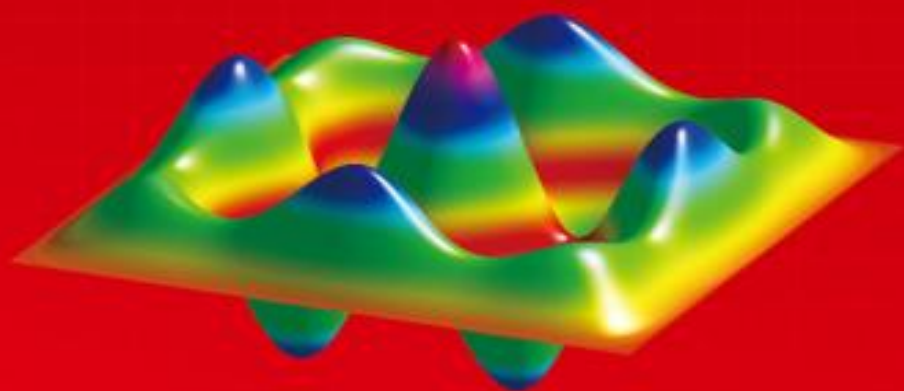


信息科学与技术丛书

张 晓 编著

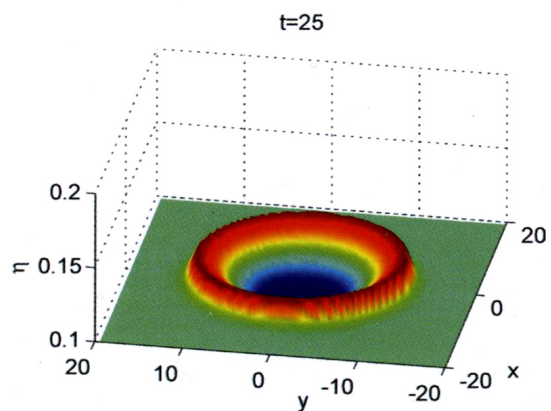
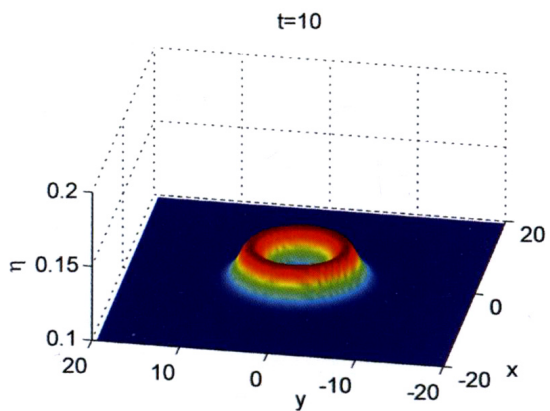
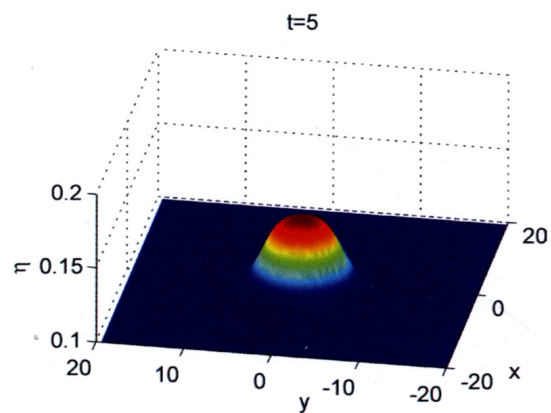
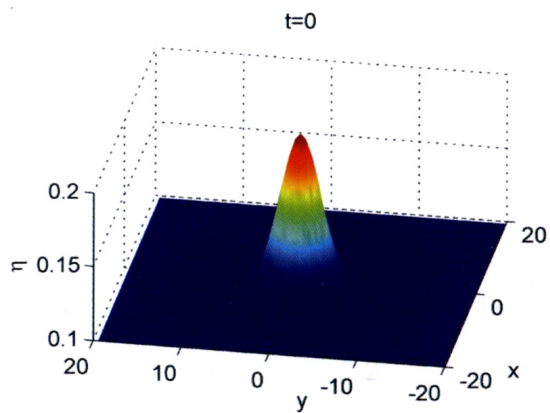


Matlab 微分方程高效解法： 谱方法原理与实现

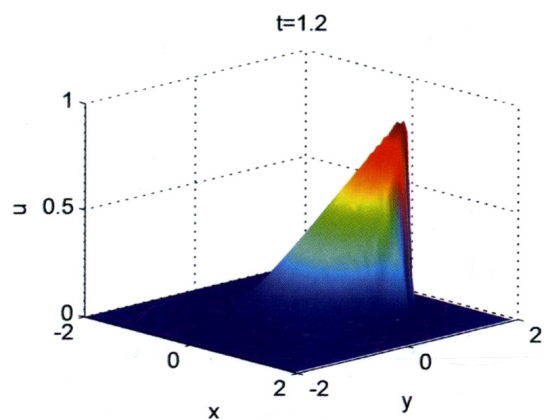
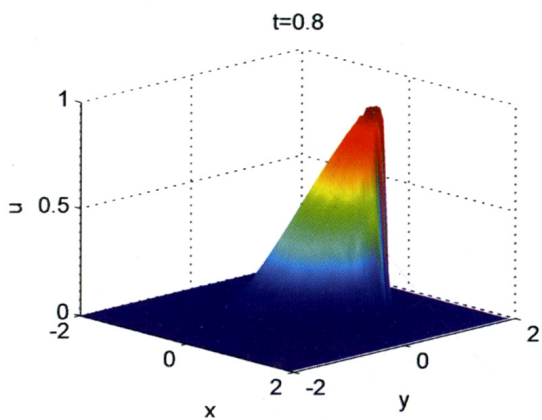
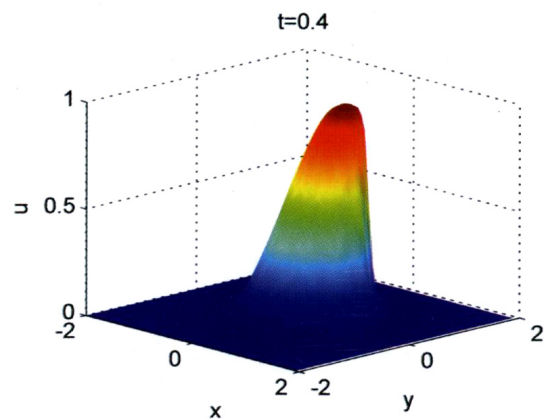
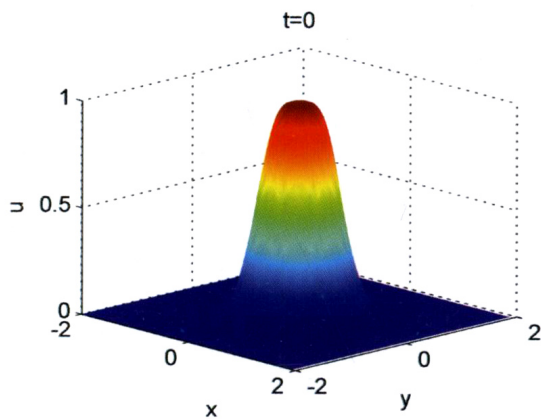
- 代码简洁——用二三十行代码解决常见微分问题
- 速度飞快——高效程序大多在二十秒内输出结果
- 结果精确——计算精度通常可达到十位有效数字



