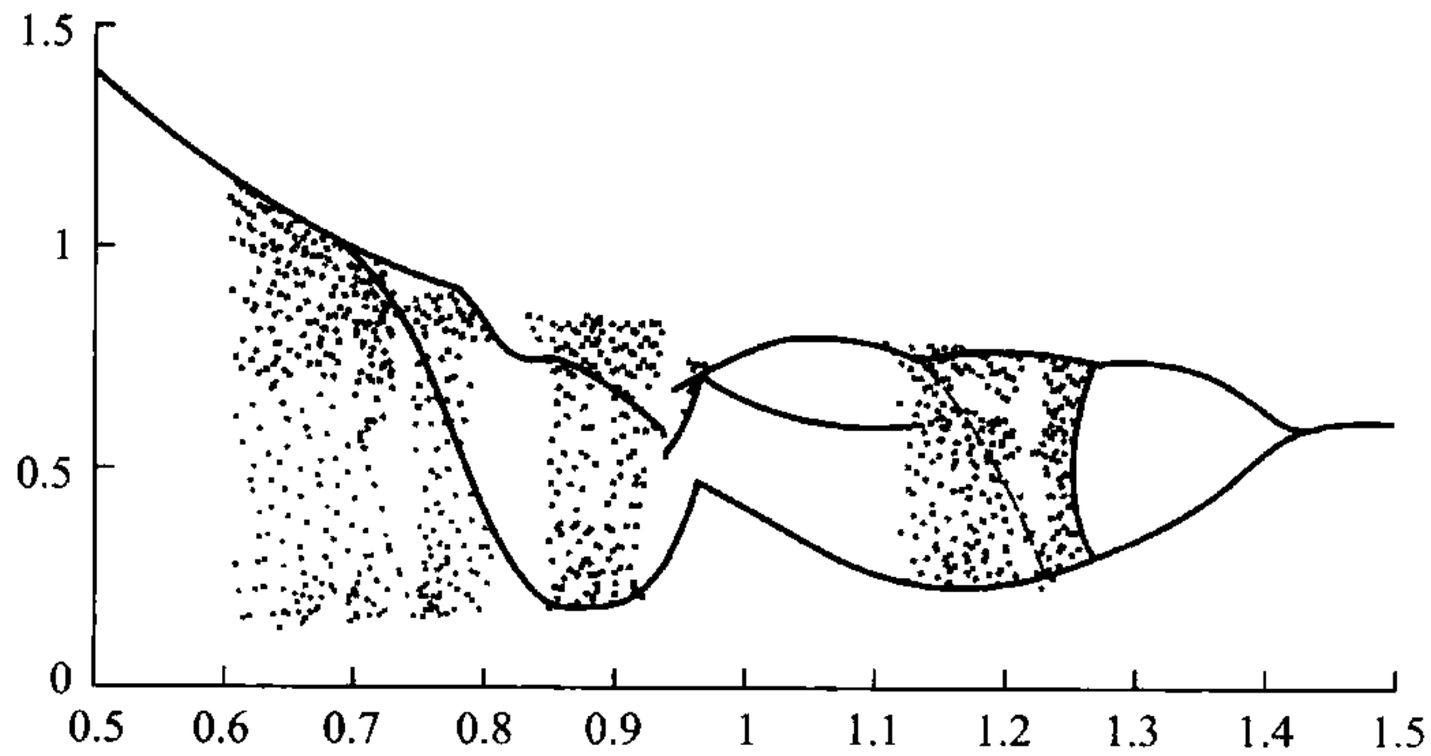


图 6.27 倒摆运动的分岔图

参数取值为$0.5 \leqslant k \leqslant 1.5$，$f = 1$，$\omega = 1$。

```
function dbd
global d
x0=0.1;   v0=0.1;  d0=0.5:0.002:1.5;
axis([0.5 1.5  -1.5 1.5])
hold on
for j=1:length(d0)
   d=d0(j);
   [t,u]=ode45(@dbfun, [0:2*pi/60:60*pi],[x0,v0]);
   plot(d,u(901:60:1800,2),'r.');
end

function ydot=dbfun(t,y)
global d
r=1;   w=1;
ydot=[y(2);
  -y(1)^3+y(1)-d*y(2)+r*cos(w*t)];
```

1. 周期 1 的情形

图 6.28 中画出角位移图，相图，庞加莱截面，傅里叶频谱图，还有动画模拟图。在图上，周期1吸引子是一个点，在频谱图是一个频率。$k = 1.5$。

2. 周期 2 的情形

在图 6.29 上，周期2吸引子是两个点，在频谱图是两个频率。$k = 1.35$。

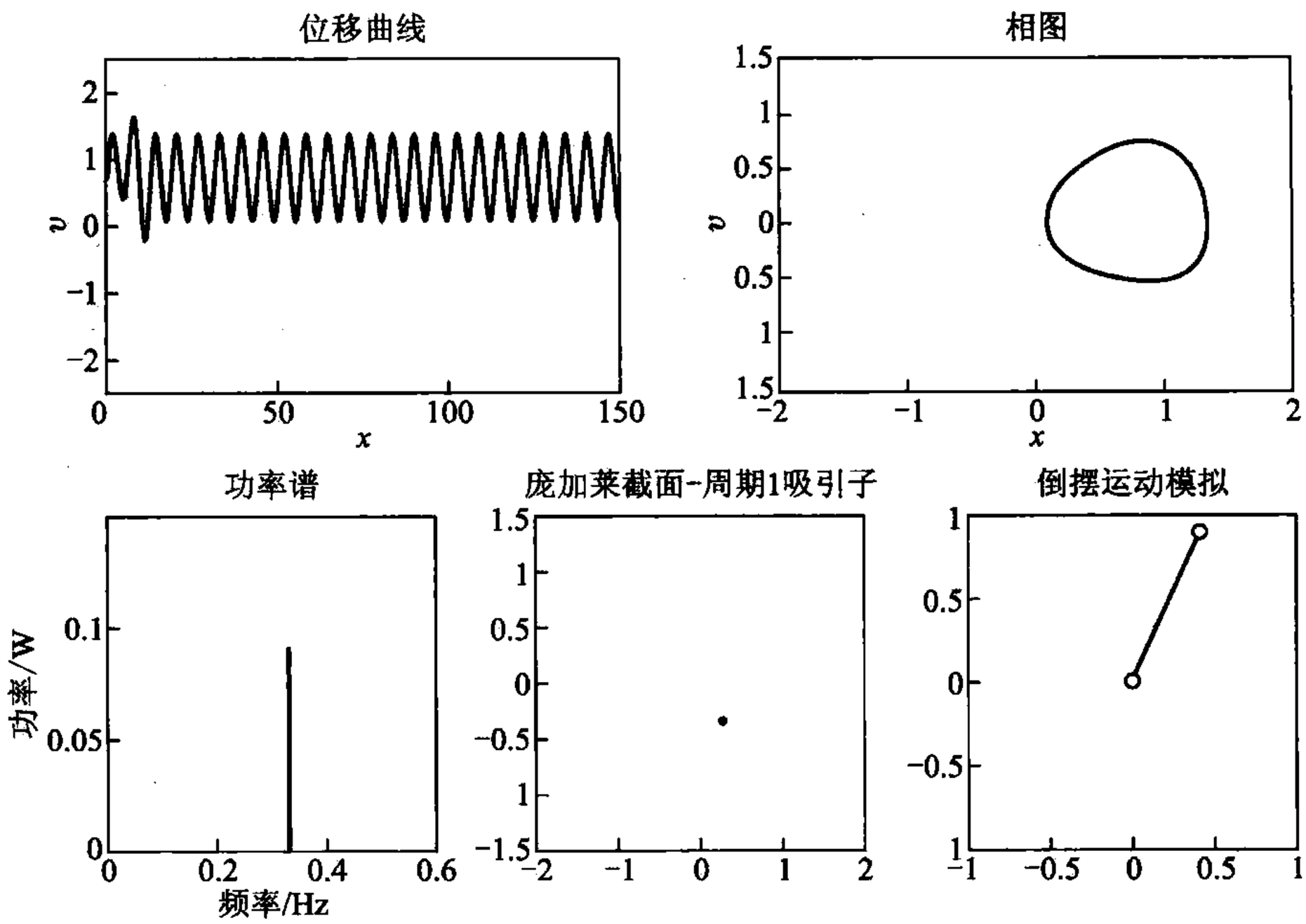


图 6.28 周期 1 运动

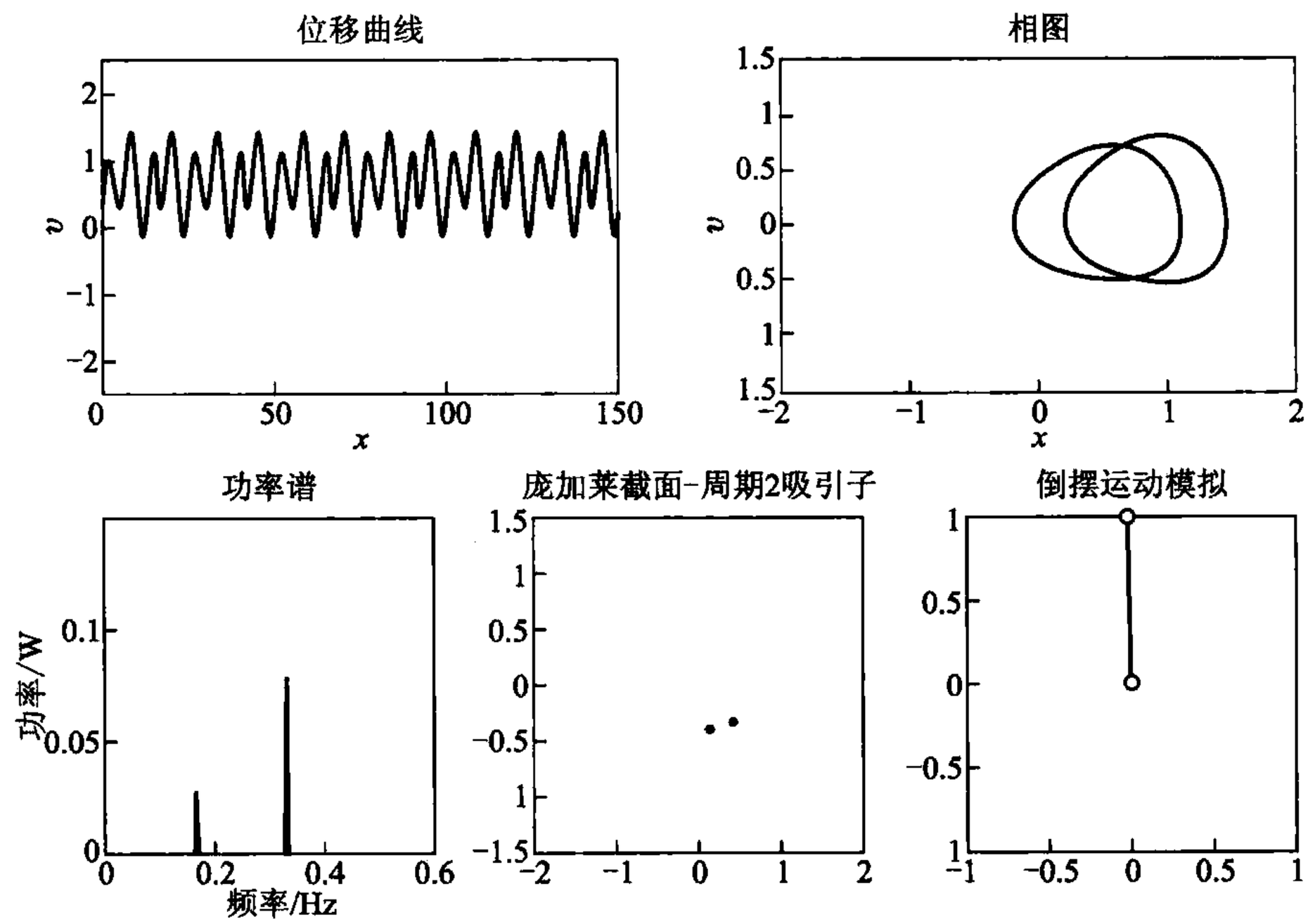


图 6.29 周期 2 运动

3. 混沌解

在图 6.30 上的奇怪吸引子，它的频谱是连续的。$k = 1.15$。

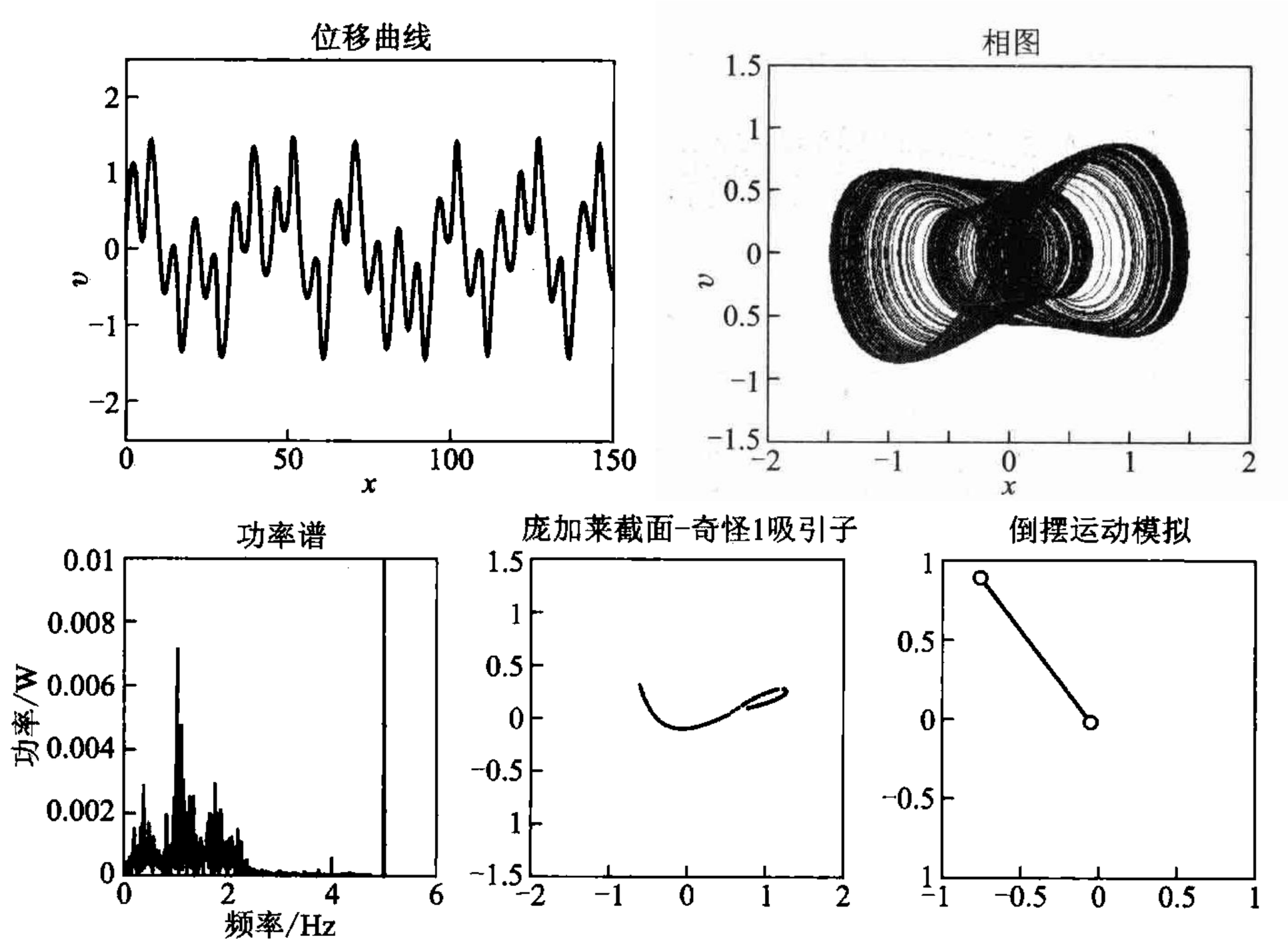


图 6.30 混沌运动

混沌状态下运动对初值十分敏感，图 6.31 是 v_0 相差0.001时形成的两条分开的曲线。

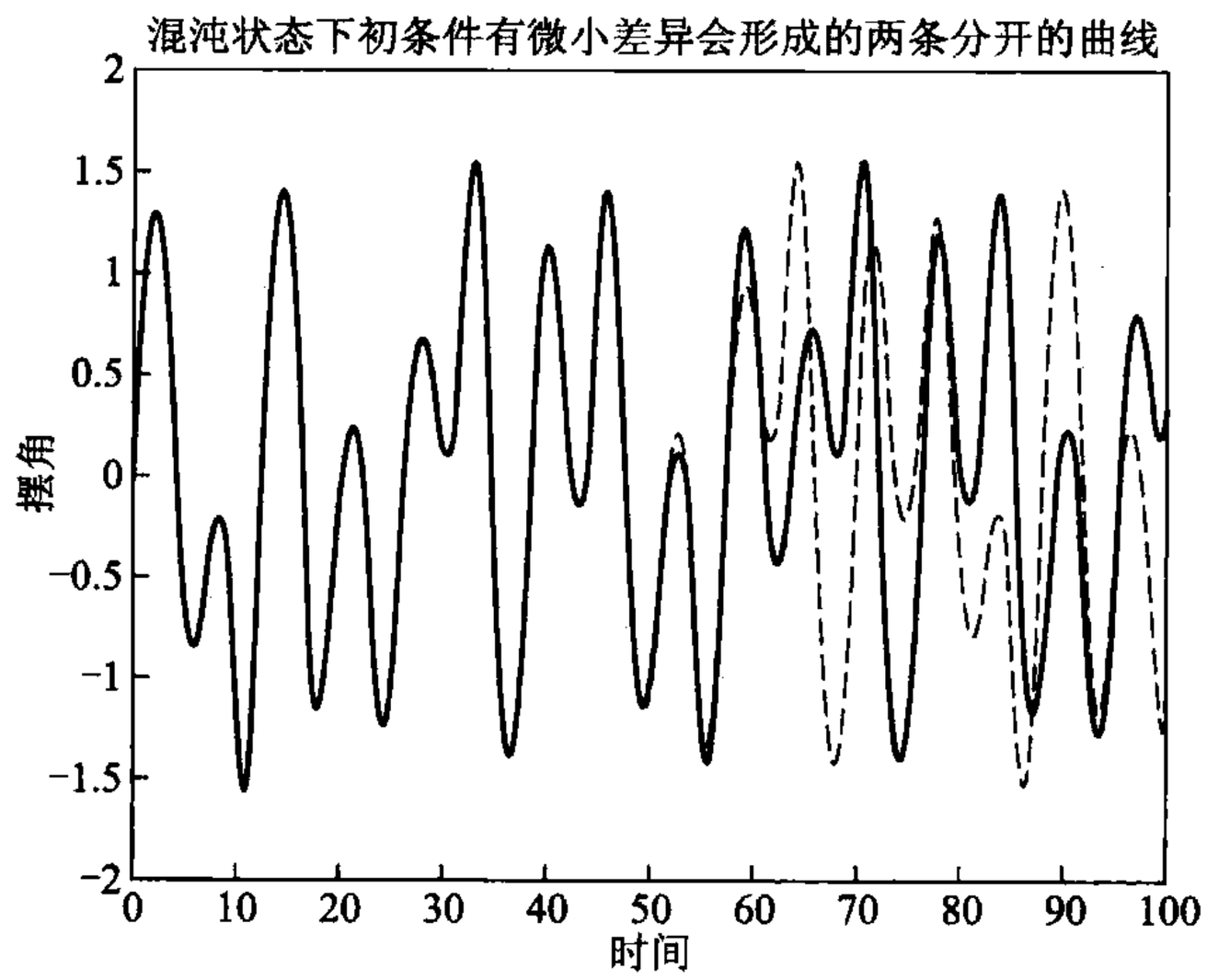


图 6.31 混沌运动对初值的敏感性

程序如下：

```
function db
```

```
global d
    x0=0.1;v0=0.1; d=0.78;        %设v0有微小的变化,比较解的变化情况
    [t,u]=ode45(@dbfun,[0:0.01:100],[x0,v0]);
    [t1,u1]=ode45(@dbfun,[0:0.01:100],[x0,v0-0.001]);
    figure
    plot(t,u(:,1),'r',t1,u1(:,1),'g')
    xlabel('%时间');
    ylabel('%摆角');
    title('%混沌状态下初条件有微小差异会形成的两条分开的曲线');

%当d=1.5,为周期1吸引子;当d=1.35为周期2吸引子;当d=1.15为奇怪吸引子,
%读者可以改变d值,以观察不同的情况
d0=[1.5,1.35,1.15];
str{1}='%庞加莱截面-周期1吸引子';
str{2}='%庞加莱截面-周期2吸引子';
str{3}='%庞加莱截面-奇怪吸引子';
for j=1:3
d=d0(j);
[t,u]=ode45(@dbfun,[0:2*pi/300:200*pi],[x0,v0]);
figure
set(gcf,'unit','normalized','Position',[0.04 0.04 0.94 0.8]);

subplot(2,2,1)                                          %位移曲线
plot(t,u(:,1))
title('%位移曲线');
axis([0,150,-2.5,2.5]);
xlabel('x');ylabel('t');

subplot(2,2,2)                                          %相图(奇怪吸引子)
plot(u(20000:end,1),u(20000:end,2))
title('%相图');
axis([-2 2 -1.5 1.5])
xlabel('x');    ylabel('v');

Y=fft(u(:,1));                                          %傅里叶功率分析
```

```
Y(1)=[]; n=length(Y); m=fix(n/2);
power=abs(Y(1:m)).^2/n^2;                                    %功率
freq=100*(1:n/2)./n;                                         %频率
subplot(2,3,4)
plot(freq,power)
axis([0 0.6 0 0.15])
title('%功率谱');
xlabel('%频率/Hz');   ylabel('%功率/w');

subplot(2,3,5)                                               %庞加莱截面
plot(u(2000:300:30000,1),u(2000:300:30000,2),'r.');
axis([-2 2 -1.5 1.5])
title(str{j});

subplot(2,3,6)                                               %实物模拟图
h=plot([0,sin(x0)],[0,cos(x0)],'o-','erasemode','xor');
axis([-1 1 -1 1])
title('%倒摆运动模拟');
for i=25000:30000
   set(h,'xData',[0,sin(u(i,1))],'yData',[0,cos(u(i,1))]);
   drawnow
end
end

function ydot=dbfun(t,y)
global d
r=1;   w=1;
ydot=[y(2);
      -y(1)^3+y(1)-d*y(2)+r*cos(w*t)];
```

§6.3 自激振动——范德波尔方程

§6.3.1 运动方程

下面研究范德波尔（Van der pol）方程

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}-\mu\left(x_0^2-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t}+\omega_0^2x=0 \tag{6.14}$$

它所描述的是非线性有阻尼的自激振动系统，其中 μ 是一个小的正的参量，x_0是常数。下面简称范德波尔方程为 VDP 方程。

在 VDP 方程中，增加外驱动力$V\cos\omega t$项所得到的方程

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}-\mu\left(x_0^2-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t}+\omega_0^2x+V\cos\omega t=0 \tag{6.15}$$

称强迫 VDP 方程，其中外驱动力的振幅、角频率分别是V和ω。

自激系统是一个非线性有阻尼的振动系统，在运动过程中伴随有能量损耗。但系统存在一种机制，使能量能够由非振动的能源通过系统本身的反馈调节，及时适量地得到补充，从而产生一个稳定的不衰减的周期运动，这样的振动称为自激振动。强迫VDP方程的特点是非线性体现在速度项上（一阶项），而杜芬方程的非线性体现在恢复力上(零阶项)，它们都能产生非线性振荡。

对 VDP 方程，可从机械振动角度理解，$-\mu\left(x_0^2-x^2\right)$是阻尼系数，它是变化的。如果$|x|>|x_0|$，则阻尼系数为正，系统将消耗能量；但如果$|x|<|x_0|$，则发生负阻尼，意味着不仅不消耗系统的能量，反而给系统提供能量。此系统能通过自动的反馈调节，使得在一个振动过程中，补充的能量正好等于消耗的能量，从而系统作稳定的周期振动。

取方程中的$x_0^2=1$，$\omega_0^2=1$，$\mu=0.3$（这些值可适当调整）。给出任一初始条件，通过计算机数值求解可以证明它的相轨道都将趋向于一条闭合曲线(称为极限环)，如图 6.32。

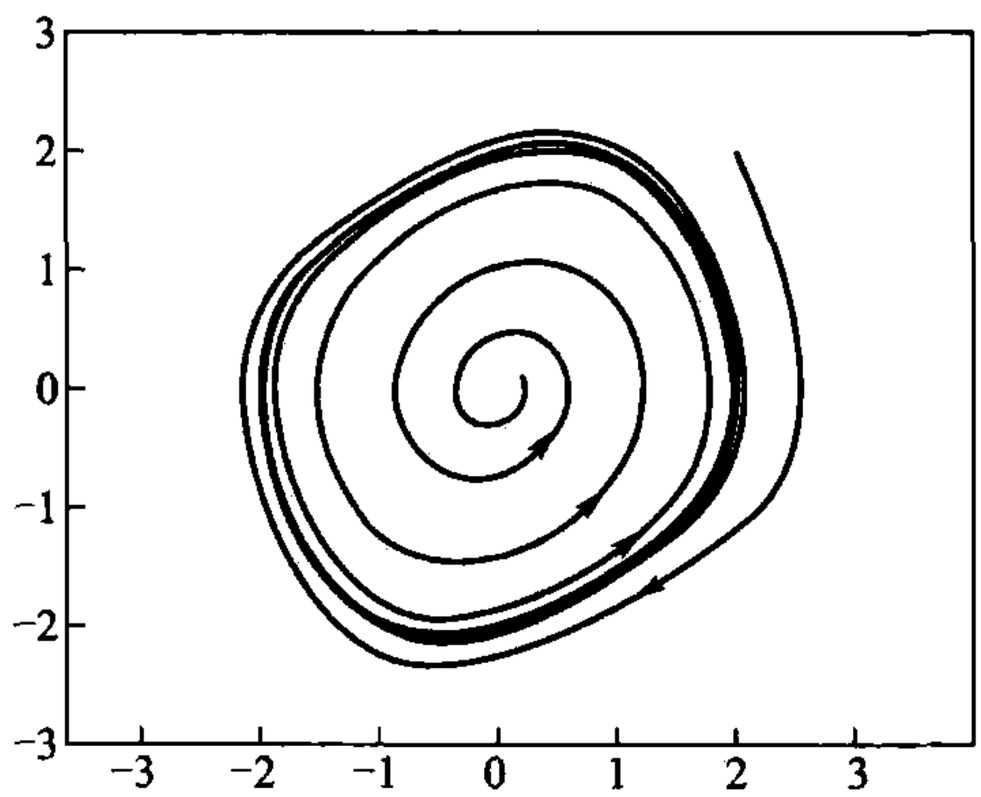


图 6.32 相图中的极限环

极限环以外的相轨道向里盘旋，而极限环以内的相轨道则向外盘旋，都趋向极限环（如图 6.32 所示），说明不论初始情况如何，系统最终都到达以极限环描述的周期性运动。由于这段程序较简单，我们没有专门编写。事实上，只要将下面编写的关于强迫VDP方程的程序中令 $V=0$，$\mu=0.3$，再取不同的初始条件，就能看到这个现象。

§6.3.2 VDP 方程通向混沌的道路

下面研究强迫 VDP 方程的行为。我们同时采用时间历程图、相图和庞加莱映像图来研究系统在不同参数条件下的动力学行为，可以看到存在不同的吸引子，即周期1吸引子、周期2吸引子、不变环面吸引子和奇怪吸引子。

设 $y_1 = x$，$y_2 = \mathrm{d}x/\mathrm{d}t$，则 (6.15) 式可化为

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = \mu\left(x_0^2 - y_1^2\right) y_2 - \omega_0^2 y_1 - V\cos\omega t \end{cases} \tag{6.16}$$

取$x_0^2 = 1$，$\omega_0^2 = 1$，作数值计算研究并画出位移图，相图，频谱图及庞加莱截面图。

(1) 在$\mu = 0.85$，$V = 1$，$\omega = 0.44$条件下，存在周期1吸引子，它的周期等于外激励的周期，代表主谐波运动。虽然频谱图上显示有多个频率，但庞加莱图表明，轨线只是按主谐波运动。为了能显示更多的频率，没有画出振幅的最大值，如图 6.33 所示。

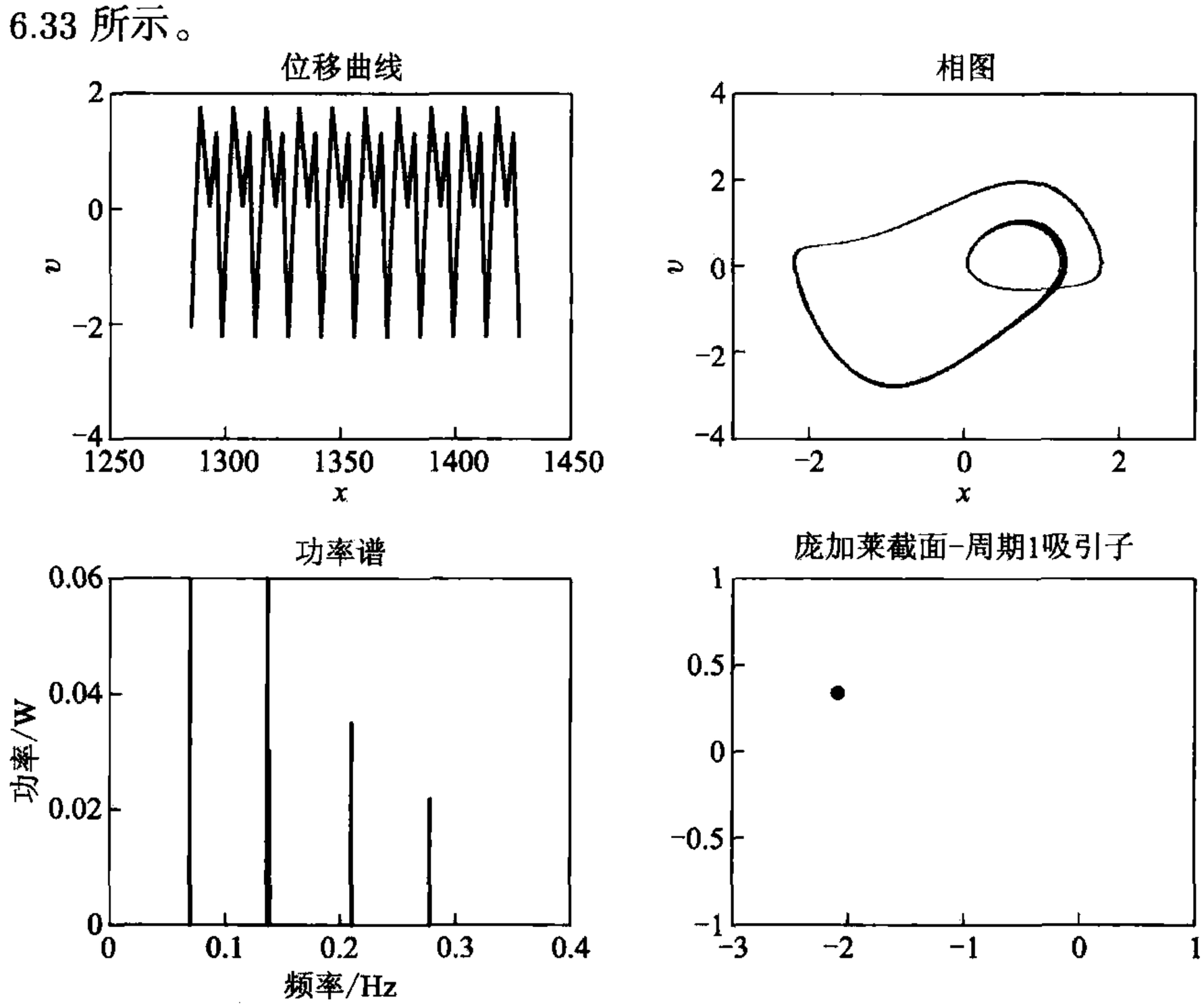


图 6.33 强迫 VDP 方程的振动的周期 1 吸引子

为了对比，画出驱动频率的位移图及频谱分析图 (图 6.34)，请注意计算中的W是圆频率，而FFT分析所得的是频率，它的值是$w/2\pi$，计算使用的是$w =$

0.44，所以在频谱图显示的是 0.07。

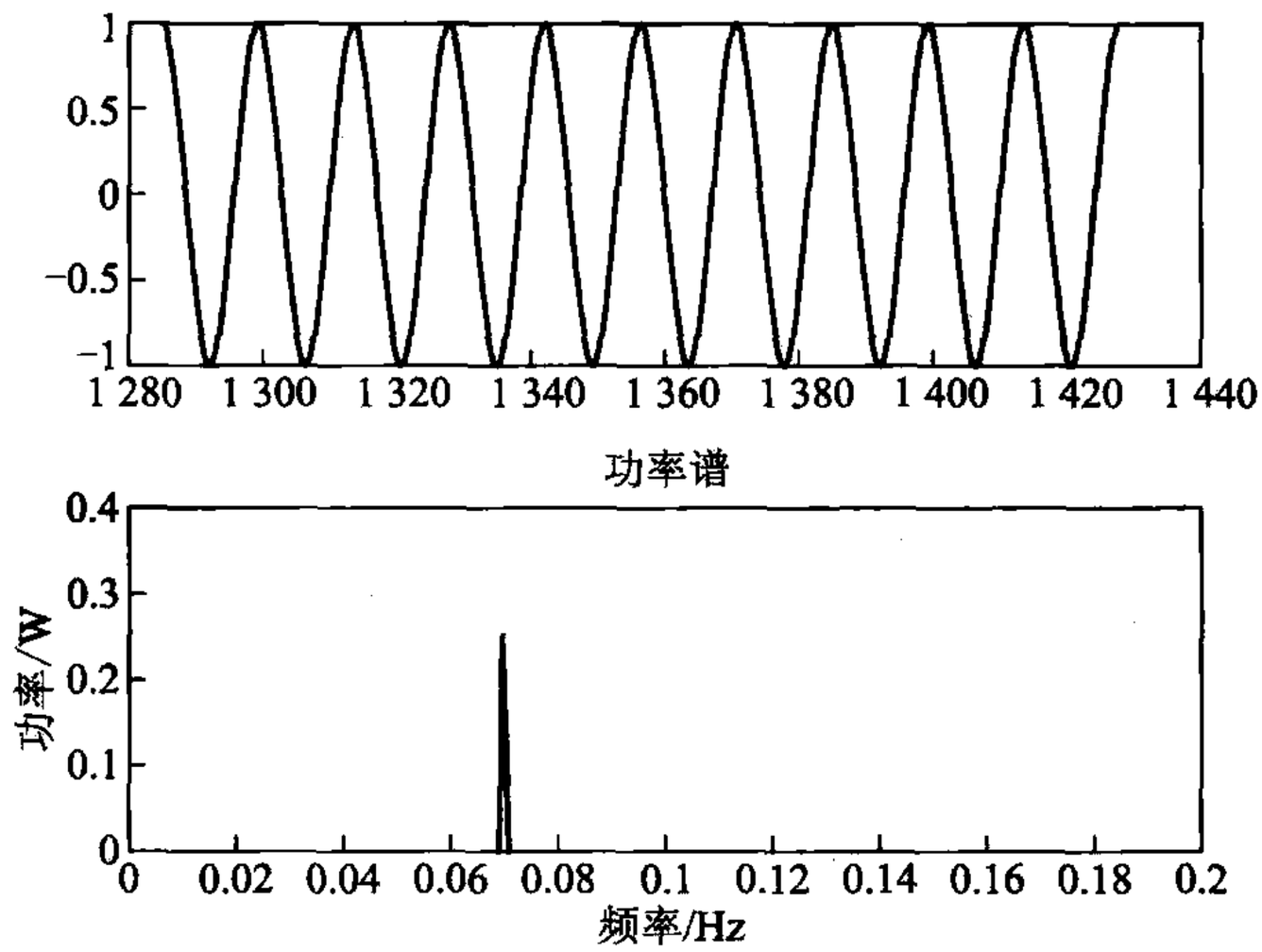


图 6.34 驱动频率的位移图与频率分布图

(2) 在$\mu = 1.02$，$V = 1$，$\omega = 0.44$条件下，存在周期2吸引子，它的周期等于外激励的整数倍，代表次谐波运动。如图 6.35 所示。

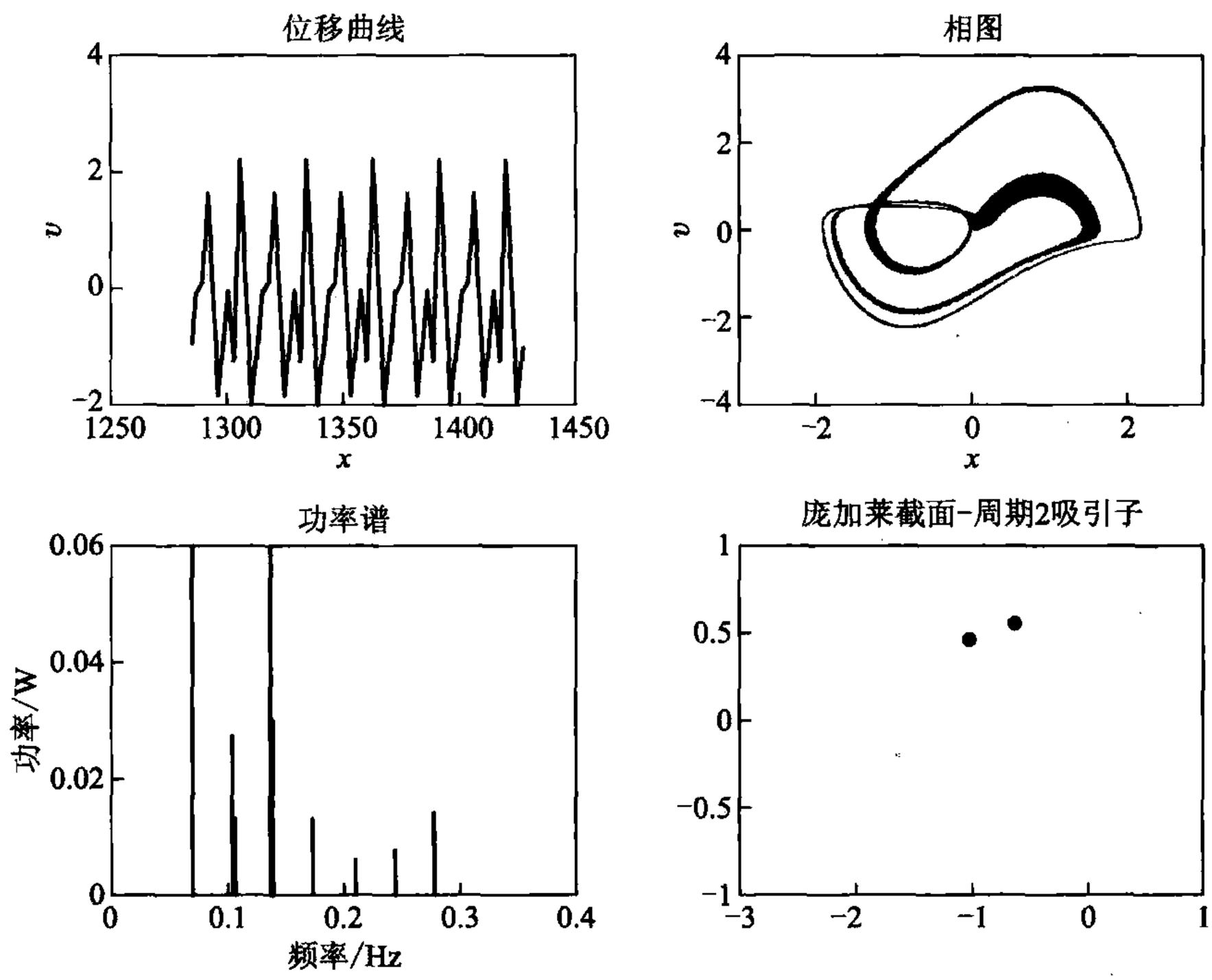


图 6.35 强迫 VDP 方程的振动的周期 2 吸引子

(3) 在$\mu = 0.66$，$V = 1$，$\omega = 0.44$ 条件下，存在不变环面吸引子，它代表拟周期运动。此时输出的波形类似于一种调幅波，系统有如处于“差拍”状态，频谱分析表明，准周期状态可以分解为一系列周期解，在频谱分布上有一定规律，如图 6.36 所示。

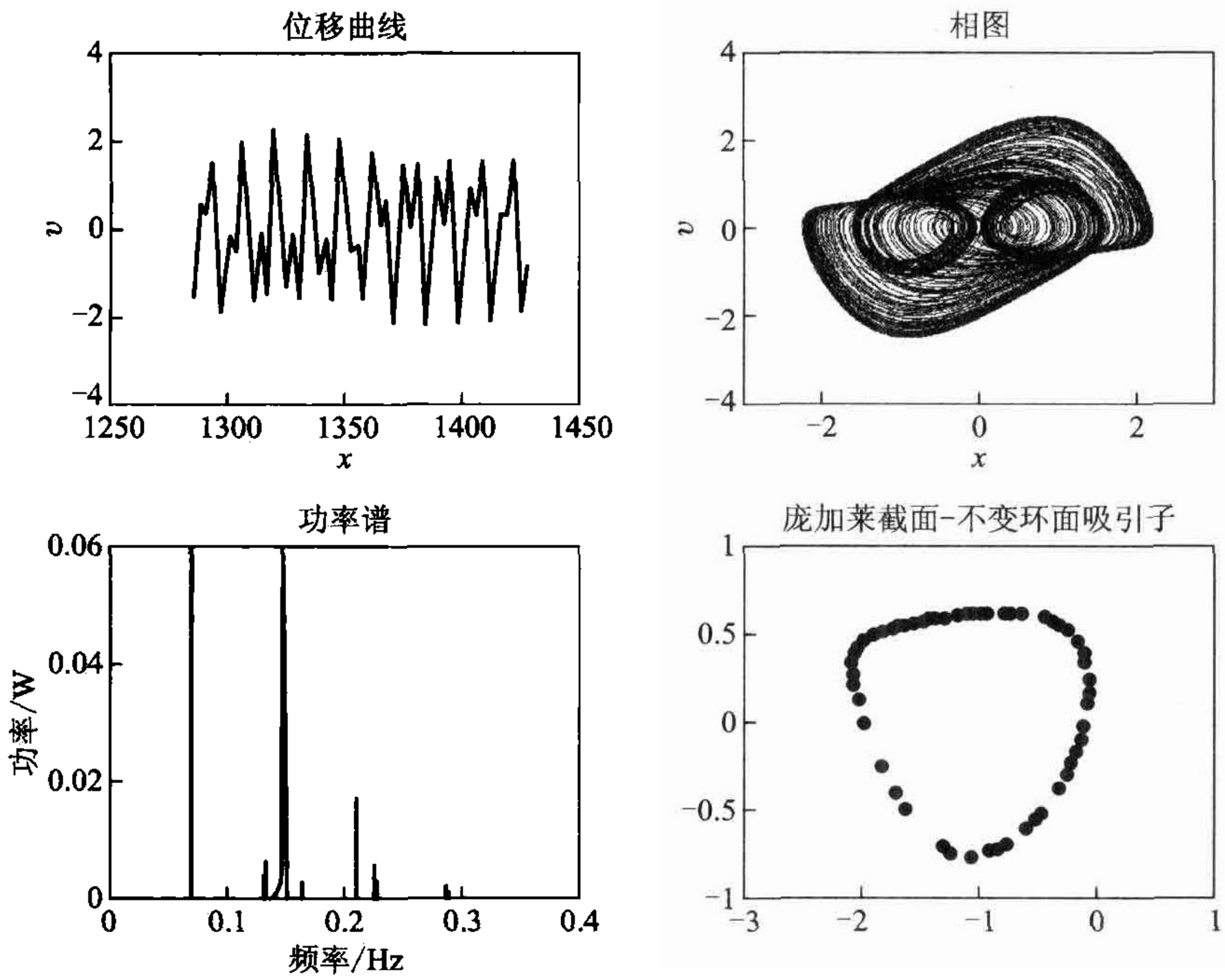


图 6.36 强迫 VDP 方程的振动的不变环面吸引子

(4) 在$\mu = 1.08$，$V = 1$，$\omega = 0.44$条件下，存在奇怪吸引子，代表混沌运动。如图 6.37 所示。

(5) 保持V和ω为定值，逐渐增大μ，将显示系统状态演化过程全貌的图，如图 6.38 所示。而前四种情况中，看到的只是μ取4个值的片段情况。图形显示，当μ由0.9连续变化到1.2时，系统运动状态逐渐由周期1过渡到周期2（发生了周期倍分岔）再过渡到混沌状态。

在程序中，这几种过程的计算是相同的，所以用 for 循环来完成前面四种计算，这就是程序 zjzd.m。计算中在每个外激励周期内计算 1000 个相点，为了作出庞加莱映像，每隔 1000 个点保留一个点数据，所以程序运行的时间较长。对第五种情况，由于计算量大，将它另外编写一个程序，这就是程序 zjzd1.m。计算中在每个周期内计算 100 个相点，庞加莱映像是每隔 100 个点保留一个点数据。

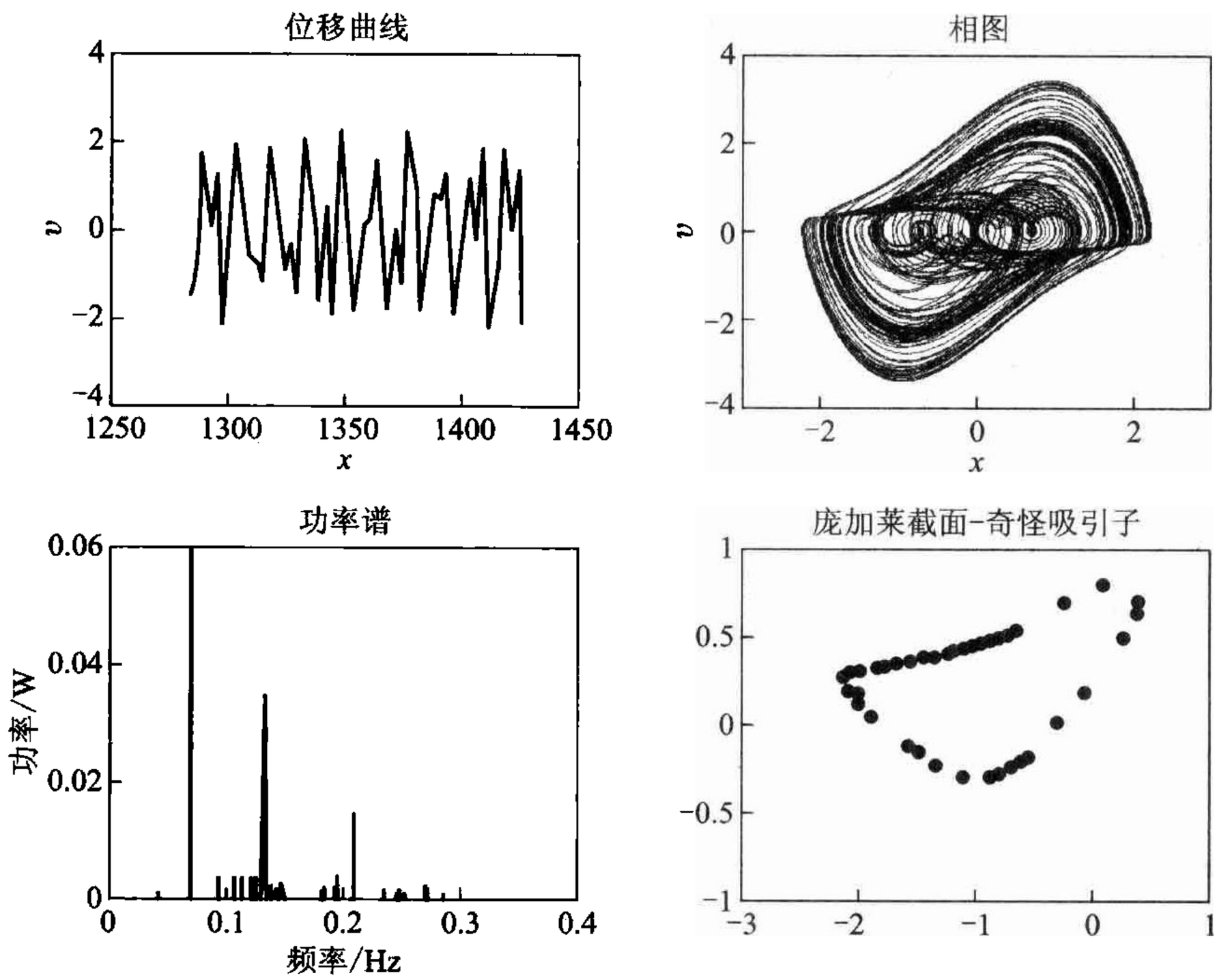


图 6.37 强迫 VDP 方程的振动的奇怪吸引子

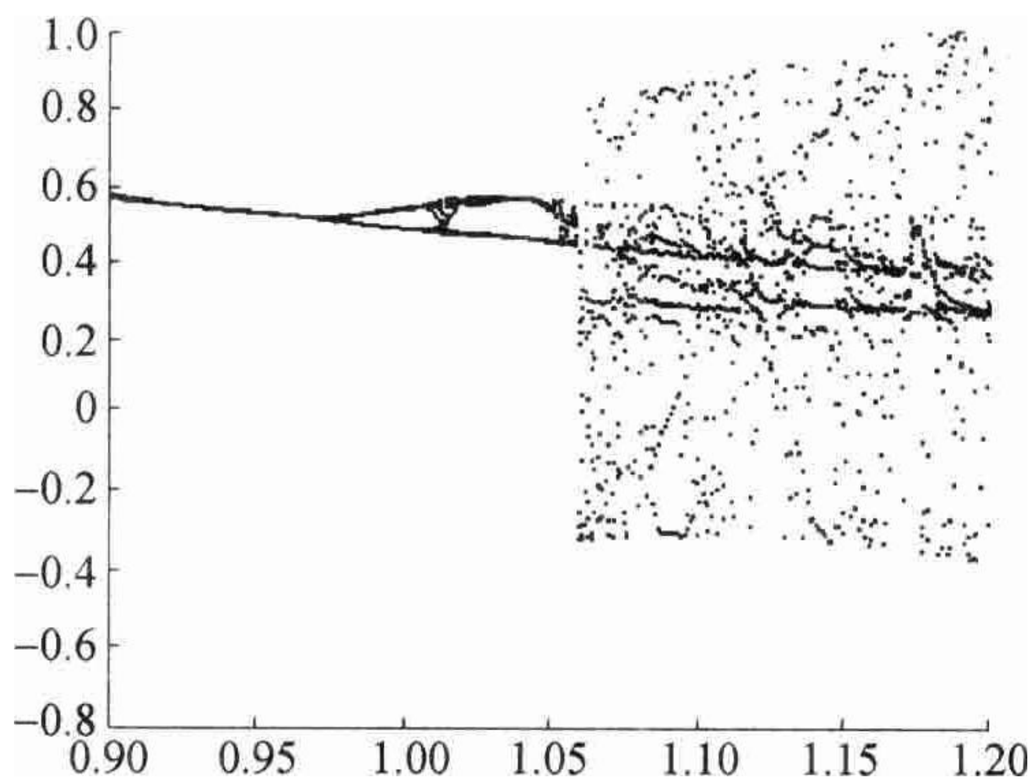


图 6.38 μ值连续变化所产生的强迫VDP方程的庞加莱映像

```
function zjzd
global uu  w  v
u=[0.85,  1.03,  0.6,  1.0732];
w=0.44;    T=2*pi/w;       v=1;
str{1}='%庞加莱截面-周期1吸引子';
str{2}='%庞加莱截面-周期2吸引子';
```

```
str{3}='%庞加莱截面-不变环面吸引子';
str{4}='%庞加莱截面-奇怪吸引子';

for  j=1:4
figure
uu=u(j);
[t,y]=ode45(@vdpfun,[0:T/1000:100*T],[4,4]);
subplot(2,2,1)
plot(t(90000:end),y(90000:end,1));
title('%位移曲线');
xlabel('x');ylabel('v');

subplot(2,2,2)
plot(y(3000:end,1),y(3000:end,2));
axis([-3 3 -4 4])
xlabel('x');ylabel('v');
title('%相图');

subplot(2,2,3)
Y=fft(y(:,1));                                          %傅里叶分析
Y(1)=[]; n=length(Y); m=fix(n/2);
power=abs(Y(1:m)).^2/n^2;                               %功率
dt=t(2)-t(1);
freq=1/dt*(1:n/2)./n;                                   %正确计算频率
plot(freq,power)
axis([0 0.4 0 0.06])
hold on
title('%功率谱');
xlabel('%频率/Hz');   ylabel('%功率/w');

subplot(2,2,4)
axis([-3 1 -1 1])
hold on
   for i=5000:1000:100000
      plot(y(i,1),y(i,2),'r.');
```

```
    end
title(str{j});

end

function ydot=vdpfun(t,y)
global uu w v
x0=1;   w0=1;
ydot=[y(2);
   uu*(x0^2-y(1)^2)*y(2)-y(1)*w0^2-v*cos(w*t)];
```

§6.3.3 吸引子类型及其频谱

在 VDP 方程的研究中，出现了一种新的吸引子，拟周期吸引子。在此对吸引子作些更详细的介绍。

前面说过，耗散系统存在吸引子,所谓吸引子是时间$t \to \infty$时状态的归宿。吸引子可分为两大类，一类是规则吸引子，它通常具有规则的几何形状，吸引子上的时间演化是规则的，如定常吸引子、周期吸引子（极限环）、拟周期吸引子（二维或三维环面）等；另一类是不规则的吸引子，如奇怪吸引子，其几何特点是形状不规则，具有分维结构。奇怪吸引子往往对应着混沌运动。简单介绍如下：

(1) 定常吸引子

具有这种吸引子的系统在相空间中其轨线趋于一个固定点，不管系统从什么初始状态出发，其长期演化的归宿是恒定不变的，总是停在相空间中的一个固定点上。如二维空间中稳定的结点和焦点，这是最简单的一类吸引子。它的功率谱上只有一个主要频率（基频），如单摆的有阻尼无驱动的运动。

(2) 周期吸引子

是二维或三维相空间中的闭环，或称极限环，它描述的是稳定振荡，例如：钟摆的周期运动，倒摆的周期2，周期4等周期分岔的运动。它是刻画周期行为的吸引子。这种系统从某一初始状态出发，经过一个短暂的过程后直接进入周期运动，一旦系统进入周期运动，在相空间中，其轨道就固定在闭环上，系统作周期运动，其相空间的轨线就沿闭环周而复始，永远循环。它的功率谱是分立的，它除了包含有基频f_0之外，还含有高次谐波的频率$2f_0$，$3f_0$，$4f_0$，$\cdots$及其分波的频率$f_0/2$，$f_0/3$，$f_0/4$，$\cdots$，如VDP方程中周期1，周期2的振荡运动。通常振幅较大的频率会表现得更明显一些。

(3) 环面吸引子，或称拟周期吸引子

它描述复合振荡的拟周期行为，它的轨道在三维相空间中的一个环面(像轮

胎面)上绕行而且永不重复永不相交。如自激振动的VDP方程中，存在着驱动频率ω与系统运动频率$\dot{x}$。将方程组 (6.16) 写成自治方程组形式

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = \mu\left(x_0^2 - y_1^2\right) y_2 - \omega_0^2 y_1 - V\cos y_3 \\ \dfrac{\mathrm{d}y_3}{\mathrm{d}t} = \omega \end{cases} \tag{6.17}$$

得到以y_1, y_2, y_3构成的三维相空间，像分析单摆运动一样，把具有周期性的y_3弯曲并将$2n\pi$与$2(n+1)\pi$面重合，就形成二维轮胎面，轨线将同时以ω与y_2（对应$\dot{x}$）二种频率在其上作环绕运动，ω表示轨线绕垂直于轮胎面的中心轴的旋转频率，y_2表示轨线在轮胎面上环绕的旋转频率，二者之比y_2/ω称为旋转数或绕数。

若绕数为有理数n/m，则表明：ω经过n次旋转而$\dot{x}$经过m次旋转轨线将重合，因此形成闭环。庞加莱截面是以y_3为固定值截取的，显示的则是y_2的旋转次数即$\dot{x}$的频率，这就是周期吸引子的情况。

若绕数为无理数，即两种频率不可公度，那么，无论沿环面横绕（沿轮胎面）或竖绕（绕中心轴）多少圈，轨线也不会闭合。轨线将稠密地分布在环面上，永远不重复已走的路。这是一种非周期运动，但又是由两个周期运动合成的，所以叫做拟周期（或准周期）运动。

拟周期吸引子的频谱也是分立的，它包含有各种频率，但与周期吸引子的频谱不同之处在于，这些频率之比是无理数。如果系统的绕数f_1/f_2是无理数，则系统的运动中除了有这两个频率之外，由于非线性相互作用，还会产生新的频率$kf_1 + lf_2$，k, l为任意整数。所以它的谱线间的间隔的规律不同于周期吸引子谱线之间等间隔的分布。如VDP方程的拟周期吸引子的频谱。

(4) 混沌吸引子

在Logistic模型，单摆，倒摆，VDP方程中我们都遇到混沌现象。对混沌有各种定义，但是还没有一个公认的普遍适用的数学定义。在这里，我们把有于别定常运动，周期运动，准周期（拟周期）运动之外的一种具有普遍意义的运动形式理解为混沌。它对初值敏感同时又具有非周期性，是确定论系统中局限于有限相空间的，轨道高度不稳定的运动。在一维和二维的自治系统是不可能出现混沌的，只有三维或三维以上的系统才有混沌。

混沌吸引子是在耗散系统中出现的吸引子，常叫奇怪吸引子。一般说来，它会具有下列特点：

- 稳定性　奇怪吸引子局限于有限区域内。这是由于耗散运动最终要收缩到相空间的有限区域即吸引子上。就大范围而言，它表现为稳定的吸引子。若以吸引域内任意一点为初值，当受到干扰时，根据吸引子的定义，其动力系统的最终状态仍然会回到这个吸引子上，并可得到几乎完全相同的奇怪吸引子。
- 低维性　在相空间存在一条低维（分维）的轨道，尽管它的自由度是有限的几个，但却有复杂的空间结构。轨道可以无穷的伸展，压缩与折叠。所以奇怪吸引子具有无穷嵌套的自相似结构。
- 非周期性　轨道永不重复，不会相交。否则就成为周期吸引子。
- 对初值敏感　相邻的轨道成指数状分离，轨道有局部不稳定性，进入奇怪吸引子区域的轨道稍有差异，其后的发展就会截然不同。

注意混沌吸引子通常具有分形结构，因此称为奇怪吸引子。实际上混沌吸引子是一种动力学概念，而奇怪吸引子则是一种几何上的概念。由于混沌吸引子可以不具有分形结构，而具有分形结构的吸引子未必就是混沌吸引子，所以二者没有必然的联系。但是研究表明，混沌吸引子往往具有分形维数，因而往往是奇怪吸引子，所以常会把混沌吸引子叫作奇怪吸引子。奇怪吸引子不是周期函数，它的谱会具有连续的分布，正如在VDP方程中所看到那样，在某段频率区间频谱是连续的。从这一点容易看出拟周期吸引子与混沌吸引子的区别，而在相图上二者似乎难以区分。

不过这种谱也不同于随机白噪声的频谱。白噪声是由大量独立的因素产生的，其功率谱的振幅与频率无关，所以各种频率的振幅高度大致相同，是一种“平谱”。而混沌运动由于不是周期运动，所以会有连续谱。但是它的运动极其复杂，以倍周期分岔为例，每次分岔都会导致一批新的分频与倍频产生，所以混沌的频谱不是平谱，即在噪声背景上出现一些宽峰。

这四种吸引子可以用李雅普诺夫指数来区分。对三维的自治系统，假定三个初始条件是$\delta x(0)$，$\delta y(0)$，$\delta z(0)$，经过t时刻以后成为

$$\begin{cases} \delta x(t) = \delta x(0)\mathrm{e}^{LE_1 \cdot t} \\ \delta y(t) = \delta y(0)\mathrm{e}^{LE_2 \cdot t} \\ \delta z(t) = \delta z(0)\mathrm{e}^{LE_3 \cdot t} \end{cases} \tag{6.18}$$

其中LE_1，LE_2，LE_3分别是三个方向的李指数。将上面三式相乘得相体积的空间变化率

$$\delta v(t) = \delta v(0)\mathrm{e}^{(LE_1+LE_2+LE_3)t}$$

两边取对数得

$$LE_1 + LE_2 + LE_3 = \frac{1}{t}\ln\frac{\delta v(t)}{\delta v(0)} \tag{6.19}$$

根据耗散定义，在相空间的体积要缩小，但 (6.18) 式表现的是局部的性质，(6.19) 式表现的是整体的平均性质。

如果记

$$\mathrm{div}\boldsymbol{u} = \frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y} + \frac{\partial \dot{z}}{\partial z}$$

则相空间相体积的时间变化率是

$$\frac{1}{V}\frac{\mathrm{d}V}{\mathrm{d}t} = \mathrm{div}\boldsymbol{u}$$

将它积分

$$\ln\frac{V(t)}{V(0)} = \int_0^t \mathrm{div}\boldsymbol{u}\mathrm{d}\boldsymbol{t}$$

与 (6.19) 式比较得

$$LE_1 + LE_2 + LE_3 = \mathrm{div}\boldsymbol{u}$$

它也是雅可比矩阵的三个本征值$\lambda_{1,2,3}$之和

$$\lambda_1 + \lambda_2 + \lambda_3 = \mathrm{div}\boldsymbol{V}$$

这里应该区分本征值λ与李指数LE，对耗散系统而言，本征值是反映平衡点附近（即局部）的轨道性质，虽然要求$\mathrm{div}\boldsymbol{V} < 0$，但该平衡点不稳定时，三个$\lambda$中,有的可以变为实部为正，这就是所谓局部不稳定的“伸长”，而整体上讲，耗散会使轨道收缩到有限范围内的吸引子上，李指数是长时间t 内的平均结果。它已经计入λ上所有各点的影响，因而是耗散系统整体上“折叠”性质。λ可是复数，但LE肯定是实数。

在三维耗散系统中，对平衡态吸引子而言，它三个方向均要收缩，因而满足$LE_1 < 0$，$LE_2 < 0$，$LE_3 < 0$。使LE按大小顺序排列，有$LE_1 \geqslant LE_2 \geqslant LE_3$，表示为$(-,-,-)$，如图 6.39(a)。

周期吸引子（极限环）在环所在的平面方向以及垂直该平面方向上都要收缩，故$LE_2 < 0$，$LE_3 < 0$，而在极限环切线方向，它既不增大也不收缩，故$LE_1 = 0$，表示为$(0,-,-)$，如图 6.39(b)。

对二维环面上的拟周期吸引子，在环面上两个方向（一个方向相当于自转，另一个方向相当于公转）上不增大也不收缩。因此，$LE_1 = LE_2 = 0$，只有在环面外，所有轨道都要向环面上收缩，即$LE_3 < 0$，表示为$(0,0,-)$，如图 6.39(c)。

只有混沌吸引子，沿轨道方向上$LE_2 = 0$，伸长方向$LE_1 > 0$，收缩方向$LE_3 < 0$，并且$LE_1 + LE_2 + LE_3 < 0$，表示为$(+,0,-)$，如图 6.39(d)。

从上面讨论来看，这 4 种吸引子中只有混沌吸引子有正的李指数，这是和其他 3 种吸引子的相区别的唯一标志。

同时也要看到，耗散系统中若有源项造成平衡点不稳定，如不稳定的结点和不稳定的焦点，此时三个李指数均为正，它也没有折叠回来的可能，这和混沌吸引子是有区别的。

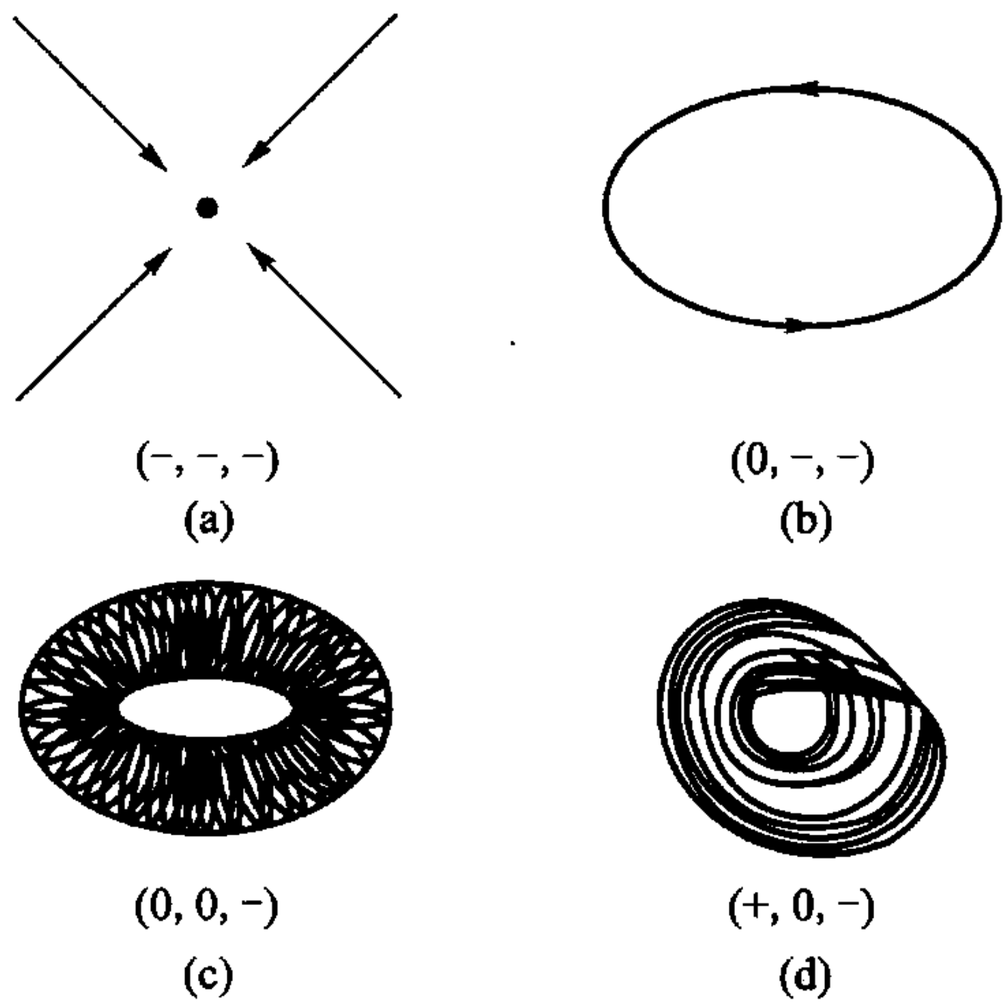


图 6.39 四种吸引子的李雅普诺夫特征指数

§6.3.4 分岔

动力学方程中一般会含有控制参数，如单摆和倒摆方程的驱动力的振幅f和阻尼系数β，k等。正如上面所分析的那样，当这些控制参数发生改变时，方程的解也发生了变化。我们曾经画出单摆的频率随驱动力振幅变化的分岔图以及倒摆的频率随阻尼系数变化的分岔图来描述这些现象。一般来说，改变系统的控制参数μ，系统解的稳定性会发生改变，当旧的解失稳时，就会有新的稳定解代替其出现。这种定性的结构变化称为分岔。

分岔原为微分方程理论中的名词，原来的含义是一分为二，后来被推广应用在动力学系统研究中，用以描述当系统的控制参数发生改变所引起的相图的拓扑结构的变化。分岔理论是研究非线性方程（代数方程、微分方程、积分方程等）解的定性行为的数学理论。它包括分岔点的位置，分岔解的方向与数目，分岔解的稳定性，分岔的类型，分岔的过程与终态（如奇怪吸引子）等。可见分岔理论除了涉及非线性稳定理论外，还含有突变和混沌等新的内容。另外，从分岔过程来看，失稳是发生分岔的物理前提。分岔以后，系统不同状态间便产生了不连续的过渡，这就是突变，然后经过不断地分岔，最后系统所达到的终态就是混沌理论的研究对象。可见，分岔在许多非线性现象中还起着桥梁和纽带的作用，从而使分岔与混沌理论为研究自然界的复杂现象提供了有效手段，它们构成了非线性动力学近代理论的基本内容。

动力学方程的解的性质是用雅可比矩阵来描述的，改变动力学方程的控制参数μ，会使雅可比矩阵的本征值λ发生变化而引起不动点失稳。这种变化可以看成是全体本征值$\lambda(\mu)$在复平面（Reλ，Imλ）上的运动，当控制参数达到某值μ_0时，系统的稳定性发生了变化，这就是分岔。一般来讲，本征值的实部都会变化，引起 $\mathrm{Re}(\lambda(\mu))=0$，即在 $\mathrm{Re}(\lambda(\mu))=0$的前后，稳定性是不同的，所以分岔点要满足条件

$$\frac{\mathrm{d}}{\mathrm{d}\mu}\mathrm{Re}(\lambda(\mu))\neq 0$$

常见的情形有：

1. 叉型分岔

典型的方程是

$$\dot{x}=\mu x-x^3=f(x,\mu)$$

式中 x，μ 是实数。当$\dot{x}=0$时，求出不动点为下列情况：

当 $\mu<0$ 时，有一个不动点 $x=0$。

当 $\mu>0$ 时，有三个不动点 $x=0$，$\pm\sqrt{\mu}$。

不动点对应的雅可比矩阵为$\mu-3x^2$，而雅可比矩阵的本征值是 $\lambda=\mu-3x^2$。再用本征值来判断不动点的稳定性，可知：

当 $x=0$ 时，本征值为$\lambda=\mu$，如果$\mu<0$，则不动点$x=0$是稳定的，如果$\mu>0$，则$x=0$ 是不稳定的；

当$x=\pm\sqrt{\mu}$时，本征值为$\lambda=-2\mu$，由于$\mu>0$，所以这两个不动点都是稳定的。

这些分析可以用下左图表示，当μ由负变正时，原来稳定的不动点$x=0$变得不稳定的，然后又分岔成两个新的稳定的不动点$x=\pm\sqrt{\mu}$。这就是叉型分岔，分岔点是$\mu=0$，$x=0$。在分岔点还满足条件

$$\begin{cases}\left.\dfrac{\partial f}{\partial x}\right|_{\substack{x=0\\ \mu=0}}\Rightarrow \left.\mu-3x^2\right|_{\substack{x=0\\ \mu=0}}=0\\[2ex] \left.\dfrac{\partial f}{\partial \mu}\right|_{\substack{x=0\\ \mu=0}}\Rightarrow \left.x\right|_{\substack{x=0\\ \mu=0}}=0\end{cases}$$

如果典型方程变成

$$\dot{x}=\mu x+x^3$$

那么同样的分析知，叉型分岔图为图 6.40(b) 。它表示当μ由正变负时，原来不稳定的不动点$x=0$变得稳定的，然后又分岔成两个新的不稳定的不动点$x=\pm\sqrt{-\mu}$。在图 6.40 中是用实线表示稳定的平衡态，用虚线表示不稳定的平衡态。图 6.40(a) 也叫超临界分岔，图 6.40(b) 叫亚临界分岔。

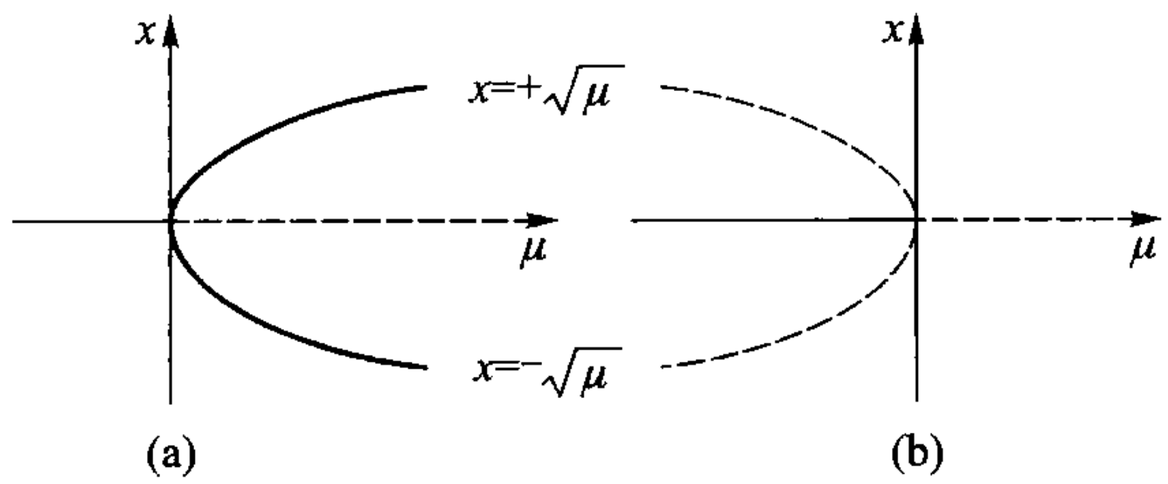


图 6.40 叉型分岔

2. 霍普夫分岔 (Hopf bifurcation)

典型方程为

$$\begin{cases} \dot{x} = -y + x[\mu - (x^2 + y^2)] \equiv f_1(x, y) \\ \dot{y} = x + y[\mu - (x^2 + y^2)] \equiv f_2(x, y) \end{cases}$$

引入$x = r\cos\theta$，$y = r\sin\theta$ 将典型方程化成极坐标形式

$$\begin{cases} \dot{r} = \dfrac{x\dot{x} + y\dot{y}}{r} = r(\mu - r^2) \equiv f(r, \mu) \\ \dot{\theta} = \dfrac{x\dot{y} - \dot{x}y}{r^2} = 1 \end{cases}$$

这里的第二个方程表明轨线以固定的角速度$\dot{\theta} = 1$旋转，而第一个方程与叉型分岔的典型方程相同，说明当$\mu < 0$时不动点为$r = 0$，$\mu > 0$时不动点为$r = 0$，$\sqrt{\mu}$。当$\mu < 0$变到$\mu > 0$，不动点$r = 0$由稳定变为不稳定，而分岔出来的新解是轨线在环$r = \sqrt{\mu}$上旋转，角速度是 1，这种解叫极限环。这种失稳后出现极限环的分岔叫霍普夫分岔。可用图 6.41(a) 来表示，霍普夫分岔点是$x = 0$，$y = 0$，$\mu = 0$。它是超临界霍普夫分岔。

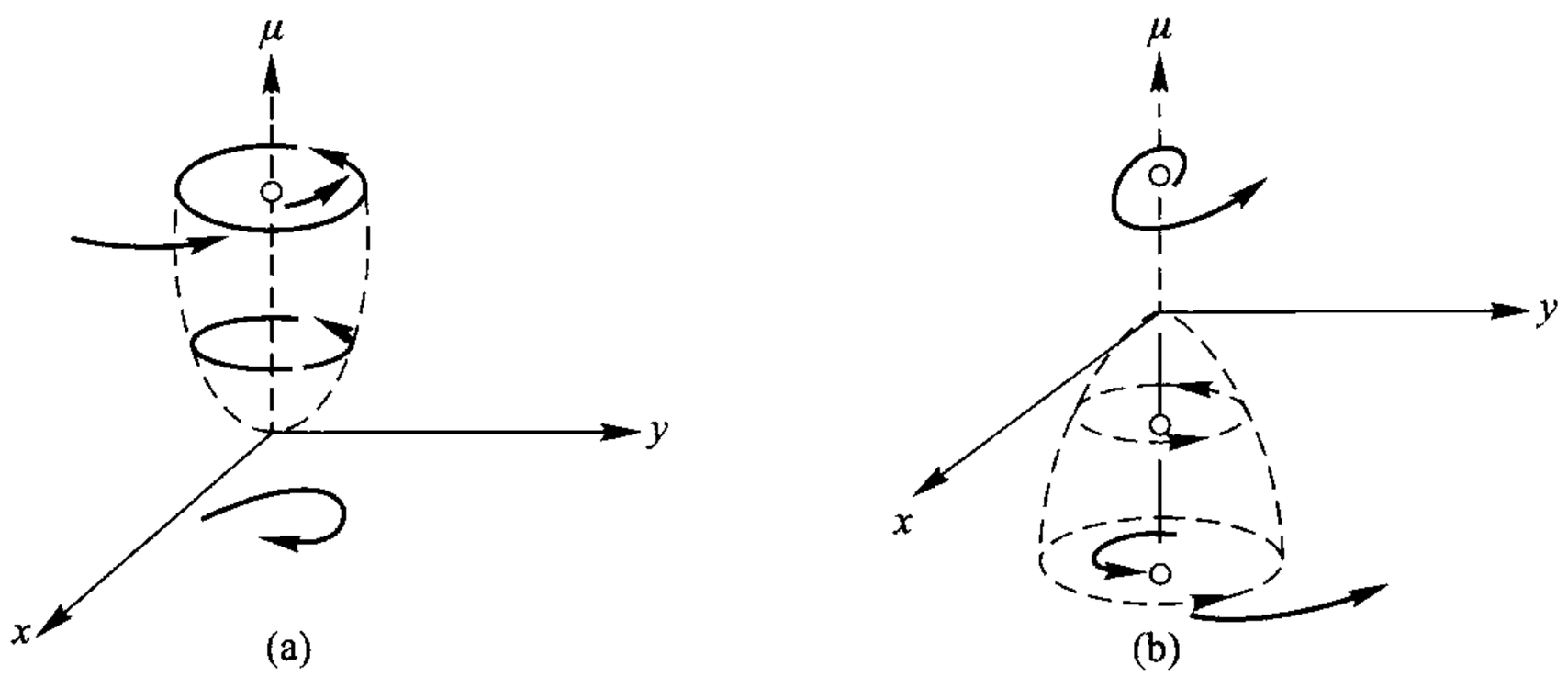


图 6.41 霍普夫分岔

如果典型方程为

$$\begin{cases} \dot{x} = -y + x[\mu + (x^2 + y^2)] \equiv f_1(x, y) \\ \dot{y} = x + y[\mu + (x^2 + y^2)] \equiv f_2(x, y) \end{cases}$$

则可用图 6.41(b) 来表示其分岔的形式，它是亚临界霍普夫分岔。

3. 鞍–结分岔 (切分岔)

典型方程为

$$\begin{cases} \dot{x} = \mu + x^2 \\ \dot{y} = \pm y \end{cases}$$

令$\dot{x} = 0$，$\dot{y} = 0$得不动点为$x = \pm\sqrt{-\mu}$，$y = 0$，考虑到x是实数，故不动点要求$\mu < 0$才能存在。此时雅可比矩阵为

$$J = \begin{pmatrix} 2x & 0 \\ 0 & \pm 1 \end{pmatrix}_{\substack{x=\pm\sqrt{-\mu} \\ y=0}} = \begin{pmatrix} \pm 2\sqrt{-\mu} & 0 \\ 0 & \pm 1 \end{pmatrix}$$

得到本征值为

$$\lambda_1 = \pm 2\sqrt{-\mu}, \qquad \lambda_2 = \pm 1$$

当两个本征值同号时，不动点是结点，当两个本征值反号时，不动点是鞍点。当μ由负变到零时，结点和鞍点合二为一，成为一个点$x = 0$，$y = 0$。当$\mu > 0$时，不存在不动点。这就叫鞍结点分岔。分岔点$x = 0$，$y = 0$，$\mu = 0$满足条件

$$\begin{vmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{vmatrix}_{\substack{x=0 \\ y=0 \\ \mu=0}} = \begin{vmatrix} 2x & 0 \\ 0 & \pm 1 \end{vmatrix}_{\substack{x=0 \\ y=0 \\ \mu=0}} = 0$$

$$\begin{vmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial \alpha} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial \alpha} \end{vmatrix}_{\substack{x=0 \\ y=0 \\ \mu=0}} = \begin{vmatrix} 2x & 1 \\ 0 & 0 \end{vmatrix}_{\substack{x=0 \\ y=0 \\ \mu=0}} = 0$$

鞍结点分岔可用图 6.42 表示：

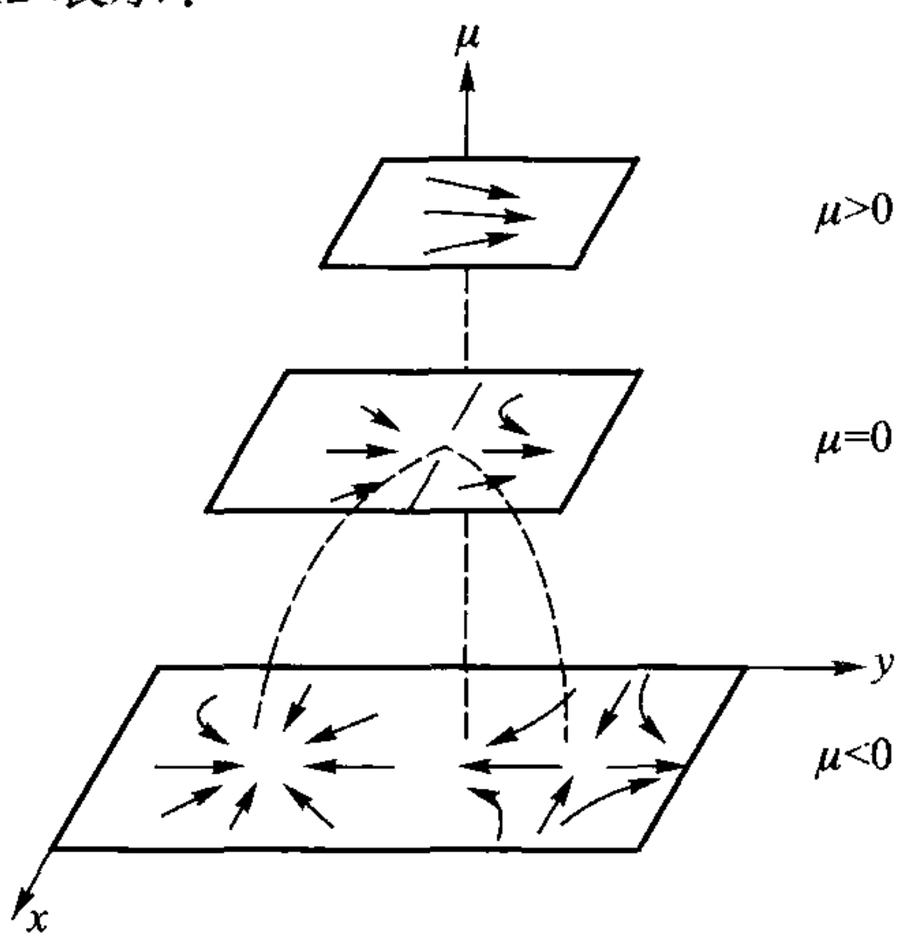


图 6.42 鞍–结分岔 (切分岔)

4. 跨临界分岔

典型方程为

$$\dot{x} = \mu x - x^2$$

其中 μ, x 为实数。无论 $\mu < 0$ 或 $\mu > 0$, 都存在两个不动点 $x = 0$, μ。

在不动点 $x = 0$, 本征值是 $\lambda = \mu$, 当 μ 由负变到正, 不动点 $x = 0$ 由稳定变成不稳定。

在不动点 $x = \mu$, 本征值是 $\lambda = -\mu$, 当 μ 由负变到正, 不动点 $x = \mu$ 由不稳定变成稳定。

可见解的稳定性只是通过分岔点发生了交换, 形态并没有变化。这种分岔就叫跨临界分岔, 如图 6.43 所示。

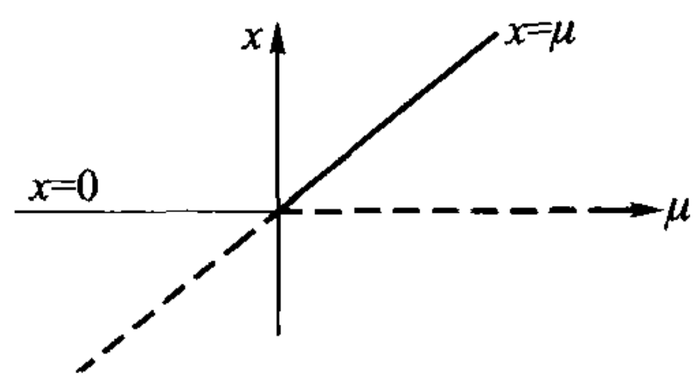


图 6.43 跨临界分岔

从分岔观点来看一下前面的例子。对无阻尼无驱动的达芬方程, 增加控制参数μ后成为

$$\ddot{x} = \mu x - x^3$$

写成方程组形式

$$\begin{cases} \dot{x} = y \\ \dot{y} = \mu x - x^3 \end{cases}$$

x, μ是实数, 无论μ为何值, $x = 0$, $y = 0$都是不动点, 只有$\mu > 0$时, $x = \pm\sqrt{\mu}$, $y = 0$ 才是不动点。

在不动点$x = 0$, $y = 0$雅可比矩阵是

$$J = \begin{pmatrix} 0 & 1 \\ \mu & 0 \end{pmatrix}_{\substack{x=0 \\ y=0}}$$

本征值是$\lambda = \pm\sqrt{\mu}$, 当μ由负变到正, 不动点$x = 0$, $y = 0$由稳定焦点变成不稳定结点。分岔点是$x = 0$, $y = 0$, $\mu = 0$。

在不动点$x = \pm\sqrt{\mu}$雅可比矩阵是

$$J = \begin{pmatrix} 0 & 1 \\ \mu - 3x^2 & 0 \end{pmatrix}_{\substack{x=\pm\sqrt{\mu} \\ y=0}} = \begin{pmatrix} 0 & 1 \\ -2\mu & 0 \end{pmatrix}$$

本征值是$\lambda = \pm\sqrt{-2\mu}$,由于$\mu > 0$,所以本征值是一对共轭虚数,对应的两个不动点都是稳定焦点。

综上所述,当μ为负,只有一个不动点$x = 0$, $y = 0$,它是稳定焦点;$\mu = 0$,这个不动点成为不稳定结点;当$\mu > 0$,分岔出两个稳定焦点$x = \pm\sqrt{\mu}$,所以这是叉型分岔。分岔点是$x = 0$, $y = 0$, $\mu = 0$。

达芬方程的势能为 $V = 0.5x^4 + \mu x^2$,画出 $\mu = -1$, 0, 1 的势能曲线如图 6.44,可以看出,$x = 0$ 点从势能最低点变成一个极大值,所以由一个稳定的平衡点变成不稳定平衡点,而 $x = \pm\sqrt{\mu} = \pm 1$ 则在 $\mu > 0$ 时成为两个新势能极小值,所以形成了两个新稳定平衡点。这些分析与达芬方程一节中的分析是完全一致的。

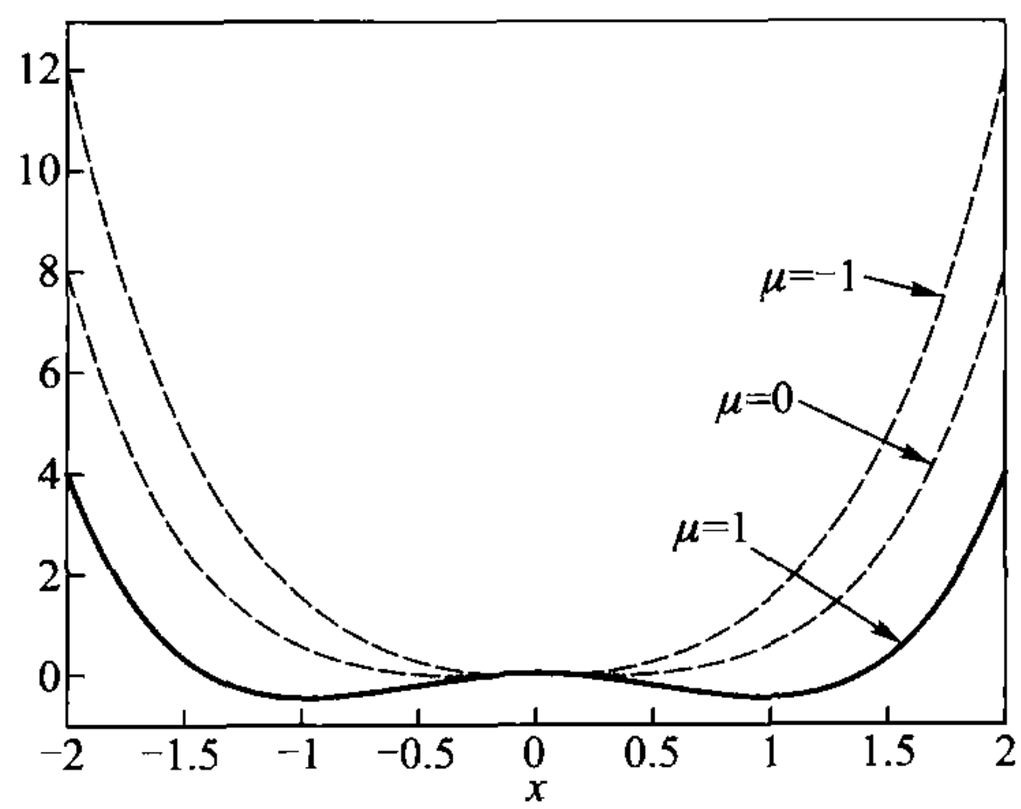


图 6.44 达芬方程的势能曲线

再看 VDP 方程,写成带控制参数的形式如下

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0 \quad (|\mu| < 2)$$

令$\dot{x} = y$,写成方程组为

$$\begin{cases} \dot{x} = y \\ \dot{y} = -x + \mu y - \mu x^2 y \end{cases}$$

令$\dot{x} = 0$, $\dot{y} = 0$得不动点为$x = 0$, $y = 0$,它的雅可比矩阵为

$$J = \begin{pmatrix} 0 & 1 \\ -1 & \mu \end{pmatrix}$$

本征方程与本征值为

$$\lambda^2 - \mu\lambda + 1 = 0$$

$$\lambda = \frac{1}{2}(\mu \pm \sqrt{\mu^2 - 4})$$

由于$|\mu| < 2$，因此有

当$\mu < 0$，λ为共轭复根，其实部为负，所以$x = 0$，$y = 0$稳定焦点；

当$\mu > 0$，λ为共轭复根，其实部为正，所以$x = 0$，$y = 0$不稳定焦点。

不动点在$\mu = 0$处也满足分岔点条件

$$\frac{\mathrm{d}}{\mathrm{d}\mu}\mathrm{Re}(\lambda(\mu)) = \frac{1}{2} \neq 0$$

所以不动点$x = 0$，$y = 0$是一个霍普夫分岔点，当μ由负变正时，分岔出一个稳定的极限环。这正是前面分析VDP方程时看到的现象。

§6.4 洛伦茨方程——奇怪吸引子

世界上对常微分方程研究最广泛的领域之一是洛伦茨吸引子，它也是近代混沌学的经典问题，是第一个实际观察的奇怪吸引子。它最先是由美国MIT大学的数学家与气象学家洛伦茨（Edward Lorenz）在 1963 年提出的，他的主要研究方向用地球大气的液体流动模型作长期天气预报。有一本极好的研究洛伦茨方程的专著，就是 C.Sparrow 的《The Lorenz Equations,Chaos,and Strange Attractors》。

洛伦茨方程是由纳维–斯托克斯（Navier-Stokes）方程组化简而来。纳维–斯托克斯方程组是牛顿定律在流体力学中一种表述方式，其中含有速度、温度、密度等物理量，它们都是时间与位置的函数，在实际问题中，它是很难用解析方法求解的，所以它很适合用计算机作数值解。一个特例是瑞利–贝纳尔（Rayleigh-Bénard）问题，这是放置在一个容器中的液体，顶层与底层有不同的温度。当温差逐渐变大时，稳定的对流现象会趋于混沌。在近 50 年前洛伦茨研究这个问题时，受计算机计算能力的限制，他将纳维–斯托克斯方程大大简化，成为如下形式

$$\begin{cases} \dot{x} = -\sigma x + \sigma y \\ \dot{y} = rx - y - xz \\ \dot{z} = xy - bz \end{cases} \tag{6.20}$$

这就是著名的洛伦茨方程。洛伦茨的工作结果最初在1963年发表，论文题目为《Deterministic Nonperiodic Flow》，发表在《Journal of the Atmospheric Sciences》杂志上，这是实际观察到的第一个奇怪吸引子。洛伦茨方程中 x 是对流的强度，y 是向上流动和向下流动的液体单位元之间的温度差，z 是垂直方向的温度梯度。式中有三个参数，σ是普兰特（Prandtl）数、r 是规范化瑞利（Rayleigh）数，b 是描述区域几何形状的参数。三个参数可任取大于0的数值。常用的组合是，$\sigma = 10$，$\beta = 8/3$，而令ρ取不同数值。如$\rho = 28$时就会出现混沌现象，有奇异

吸引子出现。

洛伦茨系统（6.20）的相空间随时间的变化率为

$$\frac{1}{V}\frac{\mathrm{d}\boldsymbol{V}}{\mathrm{d}t}=\frac{\partial\dot{x}}{\partial x}+\frac{\partial\dot{y}}{\partial y}+\frac{\partial\dot{z}}{\partial z}=-(\sigma+b+1)\quad(\sigma>0,b>1)\tag{6.21}$$

可见洛伦茨系统为耗散系统，对上式积分得

$$V(t)=V(0)\mathrm{e}^{-(\sigma+b+1)t}$$

可见，系统的相体积元是以指数收缩，这是耗散系统。

下面讨论系统的不动点与稳定性。

令$\dot{x}=\dot{y}=\dot{z}=0$得

$$\begin{cases}x=y\\rx-y-xz=0\\xy-bz=0\end{cases}$$

将其中的第一式代入后二式，得

$$\begin{cases}x=y\\x(r-1-z)=0\\x^2=bz\end{cases}$$

当$x=0$，有$y=0$，$z=0$；当$z=r-1$，有$x=y=\pm\sqrt{b(r-1)}$ 归纳起来，当$0<r<1$时，只有一个不动点$O(x=0,\ y=0,\ z=0)$；当$r>1$时，有三个不动点：

$$\begin{cases}O(x=0,y=0,z=0)\\C_1(x=y=\sqrt{b(r-1)},z=r-1)\\C_2(x=y=-\sqrt{b(r-1)},z=r-1)\end{cases}\tag{6.22}$$

O点的雅可比矩阵为

$$\begin{pmatrix}-\sigma & \sigma & 0\\ r & -1 & 0\\ 0 & 0 & -b\end{pmatrix}$$

相应的本征方程为

$$(b+\lambda)[\lambda^2+(\sigma+1)\lambda+\sigma(1-r)]=0$$

解出本征值为

$$\lambda_1=-b$$

$$\lambda_{2,3}=\frac{1}{2}[-(\sigma+1)\pm\sqrt{(\sigma+1)^2-4\sigma(1-r)}]$$

所以，当$0<r<1$，$\lambda_{1,2,3}$均为负实数，故O点为稳定结点。

当$r>1$，λ_1仍为负实数，但$\lambda_{2,3}$为一正一负，故O点为不稳定结点。

当$r=1$，三个根之中必有一个为零，O点处于临界状态，也就是有一个根沿实轴穿过虚轴发生一次叉型分岔。

在不动点C_1雅可比矩阵为

$$\begin{pmatrix} -\sigma & \sigma & 0 \\ 1 & -1 & -\sqrt{b(r-1)} \\ \sqrt{b(r-1)} & \sqrt{b(r-1)} & -b \end{pmatrix}$$

将上述矩阵中的根号取反号即得不动点C_2的雅可比矩阵。它们的本征方程都是

$$\lambda^3+(\sigma+b+1)\lambda^2+(r+\sigma)b\lambda+2b\sigma(r-1)=0 \tag{6.23}$$

按照实系数一元三次方程的解法，令$\lambda=y-(\sigma+b+1)/3$将方程化成

$$y^3+[b(\sigma+r)-(\sigma+b+1)^2/3]y+2(\sigma+b+1)^3/27-b(\sigma+r)(\sigma+b+1)/3+2\sigma b(r-1)=0$$

对形式为$y^3+py+q=0$的方程，其判别式为 $\Delta=q^2/4+p^3/27$：

当$\Delta>0$时，有一个实根和一对共轭复根；

当$\Delta=0$时，有三个实根，一个是$-2\sqrt[3]{q/2}$，另两个相等，都是$\sqrt[3]{q/2}$；

当$\Delta<0$时，有三个不同的实根。

由此可知$\delta=0$是分界，将$\sigma=10$，$b=8/3$代入得到

$$p=8r/3-961/27,\quad q=10402/729+1112r/27$$

$$\delta=-393440/243+487840/729r+865904/2187r^2+512/729r^3$$

用指令solve解得$\Delta=0$的一个实根为$r_0=1.3456171792329563180367087737 08$。进一步还可求出$\lambda_1=-11.0880$，$\lambda_{2,3}=-1.2894$。因而$r=r_0$时，$C_1$，$C_2$都是稳定结点。当$r<r_0$时，有$\Delta<0$，同时可求出三个实根$\lambda_{1,2,3}$均为负实根，所以$C_1$，$C_2$也是稳定结点。对于$r>r_0$，可如下讨论，这时的实根仍为负实数，另外两个是实部为负的共轭复数，C_1，C_2成为稳定焦点，设共轭复根为

$$\beta=\beta_1+\mathrm{i}\beta_2,\quad \bar{\beta}=\beta_1-\mathrm{i}\beta_2$$

本征方程可写为

$$(\lambda-\alpha)(\lambda-\beta)(\lambda-\bar{\beta})=0$$

展开得

$$\lambda^3-(2\beta_1+\alpha)\lambda^2+(|\beta|^2+2\beta_1\alpha)\lambda-|\beta|^2\alpha=0$$

如果r继续增大到r_c，使得实部$\beta_1=0$，则共轭复根将穿过虚轴，发生霍普夫分岔。下面求分岔点对应的r_c。

将$\beta_1=0$代入上式得

$$\lambda^3-\alpha\lambda^2+\beta_2^2\lambda-\beta_2^2\alpha=0$$

上式表明，(λ^2项的系数) × (λ项的系数)=常数项。

与 (6.23) 式相比较，则上式可表示为

$$(\sigma+b+1)(\sigma+r)b=2\sigma b(r-1)$$

取$\sigma=10$，$b=8/3$解出

$$r_c=24.74$$

即为所求。还可以证明在这点有 $\dfrac{\mathrm{d}\beta_1(r_c)}{\mathrm{d}r}>0$，即满足霍普夫分岔点条件。

归纳起来，有以下几种情况：

(1) $0\leqslant r<1$，平衡态 O 是稳定的结点。图 6.45 是 $r=0.5$，初值为 $[20,20,30]$ 的情况，图中画出 $x-y-z$ 三维图形以及 $x-y$，$x-z$，$y-z$ 的二维图，轨线在任何截面上最终都会落到 O 点。一般情况可以用图 6.46 来表示。

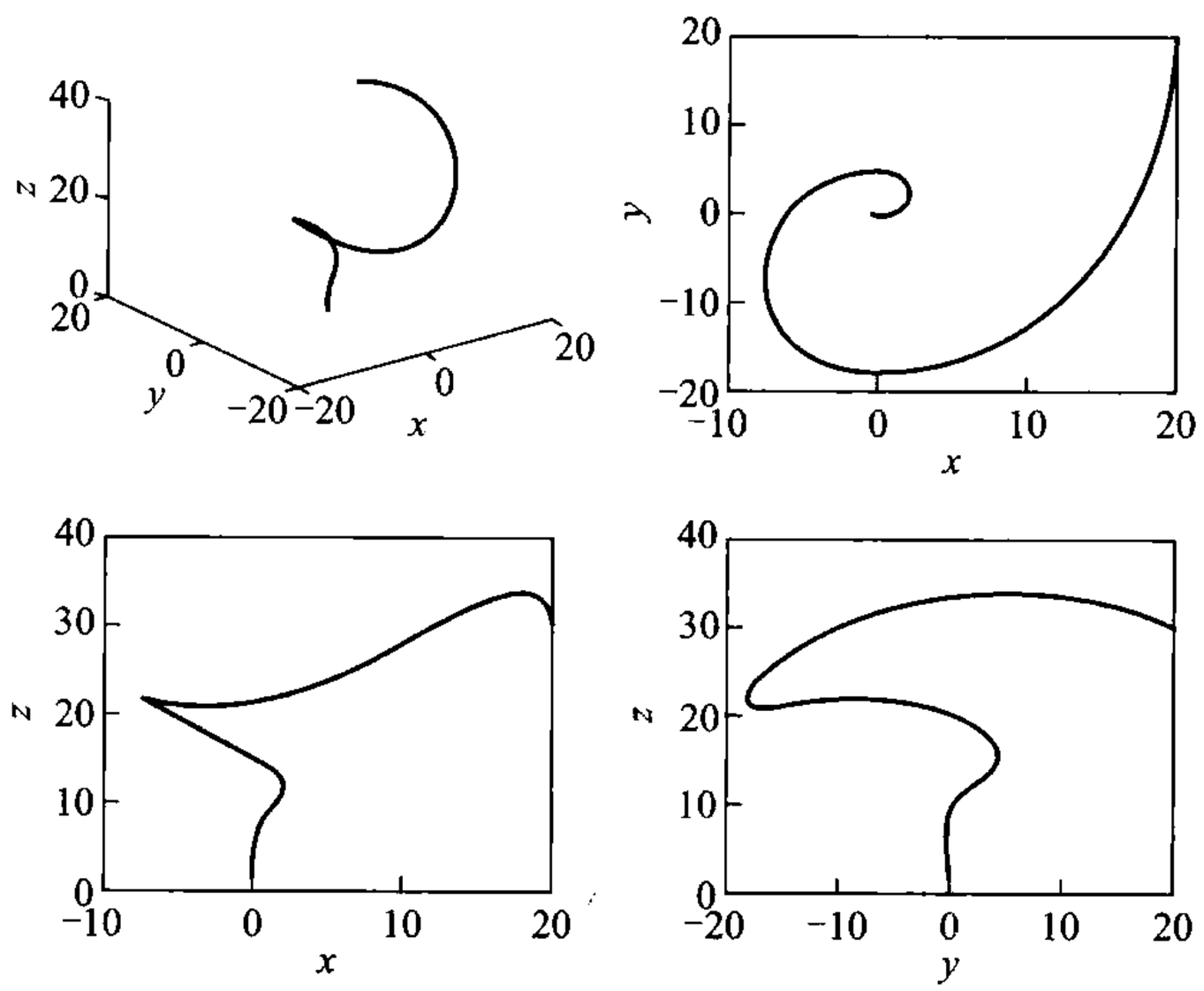


图 6.45

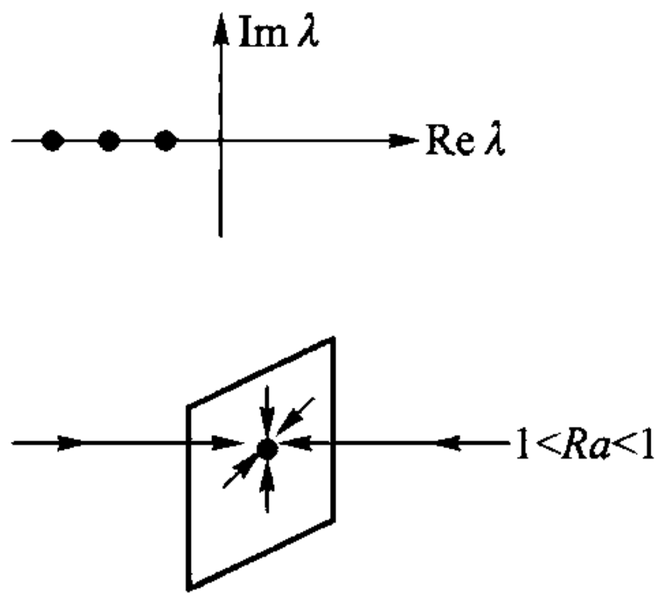


图 6.46

(2) $1 < r < r_0 = 1.346$，O点失稳变成鞍点，C_1，C_2点是新生出来的稳定结点。图 6.47 是$r = 1.1$，初值为$[0.21, 0.01, 0.01]$的情况，轨线从O点附近流向鞍点O以后又流出，最后落在稳定结点之一$C_1(0.51669, 0.51637, 0.099984)$（该值是解方程所得）。轨线的起点用星号*标出。一般的情况可用图 6.48 表示。

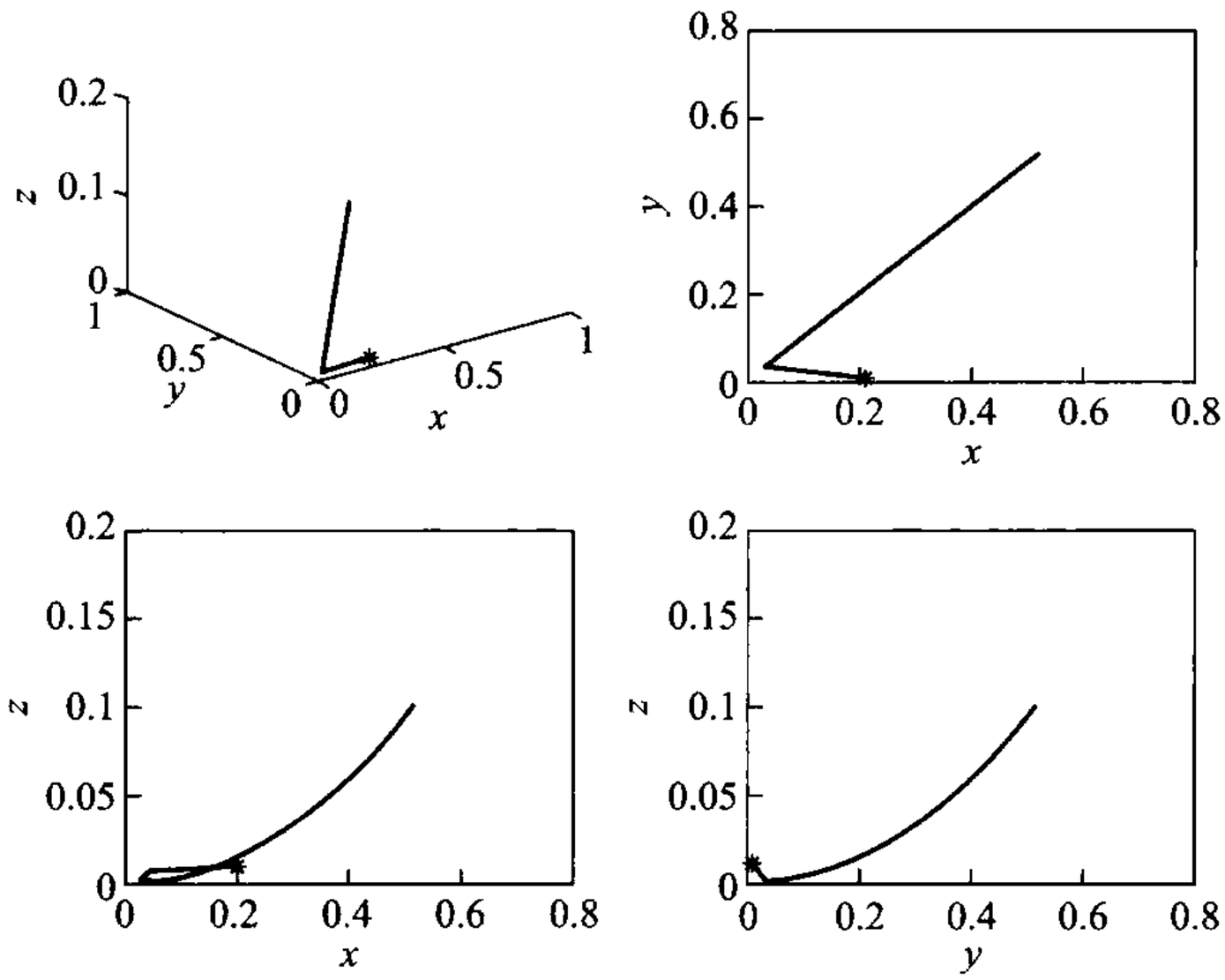


图 6.47

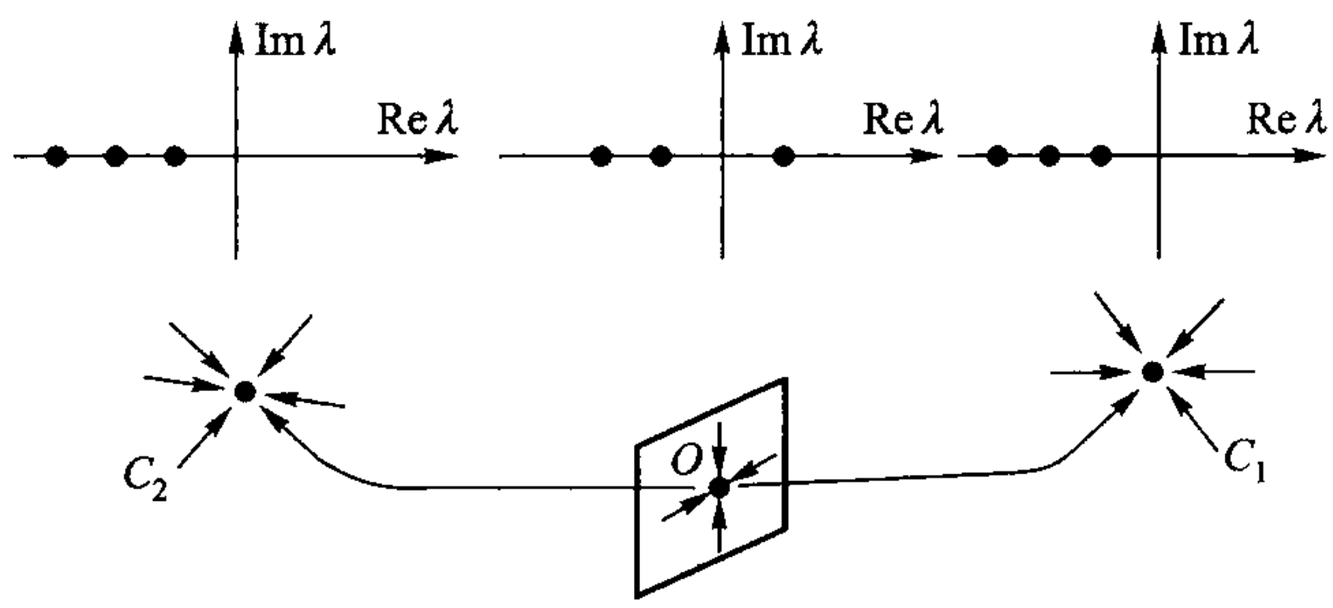


图 6.48

(3) $r_0 < r < 13.92656$，O点依然是鞍点，但C_1，C_2点变成了稳定焦点。图 6.49 是$r = 10$，初值为$[4.81, 1.21, 5.32]$的情况，轨线开始稍微流向鞍点O以后即流出，最后绕稳定焦点$C_1(4.8987, 4.899, 9.0001)$旋转并最后落在其上（该值是解方程所得）。轨线的起点用星号*标出。参数13.92656不能由解析方法求出，是由数值计算所得。一般的情况可用图 6.50 表示。

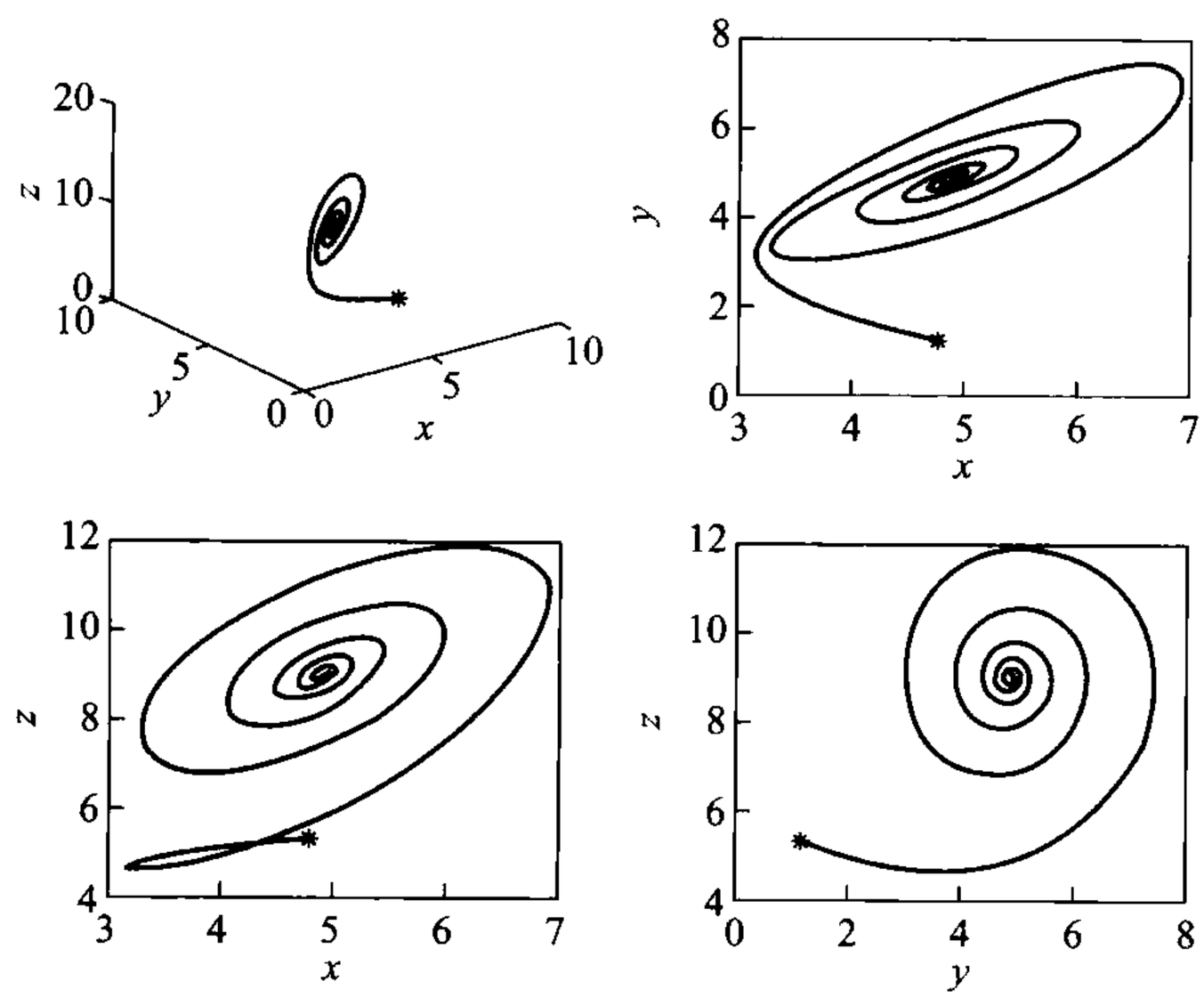


图 6.49

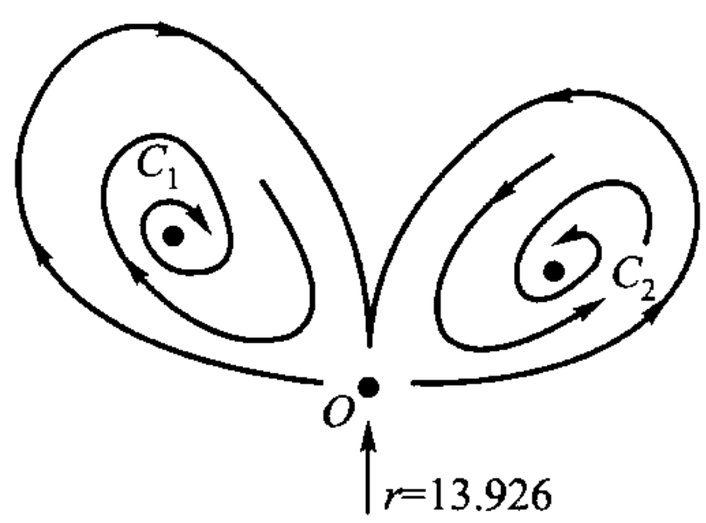


图 6.50

(4) $r = 13.92656$，是C_1，C_2从稳定焦点变成不稳定极限环的分界值。取初值为[0.01,0.00,0.00]，轨线由原点出来绕C_1一周后回到原点附近再出来绕C_2旋转，直至落到C_2上，如图 6.51 所示。将初值改为[0.01,0.00,0.00]，则轨线运动与此相反，先绕C_2一周，再绕C_1旋转并落到C_1。图 6.52 是一般情形的示意图。

(5) $13.92656 < r < r_c = 24.74$，这时$C_1$，$C_2$形成两个不稳定的极限环。取$r = 22.7$，初值为 $[-7.0,-7.1,3.30]$，在图 6.53 可以看到轨线从 C_2 点出发，绕 C_1 旋转若干次，但并不会落在 C_1 点，然后又回到 C_2 附近旋转。如果增加计算时间，可以看到更多的圈数。这里为了能看得更清楚，只画了两圈。图 6.54 是一般

情形的示意图。

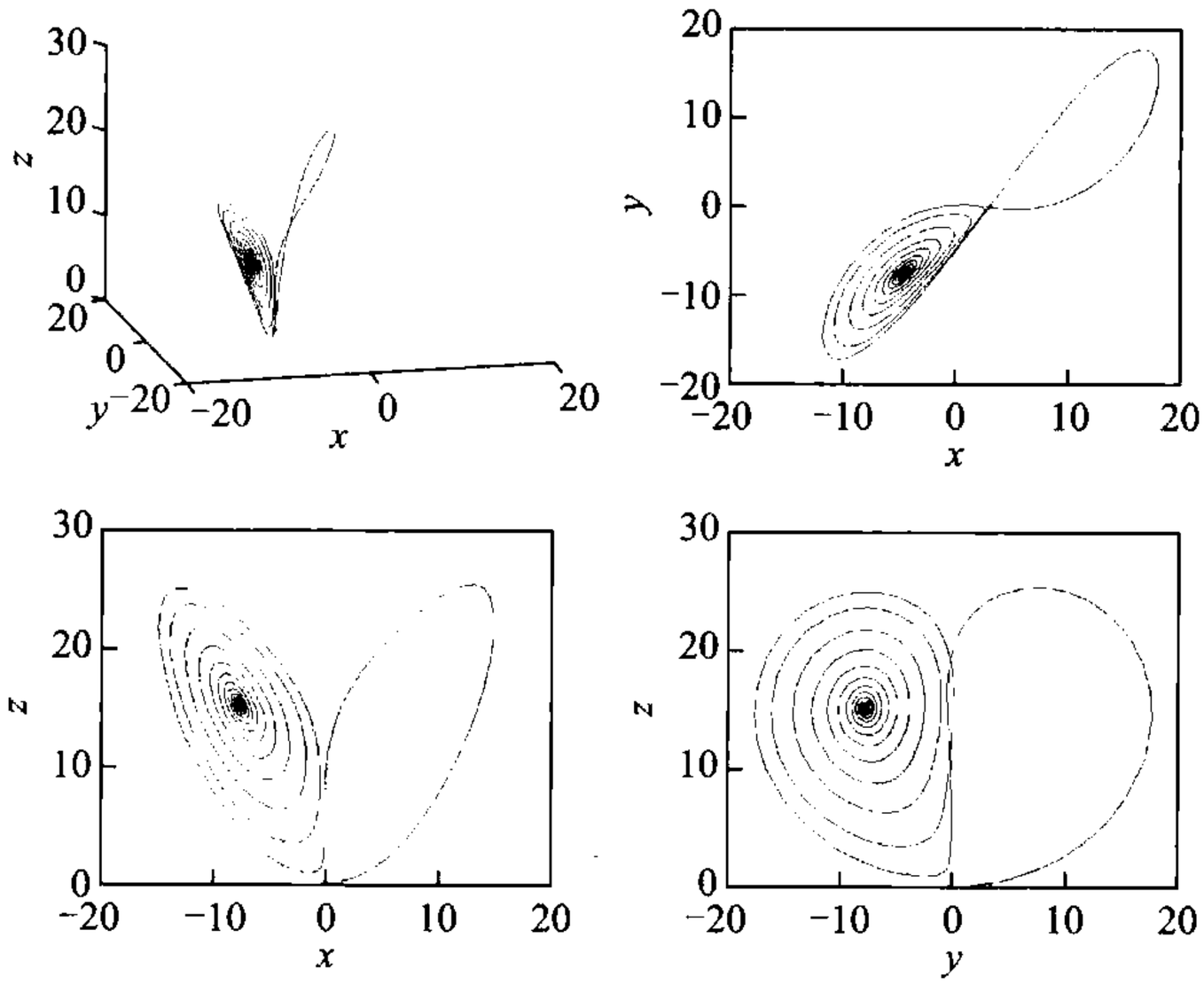


图 6.51

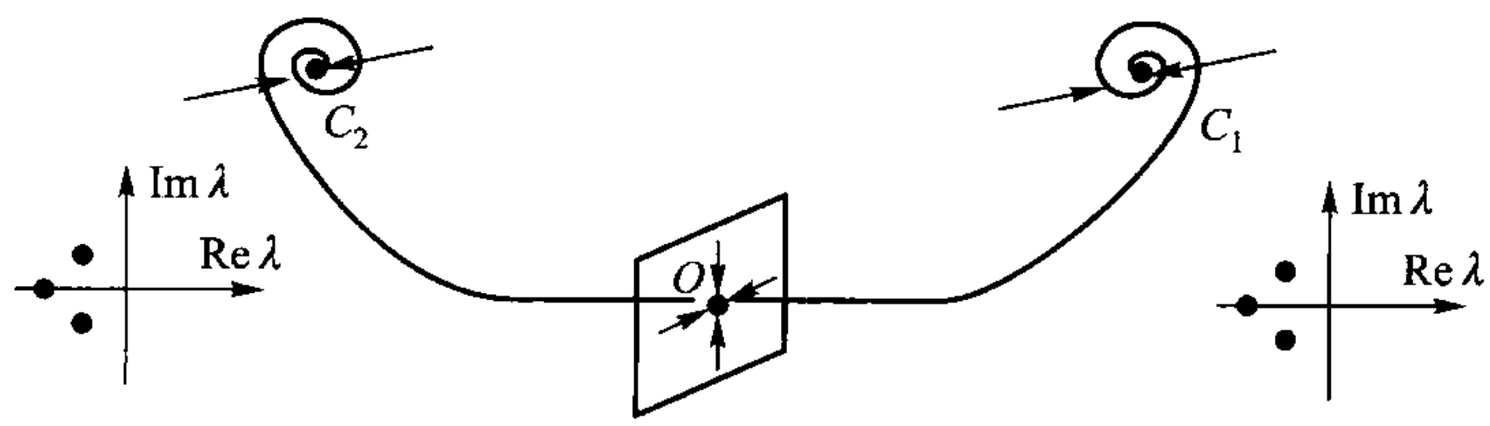


图 6.52

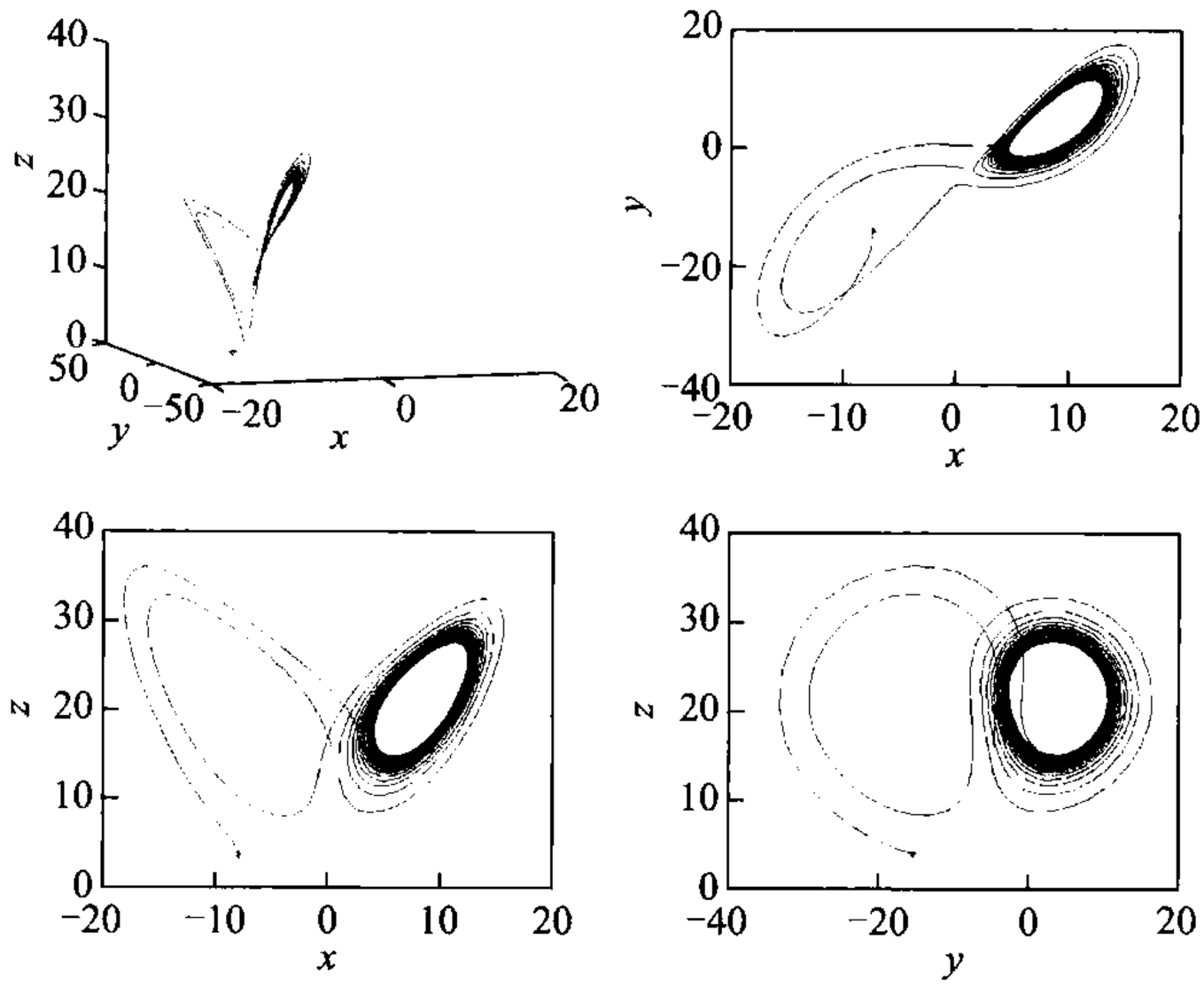


图 6.53

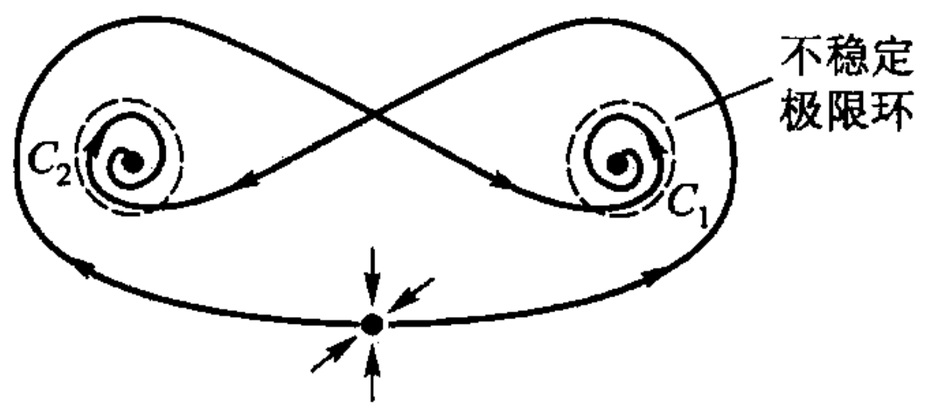


图 6.54

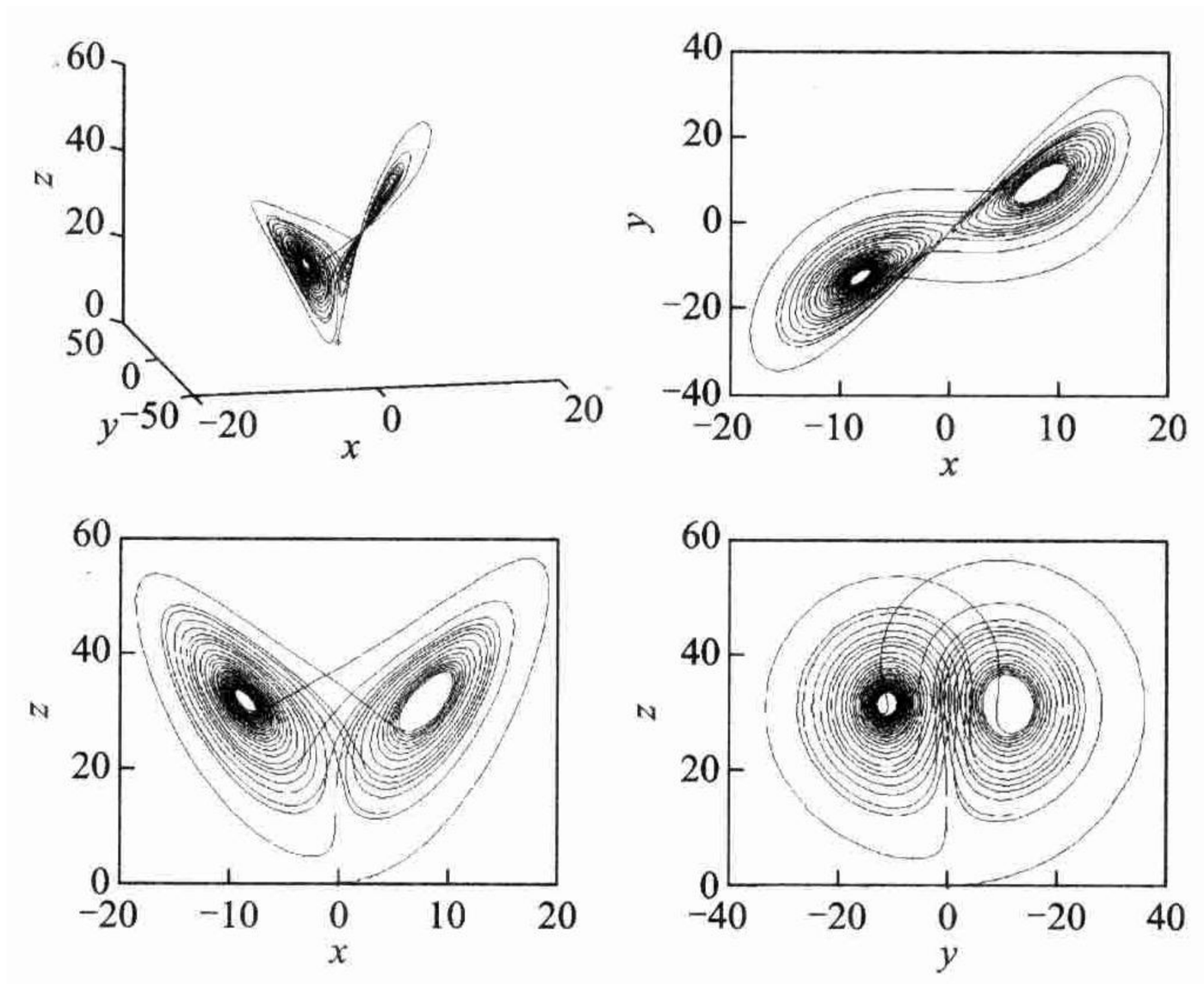


图 6.55 $r=28$

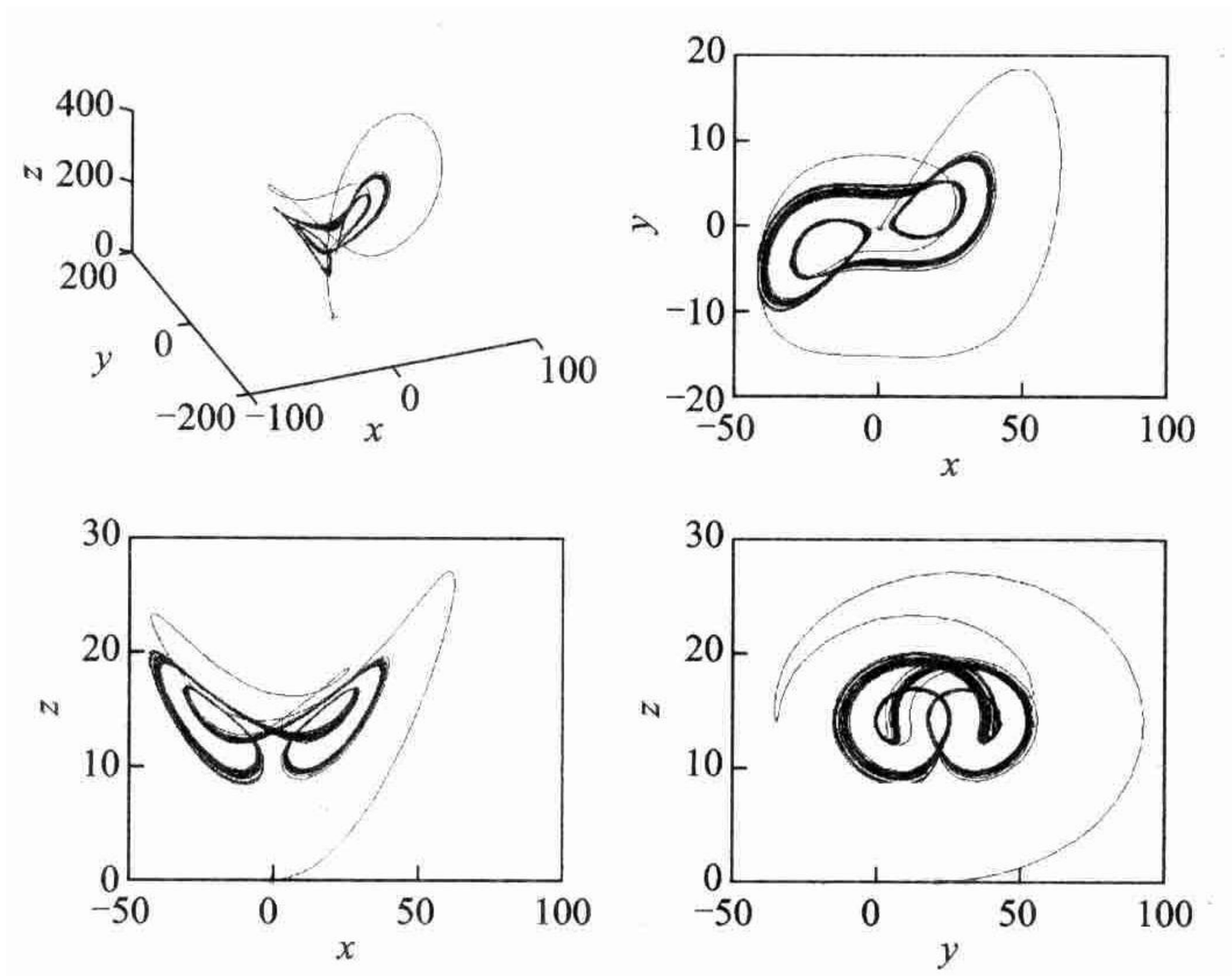


图 6.56 $r=146$

(6) $r_c < r$ 时，C_1，C_2 成为了鞍-焦点，系统出现了复杂的分岔序列。在某些 r 范围出现了奇怪吸引子，也就是混沌运动。在某些 r 范围又会出现周期解，如在 145 ~ 148，210 ~ 234，1000 等值都会出现极限环。下面画出了部分图形，其中的 r=28 (图 6.55)，r=146 (图 6.56)，r=225 (图 6.57)，r=1000 (图 6.58)，初值都取 [0.20,0.1,eps]。