机械工业出版社
CHINA MACHINE PRESS

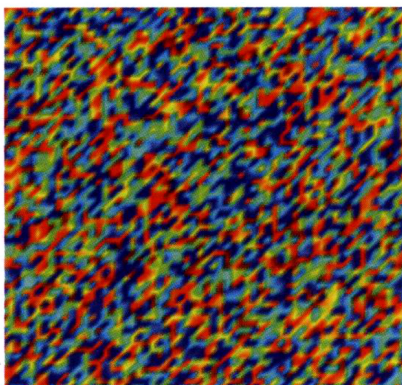


浅水方程的数值结果（水的波纹）

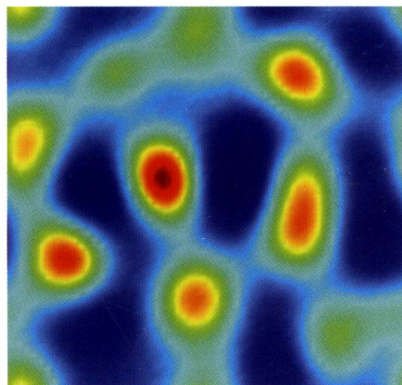


二维 Burgers 方程的数值结果（激波的形成）

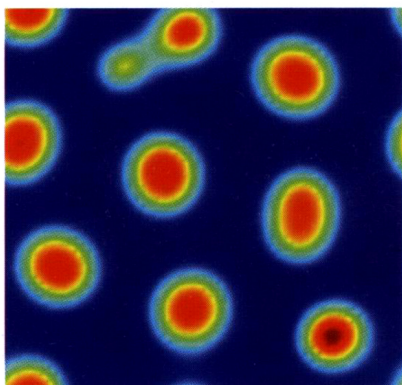
$t=0$



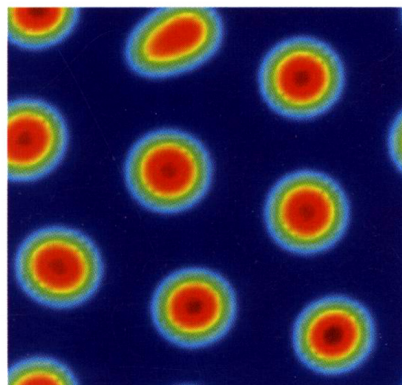
$t=0.1$



$t=0.2$

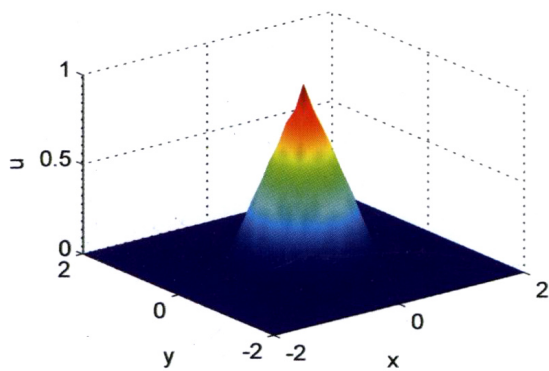


$t=0.3$

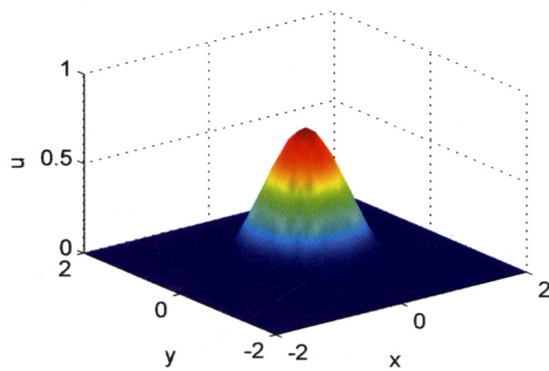


Schnakenberg 模型的数值结果（斑图的形成）

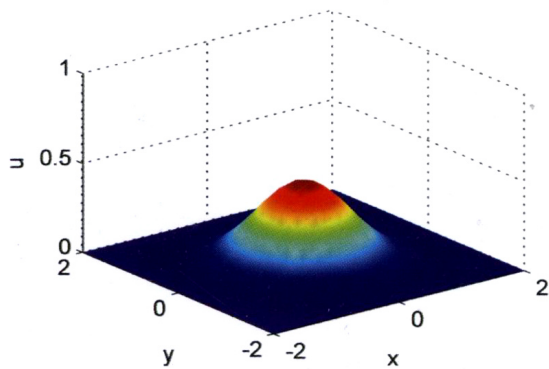
$t=0$



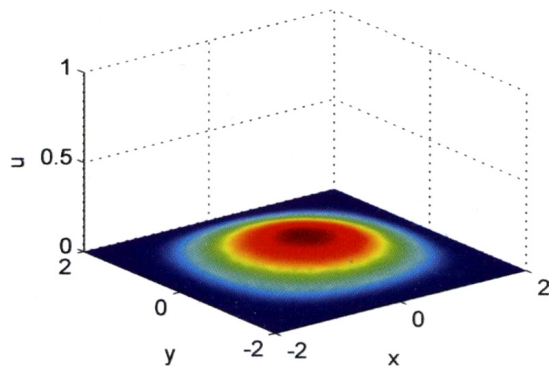
$t=0.02$



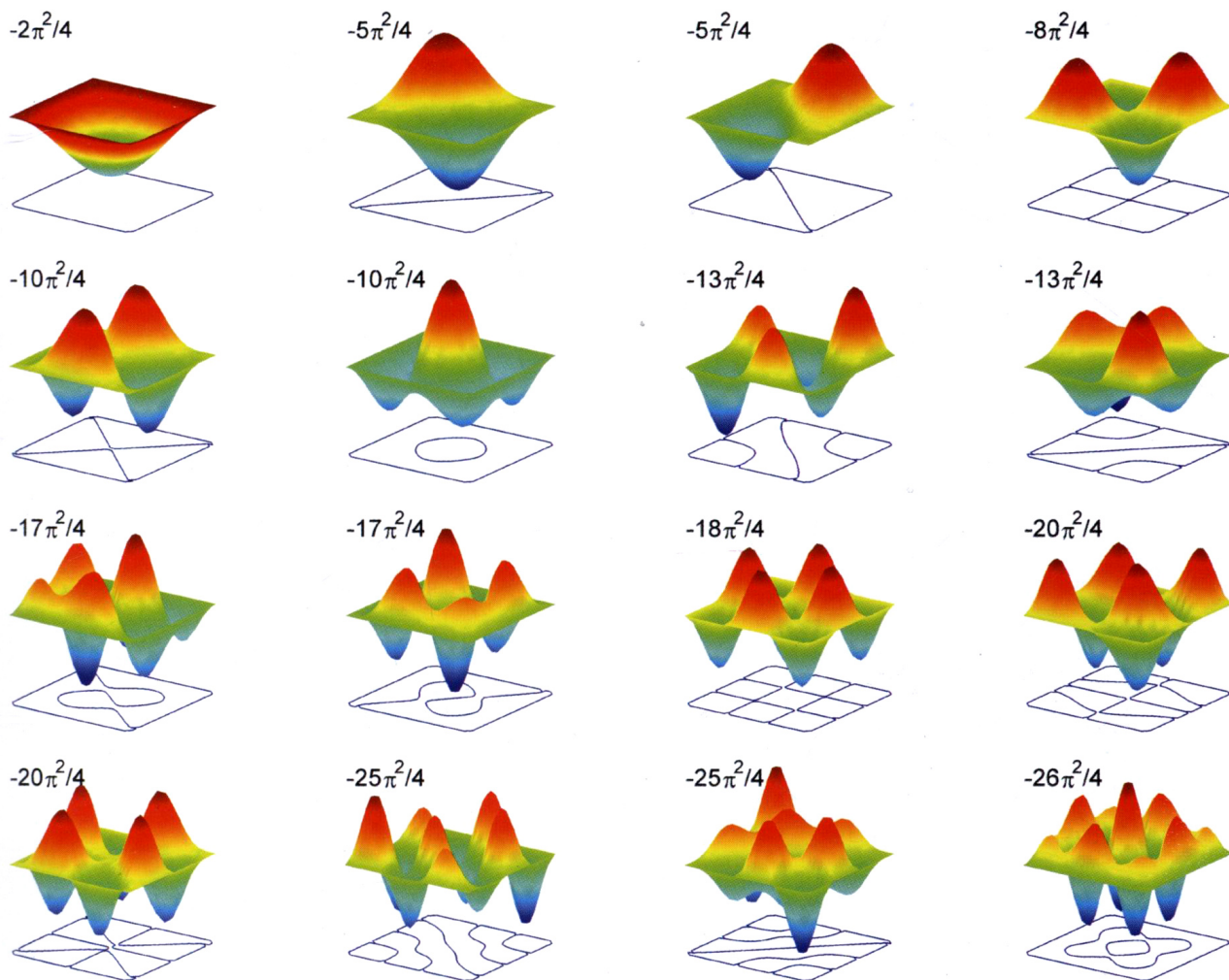
$t=0.1$



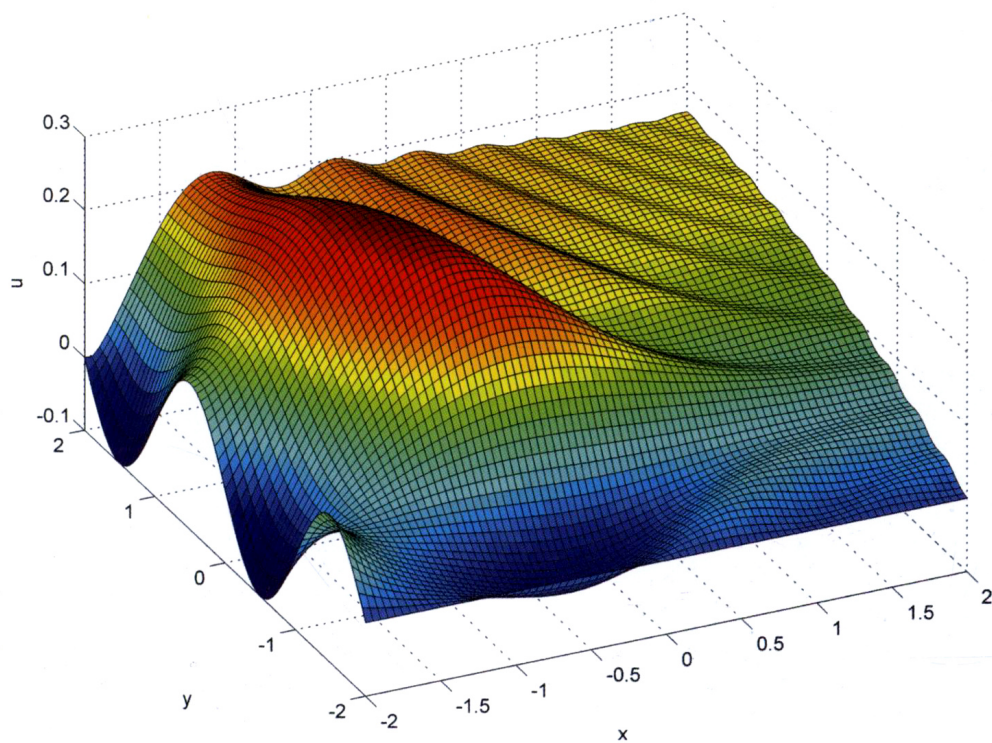
$t=0.5$



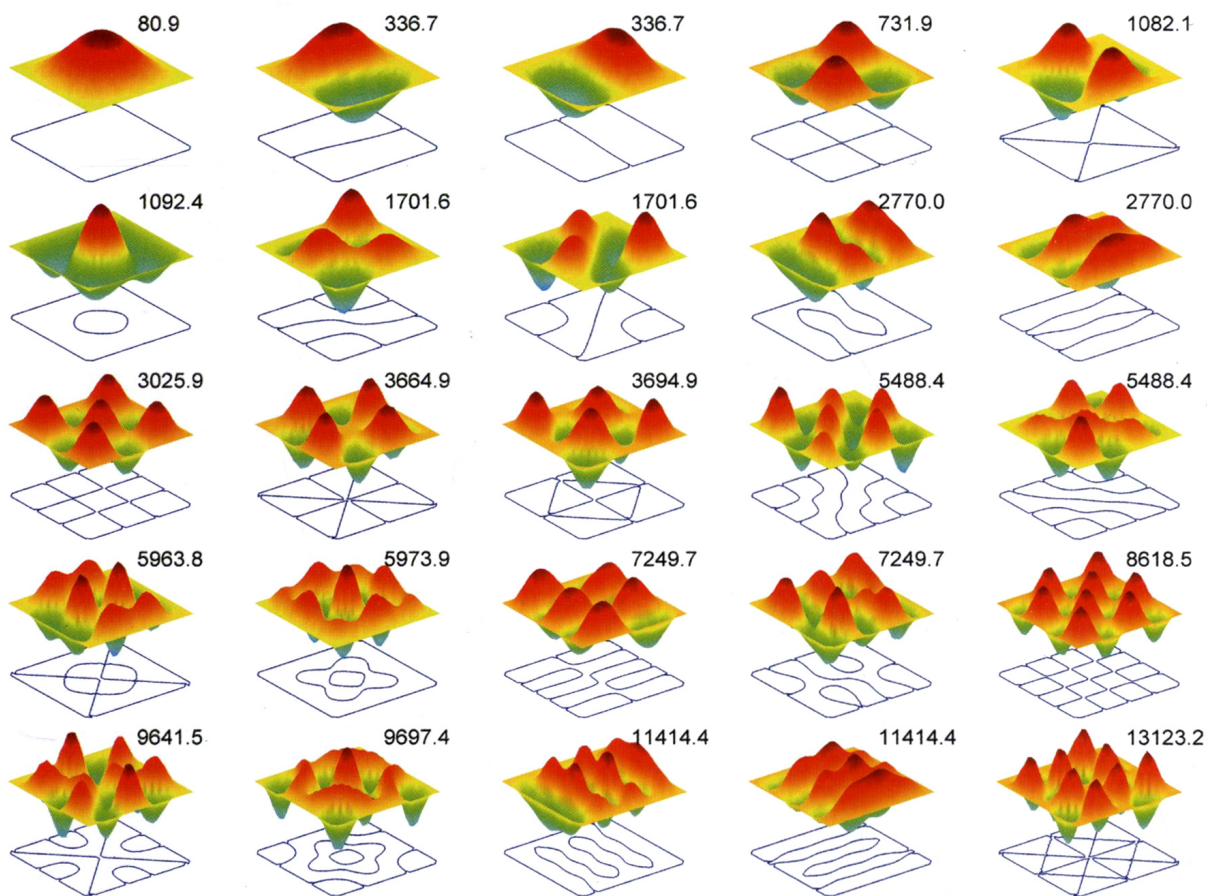
二维热传导方程的数值结果



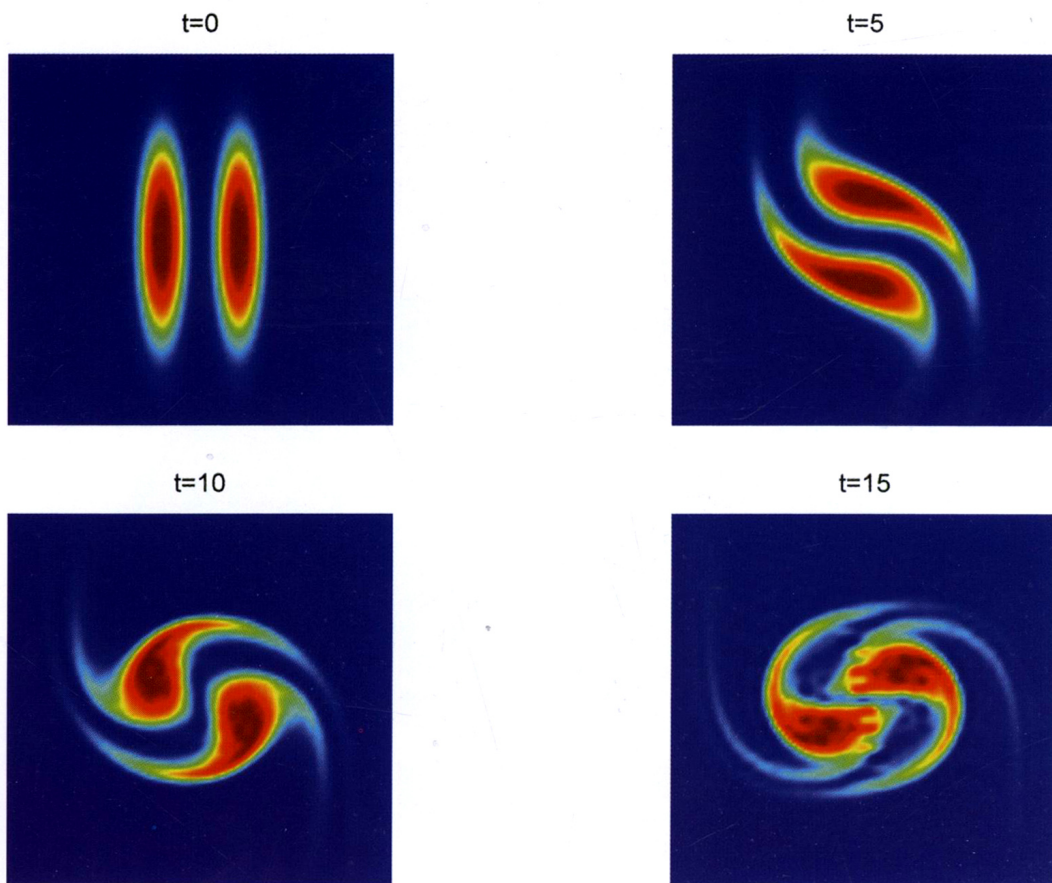
二维无限深方势阱中粒子波函数的特征值和特征函数（定态薛定谔方程）



二维泊松方程的数值解



二维双调和算符的特征值和特征函数



二维平流 - 扩散方程的数值结果

信息科学与技术丛书

Matlab 微分方程高效解法： 谱方法原理与实现

张 晓 编著



机械工业出版社

本书系统地介绍了高效求解微分问题的 Matlab 程序编写方法和技巧，并充分地给出了针对各种实际微分问题的实例，包括三类基本线性微分方程（抛物型、双曲型、椭圆型方程），四阶微分方程（双调和方程等），特征值问题（定态薛定谔方程等）和广义特征值问题，以及其他非线性、耦合的复杂微分方程（组），如：KdV 方程、非线性薛定谔方程、浅水方程、Ginzburg-Landau 方程、Burgers 方程、反应-扩散方程、平流-扩散方程。同时还介绍了不同边界条件下的具体计算形式，包括周期性边界条件，第一、二、三类边界条件。本书给出的每个实例都代表了一类问题的通用解决方法。

本书适合作为相关专业高年级本科生或研究生的教材，也可供相关工程技术人员参考。

书中代码可在 <http://www.cmpbook.com> 免费下载。

图书在版编目（CIP）数据

Matlab 微分方程高效解法：谱方法原理与实现 / 张晓编著. —北京：机械工业出版社，2015.9

（信息科学与技术丛书）

ISBN 978-7-111-51623-1

I. ①M… II. ①张… III. ①Matlab 软件—应用—微分方程
IV. ①O175.8-39

中国版本图书馆 CIP 数据核字（2015）第 226808 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：车 忱 责任校对：张艳霞