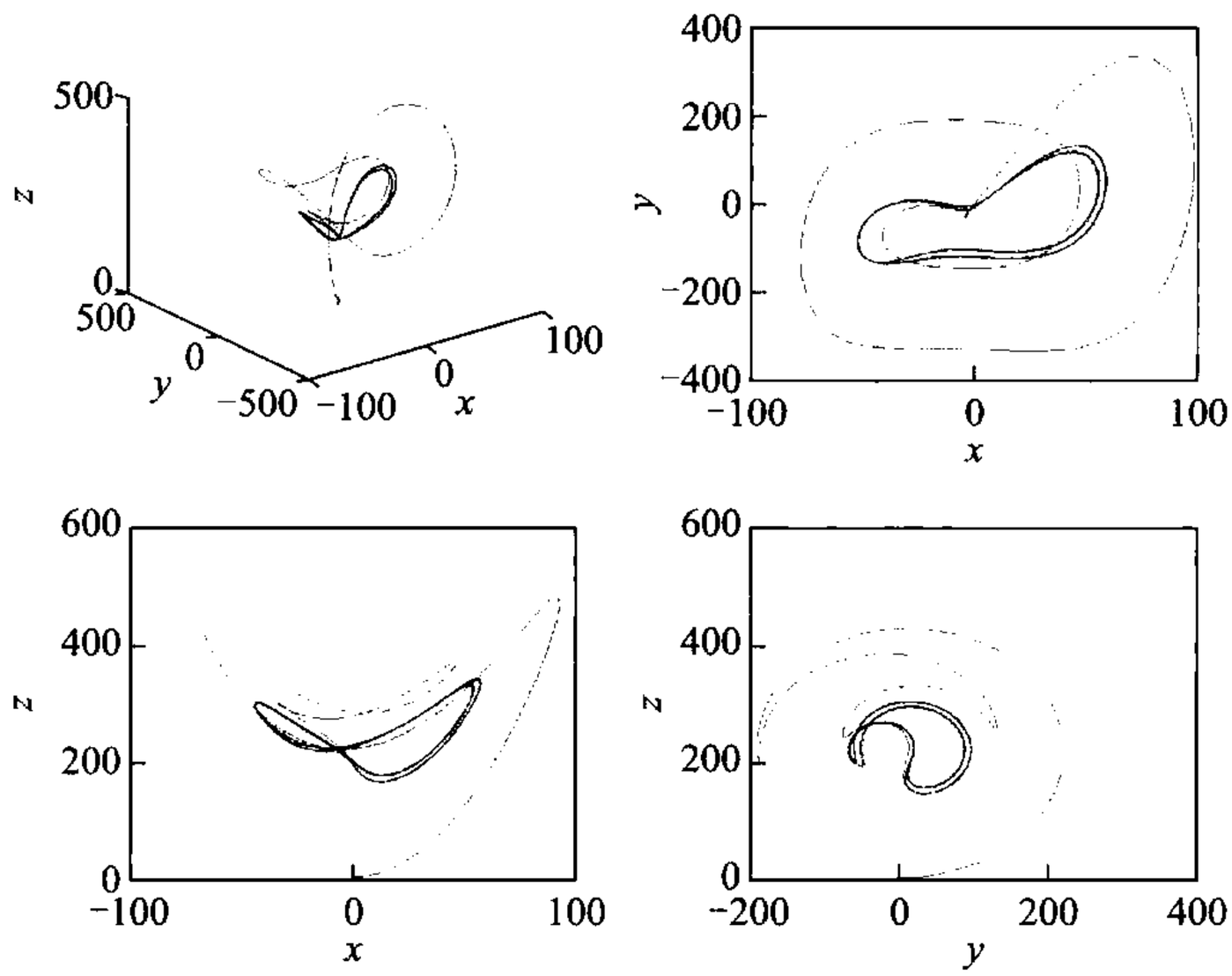


图 6.57 r=225

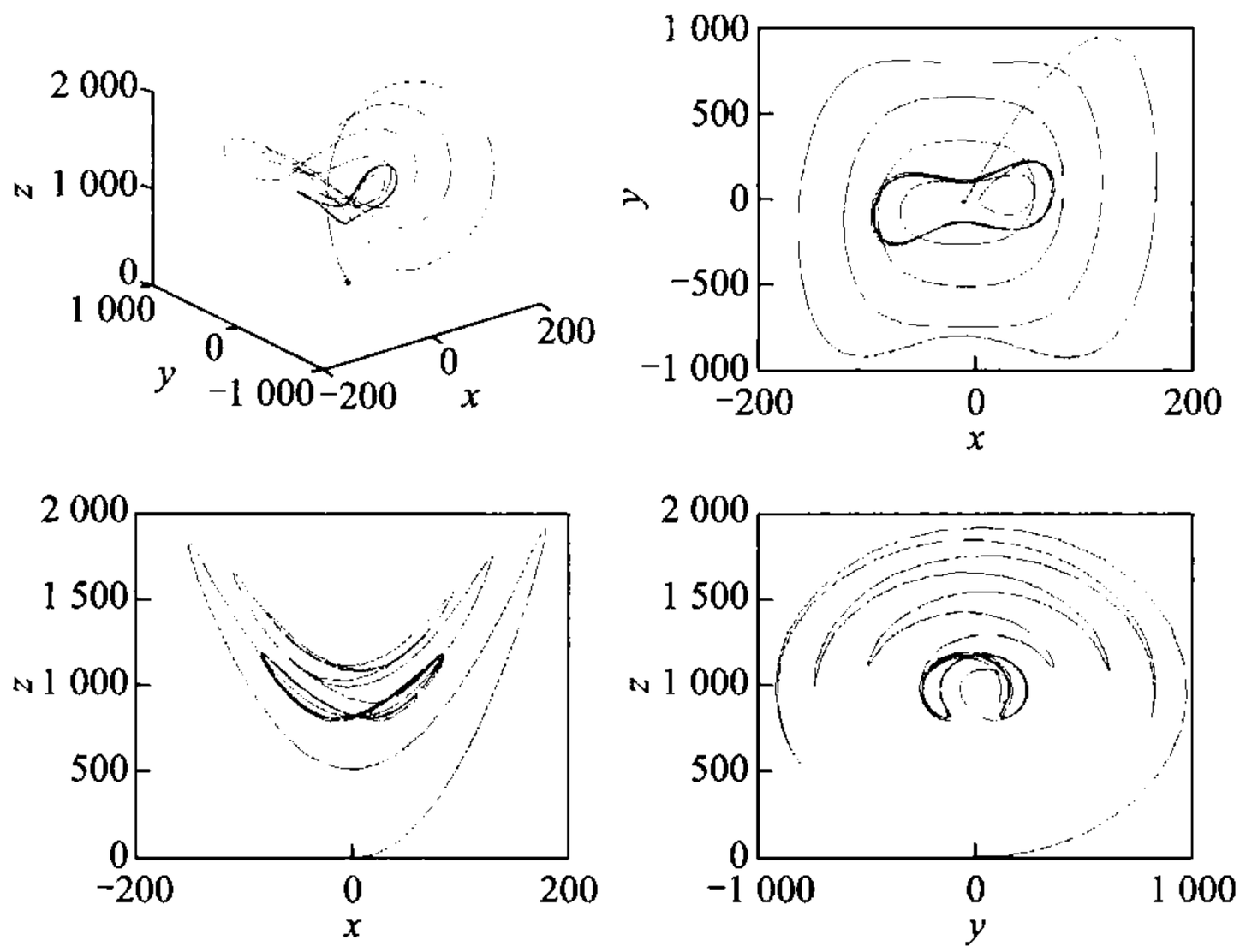


图 6.58 r=1000

混沌运动的形态表现为，从 O 出来的轨线绕一个不动点比如 C_1 旋转，然后又会回到 O 点，又出来绕 C_2 旋转，接着又回到 O 点。周而复始，永不停止。每次绕 C_1，C_2 方式不同，旋转的次数也有多有少，无法预测。这些轨线不会相交，因为一旦相交，就回到了曾经走过的某一点，其后的运动就应该在周期环上循环。

下面是作图程序，通过改变初值与r值，就可以研究洛伦茨系统的性质。

```
function lor
[t,y]=ode45( @lorfun,[0,31],[0.20,0.1,eps]);
subplot(2,2,1)
plot3(y(:,1),y(:,2),y(:,3),y(1,1),y(1,2),y(1,3),'*')
xlabel('x');   ylabel('y');   zlabel('z')
view(3)
subplot(2,2,2)
plot(y(:,1),y(:,2),y(1,1),y(1,2),'*')
xlabel('x');   ylabel('y');
subplot(2,2,3)
plot(y(:,1),y(:,3),y(1,1),y(1,3),'*')
xlabel('x');   ylabel('z');
subplot(2,2,4)
plot(y(:,2),y(:,3),y(1,2),y(1,3),'*')
xlabel('y');   ylabel('z');

function ydot=lorfun(t,y)
r=1000;
ydot=[  -10*y(1)+10*y(2);
          r*y(1)-y(1)*y(3)-y(2) ;
          y(1)*y(2)-8/3*y(3)];
```

§6.4.1 倍周期窗与费根鲍姆数

仔细地研究周期解出现的情形，可以看到倍周期分岔现象。例如$r = 100.795$，并选择初值为[20，11，115]，这个值已经在极限环上，得到的图 6.59。

可以看到，在一个周期之内，轨线是在$x > 0$的“正半空间”转了两圈，然后进到$x < 0$的“负半空间”转一圈。按照文献上的命名规则，这个解可以表示为“x^2y”或者表示为“$++-$”。当$r \approx 99.98$时，将出现周期加倍的类周期解$x^2yx^2y = (x^2y)^2$。下表是一个周期分岔序列的结果。对这个序列也可计算费根

鲍姆数δ，用更精确的方法计算出来的值是$\delta \approx 4.67$，在误差范围内，它是正确的。

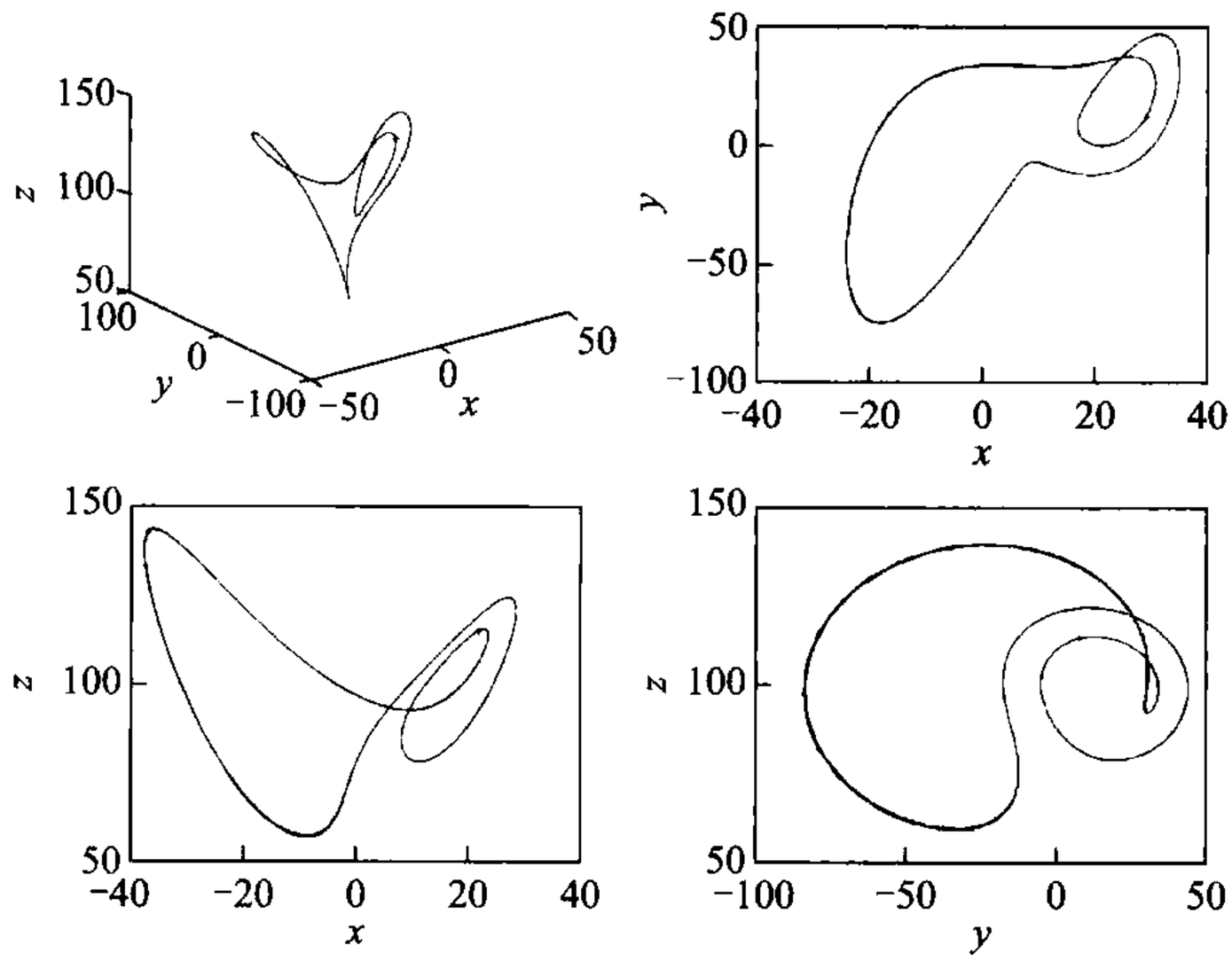


图 6.59 倍周期分岔现象

参数 r 值	周期解的类型
100.795	x^2y
99.98	$(x^2y)^2$
99.629	$(x^2y)^4$
99.547	$(x^2y)^8$
99.529	$(x^2y)^{16}$
99.5255	$(x^2y)^{32}$

§6.4.2 由阵发通向混沌

前面介绍过由倍周期分岔通向混沌的例子。洛伦茨系统具有另一种通向混沌的机制，就是经过阵发达到混沌。阵发原是湍流理论中用来描述流场中在层流的背景上湍流随机爆发的现象，表现为层流、湍流相交而使相应空间域随机的交替。在混沌理论中主要是借助于阵发这个概念来表示时间域中不规则行为和规则行为随机交替发生的现象。也就是说，当系统从有序向混沌转化时，在非平衡非线性的条件下，当某些参数的变化达到临界值时，系统的行为忽而周期，忽而混沌，在两者之间交替。当有关参数继续变化时，整个系统会由阵发混沌发展成混沌。

下面的图形演示了当初值为[1,0,0]，$r = 24, 24.3, 25$时，三个变量x, y, z随t变化的情形。图 6.60 保持了周期运动，图 6.61 已经有阵发出现，图 6.62 基本是混沌运动。

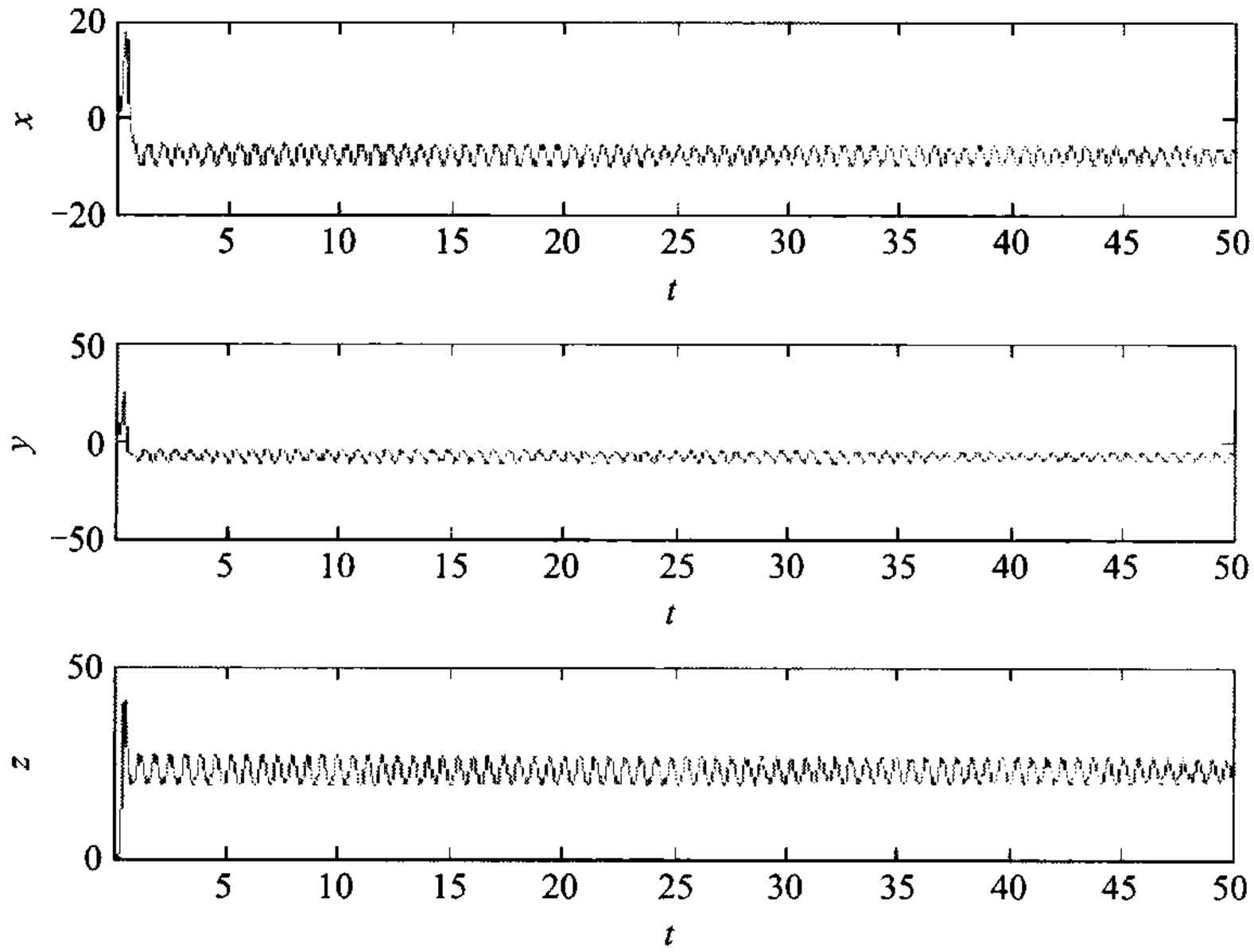


图 6.60 周期运动

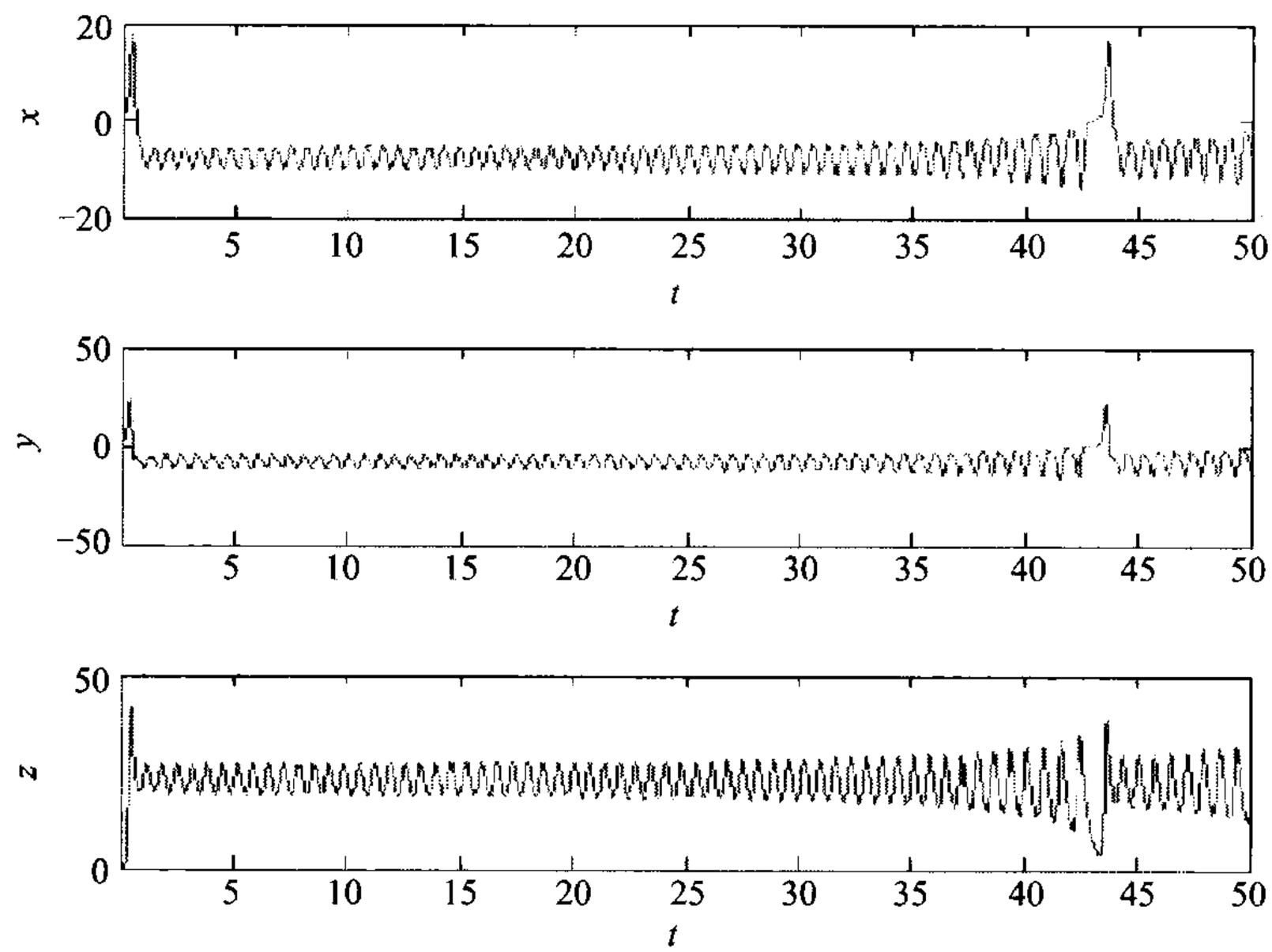


图 6.61 阵发运动

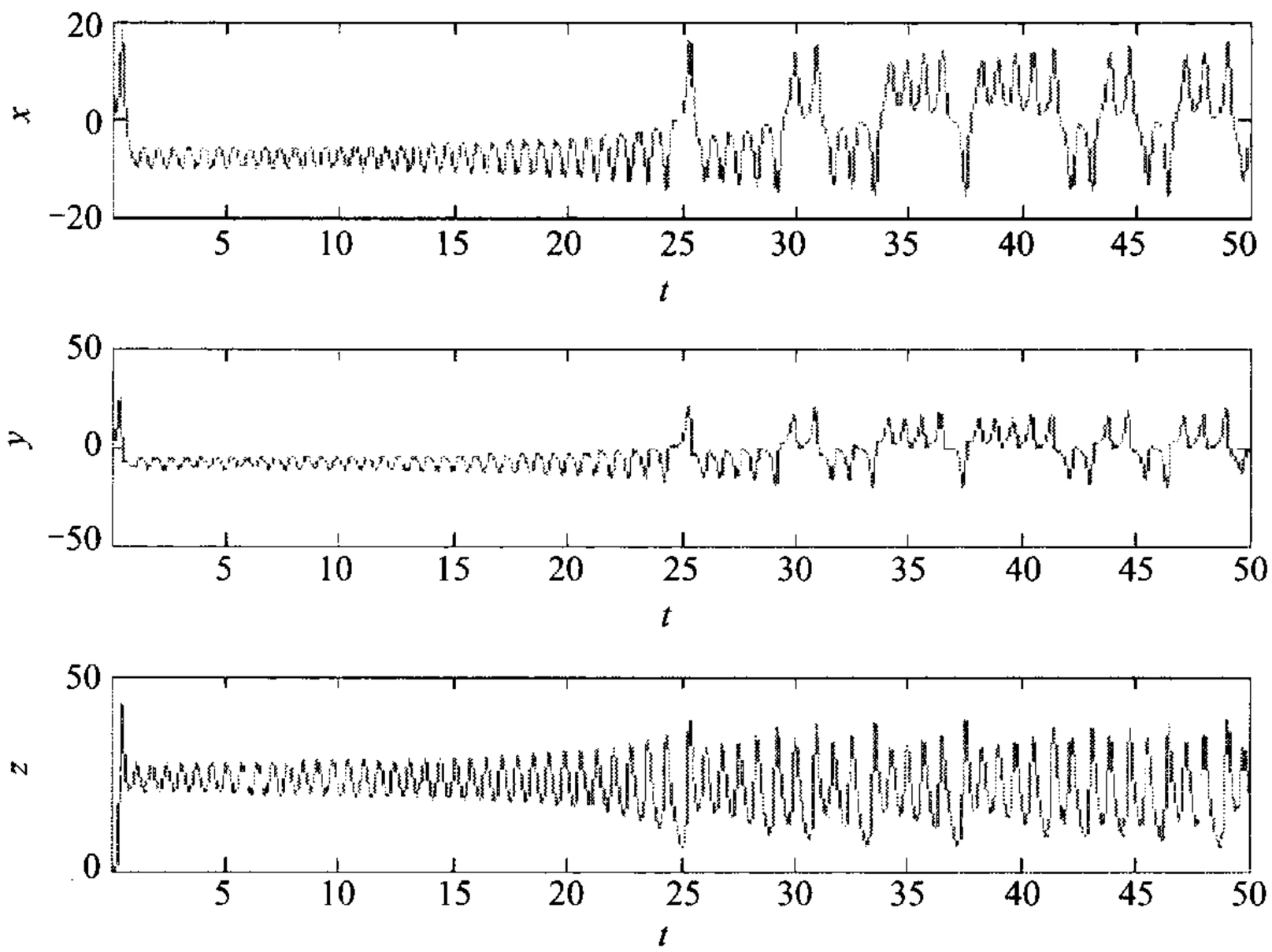


图 6.62 混沌运动

§6.4.3 庞加莱截面图

庞加莱截面图是研究混沌问题的好方法，但是洛伦茨系统与前面的几个例子有所不同，这里没有周期性的变量如驱动力等，所以不能用每个周期作一次庞加莱截面的方法。通常对这类问题作截面的方法是选取$x=0$(也可以选取$y=0$或者$z=0$)，记录下轨线穿过$x=0$平面的点，所得图形即为庞加莱截面图。图 6.63 是$r=25$，初值为[1，0，0]时，在$x=0$平面上得到的庞加莱截面，可以看到，虽然这时系统已经处于混沌状态，但是奇怪吸引子在截面上的形状却很有规律。我们不能预见系统在下一时刻会准确出现在那一点，但能大致知道它会出现在什么范围。

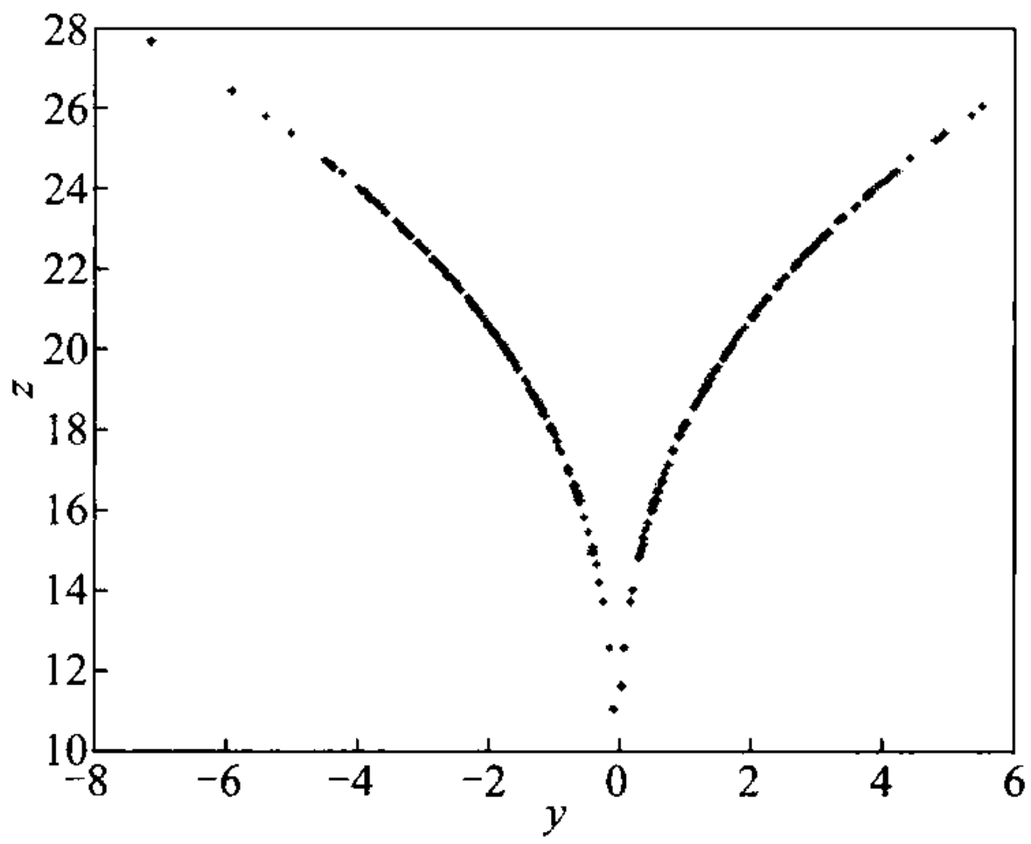


图 6.63 洛伦茨方程的庞加莱截面图

```
function lor
opts=odeset('Events',@events);
[t,y,te,ye,ie]=ode45(@lorfun,[0,600],[1,0,0] ,opts);
plot(ye(:,2),ye(:,3),'.')
xlabel('y');   ylabel('z')


    function ydot=lorfun(t,y)
    r=25;
    ydot=[  -10*y(1)+10*y(2);
      r*y(1)-y(1)*y(3)-y(2) ;
         y(1)*y(2)-8/3*y(3)];


      function [value,isterminal,direction]=events(t,y)
      value=y(1);  isterminal=0;   direction=0
```

洛伦茨系统来自于天气预报的模型，它显示了许多混沌运动的性质。真实的大气是比洛伦茨系统复杂得多的系统，也会有更复杂的混沌运动。即使建造一个能包括所有因素的复杂模型，也可能由于初始条件的微小差别而导致长期预报的失败。所以洛伦茨有一个报告题目是："Predictability: Does the Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas?" 如今常说的"蝴蝶效应"即来源于此。

§6.5 练 习

1. 单摆的周期可以有解析解(椭圆函数表示)和数值解两种方法，比较一下两者的误差。计算一下$1°, 5°, 10°$的周期是多少，相等吗?

2. 画出有阻尼$0 < \beta < 1$和过阻尼$\beta > 1$时无驱动的单摆运动的角位移和相图。

3. 画出有阻尼有驱动时单摆（耗散系）作周期2，周期4和混沌运动的角位移图。

4. 将弹簧与小球垂直碰撞的方程无量纲化。

5. 求解有阻尼无驱动的倒摆方程，初条件取 $\theta = 0$，$\dot{\theta} = \pm 0.8$，画出相图，并与图 6.26 作比较。

6. 对 VDP 方程作如下两种数值计算分析

(1) 对驱动频率用FFT作频谱分析，对比一下它是否与周期1的频率一样。

(2) 编程演示极限环现象。

7. 计算教材中各种情况下的洛伦茨方程三变量的位移图形，并解释这些图形的含义。

8. 画出$r = 25$，初值为[1，0，0]时，在$y = 0$平面上得到的洛伦茨方程的庞加莱截面，与教材中的庞加莱截面图进行比较，两个图形有什么特征，为什么会这样？这个截面是与初值无关的，例如将初值改为[0，1，1]，再画一次庞加莱图，看看是否相同。

自己再对洛伦茨系统的其他混沌情形作几个截面进行观察。

9. Henon映射是二维映射

$$\begin{cases} x_{i+1} = y_i + 1 - \mu x_i^2 \\ y_{i+1} = \eta x_i \end{cases}$$

(1) 画出混沌吸引子，取$\eta = 0.3$，$\mu = 1.4$；

(2) 画$x - \mu$的分岔图，取$\eta = 0.3$，$1.05 < \mu < 1.082$。

10. Rössler模型是

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + xz - cz \end{cases}$$

(1) 取 $a = 0.32$，$b = 3$，$c = 6$，画出三维图形及其在三个平面上的投影。

(2) 取 $b = 2$，$c = 4$，改变a将得到如下图形：$a = 0.3$(极限环)；$a = 0.35$（周期2）；$a = 0.375$（周期4）；$a = 0.386$（四带混沌吸引子）；$a = 0.3909$（周期6）；$a = 0.398$（单带混沌）；$a = 0.4$（周期5）；$a = 0.411$（周期3）。

第七章　解偏微分方程

物理量通常是时间变量与空间变量的函数，描述它们的变化常常需要偏微分方程。只有很少的一些偏微分方程可以用解析方法求解，所以数值方法求解偏微分方程就是一种很重要的研究手段。

物理上重要的偏微分方程大多数是二阶的，含有一个时间变量和三个空间变量，最简单的有以下三种类型：

椭圆型　$$-\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\varphi(x,y,z)=S(x,y,z)$$

抛物型　$$\left[\frac{\partial}{\partial t}-a^2\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\right]\varphi(x,y,z,t)=S(x,y,z,t)$$

双曲型　$$\left[\frac{\partial^2}{\partial t^2}-a^2\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\right]\varphi(x,y,z,t)=S(x,y,z,t)$$

椭圆方程的典型例子是描述静电场的拉普拉斯方程，抛物型方程的典型例子是热传导方程或物质扩散方程，双曲型方程的典型例子是波动方程。

求解这些方程还需要附加的边界条件与初给条件。双曲型方程需要两个初始条件，即初始位移与初始速度。抛物型方程需要一个初始条件即初始温度分布。椭圆型方程不需要初始条件。

对于一个、两个、三个空间变量的情况分别需要两个、四个、六个边界条件。边界条件的类型有：狄利克雷边界条件（第一类边界条件），它给出了区域边界上函数值；诺伊曼边界条件（第二类边界条件），它给出了边界上的函数的法向导数；混合边界条件（第三类边界条件），它给出了边界上的函数及其法向导数的线性组合。

§7.1　差分法解热传导方程

§7.1.1　显式差分公式

一维的热传导方程是

$$u_t=a^2u_{xx}$$

根据导数的差分公式，有

$$u_t\approx\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}$$

$$u_{xx} \approx \frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{(\Delta x)^2}$$

热传导方程可以写成差分形式(右边取t时刻的值计算)

$$\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} \approx a^2\frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{(\Delta x)^2}$$

即

$$u(x,t+\Delta t) \approx u(x,t)+\frac{\Delta t}{(\Delta x)^2}a^2\left[u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)\right]$$

令

$$x=i\Delta x,\ t=j\Delta t,\quad i,\ j=0,1,2,\cdots,n-1$$

$$r=\frac{\Delta t}{(\Delta x)^2}a^2$$

上式可写成足标形式

$$u_{i,j+1}-u_{i,j}=ru_{i-1,j}-2ru_{i,j}+ru_{i+1,j}$$

整理得显式差分公式

$$u_{i,j+1}=(1-2r)u_{i,j}+r(u_{i+1,j}+u_{i-1,j})$$

公式表示 $j+1$ 时刻 i 点的值由 j 时刻 $i-1$，i，$i+1$ 三点的值所确定。如图 7.1 所示。

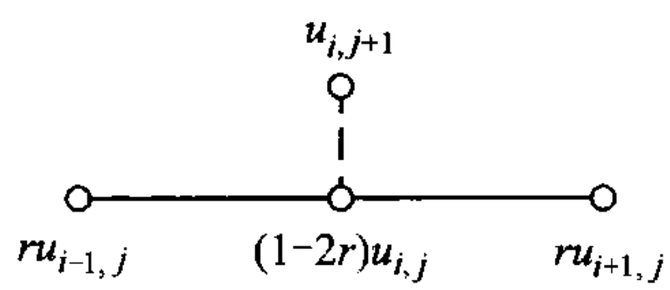


图 7.1 热传导方程显式差分公式中数据点的关系

稳定条件为 $0 \leqslant r \leqslant 1/2$，截断误差为 $O((\Delta x)^2,\ \Delta t)$。如果稳定条件不满足，则前一时刻产生的计算误差会扩大下一时刻的计算误差，造成数值解的失真。

以有限长细杆的热传导问题为例运用这个公式，定解问题是

$$\begin{cases} u_t=a^2u_{xx} \\ u(0,t)=0,\quad u(l,t)=0 \\ u(x,t=0)=\varphi(x) \end{cases}$$

为了下面进行数值计算，对其中的参数给出具体值 $l=20$，$t=25$，$a^2=10$ 且

$$\varphi(x)=\begin{cases}1 & (10\leqslant x\leqslant 11)\\ 0 & (x<10, x>11)\end{cases}$$

根据上面的公式，编写如下程序。由于是在一个图形窗口连续画图，所以出现了二维动画的效果。而语句 surf 是用三维的图形表示这个过程，所得图形如图 7.2。

```
x=0:20;    a2=10; r=a2*0.01;
u=zeros(21,25);                                    %预设矩阵以存放求得的解
u(10:11,1)=1;                                      %初始条件
for j=1:25                                         %求解及作图
    u(2:20,j+1)=(1-2*r)*u(2:20,j)+r*(  u(1:19,j)+ u(3:21,j));
    plot(x,u(:,j));   axis([0 21 0 1]);   pause(0.1)
end
meshz(u)
```

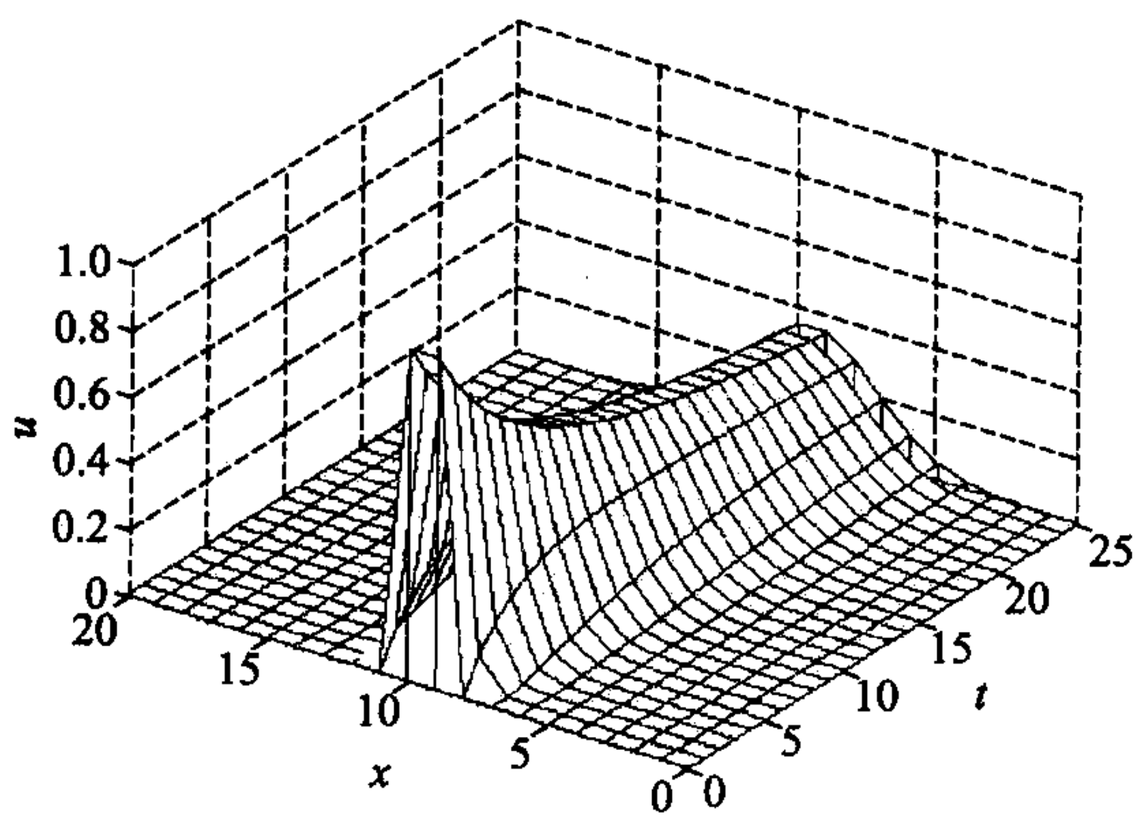


图 7.2 有限长细杆的热传导问题的解

再解一个非齐次方程的例子。非齐次方程的差分形式很容易用类似的方法导出。定解问题是[①]

$$\begin{cases}u_t-a^2u_{xx}=-bu_x & (0\leqslant x\leqslant l, t\geqslant 0)\\ u(x=0)=0, \quad u(x=l)=0\\ u(t=0)=(x-l/2)^2\end{cases}$$

① 梁昆淼. 数学物理方法. 3版. 北京: 高等教育出版社, 1998: 215

这个问题可以用解析方法求解，解析解是

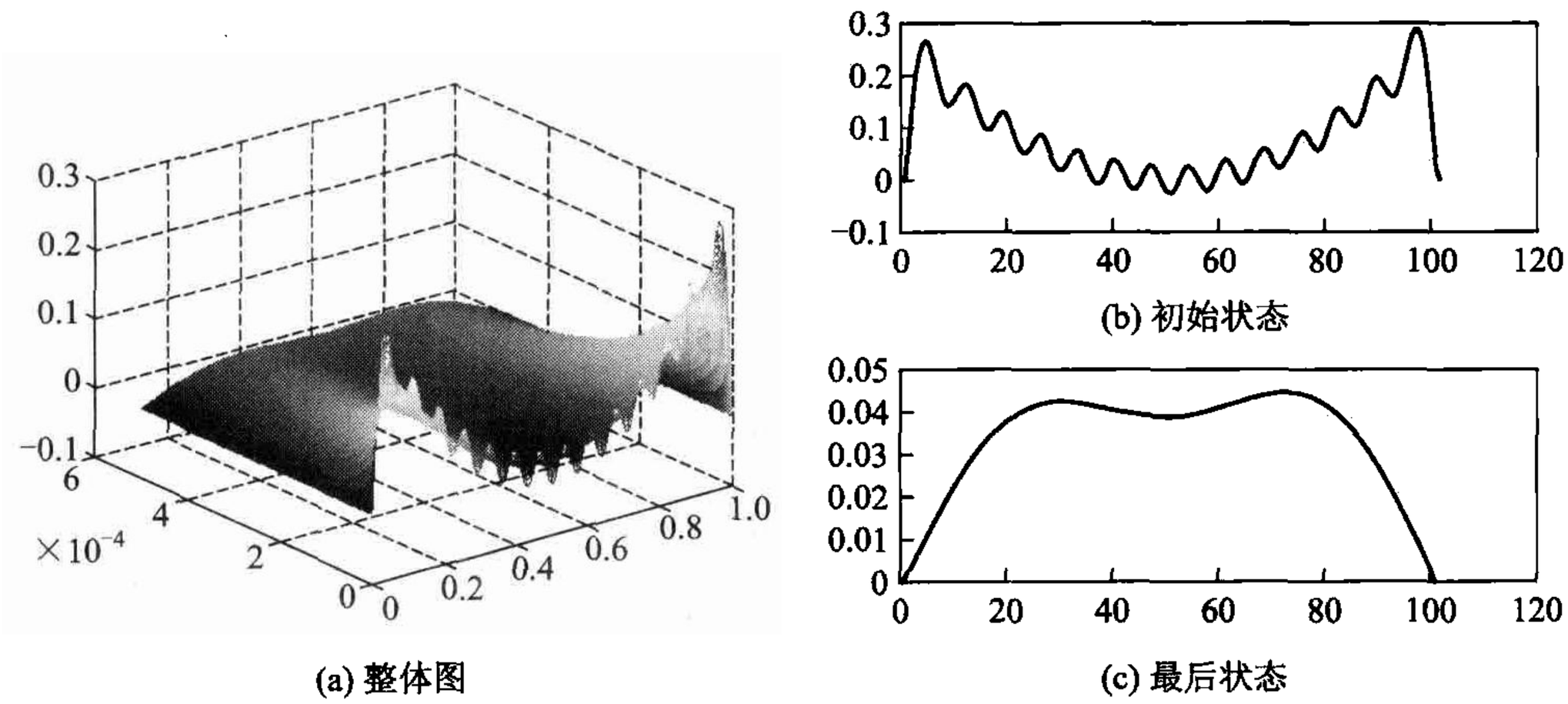
$$u(x,t)=\sum_{n=1}^{\infty}A_n\mathrm{e}^{-\left(\frac{n^2\pi^2a^2}{l^2}+\frac{b^2}{4a^4}\right)t}\mathrm{e}^{\frac{bx}{2a^2}}\sin\frac{n\pi}{l}x$$

其中

$$A_n=\frac{2}{l}\int_0^l\left(x-\frac{l}{2}\right)^2\mathrm{e}^{-\frac{bx}{2a^2}}\sin\frac{n\pi}{l}x\mathrm{d}x$$

为了与差分方法对比，先画出解析解的图形。

计算中取$0\leqslant t\leqslant 0.0005$，这个时间很短，因为这个过程的时间很短。解析解的图形如图 7.3，(a) 图是以 t，x 为变量所画的表面图，从图中可以看出变化的全貌；(b) 图是初始状态；(c) 图是最后的状态。解析解在初始状态所画出的图形与差分方程的解有一定的偏差。



(a) 整体图　(b) 初始状态　(c) 最后状态

图 7.3　非齐次方程的输运问题的解析解

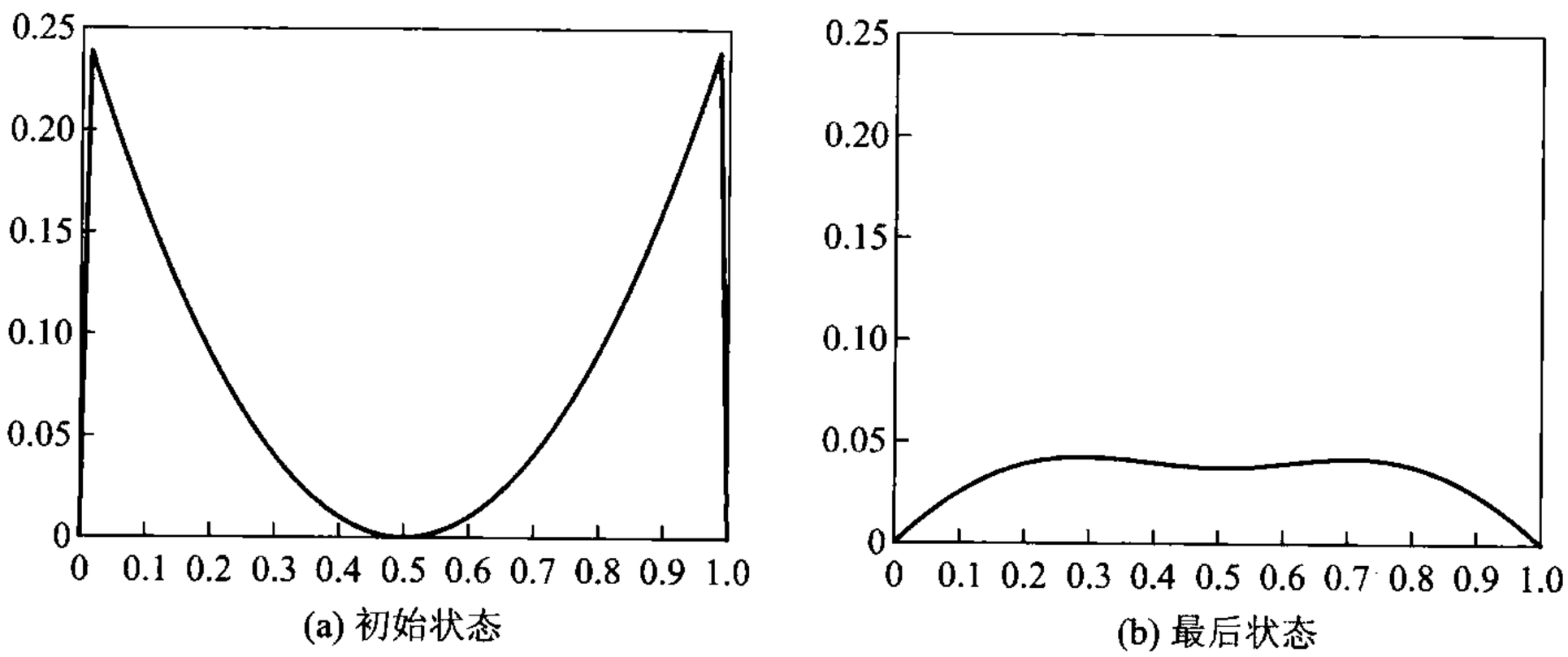
所用的程序如下，在这里$a^2=50$，$b=5$，$l=1$，为了方便比较，在差分计算中也采用了相同的数据。

```
a2=50; b=5;
[x, t] = meshgrid(0:0.01:1,0:0.000001:0.0005);
Anfun=inline('2*(x-0.5).^2.*exp(5*x./2./50).*sin(n*pi*x)','x',
'n');
u=0
for n=1:30
    An=quad(Anfun,0,1,[ ],[ ],n);
    un=An*exp(-(n*n*pi*pi*50+25/4/2500).*t).*...
       exp(5/2/50.*x).*sin(n*pi*x);
```

```
    u=u+un;
    size(u)
end
mesh(x,t,u)

figure
subplot(2,1,1)
plot(u(1,:))
subplot(2,1,2)
plot(u(end,:))
```

差分方程所得的数值解的图形如图 7.4，其中 (a) 图是开始的状态，(b) 图是最后状态。



(a) 初始状态　　(b) 最后状态

图 7.4　非齐次方程的输运问题的数值解

所用的程序是：

```
N=500;  dx=0.01;  dt=0.000001;
c=50*dt/dx/dx;
A=500; b=5;
x=linspace(0,1,100)';                          %将变量设成列向量
uu(1:100,1)=(x-0.5).^2;                        %初温度为零
figure
h=plot(x,uu(:,1),'linewidth',5);
set(h,'EraseMode','xor')
axis([0,1,0,0.25]);
```

```
for k=2:200
   uu(2:99,2)=(1-2*c)*uu(2:99,1)+c*(uu(3:100,1)+uu(1:98,1))-...
              b*dt/dx*(uu(3:100,1)-uu(2:99,1));
   uu(1,2)=0;  uu(100,2)=0;                                  %加上边界条件
   uu(:,1)=uu(:,2);
   set(h,'YData',uu(:,1)) ;
   drawnow;
   pause(0.01)
end
```

§7.1.2 隐式公式与平均隐式公式

如果热传导方程右边对空间的二阶导数取 $t+\Delta t$ 时刻的值进行计算，则得到隐式差分公式

$$\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}$$

$$\approx a^2\frac{u(x+\Delta x,t+\Delta t)-2u(x,t+\Delta t)+u(x-\Delta x,t+\Delta t)}{(\Delta x)^2}$$

用足标表示得

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t}=a^2\frac{u_{i+1,j+1}-2u_{i,j+1}+u_{i-1,j+1}}{(\Delta x)^2}$$

等式右边是矩阵运算，

$$\frac{a^2}{(\Delta x)^2}\begin{pmatrix}1 & -2 & 1 & 0 & 0 & 0 & \cdots\\ 0 & 1 & -2 & 1 & 0 & 0 & \cdots\\ 0 & 0 & 1 & -2 & 1 & 0 & \cdots\\ 0 & 0 & 0 & 1 & -2 & 0 & \cdots\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix}\begin{pmatrix}u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} & \cdots\\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} & \cdots\\ u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} & \cdots\\ u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} & \cdots\\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

引入算符L，记

$$Lu_{i,j+1}=\frac{a^2}{(\Delta x)^2}(u_{i+1,j+1}-2u_{i,j+1}+u_{i-1,j+1})$$

则得到隐式公式为

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t}=Lu_{i,j+1}$$

数据点之间关系图 7.5 所示。

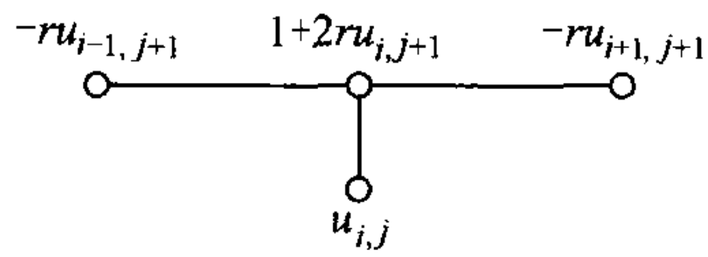


图 7.5 热传导方程隐式公式中数据点间的关系

对显式公式也引入同样的算符，即令

$$Lu_{i,j} = \frac{a^2}{(\Delta x)^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$$

则显式公式写成

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = Lu_{i,j}$$

将显式与隐式相加，得平均公式

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{1}{2}Lu_{i,j} + \frac{1}{2}Lu_{i,j+1}$$

将各项写出来就是

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{a^2}{2(\Delta x)^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1})$$

公式常写成下列形式

$$u_{i,j+1} = \frac{1 + \frac{1}{2}L\Delta t}{1 - \frac{1}{2}L\Delta t}u_{i,j}$$

这个公式也叫Crank-Nicolson公式，是六点对称的格式，数据点之间关系如图 7.6 所示

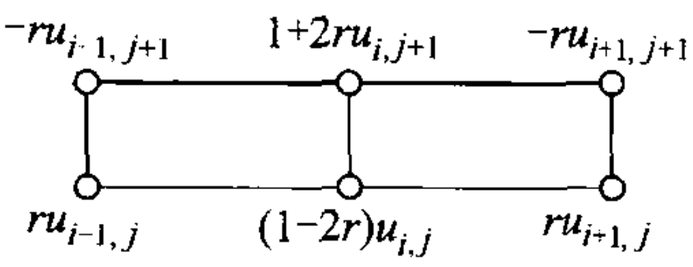


图 7.6 热传导方程平均隐式公式中数据点间的关系

可以证明，这个公式对任何步长都是恒稳定的。

下面用这个公式解一维运动粒子贯穿势垒的薛定谔方程

$$\mathrm{i}\hbar\frac{\partial\varphi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\varphi + V\varphi$$

令$\hbar=1$，$2m=1$，一维方程可以化为如下形式，

$$\frac{\partial\varphi}{\partial t}=\mathrm{i}\left(\frac{\partial^2}{\partial x^2}+v\right)\varphi$$

再引入算符

$$H=\left(\frac{\partial^2}{\partial x^2}+v\right)$$

则这个方程可以利用上面推导的平均公式求解，即

$$\varphi_{i,j+1}=\left(\frac{1+\mathrm{i}\frac{1}{2}H\Delta t}{1-\mathrm{i}\frac{1}{2}H\Delta t}\right)\varphi_{i,j}$$

改写成下述形式

$$\varphi_{i,j+1}=\left(\frac{2}{1-\mathrm{i}\frac{1}{2}H\Delta t}-1\right)\varphi_{i,j}=\frac{2\varphi_{i,j}}{1-\mathrm{i}\frac{1}{2}H\Delta t}-\varphi_{i,j}$$

引入中间函数χ，则有

$$\chi=\frac{2\varphi_{i,j}}{1-\mathrm{i}\frac{1}{2}H\Delta t}$$

$$\varphi_{i,j+1}=\chi-\varphi_{i,j}$$

最后两式就是编程计算所用的公式。编程时，先计算出每一时刻的$\varphi_{i,j}$对应的χ，程序中是把χ的系数表示为A，然后利用MATLAB的矩阵方程求解的功能求χ，再用它去求下一时刻的$\phi_{i,j+1}$。

程序在一个 220 点的格子上，定义解析形式的位势 (方形势阱或势垒，高斯势阱或势垒，位势台阶或抛物线势阱)，建立一个初始的高斯波包或洛伦茨波包，它具有规定的平均位置、动量和空间宽度，并在保持格子的端点上 φ 为零的边界条件下演化。用动画在每一时间步长上都显示概率密度 $|\varphi|^2$ 和位势，以及波包在一指定点的左边和右边两部分每一部分的总概率和平均位置。指令 diag 是生成对角矩阵，其用法前面已有介绍。

程序使用的数据是，选位势为一个方形位垒，高度是0.18，位于格点编号为105和115 的位置。初始的波包是一高斯波包，其平均波数和宽度为$k0=0.5$，$\sigma=10$，中心位置x0在编号为40的格点上。时间步长为1，演化时间为130。动画演示了波包向势垒行进，在势垒上发生透射和反射，最后分为透射波和反射波两部分向不同的方向运动。图 7.7 给出了这个过程中的几个画面。

```
x=1:220;        v(x)=0;        v(105:115)=0.18;          %势垒函数
k0=0.5;    d=10;      x0=40;     %波包的参数:平均动量, 宽度和中心
B0=exp(k0*i*x).*exp(-(x-x0).^2*log10(2)./d^2);   %初始的高斯波包
B(:,1)=B0.';                                      %转置成为列向量
A=diag(-2+2i+v)+diag(ones(219,1),1)+diag(ones(219,1),-1);
for t=1:130
   C(:,t+1)=4i*(A\B(:,t));                        %计算中间函数CHI
   B(:,t+1)=C(:,t+1)-B(:,t);
   plot(x,abs(B(:,t)).^2/norm(B(:,t)).^2,x,v/2);%画归一化后的概率
   pause(0.1)
end
```

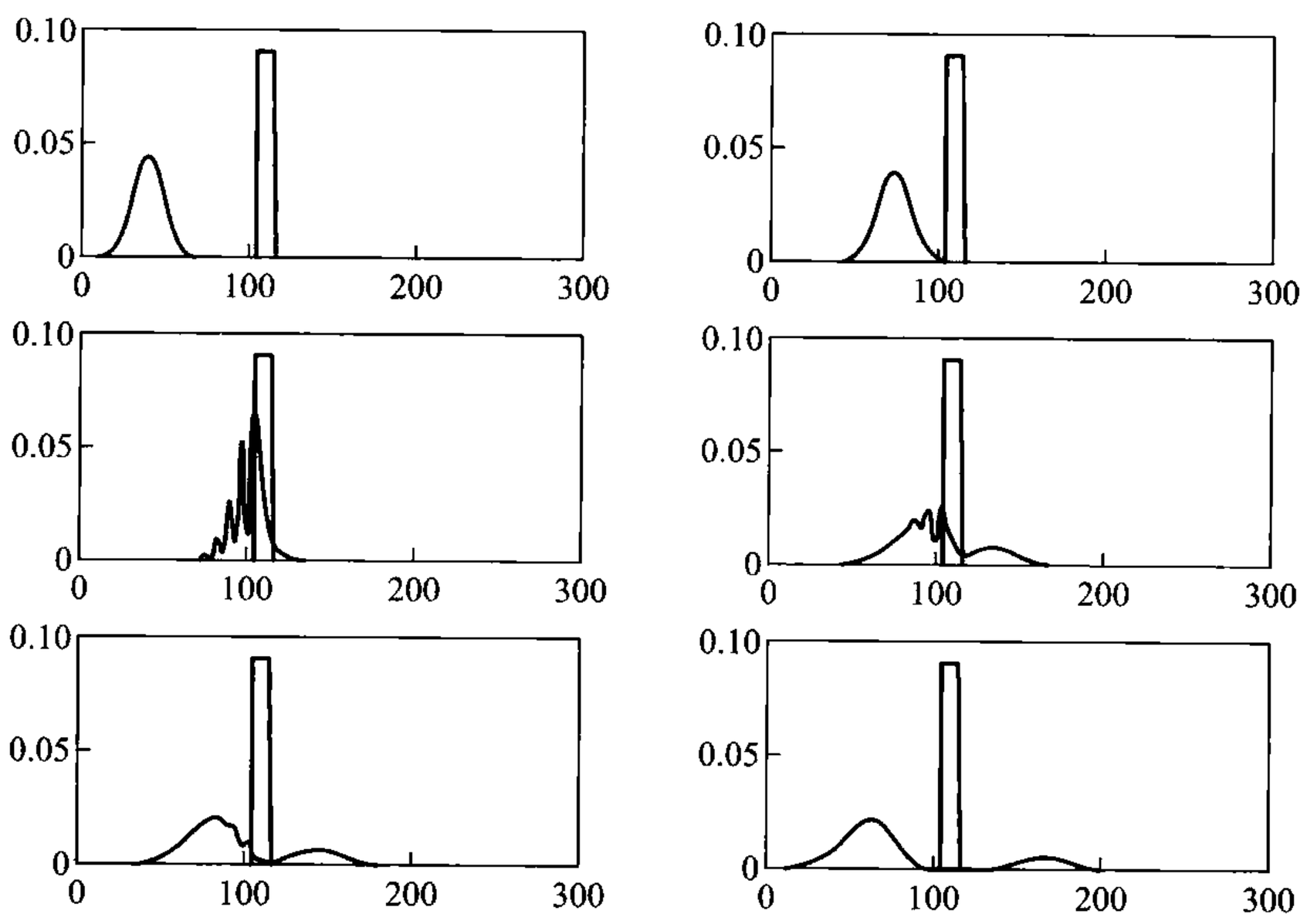


图 7.7 高斯波包贯穿势垒的过程

§7.2 差分法解弦振动方程

弦振动方程是一维双曲型方程的典型例子。下面研究两端固定的弦，相应的运动方程是

$$u_{tt} = a^2 u_{xx}$$

另外为了作计算，假定初速为零，初位移是

$$u(x,0)=\begin{cases}-x & (0\leqslant x\leqslant 3/5)\\ -1.5+1.5x & (3/5<x\leqslant 1)\end{cases}$$

§7.2.1 显式格式

仿照前面的做法，波动方程的差分形式是

$$\frac{u(x,t+\Delta t)-2u(x,t)+u(x,t-\Delta t)}{(\Delta t)^2}$$

$$=a^2\frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{(\Delta x)^2}$$

令$x=i\Delta x$，$t=j\Delta t$，按中心差分的表达式，可将偏微分变为差分式表示

$$\frac{\partial^2 u}{\partial t^2}=\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{(\Delta t)^2}$$

$$\frac{\partial^2 u}{\partial x^2}=\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\Delta x)^2}$$

代入波动方程得显式差分公式

$$u_{i,j+1}=c(u_{i+1,j}+u_{i-1,j})+2(1-c)u_{i,j}-u_{i,j-1}$$

其中 $c=a^2\dfrac{(\Delta t)^2}{(\Delta x)^2}$。当$c<1$时，解是稳定的，当$c=1$时，可得到正确的数值解。当$c>1$时，解是不稳定的。其截断误差为$O(\Delta x^2,\Delta t^2)$。同样，读者可以自行推导相应的隐式公式，不过隐式公式使用不太方便。

§7.2.2 初始条件

显式公式表明，需要两行已知的数据才能求出下一行的数值。可用图 7.8 表示。

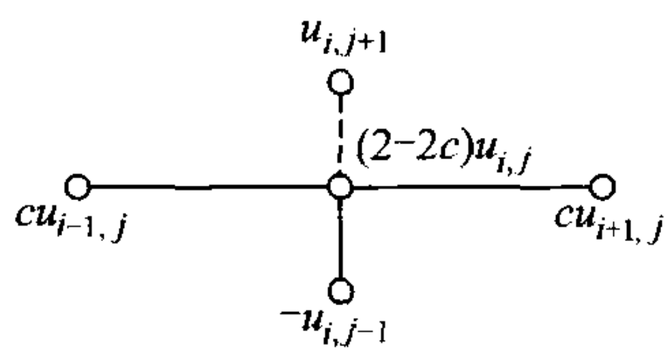


图 7.8 波动方程显式差分公式中数据间的关系

利用初始条件可以先求出两行数值作为公式计算的启动值。设$t=0$时对应$j=1$，初始位移$\phi(x)$和初始速度为$\psi(x)$离散化以后得$\phi_i=\phi(x_i)$和$\psi_i=\psi(x_i)$。则初始位移表示成

$$u_{i1}=\phi_i$$

而初速度用中心差分可表示为

$$\frac{\partial u_{i,1}}{\partial t}=\frac{u_{i,2}-u_{i,0}}{2\Delta t}=\psi_i$$

得$u_{i,0}=u_{i,2}-2\psi_i\Delta t$，把它代入显式差分公式的最后一项，得

$$u_{i,2}=\frac{c}{2}(u_{i+1,1}+u_{i-1,1})+(1-c)u_{i,1}+\psi_i\Delta t$$

更简单的公式是直接利用每一点的速度求其下一时刻的位置，即根据速度定义有

$$\psi_i=\frac{u_{i,2}-u_{i,1}}{\Delta t}$$

即

$$u_{i,2}=u_{i,1}+\psi_i\Delta t$$

为了减小误差，取较小的时间步长是明智的。

§7.2.3 实例

按照公式编成程序如下，其中取 $a=2$，$0\leqslant x\leqslant 1$。

```
a=2;  dt=0.05;   dx=0.1;  c=4*dt^2/dx^2;
x=0:dx:1;                                                    %弦长
u1(1:11)=0;    u2(1:11)=0;    u3(1:11)=0;                   %预设三个矩阵
u1(2:6)=-x(2:6);
u1(7:10)= -1.5+1.5*x(7:10);                                  %初始条件
plot(x,u1), axis([0,1,-1,1]); pause(0.3)
u2(2:10)=u1(2:10)+c/2*(u1(3:11)-2*u1(2:10)+u1(1:9));   %初始速度
plot(x,u2);    axis([0,1,-1,1]);   pause(0.3)
for k=2:12                                                   %求解方程
    u3(2:10)=2*u2(2:10)-u1(2:10)+c*(u2(3:11)-2*u2(2:10)
    +u2(1:9));
    u1=u2;   u2=u3;
    plot(x,u3),   axis([0,1,-1,1]);    pause(0.3)
end
```

程序运行速度太快，所以加上了暂停语句 pause 才能看清画出的图形。图 7.9 是用三维图表现这些图形。

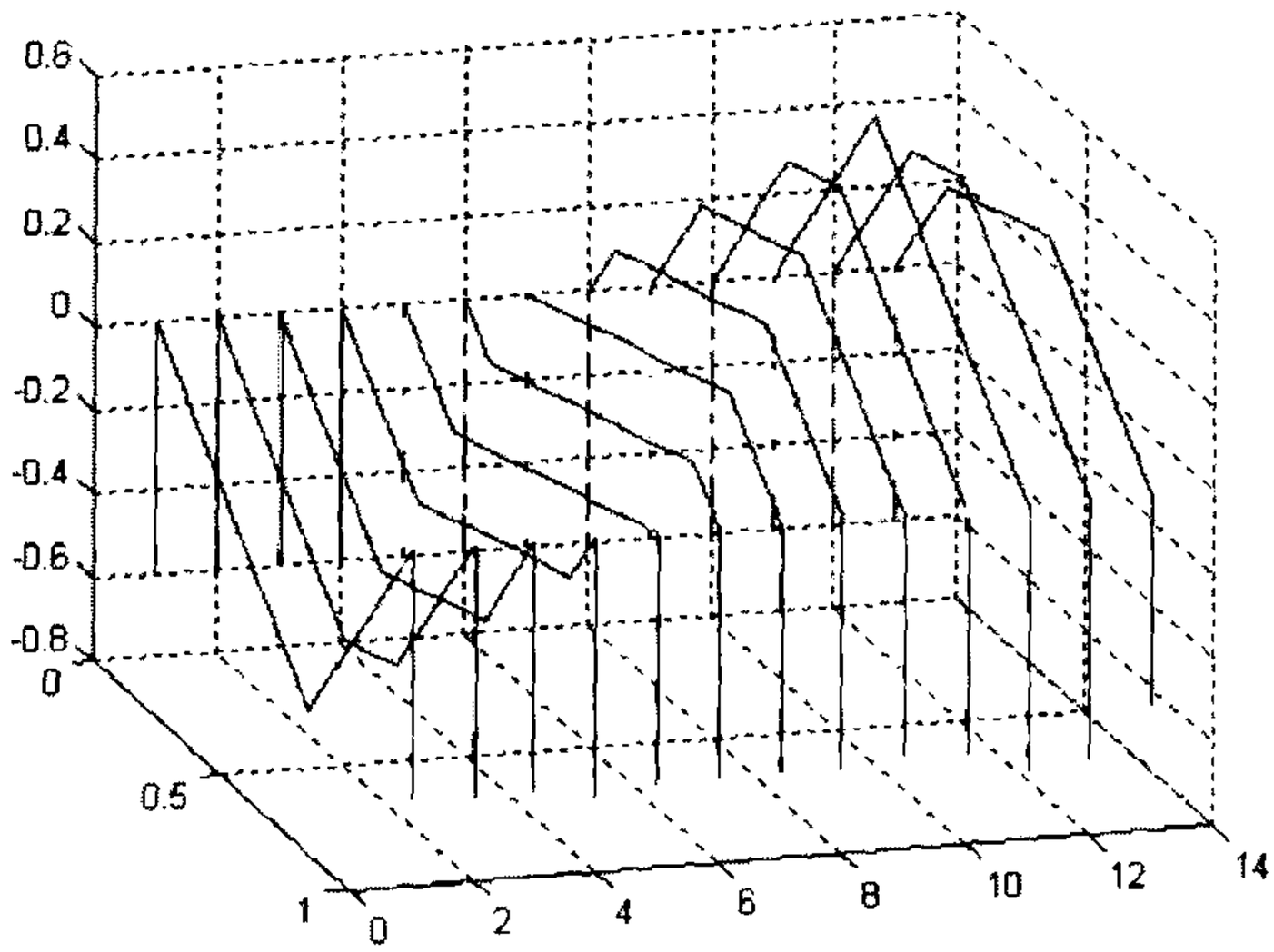


图 7.9 弦振动方程解的图像

再讨论一个例子，也是两端固定的均匀弦的自由振动，其初位移不为零初速为零即定解问题是

$$\begin{cases} u_{tt} = a^2 u_{xx} \\ u(x=0,t)=0, \qquad u(x=l,t)=0 \\ u(x,t=0)=\varphi(x), \quad u_t(x,t=0)=\psi(x) \end{cases}$$

设初位移为

$$\varphi(x) = \begin{cases} \sin\dfrac{7\pi}{l}x, & \dfrac{3l}{7} \leqslant x \leqslant \dfrac{4l}{7} \\ 0, & \text{其他情况} \end{cases}$$

作数值计算时取 $l=1$，$a=1$，差分解法所用的程序是：

```
clear
N=4010;    dx=0.0024;
dt=0.0005;     c=dt*dt/dx/dx;
x=linspace(0,1,420)';
u(1:420,1)=0;
u(181:240,1)=0.05*sin(pi*x(181:240)*7);

u(2:419,2)=u(2:419,1)+c/2*(u(3:420,1)-2*u(2:419,1)+u(1:418,1));
h=plot(x,u(:,1),'linewidth',3);
axis([0,1,-0.05,0.05]);
```

```
set(h,'EraseMode','xor','MarkerSize',18)
for k=2:N
   set(h,'XData',x,'YData',u(:,2)) ;
   drawnow;
   u(2:419,3)=2*u(2:419,2)-u(2:419,1)+c*(u(3:420,2)...
       -2*u(2:419,2)+u(1:418,2));
   u(2:419,1)=u(2:419,2);
   u(2:419,2)=u(2:419,3);
end
```

图 7.10 是其中的几个动画画面。

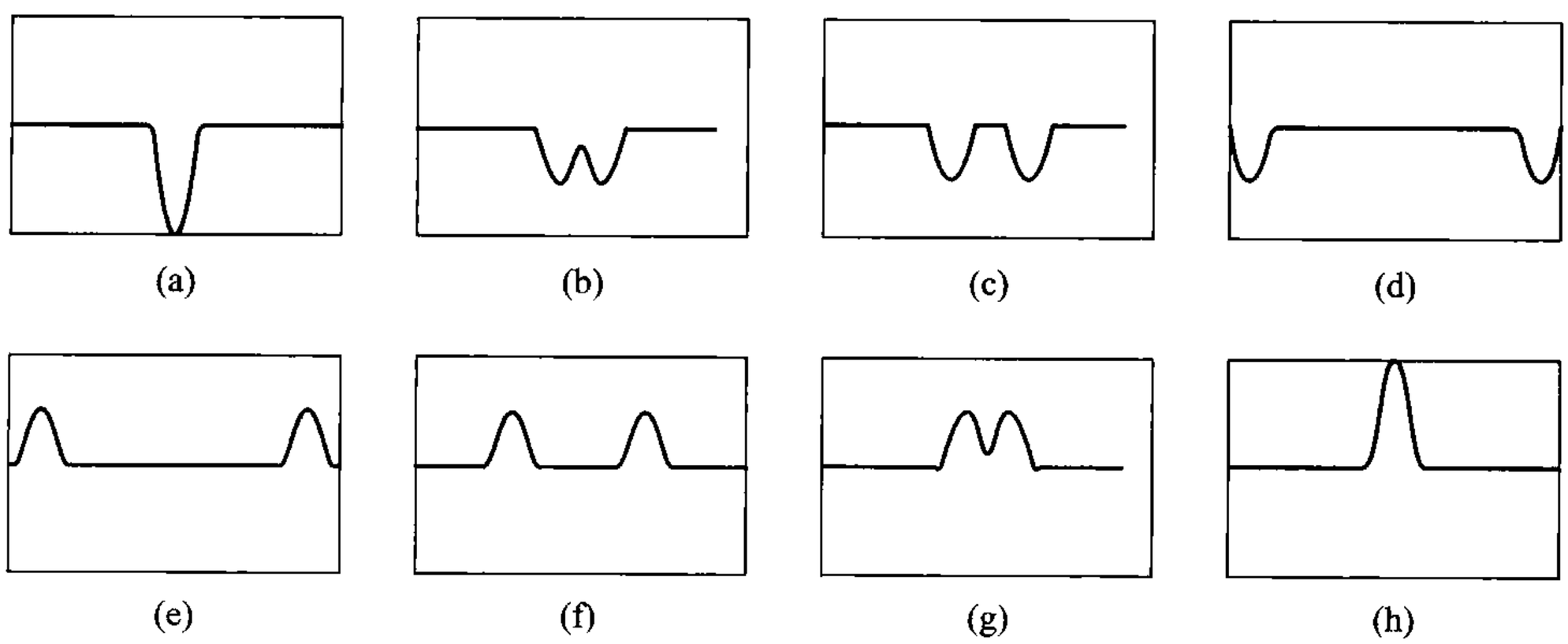


图 7.10 差分方程求得的数值解

下面将差分解法与解析解对比，它的解析解是

$$u(x,t)=\sum_{n=1}^{\infty}\left(A_n\cos\frac{n\pi at}{l}+B_n\sin\frac{n\pi at}{l}\right)\sin\frac{n\pi x}{l}$$