责任印制：乔 宇

保定市中国画美凯印刷有限公司印刷

2016 年 1 月第 1 版 · 第 1 次印刷

184mm×260mm · 12.25 印张 · 2 插页 · 303 千字

0001—2500 册

标准书号：ISBN 978-7-111-51623-1

定价：43.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

网络服务

服务咨询热线：(010) 88361066

机工官网：www.cmpbook.com

读者购书热线：(010) 68326294

机工官博：weibo.com/cmp1952

(010) 88379203

教育服务网：www.cmpedu.com

封面无防伪标均为盗版

金书网：www.golden-book.com

出版说明

随着信息科学技术的迅速发展,人类每时每刻都会面对层出不穷的新技术和新概念。毫无疑问,在节奏越来越快的工作和生活中,人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书来获取新知识和新技能,从而不断提高自身素质,紧跟信息化时代发展的步伐。

众所周知,在计算机硬件方面,高性价比的解决方案和新型技术的应用一直备受青睐;在软件技术方面,随着计算机软件的规模和复杂性与日俱增,软件技术不断地受到挑战,人们一直在为寻求更先进的软件技术而奋斗不止。目前,计算机和互联网在社会生活中日益普及,掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用,所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们了解和掌握新知识、新技能的热切期待,以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状,机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络和工程应用等内容,注重理论与实践的结合,内容实用、层次分明、语言流畅,是信息科学与技术领域专业人员不可或缺的参考书。

目前,信息科学与技术的发展可谓一日千里,机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作,为推进我国的信息化建设作出贡献。

机械工业出版社

序

Matlab是目前在科学、技术及工程中很重要且常用的一种成熟的计算机软件工具。除了一些基本简单的应用功能外,有许多更复杂的问题也可以借助Matlab来帮助解决,其中基于谱方法原理,对于各类微分方程的求解,在求解区域中,较一般的有限元、有限差分法相比,具有更高的精度。该方法虽然有数十年的发展历史,但在国内还有待于普及,相关图书很少。作者将其多年的数值计经的经验,以及在英国华盛顿大学应用数学系所学到的“谱方法”,结合Matlab的实现,编著本书,将有助于推动相关方法的实际应用,并且有助于快速解决现有的科学、技术和工程中的实际问题。

我认为这是一本和诸多实际问题紧密结合得很好的学习工具书，适合大学生、研究生、以及从事科学、技术及工程应用的人员作为参考。

薛 平

2015年11月16日

于清华园

前 言

我第一次进行微分方程数值求解是在读硕士的时候，当时查找了很多资料，用现在看来很整脚的办法写了大段的 Matlab 代码解决了一个描述激光器谐振腔的简单边值问题。随后我开始对数值计算产生了强烈的兴趣，继续用不那么专业的方法处理研究中遇到的微分方程。直到读博期间我得到了国家留学基金委的资助，到美国华盛顿大学（University of Washington）应用数学系接受联合培养，在系主任 J. Nathan Kutz 教授的课堂上、组会上以及每周 1~2 次的面对面单独指导中，我接触到了谱方法（spectral method）。谱方法巧妙的原理和导师精湛的 Matlab 编程技艺深深地震撼了我。求解微分方程（组）竟可以如此简洁、高效、优雅，那段时间内几乎每天我都会产生这样的感慨。回国后我曾在不同场合做过关于我理论结果的现场报告，于是常有研究生向我请教那些复杂的微分方程（组）是如何求解的。我在他们身上看到了我当初的影子，同时我也意识到谱方法的实际应用在国内还不普及。例如非线性光纤光学中最基本的非线性薛定谔方程（该方程也出现在非线性量子力学中）可以根据谱方法用寥寥十几行的 Matlab 代码获得高精度数值解，而我所认识的研究生还在使用既不简便也不精确的分步傅里叶法。不久之后我博士毕业，由于一心想进高校任教的我没有找到想要的教职，所以刚刚拿到博士学位后就待业了 3 个月。于是我在年近 30 岁的时候开始面对既无业、也无钱、还单身的人生悲剧，想起昔日的同窗、朋友大多已成家立业，内心的苦闷愈甚却无处排解。这时候我开始着手将已掌握的谱方法及其重要的技巧和经验总结成书，一来要让更多的人接触到这些有价值的数值方法，二来潜心写书可令我暂时忘记眼前的烦恼。之后我来到清华大学物理系做博士后研究工作，开始了一项令人振奋的研究，当然也不忘多方联系本书的出版事宜。如今我很高兴地看到本书即将出版发行，它不但是是一本可以教你如何高效、便捷地解决各类微分问题的实用教程，同时也是我身处逆境不甘认命的一个见证。

本书中，常见的微分问题均可在 40 行代码内解决，其中的核心代码不到 10 行，且代码只需在默认安装的 Matlab 上即可执行，并不依赖于第三方插件。此外，Matlab 代码中采用的谱方法以及时间步进法是在计算精度、运算量、稳定性等方面颇具优势的数值计算方法，所有掌握高等数学和 Matlab 语法的读者在看完本书之后，均可以得心应手地、高效简洁地、精确地解决各类常见微分问题。本书分为上篇、中篇和下篇。上篇介绍欧拉法、龙格-库塔法、有限差分法和有限元法等基础数值知识，旨在使读者了解在谱方法之前出现的主要数值方法的基本原理，并能够与后文中的谱方法形成对比，加深理解。中篇

和下篇分别介绍了不同边界条件下的谱方法，对数值方法比较了解的读者可快速浏览上篇后直接阅读中篇和下篇。

感谢 J. Nathan Kutz 教授，他深厚的数学功底和过人的物理直觉令我受益匪浅，极大地感召我在学术的道路上一路前行。感谢我的硕导和博导宋晏蓉教授、张新平教授，这两位教授工作出众并且将学生的自身发展放在首要位置，在研究组人手不足的情况下仍然支持学生出国学习。感谢密云二中的数学老师王嘉新，我对数学的热爱都始于他生动有趣的教学以及慧眼识才的培养。感谢清华大学物理系的薛平教授对我的悉心栽培和提携，他在我学术道路最艰难的起步阶段给予了无私的支持，并为本书作序。最后感谢我的家人、亲戚和朋友多年来予以我的支持和鼓励。

没有几位恩师兢兢业业的工作以及亲友们的关怀，就没有我的这本书，也没有我的今天。

张 晓

2015 年 8 月于清华园

提 示

本书代码中对变量的命名遵循一定规律：

以 **t** 结尾的变量代表对某函数做傅里叶变换的结果（取 **transform** 首字母），如 **ut** 为函数 u 的傅里叶变换结果（即频谱）。

以 **d** 开头的变量代表对函数求导的结果（取 **derivative** 首字母），如 **du** 代表 $\partial u / \partial t$ 。

以 **sol** 结尾的变量代表最终结果（取 **solution** 前三个字母），如 **usol** 代表函数 u 的最终结果。

有时为了传递参数，不得不把几个变量合并为一个变量，新的变量名就是直接把这几个变量名合并。如 **uv**=[**u**; **v**]。

变量 **bc** 代表边界条件相关矩阵（取 **boundary conditions** 的缩写）。

上述命名规律可组合使用。如：**utsol**、**vtsol** 分别代表函数 u 、 v 在频域上的最终结果，**uvt sol** 代表将 **utsol**、**vtsol** 合并后的结果。**dut**、**dvt** 分别代表函数 u 、 v 在频域上的导数，**duvt** 代表将 **dut**、**dvt** 合并后的结果。

目 录

出版说明

序

前言

提示

上篇 前置知识	1
第 1 章 初值问题和边值问题	1
1.1 初值问题	1
1.1.1 欧拉法	2
1.1.2 局部截断误差	3
1.1.3 改进的欧拉法	4
1.1.4 龙格-库塔法	6
1.1.5 ode 系列函数的用法	9
1.1.6 高阶微分方程的降阶	15
1.2 边值问题	17
1.2.1 打靶法	18
1.2.2 bvp 系列函数的用法	22
第 2 章 有限差分法和有限元法	25
2.1 有限差分法	25
2.1.1 有限差分法中的数值微分	25
2.1.2 求导的矩阵形式	26
2.2 偏微分方程的差分解法	30
2.2.1 二维泊松方程	30
2.2.2 一维热传导方程	34
2.2.3 一维波动方程	37
2.3 有限元法和 Matlab 偏微分工具箱	40
2.3.1 基本操作	41
2.3.2 二维泊松方程	48
2.3.3 二维热传导方程	52
2.3.4 二维波动方程	53
2.3.5 二维特征值问题	54
中篇 周期性边界条件下的谱方法	57
第 3 章 傅里叶谱方法	57
3.1 傅里叶谱方法的原理	57

3.1.1	快速傅里叶变换	57
3.1.2	求导、积分与傅里叶谱方法	61
3.1.3	傅里叶谱方法的步骤	63
3.1.4	滤波法	66
3.2	傅里叶谱方法求解基本偏微分方程 (组)	67
3.2.1	一维波动方程	67
3.2.2	二维波动方程	69
3.2.3	一维非线性薛定谔方程	71
3.3	傅里叶谱方法求解复杂偏微分方程 (组)	73
3.3.1	一维 KdV 方程	73
3.3.2	二维浅水方程组	74
3.3.3	二维粘性 Burgers 方程	76
3.3.4	二维 Schnakenberg 模型	78
第 4 章	谱求导矩阵	81
4.1	谱求导矩阵的导出和应用	81
4.1.1	谱方法插值	81
4.1.2	谱求导矩阵	82
4.1.3	用谱求导矩阵求解偏微分方程的步骤	88
4.2	利用谱求导矩阵求解基本偏微分方程 (组)	92
4.2.1	一维线性谐振子的定态薛定谔方程	92
4.2.2	二维线性谐振子的定态薛定谔方程	94
4.2.3	一维波动方程	96
4.2.4	二维波动方程	98
4.3	利用谱求导矩阵求解复杂偏微分方程 (组)	100
4.3.1	Ginzburg-Landau 方程	100
4.3.2	耦合非线性薛定谔方程组	102
4.3.3	二维 Schnakenberg 模型	103
4.3.4	二维平流-扩散方程	105
下篇	第一类、第二类和第三类边界条件下的谱方法	108
第 5 章	切比雪夫谱方法	108
5.1	切比雪夫求导矩阵的导出	108
5.1.1	吉布斯现象和龙格现象	108
5.1.2	切比雪夫求导矩阵	112
5.2	狄利克莱边界条件 (第一类边界条件)	119
5.2.1	一维泊松方程	119

5.2.2	二维泊松方程	122
5.2.3	Allen-Cahn 方程	128
5.2.4	二维热传导方程	131
5.2.5	一维特征值问题	135
5.2.6	二维特征值问题	137
5.3	诺依曼边界条件（第二类边界条件）	138
5.3.1	一维泊松方程	139
5.3.2	二维泊松方程	140
5.3.3	一维热传导方程	143
5.3.4	二维波动方程	146
5.3.5	一维四阶问题	148
5.3.6	二维四阶问题	150
5.4	洛平边界条件（第三类边界条件）	152
5.4.1	一维泊松方程	152
5.4.2	二维泊松方程	154
5.4.3	一维热传导方程	156
5.4.4	二维热传导方程	158
5.5	利用切比雪夫谱方法求解复杂偏微分方程（组）	161
5.5.1	广义特征值问题	161
5.5.2	二维 Barkley 模型	162
5.5.3	二维平流-扩散方程	165
附录		168
附录 A	Matlab 主要符号和函数	168
A.1	运算符、操作符和常量	168
A.2	矩阵、图形窗口相关函数	170
附录 B	将计算结果制作成 gif 动画	177
参考文献		179
跋		180

上篇 前置知识

第1章 初值问题和边值问题

本章将通过最基本、简单的微分问题——初值问题（initial value problem）和边值问题（boundary value problem）的数值解法，引入欧拉法、改进的欧拉法、龙格-库塔法等方法的基本思想和内在关系，给出代码的实现并予以说明。此外还介绍了局部截断误差、刚性等基本概念，为后续章节中求解更复杂的偏微分问题做铺垫。

1.1 初值问题

初值问题是科研、工程技术应用中最常见的一类问题，一阶常微分方程的初值问题表述如下：

已知 $u(x)$ 的起始点 (x_0, u_0) 以及 $u(x)$ 的一阶导数表达式为二元函数 $f(x, u)$ ，求在区间 $[x_0, \infty)$ 上满足这两个条件的 $u(x)$ 。即：

$$\begin{cases} \frac{du}{dx} = f(x, u) \\ u(x_0) = u_0 \end{cases} \quad (1-1)$$

由于起始点 (x_0, u_0) 给出了初始状态的条件，所以被称为初始条件。在少数特殊情况下，可以直接求出初值问题的解析解。对于更一般的情况，就只能通过数值解法求出数值解。数值解法的基本思路就是先对 x 和 $u(x)$ 在区间 $[x_0, \infty)$ 上进行离散化，然后构造递推公式，再步进式地得到 $u(x)$ 在这些位置的近似取值。