其中的系数是

$$A_n=\frac{2}{l}\int_l^0\varphi(\xi)\sin\frac{n\pi\xi}{l}\mathrm{d}\xi$$

$$B_n=\frac{2}{n\pi\alpha}\int_l^0\psi(\xi)\sin\frac{n\pi\xi}{l}\mathrm{d}\xi$$

将初始条件代入得

$$\begin{aligned}A_n=&\frac{1}{(7-n)\pi}\left[\sin(7-n)\pi\frac{4}{7}-\sin(7-n)\pi\frac{3}{7}\right]\\&-\frac{1}{(7+n)\pi}\left[\sin(7+n)\pi\frac{4}{7}-\sin(7+n)\pi\frac{3}{7}\right]\qquad(n\neq 7)\end{aligned}$$

$$A_7 = \frac{1}{7}$$

$$B_n = 0$$

将系数代入解的表达式中，级数中的每一项都是一个驻波。

级数解有无穷项，为了画出级数解的图形，需要给定 n 值。程序画出了 $n = 50$ 的动画，取 $n = 10$，则图形会有较大差别，读者可以自己试验一下。程序中使用了子程序 wfun.m 来计算级数中不同 k 的求和各项，然后在主程序 jxj 中将它们加起来。

```
function  jxj
N=50
t=0:0.005:2.0;
x=0:0.001:1;
ww=wfun(N,0);
ymax=max(abs(ww));
h= plot(x,ww,'linewidth',3);
axis([ 0, 1, -ymax, ymax])
sy=[ ];
for n=2:length(t)
   ww=wfun(N,t(n));
   set(h,'ydata',ww);
   drawnow;
   sy=[sy,sum(ww)];
end

function  wtx=wfun(N,t)
x=0:0.001:1;    a=1;  wtx=0;
for I=1:N
    if I~=7
      wtx=wtx+0.05*(   (sin(pi*(7-I)*4/7)-sin(pi*(7-I)*3/7))...
          /(7-I)/pi-(sin(pi*(7+I)*4/7)-sin(pi*...
          (7+I)*3/7))/(7+I)/pi  )*cos(I*pi*a*t).*sin(I*pi*x);
     else
        wtx=wtx+0.05/7*cos(I*pi*a*t).*sin(I*pi*x);
     end
end
```

其中 N 是级数求和的项数，图 7.11 是前 50 项时所得的动画图形。可以看到数值解与解析解的图像 (图 7.10 和图 7.11) 是一致的。

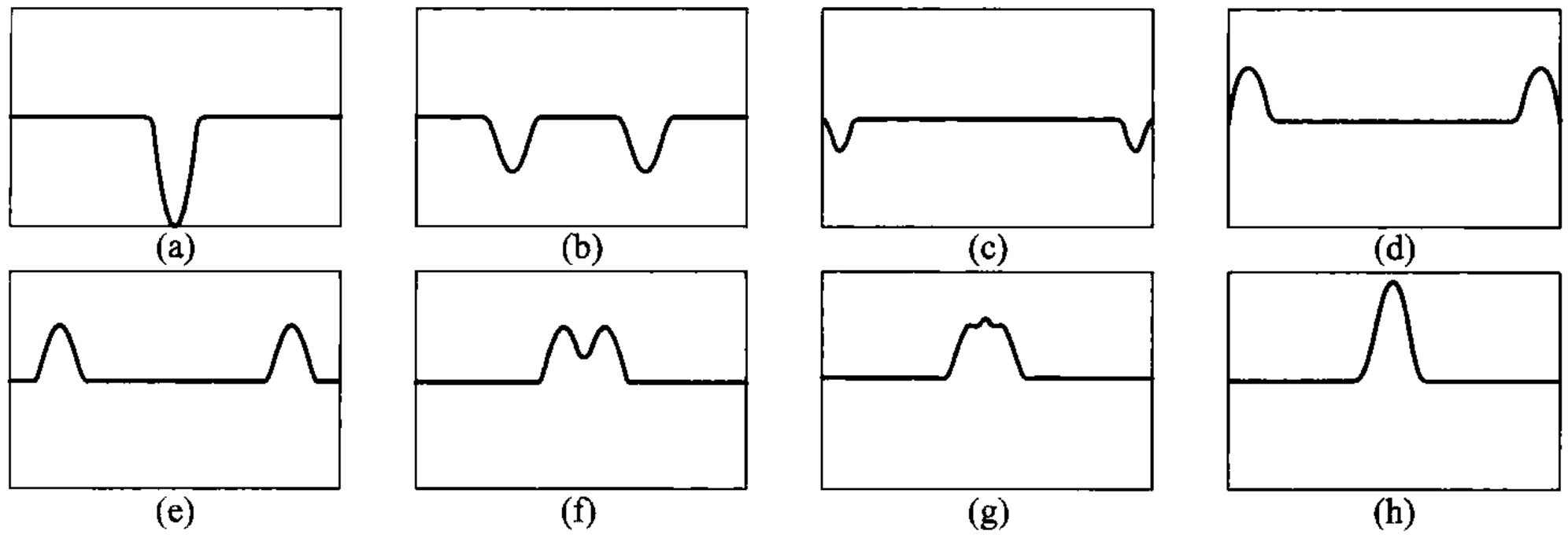


图 7.11 级数解前 50 项的图形

对有限长的弦，如果在讨论的时间范围内，边界的影响还没有到达，则产生的现象与无限长的弦是一样的。如果熟悉一维弦振动的达朗贝尔公式（可以参看数学物理方法教材），很容易将这里结果与公式进行对比。从解析解的图形还会看到以下几个结论：

(1) 在运动开始时，初始位移平均分为两半向正、负方向传播，在到达边界之前，相当于波在无限长弦上的传播。这一段时间的运动与达朗贝尔公式描述的现象一致。

(2) 此后，波形到达边界以后，发生反射，产生半波损失。

(3) 从边界反射回来的两个波，相向运动。相遇以后，产生叠加，然后继续传播。

(4) 级数解要取 50 项以后才能有较高的精度。

(5) 不同频率的驻波叠加成行波。

§7.3 差分法与松弛法解椭圆型方程

§7.3.1 显式差分公式

利用二阶导数的中心差分公式得到二维拉普拉斯方程 $u_{xx}+u_{yy}=0$ 的差分公式为

$$\frac{u(x+\Delta x,y)-2u(x,y)+u(x-\Delta x,y)}{(\Delta x)^2}$$
$$+\frac{u(x,y+\Delta y)-2u(x,y)+u(x,y-\Delta y)}{(\Delta y)^2}=0$$

记 $\Delta x=\Delta y=h$，$x=ih$，$y=jh(i,j=0,1,2,\cdots,n-1)$，得显式公式

$$u(i,j)=\frac{1}{4}\left[u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)\right]$$

这个公式的数据点之间的关系图 7.12 所示，每一个点的计算都与周围四个相邻的点有关。

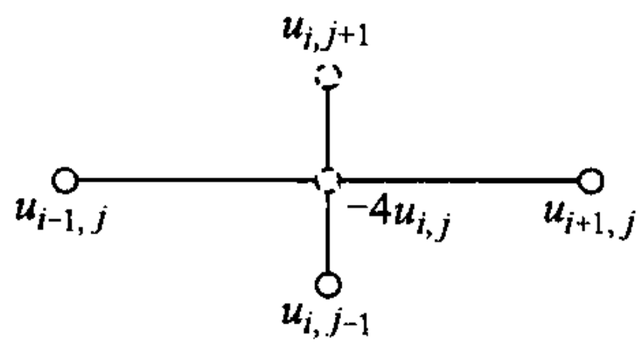


图 7.12 椭圆型方程显式差分公式中数据点间的关系

稳定条件是 $\dfrac{(\Delta y)^2}{(\Delta x)^2} \leqslant 1$。类似地推导得出隐式格式的公式为

$$
\begin{aligned}
& u(i,j+1) - 2u(i,j) + u(i,j-1) \\
= & -\frac{(\Delta y)^2}{(\Delta x)^2} \times [u(i+1,j+1) - 2u(i,j+1) + u(i-1,j+1)]
\end{aligned}
$$

它的截断误差是 $O((\Delta x)^2, (\Delta y)^2)$，该式是恒稳定的。

§7.3.2 边界条件

显式公式对每个网点作用都会产生一个代数方程，将边界条件加入进去，最终得到一个线性方程组。例如考虑一个 5×5 的方形区域图 7.13 所示，图上的数字是网点的足标。

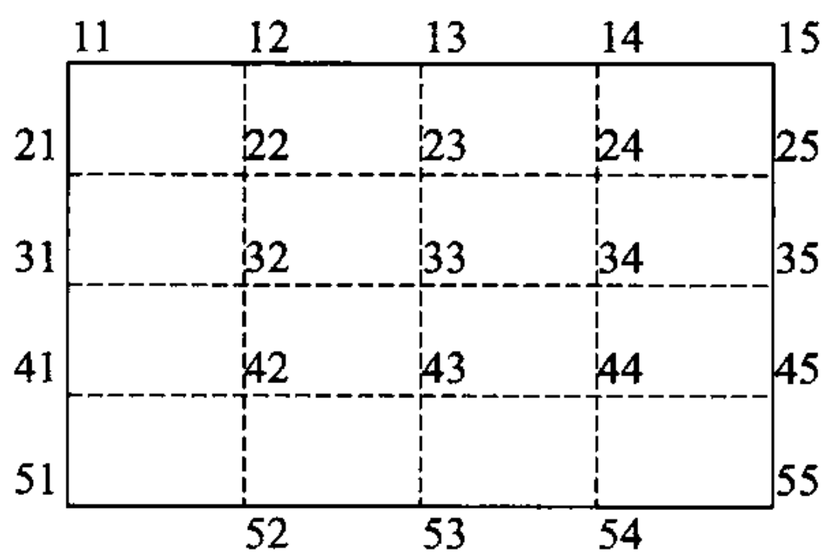


图 7.13 求解区域的网点的足标

这个区域有 16 个边界点，边界点的足标都标注在区域外，边界函数都以 ϕ 表示，则各边界点为 $\phi_{11}, \phi_{12} \cdots\cdots$ 其余的点是内部点，共有 9 个。对这 9 个点应用差分公式得到9个方程，其中点 u_{33} 不与边界点相邻接，所以方程右边为零。其余 8 个点均与边界点有不同的邻接方式，将已知的边界值移到方程右边就得到如下方程组：

$$\begin{pmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{12} \\ u_{13} \\ u_{21} \\ u_{22} \\ u_{23} \\ u_{31} \\ u_{32} \\ u_{33} \end{pmatrix} = \begin{pmatrix} -\varphi_{12} - \varphi_{21} \\ -\varphi_{13} \\ -\varphi_{14} - \varphi_{25} \\ -\varphi_{31} \\ 0 \\ -\varphi_{35} \\ -\varphi_{52} - \varphi_{41} \\ -\varphi_{53} \\ -\varphi_{54} - \varphi_{45} \end{pmatrix}$$

直接运用 MATLAB 的左除指令就能解这个方程组。这个方法的实际意义不大，因为计算这样的数目巨大的方程组，需要大量的内存计算，所以常用其他的方法如迭代法或松弛法进行计算。

对于第二类边界条件，边界值没有直接给定，但可以仿照抛物型方程中处理初始速度的方法来求出相应的边界值。例如假设边界条件是边界的法向导数为零，区域内共有 $m \times n$ 个点。对于右边界点上的某个点 u_{in}，如果要应用差分公式，则点 $u_{i,n+1}$ 的值是未知的，因为它在区域之外。由边界条件知

$$\frac{u_{i,n+1} - u_{i,n-1}}{2\Delta x} = 0$$

代入差分公式得

$$2u_{i,n-1} + u_{i-1,n} + u_{i+1,n} - 4u_{i,n} = 0$$

这个方程就是计算边界点$u_{i,n}$的公式。与此类似，参见图 7.13 知对左边界$u_{i,1}$、上边界$u_{1,j}$和下边界$u_{m,j}$分别有如下公式

$$2u_{i,2} + u_{i-1,1} + u_{i+1,1} - 4u_{i,1} = 0$$

$$2u_{2,j} + u_{1,j-1} + u_{1,j+1} - 4u_{1,j} = 0$$

$$2u_{m-1,j} + u_{m,j-1} + u_{m,j+1} - 4u_{m,j} = 0$$

利用上述公式同样也可以用第二类边界条件对椭圆型方程建立线性方程组。

§7.3.3 迭代法与松弛法

迭代法是直接用显式差分公式进行计算。如果在差分公式中，随时将上一步算得的各点的新值替代旧值，并且每次计算新值也替换成新值与旧值的“组合”，则得到下列松弛法的计算公式，其中 $0 < \omega < 2$。当 $\omega < 1$ 称为“低松弛”，$\omega > 1$ 称为“超松弛”。

$$u(i,j) = (1-\omega)u(i,j)$$

$$+\frac{\omega}{4}[u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)]$$

顺便指出，这个公式也适用于泊松方程。对于泊松方程

$$u_{xx}+u_{yy}=S(x,y)$$

松弛法的公式是

$$u(i,j)=(1-\omega)u(i,j)+\frac{\omega}{4}[u(i+1,j)+u(i-1,j)$$
$$+u(i,j+1)+u(i,j-1)-h^2S_{i,j}]$$

这里$S_{i,j}=S(ih,jh)$。

使用这两个公式都要先知道所有网点的值才能启动计算，而实际知道的只是边界条件即边界各点的值，内部各点的值并不知道。为此对所有边界点的函数值取平均，以它作为所有内部网点计算的启动值。因为数学上可以证明，区域的最大值与最小值一定在边界上。这一点从物理上很容易理解，假定研究的是一个温度场，因为内部没有热源（有源就是泊松方程），在平衡时，热量必然要从边界流入，也要从边界流出，那么温度最高与最低的点一定在边界上。所以取边界的平均值启动计算是合理的。

下面用显式差分公式解平面温度场，设正方形的一边温度为 10°C，其余各边的温度为 0°C，求稳定的温度场。定解问题是

$$\begin{cases} u_{xx}+u_{yy}=0 \\ u(0,y)=0, \quad u(100,y)=10 \\ u(x,0)=0, \quad u(x,100)=0 \end{cases}$$

按照上面的差分公式，任意设定内部各点的温度为零，并假定当两次迭代计算的值误差小于0.01时，就停止计算。另外三条边界值为零，一条为10，所以平均值粗略地取$10/4=2.5$作为计算的启动值。最后编出如下程序：

```
u=zeros(100,100); u(100,:)=10;
uold=u+2.5; unew=u;
for k=1:100
  if max(max(abs(u-uold)))>=0.01
  unew(2:99,2:99)=...
   0.25*(u(3:100,2:99)+u(1:98,2:99)+...
   u(2:99,3:100)+u(2:99,1:98));
  uold=u;   u=unew;
  end
end
surf(u)
```

计算结果得到的图形为图 7.14。

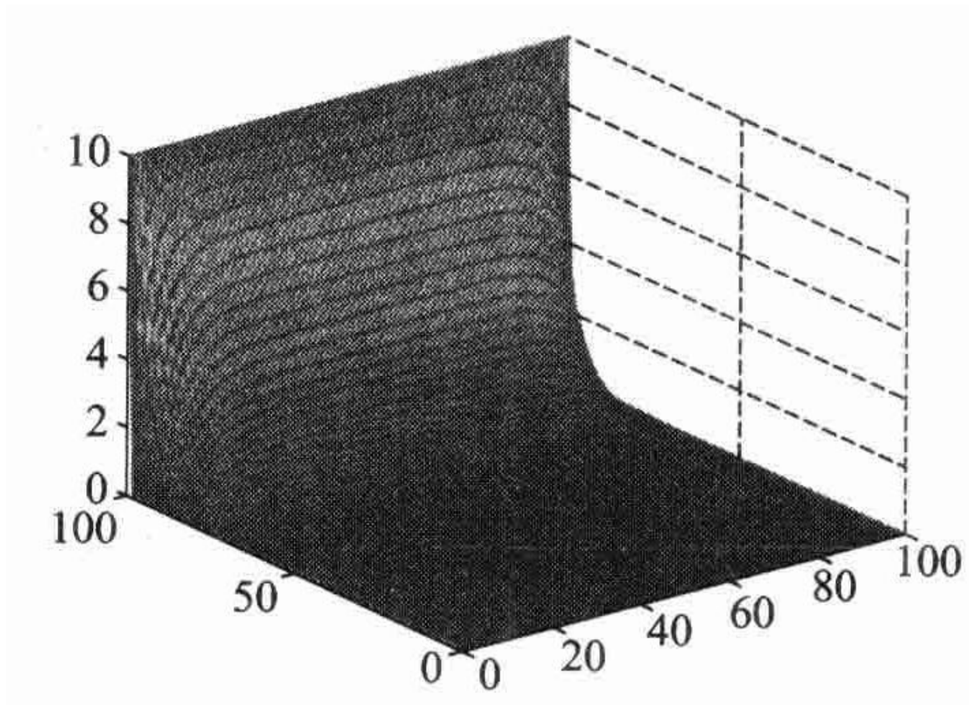


图 7.14 平面温度场的解的图形

再用松弛法解如下定解问题：

$$\begin{cases} u_{xx}+u_{yy}=0 \\ u(0,y)=0,\ u(a,y)=\mu\sin\dfrac{3\pi y}{b} \\ u(x,0)=0,\ u(x,b)=\mu\sin\dfrac{3\pi x}{a}\cos\dfrac{\pi x}{a} \end{cases}$$

为了进行数值计算，取 $\mu=1$，$a=3$，$b=2$。

编出的程序如下，所得图形为图 7.15。

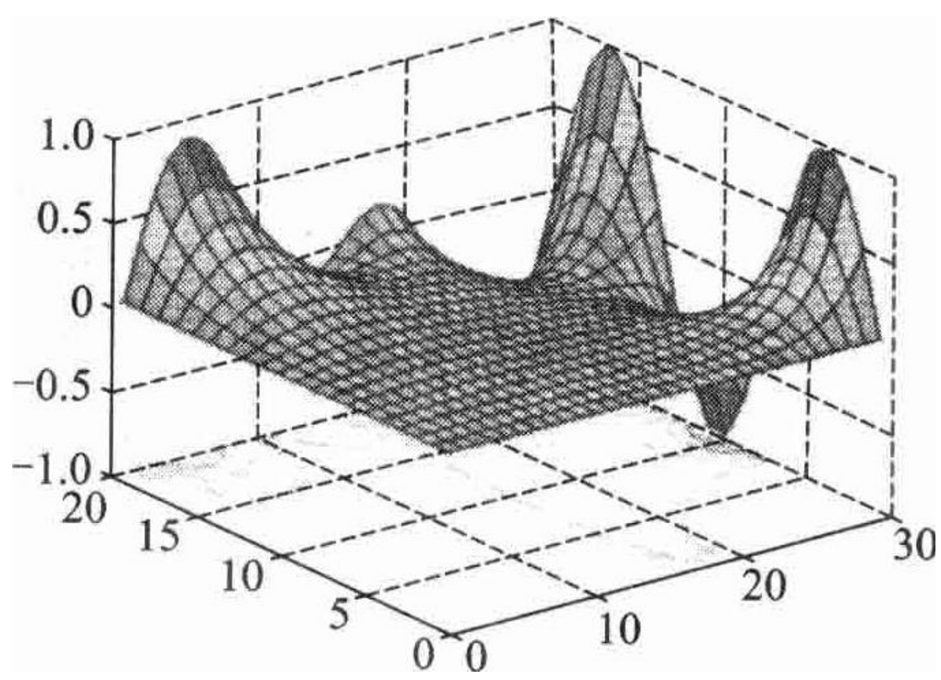


图 7.15 另一种平面温度场的解的图形

```
omega=1.5;
x=linspace(0,3,30);  y=linspace(0,2,20);
phi(:,30)=sin(3*pi/2*y)';
phi(20,:)=(sin(pi*x).*cos(pi/3*x));
for N=1:100
  for i=2:19
    for j=2:29
```

```
      ph=(phi(i+1,j)+phi(i-1,j)+phi(i,j+1)+phi(i,j-1));
      phi(i,j)=(1-omega)*phi(i,j)+0.25*omega*(ph);
    end
  end
end
colormap([0.5,0.5,0.5]);
surfc(phi)
```

§7.4 偏微分方程的工具箱 (PDETOOL)

PDETOOL 是 MATLAB 解二维偏微分方程的工具箱。指令 pdetool 可以打开工具箱图形界面，如图 7.16 所示。

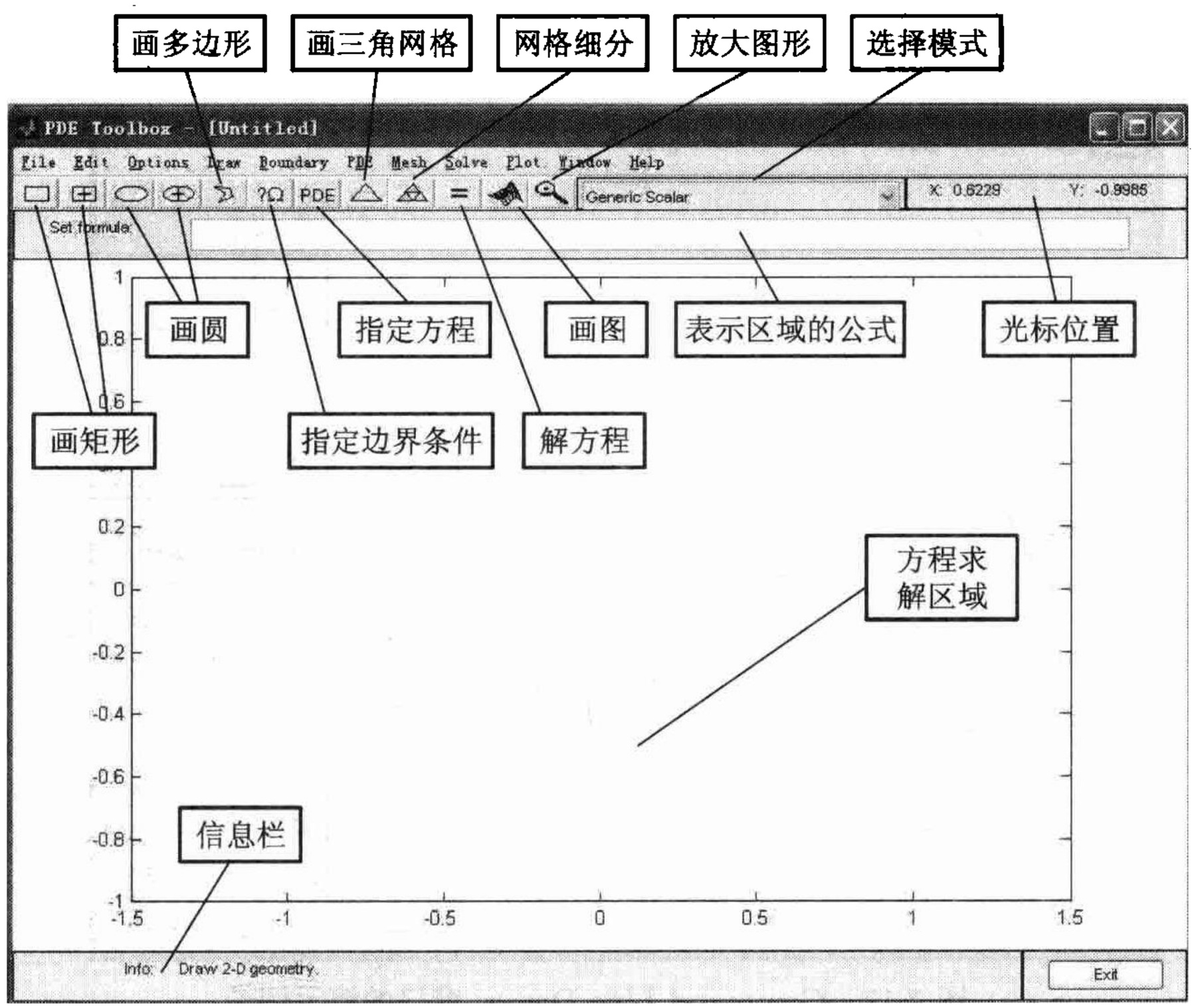


图 7.16 PDETOOL 工作界面

各个按钮的作用已经在图中标明。

解题的区域直接用画图来确定。通过不同图形的加、减组合来形成需要的复杂图形。加减的功能由作图时窗口中填写的公式来决定。

画矩形与画圆都有从中心开始画或从边上开始画两种功能。画多边形必须

使起点与终点重合才能形成封闭图形。

方程与边界条件的输入必须按照下面介绍的固定格式输入才行。

将区域分成三角形网格是因为 PDETOOL 采用的解法是有限元法，将三角形细分可以提高解题的精度。

使用 PDETOOL 界面解偏微分方程只要学习界面上按钮的操作就行，不需了解解题的过程，也不必编程。这种解法简单直观，有利于展现解的物理图像。

§7.4.1 PDETOOL 的功能演示

偏微分方程工具箱 (PDE Toolbox) 就是 MATLAB 中求解二维偏微分方程 (含时或不含时) 的工具，查看偏微分方程工具箱的演示程序可以简捷地了解偏微分方程工具箱的基本功能，操作如下：

在指令窗口键入 pdedemos，就打开一个 Command Line Demos 窗口如图 7.17 所示。

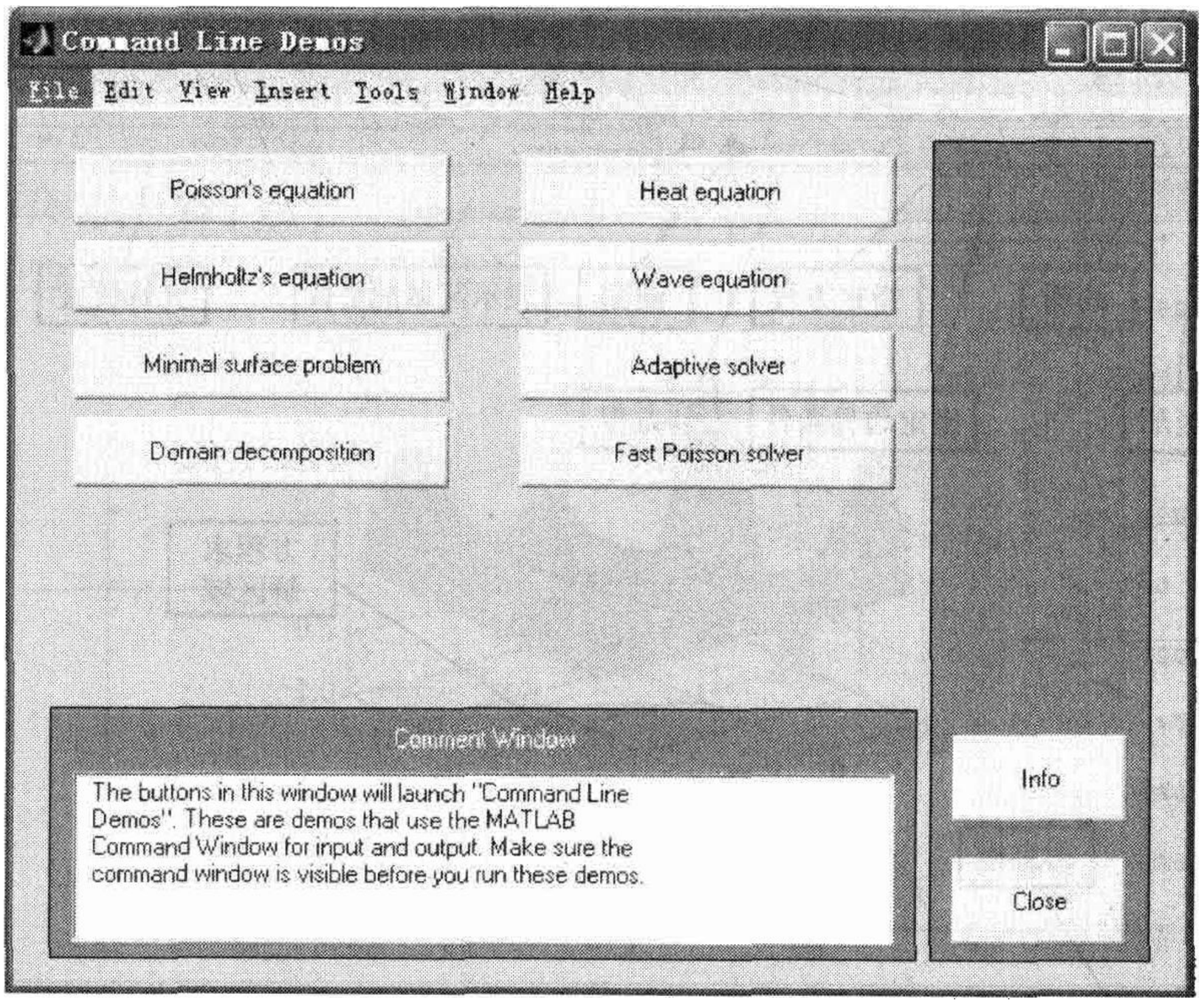


图 7.17 Command Line Demos 窗口的演示例子

窗口的左边是8个演示例子的按钮，下面是注解信息框 (Comment Window)，右边是信息提示按钮 (Info) 和退出按钮 (Close)。

从信息框中的说明可以知道，这 8 个按钮可以直接运行指令形式的 8 个偏微分方程的演示程序。演示程序的输入与输出都显示在指令窗口，所以在运行演示程序时，一定要保持指令窗口显示在前面。程序在运行时，会把程序的内容显

示在窗口中，每显示一段程序都会暂停一下，并在最后一行出现如下的提示文字

```
pause % Strike any key to end.
```

只要敲击任何一个键都会使程序继续运行，如此进行下去，直到程序运行结束。

点击信息提示按钮可以打开帮助系统中有关这个演示窗口的说明文字。这 8 个演示程序的源程序依次是 pdedemo1(2~8)，可以单独在命令窗口中运行，只需键入

```
>>pdedemo1
```

就是运行第一个程序，而键入

```
>>edit pdedemo1
```

则是查看第一个程序的源程序。对其他程序的运行方法与此相似。

§7.4.2 PDETOOL 中方程的输入格式

定义在二维有界区域 Ω 上的下列形式的方程可以用偏微分方程工具箱求解

椭圆型 $-\nabla\cdot(c\nabla u)+au=f$

抛物型 $d\dfrac{\partial u}{\partial t}-\nabla\cdot(c\nabla u)+au=f$

双曲型 $d\dfrac{\partial^2 u}{\partial t^2}-\nabla\cdot(c\nabla u)+au=f$

本征值方程 $-\nabla\cdot(c\nabla u)+au=\lambda du$

在这里，u是偏微分方程的解，在本书的叙述中，都是采用这个符号，c，a，f，d是标量复函数形式的系数，在抛物型和双曲型方程中，它们也可以是 t 的函数，λ是待求的本征值。

当系数 c，a，f 是 u 的函数时，我们称之为非线性方程，形式为

$$-\nabla\cdot(c(u)\nabla u)+a(u)u=f(u)$$

的非线性方程也可以用偏微分方程工具箱求解。

偏微分方程工具箱还可以解下列形式的偏微分方程组，在解方程组之前必须先选择菜单 Options/Application/Generic System，以选择专用于此问题的特定解法，

$$\begin{cases}-\nabla\cdot(c_{11}\nabla u_1)-\nabla\cdot(c_{12}\nabla u_2)+a_{11}u_1+a_{12}u_2=f_1\\-\nabla\cdot(c_{21}\nabla u_1)-\nabla\cdot(c_{22}\nabla u_2)+a_{21}u_1+a_{22}u_2=f_2\end{cases}$$

此时c是个2×2的矩阵形式的函数。

对于高阶的方程组系统，可以用指令求解。对于椭圆型方程，可以用自适应网格算法求解，这种方法也能与非线性解法联合使用。此外，对于泊松方程有一

种利用矩形网格的快速算法。这些方法既可以通过在菜单 Solve/Parameters 下的选项来实现，也可以用指令 adaptmesh，pdenonlin，poisolv 来实现。

§7.4.3 边界条件

解方程所需要的边界条件可以是以下两种之一：

狄利克雷 (Dirichlet) 边界条件 $hu = r$

广义诺伊曼 (Generalized Neumann) 边界条件 $\boldsymbol{n}\cdot(c\nabla u) + qu = g$

这里，$\boldsymbol{n}$ 为边界外法向单位向量，g，q，h，r 是在边界上定义的复函数。对于本征值问题，仅限于齐次条件即 $g = 0$，$r = 0$。对于非线性问题，系数 g，q，h，r 可以与 u 有关。对于抛物型方程和双曲型方程，系数可以与时间 t 有关。

在数学物理方法教材中，狄利克雷边界条件也叫第一类边界条件，而广义诺伊曼边界条件则称之为第三类边界条件，如果 $q = 0$ 则叫第二类边界条件。

对于方程组而言，狄利克雷边界条件是

$$h_{11}u_1 + h_{12}u_2 = r_1$$
$$h_{21}u_1 + h_{22}u_2 = r_2$$

而广义诺伊曼边界条件是

$$\boldsymbol{n}\cdot(c_{11}\nabla u_1) + \boldsymbol{n}\cdot(c_{12}\nabla u_2) + q_{11}u_1 + q_{12}u_2 = g_1$$
$$\boldsymbol{n}\cdot(c_{21}\nabla u_1) + \boldsymbol{n}\cdot(c_{22}\nabla u_2) + q_{21}u_1 + q_{22}u_2 = g_1$$

混合边界条件则是

$$h_{11}u_1 + h_{12}u_2 = r_1$$
$$\boldsymbol{n}\cdot(c_{11}\nabla u_1) + \boldsymbol{n}\cdot(c_{12}\nabla u_2) + q_{11}u_1 + q_{12}u_2 = g_1 + h_{11}\mu$$
$$\boldsymbol{n}\cdot(c_{21}\nabla u_1) + \boldsymbol{n}\cdot(c_{22}\nabla u_2) + q_{21}u_1 + q_{22}u_2 = g_2 + h_{12}\mu$$

其中求得的 μ 要使得狄利克雷边界条件得到满足。在有限元法中，狄利克雷边界条件也叫基本边界条件，诺伊曼边界条件也叫自然边界条件。

§7.4.4 可解问题的分类

偏微分方程工具箱将这些方程细分成8种类型来设置解法，以便于规定其中不同系数的物理含义以及根据不同问题输出不同的图形。

Generic Scalar	一般标量场
Generic system	一般的偏微分方程组
Structual Mechnics,Plane Stress	结构力学，平面应力场
Structual Mechnics,Plane Strain	结构力学，平面应变场
Electrostatics	静电场
Magnetostatics	静磁场

Ac Power Electromagnetics	交流电磁场
Conductive Media DC	直流导电介质问题
Heat Transfer	热传导问题
Diffusion	扩散问题

每种特定解法在调用以后呈现的对话框都有所不同，在对话框中对方程系数、边界条件和方程的解都用物理概念都作了说明。在这些对话框中，用户可以设置一些特殊的参数如结构力学中的杨氏模量，相关的物理量也能可视化。

§7.4.5 解题步骤

方程求解可分为三步，每步又包含几个模式 (Mode)，分述如下：

1. 设置定解问题

Draw Mode　画求解区域如矩形椭圆，多边形以及其组合。

Boundary Mode　定义边界条件。

PDE Mode　定义偏微分方程，即给定方程的类型及其系数。

2. 解方程

Mesh Mode　将区域分割为三角形网格。

Solve Mode　设置初始条件并求解，本征值问题可设搜索本征值范围。

3. 将结果可视化并输出

Plot Mode　用彩图、高度图、矢量场图，曲面图、网线图、等值线图和箭头图表现解，对抛物型方程和双曲型方程，可以用动画。

以下几节将对三类方程的求解给出一些实例。更多例子可以参看《数学物理方程的MATLAB解法与可视化》。

§7.4.6 解一维与三维问题

偏微分方程工具箱是专门设计用来解二维空间的问题，是否可以用它解决一维与三维空间的问题呢?

对于一维问题，如果巧妙地设置边界条件，也可以用 PDETOOL 求解，后面会提供一些实例。

三维问题的解决方法有几种，如降维法，如果三维空间的变化可以用二维方程来解释，那么问题就可以简化成二维空间的。例如，平面应力分析就可以通过修改材料的参数使其降维。

在某些问题中，如果第三维 (z) 的变化可以忽略，那么所有的 z 的导数都可以去掉，所得的二维方程实际上和三维方程会有完全一样的单位和系数。

还有就是，对有厚度的板可以处理成沿厚度的积分。所得到的是关于 z 方向的平均解 $D(x,y)$ 的二维方程，方程中的所有系数 c, d, a, f 都要乘以厚度。

例如想从焊接在一起的不同厚度的一堆薄板中计算其中某一片板上的应力，就可以用 $D(x,y)$ 乘以杨氏模量 E，体积力和表面张力。这种做法也适用于其他的关于厚板的例子和应用模式。

§7.4.7 拉普拉斯方程与泊松方程

例 1 带正电荷的云与大地之间会形成均匀的电场，平行于大地的电缆相当于一根无限长导体。在平行于电场的方向作垂直于电缆的截面，求截面上的电势。

用 PDETOOL 的解画的图形如图 7.18，实线是等电势线，箭头表示该处电场的大小与方向。

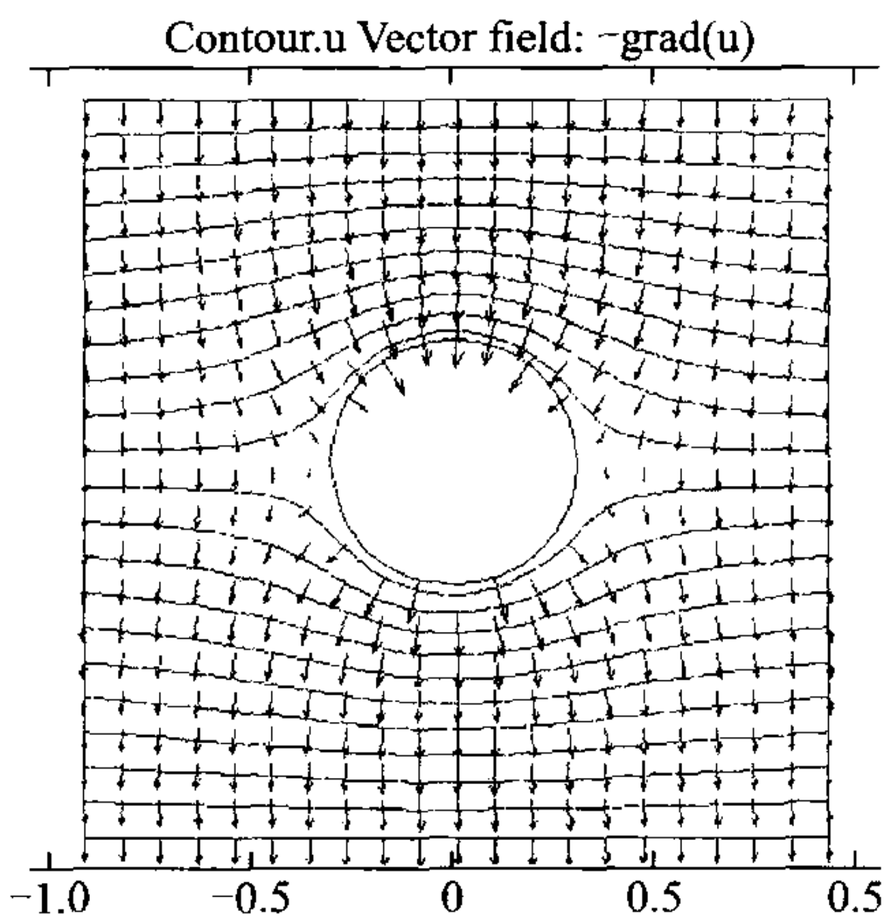


图 7.18 平行于大地的电缆的电场

求解过程是：画一个中心在原点、大小为 2×2 的正方形 SQ1，在正方形中间再画一个圆 C1，圆心在 (0,0) 处，半径为 0.3。在 Set formula 框中的公式要写成 SQ1-C1。

矩形所有的边界条件是狄利克雷边界条件，都取相同的 $h=1$，$r=y$，而圆的边界设为 $h=1$，$r=0$。

将方程的类型选为椭圆型 (Eliptic)，方程的系数设为 $c=1$，$a=0$，$f=0$。

网格初始化以后还要作一次细分。作图的选项为 Contour 和 Arrrows。

例 2 在矩形区域 $0<x<a$，$-b/2<y<b/2$ 中求解 $\Delta u=-x^2y$，u 在边界上的值为零。

定解问题是

$$\begin{cases}\Delta u=-x^2y\\ u(x=0)=0,\quad u(x=a)=0\\ u(y=-b/2)=0,\quad u(y=b/2)=0\end{cases}$$

这个问题可以用解析方法求解，它的解析解是[①]

$$u=\frac{xy}{12}(a^3-x^3)+\sum_{n=1}^{\infty}\frac{a^4b[(-1)^n n^2\pi^2+2-2(-1)^n]}{n^5\pi^5\sinh(n\pi b/2a)}\sinh\left(\frac{n\pi y}{a}\right)\sin\left(\frac{n\pi x}{a}\right)$$

为了与 PDETOOL 的结果对比，先将解析解画出。下面的程序可以画出 u 的图像，程序中取 $a=5$，$b=5$。

```
[X,Y]=meshgrid(0:0.1:5,-5/2:0.1:5/2);
Z1=0;
for n=1:10
    Z2=5^4*5*((-1)^n*n^2*pi^2+2-2*(-1)^n)*...
    sinh(n*pi.*Y/5).*sin(n*pi.*X/5)/...
    (n^5*pi^5*sinh(n*pi*5/10));
    Z1=Z1+Z2;
end
Z=Z1+X.*Y.*(5^3-X.^3)/12;
colormap(hot);
mesh(X,Y,Z)
view(119,7)
```

所得的图形是图 7.19。

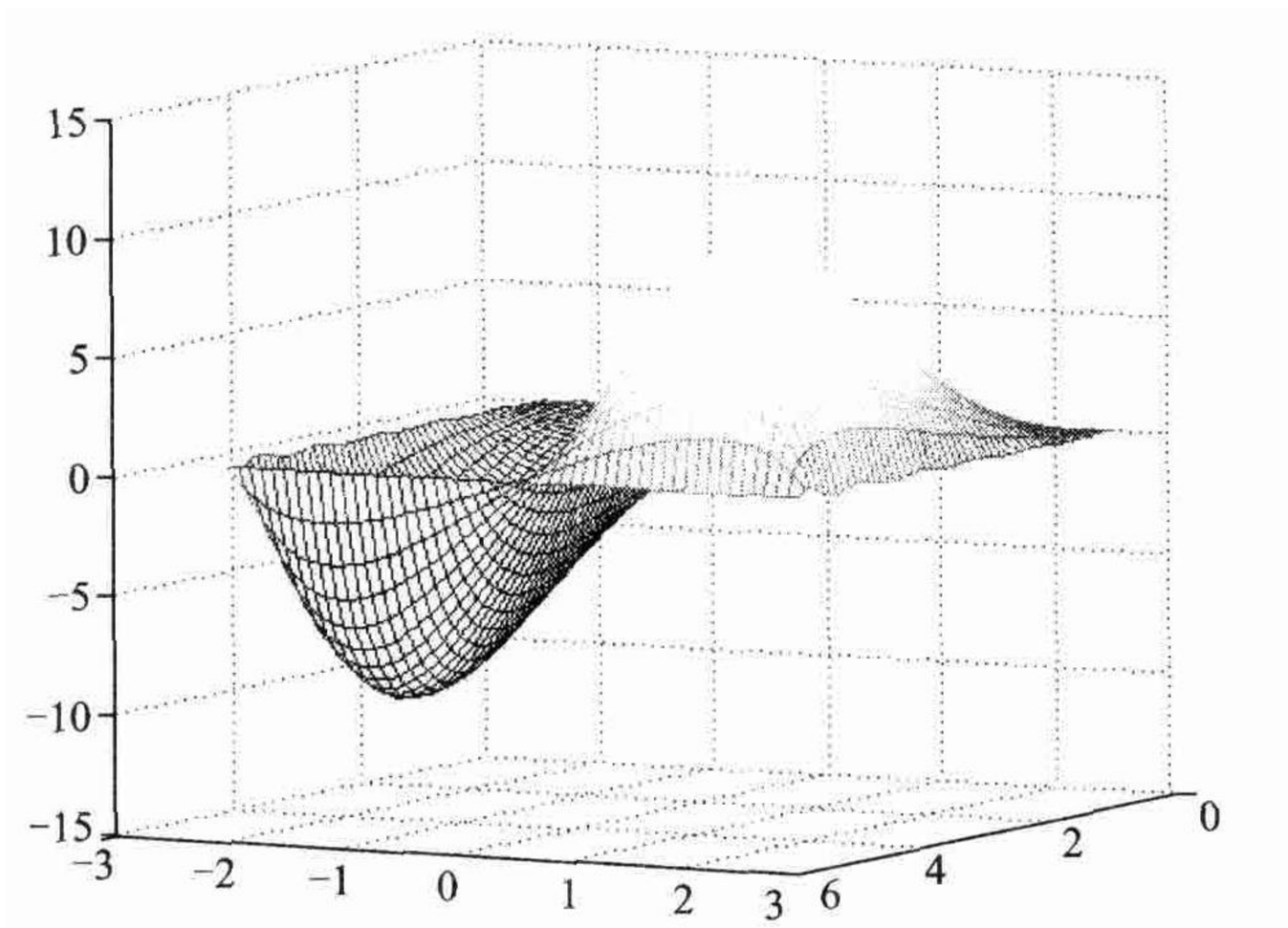


图 7.19 解析解的图形

再用偏微分方程工具箱求解这个问题。在 Options/Axes Limits 下选择 x 轴范围为 $0\sim5$，y 轴范围为 $-2.5\sim2.5$。画一个矩形，矩形的顶点为 $(0,-2.5)$，$(5,$

① 梁昆淼. 数学物理方法. 3 版. 北京: 高等教育出版社, 1998: 223

$-2.5)$, $(5, 2.5)$, $(0, 2.5)$。

按照题意，矩形的四个边界都是狄利克雷边界条件，而且是齐次的边界条件，可取 $h=1$, $r=0$。

方程的设置是 Elliptic 型，也就是 `c=1,a=0,f= - x.^2 .*y.`

为了有足够的精度，将初始化的网格作了两次细分。而作图的选项为 Color 和 Hight(3D_ plot)，同时在 colormap 中选择 hot，所得的图形如图 7.20。

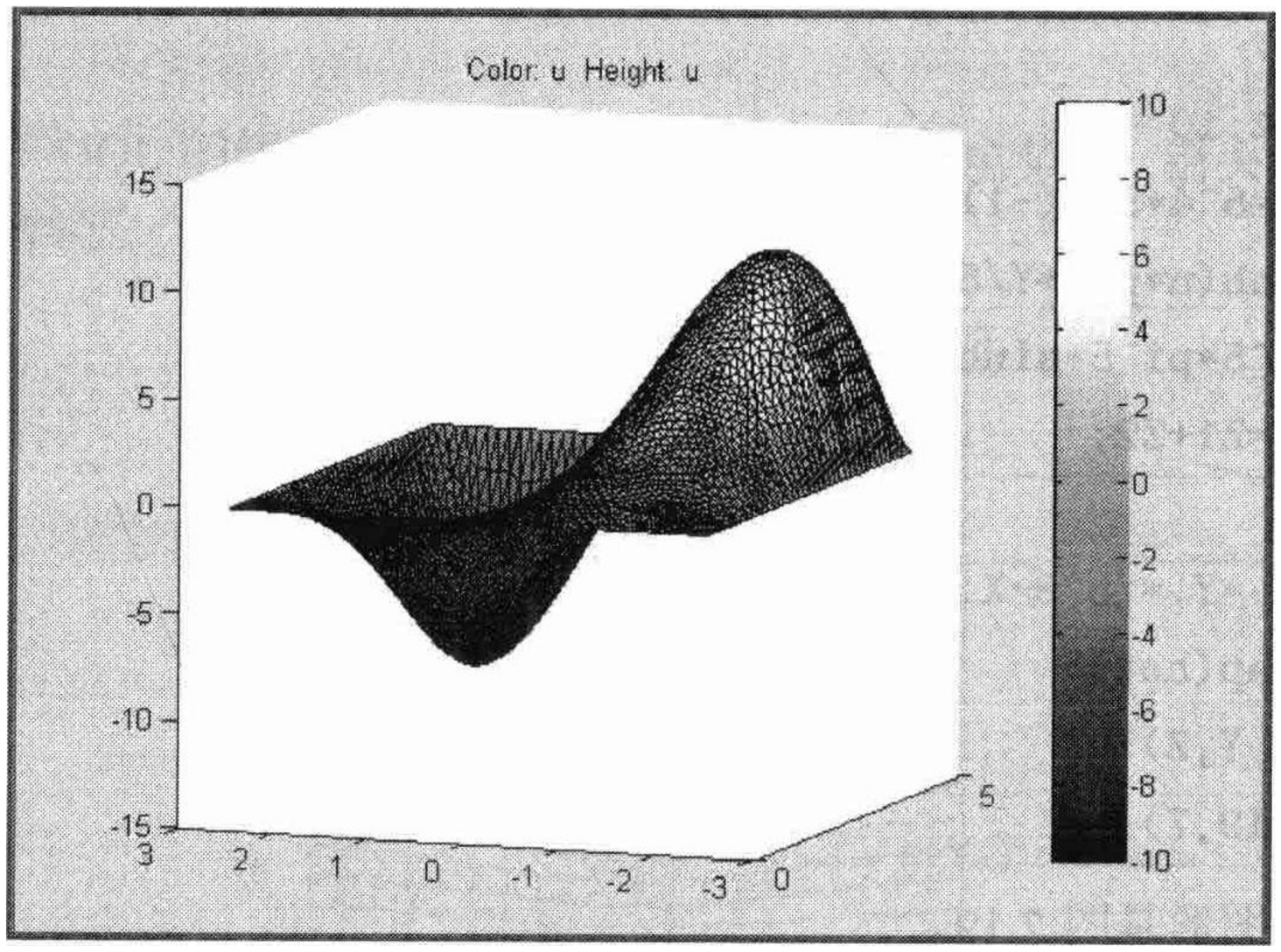


图 7.20 偏微分方程工具箱解得的结果

§7.4.8 热传导方程

例 3 讨论有限长的细杆在第一类边界条件下的热传导问题。定解问题是

$$\begin{cases} u_t = a^2 u_{xx} \\ u(0,t)=0, \quad u(l,t)=0 \\ u(x,t=0)=\varphi(x) \end{cases}$$

其中 $0 \leqslant x \leqslant 20$，即 $l=20$，取 $a=10$ 且

$$\varphi(x) = \begin{cases} 1 & (10 \leqslant x \leqslant 11) \\ 0 & (x<10, x>11) \end{cases}$$

这个问题已经用差分方法解过。现在再用PDETOOL来解这个问题。

求解区域为 $-10 \leqslant x \leqslant 10, -0.2 \leqslant y \leqslant 0.2$，矩形的上下边界为诺伊曼边界条件，取 $q=0$, $g=0$，左右边界为狄利克雷边界条件，取 $h=1$, $r=0$。

方程类型应该选抛物型 (Parabolic)。

求解参数 time 取 0 : 0.005 : 0.2。

初条件 u(t0) 为$10*\tanh(10000*x)-10*\tanh(10000(x-0.05))$。

这个初始温度分布近似于定解问题中初始条件。这个设置是为了用函数而不是用语句来描述初始温度。

作图选择的参数为 Color，Heigh(3-D plot) 和 Animation。这也是动画演示，图 7.21 是其中的一个画面。

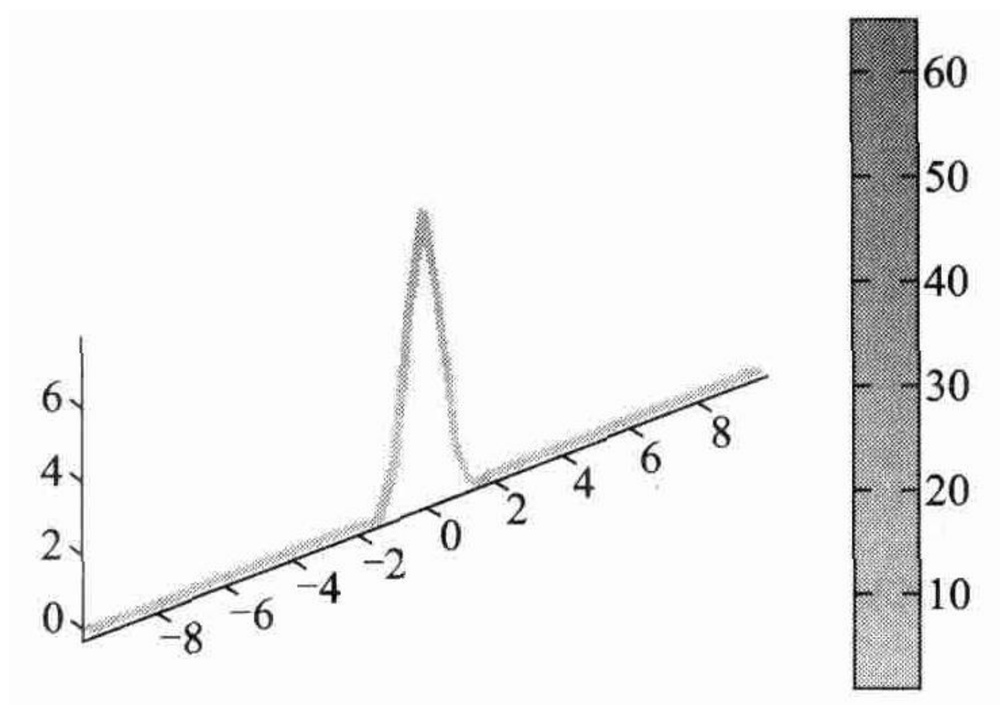


图 7.21 有限长杆热传导的偏微分方程工具箱解

为了对比，再画一下解析解的图形。它的解析解为

$$u(x,t)=\sum_{n=1}^{\infty}\frac{2}{n\pi}\left[\cos\frac{10n\pi}{20}-\cos\frac{11n\pi}{20}\right]\mathrm{e}^{-\frac{n^2\pi^2a^2}{400}t}\sin\frac{n\pi}{20}x$$

画出前 50 个分波的合成图形，在程序中用子函数来计算各个分波，然后在主函数中调用。所用的程序是：

```
function    jxj
N=50;  t=1e-5:0.00001:0.005;        x=0:0.21:20;
w=rcdf(N,t(1));
h= plot(x,w,'linewidth',5);
axis([ 0, 20, 0, 1.5])
for n=2:length(t)
    w=rcdf(N,t(n));
    set(h,'ydata',w);
    drawnow;
end

function u=rcdf(N,t)
```

```
x=0:0.21:20;   u=0;
for k=1:2*N
   cht=2/k/pi*(cos(k*pi*10/20)-cos(k*pi*11/20))*sin(k*pi*x./20);
   u=u+cht*exp(-(k^2*pi^2*10^2/400*t));                    %a=10,l=20
end
```

图 7.22 是其中的一个画面。

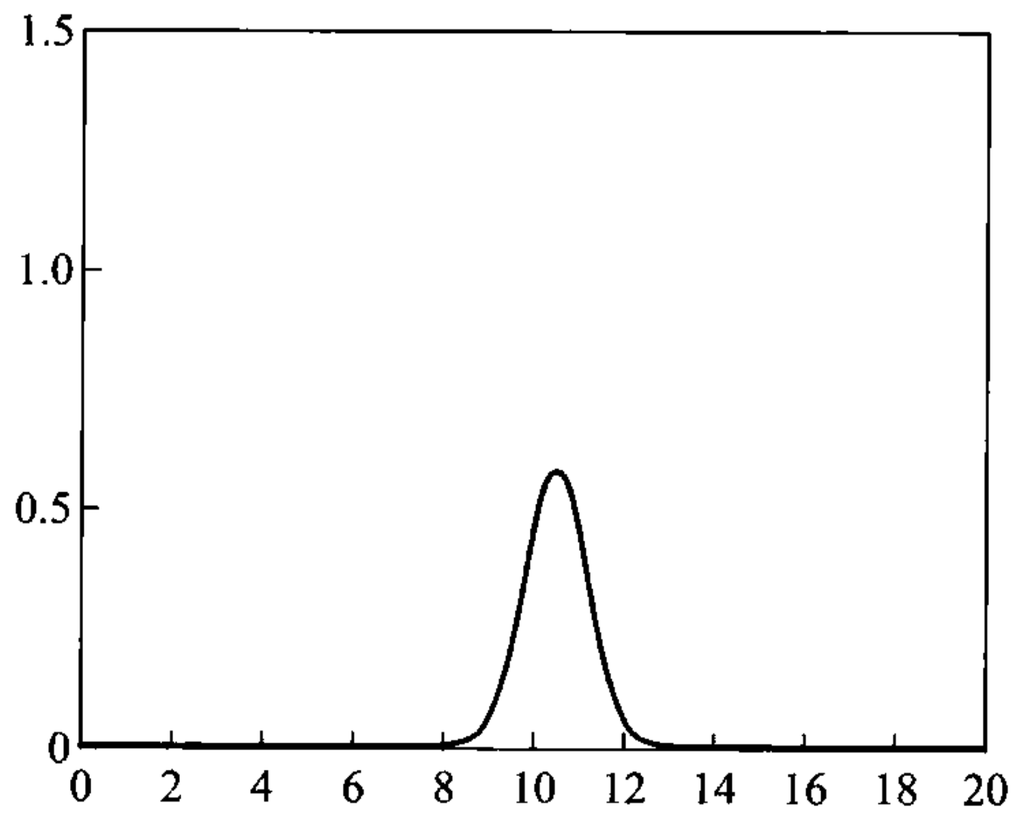


图 7.22 有限长杆热传导的解析解的动画的一个画面

§7.4.9 波动方程

一维弦的运动方程是

$$u_{tt} = a^2 u_{xx}$$

两端固定的弦有

$$u(0,t) = 0, \quad u(l,t) = 0$$

初条件为

$$\phi = \exp(-(x-1)^2 * \log10(2)/0.1^2), \qquad \psi = 0$$

用 PDETOOL 研究它的运动。

在菜单 Options/Axes Limits 下选择 x 轴范围为 $0 \sim 2$，y 轴范围为$-0.2 \sim 0.2$。以原点为顶点画一个长为 2 宽为 0.3 的矩形，矩形的顶点为 $(0,-0.15)$，$(2,-0.15)$，$(0,0.15)$，$(2,0.15)$。

矩形的左右边界是齐次的狄利克雷边界条件，可取 $h=1$，$r=0$，上下边界则取齐次的诺伊曼边界条件，即 $g=0$，$q=0$。

方程的设置是 hyperbolic 型，系数是 $c=1$，$a=0$，$f=0$，$d=1$。

为了有足够的精度，将初始化的网格作了两次细分。在解方程的参数设置对话框 Solve Parameters 中，将 Time 设为 0 : 0.05 : 3.6，u(to) 设为 `exp(-(x-1).^2.*log10(2)/0.1^2)` 而 u′(to) 设为 0，其余不变。

而作图选项对话框 Plot Parameters 中，各项选择如下，在 Plot type 下，选 Color，Height(3-D Plot)，Animation，所得的图形如图 7.23。

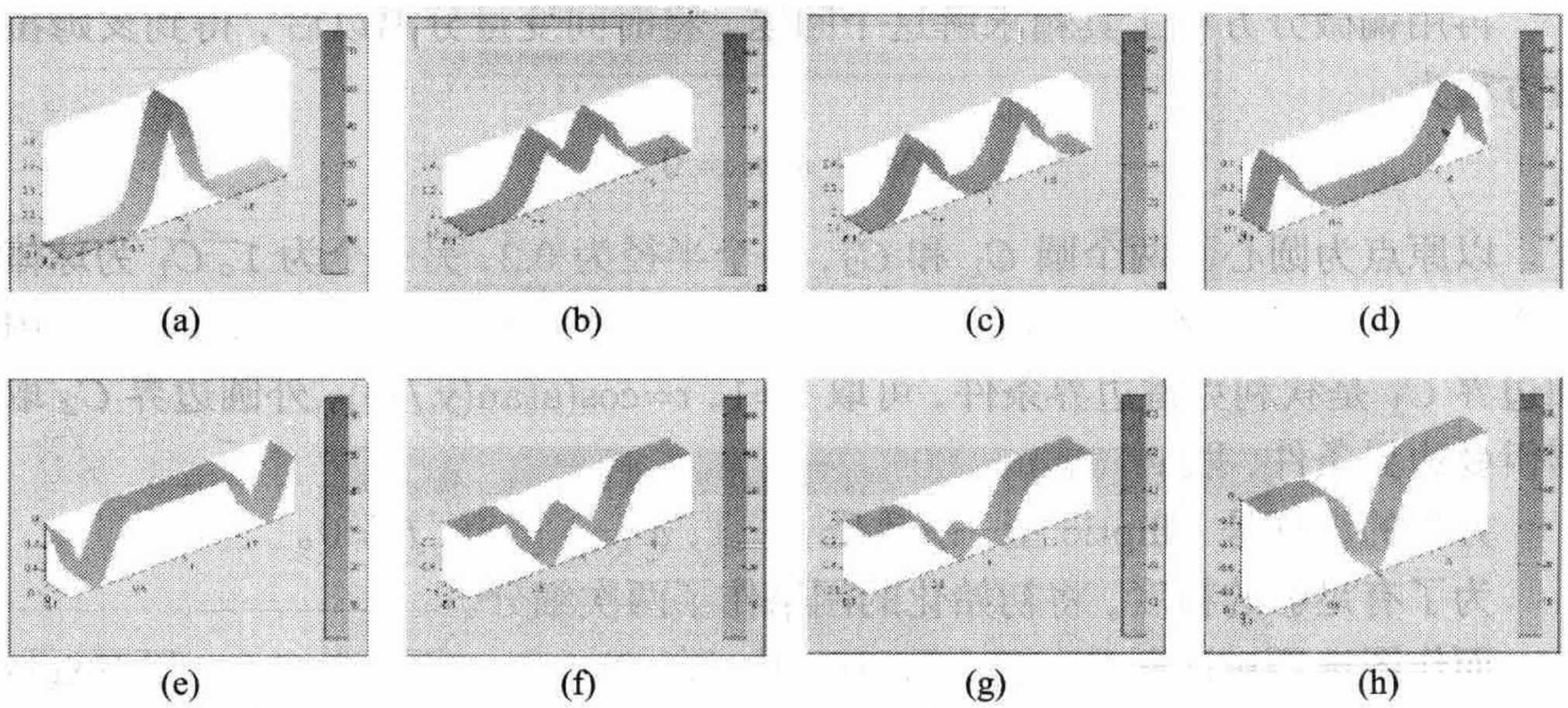


图 7.23 有半波损失的弦振动

将矩形的左右边界设为齐次的诺伊曼条件，其余不变，可得到没有半波损失的图形为图 7.24。

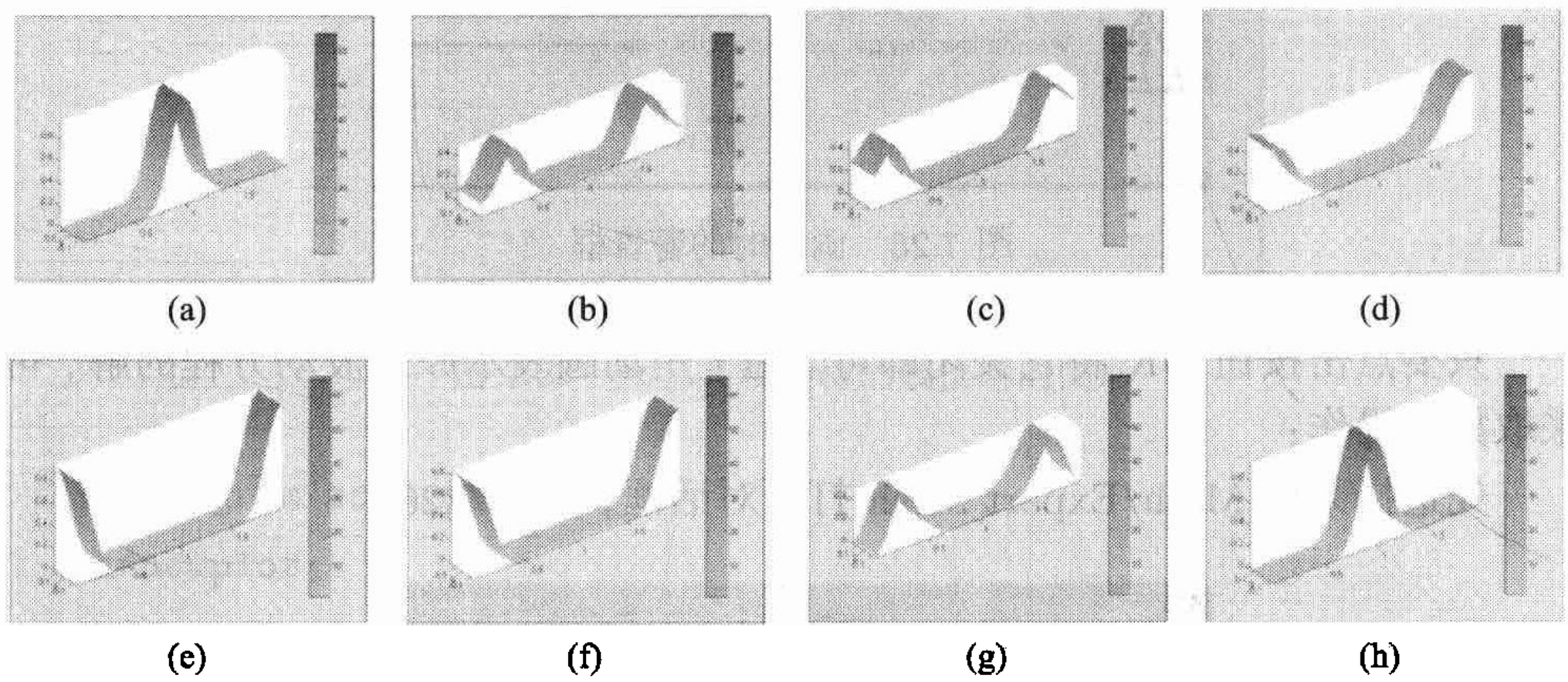


图 7.24 没有半波损失的弦振动

下面是一个三维的振动问题。

半径为 r_0 的球面径向速度分布为 $v = v_0 \cos\theta \cos\omega t$，试求解这球面所发射的稳恒声振动的速度势 u，设 r_0 远小于声波的波长 λ。[①]

用球坐标，极点取在球心，定解问题是

$$\begin{cases} u_{tt} - a^2 \Delta u = 0 \\ u_r(r = r_0) = v_0 P_1(\cos\vartheta) \mathrm{e}^{-\mathrm{i}\omega t} \end{cases}$$

① 梁昆淼. 数学物理方法. 3 版. 北京: 高等教育出版社, 1998: 369

在球面的边界条件是 $\cos\omega t$ 即为 $\mathrm{Re}(\mathrm{e}^{-\mathrm{i}\omega t})$，在上面写成了 $\mathrm{e}^{-\mathrm{i}\omega t}$，这要求在计算结果中也只取实部。

再用偏微分方程工具箱求解这个问题。将时间变量分离以后，得到亥姆霍兹方程为

$$\Delta u + k^2 u = 0$$

以原点为圆心画两个圆 C_1 和 C_2，一个半径为 0.2，另一个为 1。C_1 为球体的横截面，C_2 为求解区域的外边界。问题的求解区域是 C_2—C_1。按照题意，内圆边界 C_1 是狄利克雷边界条件，可取 h=1，r=cos(atan(y./x))，外圆边界 C_2 取诺伊曼边界条件，取 g=0，q=−60i。

方程的设置是 Elliptic 型，系数是 $c=1$，$a=-3600$，$f=0$。

为了有足够的精度，将初始化的网格作了两次细分。

而作图选项对话框 Plot Parameters 中，在 Plot type 下，选 Color 和 Height (3–D Plot)，其余不变。注意方程所得的解为复数，对于复数解，只有解的实部会画出来同时给出警告提示框，如图 7.25：

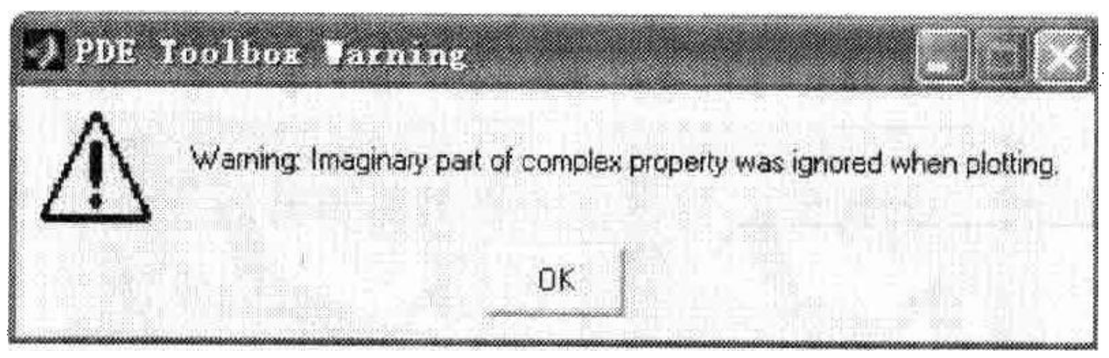


图 7.25 画图时的警告框

只要点击按钮 OK 将它关闭即可。为了用动画表示原来波动方程的解，可采取如下操作：

先选择菜单 Mesh/Export Mesh 打开对话框，如图 7.26：

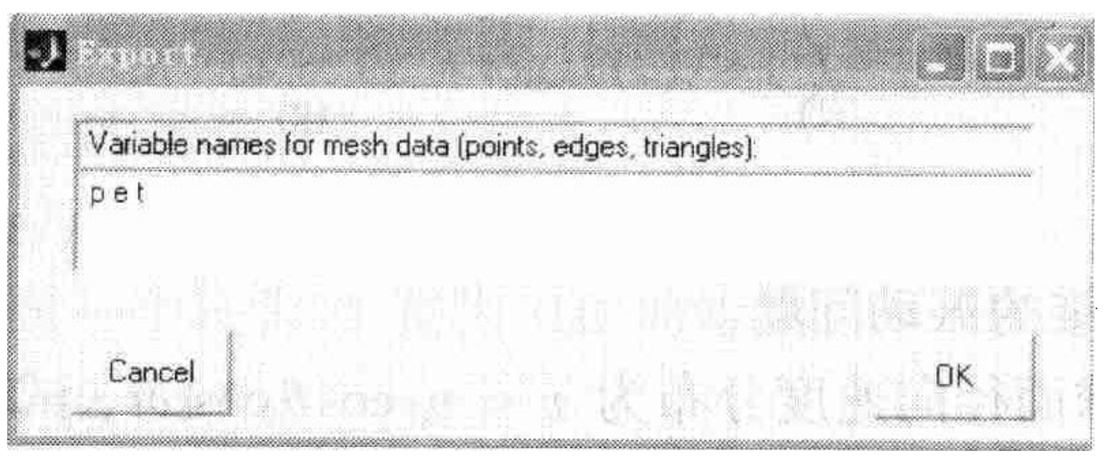


图 7.26 输出网格数据的警告框

点击 OK 按钮将网格数据 p，e，t 输出到指令窗口，关闭对话框。再选择菜单 Solve/Export Solution 将打开对话框，如图 7.27：

点击 OK 按钮将解u输出到指令窗口，关闭对话框。然后运行下面的程序文件 sswt.m，就可以看到动画演示的解，如图 7.28。这时，偶极声源形成的声波由

声源向外传播。

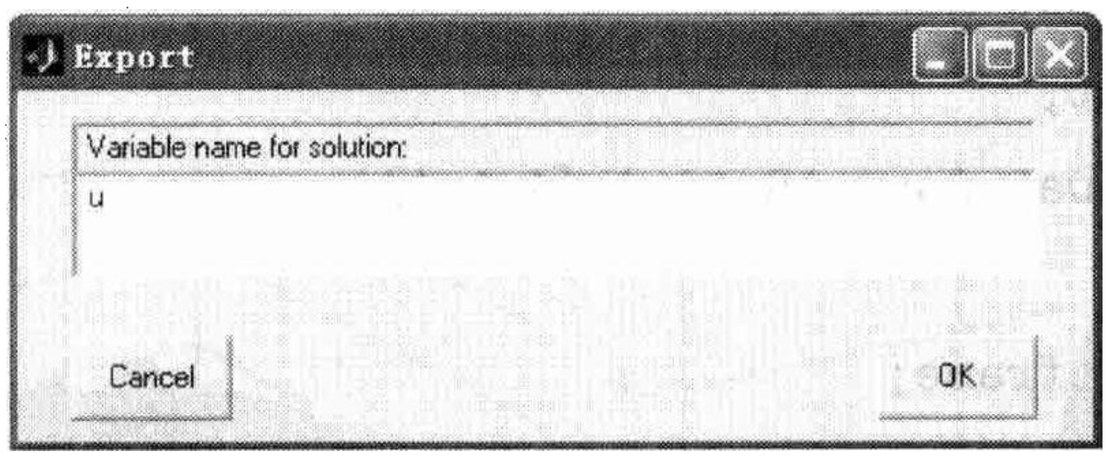


图 7.27 输出解的对话框

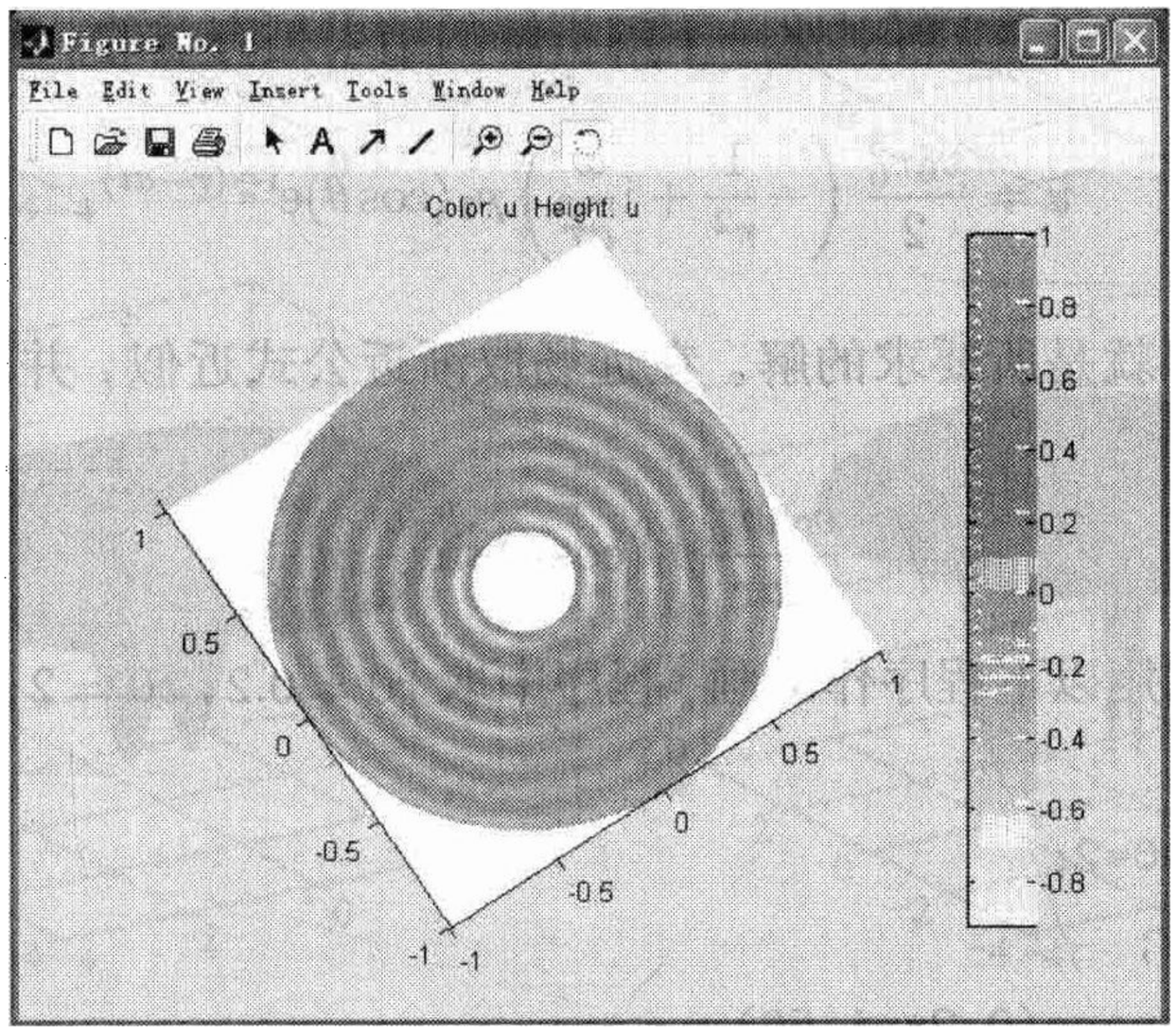


图 7.28 用工具箱所得的数值解的图形

```
% program sswt.m
h=newplot;
hf=get(h,'Parent');
set(hf,'Renderer','zbuffer')
axis tight
set(gca,'DataAspectRatio',[1 1 1]);
axis off
M=moviein(10,hf);
maxu=max(abs(u));
colormap(cool)
for j=1:10,
   ur=real(exp(-j*2*pi/10*sqrt(-1))'u);
```

```
    pdeplot(p,e,t,'xydata',ur,'colorbar','off','mesh','off');
    caxis([-maxu maxu]);
    axis tight,
    set(gca,'DataAspectRatio',[1 1 1]);
    axis off
    M(:,j)=getframe;
end
movie(hf,M,10);
```

这个问题相对复杂，再对比一下解析解的结果是有益的。问题的解析解是

$$u=\frac{v_0r_0^3}{2}\left(-\frac{1}{r^2}+\mathrm{i}\frac{\omega}{ar}\right)p_1(\cos\theta)\mathrm{e}^{\mathrm{i}\frac{\omega}{a}(r-at)}$$

上式的实部就是所要求的解。在远场取渐近公式近似，并取实部，得到的解为

$$u=-\frac{v_0\omega r_0^3}{2ar}p_1(\cos\theta)\sin\frac{\omega}{a}(r-at)$$

解析解可以用以下程序作动画，程序中取 $r0=0.2$；$v0=2$；$k=60$；$a=2$。

```
clear
r0=0.2;  v0=2;
k=60;  a=2;
theta=linspace(0,2*pi,50);
rho=0.2:0.1:4;
[Th,Rh]=meshgrid(theta,rho);
[X,Y]=pol2cart(Th,Rh);
rh=sqrt(X.^2+Y.^2);
th=atan(Y./X);
for t=0:0.001:0.03
u=real(v0/2*r0^3*(-1./rh.^2+i*k./rh).*...
   cos(th).*exp(k*(rh+2*t)*i));
surf(X,Y,u)
view(-32,28)
pause(0.5)
end;
```

图 7.29 是动画中的几幅画面，这是由球面向外传播的球面波。虽然从远场近似解的表达式可以看出远场的极化现象，也就是声波传播具有很明显的两极

的方向性，但是从近场的表达式是看不出这一点来，将近场的解析解画出图形以后，它的极化现象其实更显著。

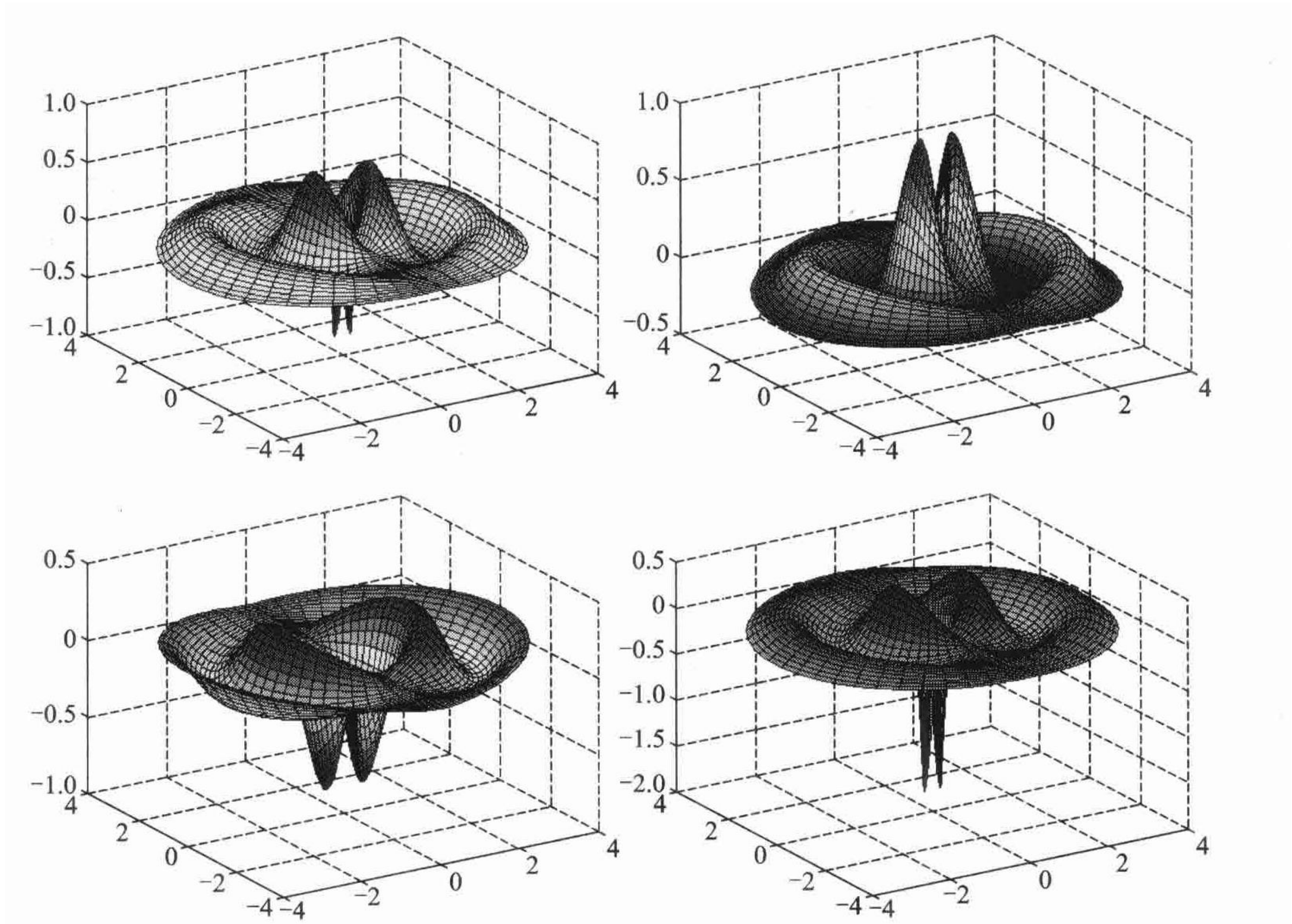


图 7.29 解析解的表面图

图 7.29 是表面图的动画演示，也可以用等值线来作动画演示，如图 7.30 所示。只要将程序中下列两句

```
surf(X,Y,u);   view(-32,28)
```

换成

```
contour(X,Y,u);  axis([-4 4 -4 4]);  axis square
```

即可。图 7.30 就是动画中的几幅画面。

§7.4.10 平面区域的本征值问题

PDETOOL 解平面区域的本征值问题十分方便。下面以矩形和圆域作为例子介绍其用法。

1. 矩形的本征值问题

边长分别为 b, c 的四周固定的矩形膜的本征值问题是

$$u(x,y)_{xx} + u(x,y)_{yy} = \lambda u(x,y)$$

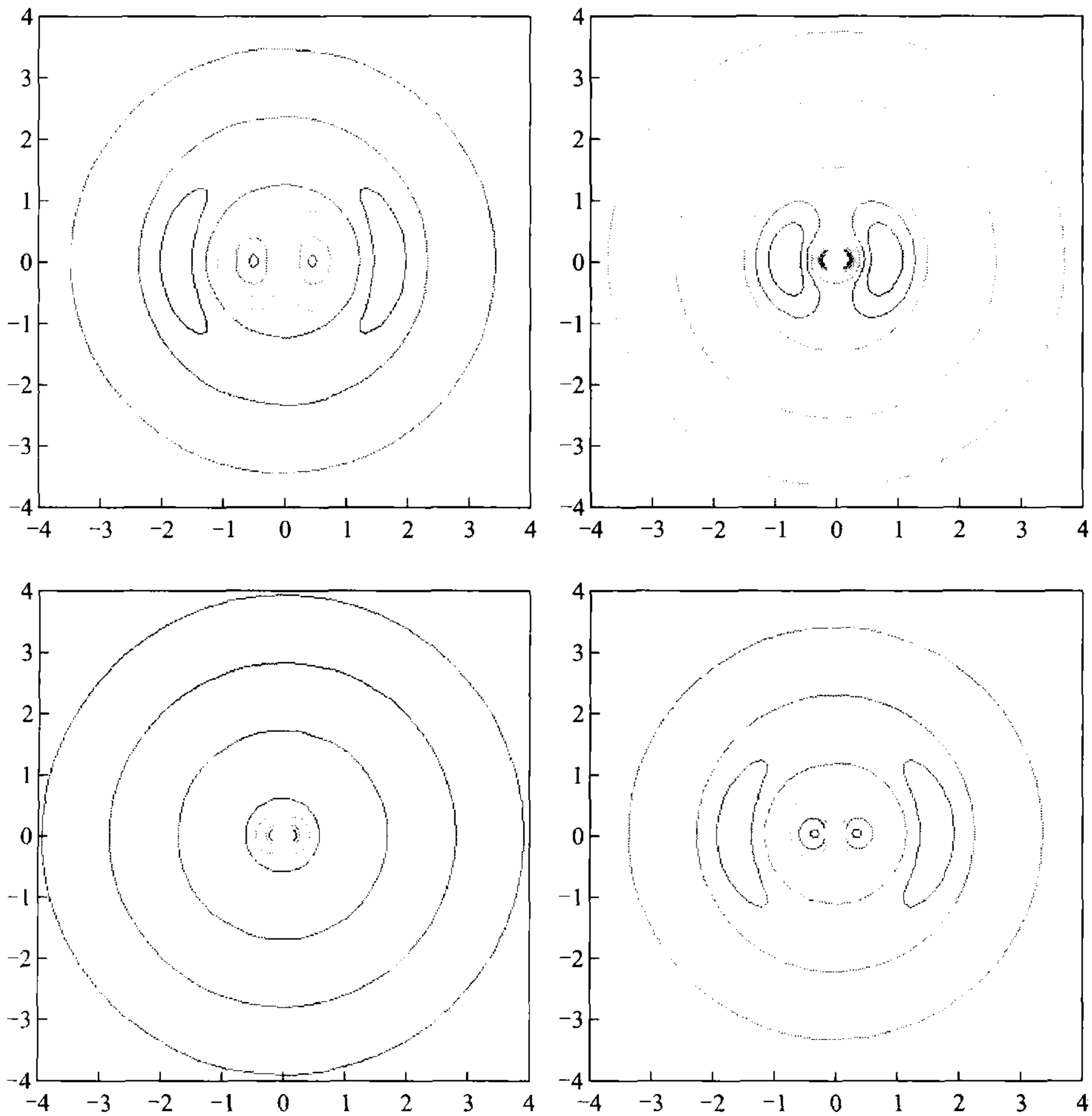


图 7.30 解析解等值线图

求解区域可以画一个任意的矩形，但为了看到更多的本征值图形，将区域设为$0 \leqslant x \leqslant 2$，$0 \leqslant y \leqslant 1$。为此，点击菜单Options /Axes Limits 将 x, y 轴的范围设为相同的范围。再用鼠标右键双击画好的矩形内部，就会打开对话框，如图 7.31 所示，按图中设置即可。

边界条件为四边固定，即正好是默认的设置，所以不必改变。方程取 Eigenmodes(本征模式)，其形式为

$$-\nabla \cdot (c\nabla u) + au = \lambda \nabla du$$

取 $c = 1$，$a = 0$，$d = 1$。

在选择菜单 Solve/Parameter 时打开的对话框，如图 7.32 所示。

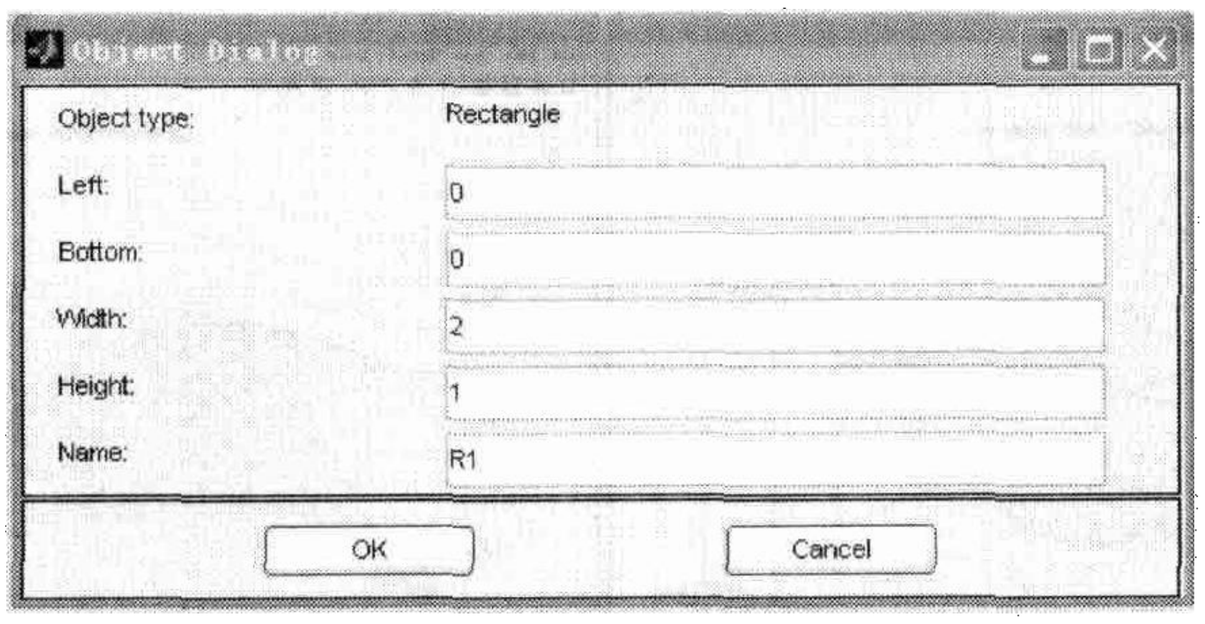


图 7.31 设置矩形位置的对话框

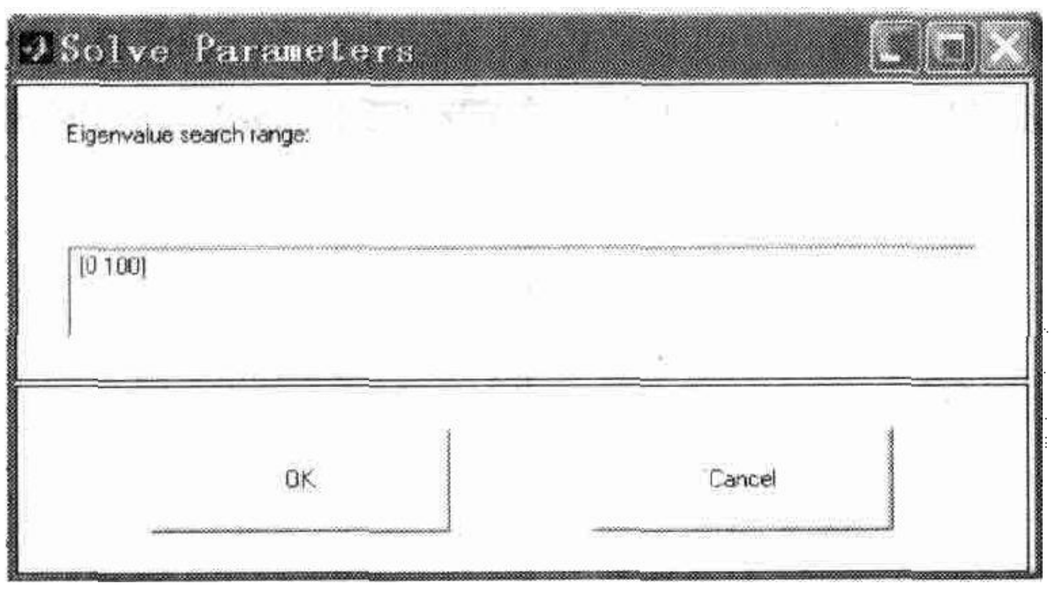


图 7.32 设置本征值搜索范围

在对话框中可设置本征值搜索范围，默认的范围是 0 ~ 100。一般可保持不变。画好三角形网格以后再细分一次以提高精度。点击按钮=进行计算就会看到一个本征值的图形。为了看到更多的本征值图形，在作图对话框选择 Hight (3−D plot)和Plot in x−y grid。注意在作图对话框中还有一个选择是 Eigenvalue(本征值)，即要求指定本征值以后再画本征函数的图形。图 7.33 是前四个本征函数的图形，相应的本征值都显示在图形上。

对于正方形区域，将会出现二重简并的情况，即相同的本征值可能对应两个不同的本征函数，不过这两个本征函数在图形上只是发生了一个转动。读者可以自己验证一下。

读者可以验证，这个方程的解析解的本征模是

$$X_m(x)Y_m(y) = \sin\frac{n\pi x}{c}\sin\frac{m\pi y}{b}$$

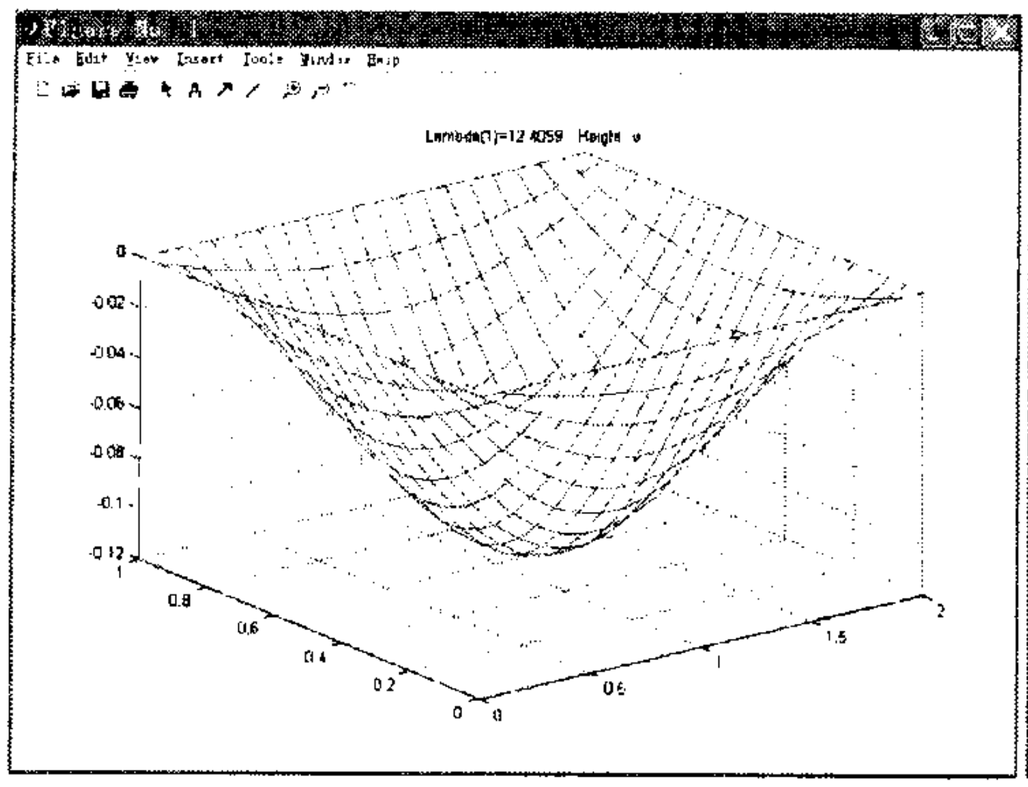
$$\lambda_{mn} = \left(\frac{m\pi}{b}\right)^2 + \left(\frac{n\pi}{c}\right)^2$$

它的图形与上面结果是一致的。

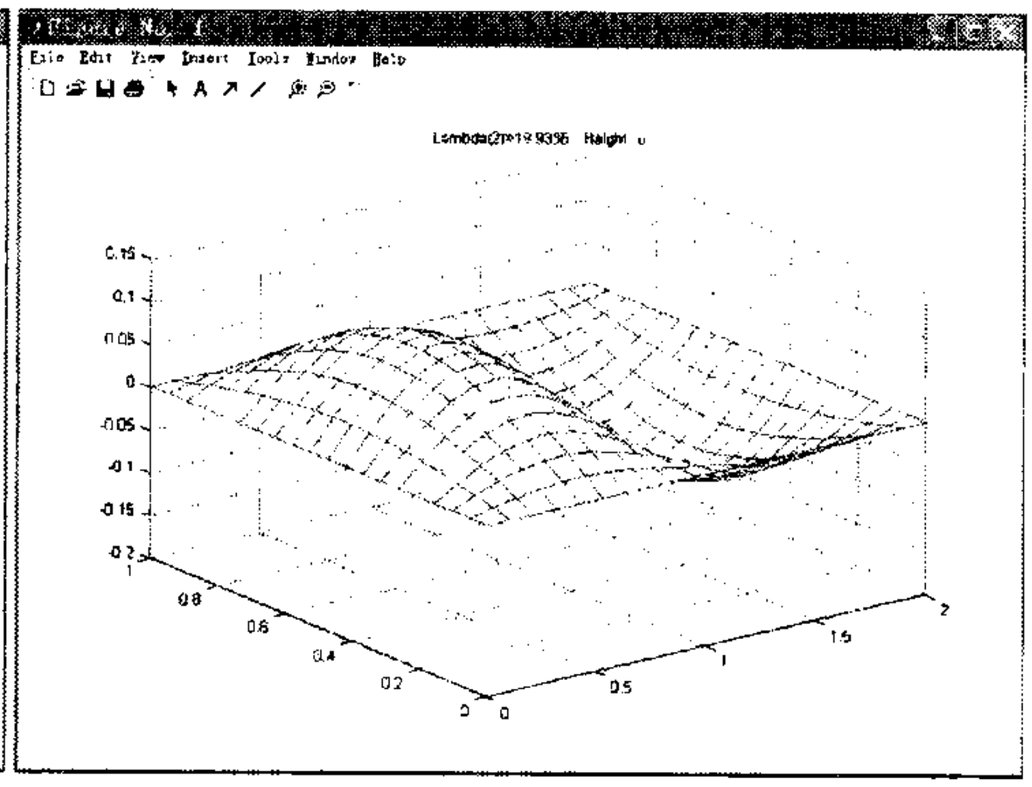
2. 圆域问题

半径为 ρ，周边固定的圆形膜的本征问题是

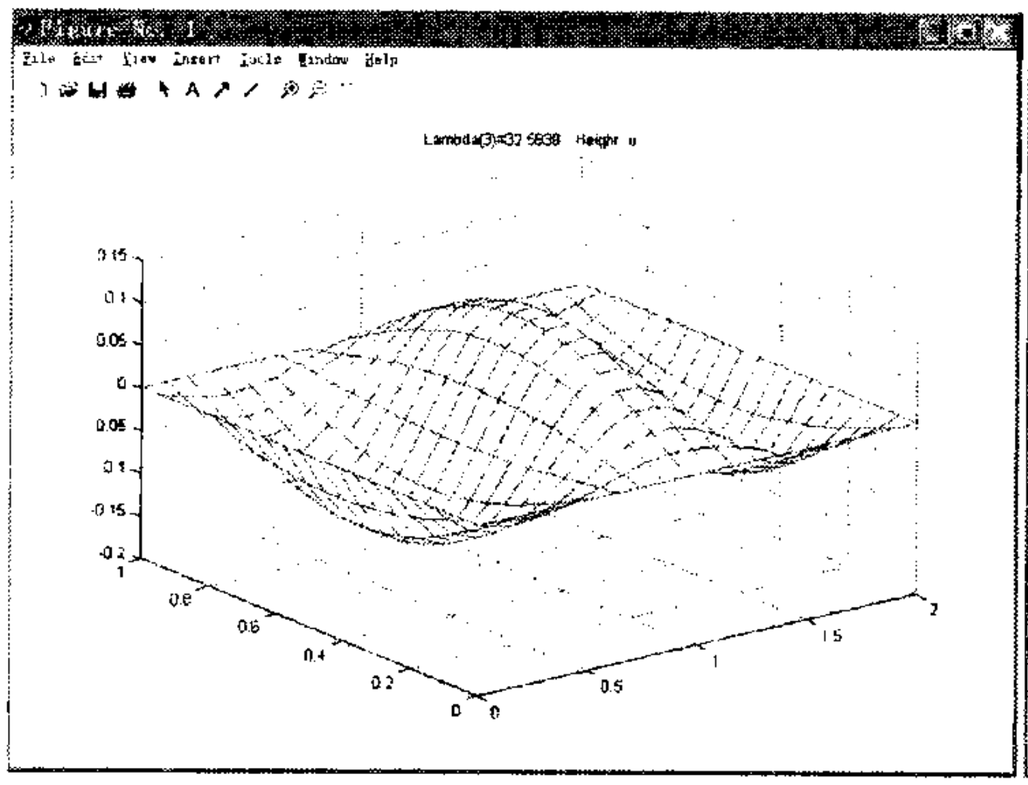
$$\Delta u(\rho,\varphi) = \lambda u(\rho,\varphi)$$



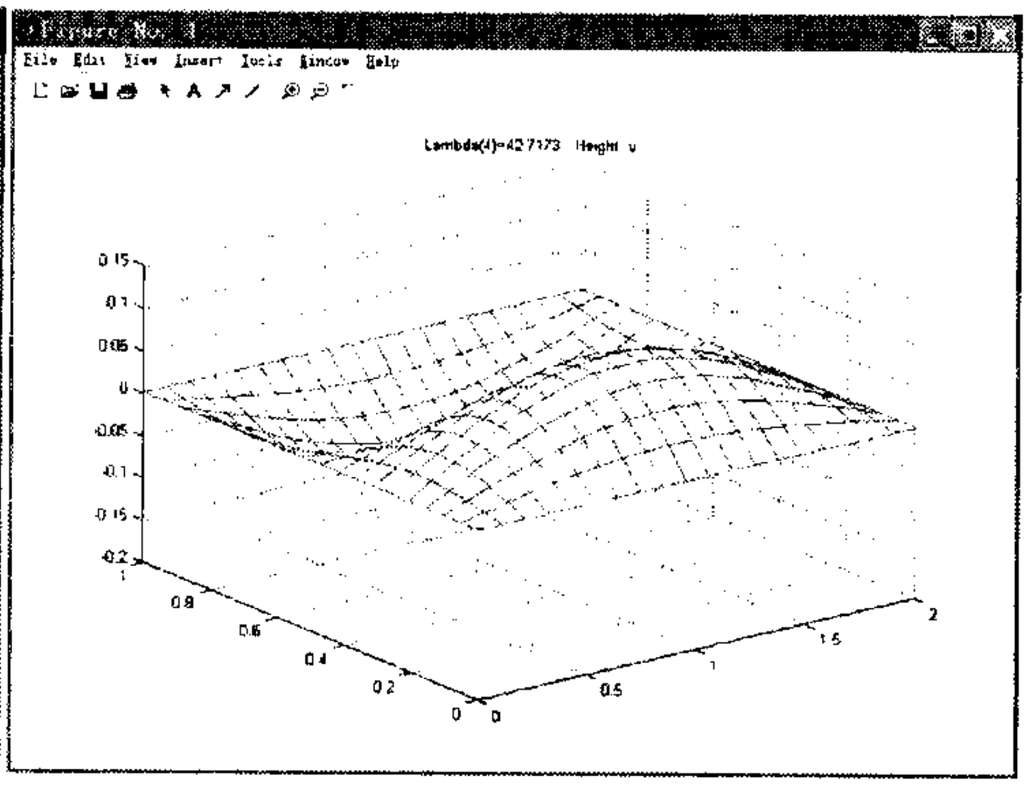
(a) 第一个本征函数



(b) 第二个本征函数



(c) 第三个本征函数



(d) 第四个本征函数

图 7.33

用偏微分方程工具箱来解这个问题步骤是

在指令窗口内键入 pdetool，打开工具箱的界面窗口。以原点为圆心，画一个半径为 1 的圆。选择方程类型为 Eigenmodes(本征模式)，方程的形式为

$$-\nabla \cdot (c\nabla u) + au = \lambda \nabla du$$

其中系数取 $c = 1$，$a = 0$，$d = 1$。边界条件为周边固定，即正好是默认的设置，所以不必改变。

网格分割以后要细分一次，以保证有足够的精度。

本征值搜索范围就用默认值是 $0 \sim 100$。

在作图对话框选择 Color，Hight (3−D plot) 和 Plot in x−y grid。Colormap 选 hsv 的效果更好。在 Eigenvalue(本征值) 中选择前 4 个不同的本征值作图，图 7.34 是前四个本征函数的图形，相应的本征值都显示在图形上。由于存在简并现象，所以会出现非常相近的本征值。

要更深入地了解这些本征函数的图形，需要与解析解的结果进行对比。

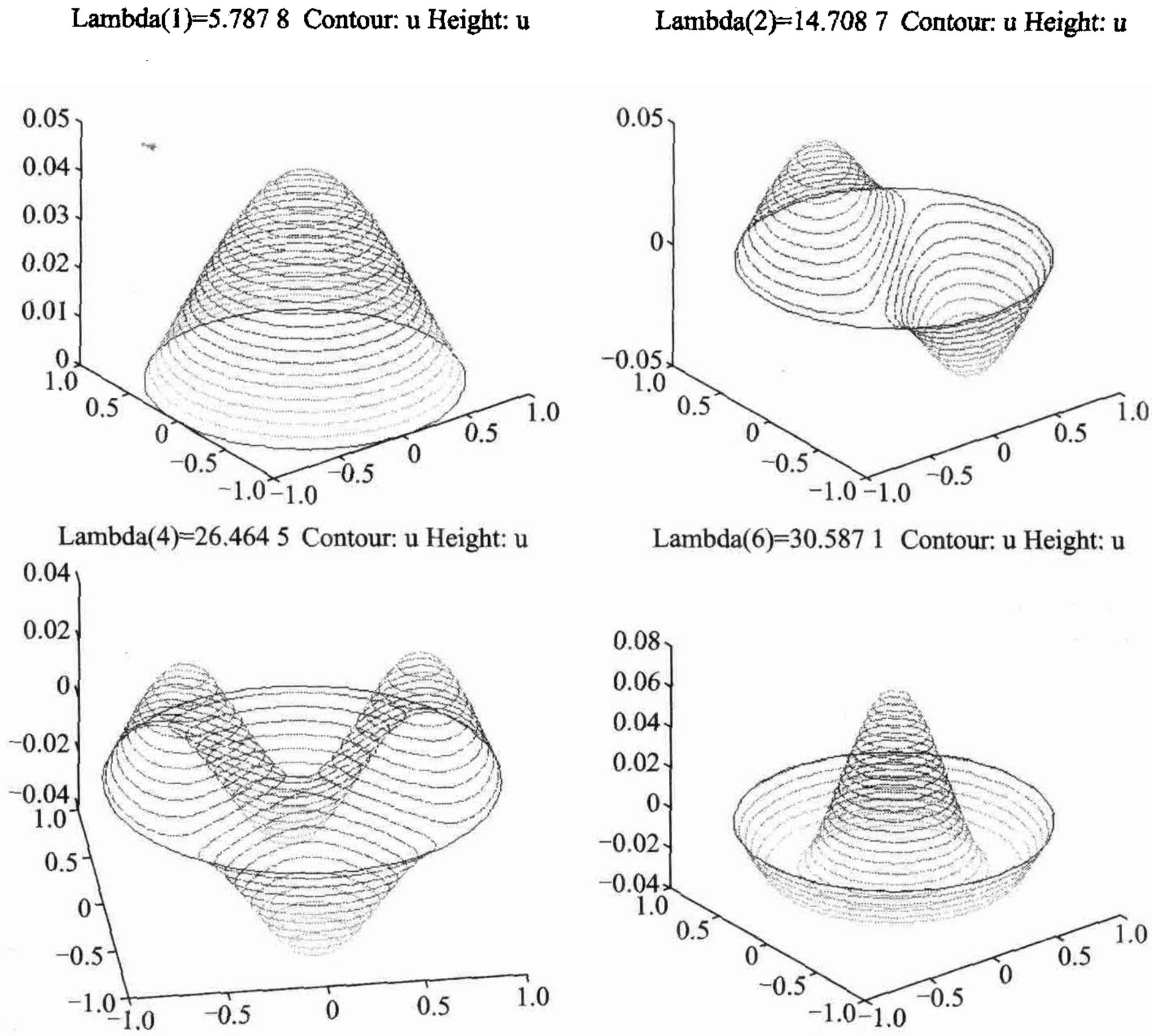


图 7.34 偏微分方程工具箱画出的圆域前 4 个本征函数

用解析方法求得的本征模是

$$J_m(\sqrt{\mu}\rho)\sin m\varphi,\quad J_m(\sqrt{\mu}\rho)\cos m\varphi$$

其中 $\mu=\dfrac{x_n^m}{\rho_0}(m=0,1,\cdots,n=0,1,2,\cdots)$，这里 x_n^m 是 m 阶贝塞尔函数的第 n 个零点。其中贝塞尔函数部分是区域 $[0,\rho_0]$ 之间的 ρ 的本征函数。贝塞尔函数的用法会在下面特殊函数一节中讲到。作为一个例子，我们讨论关于 ρ 的第一类齐次边界条件的本征值情形。

下面的程序画出 $m=0$，$n=0$，1，2，3 的贝塞尔函数前四个本征函数图形如图 7.35，其中取 $\rho_0=1$。

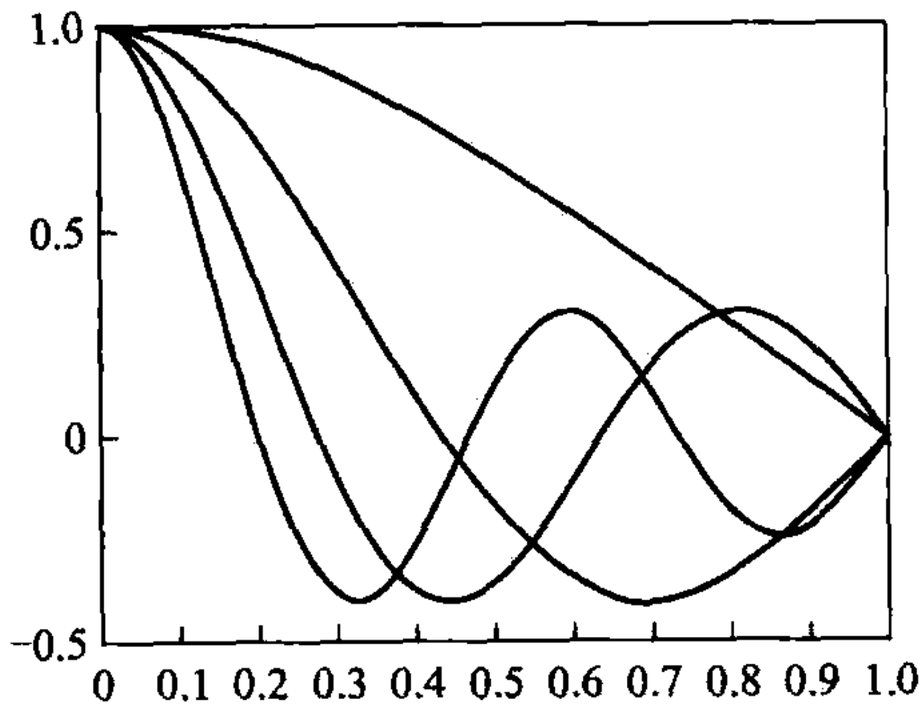


图 7.35 第一类齐次边界条件的贝塞尔函数
(m=0, n=0, 1, 2, 3) 的本征函数图形

```
r=0:0.01:1;
y=inline('besselj(0,x)','x');
x(1)=fzero(y,1);
plot(r,y(x(1)*r))
hold on
for k=1:3
   x(k+1)=fzero(y,x(k)+3);
   plot(r,y(x(k+1)*r))
end
```

φ 的本征函数有两种，即 $\sin m\varphi$，$\cos m\varphi$，φ 的本征函数和 ρ 的本征函数的组合形成了上述圆形区域的本征函数。

在圆形区域中 $m=0$ 的 4 个本征模如图 7.36，这时 φ 本征函数是 1。

所用的程序是：

```
r=0:0.01:1;
[XX,YY]=meshgrid(-1:0.01:1);
[Q, R]=cart2pol(XX,YY);
R(find(R>1))=NaN;
y=inline('besselj(0,x)','x');
x(1)=fzero(y,3)                                  %从x=3搜索第一个零点
meshc(XX,YY,y(x(1)*R))
for k=1:3
    x(k+1)=fzero(y,x(k)+2)
    figure
    meshc(XX,YY,y(x(k+1)*R))
end
```

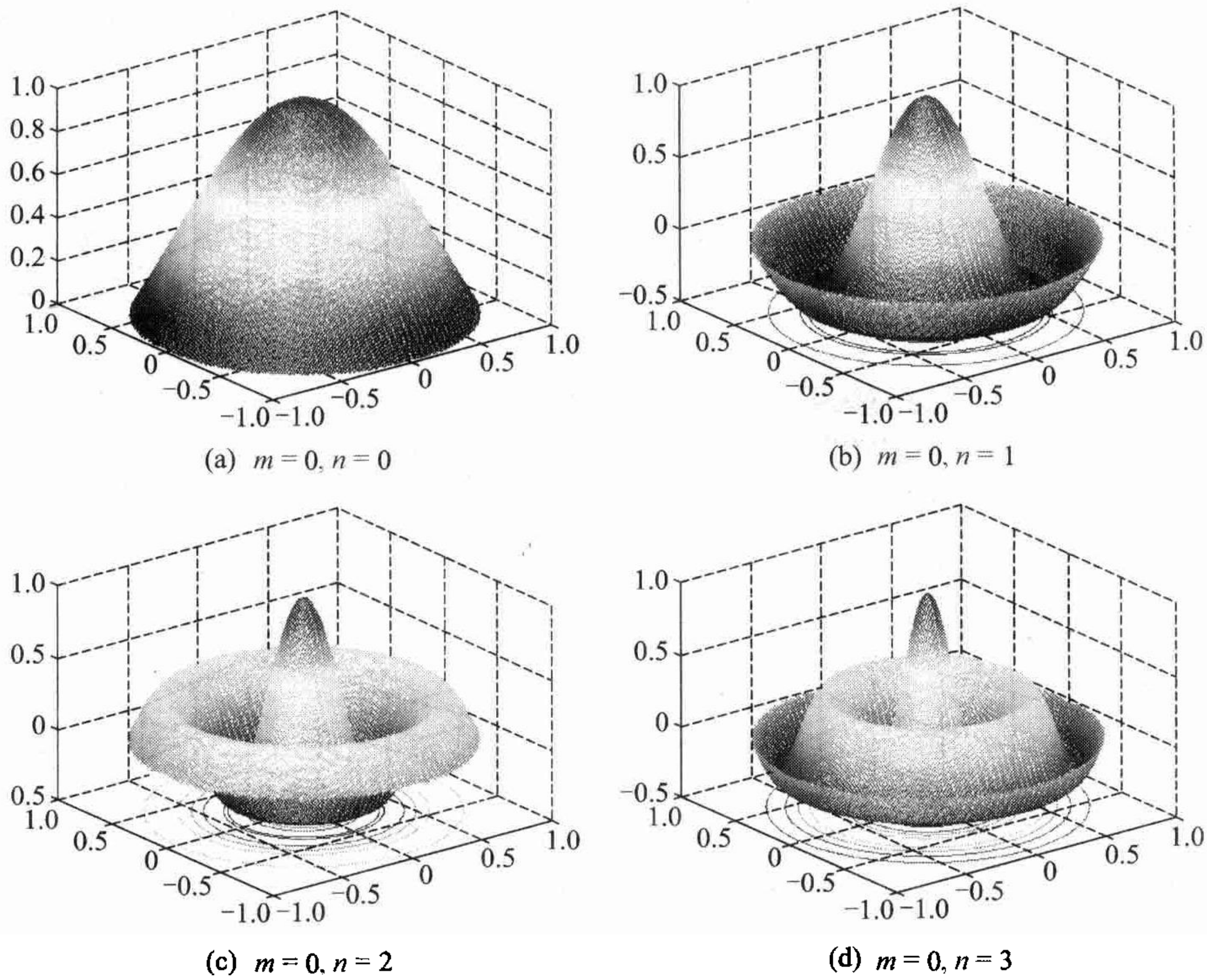


图 7.36 圆形区域的本征函数图形

下面再画出圆形区域 $m=1$ 时的前四个本征模式，作为例子我们取 φ 的本征函数 $\sin\varphi$ 来画图。

先画出第一类齐次边界条件的贝塞尔函数 (m=1，n= 0，1，2，3) 的本征函数图形，如图 7.37。

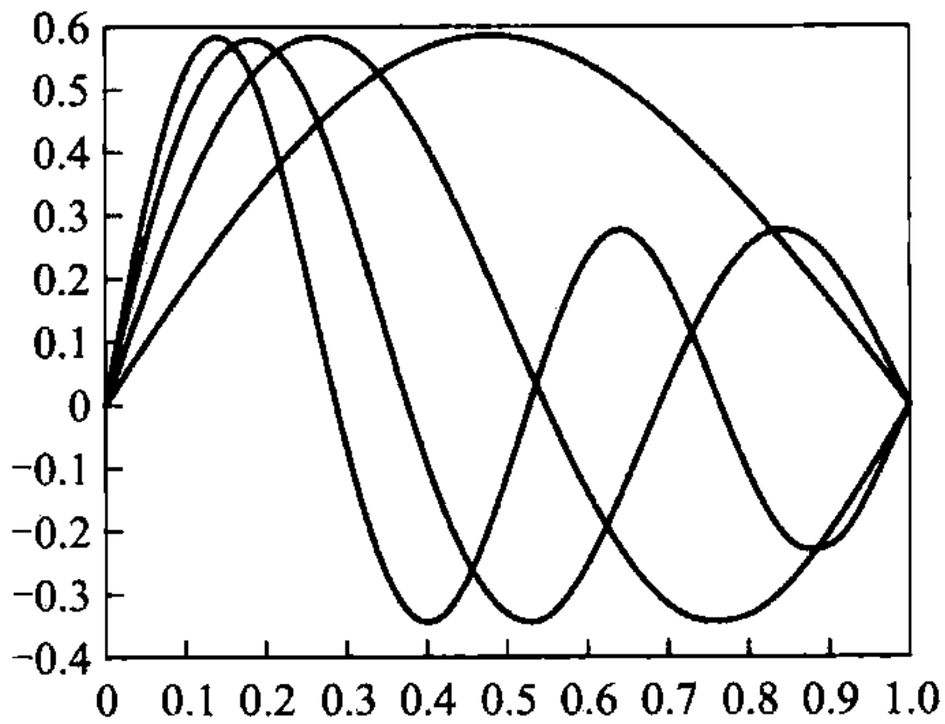
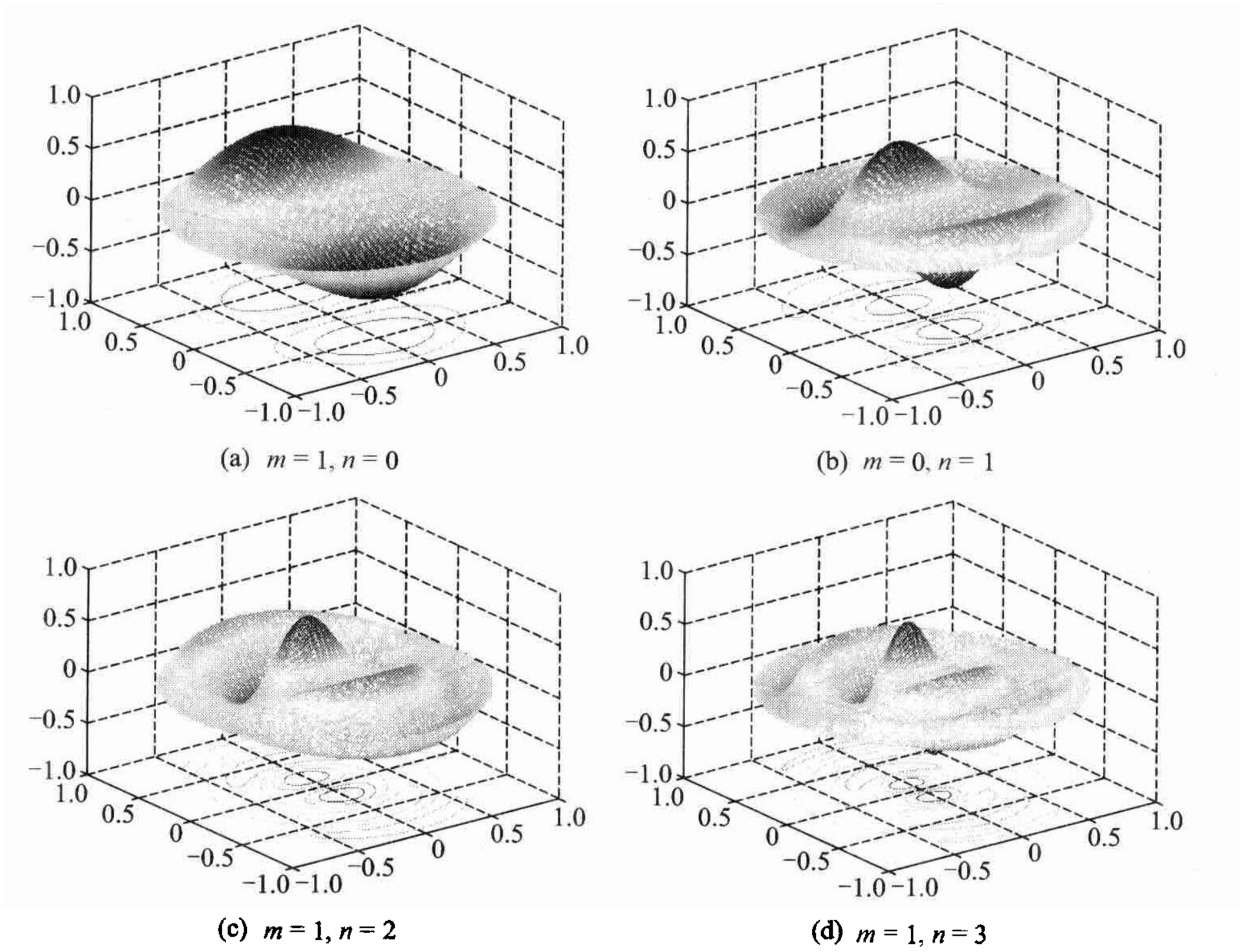


图 7.37 第一类齐次边界条件的贝塞尔函数 (m=1，n=0，1，2，3) 的本征函数图形

再画出 $m = 1$，$n = 0, 1, 2, 3$ 的圆形区域的前四个最小的本征模，如图 7.38。



(a) $m = 1, n = 0$ (b) $m = 0, n = 1$

(c) $m = 1, n = 2$ (d) $m = 1, n = 3$

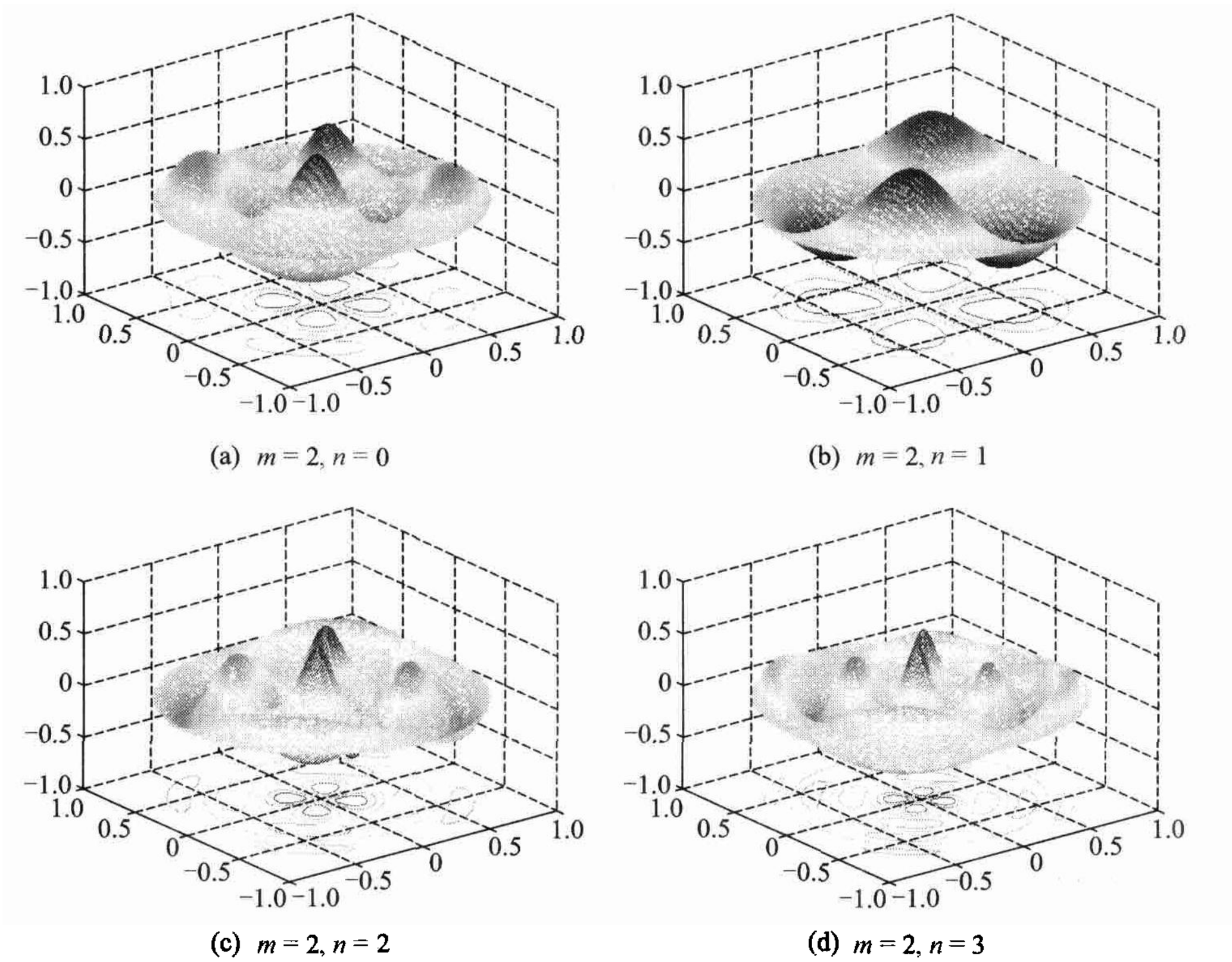
图 7.38 圆形区域内的本征函数图形

所用程序如下：

```
r=0:0.01:1;
[XX,YY]=meshgrid(-1:0.01:1);
[Q, R]=cart2pol(XX,YY);
R(find(R>1))=NaN;
y=inline('besselj(1,x)','x');
x(1)=fzero(y,3);
figure
meshc(XX,YY,y(x(1)*R).*sin(Q))
for k=1:3
    x(k+1)=fzero(y,x(k)+2)
    figure
```

```
    meshc(XX,YY,y(x(k+1)*R).*sin(Q))
end
```

可以看到，$m=0$ 的本征模是对称的，而 $m=1$ 的本征模是不对称的。不难想象，如果用 $\cos\varphi$ 作图，图形只是在 φ 的正方向旋转 $\pi/2$，它们的本征值相同，这种现象叫简并。为了与偏微分方程工具箱的数值解对比，这里再画出 $m=2$，$n=0,1,2,3$ 的图像，即取 $\sin 2\varphi$，程序与上面的相同，只需将程序中的 $\sin Q$ 改为 $\sin 2Q$ 即可，所得图像如图 7.39。



(a) $m=2, n=0$ (b) $m=2, n=1$

(c) $m=2, n=2$ (d) $m=2, n=3$

图 7.39 圆形区域的本征函数图形

现在知道，由于 pdetool 中的数值解是按本征值的大小排列本征函数的顺序，所以所画的图形有与 $m=0,1,2$ 对应的图形，读者很容易从几张图中找出与这些图形相对应的图形。

§7.5 特殊函数

在偏微分方程求解中会产生一些特殊函数，此外在其他一些计算过程中也会产生一些特殊函数。这些函数都可以在 MATLAB 中直接调用，方法是在指令

窗中键入

```
>> help matlab\specfun
```

在屏幕上就会显示已有的特种函数名称如下所示：

```
Specialized math functions.
   airy       - Airy functions.
   besselj    - Bessel function of the first kind.
   bessely    - Bessel function of the second kind.
   besselh    - Bessel functions of the third kind
                (Hankel function).
   besseli    - Modified Bessel function of the first kind.
   besselk    - Modified Bessel function of the second kind.
   beta       - Beta function.
   betainc    - Incomplete beta function.
   betaln     - Logarithm of beta function.
   ellipj     - Jacobi elliptic functions.
   ellipke    - Complete elliptic integral.
   erf        - Error function.
   erfc       - Complementary error function.
   erfcx      - Scaled complementary error function.
   erfinv     - Inverse error function.
   expint     - Exponential integral function.
   gamma      - Gamma function.
   gammainc   - Incomplete gamma function.
   gammaln    - Logarithm of gamma function.
   legendre   - Associated Legendre function.
   ......
```

这些函数的中文名称如下：

airy	艾里函数
besselj	第一类贝塞尔函数
bessely	第二类贝塞尔函数(诺伊曼函数)
besselh	第三类贝塞尔函数(汉克尔函数)
besseli	第一类虚宗量贝塞尔函数
besselk	第二类虚宗量贝塞尔函数
beta	B函数
betainc	不完全的B函数

betaln	B函数的对数
ellipj	雅可比椭圆函数
ellipke	完全的椭圆积分
erf	误差函数
erfc	余误差函数
erfcx	标度的余误差函数
erfinv	逆误差函数
expint	指数积分函数
gamma	Γ函数
gammainc	不完全的Γ函数
gammaln	Γ函数的对数
legendre	连带勒让德函数

利用 MATLAB 的帮助系统，就可以知道这些特殊函数的用法，从而计算各种特殊函数的值。借助于 MATLAB 的符号运算工具箱，还可以调用另一个优秀的数学软件 MAPLE 中更多的特殊函数。下面是一些例子。

§7.5.1 勒让德函数

1. 计算勒让德函数的指令 legendre(N,X)

指令计算所有 N 阶连带勒让德函数在 x 处的函数值。

例如，在指令窗中输入

```
>>legendre(2,0:0.1:0.2)
```

屏幕显示的矩阵如下

```
-0.5000    -0.4850    -0.4400
      0    -0.2985    -0.5879
 3.0000     2.9700     2.8800
```

它表示的结果是

	$x=0$	$x=0.1$	$x=0.2$
$m=0$	$P_2^0(0)=-0.5000$	$P_2^0(0.1)=-0.4850$	$P_2^0(0.2)=-0.4400$
$m=1$	$P_2^1(0)=0$	$P_2^1(0.1)=-0.2985$	$P_2^1(0.2)=-0.5879$
$m=2$	$P_2^2(0)=3.0000$	$P_2^2(0.1)=2.9700$	$P_2^2(0.2)=2.8800$

2. 勒让德多项式的图像

下面画出前 6 个勒让德多项式的图像 (图 7.40)。

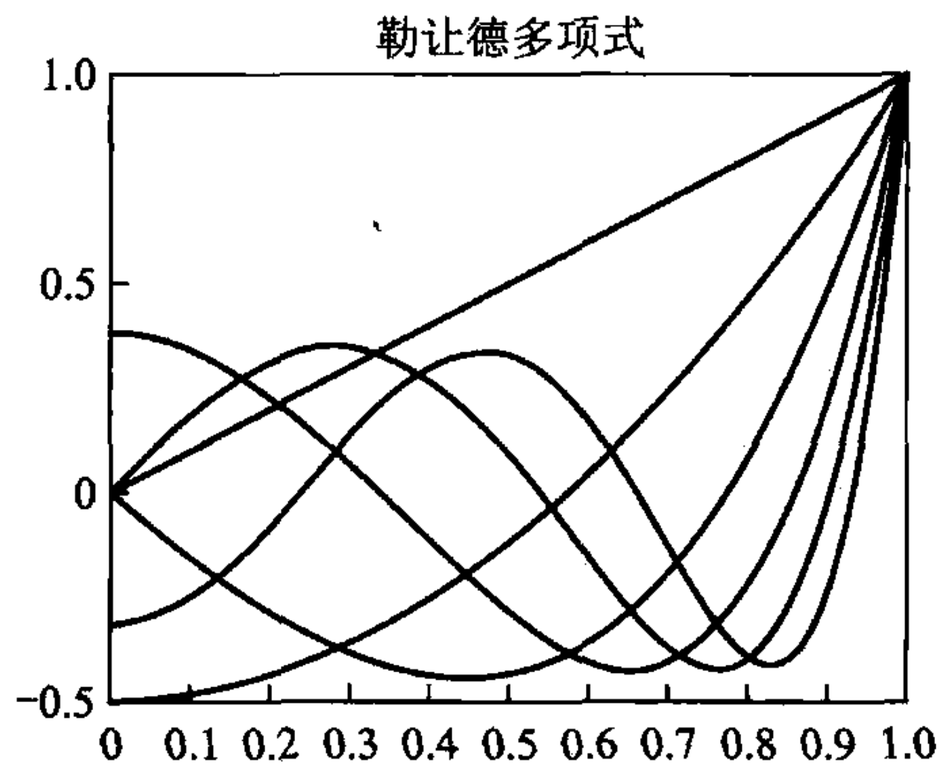


图 7.40

```
>>x=0:0.01:1;
>>y1=legendre(1,x);
>>y2=legendre(2,x);
>>y3=legendre(3,x);
>>y4=legendre(4,x);
>>y5=legendre(5,x);
>>y6=legendre(6,x);
>>plot(x,y1(1,:),x,y2(1,:),x,y3(1,:),...
x,y4(1,:),x,y5(1,:),x,y6(1,:))
```

3. 勒让德函数的图像

再画连带勒让德函数 $P_3^{0,1,2,3}$ 的图形 (图 7.41)。

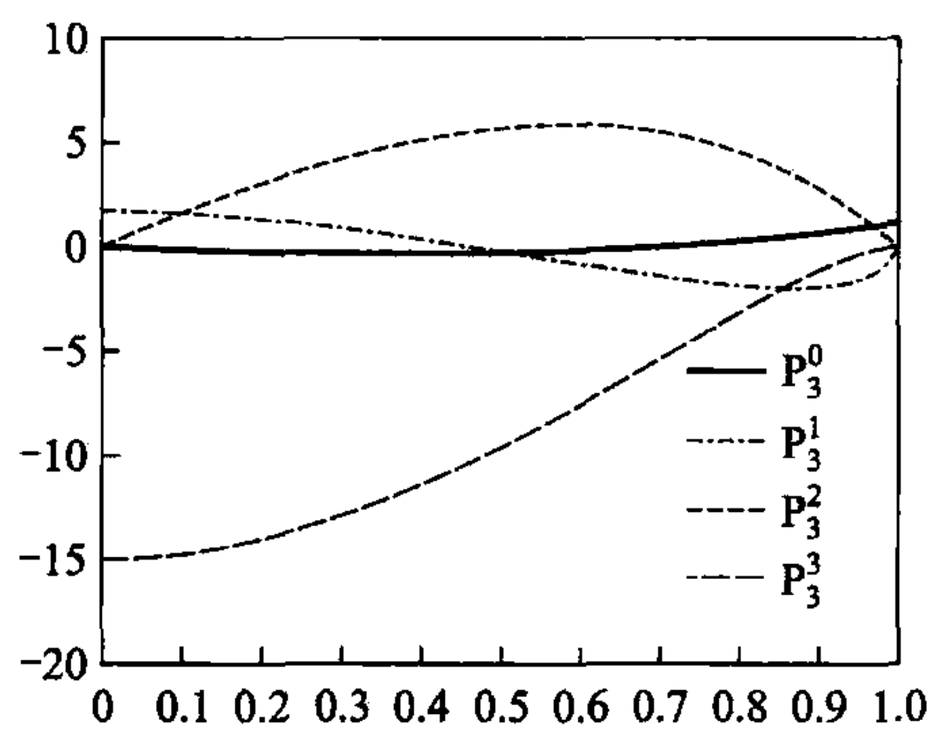


图 7.41

(1) 以 x 为变量

```
>>x=0:0.01:1;
>>y=legendre(3,x);
```

```
>>plot(x, y(1,:),'-',x, y(2,:),'-.',...
x, y(3,:),':',x, y(4,:),'--')
>>legend('P_3^0','P_3^1',...
'P_3^2','P_3^3');
```