这些离散的位置可表为：

$$x_1, x_2, \dots, x_n, \dots$$

数值解法可得到 $u(x)$ 在这些位置的近似取值：

$$u_1, u_2, \dots, u_n, \dots$$

显然， $u(x)$ 在这些离散点处的精确取值为：

$$u(x_1), u(x_2), \dots, u(x_n), \dots$$

令相邻两个离散点的间隔为一定值 h ，则有 $x_n = x_0 + nh$ ($n=1, 2, \dots$)，这里 h 也被称为数

值计算的步长。若能构造出这些位置的近似取值 u_n 的递推关系，也就是通过 u_n 获得 u_{n+1} 的公式，那么就可以依次获得 $u_1, u_2, \dots$ 。由于计算 u_{n+1} 只需要 u_n 的取值，所以上述方法也称为单步法。与之相对，若用于计算 u_{n+1} 的递推公式除包含 u_n 外，还包含 $u_{n-1}, u_{n-2}, \dots$ ，那么这种数值方法就称为多步法。

数值计算中最重要的就是递推公式，计算的精度、运算量都是由递推公式决定的。下面由浅入深地介绍几种基本的构造递推公式的方法，以引入数值计算的思路 and 重要概念。

1.1.1 欧拉法

欧拉法 (Euler method) 的核心就是将导数 (微商) 近似为差商。将导数近似为向前差商，则有：

$$\left. \frac{du}{dx} \right|_{x=x_n} \approx \frac{u(x_{n+1}) - u(x_n)}{h} \quad (1-2)$$

代入式 (1-1) 中的第 1 个式子，得：

$$u(x_{n+1}) = u(x_n) + hf(x_n, u(x_n)) \quad (1-3)$$

用 u_n 和 u_{n+1} 近似代替 $u(x_n)$ 和 $u(x_{n+1})$ ，得：

$$u_{n+1} = u_n + hf(x_n, u_n) \quad (1-4)$$

上式称为欧拉法，其中 $n=0, 1, \dots$ 。若已知 u_0 的取值，就可以用递推公式 (1-4) 步进式地计算出 $x_1, x_2, \dots$ 处的 $u_1, u_2, \dots$ 。欧拉法的几何意义如图 1-1 所示，从点 (x_0, u_0) 出发，以 $f(x_0, u_0)$ 为斜率做一直线段，与直线 $x=x_1$ 交于点 (x_1, u_1) ，该交点的纵坐标 u_1 就是 $u(x)$ 在该处的取值 $u(x_1)$ 的近似。再从点 (x_1, u_1) 出发，以 $f(x_1, u_1)$ 为斜率做一直线段，与直线 $x=x_2$ 交于 (x_2, u_2) ，其纵坐标 u_2 就是 $u(x_2)$ 的近似。继续重复这一过程，最终获得一条折线作为曲线 $u(x)$ 的近似，因此欧拉法也叫折线法。

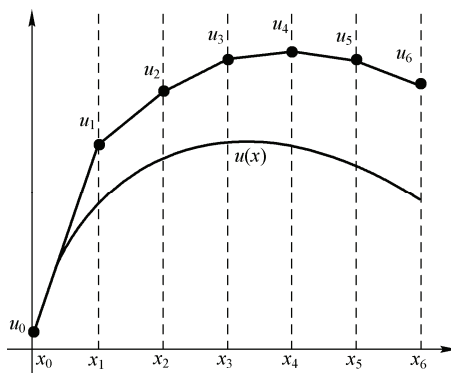


图 1-1 欧拉法的几何意义

如果将导数近似为向后差商，有：

$$\left. \frac{du}{dx} \right|_{x=x_{n+1}} \approx \frac{u(x_{n+1}) - u(x_n)}{h} \quad (1-5)$$

同样代入式 (1-1) 中的第 1 个式子, 并用 u_n 和 u_{n+1} 近似代替 $u(x_n)$ 和 $u(x_{n+1})$, 得:

$$u_{n+1} = u_n + hf(x_{n+1}, u_{n+1}) \quad (1-6)$$

上式称为后退的欧拉法 (backward Euler method), 其中 $n=0, 1, \dots$ 。它与欧拉法截然不同, 式 (1-4) 可直接用 x_n 和 u_n 计算得到 u_{n+1} , 而式 (1-6) 等号两边同时包含 u_{n+1} , 无法直接计算 u_{n+1} 具体数值, 所以也称为隐式欧拉法 (implicit Euler method)。

如果将导数近似为中心差商:

$$\left. \frac{du}{dx} \right|_{x=x_n} \approx \frac{u(x_{n+1}) - u(x_{n-1}))}{2h} \quad (1-7)$$

代入式 (1-1) 中的第 1 个式子, 并用 u_{n-1} 和 u_{n+1} 近似代替 $u(x_{n-1})$ 和 $u(x_{n+1})$, 得:

$$u_{n+1} = u_{n-1} + 2hf(x_n, u_n) \quad (1-8)$$

用上式计算 u_{n+1} 的时候需要 u_{n-1} 和 u_n 的值, 所以称其为二步欧拉法。实际使用二步欧拉法的时候, 起初就要用到 u_0 和 u_1 的取值, 若 u_1 未知, 一般就先用欧拉法求出 u_1 的值, 再使用二步欧拉法。

1.1.2 局部截断误差

在利用某种数值计算方法由 $u_1, u_2, \dots, u_n$ 逐步递推的过程中, 每一步产生的误差都会积累在一起。在 x_n 处的误差为 $u(x_n) - u_n$, 称为整体截断误差。整体截断误差由此前每一步的误差以及误差之间的传递关系决定, 分析它是极为复杂的。所以为了简化问题, 只考虑局部的情况。那么假设 $u(x_n) = u_n$, 由 x_n 开始计算一步的误差为 $u(x_{n+1}) - u_{n+1}$, 此即为局部截断误差。若某种数值计算方法的局部截断误差为 $O(h^{p+1})$, 则称该方法的精度为 p 阶。

下面分析欧拉法、后退的欧拉法、二步欧拉法的精度。

(1) 对于欧拉法:

$$u_{n+1} = u_n + hf(x_n, u_n) \quad (1-9)$$

假设 $u_n = u(x_n)$, 并利用 $u'(x) = f(x, u)$:

$$u_{n+1} = u(x_n) + hu'(x_n) \quad (1-10)$$

用泰勒公式将 $u(x_{n+1})$ 在 x_n 处展开:

$$u(x_{n+1}) = u(x_n) + hu'(x_n) + O(h^2) \quad (1-11)$$

上两式相减:

$$u(x_{n+1}) - u_{n+1} = O(h^2) \quad (1-12)$$

所以，欧拉法的精度为 1 阶。

(2) 对于隐式欧拉法：

$$u_{n+1} = u_n + hf(x_{n+1}, u_{n+1}) \quad (1-13)$$

假设等号右边有 $u_n = u(x_n)$ 、 $u_{n+1} = u(x_{n+1})$ ，并利用 $u'(x) = f(x, u)$ ：

$$u_{n+1} = u(x_n) + hu'(x_{n+1}) \quad (1-14)$$

将上式中的 $u'(x_{n+1})$ 在 x_n 处做泰勒展开：

$$u_{n+1} = u(x_n) + h[u'(x_n) + O(h)] = u(x_n) + hu'(x_n) + O(h^2) \quad (1-15)$$

将 $u(x_{n+1})$ 在 x_n 处泰勒展开：

$$u(x_{n+1}) = u(x_n) + hu'(x_n) + O(h^2) \quad (1-16)$$

前两式相减：

$$u(x_{n+1}) - u_{n+1} = O(h^2) \quad (1-17)$$

则：隐式欧拉法的精度为 1 阶。

(3) 对于二步欧拉法：

$$u_{n+1} = u_{n-1} + 2hf(x_n, u_n) \quad (1-18)$$

假设有 $u_n = u(x_n)$ 、 $u_{n-1} = u(x_{n-1})$ ，并利用 $u'(x) = f(x, u)$ ：

$$u_{n+1} = u(x_{n-1}) + 2hu'(x_n) \quad (1-19)$$

将上式中的 $u(x_{n-1})$ 在 x_n 处泰勒展开：

$$\begin{aligned} u_{n+1} &= u(x_n) - hu'(x_n) + \frac{h^2}{2}u''(x_n) + O(h^3) + 2hu'(x_n) \\ &= u(x_n) + hu'(x_n) + \frac{h^2}{2}u''(x_n) + O(h^3) \end{aligned} \quad (1-20)$$

将 $u(x_{n+1})$ 在 x_n 处泰勒展开：

$$u(x_{n+1}) = u(x_n) + hu'(x_n) + \frac{h^2}{2}u''(x_n) + O(h^3) \quad (1-21)$$

前两式相减：

$$u(x_{n+1}) - u_{n+1} = O(h^3) \quad (1-22)$$

那么，二步欧拉法的精度为 2 阶。

1.1.3 改进的欧拉法

对式 (1-1) 中的第 1 个式子在 $[x_n, x_{n+1}]$ 上积分，可得：

$$u(x_{n+1}) = u(x_n) + \int_{x_n}^{x_{n+1}} f(x, u) dx \quad (1-23)$$

其中, $n=0, 1, \dots$ 。用不同方式来近似上式中的积分运算, 就会得到不同的递推公式。比如使用左端点计算矩形面积并取近似:

$$\int_{x_n}^{x_{n+1}} f(x, u) dx \approx hf(x_n, u(x_n)) \quad (1-24)$$

然后用 u_n 和 u_{n+1} 近似代替 $u(x_n)$ 和 $u(x_{n+1})$, 则式 (1-23) 变成:

$$u_{n+1} = u_n + hf(x_n, u_n) \quad (1-25)$$

这恰好就是欧拉法。若使用右端点计算矩形面积并取近似:

$$\int_{x_n}^{x_{n+1}} f(x, u) dx \approx hf(x_{n+1}, u(x_{n+1})) \quad (1-26)$$

然后用 u_n 和 u_{n+1} 近似代替 $u(x_n)$ 和 $u(x_{n+1})$, 则:

$$u_{n+1} = u_n + hf(x_{n+1}, u_{n+1}) \quad (1-27)$$

得到的正是隐式欧拉法。若使用梯形的面积做近似:

$$\int_{x_n}^{x_{n+1}} f(x, u) dx \approx \frac{h}{2} [f(x_n, u(x_n)) + f(x_{n+1}, u(x_{n+1}))] \quad (1-28)$$

再用 u_n 和 u_{n+1} 近似代替 $u(x_n)$ 和 $u(x_{n+1})$, 得:

$$u_{n+1} = u_n + \frac{h}{2} [f(x_n, u_n) + f(x_{n+1}, u_{n+1})] \quad (1-29)$$

上式称为梯形公式 (trapezoidal rule)。不难发现, 它其实就是欧拉法和隐式欧拉法的平均。由于用梯形面积近似积分比用矩形面积近似积分的精度要高, 所以梯形公式也要比欧拉法和隐式欧拉法的精度更高, 证明如下:

假设式 (1-29) 等号右边有 $u_n = u(x_n)$ 、 $u_{n+1} = u(x_{n+1})$, 并利用 $u'(x) = f(x, u)$, 得:

$$u_{n+1} = u(x_n) + \frac{h}{2} [u'(x_n) + u'(x_{n+1})] \quad (1-30)$$

将上式中的 $u'(x_{n+1})$ 在 x_n 处泰勒展开:

$$\begin{aligned} u_{n+1} &= u(x_n) + \frac{h}{2} [u'(x_n) + u'(x_n) + hu''(x_n) + O(h^2)] \\ &= u(x_n) + hu'(x_n) + \frac{h^2}{2} u''(x_n) + O(h^3) \end{aligned} \quad (1-31)$$

将 $u(x_{n+1})$ 在 x_n 处泰勒展开:

$$u(x_{n+1}) = u(x_n) + hu'(x_n) + \frac{h^2}{2} u''(x_n) + O(h^3) \quad (1-32)$$

式 (1-32) 与式 (1-31) 相减得:

$$u(x_{n+1}) - u_{n+1} = O(h^3) \quad (1-33)$$

因此，梯形公式的精度为 2 阶。

欧拉法虽然精度偏低，但它是显式的，可直接得到结果。而梯形公式是隐式的，虽然精度较高，却无法通过一步计算得到结果，若用迭代法计算，运算量非常大。综合这两种方法，可以相得益彰：先用显式却低精度的欧拉法计算得到一个粗略的预测值 $\bar{u}_{n+1}$ ，再将这个预测值 $\bar{u}_{n+1}$ 代入梯形公式进行修正，得到较高精度的结果 u_{n+1} 。这就是预测-校正法 (predictor-corrector method)，也称为改进的欧拉法。它的过程可表示如下：

$$\begin{cases} \bar{u}_{n+1} = u_n + hf(x_n, u_n) \\ u_{n+1} = u_n + \frac{h}{2}[f(x_n, u_n) + f(x_{n+1}, \bar{u}_{n+1})] \end{cases} \quad (1-34)$$

式 (1-34) 中，前一步为预测过程，后一步为校正过程。它也可以写成嵌套形式，即显式形式：

$$u_{n+1} = u_n + \frac{h}{2}[f(x_n, u_n) + f(x_{n+1}, u_n + hf(x_n, u_n))] \quad (1-35)$$

或写成平均化的形式：

$$\begin{cases} u_p = u_n + hf(x_n, u_n) \\ u_c = u_n + hf(x_{n+1}, u_p) \\ u_{n+1} = (u_p + u_c)/2 \end{cases} \quad (1-36)$$

显然，它的不同表达式——式 (1-34)、式 (1-35)、式 (1-36) 都是等价的，其精度是 2 阶，证明从略。

1.1.4 龙格-库塔法

将欧拉法写成式 (1-37) 的形式，可以看出它实际上利用了 $f(x, u)$ 在 x_n 处的取值来计算 u_{n+1} 的值。类似地，隐式欧拉法使用了 $f(x, u)$ 在 x_{n+1} 处的近似值来计算 u_{n+1} 的值，而欧拉法和隐式欧拉法的精度都是 1 阶。

$$\begin{cases} u_{n+1} = u_n + k_1 \\ k_1 = hf(x_n, u_n) \end{cases} \quad (1-37)$$

同样，将预测-校正法写成如下形式，不难发现它其实是用 $f(x, u)$ 在 x_n 和 x_{n+1} 两处的取值来计算 u_{n+1} 的值，并达到了 2 阶的精度。

$$\begin{cases} u_{n+1} = u_n + (k_1 + k_2)/2 \\ k_1 = hf(x_n, u_n) \\ k_2 = hf(x_n + h, u_n + k_1) \end{cases} \quad (1-38)$$

上述方法都是通过 $f(x, u)$ 在不同位置的线性组合来计算 u_{n+1} 的值，所考虑的位置越多，

精度也就越高。顺着这样的思路想下去，如果用 $f(x, u)$ 在更多位置的线性组合来构造递推公式，将会得到更高的精度，这就是龙格-库塔法（Runge-Kutta method）的思想。这样，递推公式将有如下形式：

$$\begin{cases} u_{n+1} = u_n + \sum_{i=1}^r R_i k_i \\ k_1 = hf(x_n, u_n) \\ k_i = hf\left(x_n + a_i h, u_n + \sum_{j=1}^{i-1} b_{ij} k_j\right), \quad i = 2, 3, \dots, r \end{cases} \quad (1-39)$$

其中， R_i 、 a_i 、 b_{ij} 为待定常数。由于使用了 $f(x, u)$ 在 r 个位置的取值，称递推公式 (1-39) 为 r 阶龙格-库塔法。为了确定其在某一精度下的待定系数，需要将式中的 u_{n+1} 在 x_n 处泰勒展开，并与 $u(x_{n+1})$ 在 x_n 处的泰勒展开式比较：

$$u(x_{n+1}) = u(x_n) + \frac{h}{1!} u'(x_n) + \frac{h^2}{2!} u''(x_n) + \dots + \frac{h^k}{k!} u^{(k)}(x_n) + O(h^{k+1}) \quad (1-40)$$

若待定常数 R_i 、 a_i 、 b_{ij} 的取值使得两式最多有前 $k+1$ 项相同，那么：

$$u(x_{n+1}) - u_{n+1} = O(h^{k+1}) \quad (1-41)$$

这即是说，此时 R_i 、 a_i 、 b_{ij} 的取值使得式 (1-39) 具有 k 阶的精度，下面举例说明。

当 $r=2$ 时，龙格-库塔法递推公式为：

$$\begin{cases} u_{n+1} = u_n + R_1 k_1 + R_2 k_2 \\ k_1 = hf(x_n, u_n) \\ k_2 = hf(x_n + ah, u_n + bk_1) \end{cases} \quad (1-42)$$

将 k_2 的表达式在 x_n 处展开：

$$\begin{aligned} k_2 &= hf(x_n + ah, u_n + bk_1) \\ &= h[f(x_n, u_n) + ahf_x(x_n, u_n) + bk_1 f_u(x_n, u_n) + O(h^2)] \\ &= hf(x_n, u_n) + h^2[af_x(x_n, u_n) + bf_u(x_n, u_n)f(x_n, u_n)] + O(h^3) \end{aligned} \quad (1-43)$$

将 k_1 和 k_2 代入 u_{n+1} 的表达式：

$$u_{n+1} = u_n + h(R_1 + R_2)f(x_n, u_n) + h^2[aR_2 f_x(x_n, u_n) + bR_2 f_u(x_n, u_n)f(x_n, u_n)] + O(h^3) \quad (1-44)$$

假设 $u_n = u(x_n)$ ，并根据式 (1-1) 中的第 1 个式子，使式 (1-40) 和式 (1-44) 的前 3 项相同，得到：

$$\begin{cases} R_1 + R_2 = 1 \\ aR_2 = 1/2 \\ bR_2 = 1/2 \end{cases} \quad (1-45)$$

满足上式的 R_1 、 R_2 、 a 、 b 可以有无穷多种情况，每种情况的局部截断误差都是 $O(h^3)$ ，都被称为二阶龙格-库塔法。如果取 $R_1=R_2=1/2$ 、 $a=b=1$ ，就是前面提到的预测-校正法。如果取 $R_1=0$ 、 $R_2=1$ 、 $a=b=1/2$ ，就得到中点公式（midpoint method）：

$$\begin{cases} u_{n+1} = u_n + k_2 \\ k_1 = hf(x_n, u_n) \\ k_2 = hf\left(x_n + \frac{1}{2}h, u_n + \frac{1}{2}k_1\right) \end{cases} \quad (1-46)$$

二阶龙格-库塔法不能保证局部截断误差为 $O(h^4)$ ，所以接下来考虑 $r=3$ 的情况，即：

$$\begin{cases} u_{n+1} = u_n + R_1k_1 + R_2k_2 + R_3k_3 \\ k_1 = hf(x_n, u_n) \\ k_2 = hf(x_n + a_2h, u_n + b_{21}k_1) \\ k_3 = hf(x_n + a_3h, u_n + b_{31}k_1 + b_{32}k_2) \end{cases} \quad (1-47)$$

与 $r=2$ 的情况类似，先写出上式中的 u_{n+1} 在 x_n 处的泰勒展开式，并假设 $u_n = u(x_n)$ ，与式（1-40）比较，要保证局部截断误差为 $u(x_{n+1}) - u_{n+1} = O(h^4)$ ，得：

$$\begin{cases} R_1 + R_2 + R_3 = 1 \\ a_2 = b_{21} \\ a_3 = b_{31} + b_{32} \\ a_2R_2 + a_3R_3 = 1/2 \\ a_2^2R_2 + a_3^2R_3 = 1/3 \\ a_2b_{32}R_3 = 1/6 \end{cases} \quad (1-48)$$

方程组（1-48）的解有无穷多个，每组解都确定一个三阶龙格-库塔法的公式。比较简单且重要的一组解是 $R_1=1/6$ 、 $R_2=4/6$ 、 $R_3=1/6$ 、 $a_2=1/2$ 、 $a_3=1$ 、 $b_{21}=1/2$ 、 $b_{31}=-1$ 、 $b_{32}=2$ ，它对应的是库塔公式（Kutta's third-order method），如下：

$$\begin{cases} u_{n+1} = u_n + \frac{1}{6}(k_1 + 4k_2 + k_3) \\ k_1 = hf(x_n, u_n) \\ k_2 = hf\left(x_n + \frac{1}{2}h, u_n + \frac{1}{2}k_1\right) \\ k_3 = hf(x_n + h, u_n - k_1 + 2k_2) \end{cases} \quad (1-49)$$

类似地， $r=4$ 时，龙格-库塔法的递推公式的形式为：

$$\begin{cases} u_{n+1} = u_n + R_1 k_1 + R_2 k_2 + R_3 k_3 + R_4 k_4 \\ k_1 = hf(x_n, u_n) \\ k_2 = hf(x_n + a_2 h, u_n + b_{21} k_1) \\ k_3 = hf(x_n + a_3 h, u_n + b_{31} k_1 + b_{32} k_2) \\ k_4 = hf(x_n + a_4 h, u_n + b_{41} k_1 + b_{42} k_2 + b_{43} k_3) \end{cases} \quad (1-50)$$

在局部截断误差为 $O(h^5)$ 的前提下，得到关于上式中待定常数的方程组的过程比较复杂，这里从略。下面给出两组常用的四阶龙格-库塔法的公式。

标准四阶龙格-库塔公式 (classic fourth-order method):

$$\begin{cases} u_{n+1} = u_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = hf(x_n, u_n) \\ k_2 = hf\left(x_n + \frac{1}{2}h, u_n + \frac{1}{2}k_1\right) \\ k_3 = hf\left(x_n + \frac{1}{2}h, u_n + \frac{1}{2}k_2\right) \\ k_4 = hf(x_n + h, u_n + k_3) \end{cases} \quad (1-51)$$

吉尔 (Gill) 公式:

$$\begin{cases} u_{n+1} = u_n + \frac{1}{6}[k_1 + (2 - \sqrt{2})k_2 + (2 + \sqrt{2})k_3 + k_4] \\ k_1 = hf(x_n, u_n) \\ k_2 = hf\left(x_n + \frac{1}{2}h, u_n + \frac{1}{2}k_1\right) \\ k_3 = hf\left(x_n + \frac{1}{2}h, u_n + \frac{\sqrt{2}-1}{2}k_1 + \frac{2-\sqrt{2}}{2}k_2\right) \\ k_4 = hf\left(x_n + h, u_n - \frac{\sqrt{2}}{2}k_2 + \frac{2+\sqrt{2}}{2}k_3\right) \end{cases} \quad (1-52)$$

需要强调的是， r 的取值并不总是和精度的阶数相同，二者关系如表 1-1 所示。随着 r 的增加，每递推一步的运算量也在增加，对于多数实际问题，四阶龙格-库塔法的精度就已经足够了，因此它也是最常用的龙格-库塔法。

表 1-1 r 与计算精度的关系

r 取值	1	2	3	4	5	6	7	≥ 8
精度阶数	1	2	3	4	4	5	6	$r-2$

1.1.5 ode 系列函数的用法

Matlab 的 ode 系列函数 (统称 odesolver) 专门用于求解形如式 (1-53) 的常微分方程

的初值问题，或者求解形如式（1-54）的常微分方程组的初值问题。ode 是常微分方程（ordinary differential equation）的英文缩写。

$$u' = f(x, u), \quad u(x_0) = u_0 \quad (1-53)$$

$$u'_j = f_j(x, u_1, u_2, \dots, u_n), \quad u_j(x_0) = \eta_j, \quad j = 1, 2, \dots, n \quad (1-54)$$

数值求解常微分方程（组）的初值问题一般需要两个文件：主程序文件（如 main.m）和定义函数 odefun 的文件（即 odefun.m）。简单来讲，主程序文件 main.m 包括变量的定义、初始化、odesolver 的调用、计算结果的处理和显示。而文件 odefun.m 中定义的函数 odefun 用来给出 $u'(x)$ （或 $u'_j(x)$ ）在任一位置的取值，也就是实现 $f(x, u)$ （或 $f_j(x, u_1, u_2, \dots, u_n)$ ）。后面章节的代码示例可帮助读者更好地理解这两个文件的作用和关系。

ode 系列函数的一般调用语法为：

$$[X, U] = \text{odesolver}(\text{odefun}, \text{xspan}, \text{u0}, \text{options}, \text{parameter1}, \text{parameter2}, \dots)$$

下面对每一项进行说明：

（1）“odesolver”是所使用的 ode 系列函数名称，可以是 ode45、ode23、ode113、ode15s、ode23s、ode23t 和 ode23tb 中的一个。

（2）“odefun”是定义了 $f(x, u)$ （或 $f_j(x, u_1, u_2, \dots, u_n)$ ）的函数的句柄，可以是单引号括起来的函数名，也可以是前面加“@”的函数名。

（3）“xspan”为常微分方程（组）自变量 x 的取值范围。它可以是一个二维向量，如“[x0 xn]”。这种情况下，常微分方程（组）将在该区间上被求解，计算后所返回结果是 $u(x)$ （或 $u_j(x)$ ）在这一区间内某些位置的取值，具体位置由 odesolver 自行决定。xspan 也可以是一个 $n+1$ 维向量，如“[x0, x1, ..., xn]”，这样计算后将返回 $u(x)$ （或 $u_j(x)$ ）在向量 xspan 中每个元素所对应位置处的数值结果，注意向量 xspan 中的元素必须按照从大到小或者从小到大的顺序排列。显然，xspan 的第一个元素必然是初始条件的位置 x_0 。

（4）“u0”是一个数值（向量），代表常微分方程（组）初值问题的初始条件，即 $u(x)$ （或 $u_j(x)$ ）在初始位置的取值。当然，初始条件个数等于常微分方程组的方程个数才能求解。

（5）“options”为可选项，它是一个能改变 odesolver 默认参数设置的结构体，由 odeset 函数生成。若不改变默认参数设置，可将其设为空，即“[]”。

（6）“parameter1, parameter2, ...”为传递给 odefun 的参数，个数随意决定，是可选项。

（7）计算结果保存到“[X, U]”中，向量 X 包含一系列位置 $x_0, x_1, \dots, x_n$ ，向量（矩阵）U 包含 $u(x)$ （或 $u_j(x)$ ）在这些位置的数值解。

使用 ode 系列函数时还需要定义 odefun 函数，一般这是一个单独的文件，文件名与函数名相同，即：odefun.m。与 odesolver 的调用形式相对应，该文件开头声明如下：

$$\text{function du} = \text{odefun}(x, u, \text{dummy}, \text{parameter1}, \text{parameter2}, \dots)$$