(2) 以 θ 为变量

下面画出 $P_1^{(0,1)}$, $P_2^{(0,1,2)}$, $P_3^{(0,1,2,3)}$ 在极坐标下的图形 (图 7.42)。

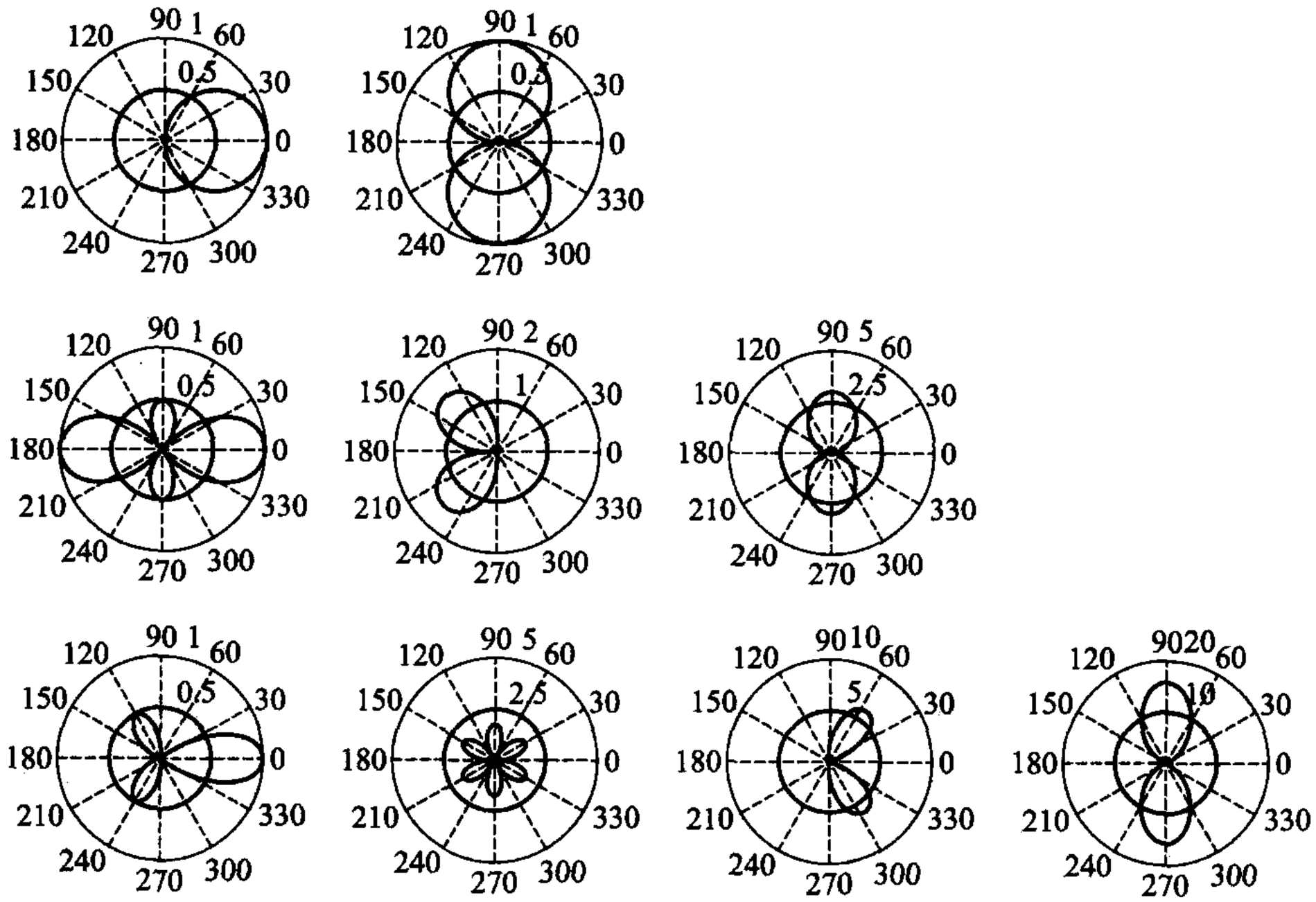


图 7.42

第一行是 P_1 的两个图形，第二行是 P_2 的三个图形，第三行是 P_3 的四个图形。

```
rho=legendre(1,cos(0:0.1:2*pi));
t=0:0.1:2*pi;
rho1=legendre(2,cos(0:0.1:2*pi));
rho2=legendre(3,cos(0:0.1:2*pi));

subplot(3,4,1) polar(t,rho(1,:))
subplot(3,4,2) polar(t,rho(2,:))
subplot(3,4,5) polar(t,rho1(2,:))
subplot(3,4,6) polar(t,rho1(2,:))
subplot(3,4,7) polar(t,rho1(2,:))
subplot(3,4,9) polar(t,rho2(2,:))
```

```
subplot(3,4,10) polar(t,rho2(2,:))
subplot(3,4,11) polar(t,rho2(2,:))
subplot(3,4,12) polar(t,rho2(2,:))
```

§7.5.2 贝塞尔函数

MATLAB 有 5 种计算贝塞尔函数的指令：

计算指令	所计算的函数
besselj	计算第一类贝塞尔函数，简称贝塞尔(Bessel)函数
bessely	计算第二类贝塞尔函数，即诺伊曼(Neumann)函数
besselh	计算第三类贝塞尔函数，即汉克尔(Hankel)函数
besseli	计算第一类虚宗量贝塞尔函数(虚宗量贝塞尔函数)
besselk	计算第二类虚宗量贝塞尔函数(虚宗量汉克尔函数)

1. 贝塞尔函数

```
>> y=besselj(0:3,(0:.2:10)');
>> figure(1)
>> plot((0:.2:10)',y)
>> legend('J 0', 'J 1',  'J 2' ,  'J 3')
```

可得到 $J_{0,1,2,3}$ 的曲线如图 7.43。

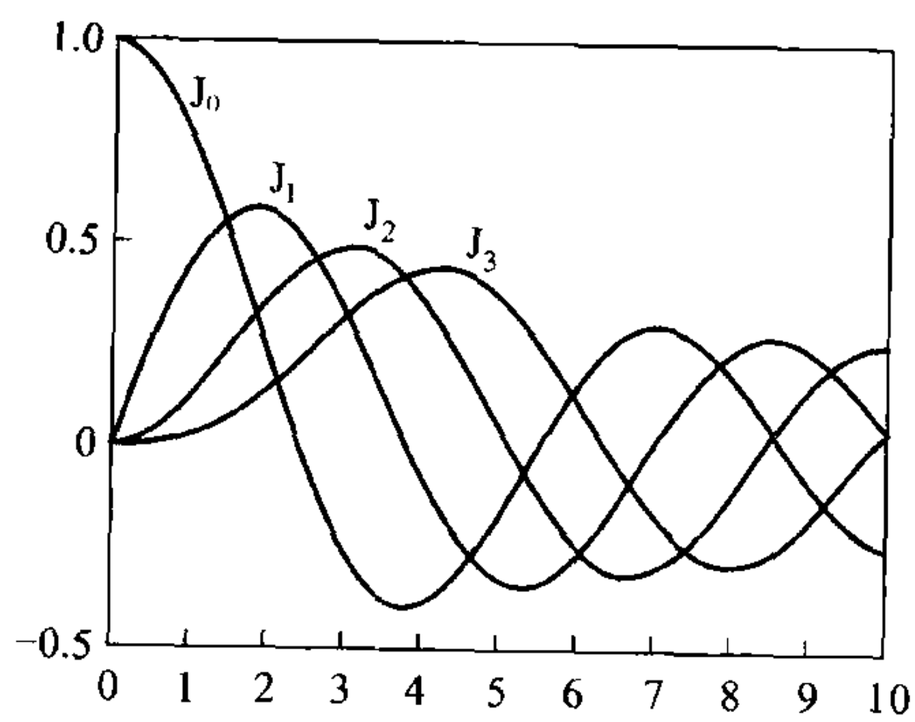


图 7.43

计算插值的指令有 interp1，interp1q 和 interp2 等。用内插法可以找到贝塞尔函数的零点，也可以用指令 fzero 求解。下面的程序是找 $0 \leqslant x \leqslant 50$ 之间的零点。

```
x=0:0.05:50; y=besselj(0,x); LD=[];
for k=1:1000,
   if y(k)*y(k+1)<0
      h=interp1(y(k:k+1), x(k:k+1),0);
```

```
        LD=[LD,h]
    end
end
```

2. 诺伊曼函数

下面的程序画出诺伊曼函数 $N_{0,1}$ 的曲线 (图 7.44)。

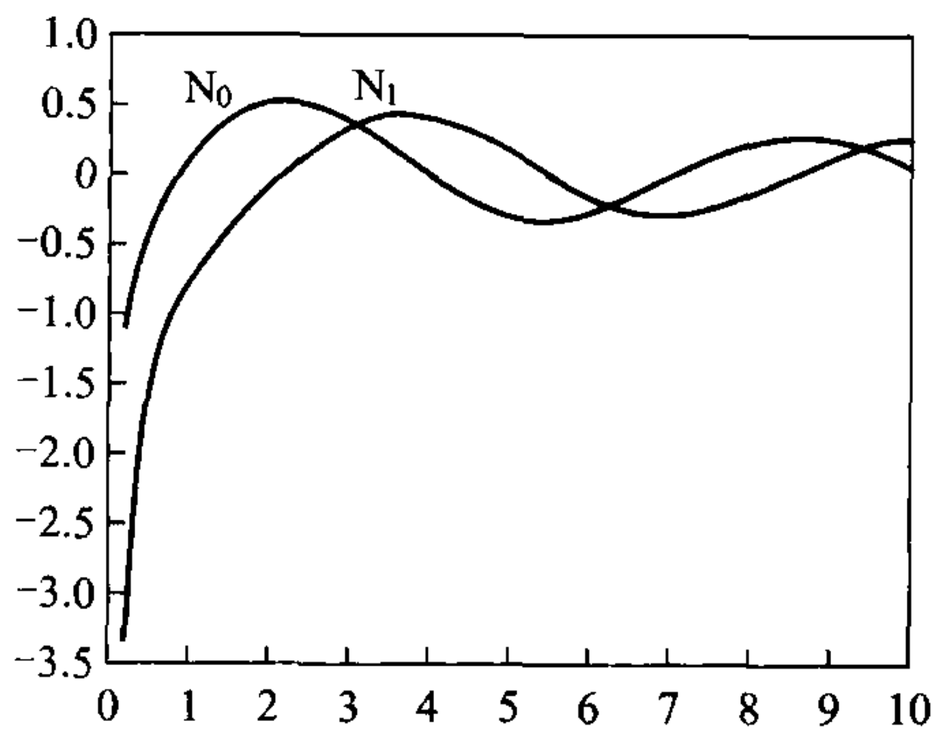


图 7.44

```
>> y=bessely(0:1,(0:.2:10)');
>> plot((0:.2:10)',y)
>> grid on
```

3. 虚宗量贝塞尔函数

下面画虚宗量贝塞尔函数 $I_{0,1}$ 的曲线 (图 7.45)。

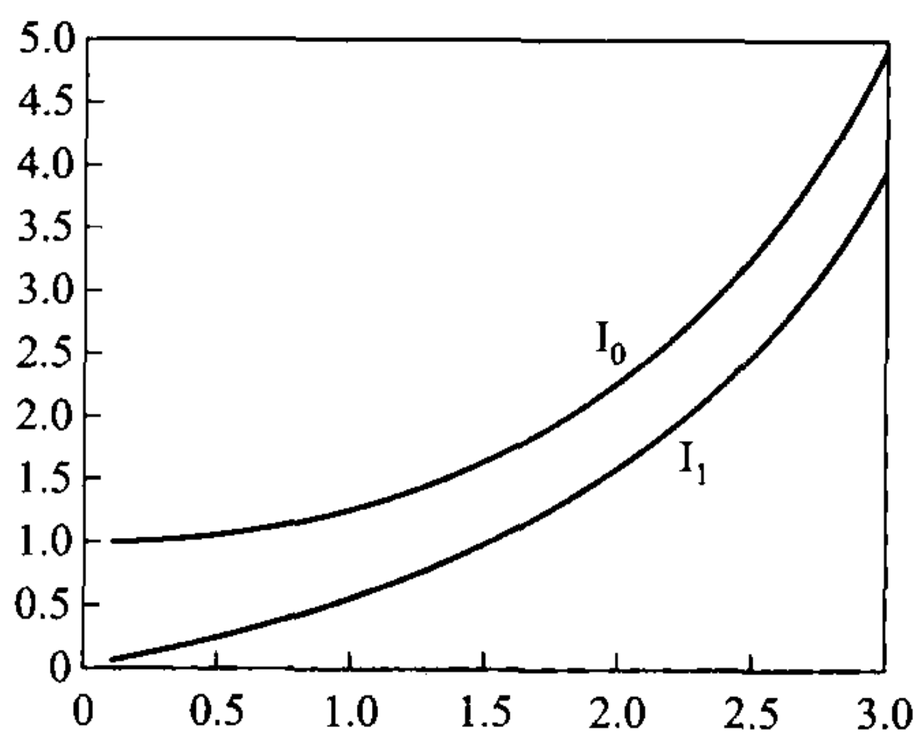


图 7.45

```
>>I=besseli(0:1,(0.1:0.1:3)');
>>plot((0.1:0.1:3)',I)
```

4. 虚宗量汉克尔函数

下面画虚宗量汉克尔函数 $K_{0,1}$ 的曲线 (图 7.46)。

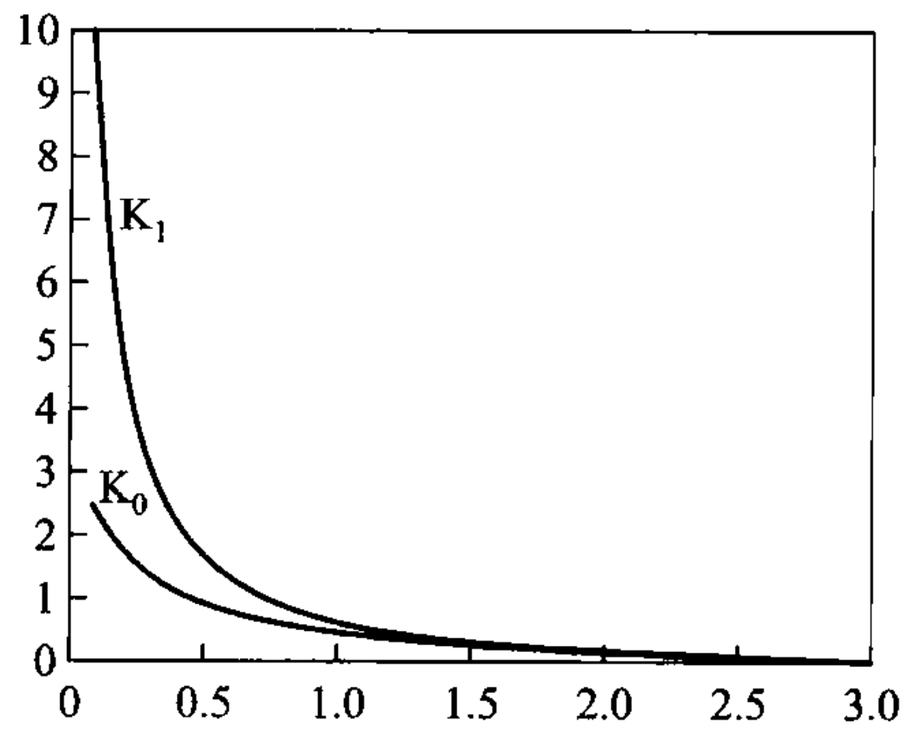


图 7.46

```
>>K=besselk(0:1,(0.1:0.1:3)');
>>plot((0.1:0.1:3)',k)
```

5. 球贝塞尔函数

下面画球贝塞尔函数 $j_{0,1,2,3}$ 的图形 (图 7.47)。由于在 $x=0$ 处函数值是 0/0，所以作图不是从原点出发。

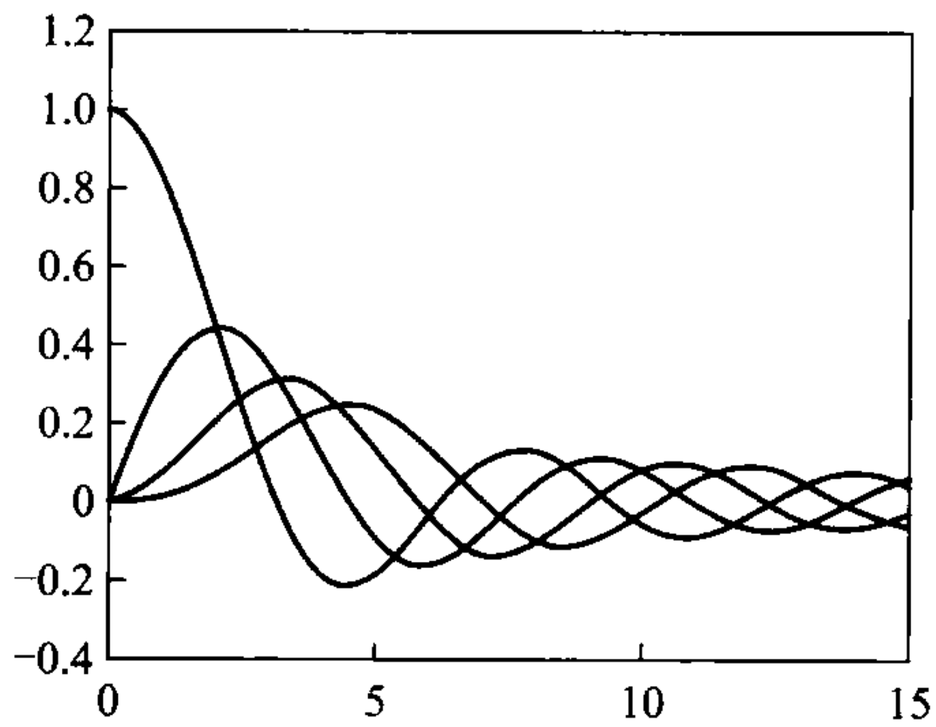


图 7.47

```
x=eps:0.2:15;
y1=sqrt(pi/2./x).*besselj(1/2,x);
y2=sqrt(pi/2./x).*besselj(3/2,x);
y3=sqrt(pi/2./x).*besselj(5/2,x);
y4=sqrt(pi/2./x).*besselj(7/2,x);
plot(x,y1,x,y2,x,y3,x,y4)
```

6. 球诺伊曼函数

下面画球诺伊曼函数 $n_{0,1,2,3}$ 的图形 (图 7.48)。

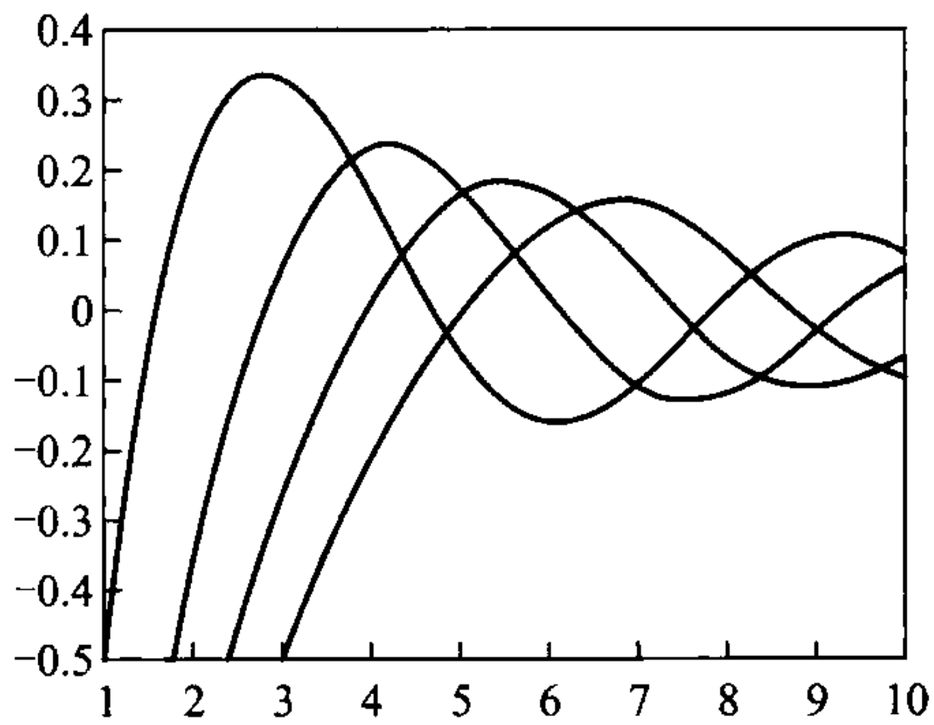


图 7.48

```
x=0.8:0.2:15;
y1=sqrt(pi/2./x).*bessely(1/2,x);
y2=sqrt(pi/2./x).*bessely(3/2,x);
y3=sqrt(pi/2./x).*bessely(5/2,x);
y4=sqrt(pi/2./x).*bessely(7/2,x);
plot(x,y1,x,y2,x,y3,x,y4)
axis([1 10  -0.5  0.4])
grid on
```

§7.5.3 调用 MAPLE 计算特殊函数

MAPLE 是一个符号计算功能很强的数学软件，在MATLAB 中可以调用 MAPLE 的某些功能，调用的指令一共有以下五个：

mfun	对MAPLE中的特殊函数作数值计算的指令
mfunlist	列出在MAPLE中能用mfun计算的特殊函数
mhelp	查阅MAPLE中的库函数及其调用方法
maple	调用MAPLE的计算指令
procread	读入MAPLE程序

这里用这些指令来计算特殊函数.

1. mfunlist 的用法

例如，为了查看在 MATLAB 中可以用 MAPLE 计算的特殊函数，在指令窗中键入

```
>>mfunlist
```

可以在屏幕上看到全部可调用的MAPLE的特殊函数如下：

```
MFUNLIST Special functions for MFUN.
    The following special functions are listed in alphabetical
```

order according to the third column. n denotes an integer argument, x denotes a real argument, and z denotes a complex argument. For more detailed descriptions of the functions, including any argument restrictions, see the Reference Manual, or use MHELP.

bernoulli	n	Bernoulli Numbers
bernoulli	n,z	Bernoulli Polynomials
BesselI	x1,x	Bessel Function of the First Kind
BesselJ	x1,x	Bessel Function of the First Kind
BesselK	x1,x	Bessel Function of the Second Kind
BesselY	x1,x	Bessel Function of the Second Kind
Beta	z1,z2	Beta Function
binomial	x1,x2	Binomial Coefficients
LegendreKc	x	Complete Elliptic Integral of First Kind
LegendreEc	x	Complete Elliptic Integral of Second Kind
LegendrePic	x1,x	Complete Elliptic Integral of Third Kind
LegendreKc1	x	LegendreKc using Complementary Modulus
LegendreEc1	x	LegendreEc using Complementary Modulus
LegendrePic1	x1,x	LegendrePic using Complementary Modulus
erfc	z	Complementary Error Function
erfc	n,z	Complementary Error Function's Iterated Integrals
Ci	z	Cosine Integral
dawson	x	Dawson's Integral
Psi	z	Digamma Function
dilog	x	Dilogarithm Integral
erf	z	Error Function
euler	n	Euler Numbers
euler	n,z	Euler Polynomials
Ei	x	Exponential Integral
Ei	n,z	Exponential Integral
FresnelC	x	Fresnel Cosine Integral
FresnelS	x	Fresnel Sine Integral
GAMMA	z	Gamma Function

```
harmonic    n        Harmonic Function
Chi         z        Hyperbolic Cosine Integral
Shi         z        Hyperbolic Sine Integral
hypergeom   X1,X2    (Generalized) Hypergeometric Function
LegendreF   x,x1     Incomplete Elliptic Integral of First Kind
LegendreE   x,x1     Incomplete Elliptic Integral of Second Kind
LegendrePi  x,x2,x1  Incomplete Elliptic Integral of Third Kind
GAMMA       z1,z2    Incomplete Gamma Function
W           z        Lambert's W Function
W           n,z      Lambert's W Function
lnGAMMA     z        Logarithm of the Gamma function
Li          x        Logarithmic Integral
Psi         n,z      Polygamma Function
Ssi         z        Shifted Sine Integral
Si          z        Sine Integral
Zeta        z        (Riemann) Zeta Function
Zeta        n,z      (Riemann) Zeta Function
Zeta        n,z,x    (Riemann) Zeta Function


        Orthogonal Polynomials (Extended Symbolic Math
        Toolbox only)
T       n,x          Chebyshev of the First Kind
U       n,x          Chebyshev of the Second Kind
G       n,x1,x       Gegenbauer
H       n,x          Hermite
P       n,x1,x2,x    Jacobi
L       n,x          Laguerre
L       n,x1,x       Generalized Laguerre
P       n,x          Legendre


    See also MFUN, MHELP.
```

这些函数的中文名称如下：

bernoulli(n)	伯努利数
bernoulli(n,z)	伯努利多项式
BesselI(x1,x)	虚宗量贝塞尔函数

BesselJ(x1,x)	第一类贝塞尔函数
BesselK(x1,x)	虚宗量汉克尔函数
BesselY(x1,x)	第二类贝塞尔函数(诺伊曼函数)
Beta(z1,z2)	B函数
binomial(x1,x2)	二项式系数
LegendreKc(x)	第一类完全椭圆积分
LegendreEc(x)	第二类完全椭圆积分
LegendrePic(x1,x)	第三类完全椭圆积分
LegendreKc1(x)	使用补充系数的第一类完全椭圆积分
LegendreEc1(x)	使用补充系数的第二类完全椭圆积分
LegendrePic1(x1,x)	使用补充系数的第三类完全椭圆积分
erfc(z)	余误差函数
erfc(n,z)	余误差函数的内插积分
Ci(z)	余弦积分
dawson(x)	Dawson 积分
Psi(z)	Ψ函数
dilog(x)	Dilogarithm 积分
erf(z)	误差函数
euler(n)	欧拉数
euler(n.z)	欧拉多项式
Ei(x)	指数积分
Ei(n,z)	指数积分
FresnelC(x)	菲涅尔余弦积分
FresnelS(x)	菲涅尔正弦积分
GAMMA(z)	Γ函数
harmonic(n)	谐函数
Chi(z)	双曲余弦积分
Shi(z)	双曲正弦积分
hypergeom(X1,X2)	(广义的)超几何函数
LegendreF(x,x1)	第一类不完全椭圆积分
LegendreE(x,x1)	第二类不完全椭圆积分
LegendrePi(x,x2,x1)	第三类不完全椭圆积分
GAMMA(z1,z2)	不完全Γ函数
W(z)	兰伯特 W 函数
W(n,z)	兰伯特 W 函数

lnGAMMA(z)	Γ函数的对数
Li(x)	对数积分
Psi(n,z)	Ψ函数
Ssi(z)	转换正弦积分
Si(z)	正弦积分
Zeta(z)	黎曼ζ函数
Zeta(n,z)	黎曼ζ函数
Zeta(n,z,x)	黎曼ζ函数
(以下的正交多项式只在扩展的符号工具箱中使用)	
T(n, x)	生成n阶的第一类切比雪夫多项式
U(n, x)	生成n阶第二类切比雪夫多项式
G(n, x1, x)	生成n阶盖根鲍尔多项式
H(n, x)	生成n阶厄米项式
P(n, x1, x2, x)	生成n阶雅可比多项式
L(n, x)	生成n阶拉盖尔多项式
L(n, x1, x)	生成n阶广义拉盖尔多项式
P(n, x)	生成n阶勒让德多项式

从中可以看到，有一部分特殊函数是可以用指令 mfun 来调用，而另一些正交多项式 T，U，G，H，P，L 等则只能在扩展的符号工具箱中使用。如果想知道某个具体的函数的用法，可以用 mhelp 查询。例如下面的操作是查阅误差函数的用法：

```
>> mhelp erf
erf - The Error Function
erfc - The Complementary Error Function and its Iterated Integrals
erfi - The Imaginary Error Function
Calling Sequence:
     erf(x)
     erfc(x)
     erfc(n, x)
     erfi(x)
     ......
```

屏幕上显示了误差函数的三种类型及其四种调用的方法，还举了一些例子。

2. mfun 的用法

用 mfun 对特殊函数作数值计算的格式是：

MFUN('fun',MFUN('fun',p1,p2,...,pk))

这里，fun 是特殊函数，p1，p2，$\cdots$，pk 是函数中的参数。例如要计算误差函数的值所用的操作是

```
>>x = 0:0.1:5.0;
>>y=mfun('erf',x)
```

3. mhelp 的用法

指令 mhelp 是用来查找帮助文件。例如要查找调用正交多项式厄米函数的方法，操作如下：

```
>>mhelp H
orthopoly[H] - Hermite polynomial
Calling Sequence:
     H(n, x)
     orthopoly[H](n, x)

Parameters:
     n - a non-negative integer
     x - any algebraic expression

Description:
- The function H will compute the nth Hermite polynomial,
  evaluated at x.
- These polynomials are orthogonal on the interval
  (-infinity, infinity), with respect to the weight
  function w(x) = exp(-x2̂).
  Thus
    ∫_∞^∞ w(x)H(n,x)H(m,x)dx = 0
  if m is not equal to n.
- They satisfy the following recurrence relation:
        H(0,x) = 1,
        H(1,x) = 2*x,
        H(n,x) = 2*x*H(n-1,x) - 2*(n-1)*H(n-2,x),
        for n>1.
- The command with(orthopoly,H) allows the use of the
  abbreviated form of this command.
```

```
Examples:
>> with(orthopoly);
    [G, H, L, P, T, U]
>> H(2,x);
```

$4x^2 - 2$

```
>> H(3,x);
```

$8x - 12x^3$

```
See Also:
linalg[hermite]
```

从帮助信息中知道，调用 H 的格式有两种，H(n,x)，和 orthopoly[H](n, x)，而第一种简略格式要在使用了指令 with(orthopoly,H) 才能使用。此外，帮助信息还给出了正交关系，递推关系以及几个实例。但是，如果直接将例子中的指令键入指令窗，却发现它们并不能执行。那么问题在哪里呢? 问题就在于，这些指令都是软件 MAPLE 的指令，要在 MAPLE 的环境下使用。所以正确的用法是使用指令 maple。

4. maple 的用法

```
>>yy = maple('orthopoly[H](2, x)')
yy = 4*x^2-2
```

或者是

```
>>maple('with(orthopoly)')
ans = [G, H, L, P, T, U]
>>yy = maple('H(2,x)')
yy = 4*x^2-2
```

这样得到的 yy 就是二阶的厄米多项式。如果要画出它的图形，指令是

```
>>ezplot(yy)
```

所得的图形为图 7.49。

如果要计算 $x = 1:0.2:2$ 时 yy 的值，要使用给符号变量赋值的指令 sub:

```
>>subs(yy,1:0.2:2)
ans =
2.0000    3.7600    5.8400    8.2400    10.9600   14.0000
```

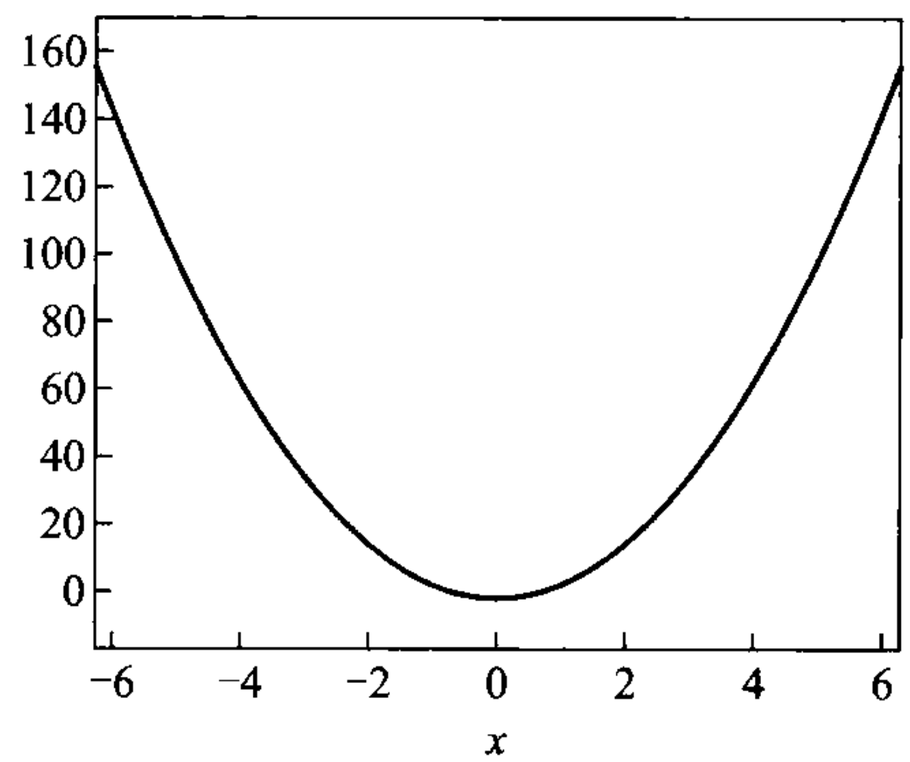


图 7.49 二阶的厄米多项式的图形

§7.5.4 平面波展开

本节介绍特殊函数在广义傅里叶展开方面应用的两个例子。

1. 平面波展开为球面波的叠加

平面波展开为球面波的叠加是广义傅里叶级数的一个很好的例子，这个公式在研究量子力学散射问题时很有用处。展开公式是[①]

$$\mathrm{e}^{-\mathrm{i}kr\cos\theta}=\sum_{l=0}^{\infty}(2l+1)\mathrm{i}^l\mathrm{j}_l(kr)\mathrm{P}_l(\cos\theta)$$

等式左边的各个分量代表的是 $m=0$，l 取不同值时不同的球面波。这时球面波与 φ 无关。以水平面作为 Oxz 面，画出 $l=0,1,2,\cdots,5$ 的分量的图形如图 7.50。

所用的程序如下：

```
[X,Z]=meshgrid(0.05:0.1:10);
figure
[Q,R]=cart2pol(X,Z);
sqrtR=sqrt(pi/2./R);
Leg0=legendre(0,cos(Q));
Bes0=sqrtR.*besselj(0,R);
qiu=Bes0.*Leg0;
surfc(X,Z,qiu)
for k=1:5
    Leg=legendre(k,cos(Q));                                  %从1开始
```

① 梁昆淼. 数学物理方法. 3 版. 北京: 高等教育出版社, 1998: 371

```
    Legk=squeeze(Leg(1,:,:));
    Bes=sqrtR.*besselj(k,R);
    qiuk=Bes.*Legk;
    figure
    surfc(X,Z,qiuk)
    xlabel('x')
end
```

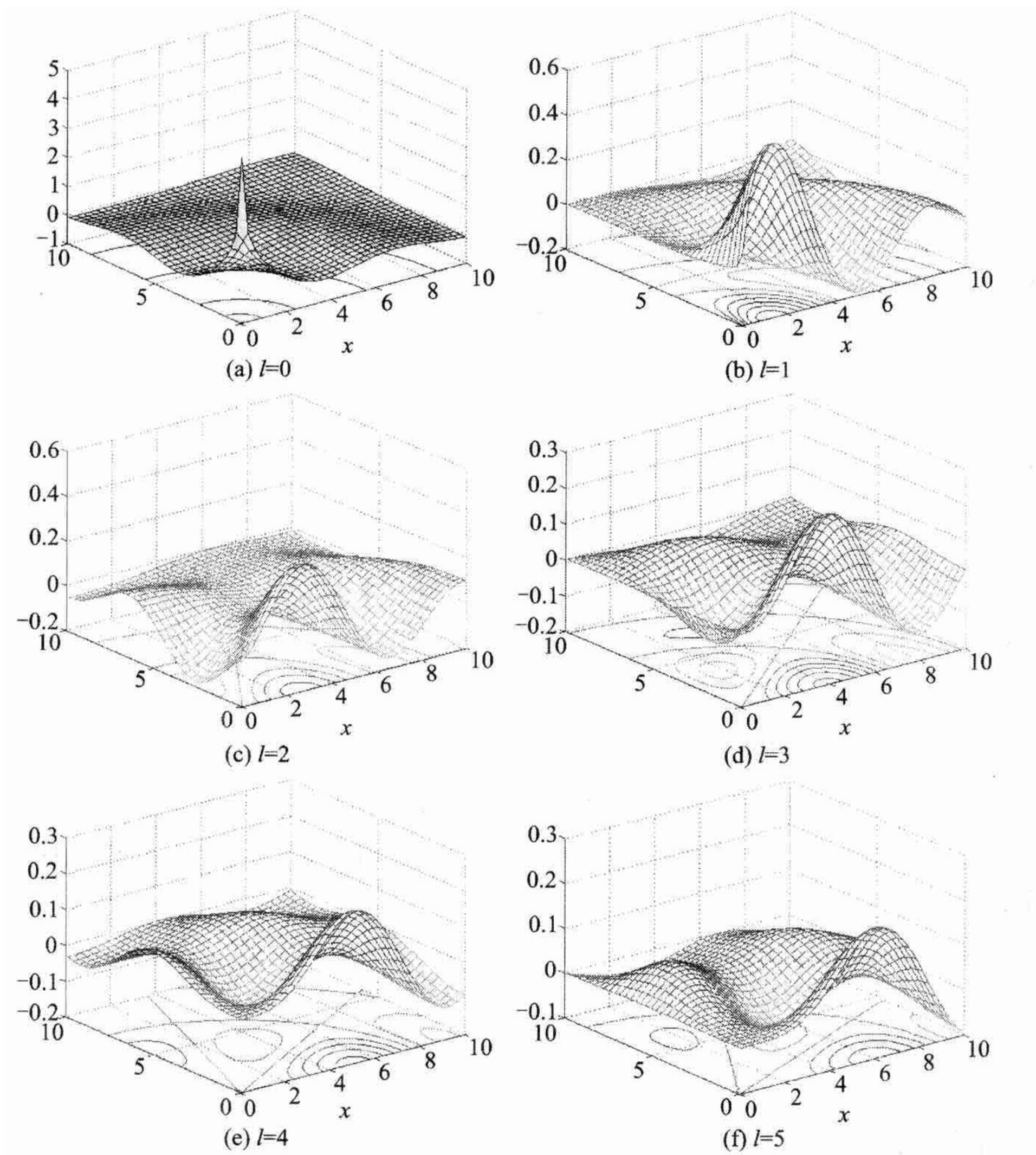


图 7.50 球面波分量图

取上式两边的实部，在等式的右边，当 $l=2n$ 时得实部：

$$\cos(kr\cos\theta)=\sum_{n=0}^{\infty}(4n+1)\mathrm{i}^{2n}\mathrm{j}_{2n}(kr)\mathrm{P}_{2n}(\cos\theta)$$

等式的左边是向 z 传播的平面波，其图形如图 7.51。

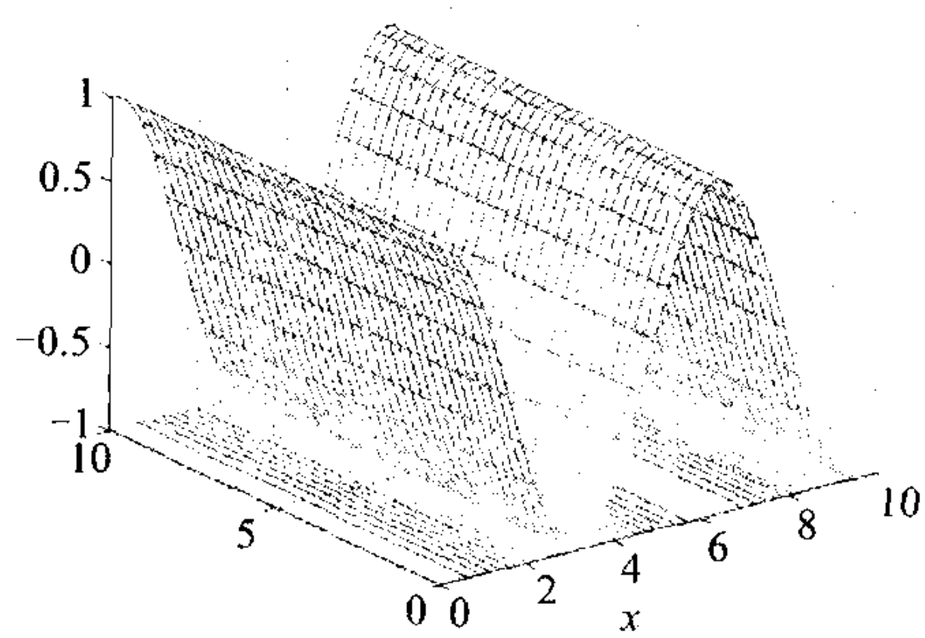


图 7.51 向 z 方向传播的平面波

等式右边分别取 $k=1$，n 最大分别取 10 和 50，得到叠加后的图形如图 7.52。

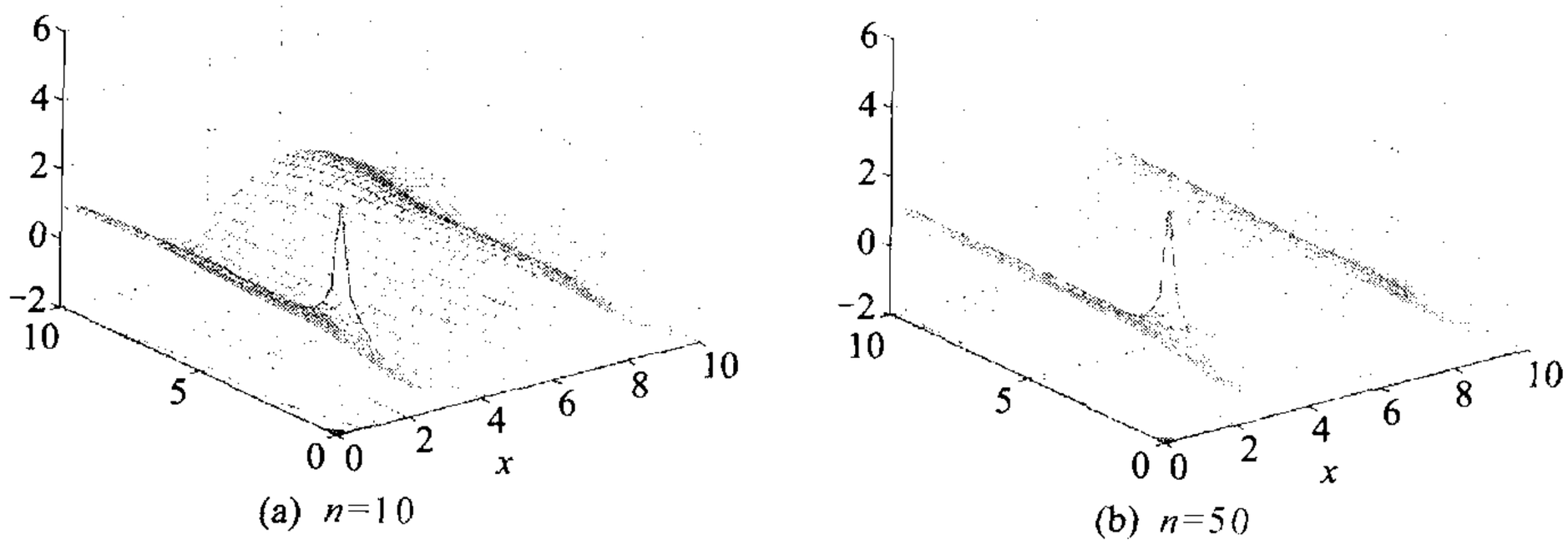


图 7.52 n 最大取分别 10，50 时得到球面波叠加的图形

可以看到，n 取到 50 的叠加效果比 n 取到 10 的效果要好得多。另外在原点附近，有奇异性，所以球面波叠加的效果与平面波有一明显的差异。画图的程序如下：

```
[X,Z]=meshgrid(0.05:0.1:10);
figure
g=cos(X);
contour(g)
meshc(X,Z,g)
xlabel('z')

[Q,R]=cart2pol(X,Z);
sqrtR=sqrt(pi/2./R);
Leg0=legendre(0,cos(Q));
```

```
Bes0=sqrtR.*besselj(0,R);
qiu=Bes0.*Leg0;
for k=2:2:50
    Leg=legendre(k,cos(Q));                                  %从1开始
    Legk=squeeze(Leg(1,:,:));
    Bes=sqrtR.*besselj(k,R);
    qiuk=(2*k+1)*i^k*Bes.*Legk;
    qiu=qiu+qiuk;
end
figure
meshc(X,Z,qiu)
xlabel('z')
```

2. 平面波展开为柱面波的叠加

平面波展开为柱面波的叠加是另一个很有趣的广义傅里叶级数的例子。展开公式是①

$$\mathrm{e}^{\mathrm{i}k\rho\cos\varphi}=\mathrm{J}_0(k\rho)+2\sum_{m=1}^{\infty}\mathrm{i}^m\mathrm{J}_m(k\rho)\cos m\varphi$$

等式左边的各个分量代表的是不同 m 的柱面波。这时柱面波与 z 无关，在水平面 Oxy 内画出 $m=0,1,2,\cdots,5$ 的分量的图形如图 7.53。

所用的程序如下：

```
[X,Y]=meshgrid(0.05:0.1:10);
[Q,R]=cart2pol(X,Y);
figure
zu=besselj(0,R);
meshc(X,Y,zu)
for k=1:5
    zu=besselj(k,R).*cos(k*Q);
    figure
    meshc(X,Y,zu)
    xlabel('x')
end
```

取上式两边的实部，令 $x=\rho\cos\varphi$，得

$$\cos kx=\mathrm{J}_0(k\rho)+2\sum_{n=1}^{\infty}\mathrm{i}^{2n}\mathrm{J}_{2n}(k\rho)\cos 2n\varphi$$

① 姚端正. 数学物理方法学习指导. 北京：科学出版社，2003：513

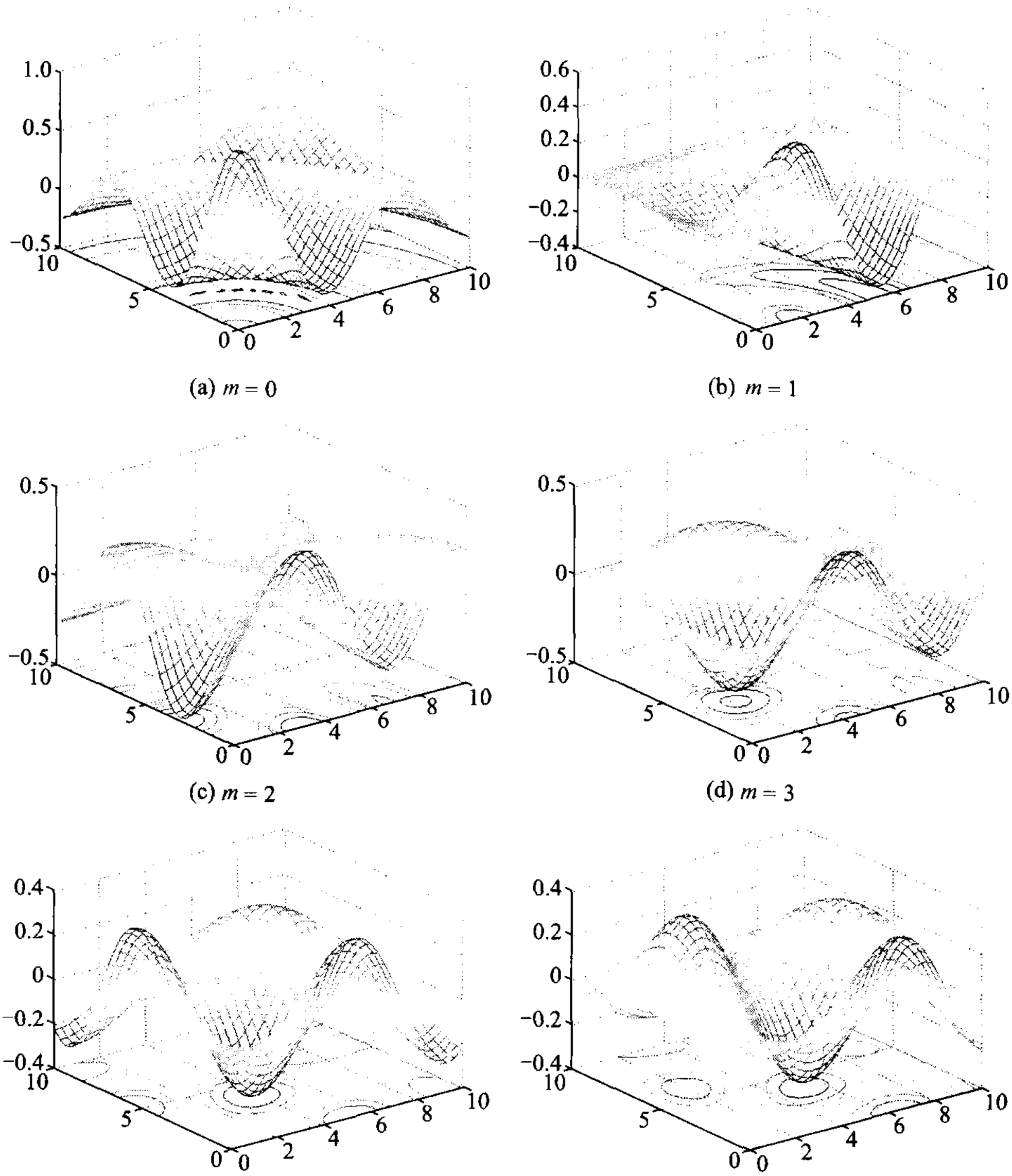


图 7.53 柱面波分量图

平面波的图形与前同，故不再画。等式右边分别取 $k=1$，n 最大分别取 10 和 20，得到叠加后的图形，如图 7.54。

可以看到，n 取到 20 的叠加效果已经很不错了。所以柱面波叠加的效果比球面波更好。画图的程序如下：

```
[X,Y]=meshgrid(0.05:0.1:10);                                    %k=1
[Q,R]=cart2pol(X,Y);
```

```
zu=besselj(0,R);
for k=2:2:20
    zuk=2*i^k*besselj(k,R).*cos(k*Q);
    zu=zu+zuk;
end
figure
meshc(X,Y,zu)
xlabel('x')
```

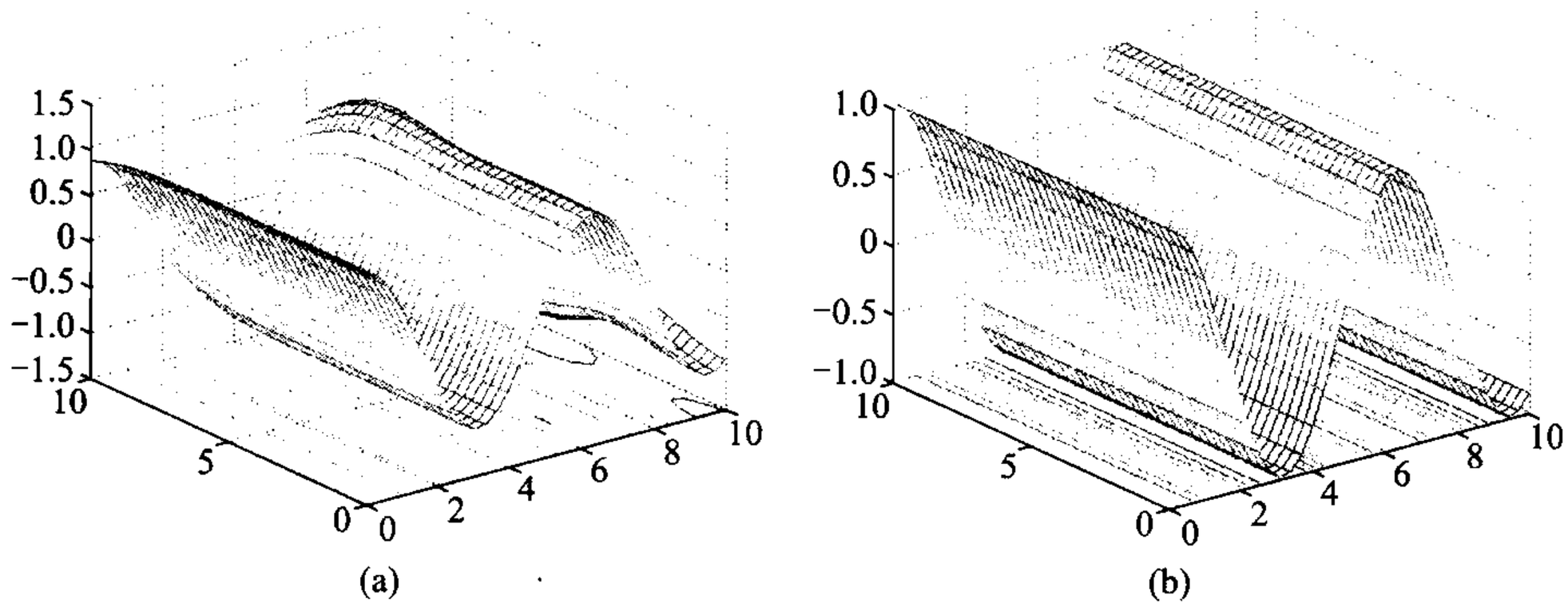


图 7.54 n 最大分别取 10, 20 时得到柱面波叠加的图形

§7.5.5 环形电流的磁场

圆线圈的半径为 a, 通有电流 I, 求空间任意一点的磁感应强度 B。

这是一个有趣的例子。我们曾经用数值积分的方法计算过它的磁场。这个问题是可以求出解析表达式的，但是都涉及特殊函数，这里提供两种解析表达式的图像，作为熟悉特殊函数的例子。两种不同解法所得表达式也不同，但是所得到的磁场图像与数值计算的结果是一致的。当然，要从画出的磁场线才能看出这一点。而这需要用到 MATLAB 的画场线的专用指令。如果仅仅画出磁场的分量，由于所用的坐标不同，所得到的图形并不能看出它们的一致性。

1. *用连带勒让德函数表示的磁感应强度的解析解*

首先，先解矢势的方程，再求磁感应强度 B，得到问题的解是[①]

$$B_r(r,\theta) = -\frac{\mu_0 I}{2r}\sum_{l=0}^{\infty}\left(\frac{r}{a}\right)^{2l+1}\mathrm{P}_{2l+1}^{1}(0)\mathrm{P}_{2l+1}(\cos\theta) \quad (r<a)$$

$$B_r(r,\theta) = -\frac{\mu_0 I}{2r}\sum_{l=0}^{\infty}\left(\frac{a}{r}\right)^{2l+2}\mathrm{P}_{2l+1}^{1}(0)\mathrm{P}_{2l+1}(\cos\theta) \quad (r>a)$$

① 吴崇试. 数学物理方法. 北京: 高等教育出版社, 2003

$$B_\theta(r,\theta) = -\frac{\mu_0 I}{2r}\sum_{l=0}^{\infty}\frac{1}{2l+1}\left(\frac{r}{a}\right)^{2l+1}\mathrm{P}_{2l+1}^1(0)\mathrm{P}_{2l+1}^1(\cos\theta) \quad (r<a)$$

$$B_\theta(r,\theta) = \frac{\mu_0 I}{2r}\sum_{l=0}^{\infty}\frac{1}{2l+2}\left(\frac{a}{r}\right)^{2l+2}\mathrm{P}_{2l+1}^1(0)\mathrm{P}_{2l+1}^1(\cos\theta) \quad (r>a)$$

下面的程序可以将解可视化，编程思路为，我们希望在 Oxz 面上表现解的结果，首先建立一个以直角坐标表示的数据网格。因为解是以球坐标表示的，所以要变换成对应的球坐标。考虑到这里只有两个元素 r,θ，所以没有使用球坐标变换的指令 cart2pol，而是使用极坐标变换的指令 cart2pol，但是根据对应关系改变了直角坐标变量的位置。然后将 r 分成环内 r_{in} 和环外 r_{out} 两部分。

在计算级数中的各项时，首先计算勒让德函数的值，然后分别计算每项中环内与环外的 B_r，B_θ 并用 for 循环累加起来。级数的项数根据所要精度决定，这里是选取了 22 项。最后还要将环内外的场合并成一个场。为了画出场线即磁感应强度线，还要将结果再变换成直角坐标下的表示。

上面的计算只是对 Oxz 面上第一象限作了计算，利用对称性，把其他的象限再画出来。实际程序如下：

```
a=0.35;
[X,Y]=meshgrid(0:0.1:1);
[theta,r]=cart2pol(Y,X);      %与极轴夹角,注意角的计算不是从X轴起
rout=r;  rin=r;
rin(find(rin>a))=NaN;                                   %逻辑判断语句
rout(find(rout<a))=NaN;
Boutr=0;  Binr=0;  BoutQ=0;  BinQ=0;
for k=1:2:43
   fun10=legendre(k,0) ;
   fun0=fun10(2);
   fun=legendre(k,cos(theta));
   rfun=squeeze(fun(1,:,:));                            %将三维压缩成二维
   Qfun=squeeze(fun(2,:,:));
%-------------------------------------------------------
   Cr=1/a^k*(rin).^(k-1);                               %球内r分量
   Bkinr =- 1/2*fun0*Cr.*rfun;                          %第K项
   Binr=Binr + Bkinr;                                   %总和
   CQ=1/k /a^k*rin.^(k-1);                              %球内θ分量
   BkinQ= -1/2*fun0*CQ.*Qfun;
```

```
    BinQ=BinQ + BkinQ;
%-----------------------------------------
    Dr=a^(k+1) ./rout.^(k+2);                          %球外r分量
    Bkoutr =- 1/2*fun0*Dr.*rfun;
    Boutr=Boutr + Bkoutr;
    DQ=1/(k+1)  *a^(k+1)./rout.^(k+2);                 %球外θ分量
    BkoutQ = 1/2*fun0*DQ.*Qfun;
    BoutQ=BoutQ + BkoutQ;
end

L1=isnan(Binr);                                  %将非数元素找出并令其为零
Binr(find(L1==1))=0;    BinQ(find(L1==1))=0;
L2=isnan(Boutr);
Boutr(find(L2==1))=0;    BoutQ(find(L2==1))=0;
Br=Binr+Boutr;    BQ=BinQ+BoutQ;                         %合并为一个场B
By=Br.*cos(theta)-BQ.*sin(theta);            %换成直角坐标以作流线图
Bx=Br.*sin(theta)+BQ.*cos(theta);
vy=0;    vx=[0:0.01:0.35];
[Vx, Vy]=meshgrid(vx,vy);
cla
streamline(X,Y,Bx,By,Vx,Vy)
hold on
streamline(-X,Y,-Bx,By,-Vx,Vy)              %利用对称性画其余的象限
streamline(-X,-Y,-Bx,-By,-Vx,-Vy)
streamline(X,-Y,Bx,-By,Vx,-Vy)
title('B的场线')
figure
subplot(2,2,1)
surf(X,Y,Br)
title('Br分量')
view(-130,20)
subplot(2,2,2)
contour(X,Y,Br,10)
title('Br的等值线')
subplot(2,2,3)
```

```
surf(X,Y,BQ)
title('Bθ 分量')
subplot(2,2,4)
contour(X,Y,BQ,10)
title('Bθ 的等值线')
```

所得的图形是图 7.55。

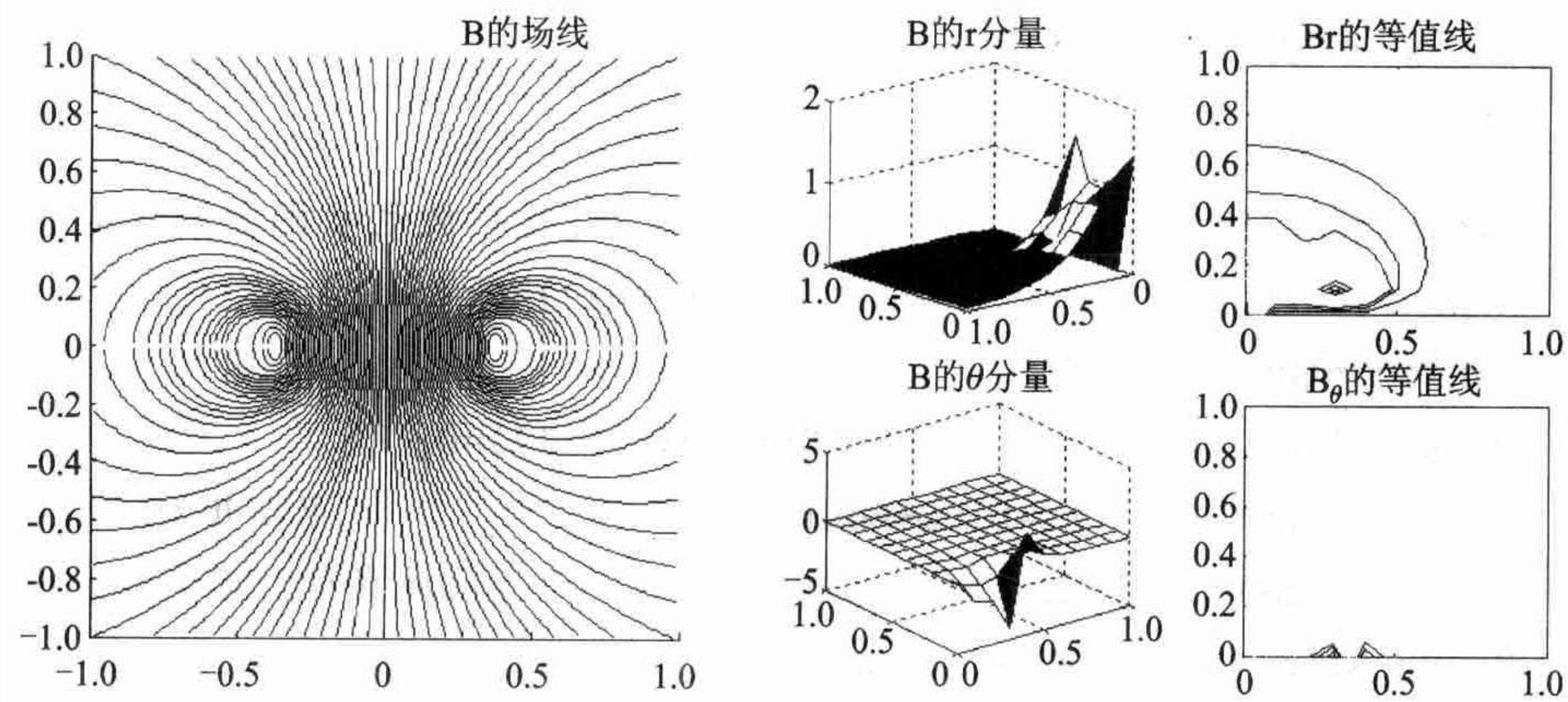


图 7.55 用勒让德函数表示的解析解的图形

2. 利用椭圆函数表示的磁感应强度解析表达式

利用毕奥-萨伐尔定律

$$\boldsymbol{B}=\frac{\mu_0}{4\pi}\oint\frac{I\mathrm{d}\boldsymbol{l}\times\boldsymbol{R}}{R^3}$$

求出圆环电流在空间的磁感应强度解析表达式，如图 7.56 建立坐标系。

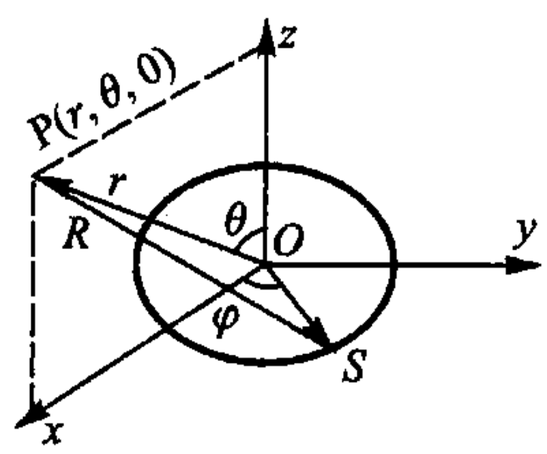


图 7.56 环形电流的示意图

积分计算得：[①]

① 曾令宏，张之翔. 圆环电流的磁场以及两共轴圆环电流之间的相互作用力. 大学物理，2002，21(9)：14—16

$$B_x = \frac{\mu_0 I}{4\pi}\frac{2\cos\theta}{\sin\theta\sqrt{a^2+r^2+2ar\sin\theta}}\left[\frac{a^2+r^2}{a^2+r^2-2ar\sin\theta}E-K\right]$$

$$B_y = 0$$

$$B_z = \frac{\mu_0 I}{4\pi}\frac{2}{\sqrt{a^2+r^2+2ar\sin\theta}}\left[\frac{a^2-r^2}{a^2+r^2-2ar\sin\theta}E+K\right]$$

其中，E, K 为椭圆函数：

$$\begin{aligned}
K &= \int_0^{\pi/2}\frac{\mathrm{d}x}{\sqrt{1-k^2\sin^2 x}} \\
&= \frac{\pi}{2}\left[1+\left(\frac{1}{2}\right)^2 k^2+\left(\frac{1\cdot 3}{2\cdot 4}\right)^2 k^4+\left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)^2 k^6+\cdots\right] \\
E &= \int_0^{\pi/2}\sqrt{1-k^2\sin^2 x}\mathrm{d}x \\
&= \frac{\pi}{2}\left[1-\left(\frac{1}{2}\right)^2 k^2+\left(\frac{1\cdot 3}{2\cdot 4}\right)^2\frac{k^4}{3}-\left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)^2\frac{k^6}{5}+\cdots\right] \\
k^2 &= \frac{4ar\sin\theta}{a^2+r^2+2ar\sin\theta}
\end{aligned}$$

下面的程序画出了解的图形 (图 7.57)，在程序中我们使用了 MATLAB 计算椭圆函数的指令 Ellipke。

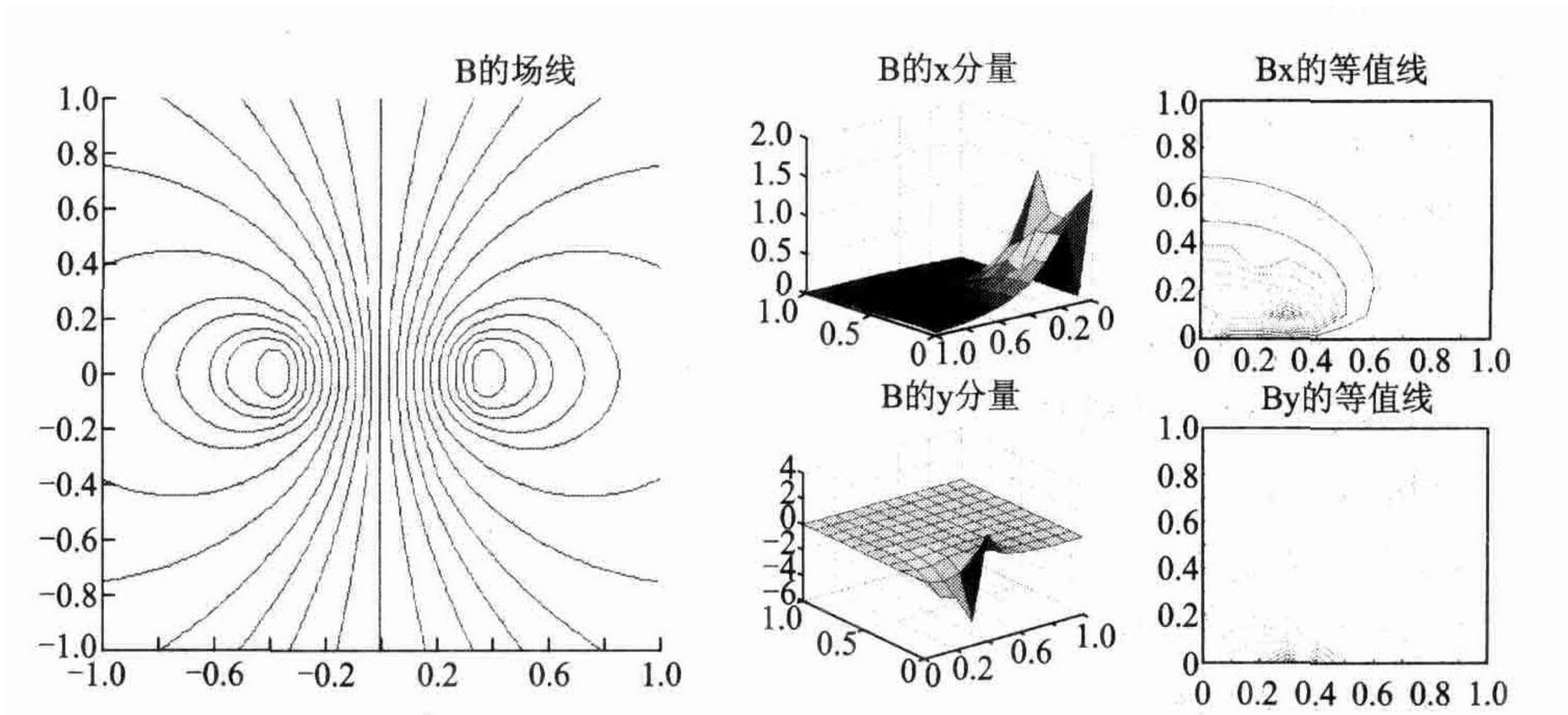


图 7.57 用椭圆函数表示的解析解的图形

```
a=0.35;
[X,Y]= meshgrid( 0:0.1:1);
```

```
XY=X.^2+Y.^2;
k2=4*a*X./(a^2+XY+2*a*X);
[K, E]=ellipke(k2);
aXY=a^2+XY-2*a*X;
aXY=aXY+eps*(aXY==0);
aE1=(a^2-XY)./aXY.*E;
aE2=(a^2+XY)./aXY.*E;
By=1/pi./sqrt(a^2+XY+2*a*X).*(aE1+K);
Bx=1/pi*Y./(X+eps*(X==0))./sqrt(a^2+XY+2*a*X).*(aE2-K); vy=0;
vx=[0.0001:0.01:0.35];
[Vx, Vy]=meshgrid(vx,vy);

streamline(X,Y,Bx,By,Vx,Vy)
hold on
streamline(-X,Y,-Bx,By,-Vx,Vy)
streamline(-X,-Y,-Bx,-By,-Vx,-Vy)
streamline(X,-Y,Bx,-By,Vx,-Vy)

title('B的场线')
figure
subplot(2,2,1)
surf(X,Y,Bx)
title('B的x分量')
view(-130,20)
subplot(2,2,2)
contour(X,Y,Bx,10)
title('Bx的等值线')
subplot(2,2,3)
surf(X,Y,By)
title('B的y 分量')
subplot(2,2,4)
contour(X,Y,By,10)
title('By的等值线')
```