（1）“odefun”为函数名，可随意选取，但是必须与文件名相同。

（2）“x”为自变量。

(3) “u”是常微分方程(组)的 $u(x)$ (或 $u_i(x)$)在 x 处的取值,是一个数值(向量)。

(4) “dummy”是一个用来占位的变量,无实际意义。如果调用`odesolver`时使用了`options`,就需要在这里加入一个无用的变量与之对应。`dummy`在英语中意为傀儡。

(5) “parameter1,parameter2,...”用于接收传递来的参数,注意前面必须有`dummy`占用一个位置才不会出错。

(6) “du”为函数`odefun`输出结果的变量名,可随意选取。

这里需要特别强调一下,如果调用`odesolver`时需要给函数`odefun`传递参数`parameter1,parameter2,...`,根据语法,这些参数是排在`options`后的,所以在`options`的位置必须有一变量,如无需改变`odesolver`的默认设置,`options`的位置只填入“[]”即可。相应地,函数`odefun`接收参数时也需要在该位置填入一无意义变量`dummy`。换句话说,由于`options`排在参数`parameter1,parameter2,...`之前,所以只要用到传参功能,即使不想通过`options`改变默认设置,也要占用该位置才不致出错。

若既不传递参数也不改变设置,那么调用`odesolver`和定义函数`odefun`的语句为:

$$[X,U] = \text{odesolver}(\text{odefun}, \text{xspan}, u0)$$

和

$$\text{function du} = \text{odefun}(x,u)$$

1.1.4 小节介绍的龙格-库塔法计算公式实际上属于定步长算法,因为它的步长 h 为定值。步长 h 的选取对数值计算有很大影响,过大会导致误差较大,过小则会引起运算量增大以及舍入误差大量积累。这一不足的改进方法就是算法根据误差的情况来自动调整每一步的步长,在方程的解变化快的地方采用小步长,变化慢的地方采用大步长。较之定步长算法,这种变步长算法能更灵活地寻求运算量和误差之间的平衡。

然而,常用的变步长四、五阶龙格-库塔法只适用于非刚性(nonstiff)方程。对于所谓的刚性(stiff)方程,上述变步长算法为了保证精度,会自动采用非常短的步长计算。这就导致计算速度极慢以致无法接受,所以刚性方程由另外的变步长算法来进行数值计算。也就是说,刚性方程和非刚性方程对步长选取的要求存在差异,只能用不同的变步长算法来求解。

顾名思义,`ode45`使用的是变步长的四、五阶龙格-库塔法,它的精度适中,可用于求解大多数常微分方程(组),是`odesolver`中的首选。`ode23`采用变步长的二、三阶龙格-库塔算法,它的精度略差,但比`ode45`的计算速度更快。`ode23`和`ode45`都属于单步算法,而`ode113`用了Adams-Bashforth-Moulton多步算法,它的精度和速度在某些情况下会比`ode45`更高。前面提到的这三种`odesolver`都仅适用于非刚性方程(组)。针对刚性方程(组)只能用`ode15s`、`ode23s`、`ode23t`、`ode23tb`。如果不能判断常微分方程(组)是否刚性,优先推荐选用`ode45`求解,遇到运行失败或者迟迟不见输出结果的情况,再用`ode15s`,这样可以处理绝大多数的常微分方程(组)。仍不行的话就尝试其他的`odesolver`。表1-2给出了`ode`系列函数的基本介绍。

表 1-2 ode 系列函数简介

函 数	处理问题类型	精 度	描 述
ode45	非刚性	中	大多数情况下的首选
ode23	非刚性	低	处理对精度要求不高或中等刚性的问题
ode113	非刚性	低中高	处理某些问题比 ode45 更精确更高效
ode15s	刚性	低中	ode45 失效的时候优先尝试
ode23s	刚性	低	处理对精度要求不高的刚性问题
ode23t	中等刚性	低	处理中等刚性问题
ode23tb	刚性	低	处理对精度要求不高的刚性问题

下面以初值问题（1-55）为例，示范几种数值方法的代码实现。

$$\begin{cases} u' = -3u + 6x + 5 \\ u(0) = 3 \end{cases} \quad (1-55)$$

上式的解析解为： $u=2e^{-3x}+2x+1$ 。这里分别用欧拉法、预测-校正法、ode45 求解该问题，并分别计算三种方法的数值解与解析解的误差。代码如下：

程序 1-1

```
clear all; close all;
L=1; h=0.05; x=0:h:L;
u_Euler=zeros(length(x),1); u_Euler(1)=3;
u_pc=u_Euler; u_ode45=u_Euler;
for n=1:length(x)-1
    %欧拉法
    u_Euler(n+1)=u_Euler(n)+h*(-3*u_Euler(n)+6*x(n)+5);
    %预测-校正法
    k1=h*(-3*u_pc(n)+6*x(n)+5);
    k2=h*(-3*(u_pc(n)+k1)+6*(x(n)+h)+5);
    u_pc(n+1)=u_pc(n)+(k1+k2)/2;
end
%ode45
[x,u_ode45]=ode45(@(x,u)[-3*u+6*x+5],x,u_ode45(1));
%解析解
u_exact=2*exp(-3*x)+2*x+1;
%画图
plot(x,u_Euler,'xk',x,u_pc,'ok',x,u_ode45,'+k',x,u_exact,'k','MarkerSize',10,'LineWidth',1.5)
axis([0 1 2.3 3.15]), xlabel x, ylabel u, set(gca,'FontSize',18)
legend('欧拉法','预测-校正法','ode45','解析解','location','North')
```

```
%三种算法的误差
Error(:,1)=abs(u_Euler-u_exact);
Error(:,2)=abs(u_pc-u_exact);
Error(:,3)=abs(u_ode45-u_exact);
```

在调用 ode45 时，因为表达式 $u' = -3u + 6x + 5$ 比较简单，所以没有单独新建一个函数文件，而是使用了匿名函数（anonymous function）。定义匿名函数的语法为：

`fhandle = @(arglist) expr`

“fhandle”为函数句柄。“arglist”为该函数的参数列表，多个参数以逗号隔开。“expr”为该函数的表达式。代码中的“`@(x,u)[-3*u+6*x+5]`”与新建一个 odefun.m 文件，里面填上如下内容是等效的。

```
function du=odefun(x,u)
du=-3*u+6*x+5;
```

使用匿名函数可以减少一个函数文件，但仅针对函数表达式很简单的情況。由于本书后面所解决的问题都比较复杂，所以其余代码都将使用创建函数文件的方法。运行代码得到结果如图 1-2 所示。可以看到，在步长同为 $h=0.05$ 的情况下，欧拉法的数值结果明显偏离了解析解，而预测-校正法、ode45 的数值结果与解析解重合得较好。

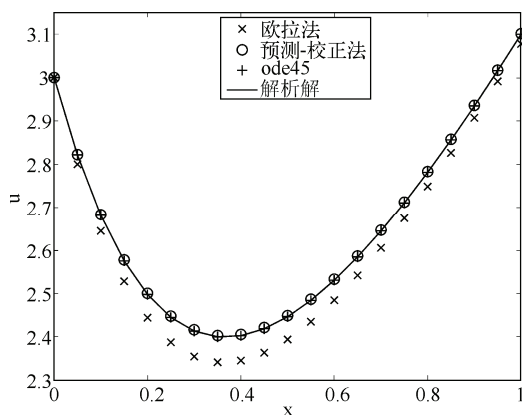


图 1-2 用欧拉法、预测-校正法、ode45 求出数值解，并与解析解相比较

表 1-3 列出了三种方法在每个位置所得数值结果以及误差。显然，三者误差上的关系有：欧拉法 > 预测-校正法 > ode45，这与此前局部截断误差的分析结果是一致的。

表 1-3 欧拉法、预测-校正法、ode45 的数值结果与解析解的比较

位置	欧 拉 法		预测-校正法		ode45		解 析 解
	u_n	$ u_n - u(x_n) $	u_n	$ u_n - u(x_n) $	u_n	$ u_n - u(x_n) $	$u(x_n)$
x_1	2.8	0.021416	2.8225	0.001084	2.821416	6.99E-09	2.821416
x_2	2.645	0.036636	2.683503	0.001867	2.681634	2.81E-06	2.681636

(续)

位 置	欧 拉 法		预测-校正法		ode45		解 析 解
	u_n	$ u_n - u(x_n) $	u_n	$ u_n - u(x_n) $	u_n	$ u_n - u(x_n) $	$u(x_n)$
x_3	2.52825	0.047006	2.577667	0.002411	2.575257	4.26E-07	2.575256
x_4	2.444013	0.053611	2.500391	0.002767	2.497622	1.72E-06	2.497623
x_5	2.387411	0.057322	2.447712	0.002978	2.444734	6.27E-07	2.444733
x_6	2.354299	0.05884	2.416217	0.003077	2.413138	1E-06	2.413139
x_7	2.341154	0.058721	2.402967	0.003091	2.399876	6.95E-07	2.399875
x_8	2.344981	0.057407	2.40543	0.003041	2.402388	5.45E-07	2.402388
x_9	2.363234	0.055247	2.421427	0.002946	2.418481	6.86E-07	2.418481
x_{10}	2.393749	0.052512	2.449079	0.002818	2.44626	2.57E-07	2.44626
x_{11}	2.434686	0.049413	2.486769	0.002669	2.4841	6.35E-07	2.4841
x_{12}	2.484484	0.046114	2.533105	0.002507	2.530598	8.12E-08	2.530598
x_{13}	2.541811	0.042737	2.586886	0.002338	2.584549	5.64E-07	2.584548
x_{14}	2.605539	0.039374	2.647081	0.002168	2.644913	2.06E-08	2.644913
x_{15}	2.674708	0.03609	2.712798	0.002	2.710799	4.87E-07	2.710798
x_{16}	2.748502	0.032934	2.783273	0.001837	2.781436	7.51E-08	2.781436
x_{17}	2.826227	0.029936	2.857844	0.00168	2.856164	4.13E-07	2.856163
x_{18}	2.907293	0.027118	2.935943	0.001532	2.934411	1E-07	2.934411
x_{19}	2.991199	0.02449	3.017081	0.001392	3.015689	3.44E-07	3.015689
x_{20}	3.077519	0.022055	3.100836	0.001262	3.099574	2.96E-07	3.099574

接下来再给出一个数值求解刚性方程组的例子。van der Pol 方程是 Matlab 帮助文件提到的一个很典型的刚性问题，表达式可写成如下形式：

$$\begin{cases} u_1' = u_2, & u_1(0) = 2 \\ u_2' = 1000(1 - u_1^2)u_2 - u_1, & u_2(0) = 0 \end{cases} \quad (1-56)$$

用专门处理刚性方程的 ode15s 在区间[0, 3000]上数值求解 $u_1(x)$ ，代码如下：

程序 1-2

主程序代码如下：

```
clear all; close all;
[T,U]=ode15s(@vdp,[0 3000],[2 0]);
plot(T,U(:,1),'k','LineWidth',1.5)
xlabel x, ylabel u_1
```

文件 vdp.m 代码如下：

```
function du=vdp(x,u)
du=[u(2); 1000*(1-u(1)^2)*u(2)-u(1)];
```

`vdp.m` 文件中定义的 `vdp` 函数将 u'_1 、 u'_2 的值放在一个二维向量中输出。此外，存放计算结果的变量 `U` 是一个两列的矩阵，分别对应于 u_1 和 u_2 的数值结果。这是求解具有两个方程的方程组与求解一个单独方程的差别。程序输出如图 1-3 所示，可以看到 u_1 在某些位置的变化十分剧烈。如果将代码中的 `ode15s` 换成针对非刚性问题的 `ode45`，CPU 占用率将长时间维持在 100%，并且迟迟得不到计算结果；而这对于 `ode15s` 只需要一瞬间，这说明处理刚性/非刚性方程时采用正确的方法可以事半功倍。

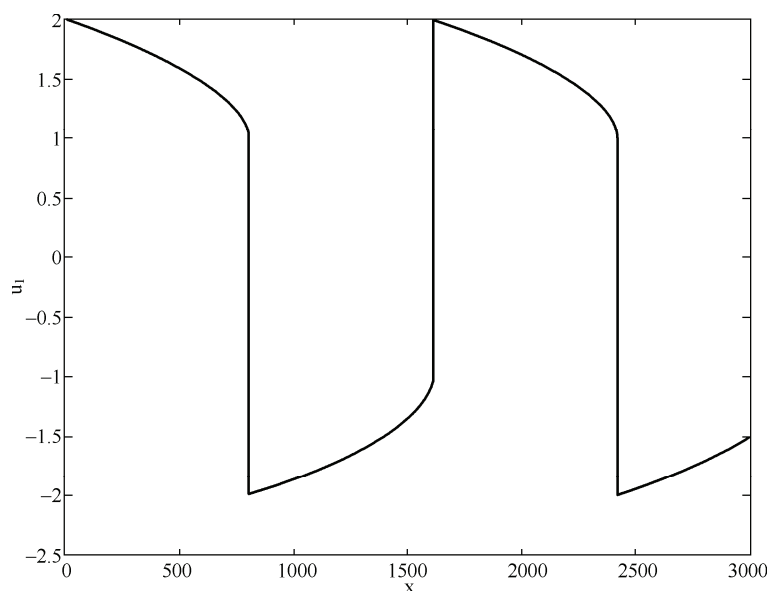


图 1-3 刚性问题 van der Pol 方程的数值解

1.1.6 高阶微分方程的降阶

在初值问题中，`odesolver` 只能用来数值求解一阶微分方程（组）。对于高阶微分方程，可以将其化成一阶微分方程组，进而用 `odesolver` 求解，过程如下。

有 n 阶微分方程：

$$\frac{d^n u}{dx^n} = f\left(x, u, \frac{du}{dx}, \dots, \frac{d^{n-1}u}{dx^{n-1}}\right) \quad (1-57)$$

其在 x_0 处的初始条件为：

$$\begin{cases} u(x_0) = \eta_1 \\ u'(x_0) = \eta_2 \\ \vdots \\ u^{(n-1)}(x_0) = \eta_n \end{cases} \quad (1-58)$$

做变量代换：

$$\begin{cases} u_1 = u \\ u_2 = u' \\ \vdots \\ u_n = u^{(n-1)} \end{cases} \quad (1-59)$$

则 n 阶微分方程可写为一阶微分方程组：

$$\begin{cases} u_1' = u_2 \\ u_2' = u_3 \\ \vdots \\ u_{n-1}' = u_n \\ u_n' = f(x, u_1, u_2, \dots, u_n) \end{cases} \quad (1-60)$$

相应初始条件为：

$$\begin{cases} u_1(x_0) = \eta_1 \\ u_2(x_0) = \eta_2 \\ \vdots \\ u_n(x_0) = \eta_n \end{cases} \quad (1-61)$$

到这一步就可利用 `odesolver` 直接对其求解了，后面章节在解决高阶微分方程的初值问题时将反复应用此原理。

下面以阻尼振动问题为例：

$$u'' + 2\beta u' + \omega^2 u = 0, \quad u(0) = 1, \quad u'(0) = 0 \quad (1-62)$$

这是一个二阶常微分方程的初值问题，它描述了一个弹簧振子或单摆在振动过程中，由于阻力的作用，振幅逐渐减小的现象，其中 β 、 ω 为常数，取 $\beta=1$ 、 $\omega=10$ 。先做变量代换：

$$\begin{cases} u_1 = u \\ u_2 = u' \end{cases} \quad (1-63)$$

则式 (1-62) 变为：

$$\begin{cases} u_1' = u_2, & u_1(0) = 1 \\ u_2' = -2\beta u_2 - \omega^2 u_1, & u_2(0) = 0 \end{cases} \quad (1-64)$$

接下来用 `ode45` 数值求解，代码如下：

程序 1-3

主程序代码如下：

```
clear all; close all;
beta=1; omega=10; xspan=[0 5];
```

```
[x,usol]=ode45('damp',xspan,[1 0],[1],beta,omega);
plot(x,usol(:,1),'k','LineWidth',1.5)
xlabel x, ylabel u
```

文件 damp.m 代码如下：

```
function du=damp(t,u,dummy,beta,omega)
du=[u(2); -2*beta*u(2)-omega^2*u(1)];
```

根据初始条件和方程的物理意义，振子在初始时刻的位移 u 为 1、速度 u' 为 0。可以预测，在随后的振动过程中，振子克服阻力做功，振幅逐渐变小，这与程序的输出图 1-4 是吻合的。

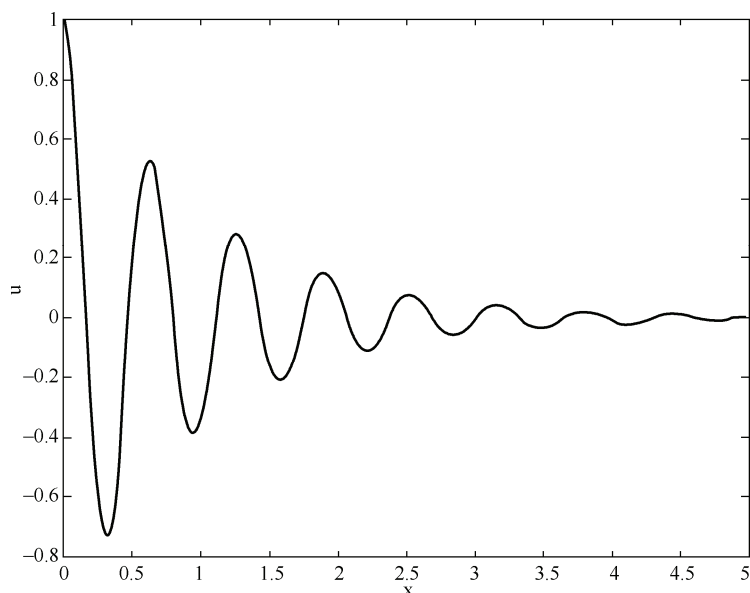


图 1-4 阻尼振动问题的数值解

1.2 边值问题

前面分析的是初始条件下常微分方程（组）的数值求解，但在某些实际问题中，已知的是边界上的条件（即边界条件）而非初始条件，如：梁的形变问题、激光器谐振腔的理论模型、可转化为边值问题的特征值问题等。本小节讨论典型的二阶常微分方程的边值问题：

$$\frac{d^2 u}{dx^2} = f\left(x, u, \frac{du}{dx}\right), \quad a < x < b \quad (1-65)$$

其边界条件为 $(\alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2)$ 均为常数)：

$$\begin{cases} \alpha_1 u(a) + \beta_1 u'(a) = \gamma_1 \\ \alpha_2 u(b) + \beta_2 u'(b) = \gamma_2 \end{cases} \quad (1-66)$$

上式具有很强的—般性，这里先从一种简单的边界条件开始分析（ α 、 β 为常数）：

$$u(a) = \alpha, \quad u(b) = \beta \quad (1-67)$$

1.2.1 打靶法

对于二阶常微分方程（1-65）及边界条件（1-67），利用代换 $u_1 = u$ 、 $u_2 = u'$ ，可将其写为等价的方程组形式：

$$\begin{cases} u_1' = u_2, & u_1(a) = \alpha \\ u_2' = f(x, u_1, u_2), & u_1(b) = \beta \end{cases} \quad (1-68)$$

它与初值问题的不同之处在于， $u_2(a)$ 的值是未知的，已知条件中取而代之的是 $u_1(b)$ 的值，否则就可以直接通过 `odesolver` 数值求解了。假如可以确定一个值 m ，使初值问题（1-69）的解 $u_1(x)$ 满足 $|u_1(b) - \beta| < \varepsilon$ ，其中 ε 为可允许的误差，那么边值问题（1-68）和初值问题（1-69）的解就是近似相等的，也就相当于把边值问题转化成了初值问题。

$$\begin{cases} u_1' = u_2, & u_1(a) = \alpha \\ u_2' = f(x, u_1, u_2), & u_2(a) = m \end{cases} \quad (1-69)$$

打靶法就是用来确定 m 值的，大致思路如下：

- （1）随意选取 m 的起始值。
- （2）求出初值问题（1-69）的解 $u_1(x)$ 。
- （3）若不符合 $|u_1(b) - \beta| < \varepsilon$ ，就修正 m 的值。
- （4）重复前面两项，直到符合 $|u_1(b) - \beta| < \varepsilon$ ，此时的 $u_1(x)$ 就是数值解。

修正 m 取值的方法可以是线性插值法、牛顿法等，这里使用比较简单的线性插值法。设 $m = m_n$ 时初值问题（1-69）的解 $u_1(x)$ 在 $x = b$ 处有 $u_1(b) = \beta_n$ ，其中 $n = 1, 2, \dots$ 。若假设 m_n 与 β_n 是线性关系，为了让 m 取最新修正值 m_{n+2} 时有 $\beta_{n+2} = \beta$ ，则 m_{n+2} 需要满足 $(m_{n+2} - m_n) / (\beta - \beta_n) = (m_{n+1} - m_n) / (\beta_{n+1} - \beta_n)$ ，即：

$$m_{n+2} = m_n + \frac{m_{n+1} - m_n}{\beta_{n+1} - \beta_n} (\beta - \beta_n) \quad (1-70)$$

其中的 m_n 、 β_n 、 m_{n+1} 、 β_{n+1} 为已知，这就是说如果使用线性插值法修正 m 的取值，在打靶法开始时需要先随意选取 2 个 m 的起始值 m_1 、 m_2 ，计算出相应的 β_1 、 β_2 ，才可以根据上式得到修正值 m_3 ，进而得到 β_3 ，然后得到修正值 $m_4 \dots \dots$ 直到找到满足 $|\beta_n - \beta| < \varepsilon$ 的 m 值。此时初值问题（1-69）的解即近似为边值问题（1-68）的解，而初值问题是容易数值求解的。

打靶法的过程如图 1-5 所示, 虽然 m 取 m_1 、 m_2 时, 最终结果在 $x=b$ 处的取值与应有的边界条件相差甚远, 但通过一次又一次地修正, 逐渐缩小距离目标的差距, 最终总会得到在一定误差内满足 $x=b$ 处边界条件的解。

对于更复杂的边界条件 (1-66), 照样可以用打靶法解决。尽管 $u(a)$ 和 $u'(a)$ 的值都是未知的, 但只要猜测它们其中一个的取值, 就可以通过关系 $\alpha_1 u(a) + \beta_1 u'(a) = \gamma_1$ 得到另外一个的值。有了 $u(a)$ 和 $u'(a)$ 的值, 再通过 `odesolver` 求解式 (1-65) 得到 $u(b)$ 和 $u'(b)$ 的值 (当然需要变量代换 $u_1=u$ 、 $u_2=u'$), 然后使用 $|\alpha_2 u(b) + \beta_2 u'(b) - \gamma_2| < \varepsilon$ 判断猜测的初始值是否符合要求, 如失败就修正猜测值后继续尝试, 直到符合要求。这样一来, 同样可以将边界条件 (1-66) 下的边值问题当做初值问题处理。

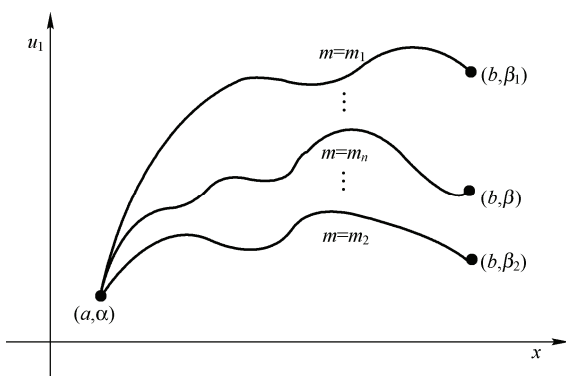


图 1-5 打靶法示意图

下面举一个例子, 考虑如下二阶常微分方程:

$$u'' + |u| = 0, \quad 0 < x < 4 \quad (1-71)$$

其边界条件为:

$$u(0) = 0, \quad u(4) = -2 \quad (1-72)$$

做代换 $u_1=u$ 、 $u_2=u'$, 写为一阶常微分方程组形式:

$$\begin{cases} u_1' = u_2, & u_1(0) = 0 \\ u_2' = -|u_1|, & u_1(4) = -2 \end{cases} \quad (1-73)$$

将此问题转化为初值问题, 就需要找到一个 $m=m_n$, 使式 (1-74) 的解 $u_1(x)$ 在 $x=4$ 处的值 $u_1(4)=\beta_n$ 满足 $|\beta_n+2|<10^{-6}$ 。

$$\begin{cases} u_1' = u_2, & u_1(0) = 0 \\ u_2' = -|u_1|, & u_2(0) = m \end{cases} \quad (1-74)$$

根据打靶法思想, 确定 m 取值的流程如图 1-6 所示, 相应代码如下:

程序 1-4

主程序代码如下：

```
clear all; close all;
n=0; m(1)=1; alpha=0; beta0=-2;
while n==0||abs(beta(n)-beta0)>=1e-6
    n=n+1;
    %在猜测的初始条件下求解初值问题
    [x,usol]=ode45('shoot',[0 4],[alpha m(n)]);
    beta(n)=usol(end,1);
    if n==1
        m(2)=m(1)-0.1;
    else
        %线性插值法修正 m 的取值
        m(n+1)=m(n-1)+(m(n)-m(n-1))*(beta0-beta(n-1))/(beta(n)-beta(n-1));
    end
end
plot(x,usol(:,1),'k','LineWidth',1.5), xlabel x, ylabel u
```

文件 shoot.m 代码如下：

```
function du=shoot(x,u)
du=[u(2); -abs(u(1))];
```

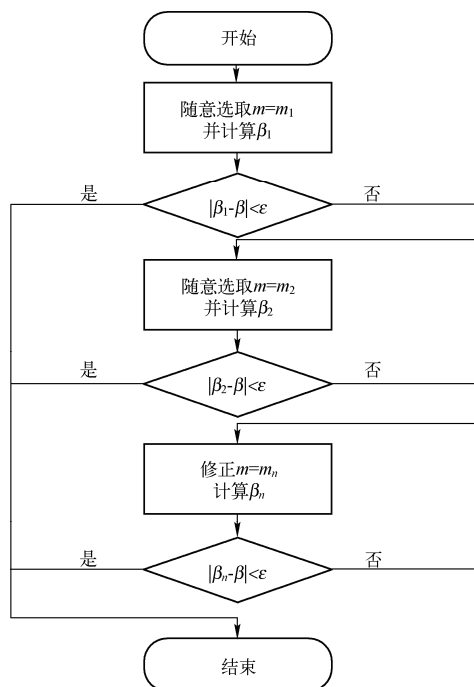


图 1-6 打靶法流程图

代码中使用的 m 的前两个猜测值为 1 和 0.9，程序仅经过了 5 次循环，就满足条件 $|\beta_5+2|<10^{-6}$ ，得到结果如图 1-7 所示。如选取 m 的前两个猜测值为 -1 和 -1.1，还会得到另外一个截然不同的解。因为边值问题的解往往不是唯一的，所求得解与猜测值的选取有关，这是边值问题与初值问题的一个很大不同。