还可以将结果做成三维的动画，程序中先用类似于前面的程序的方法，画

出三维的磁感应线和相应的等值线，然后作动画。磁感应线的旋转是利用改变相机视角的方法，指令是 camorbit，而显示不同位置的等值线的动画，则是用实时动画的方式。

```
a=0.35;
[X,Y,Z]=meshgrid(-1:0.1:1, -1:0.1:1 , 0:0.1:1);
rho2=X.^2+Y.^2; rho=sqrt(rho2);
theta=atan2(rho,Z);
r=sqrt(rho2+Z.^2);
rout=r;    rin=r;
rin(find(rin>a))=NaN;
rout(find(rout<a))=NaN;
Boutr=0;    Binr=0;    BoutQ=0;    BinQ=0;
for k=1:2:61
fun10=legendre(k,0) ;
   fun0=fun10(2);
   fun=legendre(k,cos(theta));
   rfun=squeeze(fun(1,:,:,:));
   Qfun=squeeze(fun(2,:,:,:));
%-------------------------------------------------------
   Cr=1/a^k*(rin).^(k-1);                                  %球内r分量
   Bkinr = - 1/2*fun0*Cr.*rfun;
   Binr=Binr + Bkinr;
   CQ=1/k  /a^k   *rin.^(k-1);                             %球内θ分量
   BkinQ= -1/2*fun0*CQ.*Qfun;
   BinQ=BinQ + BkinQ;
%-----------------------------------------
   Dr=a^(k+1) ./rout.^(k+2);                               %球外r分量
   Bkoutr =- 1/2*fun0*Dr.*rfun;
   Boutr = Boutr + Bkoutr;
   DQ=1/(k+1)  *a^(k+1)./rout.^(k+2);                      %球外θ分量
   BkoutQ = 1/2*fun0*DQ.*Qfun;
   BoutQ=BoutQ + BkoutQ;
end

L1=isnan(Binr);                                            %合并为一个场B
```

```
Binr(find(L1==1))=0;    BinQ(find(L1==1))=0;
L2=isnan(Boutr);
Boutr(find(L2==1))=0;    BoutQ(find(L2==1))=0;
Br=Binr+Boutr;    BQ=BinQ+BoutQ;
Bz=Br.*cos(theta)-BQ.*sin(theta);
Bxy=Br.*sin(theta)+BQ.*cos(theta);
phi=atan2(Y,X);
Bx=Bxy.*cos(phi);
By=Bxy.*sin(phi);
vy=0;    vx=[-0.3:0.1:0.3];
[Vx, Vy, Vz]=meshgrid(vx,vx,0);

cla
streamline(X,Y,Z,Bx,By,Bz,Vx,Vy,Vz)
view(-8,38)
hold on

B=sqrt(Bx.^2+By.^2+Bz.^2);
Zmin=min(Z(:));
hcont = contourslice(X,Y,Z,B,[ ],[ ],0.1);                    %等值线
set(hcont,'EdgeColor','red','LineWidth',.5);

axis vis3d
for i=1:20
   camorbit(10,0,'data',[0,0,1])                              %B线的动画
   drawnow
   pause(0.5 )
end

for k=1:20
   delete(hcont)
   hcont=contourslice(X,Y,Z,B,[ ],[ ],0.1+0.05*k); %等值线的动画
   pause(0.5)
end
```

图 7.58 是其中的一个画面。

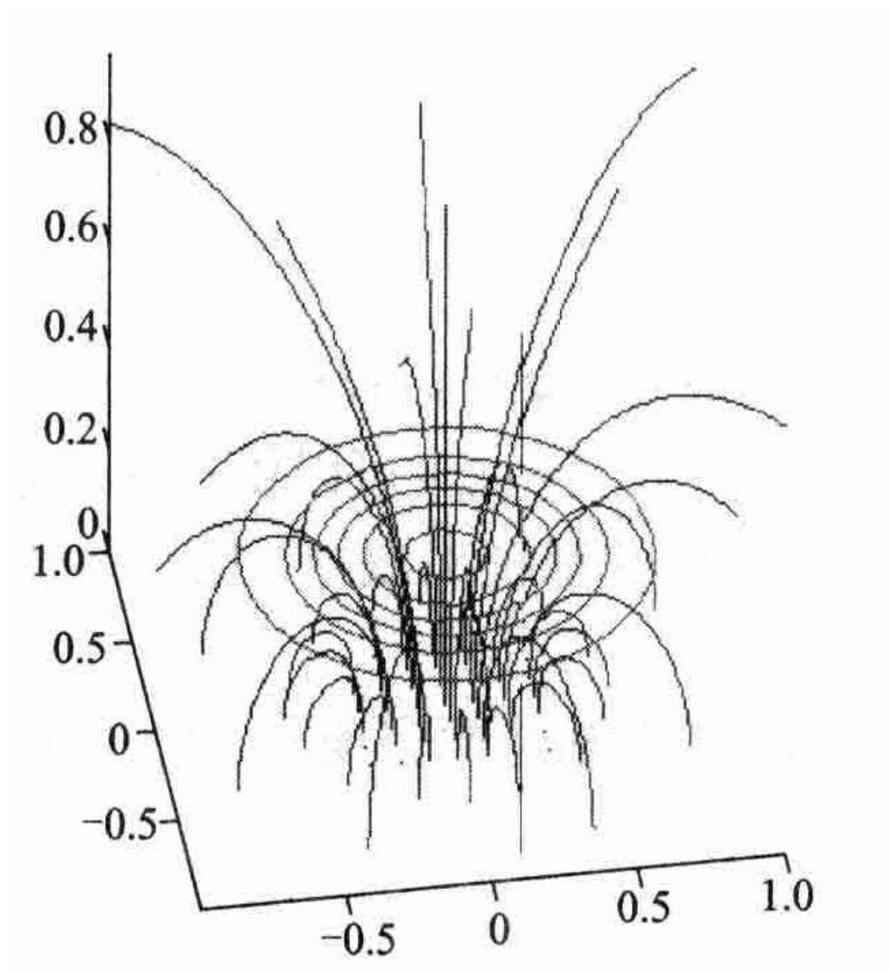


图 7.58 电流环磁感应线的动画的一个画面

§7.5.6 高斯积分

高斯积分法的基本思想是：在求积过程中，当节点数 n 是固定时，适当选取节点 $\{x_n\}$ 与求积系数 $\{A_n\}$，使求积公式具有最高的求积精度。

设要计算

$$I=\int_{-1}^{1} f(x)\mathrm{d}x$$

在以前讨论过的积分公式属于下面这种形式：

$$I \approx \sum_{n=1}^{N} w_n f(x_n) \tag{7.1}$$

其中

$$x_n=-1+2\frac{n-1}{N-1}$$

是等间隔的格点。例如，对于辛普森积分公式：

$$I=\frac{h}{3}\left(f_0+f_{2n}+2\sum_{k=1}^{n-1} f_{2k}+4\sum_{k=0}^{n-1} f_{2k+1}\right)$$

当 $N=3$ 时，可求出

$$x_1=-1,\ x_2=0,\ x_3=1;\quad w_1=w_3=\frac{1}{3},\ w_2=\frac{4}{3}$$

从辛普森法则的推导过程可以看出，当 f 是一个 3 次或更低次的多项式时上述公式是精确的。更一般的，如果一个以泰勒级数为基础的求积公式使用 N 个点，

那么此公式将对一个 $N-1$ 次多项式进行精确的积分。也就是说，可以选择 N 个权重 w_n 使之满足 N 个线性方程

$$\int_{-1}^{1} x^p \mathrm{d}x = \sum_{n=1}^{N} w_n x_n^p, \quad p = 0, 1, \cdots, N-1 \tag{7.2}$$

如果不要求积点是等间隔的，就可以择优选择 x_n，于是在建立求积公式时，就有 $2N$ 个参量 (N 个 x_n 和 N 个 w_n) 可以供选择。通过选择这些参量，使得 (7.1) 式对于从 0 到 $2N-1$ 的 p 都精确成立。这显然要比使用等间隔格点的效率高。

为了看出如何挑选 x_i，考虑勒让德多项式，它在区间 $[-1,1]$ 上是正交的。

$$\int_{-1}^{1} \mathrm{P}_i(x)\mathrm{P}_j(x)\mathrm{d}x = \frac{2}{2i+1}\delta_{ij} \tag{7.3}$$

由正交多项式的性质可知 $\mathrm{P}_i(x)$ 在 $[-1,1]$ 区间中有 i 个实根。于是任何一个 $2N-1$ 次或更低次多项式可以写成

$$f(x) = Q(x)\mathrm{P}_N(x) + R(x)$$

其中 Q, R 的次数小于或等于 $N-1$。于是将 Q 展成 P_i 的级数并利用勒让德多项式的正交性，得

$$I = \int_{-1}^{1} f(x)\mathrm{d}x = \int_{-1}^{1} (Q\mathrm{P}_N + R)\mathrm{d}x = \int_{-1}^{1} R\mathrm{d}x \tag{7.4}$$

如果我们取 x_i 为 P_N 的 N 个零点，应用 (7.1) 式给出

$$I = \sum_{n=1}^{N} w_n[Q(x_n)\mathrm{P}_N(x_n) + R(x_n)] = \sum_{n=1}^{N} w_n R(x_n) \tag{7.5}$$

剩下的工作就是选择 w_n，使得 R 的积分精确的成立。即当 x_n 是 P_n 的根时，w_n 满足线性方程组 (7.2)。可以证明，解出的 w_n 满足：

$$w_n = \frac{2}{(1-x_n^2)[\mathrm{P}'_N(x_n)]^2}$$

这就得到了高斯-勒让德求积公式。对于积分限 $x=a$ 和 $x=b$ 之间的积分，作变量代换

$$t = -1 + 2\frac{x-a}{b-a}$$

就把积分化成所需要的形式。

当被积函数取特殊形式时，其他正交多项式也提供了有用的高斯求积公式。例如拉盖尔多项式 $\mathrm{L}_i(x)$ 在区间 $[0,+\infty)$ 上以 e^{-x} 为权函数正交，它们导致了高斯–拉盖尔求积公式

$$\int_0^{\infty} \mathrm{e}^{-x} f(x)\mathrm{d}x \approx \sum_{n=1}^{N} w_n f(x_n) \tag{7.6}$$

其中各个 x_n 是 $\mathrm{L}_N(x)$ 的根。w_n可以通过解一线性方程组得到。类似地，厄米多项式对形式为

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} f(x)\mathrm{d}x \approx \sum_{n=1}^{N} w_n f(x_n)$$

的积分提供了高斯–厄米求积公式。

归纳起来，高斯求积公式的使用分为两步：

(1) 判断能否用高斯积分以及用那种高斯积分。

一般来说，被积函数是连续函数或者被积函数是连续函数与正交多项式的权函数的乘积，都可以用高斯积分。如果被积函数变化很快，可以分段应用高斯求积公式。若被积函数是一些离散的数据，就只好用以前讨论的梯形公式一类的方法。

(2) 从数值计算的书中查找 x_n, w_n 的值，也有计算这些数据的程序库。根据精度选择要用的数据，计算出被积函数相应的函数值 $f(x_n)$，再按照公式求和就完成了积分。

例如，下面用 3 点高斯–勒让德求积公式来计算积分

$$I = \int_0^3 (1+t)^{1/2}\mathrm{d}t = 4.666\ 67 \tag{7.7}$$

作变数代换

$$x = -1 + \frac{2}{3}t$$

得到

$$I = \frac{3}{2}\int_{-1}^{1} \left(\frac{3}{2}x + \frac{5}{2}\right)^{\frac{1}{2}} \mathrm{d}x \tag{7.8}$$

对于 $N=3$，高斯–勒让德求积公式的坐标值和权重为

$$x_1 = -x_3 = 0.774\ 597,\ x_2 = 0;\ w_1 = w_3 = 0.555\ 556, w_2 = 0.888\ 889$$

这时高斯–勒让德求积公式直接求值得出 $I = 4.666\ 83$，而用辛普森法则令 $h = 1.5$ 来求 (7.7) 式之值则给出 4.662 28。因此高斯求积比辛普森法则更精确，大约差 27 倍，而所需的计算被积函数的次数则相同 (3 次)。

§7.6 练 习

1. 对热传导方程一节中的例子，计算不稳定的情况。

2. 对弦振动方程的例子，计算不稳定的情况。

3. 解方程

$$\begin{cases} u_{xx}+u_{yy}=x\mathrm{e}^y \quad (0<x<2,0<y<1) \\ u(0,y)=0, \quad u(2,y)=2\mathrm{e}^y \\ u(x,0)=x, \quad u(x,1)=\mathrm{e}x \end{cases}$$

取 $\triangle x=1/3,\triangle y=1/5$。

4. 解方程

$$\begin{cases} u_t=u_{xx} \quad (0<x<1,0<t<0.5) \\ u(x,0)=\sin\pi x \\ u(0,t)=u(1,t)=0 \end{cases}$$

分别取 $\triangle x=0.1,\triangle t=0.01$ 和 $\triangle x=0.1,\triangle t=0.001$ 进行计算。并与解析解 $u(x,t)=\exp(-\pi^2t)\sin(\pi x)$ 比较。

5. 用差分法和 PDETOOL 解方程

$$\begin{cases} u_{tt}=a^2u_{xx} \quad (0<x<l,0<t) \\ u(0,t)=0, \quad u(l,t)=A\sin\omega t \\ u(x,0)=0, \quad u_t(x,0)=0 \end{cases}$$

问题中的参数由自己选择。

6. 在圆域 $\rho<a$ 上用偏微分方程工具箱求解方程 $\triangle u=-xy$，边界条件是 $u(\rho=0)=0$。

7. 用偏微分方程工具箱求解如下的定解问题

$$\begin{cases} u_t=a^2u_{xx} \\ u(0,t)=0, \quad u_x(l,t)=0 \\ u(x,0)=u_0x/l \end{cases}$$

8. 用辛普森积分公式和高斯-勒让德求积公式计算

$$\int_{-1}^{1}(1-x^2)^{1/2}\mathrm{d}x=\frac{\pi}{2}$$

注意对被积函数在 $x=\pm1$ 附近的行为需要特别小心。对不同的坐标点数，比较这些方法的精度和效率。注意这个积分可以用一个“1”点的高斯-切比雪夫求积

公式准确求值，这个公式的形式为

$$\int_{-1}^{1}(1-x^2)^{1/2}f(x)\mathrm{d}x=\sum_{n=1}^{N}w_nf(x_n)$$

其中

$$x_n=\cos\frac{n}{N+1}\pi,\quad w_n=\frac{\pi}{N+1}\sin^2\frac{n}{N+1}\pi$$

9. 根据量子力学，一维线性谐振子的波函数是

$$\psi_n=N_n\mathrm{H}_n(\xi)\mathrm{e}^{-\frac{1}{2}\xi^2},\quad n=0,1,2,\cdots$$

其中 N_n 为归一化常数，$\mathrm{H}_n(\xi)$ 是厄米多项式。画出谐振子的前 4 个波函数（作图时可以不必考虑 N_n），并将图形与 §5.10.3 节数值计算所得图形作比较。

10. 均匀细圆环，半径为 a，环上带有 $4\pi\varepsilon_0q$ 单位的电荷，它在空间的电势分布为①

$$u(r,\theta)=\begin{cases}\dfrac{q}{a}\displaystyle\sum_{l=0}^{\infty}(-1)^l\frac{(2l)!}{2^{2l}l!l!}\left(\frac{r}{a}\right)^{2l}\mathrm{P}_{2l}(\cos\theta) & (r<a)\\[2ex] \dfrac{q}{a}\displaystyle\sum_{l=0}^{\infty}(-1)^l\frac{(2l)!}{2^{2l}l!l!}\left(\frac{a}{r}\right)^{2l+1}\mathrm{P}_{2l}(\cos\theta) & (r>a)\end{cases}$$

画出空间的电场线分布。

① 梁昆淼. 数学物理方法. 3 版. 北京：高等教育出版社，1998

第八章　蒙特卡罗方法

蒙特卡罗方法 (MC) 又称之为随机取样法，统计模拟法，是利用随机数的统计规律来进行计算和模拟的方法。它可用于数值计算，也可用于数值仿真。

§8.1　蒙特卡罗方法的发展过程

第一个与 MC 方法有关的实验是 1977 年蒲丰（Comte de Buffon）做的投针实验。在平面上画一组间距为 d 的平行线，向平面随意投掷长度为 $l(l<d)$ 的细针，测量针与平行线相交的概率。经过大量的实验并进行了数学推算，他证明概率 p 是

$$p=\frac{2l}{\pi d}$$

后来拉普拉斯提出，如果用实验测出概率，可以用这个方法来估算 π 的值，这成了 MC 方法思想的雏形。

后来，概率论与随机行走理论在 MC 方法上的应用取得了巨大进展。例如 Curant 等人证明了某种随机行走与解某类偏微分方程的等效性。1930 年 Fermi 进行了一些数值实验，现在称之为 MC 计算，在研究当时新发现的中子的行为时，他通过抽样计算中子与浓缩物质相互作用的可能性。这导致了对这个现象的物理意义的实质性的解释，并建立了中子扩散与输运的理论。

第二次世界大战期间，在 Newmann，Fermi，Ulam 和 Metropolis 等人的带动下，加之现代计算机的出现，使 MC 方法有了快速的发展。在 1940 年代后期与 1950 年代早期，人们对 MC 方法表现了极大的兴趣。描述这种方法以及如何将其用于解决统计物理，输运问题，经济模型等领域的论文大量发表。不幸的是，当时的计算机除了进行一些示范性的工作，在很多领域还不符合实际应用的需求。后来，计算机的计算能力越来越强，人们开展了规模越来越大的计算并从失败中不断改进。

同时，理论的发展与实际可用的有效的误差修正理论意味着MC方法应用已经超出了纯粹依赖计算机的计算速度与存储能力的发展。两个最有影响的进展是输运方程在方法上的改进，尤其是可靠的“重要抽样法”与 Metropolis 算法的发明。最终的结果突破了 1940 年代的先驱者们最乐观的估计。

为了理解 MC 方法的思想，我们用 MC 方法来计算圆周率。如图 8.1 所示，

单位圆的面积是 π，它在第一象限的面积为 π/4，因此有积分式

$$\pi = 4\int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2 \cdot \theta(1 - x_1^2 - x_2^2)$$

其中 θ 是单位阶跃函数，

$$\theta(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x < 0) \end{cases}$$

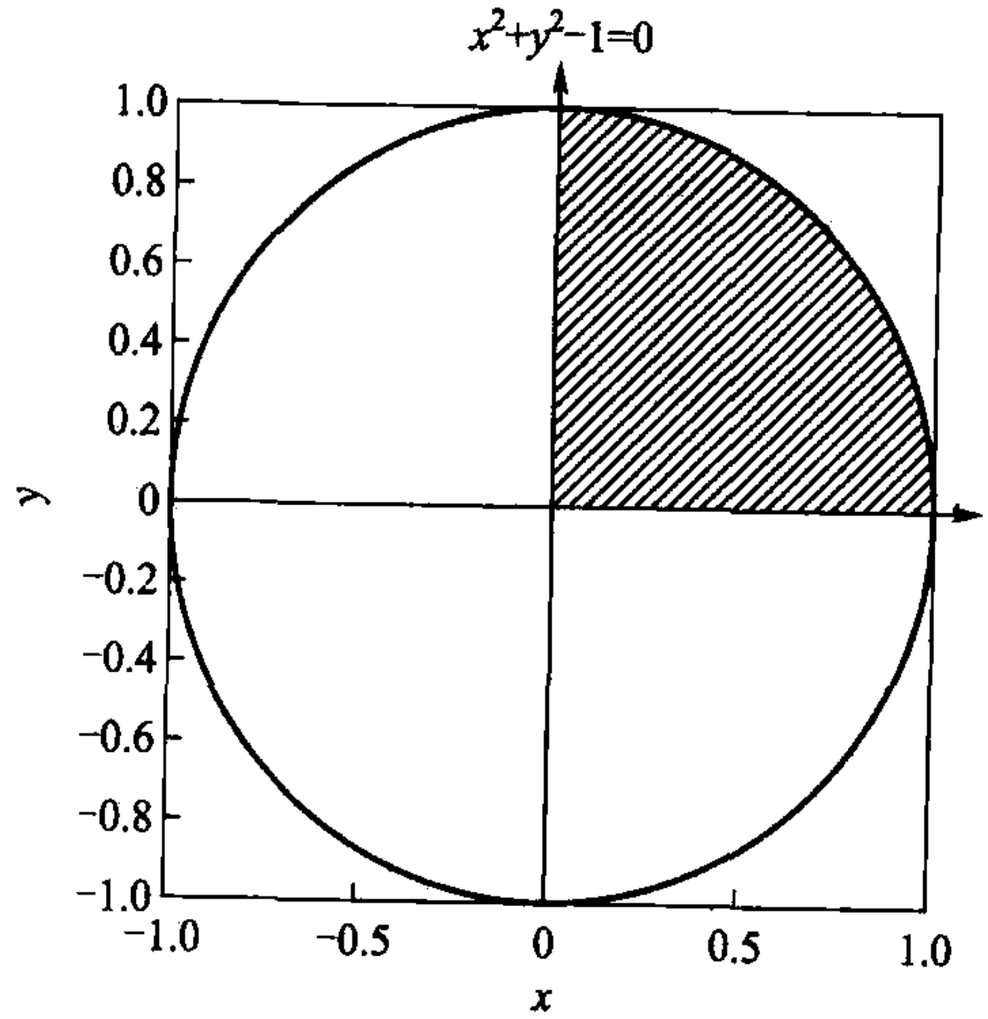


图 8.1 用单位圆面积计算 π

计算时，在第一象限内的正方形 $0 \leqslant x \leqslant 1$，$0 \leqslant y \leqslant 1$ 内生成二维的等概率分布的随机数 (x, y)，统计所有满足 $x^2 + y^2 < 1$ 的点数，计算它们与正方形内的总点数之比，就是所求。

用 MC 方法计算这个二维积分的指令是

```
p=4/1000000*length(find(sum(rand(2,1000000).^2)<1))
```

这里取 $N = 10^6$。

几次计算的结果如下：3.143 2，3.145 3，3.140 3，3.141 4，3.143。其平均值是 3.142 6。可见在小数点后第三位数已经有较大的偏差。

用 MC 方法解问题大致可以分为两步：

(1) 为物理问题建立一个关于随机变量 X 的模型，问题的解是变量 X 出现的概率或 X 数学期望值。

(2) 用计算机进行数值模拟实验，也就是对模型进行随机抽样，产生随机变量 X 的值，求出 X 的概率或平均值作为问题的近似解。

例如上面在求 π 值时，建立的模型就是计算二维积分，使用的随机变量就是点的坐标，点在圆周线内出现的概率就对应要计算的 π 值。

使用 MC 方法，有两点要注意，第一是要生成所需要的具有特定分布的随机数，这里应用的是二维的等概率分布的随机数；第二是误差估计，例如上面计算的 π 值，在小数后第三位数就会出现误差，可见其精度有限。由于这是一种按概率分布抽样后求统计平均的方法，误差的存在是必然的，但不意味这种方法没有价值，在没有其他方法可用或其他方法很难应用的情况下，MC 方法仍然可以得到具有几位有效数字的定量的结果。概率论中的大数定理和中心极限定理是 MC 方法的依据。大数法则反映了大量随机数之和的性质。而中心极限定理可以判断出，在抽样数目有限但是足够大的时候，MC 的估计值是如何分布的。

§8.2 随机变量、密度函数与分布函数

下面介绍 MC 方法中用到的几个概念。

1. 随机变量

随机变量的特点是，变量的每次取值无法事先预言,但它的取值分布是已知的,即知道它取某一个值的概率是多少。

对于连续分布的随机变量 x，如果它在 $[a,b]$ 间取值的概率为

$$p(a \leqslant x \leqslant b) = \int_a^b f(x)\mathrm{d}x$$

则 $f(x)$ 是随机变量 x 概率分布密度函数。

分布函数定义为

$$F(x) = \int_{-\infty}^{x} f(\xi)\mathrm{d}\xi$$

二者的关系是

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}$$

如果 $f(x)$ 是归一化的分布密度函数，则该函数在 ξ 的全部变化范围内积分为 1。利用分布函数来确定随机变量 ξ 在 $[a,b]$ 间取值的概率为

$$p(a \leqslant \xi \leqslant b) = F(b) - F(a)$$

生成 [0, 1] 间的等概率随机数的指令是 rand，它可以生成一维的随机数，也可以生成多维的随机数。对于一维的随机数可以用指令 hist 来检验它所生成的数是否为等概率分布。操作指令如下：

```
>>A=rand(1,10000);    hist(A)
```

2. 正态分布

若随机变量 x 的概率密度是

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (-\infty < x < \infty)$$

其中 μ 是任意常数，σ 是大于零的常数。则称 x 是服从参数 μ，σ 的正态分布，记作 $x \sim N(\mu, \sigma^2)$。其图形如图 8.2 所示。可以证明

$$\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{+\infty} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}\mathrm{d}x = 1$$

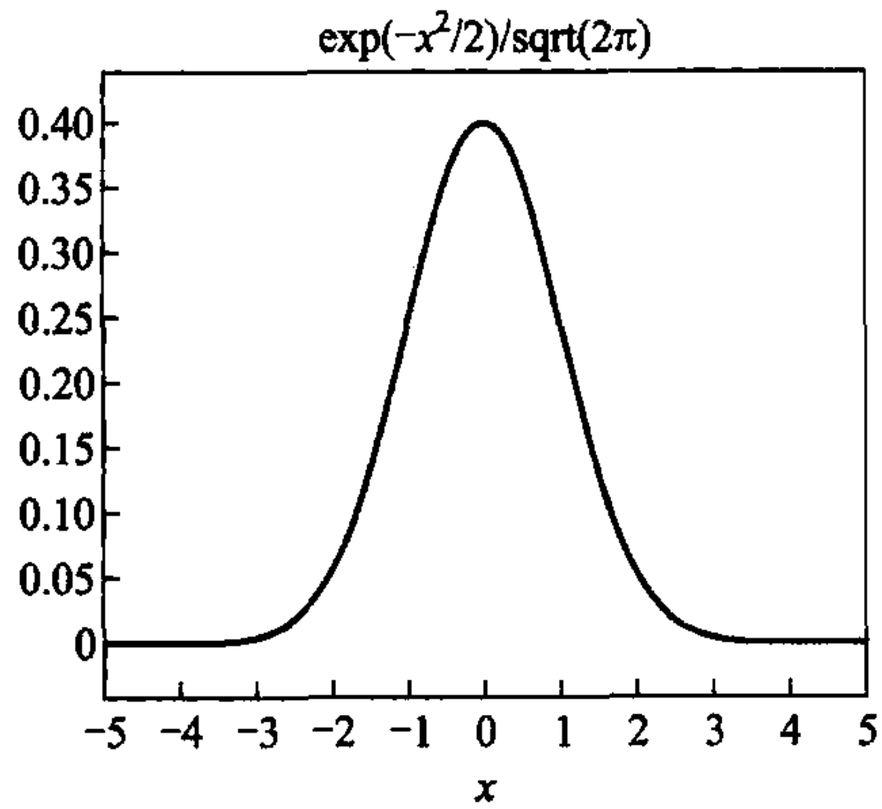


图 8.2 正态分布曲线

它的分布函数是

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x} \mathrm{e}^{-\frac{(\xi-\mu)^2}{2\sigma^2}}\mathrm{d}\xi$$

当 $\mu = 0$，$\sigma = 1$ 时，称为标准正态分布，记为 $N(0, 1)$。任何一个一般的正态分布 $N(\mu, \sigma)$ 都可以通过线性变换转化为标准正态分布。即若 $X \sim N(\mu, \sigma^2)$，则有 $Y = \dfrac{X-\mu}{\sigma} \sim N(0, 1)$。注意有 $F(-x) = 1 - F(x)$。

指令 normpdf(x, μ, σ) 和 normcdf(x, μ, σ) 分别计算正态分布的密度函数与分布函数。

由于

$$P(|X \leqslant 1|) = 2F(1) - 1 = 0.682\,6$$

$$P(|X \leqslant 2|) = 2F(2) - 1 = 0.954\,4$$

$$P(|X \leqslant 3|) = 2F(3) - 1 = 0.997\,4$$

所以 X 的取值几乎全部集中在 $[-3, 3]$ 的区间。这叫 3σ 原则，也叫三倍标准差原则。

函数 $g(x)$ 的数学期望值定义为它的平均值

$$E\{g\} = \int g(x)\mathrm{d}F(x) = \int g(x)f(x)\mathrm{d}x$$

这里 $F(x)$ 是 x 的分布函数，如果 x 是在 $[a,b]$ 区间均匀分布的随机变量，即 $\mathrm{d}F=\mathrm{d}x/(b-a)$，则这时的期望值为

$$E\{g\}=\frac{1}{b-a}\int_a^b g(x)\mathrm{d}x$$

随机变量 ξ 数学期望值就是平均值，定义为

$$E\{\xi\}=\int \xi\mathrm{d}F(\xi)=\int \xi f(\xi)\mathrm{d}\xi$$

函数或变量的方差定义为

$$D\{f\}=E\{(f-E\{f\})^2\}=\int [f-E\{f\}]^2\mathrm{d}F$$

标准误差是方差的平方根，也称为均方根误差，也有的称为均方差。

§8.3 大数定理与中心极限定理

大数定理依据的客观事实是，在大量随机试验中，事件发生的频率分布在测量值的算术平均值附近，且具有稳定性。如大量抛掷硬币正面出现频率，生产过程中的废品率，字母使用频率，子弹的命中率等。

1. 大数定理

大数定理可叙述为：设随机变量 x_1，x_2，$\cdots$，x_n，$\cdots$ 相互独立，且具有相同的数学期望值和方差

$$E(x_k)=\mu,\quad D(x_k)=\sigma^2\qquad (k=1,2,\cdots)$$

则序列

$$\bar{x}=\frac{1}{n}\sum_{k=1}^{n}x_k$$

依概率收敛于 μ。

依概率收敛比高等数学中普通意义上的收敛弱些，它具有某种不确定性。它表示在 n 充分大时，事件发生的概率很大，接近于 1，但并不意味事件一定会发生。更严格的表述如下。

大数定理：设随机变量 x_1，x_2，$\cdots$，x_n，$\cdots$ 相互独立且有相同的数学期望值与方差：

$$E(x_k)=\mu,\quad D(x_k)=\sigma^2\qquad (k=1,2,\cdots)$$

作前 n 个随机变量的算术平均 $\bar{x}=\dfrac{1}{n}\sum\limits_{k=1}^{n}x_k$，则对任意的 $\varepsilon>0$，有

$$\lim_{n\to\infty}P\{|\bar{x}-\mu|<\varepsilon\}=\lim_{n\to\infty}P\left\{\left|\frac{1}{n}\sum_{k=1}^{n}x_k-\mu\right|<\varepsilon\right\}=1$$

定理表明随机变量 $x_1, x_2, \cdots, x_n, \cdots$ 的算术平均接近它的数学期望值。

大数定理表明，抽取的样本越多，计算结果就会越接近正确值。由于不可能有无穷多个样本，所以有必要根据选取样本数来估算结果的收敛程度，判断出它的误差水平。也就是需要知道 $\frac{1}{n}\sum_{k=1}^{n} x_k$ 的极限分布，而中心极限定理就可用来解决这个问题。

2. 中心极限定理

人们从实际中发现，生产与科学实验中很多随机变量的概率分布都可以近似地用正态分布来描述。例如，在生产条件不变的情况下，产品的抗压强度、口径、长度等指标；同一种生物体的身长、体重等指标；同类种子的重量；子弹的弹着点分布；某个地区的年降水量；以及理想气体分子的速度分量等。

这表明，如果一个随机现象是由大量的相互独立的随机因素的综合影响所形成，而其中每一个因素在总的影响中所起的作用又是微小的。这类随机变量一般都服从或近似服从正态分布。

人们熟知的伽尔顿板实验就是典型例证，小球下落时的初速度与位置具有随机性，下落时会碰到那个钉子，会弹跳到那个方向都有随机性，下落时会碰到几个钉子也具有随机性。所有这些相互独立的随机因素的综合影响共同决定了小球最终到达的位置，而其中的任一个因素都只能在一段短暂的时间发挥作用。所以大量的小球的落点分布就形成了正态分布。这就是中心极限定理所描述的现象。

中心极限定理：设随机变量 $x_1, x_2, \cdots, x_n, \cdots$ 相互独立且有相同的数学期望值与方差：

$$E(x_k) = \mu, \quad D(x_k) = \sigma^2 \quad (k = 1, 2, \cdots)$$

则随机变量

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

在 n 趋于无限大时，服从参数为 μ 和 σ^2 的正态分布 $N(\mu, \sigma^2)$。

将该定理应用到抽样调查，就有这样一个结论：如果抽样对象的数学期望值 μ 和方差 σ^2 是完全确定的，无论抽样对象服从什么分布，从中抽取 n 个样本时，只要 n 足够大，其样本平均数的分布就趋于数学期望为 μ，方差为 σ^2 的正态分布。

由此看来，在奥林匹克运动会上，有时候夺冠呼声最高的选手竟会在关键的一枪上打出极低的靶环数甚至出现脱靶。这种看似不可思议的现象其实就是中心极限定理一个简单例证，因为再优秀的射击运动员也不能保证枪枪命中靶心，他们的弹着点肯定是一个正态分布，只是方差不同而已。

MATLAB 的指令 rand 生成的一维等概率分布的随机数，它服从正态分布，我们用它来检验一下中心极限定理。步骤如下：

(1) 生成 1000 个随机数组成的数列 R。作出它的统计直方图，这相当于每次只取一个数作平均所得的结果。

(2) 在数列 R 中，每次任取 6 个数相加，其和又是一个随机数，共取 1000 次，得到随机数列 Y。选取 6 个任意数的方法是，先生成一个 6×1000 整数随机数列，按照这个数列来选取 R 中的元素。

(3) 作 Y 统计直方图，程序与图形如下 (图 8.3)。

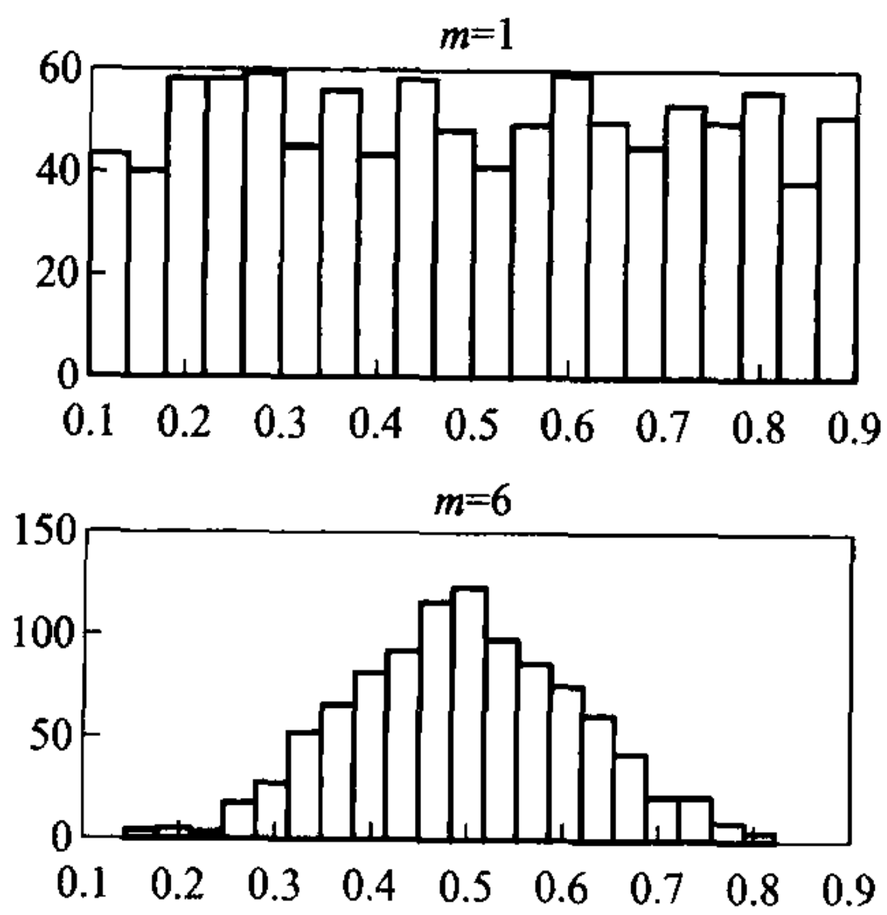


图 8.3 中心极限定理的数据模拟

```
R=rand(1,1000);
subplot(2,1,1)
hist(R,20)
Z=ceil(1000*rand(6,1000));
for k=1:1000
  kk=Z(:,k);
  Y(k)=(R(kk(1))+R(kk(2))+R(kk(3))+...
  R(kk(4)) +R(kk(5)) +R(kk(6)))/6;
end
 subplot(2,1,2)
 hist(Y,20)
```

从图形可以看到，当 $m=6$ 时，分布已经比较接近正态分布，在练习中，要求读者取不同的 m 值，计算它的统计分布图并与高斯曲线进行拟合。

中心极限定理还可以表述为：设随机变量 $x_1, x_2, \cdots, x_n, \cdots$ 相互独立且

有相同的数学期望值与方差：

$$E(x_k)=\mu,\quad D(x_k)=\sigma^2\qquad(k=1,2,\cdots)$$

则对于任意实数 λ，当 $n\to\infty$ 时有

$$\lim_{n\to\infty}P\left(\frac{\frac{1}{n}\sum\limits_{k=1}^{n}x_k-\mu}{\frac{\sigma}{\sqrt{n}}}<\lambda\right)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\lambda}\mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t$$

可改写为

$$\begin{aligned}&\lim_{n\to\infty}P\left(-\lambda<\frac{\frac{1}{n}\sum\limits_{k=1}^{n}x_k-\mu}{\frac{\sigma}{\sqrt{n}}}<\lambda\right)\\&=\lim_{n\to\infty}P\left(\frac{\frac{1}{n}\sum\limits_{k=1}^{n}x_k-\mu}{\frac{\sigma}{\sqrt{n}}}<\lambda\right)-\lim_{n\to\infty}P\left(\frac{\frac{1}{n}\sum\limits_{k=1}^{n}x_k-\mu}{\frac{\sigma}{\sqrt{n}}}<-\lambda\right)\\&=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\lambda}\mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t-\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{-\lambda}\mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t\\&=\frac{2}{\sqrt{2\pi}}\int_{0}^{\lambda}\mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t\end{aligned}$$

于是有

$$\lim_{n\to\infty}P\left(\left|\frac{1}{n}\sum_{k=1}^{n}x_k-\mu\right|<\frac{\lambda\sigma}{\sqrt{n}}\right)=\frac{2}{\sqrt{2\pi}}\int_{0}^{\lambda}\mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t$$

若记

$$\frac{2}{\sqrt{2\pi}}\int_{0}^{\lambda}\mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t=1-\alpha\tag{8.1}$$

则表示，当 n 很大时，不等式

$$\left|\frac{1}{n}\sum_{k=1}^{n}x_k-\mu\right|<\frac{\lambda\sigma}{\sqrt{n}}\tag{8.2}$$

成立的概率为 $1-\alpha$，通常称 α 为可信度，$1-\alpha$ 就是可信水平。根据 (8.1) 式，可以利用 MATLAB 计算误差函数的指令 erf 得到

$$\alpha=1-\mathrm{erf}\left(\frac{\lambda}{\sqrt{2}}\right)$$

例如对于 $\lambda = 0.674\,5, 1.960\,0, 2.326\,3, 2.575\,8$ 分别算得 $\alpha = 0.5, 0.05, 0.02, 0.01$。特别称 $\alpha = 0.5$ 时的误差 $0.674\,5\sigma/\sqrt{n}$ 为概率误差。式 (8.2) 表示，$\frac{1}{n}\sum_{k=1}^{n} x_k$ 收敛到 μ 速度的阶为 $O\left(\frac{1}{\sqrt{N}}\right)$。

记 $X = \frac{1}{n}\sum_{k=1}^{n} x_k$，以 $E(X)$ 表示 X 的平均值，考虑其相对误差

$$\frac{X - E(X)}{E(X)} \leqslant \frac{\lambda\sigma}{\sqrt{n}E(X)} = \varepsilon_n$$

此时，$E(X)$，σ 都是未知量，作为近似，以 $\sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - X)^2}$ 替代 σ，以 X 替代 $E(X)$，得

$$\varepsilon_n = \frac{\lambda\sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - X)^2}}{X\sqrt{n}}$$

所以给定可信度 α 以后，误差 ε_n 由 σ 和 $\sqrt{n}$ 决定。要减小 ε_n，就要增大 n 或减小 σ。对于固定的 σ，要提高精度一位数字（即 10 倍），就要增加 100 倍的工作量。因此，单纯增大 N 不是最有效的办法。而将 σ 减小一半，则误差也会减小一半，这相当于 n 增大 4 倍的效益。因此，降低 σ 的各种技巧引起了人们极大的兴趣。常见的技巧有“重要抽样”，“系统抽样”，“相关”，“对偶变数”，“半解析方法”，“统计估计”，“分裂与俄罗斯轮盘赌”等。不过也要注意，这种降低方差的技巧，往往会使观察一个子样的时间增加。因此，在固定的时间内，会使观察的样本的数目减小。所以，一种方法的优劣，不能光看方差降低了多少，而要由方差和观察一个样本费用 C（使用计算机的时间）来衡量，这就是MC方法中引用效率 $\eta = \sigma^2 \cdot C$ 的原因。

§8.4 随机数与随机抽样

要运用 MC 方法，首先要解决随机数的产生与随机抽样的方法。

真正的随机数除统计规律外无任何其他规律可循，例如单位时间内放射源释放的粒子数目。计算机上使用的随机数实际是伪随机数，是按照某种算法产生的似乎是随机的数。这种数存在某种算法规律，也有一定周期性，但是如果它的周期足够长，且在使用中不会出现，那么它也是实用的。

已经有众多的用计算产生伪随机数的方法，通常是产生在 [0, 1] 之间等概率分布的随机数，例如 MATLAB 指令 rand 就是产生 [0, 1] 之间的一维或高维均匀分布的随机数。下面简单介绍一下等概率分布随机数的几个方法。

§8.4.1 随机数的产生

1. 线性同余法 (乘加同余法)

递推公式是

$$X_{k+1} = (\lambda X_k + C) \bmod M$$

$$r_{k+1} = \frac{X_{k+1}}{M}$$

这里 M, λ, C 都是整数，M 是模，λ 是乘子，C 增量，X_0 是随机数列的种子，r_{k+1} 为求得的随机数。由于 X_{k+1} 是 $\lambda X_k + C$ 除以 M 所得的余数，所以小于 M。因此 r_{K+1} 肯定分布在 [0, 1] 之间。这个随机数列最多只能有 M 个不同的数，就是说，序列 r_k 的周期 T 小于 M。周期 T 的长短，以及在一个周期内序列 r_k 的随机性的好坏，同参数 M, λ, C 的选取有关。例如可以选取

$$\lambda = 1\ 103\ 515\ 245, \quad C = 12\ 345, \quad M = 2^{32}$$

目前，线性同余法仍然是计算机常用的算法。

2. 组合发生器

用一个随机数列来“扰乱”另一个随机数列会使随机数列的周期变长，随机性也增强。在对随机数要求较高的情况下可以考虑采用这种方法。

3. 混沌法

Logestic 模型中的迭代关系 $x_{k+1} = ax_k(1 - x_k)$，当 $3.569\ 9 < a < 4$ 时迭代值会进入混沌状态，理论上可以看成是真正的随机数，但不是均匀分布。再作一变换，得到均匀分布的随机数：

$$y(x) = \frac{2}{\pi} \arcsin \sqrt{x}$$

经检验，其独立性很好，由于计算位数有限，仍有周期性。如果与组合发生器结合效果更好。

任何一种产生随机数的方法都要经过检验，检验内容有均匀性检验即检查它在 [0, 1] 之间是否均匀分布，以及独立性检验即检查随机数的相关性。

§8.4.2 随机抽样

实际工作中需要的随机数可能具有不同的分布函数 $F(x)$，通过随机抽样，就可以利用在区间 [0, 1] 之间均匀分布的随机数 r_k 产生具有各种分布函数的随机数。举例如下：

1. 连续分布的随机取样

设连续随机变量的分布函数为 $F(x)$，归一化的密度函数为 $f(x)$，按定义有

$$F(x) = \int_{-\infty}^{x} f(t)\mathrm{d}t$$

这里 $F(x)$ 是在 [0, 1] 区间的随机数。若 r_i 为 [0, 1] 区间的均匀分布随机数，令

$$F(x) = r_i$$

得到

$$x_i = F^{-1}(r_i) \tag{8.3}$$

则 x_i 就是具有密度函数 $f(x)$ 的随机数。式中 F_{-1} 表示反函数。

例 1 在 $[a, b]$ 区间均匀分布的随机取样。

在 $[a, b]$ 区间均匀分布的概率密度函数为

$$f(x) = \begin{cases} 0 & (x < a, x > b) \\ \dfrac{1}{b-a} & (a \leqslant x \leqslant b) \end{cases}$$

分布函数为

$$F(x) = \frac{x-a}{b-a}$$

依据式 (8.3)，如果 r_i 是 [0, 1] 之间均匀分布的随机数，令

$$\frac{x-a}{b-a} = r_i$$

得到

$$x_i = (b-a)r_i + a$$

可见，如果想得到 $[0, 2\pi]$ 之间均匀分布的随机数 φ_i，则

$$\varphi = 2\pi r_i$$

这意味事件的发生在平面上是各向同性的。

例 2 指数分布的随机取样。

指数分布的概率密度函数是

$$f(x) = \lambda \mathrm{e}^{-\lambda x}$$

相应的分布函数是

$$F(x) = 1 - \mathrm{e}^{-\lambda x}$$

同样令

$$1 - \mathrm{e}^{-\lambda x} = r_i$$

得

$$x_i = -\frac{1}{\lambda}\ln(1 - r_i)$$

由于 $1 - r_i$ 与 r_i 相同的分布，上式改写为

$$x_i = -\frac{1}{\lambda}\ln(r_i)$$

2. 舍取法

上面的方法要计算反函数，如果反函数很难计算，则这个方法就不好用。下面的方法就不需要计算反函数，也能产生符合要求的随机数。

如图 8.4 所示，如果要产生在 0 和 1 之间按 $f(x)$ 分布的 x，设 $f(x)$ 的极大值为 M，可以画一个矩形，宽为 1 高为 M，曲线 $f(x)$ 将包含在这个矩形内。在矩形内产生均匀的投点，如果这个点在 $f(x)$ 曲线之下，则这个点就保留，反之就舍去。具体的做法是，产生 [0, 1] 间的均匀分布的两个随机数 t'，t''，由它们形成的投点坐标是 $P(r', r'')$，

$$r' = t', \quad r'' = Mt''$$

如果 $r'' < f(r')$，则这个点在曲线 $f(x)$ 之下，随机变量的取值为 $\eta = r'$，否则就舍弃它，重新选点。可以证明这样产生随机数符合要求。

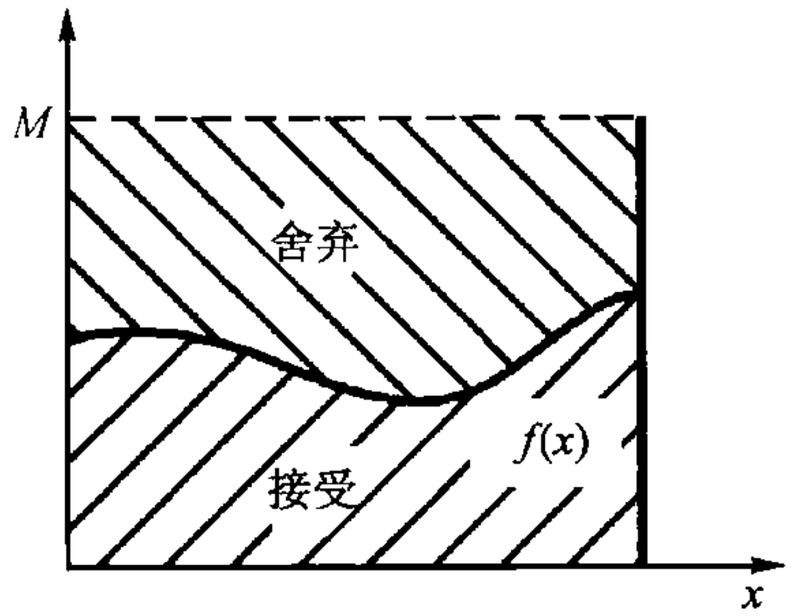


图 8.4 舍取法示意图

3. MATLAB 指令

对常见的 20 个特殊分布函数，MATLAB 有专门的计算指令，列出如下：

分布类型名	指令名	分布类型名	指令名
β 分布	betarnd	二项式分布	binornd
χ^2 分布	chi2rnd	指数分布	exprnd
F 分布	frnd	γ 分布	gamrnd
几何分布	geornd	超几何分布	hygernd
对数正态分布	lognrnd	负二项分布	nbinrnd
非中心 F 分布	ncfrnd	非中心t分布	nctrnd

续表

分布类型名	指令名	分布类型名	指令名
非中心 χ^2 分布	ncx2rnd	正态分布	normrnd
泊松分布	poissrnd	瑞利分布	raylrnd
学生 t 分布	trnd	离散均匀分布	unidrnd
连续均匀分布	unifrnd	威布尔分布	weibrnd

例 3 氢原子电子云的模拟。这是应用舍取法一个例子。

氢原子的基态 $(n=1,\ l=0,\ m=0)$ 的电子分布概率密度函数是 $D=\frac{4r^2}{a_1^3}\mathrm{e}^{-2r/a_1}$，$a_1=5.29\times10^{-2}$ nm，D 的最大值 $Dr_{\max}=1.1$，$r_0=0.25$ nm 是 D 的收敛点。

模拟是用点的密度来表示电子的概率分布密度。模拟时先产生一个随机的电子轨道半径 $r_1=r_0\mathrm{rand}(1)$，显然有 $0\leqslant r_1\leqslant r_0$，由 r_1 计算出 $Dr(r_1)$。再产生一个随机的概率判据 $r_2=Dr_{\max}\mathrm{rand}(1)$，显然有 $0\leqslant r_2\leqslant Dr_{\max}$，然后进行判断，如果 $r_2>Dr$，则舍弃它，反之就计算一个随机的角度值，$\theta=2\pi\ \mathrm{rand}(1)$，最后得到的点的坐标是 $x=r\cos\theta, y=r\sin\theta$。在程序中使用矢量化编程以提高计算速度。程序如下：

```
N=400000;    a=0.529;
r1=5*rand(1,N);
Dr=4/a^3*r1.^2.*exp(-2/a*r1);
M=max(Dr);
r2=M*rand(1,N);
r=r1(find(r2<Dr));
n=length(r);
Q=2*pi*rand(1,n);
[X,Y]=pol2cart(Q,r);
plot(X,Y,'r.','marker','.','markersize',1)
axis equal
figure
r=0:0.01:5;
Dr=4/a^3*r.^2.*exp(-2/a*r);
plot(r,Dr)
```

所得氢原子基态的电子概率密度函数如图 8.5 所示，电子概率密度图形如图 8.6 所示。

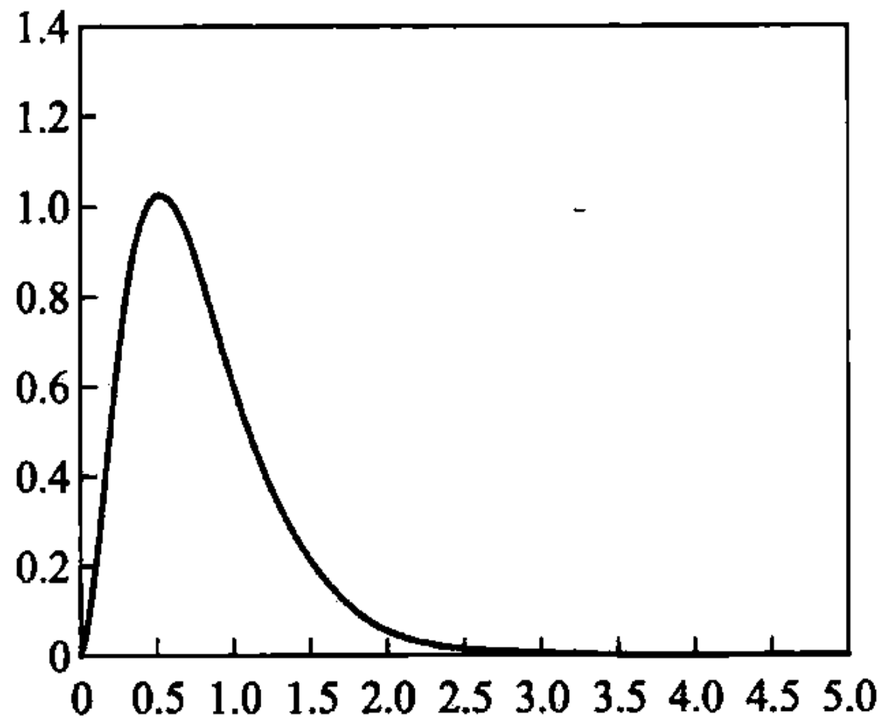


图 8.5 电子概率密度函数曲线

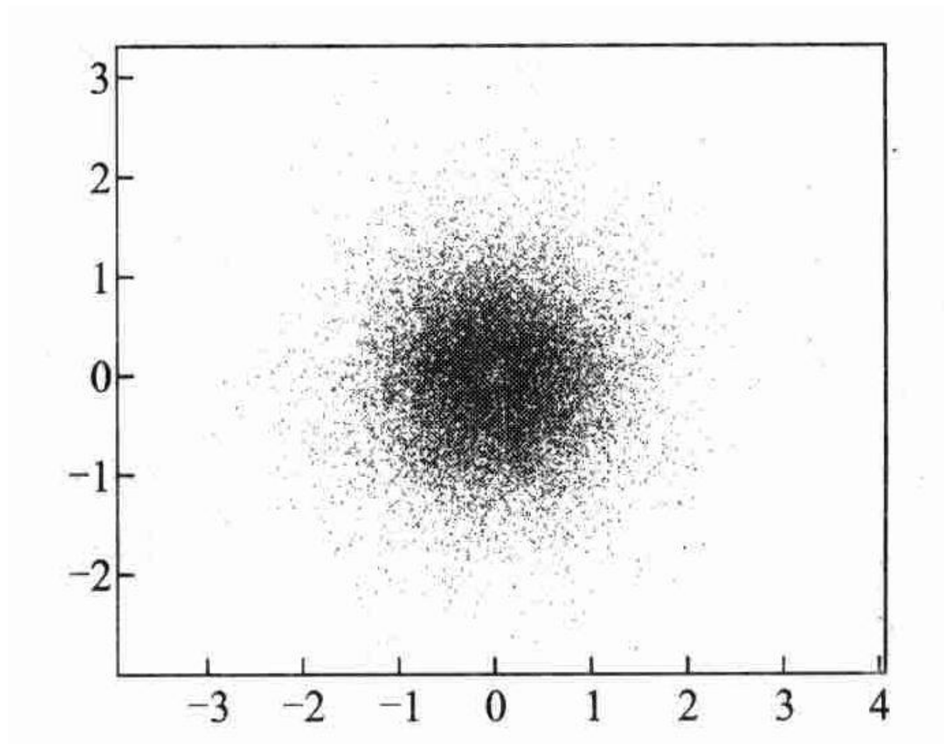


图 8.6 电子概率密度分布

§8.5 计算定积分

1. 投点法

本节开头计算 π 的方法也可以用来计算高维积分。因为它实际计算的就是圆周线下的面积，也就是计算一维定积分。下面详细说明这种方法。

假设要计算一维定积分

$$s = \int_a^b f(x)\mathrm{d}x$$

如图 8.7 所示，在积分区间内，函数的最大值为 h。在高为 h 宽为 $b-a$ 的矩形内均匀投点 M 个，落在曲线 $f(x)$ 下的点数为 N，则曲线下的面积与矩形的面积 S 比等于 N/M，即计算公式为

$$S = h(b-c), \quad s = S\frac{N}{M}$$

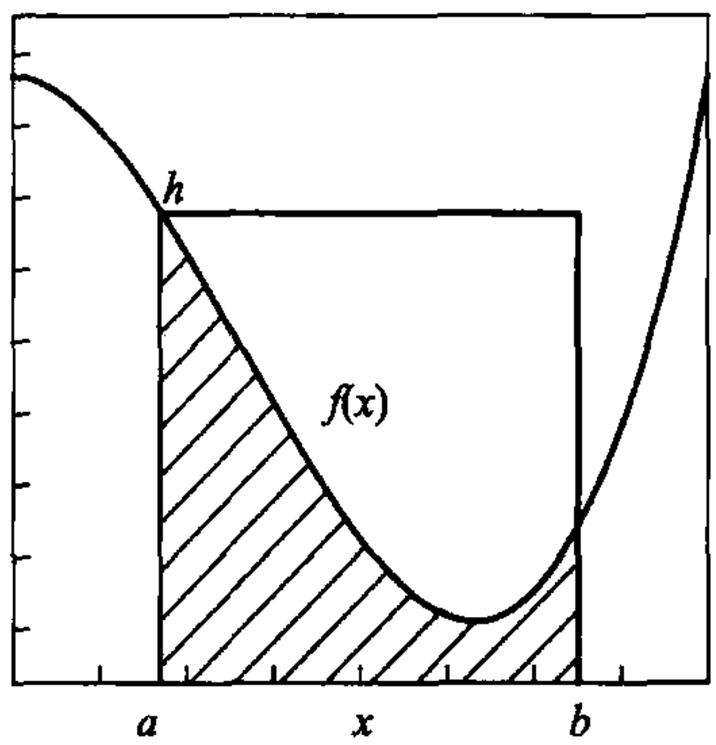


图 8.7 投点法计算定积分

用 MC 法计算时，首先要生成 M 个二维的等概率分布的随机数组 p_i，q_i，它们满足

$$a < p_i < b, \quad 0 < q_i < h$$

统计满足条件 $q_i < f(p_i)$ 的投点的数目，代入上面计算公式即得。

以此类推，计算二重积分就是计算空间面积 σ，需要的构造一个更大的空间面积 Σ，它能包围所计算的空间面积，而且便于计算面积。生成在 Σ 上面的三维投点 M 个，其中落在 σ 上的有 N 个，则有

$$\sigma = \Sigma \frac{N}{M}$$

如果要计算三重积分（空间体积 v），被积函数在积分区间的最大值为边构成一个立方体，其体积假设是 V，生成 V 内均匀分布的三维随机数组作为投点，体积内的投点数 N 与立方体内的总的投点数 M 之比就是所求的体积，即

$$v = V \frac{N}{M}$$

例 4 如图 8.8，半径为 0.5 m 球体上有一个半径为 0.3 m 的柱体空洞，计算球体剩余部分的体积。

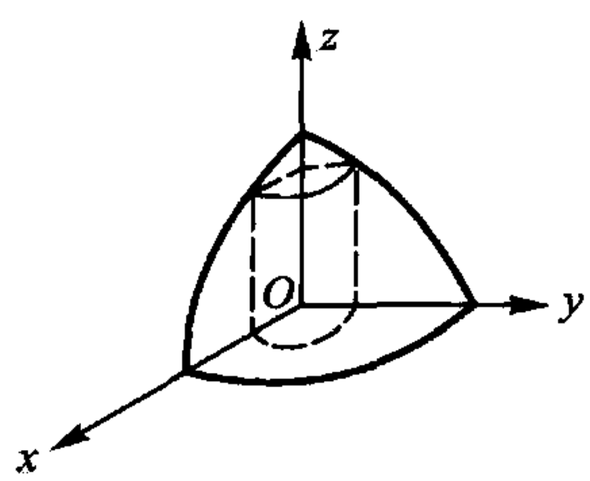


图 8.8 带柱体空洞的球体

作一个边长为 1 m 的正方体，令球心与正方体的中心重合，则球体被体积为 1 m^3 的正方体所包围。以球心为坐标原点，建立坐标轴。如图所示，根据对称性，只要计算第一象限的体积再乘以 8 就得到整个体积。首先生成 M 个三维均匀分布的随机数组 p_i，q_i，r_i，取值范围都是 $[0,0.5]$。判断投点是否在体积内的判据是

$$p_i^2+q_i^2+t_i^2<0.5^2,\quad p_i^2+q_i^2>0.3^2$$

求出落在体积内投点的数目 N，程序中用关系运算符来完成这个任务，满足条件时为真，即值为 1，不满足条件时为假，即值为 0。就可以计算在第一象限的体积为 $0.5^3N/M$，再乘以 8 得到全部体积为 N/M。

程序如下：

```
M=1000000;
A=0.5.*rand(3,M);
a1=sqrt(sum(A.*A))<0.5;                    %找出落在球内的点的位置
A(1,:)=A(1,:).*a1;                         %去掉代表球外的点
A(2,:)=A(2,:).*a1;
a2=sqrt(A(1,:).^2+A(2,:).^2)>0.3;          %找出落在柱体外的点
sum(a2)/N                                  %计算体积
```

2. 平均值法

以前计算定积分的公式形式为

$$\int_0^1 f(x)\mathrm{d}x\approx\sum_{i=1}^N w_i f(x_i)$$

例如对矩形积分公式有

$$w_i=1/N$$

如果在 [0，1] 之间等概率的取 N 个点，计算 $f(x)$ 在这些点的平均值。则公式的形式为

$$\int_0^1 f(x)\mathrm{d}x\approx\frac{1}{N}\sum_{i=1}^N f(x_i)$$

这相当于认为 x 是一个均匀分布的随机数，然后用样本平均来代替期望值。这个公式与矩形积分的差别在于，现在的 x_i 随机选取的，而不是特定的等间隔的点。这就是 MC 方法的积分公式，它包括两种基本运算，以某种概率随机地选取样点，再用平均值代替积分值。

如果把 $f_i\equiv f(x_i)$ 看成一个随机变量并且 N 是足够大的数，这个积分公式

的误差为

$$\sigma^2 \approx \frac{1}{N}\sigma_f^2 = \frac{1}{N}\left[\frac{1}{N}\sum_{i=1}^{N} f_i^2 - \left[\frac{1}{N}\sum_{i=1}^{N} f_i\right]^2\right]$$

其中 σ_f^2 是 f 的方差，即 f 在积分区间上偏离其平均值的程度的量度。

以此类推，二重积分和三重积分的 MC 方法的公式是

$$I = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i)$$

$$I = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i, z_i)$$

对于形式为 $I = \int \mathrm{d}\boldsymbol{x} f(\boldsymbol{x})$ 的 d 维积分（$\boldsymbol{x}$ 是一个 d 维矢量），积分公式就是

$$I \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i)$$

其中随机点的各个分量 $\boldsymbol{x}_i$ 都是独立的随机变量。

作为练习，下面的程序对积分

$$I = \int_0^1 \frac{\mathrm{d}x}{1+x^2} = \frac{\pi}{4} = 0.785\ 4$$

进行了计算，输入的不同的 N 值可得到相应的积分值和对积分精度的估计。

```
N=10000;
x=rand(N,1);
F=1./(1+x.^2);
PI=4*sum(F)/N
SIGMA=sqrt(sum(F.^2)-PI^2)/N                    %求均方差
```

例如在一次运算中所得值为 $\pi = 3.135\ 3$，sigma $= 0.008\ 0$，对不同的 N 值运行这个程序。算出的结果同精确值只差标准偏差的几倍(通常不到一倍)，并且随着 N 增大求积变得更精确。

3. 辅助函数法

在平均值法中，被积函数 f 在积分区间内变化越平缓，积分精度越高。最好的情况是 f 为常数，这时函数在任一点的值都是它的平均值。最差的情况是，f 除了在 x 的某个值附近有一个很窄的峰之外其余都是零，如果还是以等概率在区间 [0,1] 选取 x_i，那么除了少数几个之外绝大多数 x_i 都将位于 f 的尖峰之外，而只有这少数几个的 f_i 才不是零，这将导致对 I 的一个很差的估值。

为了提高精度，引入辅助函数 $w(x)$，这是正的规一化的函数 $w(x)$，

$$\int_0^1 w(x)\mathrm{d}x = 1$$

将积分写成

$$I = \int_0^1 \mathrm{d}x w(x) \frac{f(x)}{w(x)}$$

定义

$$y(x) = \int_0^x \mathrm{d}x' w(x') \tag{8.4}$$

因而

$$\frac{\mathrm{d}y}{\mathrm{d}x} = w(x), \quad y(x=0) = 0, \quad y(x=1) = 1$$

那么积分就变为

$$I = \int_0^1 \mathrm{d}y \frac{f(x(y))}{w(x(y))}$$

如果 w 的行为和 f 近似相同，就会使上式中被积函数 f/w 变化平缓，从而减小了 MC 估值的方差。这时 MC 方法的积分公式成为

$$I \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x(y_i))}{w(x(y_i))}$$

上面变量变换的意义在于，与点在 y 内均匀分布相对应的是点在 x 的分布(概率密度) 是 $\mathrm{d}y/\mathrm{d}x = w(x)$。而这又意味着点集中在最“重要”的 x 值附近（所以这个方法也叫重要抽样法），此处 w 大(因而 f 也有希望大)。对于那些 w 和 f 值小的“不重要的”x 值，只花费很少的精力去计算相应的被积函数值。

现在用变量替换重新计算上面例题中的积分 I。取权函数为

$$w(x) = \frac{1}{3}(4 - 2x)$$

这个函数是正的归一化的，在积分区间上是单调下降 (f 也是)。此外，由于在 $x = 0$ 和 $x = 1$ 两个端点上都有 $f/w = 3/4$，w 对 f 的行为近似得很好。根据 (8.4) 式，新的积分变量是

$$y = \frac{1}{3}x(4 - x)$$

从它可以反推出

$$x = 2 - (4 - 3y)^{1/2}$$

```
N=10000;
Y=rand(N,1);
X=2-sqrt(4-3*Y);
W=(4-2*X)/3;                                          %权函数
FF=1./(1+X.^2)./W;                                    %反函数
PI2=4*sum(FF)/N
SIGMA2=sqrt(sum(FF.^2)-PI2^2)/N                       %求均方差
```

运行这个程序一次，所得值为 $\pi = 3.140\ 9$，$\sigma = 0.007\ 8$。处理的结果有了改进。

§8.6 热力学的平衡态

如图 8.9 所示，密闭的盒子中间用隔板分为两半，隔板上有一小洞，开始时分子都在左边，经过足够长的时间，在达到平衡状态以后，两边的分子数将趋于相等。

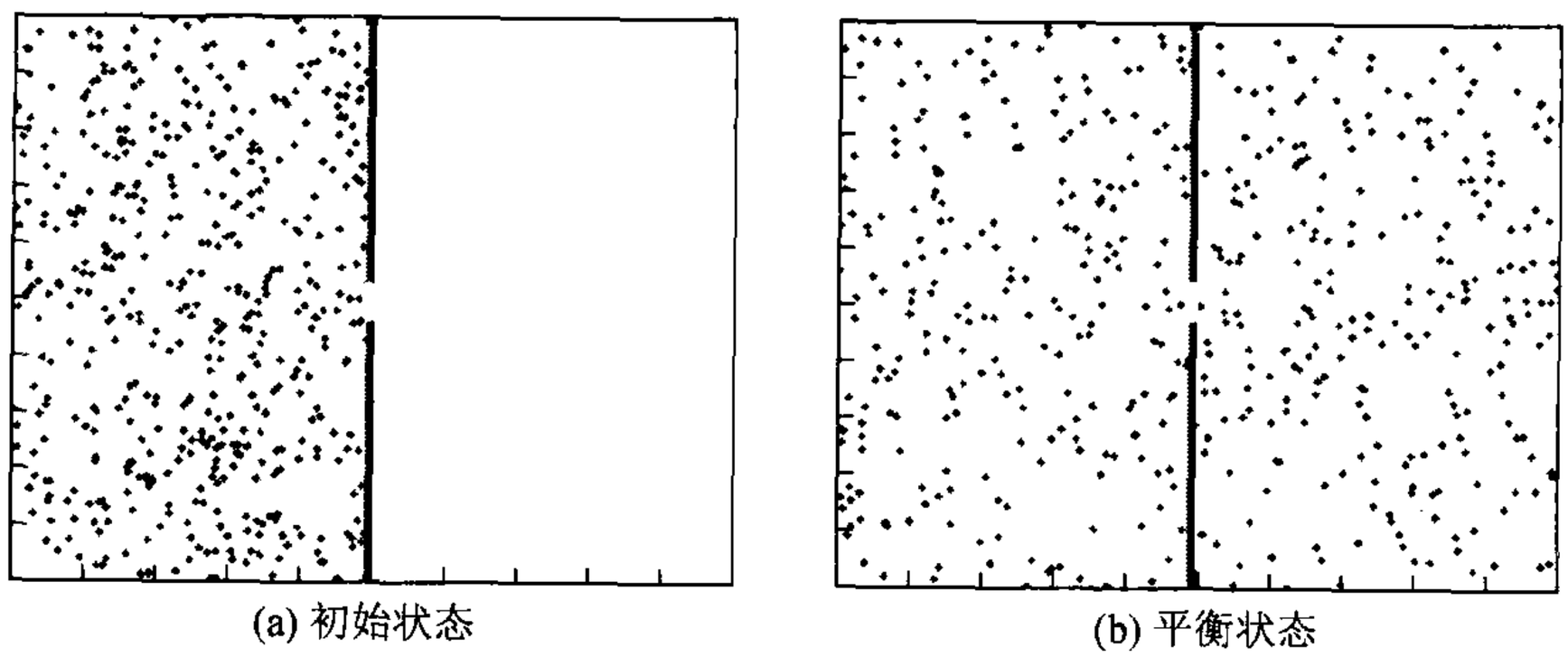


图 8.9 有隔板的盒子

下面模拟此过程。用一个数组表示全部分子的编号，在左边时，其值为 1，在右边时其值为 −1，每次在所有的分子中随机的取一个分子穿过隔板，也就是说，如果被选中的分子如果在左边，那它将穿过隔板进入右边，反之亦然。同时随时统计左边的分子数，经过足够长次数，将得到其分子数接近于总数的一半。程序如下，所得图形为图 8.10。

```
n=2000;                                               %总的分子数
B=ones(1,n);                                          %起初所有分子在左边，其值为1
for j=1:10000                                         %经过10000次的随机选择
k=ceil(rand(1)*n);                                    %每次是第n个分子穿过隔板
B(k)=-B(k);                                           %穿过隔板后改变符号
```

```
cc=length(find(B==1));                    %统计左边的分子数
c(j)=cc;
end
subplot(2,1,1)
plot(c)
subplot(2,1,2)
x=0:10000;
y=n*0.5*(1+exp(-2*x./n));                 %用解析式计算右边分子数
plot(x)
```

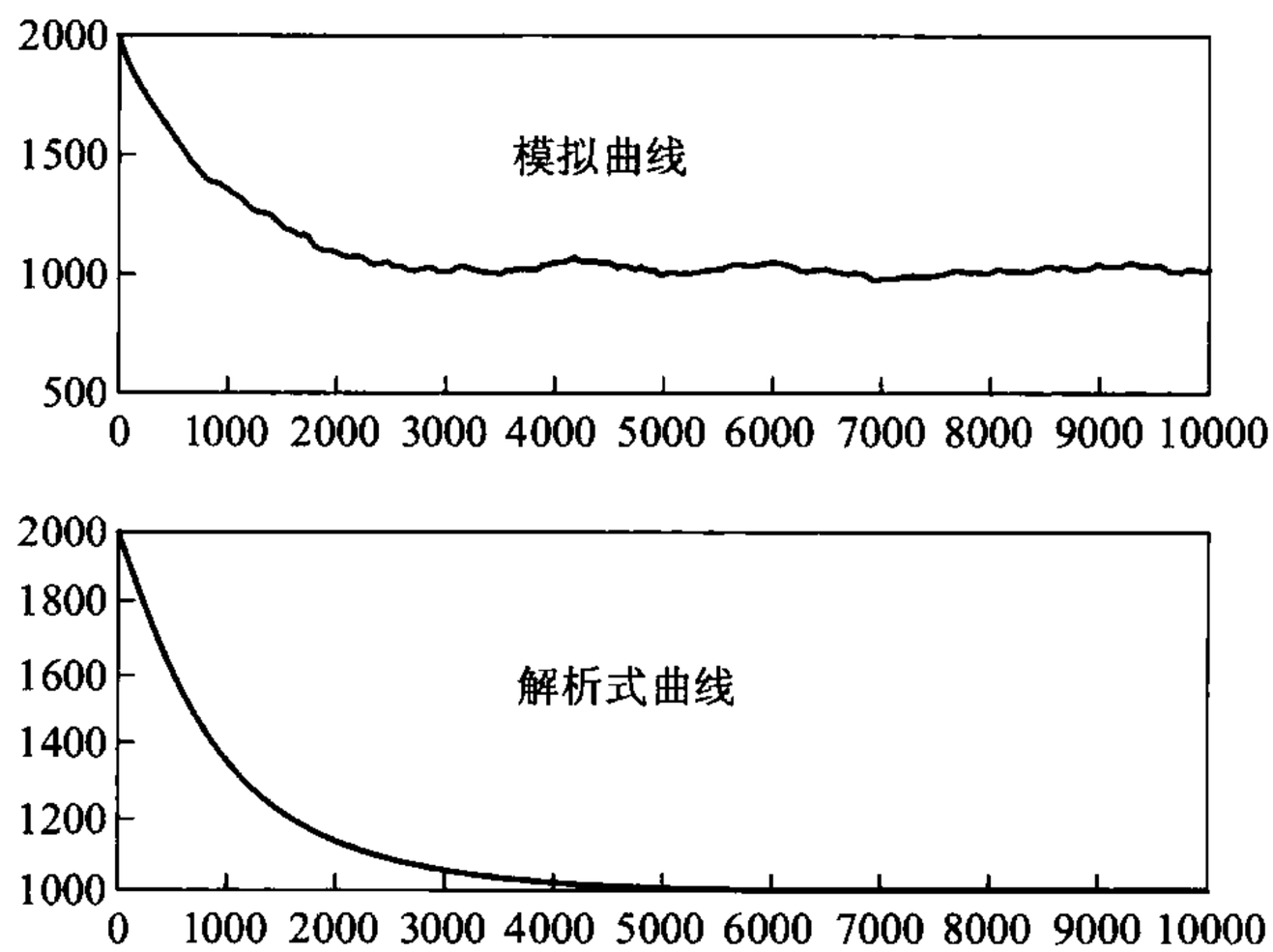


图 8.10 平衡状态下隔板两边的分子数

这个问题可以作解析分析。开始时，左边有 N 个分子右边没有分子，经过一段时间后，有 X 个分子穿过了隔板，这里包括从左边到右边的，也有从右边到左边的。这时，右边有了 n 个分子。下一次一个分子穿过隔板的概率对于右边的分子来说是 $p_{\mathrm{r}}=n/N$，对左边的分子来说，是 $p_{\mathrm{l}}=1-p_{\mathrm{r}}$。因此以后每当一个分子穿过隔板，右边的分子数的变化为

$$\Delta n=p_{\mathrm{l}}-p_{\mathrm{r}}=1-\frac{2n}{N} \tag{8.5}$$

每一次发生的状态变化记为 ΔX，则有

$$\Delta X=X(t_{i+1})-X(t_i)=1$$

于是 (8.5) 式形式上可写为

$$\Delta n=(1-\frac{2n}{N})\Delta X$$

把 n, X 看成连续变量，得

$$\frac{\mathrm{d}n}{1-2n/N}=\mathrm{d}X$$

积分后得

$$-\frac{N}{2}\ln(N-2n)=X+C$$

利用初始条件 $X=0$, $n=0$, 可得

$$C=-\frac{N\ln N}{2}$$

将 C 代入上式，整理得

$$p_{\mathrm{r}}=\frac{n}{N}=\frac{1}{2}(1-\mathrm{e}^{-2x/N})$$

$$p_{\mathrm{l}}=1-p_{\mathrm{r}}=\frac{1}{2}(1+\mathrm{e}^{-2x/N})$$

在上面程序中也画出了这个解析式 p_{r} 所表示右边的分子数。可以看到，图 8.10 中两个图形是一致的。

§8.7 麦克斯韦速率分布律

两个同样的理想气体分子在平面上运动，它们速度方向是在 $[0, 2\pi]$ 间随机的两个角度 α_1, α_2, 速率为 v_1, v_2。在碰撞前它们在实验室坐标系 Oxy 上的投影是

$$v_{1x}=v_1\cos\alpha_1,\qquad v_{1y}=v_1\sin\alpha_1$$

$$v_{2x}=v_2\cos\alpha_2,\qquad v_{2y}=v_2\sin\alpha_2$$

质心速度是 $\boldsymbol{v}=\frac{1}{2}(\boldsymbol{v}_1+\boldsymbol{v}_2)$。在质心坐标系有

$$\boldsymbol{w}_1=\boldsymbol{v}_1-\boldsymbol{v}=\frac{1}{2}(\boldsymbol{v}_1-\boldsymbol{v}_2)$$

$$\boldsymbol{w}_2=\boldsymbol{v}_2-\boldsymbol{v}=\frac{1}{2}(\boldsymbol{v}_2-\boldsymbol{v}_1)=-\boldsymbol{w}_1$$

由动量与能量守恒定律，在质心坐标系中，分子碰撞使 $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ 方向发生变化，而大小不变，如图 8.11 所示。

图中，带撇的符号表示碰撞后的量。在经典散射的意义上，取向角是各向同性的，即是 $[0, 2\pi]$ 上均匀分布的随机变量。又从图中知

$$w'_{1x}=w_{1x}\cos\gamma+w_{1y}\sin\gamma$$

$$w'_{1y} = -w_{1x}\sin\gamma + w_{1y}\cos\gamma$$

$$\boldsymbol{w}'_2 = -\boldsymbol{w}'_1$$

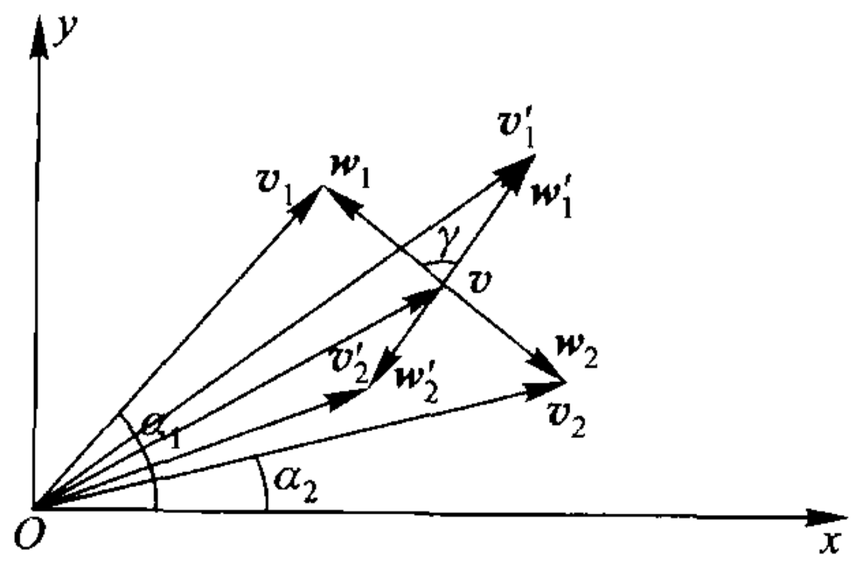


图 8.11 两个气体分子在平面上的碰撞运动

在实验室坐标系中有

$$\boldsymbol{v}'_1 = \boldsymbol{w}'_1 + \boldsymbol{v}, \qquad \boldsymbol{v}'_2 = \boldsymbol{w}'_2 + \boldsymbol{v}$$

故可以得到

$$(v'_1)^2 = \frac{1}{2}\left(v_1^2 + v_2^2\right) + \left[\frac{1}{2}\left(v_1^2 - v_2^2\right)\cos\gamma + v_1 v_2 \sin\alpha\sin\gamma\right]$$

$$(v'_2)^2 = \frac{1}{2}\left(v_1^2 + v_2^2\right) - \left[\frac{1}{2}\left(v_1^2 - v_2^2\right)\cos\gamma + v_1 v_2 \sin\alpha\sin\gamma\right]$$

其中 $\alpha = \alpha_1 - \alpha_2$ 为在 $[0, 2\pi]$ 间的均匀分布的随机变量。

程序的计算步骤为

(1) 随机选择两个分子；

(2) 随机选择碰撞前两个分子的速度夹角 α；

(3) 随机选择质心坐标系散射后的取向角 γ；

(4) 计算在实验室坐标系中分子碰撞后的速度；

(5) 统计在各速率间隔内的分子数和比率。

为了对比，可以用理论公式加以对比。

```
s=1000;                                          %粒子数1000
v=linspace(0,800,s);                             %速率分布
for j=1:10000                                    %碰撞次数
    k=ceil(s*rand(1,2));                         %任取两个粒子
    vk1=v(k(1));   vk2=v(k(2));                  %两个粒子的速率
    a=2*3.14159265*rand(1);
    r=2*3.14159265*rand(1);                      %选择角度
```

```
        vk1new = 1/2*vk1^2+1/2*vk2^2+1/2*cos(r)*vk1^2-...
                 1/2*cos(r)*vk2^2+vk1*vk2* sin(r)*sin(a); %计算速率
      vk2new = 1/2*vk1^2 +1/2*vk2^2 -1/2*cos(r)*vk1^2+1/2*...
                 cos(r)*vk2^2 - vk1*vk2* sin(r)*sin(a);
      v(k(1)) =   sqrt(vk1new);
      v(k(2)) =   sqrt(vk2new);
  end
  hist(v,20)                                                  %画直方图
  hold on
  NA=6.022e23;                                                %画理论曲线对比
  m=28e-3/NA;  T=200;  k=1.381e-23; m2kt=m/2/k/T; v2=v.^2;
  f=4*pi*(m2kt/pi)^1.5.*v2.*exp(-m2kt.*v2)*55000;
  plot(v,f,'*r')
```

所得图形如图 8.12。

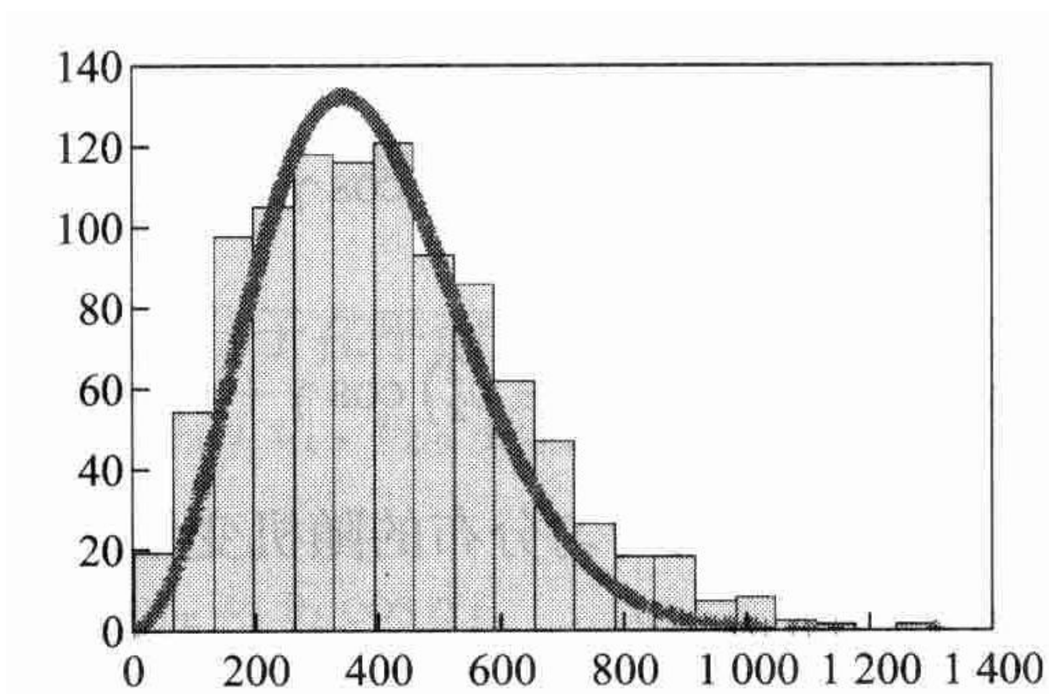


图 8.12 MC 方法模拟的理想气体分子的麦克斯韦速率分布曲线

§8.8 链式反应的模拟

本节用 MC 方法模拟铀核的链式反应。

一个 U^{235} 吸收一个中子产生裂变后会放出两个中子，这两个中子如果被两个 U^{235} 吸收并发生裂变，就会放出四个中子，如果这个过程能够持续下去，经过 30 次核反应，发生裂变的核可达到 $2^{30} \approx 10^9$ 个，即达到约 10 亿个。这时会释放出大量的能量甚至爆炸，这就是链式反应。

使这个过程能够维持的条件与很多因素有关，其中的一个条件就是铀块的体积要达到一定的大小，才能保证中子在飞出铀核之前有机会被另一个 U^{235} 吸收。具体说就是：假定开初有自发裂变产生的 N_0 个中子，这些中子经过一次裂

变反应后，最终又产生了 N 个中子，定义增殖系数

$$k = \frac{N}{N_0}$$

$k > 1$ 是维持链式反应的必要条件，当 $k < 1$ 时，链式反应不能维持，称之为熄火过程，$k = 1$ 时铀块的体积就是临界体积 V，

对应的临界质量为

$$M = V\rho$$

ρ 是质量密度，计算时可以取为 1。

作为一个简化模型，取铀块形状为 $a \times a \times b$ 的长方体，如图 8.13 所示，裂变最初从长方体内任意一点 (x_0, y_0, z_0) 开始，即

$$0 \leqslant x_0 \leqslant 1, \quad 0 \leqslant y_0 \leqslant 1, \quad 0 \leqslant z_0 \leqslant 1,$$

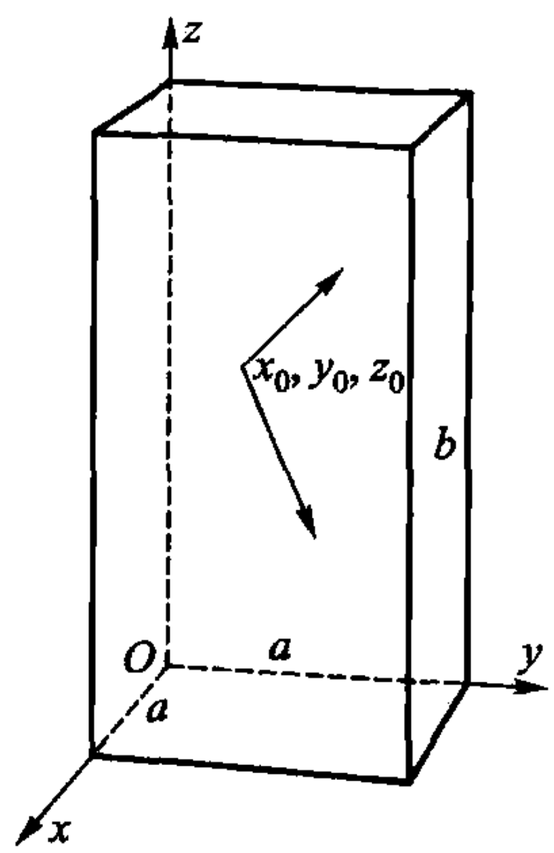


图 8.13 链式反应的模拟

裂变放出的两个中子的飞行方向是各向同性的，即按立体角均匀分布。立体角的表示为

$$\sin\theta \mathrm{d}\theta \mathrm{d}\phi = \mathrm{d}\phi \mathrm{d}\cos\theta$$

在立体角均匀分布就是要求 ϕ 在 $0 \sim 2\pi$ 之间是均匀的，$\cos\theta$ 在 $-1 \sim +1$ 之间均匀分布，因为 $0 \leqslant \theta \leqslant \pi$。注意，这里不能是 θ 角均匀分布而是 $\cos\theta$ 均匀分布。如果按 θ 均匀分布，则从立体角的分布看，在 $\theta = \pi/2$ 附近有较大的分布。

放出的两个中子能否与核碰撞并被吸收取决于它在铀块内经过的距离。描述碰撞（吸收）用平均自由程的概念，它表示裂变中子能与核碰撞时所行走的平均路程。假定在平均自由程内中子与核碰撞的概率相等，例如平均自由程为

1 cm，中子飞行的距离是 0.3 cm，则碰撞发生的概率为 30%。一个中子在铀块内飞行的距离可以用 0 ~ 1 之间的随机数 d 来表示，那么中子最终的位置是

$$
\begin{aligned}
x &= x_0 + d\sin\theta\cos\phi \\
y &= y_0 + d\sin\theta\sin\phi \\
z &= z_0 + d\cos\theta
\end{aligned}
$$

如果 x_1，y_1，z_1 还在铀块内，意味着这个中子会引起一次新的裂变；反之，则表明这个中子飞出了铀块，没有引起新的裂变。假定自发裂变产生了 N_0 个中子，计算出全部 N_0 个随机点的裂变情况，求出总碰撞数 N，就可以得到增殖系数 $k = N/N_0$。

程序计算的步骤如下：

(1) 输入最初的随机点数 N_0，铀块的质量 M，铀块的边长比 $s = a/b$。这里有

$$
M = V = a^2 b = \frac{a^3}{s} = \frac{b^3}{s^2}
$$

$$
a = (Ms)^{1/3}, \quad b = (Ms^{-2})^{1/3}
$$

(2) 产生 9 个随机数 r_1，r_2，$\cdots$，r_9。

(3) 最初的核裂变的位置 $x_0 = ar_1$，$y_0 = ar_2$，$z_0 = br_3$。

(4) 裂变放出的两个中子的飞行方向与距离。

$$
\phi_1 = 2\pi r_4, \quad \cos\theta_1 = 2r_5 - 1, \quad d_1 = r_6
$$

$$
\phi_2 = 2\pi r_7, \quad \cos\theta_2 = 2r_8 - 1, \quad d_2 = r_9
$$

(5) 两个中子最终到达的位置是

$$
\begin{aligned}
x_1 &= x_0 + d_1\sin\theta_1\cos\phi_1 \\
y_1 &= y_0 + d_1\sin\theta_1\sin\phi_1 \\
z_1 &= z_0 + d_1\cos\theta_1 \\
x_2 &= x_0 + d_2\sin\theta_2\cos\phi_2 \\
y_2 &= y_0 + d_2\sin\theta_2\sin\phi_2 \\
z_2 &= z_0 + d_2\cos\theta_2
\end{aligned}
$$

(6) 检查发生新的核裂变的可能性，并计算新的中子数 N。

如果 $0 \leqslant (x_1, y_1, z_1) \leqslant 1$，则 N 增加 1；

如果 $0 \leqslant (x_2, y_2, z_2) \leqslant 1$，则 N 也增加 1；

(7) 对 N_0 个最初的中子都作上述计算，再计算 $k = N/N_0$，若 $k = 1$，此时的 M 就是临界质量，若 $k \neq 1$，则调整 M，s 的值重新计算。

参考程序如下，程序中的 M，S，N_0 值是从键盘输入的。

```
M=input('M=')
S=input('S=')
N0=input('N0=')
R=rand(9,N0);
N=0;
a=(M*S)^(1/3);    b=(M/S^2)^(1/3);
for k=1:N0
    x0=a*R(1,k) ;   y0=a*R(2,k);     z0=b*R(3,k);
    phi1=2*pi*R(4,k);       cthi1=2*R(5,k)-1;       d1=R(6,k);
    phi2=2*pi*R(7,k);       cthi2=2*R(8,k)-1;       d2=R(9,k);
    sthi1=sqrt(1-cthi1^2);        sthi2=sqrt(1-cthi2^2);
    x1=x0+ d1*sthi1*cos(phi1);
    y1=y0+ d1*sthi1*sin(phi1);
    z1=z0+ d1*cthi1;
    x2=x0+ d2*sthi2*cos(phi2);
    y2=y0+ d2*sthi2*sin(phi2);
    z2=z0+ d2*cthi2;
    if  ((0<= x1)&(x1<=1)&(0<= y1)&(y1<=1)&(0<= z1)&(z1<=1))|...
            ((0<= x2)&(x2<=1)&(0<= y2)&(y2<=1)&(0<= z2)&(z2<=1))
        N=N+1 ;
     end
end
    f=N/N0
```

§8.9 迭代函数系统 (IFS)

随机数在分形学上也有很多用处。本节与下节介绍一些这方面的内容。

取平面上不共线的三个点 A, B, C 和三个概率数如 $P_A = 1/3$, $P_B = 1/3$, $P_C = 1/3$。然后在以 A, B, C 三点作为三角形的三个顶点，在三角形内部任取一点 Z 按照以下式子进行迭代：

$$Z_{n+1} = \begin{cases} \dfrac{Z_n + A}{2}, & \text{以概率} P_A \\ \dfrac{Z_n + B}{2}, & \text{以概率} P_B \\ \dfrac{Z_n + C}{2}, & \text{以概率} P_C \end{cases}$$

按照这个要求，编出如下程序：

```
N=10000;
ABC=[-1,    sqrt(2)*i,   1];                    %三个初始点坐标
Z=i;
v=rand(N,1);                                    %生成0到1之间等概率分布的数
for kk=2:N
      if   v(kk)<1/3
           Z(kk)=0.5*Z(kk-1)+0.5*ABC(1);
      elseif   v(kk)<2/3
           Z(kk)=0.5*Z(kk-1)+0.5*ABC(2);
      else
           Z(kk)=0.5*Z(kk-1)+0.5*ABC(3);
      end
end
plot(Z,  '.'  ,'MarkerSize',4)
```

当迭代次数分别为 500，1000，5000 次时，得到的图形如图 8.14 所示。

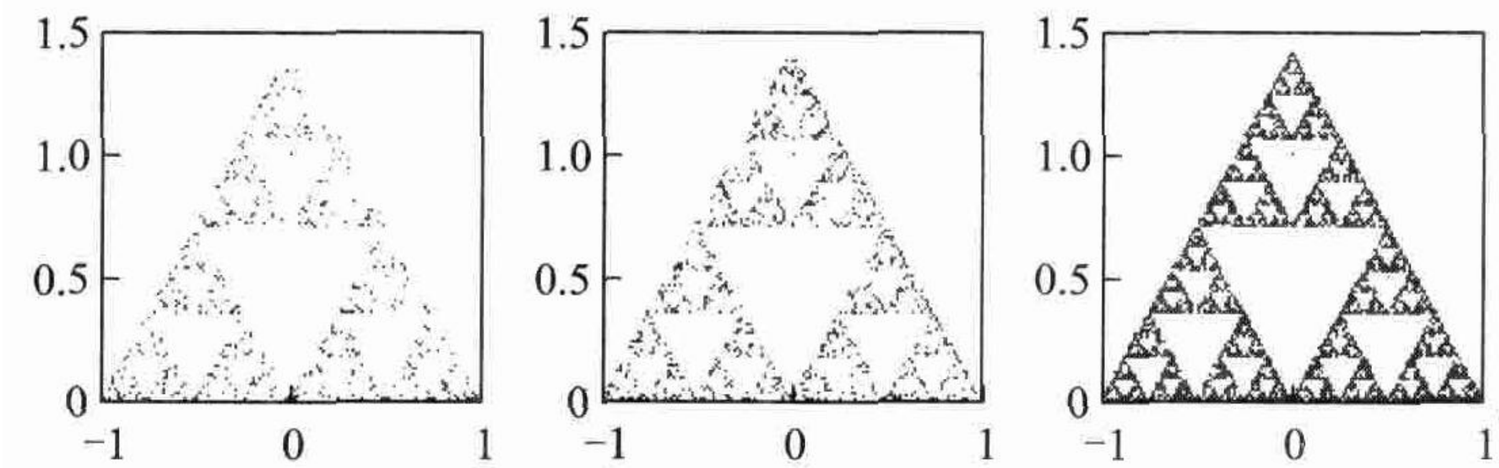


图 8.14 用 IFS 画出的 Siepinski 三角形

可以看到，随着迭代次数的增加，会显现出一越来越清楚的图像，它正是前面所画分形三角形图形。这种作图方法就是所谓的迭代函数系统 (IFS)。

迭代函数系统是美国佐治亚理工学院的 Barnsley 教授首创的。与L系统一样，它也是模拟生物形态最成功的系统之一。它的理论与方法已经成为分形自然景观模拟及分形图像压缩的理论基础。

迭代函数系统的算法是确定性算法和随机性算法相结合的方法。“确定性”是指用以迭代的规则是确定性的，它们由一组变换规则 (如 R_1，R_2，R_3，$\cdots$) 构成；“随机性”指迭代过程是不确定的，每一次迭代究竟用哪一个规则，即选 R_i 中的哪一个，不是预先定好的，而是随机决定的。也就是说，随机的从 R_i 中选一个迭代规则迭代一次，再从 R_i 中选一个再迭代一次，直到最终生成需要的图形。

由此可见，开头的例子就是迭代函数系统作图的一个例子，三条迭代规则都是：用三角形内部的一点 Z_n 与三角形的某个顶点之间的中点作为 Z_{n+1} 画出新点，而三个概率数都是 1/3。

对于平面上的点，一条迭代规则将一个旧点替换为一个新点，可用下式表示

$$\begin{cases} x' = ax + by + e \\ y' = cx + dy + f \end{cases}$$

也可以用矩阵形式表示

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

迭代规则的多少取决于问题的复杂程度，少则 2 条，多的可以有 16 条。下面再介绍两个例子。

例 5　羊齿叶图案

这是一种植物的叶子，称之为 Barnsley's Fern。共有 4 条迭代规则如下，其中的百分比代表概率。

$$\begin{pmatrix} x \\ y \end{pmatrix}_{n+1} = \begin{cases} \begin{pmatrix} 0.5 \\ 0.27y_n \end{pmatrix} & 2\% \\ \begin{pmatrix} -0.139x_n + 0.263y_n + 0.57 \\ 0.246x_n + 0.224y_n - 0.036 \end{pmatrix} & 15\% \\ \begin{pmatrix} 0.17x_n - 0.215y_n + 0.408 \\ 0.222x_n + 0.176y_n + 0.089\ 3 \end{pmatrix} & 13\% \\ \begin{pmatrix} 0.781x_n + 0.034y_n + 0.107\ 5 \\ -0.032x_n + 0.739y_n + 0.27 \end{pmatrix} & 70\% \end{cases}$$

如果用 r 表示 0 到 1 之间的等概率随机数，则上式可以表示成

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{cases} \begin{pmatrix} 0.5 \\ 0.27y_n \end{pmatrix} & r < 0.02 \\ \begin{pmatrix} -0.139x_n + 0.263y_n + 0.57 \\ 0.246x_n + 0.224y_n - 0.036 \end{pmatrix} & 0.02 \leqslant r \leqslant 0.17 \\ \begin{pmatrix} 0.17x_n - 0.215y_n + 0.408 \\ 0.222x_n + 0.176y_n + 0.089\ 3 \end{pmatrix} & 0.17 < r \leqslant 0.3 \\ \begin{pmatrix} 0.781x_n + 0.034y_n + 0.107\ 5 \\ -0.032x_n + 0.739y_n + 0.27 \end{pmatrix} & 0.3 < r < 1 \end{cases}$$

图形为图 8.15，程序如下：

图 8.15　羊齿叶图案

```
N=10000;
v=rand(N,1);
x=0;        y=0;
for kk=2:N
    vv=v(kk);
    if vv<=.02
        x(kk)=0.5;
        y(kk)=0.27*y(kk-1);
    elseif vv<=.17
        x(kk)=-.139*x(kk-1)+0.263*y(kk-1)+0.57;
        y(kk)=.246*x(kk-1)+0.224*y(kk-1)-0.036;
    elseif vv<=.3
        x(kk)=.17*x(kk-1)-.215*y(kk-1)+.408;
        y(kk)=.222*x(kk-1)+.176*y(kk-1)+.0893;
    else
        x(kk)=.781*x(kk-1)+.034*y(kk-1)+.1075;
        y(kk)=-.032*x(kk-1)+.739*y(kk-1)+.27;
    end
end
plot(x(1:N),y(1:N),'.','markersize',4)
```

这个程序若改写成矩阵形式，运算速度会有提高：

```
%羊齿叶草
N=10000;
v=rand(N,1);
```

```
a=[0.5;   0];      b=[0.57;   -0.036];
c=[0.408;   0.0893];   d=[0.1075;  0.27];                    %4个固定点
A=[0,  0;   0,  0.27];                                       %4个变换矩阵
B=[-0.139,   0.263;   0.246,    0.224];
C=[0.17,   -0.215;    0.222,    0.176];
D=[0.781,   0.034;   -0.032,   0.739];
u=[0;  0];                                                   %出发点
for k=2:N
    vv=v(k);
    if vv<=.02                                               %法则一
        u(:, k)=A*u(:, k-1)+ a;
    elseif vv<=.17                                           %法则二
        u(:, k)=B*u(:, k-1)+ b;
    elseif vv<=.3                                            %法则三
        u(:, k)=C*u(:, k-1)+ c;
    else                                                     %法则四
        u(:, k)=D*u(:, k-1)+ d;
    end
end
plot(u(1,:  ),   u(2, : ) ,'.','markersize',4)
```

例 6　概率树

这是一幅类似铅笔画一样的树形图。迭代规则有6条，其中的百分比表示概率。

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{cases} \begin{pmatrix} 0.05x_n \\ 0.6y_n \end{pmatrix} & 10\% \\ \begin{pmatrix} 0.05x_n \\ -0.5y_n + 1.0 \end{pmatrix} & 10\% \\ \begin{pmatrix} 0.46x_n - 0.15y_n \\ 0.39x_n + 0.38y_n + 0.6 \end{pmatrix} & 20\% \\ \begin{pmatrix} 0.47x_n - 0.15y_n \\ 0.17x_n + 0.42y_n + 1.1 \end{pmatrix} & 20\% \\ \begin{pmatrix} 0.43x_n + 0.28y_n \\ -0.25x_n + 0.45y_n + 1.0 \end{pmatrix} & 20\% \\ \begin{pmatrix} 0.42x_n + 0.26y_n \\ -0.35x_n + 0.31y_n + 0.37 \end{pmatrix} & 20\% \end{cases}$$

和前面一样，程序非常简单，列在下面供参考。所得图形为图 8.16。

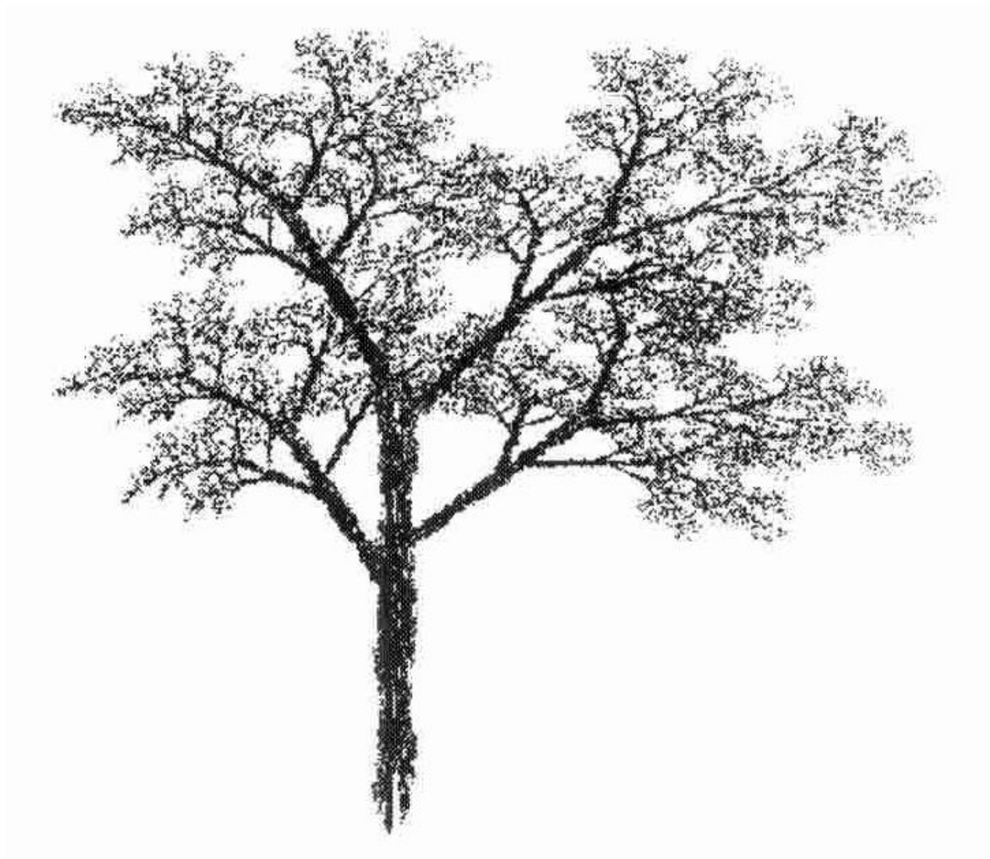


图 8.16 概率树

```
N=50000;
v=rand(N,1);
x= 0.5;
y= 0;
for kk=2:N
    vv=v(kk);
    if vv<0.1,
        x(kk)=0.05*x(kk-1);
        y(kk)=0.6*y(kk-1);
    elseif vv<0.2,
        x(kk)=0.05*x(kk-1);
        y(kk)=-0.5*y(kk-1)+1;
    elseif vv<0.4,
        x(kk)=0.46*x(kk-1)-0.15*y(kk-1);
        y(kk)=0.39*x(kk-1)+0.38*y(kk-1)+0.6;

    elseif vv<0.6,
        x(kk)=0.47*x(kk-1)-0.15*y(kk-1)  ;
        y(kk)=0.17*x(kk-1)+0.42*y(kk-1)+1.1;
    elseif vv<0.8,
        x(kk)=0.43*x(kk-1)+0.28*y(kk-1)  ;
        y(kk)=-0.25*x(kk-1)+0.45*y(kk-1)+1;
    else,
```

```
    x(kk)=0.42*x(kk-1)+0.26*y(kk-1);
    y(kk)=-0.35*x(kk-1)+0.31*y(kk-1)+0.7;
  end
end
plot(x(1:N),y(1:N),'.','MarkerSize',4)
```

§8.10 分形生长模型

本节介绍分形生长模型都是用计算机模拟固体表面的薄膜生长，程序虽然简单，但问题很有特点。

§8.10.1 抛射沉积模型

粒子经过沉积形成薄膜是一种普遍的现象。由于粒子是受热后从热灯丝蒸发出来，这个过程存在着随机性。但是由沉积形成的薄膜通常却会呈现很规整的结构。下面用计算机来模拟薄膜的生长过程。

模拟是一维的，粒子下落后聚集在长度为 L 的线上。为了简单，假定这是 200 个粒子集合而成的线。所有的粒子都来源于相同的高度，为了模拟它们有不同的速度，假定它们离线左边的距离是随机的。

模拟是由 0 到 L 的均匀的随机集合组成，粒子降落后在降落点聚集成一列。实际上，各列高度可能不同，它阻止粒子形成一条均匀的横线，粒子也可能四处跳动，直到落在一个洞里。为此假定，如果粒子所落在的列的高度比它旁边的两列都高的话，它就会增加这列的高度。如果粒子落在一个洞里，或者旁边存在一个洞，那么它就会填在这个洞里。为了提高计算模拟的速度，设定这个洞会填到与相邻的列的最大高度一样。

模拟的步骤如下

(1) 选择随机变量 r；

(2) 用向量 h_r 代表 r 处的列的高度；

(3) 按下式选择粒子将落下的位置

$$h_r = \begin{cases} h_r + 1 & (h_r \geqslant h_{r-1}, h_r > h_{r+1}) \\ \max(h_{r-1}, h_{r+1}) & (h_r < h_{r-1}, h_r < h_{r+1}) \end{cases}$$

程序如下：

```
%薄膜生长
h=zeros(1,200);                          %各列的初始高度
q=plot(h,' b.')                          %画出初始高度,准备动画
```

```
 set(q,'erasemode','none')
 axis([1 200 0 250]);
 hold on;
 Q=round(rand(1,20000)*197)+2;                          %建立随机数
for m=1:20000
     n=Q(m);
     if h(n)>=h(n-1)&h(n)>=h(n+1)                       %判断粒子落点
        h(n)=h(n)+1;
     else
        h(n)=max(h(n-1),h(n+1));
     end
     set(q,'Xdata',n,'Ydata',h(n))
     drawnow;
end
```

程序用动画模拟薄膜生长，下面的图 8.17 是一次模拟的最后图形，可以看到，在整条线上散布着几个空的区域，这正是薄膜生长过程的统计特征。这个模拟表现出膜的平均高度是随时间线性增加而且表面存在分形现象。

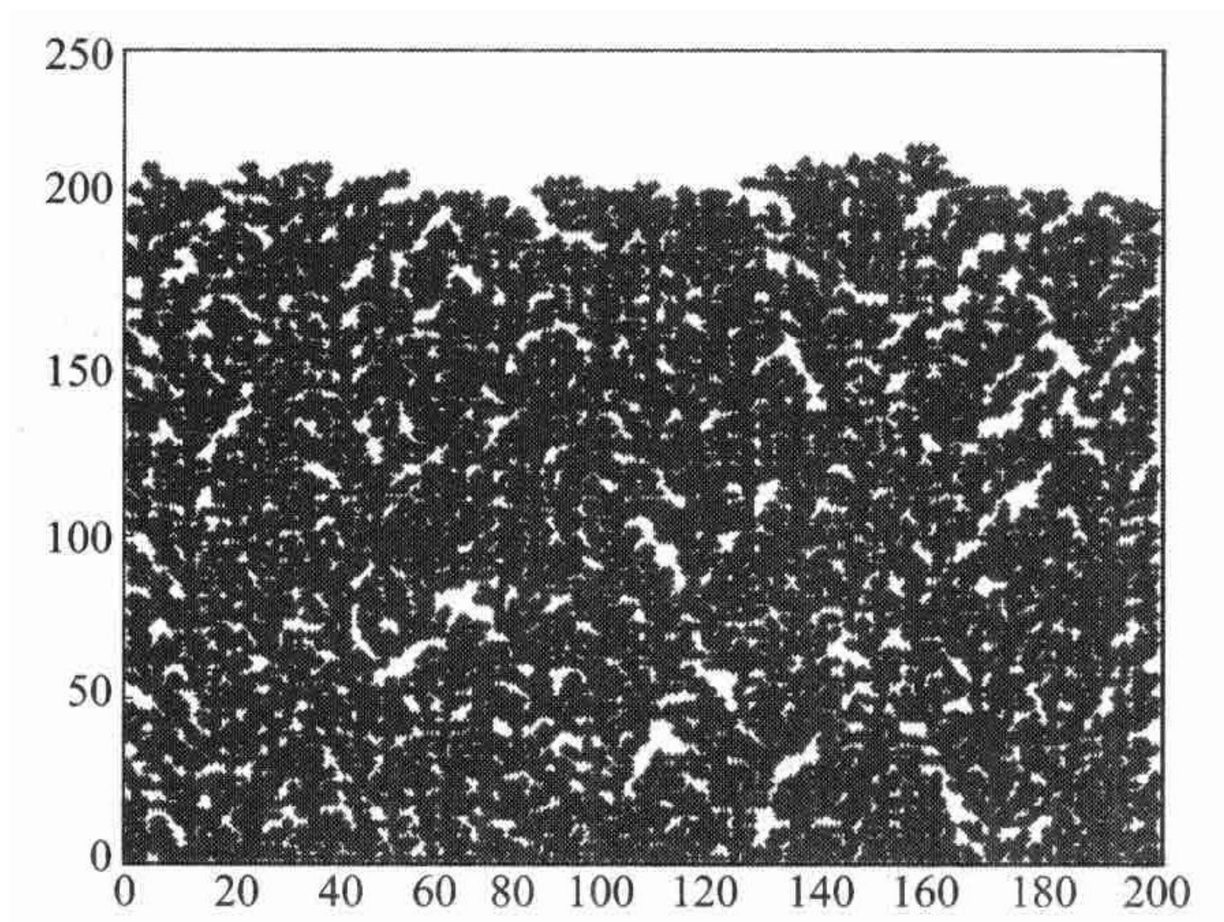


图 8.17 薄膜生长的模拟

§8.10.2 森林和薄膜的关联生长

我们都有这种经验，在自然界，如果一棵植物的旁边有另一棵植物，则它成活的机会就会增加。这种现象在薄膜生成的过程中也存在。这就是森林和薄膜关联生长。我们把这个因素加入到上述程序中去，也就是认为新沉积下的粒子会吸引另一个粒子，如图 8.18 所示。

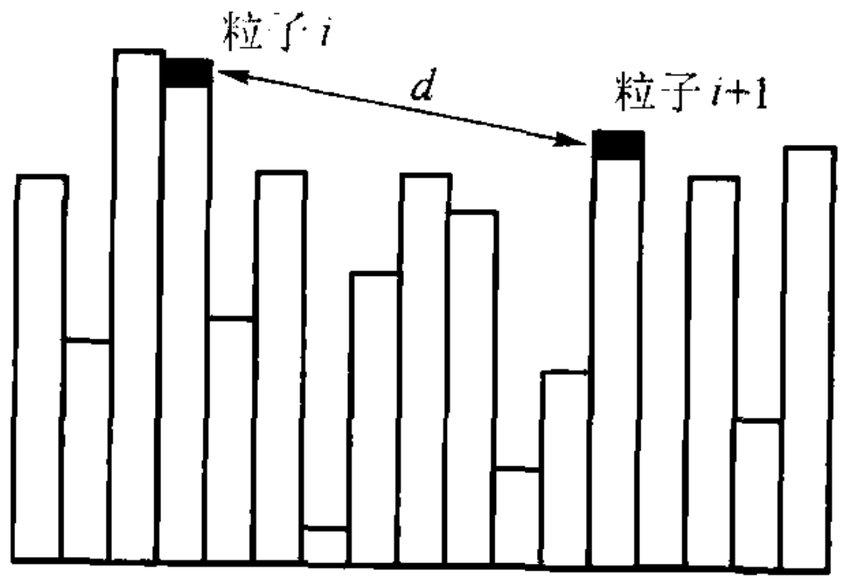


图 8.18 森林和薄膜的关联生长

我们假定沉积的概率 p 依赖于新增加的粒子到刚才落下的粒子之间的距离 d:

$$p = cd^{\eta}$$

这里的 η 是参数而 c 是决定概率大小的常数。我们取 $\eta = -2$，这意味粒子之间的吸引是反平方的（距离越远吸引越弱）。

如上例一样，用在 $[0, L]$ 中等概率的数来决定粒子会落在那一列，这个规则继续有效。但现在还要根据上式使用了第二个随机数来决定粒子是否会落下。也就是，如果算出的概率是 0.6 而 $r < 0.6$，则粒子被接受（落下），如果 $r > 0.6$，粒子将被拒绝。

程序如下：

```
%树林关联生长
h=zeros(1,200);
m=101;                                                    %m为启动值
Q=round(rand(1,100000)*197)+2;
q=plot(h,' b.')
set(q,'erasemode','none')
axis([0 200 0 120]);
hold on;
for k=1:100000
    n=Q(k);
    d=(m-n)^2+(h(m)-h(n))^2;                              %粒子间距
    if d>0,     p=14/d;                                   %粒子相互吸引的概率
      if rand(1) < p                                      %判断粒子是否落下
            if h(n)>=h(n-1)&h(n)>=h(n+1)
             h(n)=h(n)+1;
            else
```

```
            h(n)=max(h(n-1),h(n+1));
            end
        set(q,'Xdata',n,'Ydata',h(n))
        drawnow;
        m=n;
      end
    end
end
```

模拟结果以动画表现，图 8.19 是一次模拟的图形。

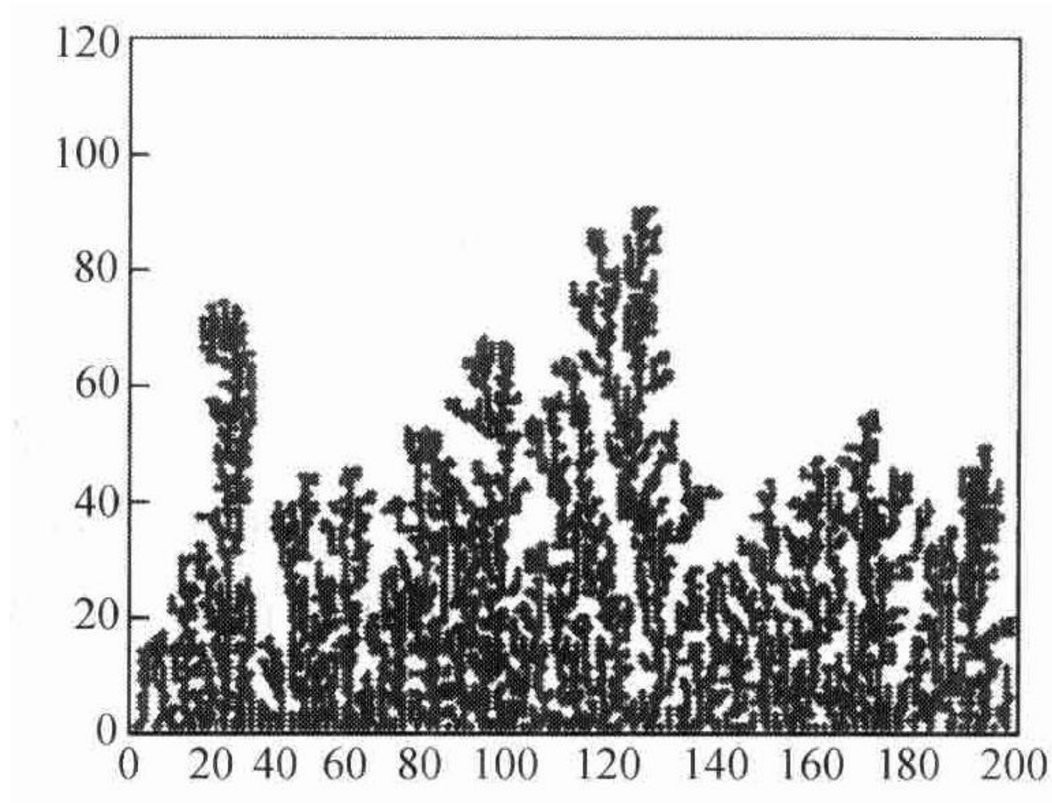


图 8.19 模拟的森林关联生长

§8.10.3 DLA

扩散限制聚集模型（DLA）成功地解释了团簇的周长与质量的关系。我们从 2D 的格子出发，中心包含一个种子粒子。围绕种子粒子画个圆，以相同的角概率在圆周上放上另一个粒子。然后释放这个粒子并让它随机行走，但只能在垂直或水平方向行走一个格子。这是一种模拟扩散过程的布朗运动。为使模拟更真实，我们让每一步的步长随高斯随机分布。一旦它行走到某一点时，发现在一个格子的距离之内存在另一个粒子，那么它们会粘在一起，中止行走。如果粒子走出了它被释放的圈子之外，它就永远地消失了。这个过程根据需要可以一直重复下去。一个典型的团簇生长如图 8.20 所示。

编写程序的步骤如下：

(1) 定义一个 400×400 的格点组成的网格 grid(400, 400)，各格点的初始值取为零。在网格中央放置一个种子，即让 grid(200, 200)=1。

(2) 想象一个半径为 180 格，圆心在 (200, 200) 处的圆，这就是我们用以释放粒子的圆。产生一个在 0 到 2π 均匀的角分布概率数，用以确定新粒子在圆周

上的位置。将这个位置换成直角坐标 x, y 表示。

(3) 产生一个等概率数 $0 < r_{xy} < 1$，如 $r_{xy} < 0.5$ 则粒子水平移动，如 $r_{xy} > 0.5$ 则粒子垂直移动。

(4) 用指令 randn 产生一个在 $(-\infty, +\infty)$ 的高斯随机数分布，这是步长分布的范围，同时它的符号指示方向的正负。

(5) 现在我们能确定粒子总的距离和方向。每次粒子将跳动一格，直到全部距离被覆盖。每次跳跃之前，核查一下最近的相邻的格点是否被占据。如果被占，则粒子停在现在的位置，而且行走结束；如果没有被占，粒子将跳动一格;继续这个过程直到整个距离被覆盖，或粒子被粘住，或者粒子离开了这个圆。

一旦随机行走结束，再释放另一个粒子，而新的过程又开始了。

由于许多粒子会丢失，所以需要成千上万个粒子才能形成一个只有几百个粒子的团簇。这个模拟需要一些时间。

程序如下：

```
figure                                          %扩散限制聚集(DLA)模型
u=zeros(400,400);                               %建立方格
u(200,200)=1;                                   %中心放置种子
gs=1e9*randn(1,40000);                          %高斯随机分布
theta=2*pi*rand(1,40000);                       %建立均匀的角概率数
rad=180*ones(1,40000);                          %半径
[X1,Y1]=pol2cart(theta,rad);                    %换成直角坐标
XX=round(X1)+200;
 YY=round(Y1)+200;                              %直角坐标原点在(200,200)为中心
for n=1:40000
    flag=0;     X=XX(n);      Y=YY(n);
    s=gs(n);   sf=sign(s);    trav=0;        dist=abs(s);
    while   (flag==0) & (X>1) & (X<400) & (Y>1) & (Y<400)...
            & (trav<dist)
        if u(X+1,Y)+ u(X-1,Y)+ u(X,Y+1)+u(X,Y-1)>=1
        flag=1;                                 %邻近的格子有粒子占据
        u(X,Y)=1;                               %新粒子凝聚,行走结束
    elseif rand(1)<0.5                          %任取一个随机数作比较
        X=X+sf;                                 %新粒子水平移动
    else
        Y=Y+sf;                                 %新粒子垂直移动
    end
```

```
            trav=trav + 1.0;
        end
    end
    pcolor(u);                                  %用颜色表示粒子占据的位置
    colormap(summer)
    axis([140 260 140 260])
```

图 8.20 是一次模拟的图形。

图 8.20 DLA 生长的模拟

§8.11 练 习

1. 用舍取法产生标准正态分布的随机数。

2. 用 MC 方法计算单位直径的小球体积，取 10 次计算的结果作平均计算均方差。

3. 模仿教材中的做法验证中心极限定理。要求取 10 个随机数相加，并用正态分布函数来拟合。

4. 用平均值法和辅助函数法计算积分

$$I = \int_0^1 \mathrm{e}^x \mathrm{d}a$$

分别取 5 次计算结果计算均方差，比较二者的精度。

5. 用面积法计算

$$I = \int_0^2 (4 - x^2)\mathrm{d}x$$

6. 用 MC 方法模拟伽尔顿板实验。思路是：假设有一个钉子组成的等间距的垂直阵列，小珠从上部滚落下来，碰到一个钉子后将等概率向左或向右运动。

为了简单，还要假定，向左运动的小珠将碰到下一排左边一格的钉子，向右运动的钉子将碰到下一排右边一格的钉子。小珠一直这样落下去，直到底部。用模拟证明，即使所有小珠都从一个位置下落，在底部也会形成正态分布。

7. 用 MC 方法模拟一维随机行走模型。假定在 t 时刻粒子的位置为 $x(t)$，在时刻 $t+\tau$ 粒子的位置为

$$x(t+\tau)=x(t)\pm l$$

这里 l 是平均自由程。符号的正负是随机的，可以这样选取，先产生 [0, 1] 的等概率数 r。如果 $r<0.5$ 则取负号，反之取正号。统计 1000 个粒子经过一段时间间隔 (比如 50τ，100τ) 后位置的分布状况。

这种随机行走模型是模拟一维扩散运动的基础。

8. 将正方形九等分得九个较小的正方形。舍去每边中间小正方形，保留四角的四个正方形。此后将这四个较小的正方形按上述分割与舍去法则操作下去，得到一种介于线段与面之间的几何图形。下面是 IFS 系统迭代法作出图形的程序，这个图形有几条迭代规则? 根据程序的语句写出它们的迭代规则，同时编写一个与三角形分形图类似的画图程序。

```
N=50000; v=rand(N,1);
ABC=[0, 2*i, 2*1+2i,2];
xy=[i; zeros(N-1,1)];
for kk=2:N
if v(kk)<1/4, xy(kk)=1/3*xy(kk-1)+1/3*ABC(1);
elseif v(kk)<2/4, xy(kk)=1/3*xy(kk-1)+1/3*ABC(2);
elseif v(kk)<3/4, xy(kk)=1/3*xy(kk-1)+1/3*ABC(3);
else  xy(kk)=1/3*xy(kk-1)+1/3*ABC(4);
end
end
plot(xy, '.' ,'MarkerSize',4)
axis equal
```

9. 画出如图 8.21 所示的 IFS 分形树冠，它的迭代法则如下，作图的出发点可取在 (0, 0) 处。后面的百分数表示概率。

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{cases} \begin{pmatrix} 0.01x_n \\ 0.45y_n \end{pmatrix} & 5\% \\ \begin{pmatrix} -0.01x_n \\ -0.45y_n + 0.2 \end{pmatrix} & 15\% \\ \begin{pmatrix} 0.42x_n - 0.42y_n \\ 0.42x_n + 0.42y_n + 0.2 \end{pmatrix} & 40\% \\ \begin{pmatrix} 0.42x_n + 0.42y_n \\ -0.42x_n + 0.42y_n + 0.2 \end{pmatrix} & 40\% \end{cases}$$

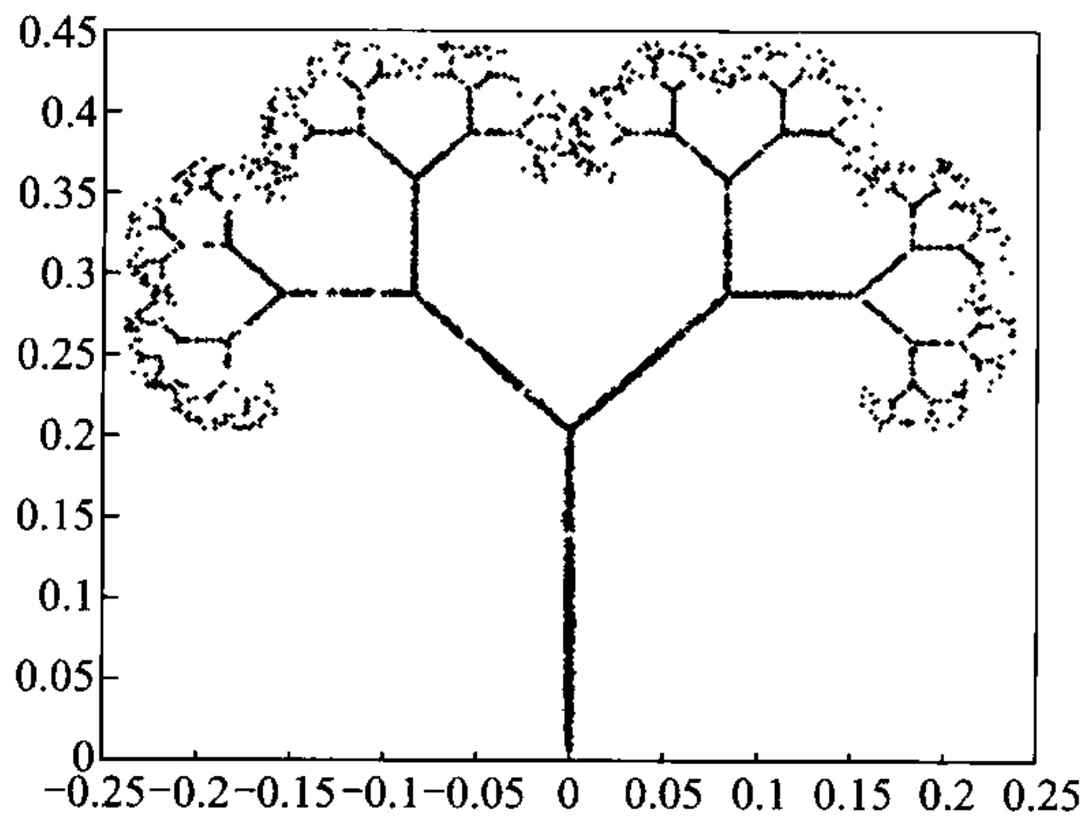


图 8.21 用 IFS 方法画出的分形树

10. 利用迭代可以画出山水图、烟雾、火焰等美妙的图形，也可以作出分形的动画，尝试一下从网上找到类似的程序或者自己编写这类程序，体会一下数学的神奇。

参 考 文 献

[1] Nicholas J. Giordano, Hisao Nakanishi. 计算物理 (影印版). 2 版. 北京：清华大学出版社，2007

[2] Rubin H. Landau, Manuel J. Páez, Cristian C. Bordeianu. Computational Physics: Problem Solving with Computers. 2nd Edition. Wiley-VCH, 2007

[3] Harvey Gould. An Introduction to Computer Simulation Methods: Applications to Physical Systems (计算机模拟方法在物理学中的应用) (影印版). 北京：高等教育出版社，2006

[4] Robert J. Schilling, Sandra L. Harris. 应用数值方法——使用 MATLAB 和 C 语言 (英文版). 北京：机械工业出版社，2004

[5] Tao Pang. An Introduction to Computational Physics (计算物理学导论) (影印版). 北京：世界图书出版公司北京公司，2001

[6] Michael T. Heath. 科学计算导论 (影印版). 2 版. 北京：清华大学出版社，2001

[7] Samuel S M Wong. 物理学和工程学中的计算方法 (英文版). 2 版. 北京：世界图书出版公司北京公司，2000

[8] 海因茨 · 奥托 · 佩特根，哈特穆特 · 于尔根斯，迪特马尔 · 绍伯. 混沌与分形——科学的新疆界. 2 版. 田逢春，译. 北京：国防工业出版社，2008

[9] Cleve B. Moler. MATLAB 数值计算. 喻文健，译. 北京：机械工业出版社，2006

[10] 李水根，赵翔鹏. 二维和高维空间的分形图形艺术. 北京：科学出版社，2009

[11] John H. Mathews, Kurtis D. Fink. 数值方法 (MATLAB 版). 4 版. 周璐，陈渝，钱方，等译. 北京：电子工业出版社，2005

[12] Steven E. Koonin. 计算物理学. 秦克诚，译. 北京：高等教育出版社，1992

[13] 张志涌，杨祖樱. MATLAB 教程. 北京：北京航空航天出版社，2006

[14] 彭芳麟. 数学物理方程的 MATLAB 解法与可视化. 北京：清华大学出版社，2004

[15] 彭芳麟，等. 理论力学计算机模拟. 北京：清华大学出版社，2002

[16] Alfio Quarteroni, Fausto Saleri. Scientific Computing with MATLAB and Octave. 2nd ed. New York: Springer, 2006

[17] Claude Gomez (Editor). Engineering and Scientific Computing with Scilab. Boston: Birkhäuser, 1999

郑重声明

高等教育出版社依法对本书享有专有出版权。任何未经许可的复制、销售行为均违反《中华人民共和国著作权法》，其行为人将承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。为了维护市场秩序，保护读者的合法权益，避免读者误用盗版书造成不良后果，我社将配合行政执法部门和司法机关对违法犯罪的单位和个人给予严厉打击。社会各界人士如发现上述侵权行为，希望及时举报，本社将奖励举报有功人员。

反盗版举报电话：(010) 58581897/58581896/58581879

传　　真：(010) 82086060

E－mail：dd@hep.com.cn

通信地址：北京市西城区德外大街 4 号
高等教育出版社打击盗版办公室

邮　　编：100120

购书请拨打电话：(010) 58581118

策划编辑　高　建
责任编辑　李　茜
封面设计　赵　阳
责任绘图　尹　莉
版式设计　张　岚
责任校对　张　颖
责任印制　尤　静

封面
书名
版权
前言
目录
第一章　MATLAB简介
1.1　MATLAB的操作界面
1.1.1　操作界面介绍
1.1.2　指令窗中的功能
1.1.3　数据存储与显示
1.2　数据格式与算符
1.2.1　量的输入
1.2.2　矩阵
1.2.3　列阵
1.2.4　数据网格
1.2.5　基元列阵
1.2.6　结构数组
1.2.7　字符和文本
1.3　编程
1.3.1　编辑程序
1.3.2　调试程序
1.3.3　设置搜索路径
1.3.4　两类程序文件
1.3.5　流程控制
1.3.6　数据输入与输出
1.4　作图
1.4.1　作图功能概述
1.4.2　二维曲线作图指令
1.4.3　复数作图
1.4.4　基本的三维图
1.4.5　四维数据的表现——切片函数slice
1.4.6　复变函数图形
1.4.7　动画
1.5　画物理场的专用指令
1.5.1　等值线表现二维标量场
1.5.2　等值面表现三维标量场
1.5.3　用箭头、流线表现二维矢量场
1.5.4　用箭头、流线、锥体、流管、流带表现三维矢量场
1.6　图形窗口的编辑功能
1.7　制作图形用户界面（GUI）
1.8　浮点运算与数值计算的误差
1.9　物理应用：Logistic模型的周期分岔与混沌现象
1.10　练习
第二章　迭代-分形图形
2.1　迭代与分形
2.1.1　分形树
2.1.2　Sierpinski三角形
2.1.3　科赫雪花曲线
2.1.4　L系统
2.2　复变函数迭代
2.2.1　Julia集
2.2.2　Mandelbrot集
2.3　分形与分形维
2.4　练习
第三章　数值微分与数值积分

３．１　数值微分与数值积分
３．１．１　数值微分的算法
３．１．２　数值积分的算法
３．２　ＭＡＴＬＡＢ指令
３．２．１　差分运算（ｄｉｆｆ）
３．２．２　梯度计算（ｇｒａｄｉｅｎｔ）
３．２．３　离散拉普拉斯算符（ｄｅｌ２）
３．２．４　梯形积分（ｔｒａｐｚ）
３．２．５　累计梯形积分（ｃｕｍｔｒａｐｚ）
３．２．６　函数积分（ｑｕａｄ，ｑｕａｄｌ）
３．２．７　函数的二重积分（ｄｂｌｑｕａｄ）
３．２．８　函数的三重积分（ｔｒｉｐｌｅｑｕａｄ）
３．３　用符号工具箱计算微积分
３．４　环形电流的磁场——物理场的可视化
３．５　分子振动的半经典量子化
３．６　练习
第四章　数据处理
４．１　插值
４．１．１　插值分类
４．１．２　拉格朗日插值法
４．１．３　分段三次埃尔米特插值
４．１．４　ＭＡＴＬＡＢ插值指令
４．２　曲线拟合
４．２．１　曲线拟合的最小二乘法
４．２．２　多项式拟合
４．２．３　多项式拟合的指令ｐｏｌｙｆｉｔ
４．２．４　图形窗口的曲线拟合功能
４．２．５　指数拟合
４．２．６　线性最小二乘法的一般形式
４．３　曲线拟合工具箱
４．４　求方程的零点
４．４．１　求单调连续函数ｆ（ｘ）的实数根的算法
４．４．２　ＭＡＴＬＡＢ解方程的指令
４．４．３　搜寻函数的极小值
４．５　快速傅里叶变换
４．５．１　离散傅里叶变换
４．５．２　傅里叶级数与傅里叶积分
４．５．３　快速傅里叶变换的ＭＡＴＬＡＢ指令
４．５．４　快速傅里叶变换的算法
４．６　练习
第五章　解常微分方程
５．１　龙格 - 库塔法
５．１．１　基本思想
５．１．２　二阶龙格 - 库塔法
５．１．３　三阶与四阶龙格 - 库塔法
５．２　变步长的龙格 - 库塔法
５．３　常微分方程组的初值问题
５．４　用ＭＡＴＬＡＢ解常微分方程
５．５　刚性问题
５．６　事件
５．７　误差
５．８　性能
５．９　物理应用
５．９．１　刚体绕瞬心的转动方程
５．９．２　弹簧摆运动

５．９．３　圆锥陀螺运动
５．１０　边值问题和本征值
５．１０．１　边值问题的直接积分
５．１０．２　打靶法求弦振动方程本征值
５．１０．３　一维薛定谔方程的定态解
５．１１　用指令ｂｖｐ４ｃ解边值问题与本征值问题
５．１１．１　用指令ｂｖｐ４ｃ解一维本征值问题
５．１１．２　有两个解的边值问题
５．１１．３　马蒂厄方程的本征值
５．１１．４　艾登方程
５．１１．５　Ｆａｌｋｎｅｒ－Ｓｋａｎ边值问题
５．１１．６　在ｘ＝０处有突变的问题
５．１２　练习
第六章　混沌
６．１　单摆——从周期运动到混沌
６．１．１　单摆的动力学方程
６．１．２　周期运动
６．１．３　有阻尼有驱动的情况——耗散系的混沌
６．２　倒摆与达芬方程
６．２．１　倒摆的运动方程
６．２．２　倒摆的混沌运动
６．３　自激振动——范德波尔方程
６．３．１　运动方程
６．３．２　ＶＤＰ方程通向混沌的道路
６．３．３　吸引子类型及其频谱
６．３．４　分岔
６．４　洛伦茨方程——奇怪吸引子
６．４．１　倍周期窗与费根鲍姆数
６．４．２　由阵发通向混沌
６．４．３　庞加莱截面图
６．５　练习
第七章　解偏微分方程
７．１　差分法解热传导方程
７．１．１　显式差分公式
７．１．２　隐式公式与平均隐式公式
７．２　差分法解弦振动方程
７．２．１　显式格式
７．２．２　初始条件
７．２．３　实例
７．３　差分法与松弛法解椭圆型方程
７．３．１　显式差分公式
７．３．２　边界条件
７．３．３　迭代法与松弛法
７．４　偏微分方程的工具箱（ＰＤＥＴＯＯＬ）
７．４．１　ＰＤＥＴＯＯＬ的功能演示
７．４．２　ＰＤＥＴＯＯＬ中方程的输入格式
７．４．３　边界条件
７．４．４　可解问题的分类
７．４．５　解题步骤
７．４．６　解一维与三维问题
７．４．７　拉普拉斯方程与泊松方程
７．４．８　热传导方程
７．４．９　波动方程
７．４．１０　平面区域的本征值问题
７．５　特殊函数

７．５．１　勒让德函数
７．５．２　贝塞尔函数
７．５．３　调用ＭＡＰＬＥ计算特殊函数
７．５．４　平面波展开
７．５．５　环形电流的磁场
７．５．６　高斯积分
７．６　练习
第八章　蒙特卡罗方法
８．１　蒙特卡罗方法的发展过程
８．２　随机变量、密度函数与分布函数
８．３　大数定理与中心极限定理
８．４　随机数与随机抽样
８．４．１　随机数的产生
８．４．２　随机抽样
８．５　计算定积分
８．６　热力学的平衡态
８．７　麦克斯韦速率分布律
８．８　链式反应的模拟
８．９　迭代函数系统（ＩＦＳ）
８．１０　分形生长模型
８．１０．１　抛射沉积模型
８．１０．２　森林和薄膜的关联生长
８．１０．３　ＤＬＡ
８．１１　练习
参考文献