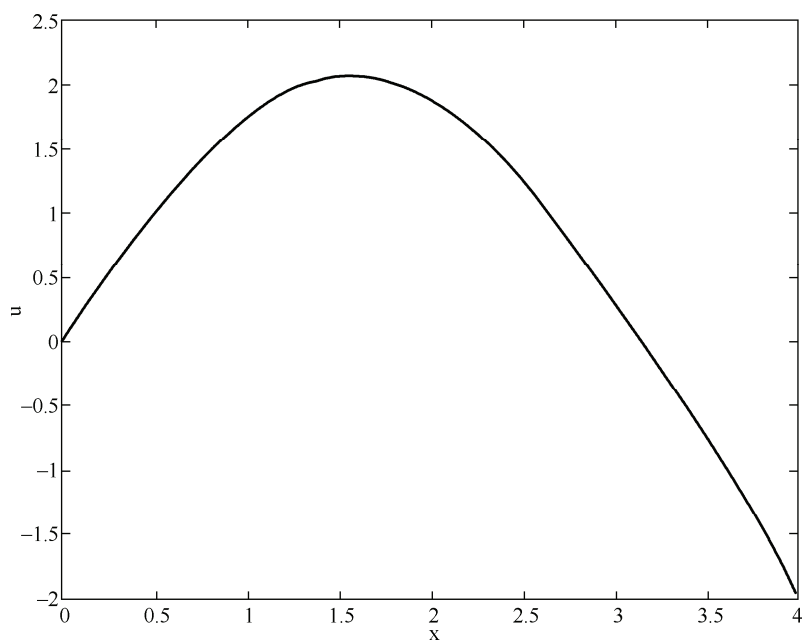


图 1-7 边值问题的解

查看程序运行后 m_n 和 β_n 的前 5 个值，可发现线性插值法的修正效率还比较令人满意。

```
>> m(1:5)
```

```
ans =
```

```
1.0000    0.9000    2.0480    2.0640    2.0641
```

```
>> beta(1:5)
```

```
ans =
```

```
-0.9699   -0.8716   -1.9845   -1.9999   -2.0000
```

1.2.2 bvp 系列函数的用法

与专门针对初值问题的 ode 系列函数类似, Matlab 中还有专门用于解决边值问题的 bvp 系列函数, 其调用语法为:

$$\text{sol} = \text{bvpsolver}(\text{odefun}, \text{bcfun}, \text{solinit}, \text{options})$$

下面对每一项进行说明:

(1) “bvpsolver” 是 bvp 系列函数的名称, 可以是 bvp4c 或 bvp5c, 后者比前者的精度高一些。

(2) “odefun” 是包含常微分方程组的函数句柄, 这与 ode 系列函数中的 odefun 是一样的。

(3) “bcfun” 是包含边界条件的函数句柄。

(4) “solinit” 是包含初始网格结点、初始猜测值的结构体, 由 bvpinit 函数生成。

(5) “options” 用于改变 bvpsolver 的默认设置, 由 bvpset 函数生成, 为可选项。

(6) “sol” 为包含计算结果的结构体。

如不需要用到 options, 则 bvp 系列函数的调用形式为:

$$\text{sol} = \text{bvpsolver}(\text{odefun}, \text{bcfun}, \text{solinit})$$

用于生成结构体 solinit 的 bvpinit 函数的调用语法为:

$$\text{solinit} = \text{bvpinit}(\mathbf{x}, \mathbf{uinit}, \text{params})$$

其中:

(1) “x” 为计算区间内的初始网格结点, 起止点即为边界条件所在的两处位置, 这些结点需要按大小单调排列。

(2) “uinit” 是解的初始猜测值, 也可以是每个结点处的解的猜测值。

(3) “params” 是边值问题中的未知参数 (如果有的话) 的猜测值, 为可选项。

下面举例说明 bvp4c 的用法, 边值问题如下:

$$\begin{cases} u'' = -u + 2(x-1)e^{-x}, & 0 < x < \pi \\ u(0) = 1, & u(\pi) = \pi/e^\pi - 1 \end{cases} \quad (1-75)$$

先利用代换 $u_1 = u$ 、 $u_2 = u'$, 化为一阶常微分方程组形式:

$$\begin{cases} u_1' = u_2, & u_1(0) = 1 \\ u_2' = -u_1 + 2(x-1)e^{-x}, & u_1(\pi) = \pi/e^\pi - 1 \end{cases} \quad (1-76)$$

将此常微分方程组和边界条件分别写进 shoot2.m 和 bc.m 中，用 bvp4c 求解，代码如下：

程序 1-5

主程序代码如下：

```
clear all; close all;
x=linspace(0,pi,20);
solinit=bvpinit(x,[2 2]);
%bvp4c 解边值问题
sol=bvp4c(@shoot2,@bc,solinit);
%从结构体 sol 获取指定位置的数值结果
u=deval(sol,x);
%解析解
u_exact=cos(x)+x.*exp(-x);
%画图
plot(x,u(1,:),'+k',x,u_exact,'k','MarkerSize',10,'LineWidth',1.5)
set(gca,'FontSize',16), xlabel x, ylabel u
legend('bvp4c','解析解',0), axis([0 pi -1 1.5])
```

文件 shoot2.m 代码如下：

```
function du=shoot2(x,u)
du=[u(2); -u(1)+2*(x-1)*exp(-x)];
```

文件 bc.m 代码如下：

```
function res=bc(ua,ub)
%边界条件
res=[ua(1)-1; ub(1)+1-pi/exp(pi)];
```

这里重点说明一下包含边界条件的文件 bc.m，其中定义了一个同名函数 bc，参数 ua 和 ub 分别代表左边界和右边界的情况。其中，ua(1)和 ua(2)为函数在左边界的取值和一阶导数，ub(1)和 ub(2)为函数在右边界的取值和一阶导数。代码中的“ua(1)-1”就是 $u(0)-1=0$ 的简写，代表边界条件 $u(0)=1$ ，同理，“ub(1)+1-pi/exp(pi)”表示 $u(\pi)=\pi/e-1$ 。

代码中的初始猜测值为 $u_1=2$ 、 $u_2=2$ ，假如边值问题的解是唯一的，初始猜测值的选取是不会影响结果的。图 1-8 同时显示了 bvp4c 得到的数值解和式 (1-75) 的解析解 $u=\cos(x)+xe^{-x}$ ，二者相当吻合。值得一提的是，如果在 bvpinit 函数中选取的初始网格结点数量偏少的话，会造成数值解和解析解存在一定的差异，在这一点上，bvp4c 比打靶法更难使用。

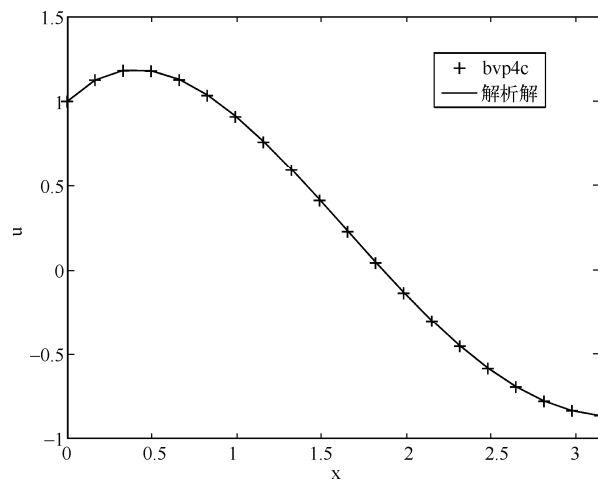


图 1-8 边值问题的数值解与解析解

第 2 章 有限差分法和有限元法

有限差分法 (finite difference method) 和有限元法 (finite element method) 是两种历史悠久且十分成熟的求解偏微分问题的方法, 相关书籍众多。本章只介绍其简单的应用, 一来帮助读者了解数值方法的发展过程, 二来可供读者将这两种方法与后续章节中的谱方法做比较。

2.1 有限差分法

2.1.1 有限差分法中的数值微分

数值微分是根据函数在一些离散点的值计算它在某点的 1 阶导数或高阶导数近似值的方法。先从一个最简单的例子开始, 在区间 $[a, b]$ 上取一系列间距为 h 的点: $x_0, x_1, \dots, x_N$, 如图 2-1 所示。

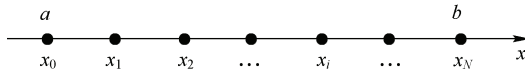


图 2-1 取区间上等间距的离散点

要利用函数 $u(x)$ 在这些点上的取值 $u(x_i)$ 来计算 $u'(x_i)$ 和 $u''(x_i)$ 的近似值。最直接的办法就是通过泰勒公式将 $u(x_{i+1})$ 在 $x=x_i$ 处展开, 整理可得:

$$u'(x_i) = \frac{1}{h}[u(x_{i+1}) - u(x_i)] - \frac{h}{2}u''(\xi_i), \quad x_i < \xi_i < x_{i+1} \quad (2-1)$$

将上式右端的小量 $O(h)$ 作为局部截断误差忽略掉, 可得计算 $u'(x_i)$ 的公式 (2-2)。由于 x_{i+1} 在 x_i 的前面, 所以称之为 1 阶导数的向前差商。

$$u'(x_i) = \frac{1}{h}[u(x_{i+1}) - u(x_i)], \quad 0 \leq i \leq N-1 \quad (2-2)$$

注意上式中的 i 是不能取到 N 的, 除非还有条件给出 $u(x_N+h)$ 的值。或者, 对于周期为 $b-a+h$ 的周期函数 $u(x)$, 可得 $u(x_N+h)=u(x_0)$, 那么 $i=N$ 时有 $u'(x_N)=[u(x_0)-u(x_N)]/h$ 。另外, 式 (2-2) 的局部截断误差为 $O(h)$, 从理论上讲, h 越小则精度越高, 但实际上当 h 过小时, $u(x_{i+1})$ 和 $u(x_i)$ 将是两个十分接近的数, 它们相减会造成有效数字的损失, 引起误差增加。所以 h 的选取不可过大或过小。

同理, 还可以得到 1 阶导数的向后差商:

$$u'(x_i) = \frac{1}{h}[u(x_i) - u(x_{i-1})], \quad 1 \leq i \leq N \quad (2-3)$$

类似地，分别将 $u(x_{i+1})$ 和 $u(x_{i-1})$ 在 $x=x_i$ 处做泰勒展开，相减并整理，得：

$$u'(x_i) = \frac{1}{2h}[u(x_{i+1}) - u(x_{i-1})] - \frac{h^2}{6}u^{(3)}(\xi_i), \quad x_{i-1} < \xi_i < x_{i+1} \quad (2-4)$$

忽略小量 $O(h^2)$ ，就是 1 阶导数的中心差商公式，它的精度要高于向前差商和向后差商：

$$u'(x_i) = \frac{1}{2h}[u(x_{i+1}) - u(x_{i-1})], \quad 1 \leq i \leq N-1 \quad (2-5)$$

此外，分别将 $u(x_{i+1})$ 和 $u(x_{i-1})$ 在 $x=x_i$ 处做泰勒展开，相加并整理，得：

$$u''(x_i) = \frac{1}{h^2}[u(x_{i-1}) - 2u(x_i) + u(x_{i+1})] - \frac{h^2}{12}u^{(4)}(\xi_i), \quad x_{i-1} < \xi_i < x_{i+1} \quad (2-6)$$

忽略小量 $O(h^2)$ ，即为 2 阶导数的中心差商公式：

$$u''(x_i) = \frac{1}{h^2}[u(x_{i-1}) - 2u(x_i) + u(x_{i+1})], \quad 1 \leq i \leq N-1 \quad (2-7)$$

2.1.2 求导的矩阵形式

2.1.1 小节给出了数值微分的原理及公式，要在 Matlab 中实现这样的数值微分，既可以用循环语句，也可以用矩阵乘法的形式。众所周知，Matlab 在矩阵运算方面的速度比较快，而且矩阵形式的代码也更加简洁明了，所以接下来将数值微分写成矩阵的形式。

(1) 若 $u(x)$ 为周期函数且周期为 $b-a+h$ 。

1 阶导数的向前差商公式可写为：

$$\begin{pmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \frac{1}{h} \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} \quad (2-8)$$

1 阶导数的向后差商公式可写为：

$$\begin{pmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \frac{1}{h} \begin{pmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} \quad (2-9)$$

1 阶导数的中心差商公式可写为：

$$\begin{pmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \frac{1}{2h} \begin{pmatrix} 0 & 1 & & -1 \\ -1 & 0 & 1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 0 & 1 \\ 1 & & & -1 & 0 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} \quad (2-10)$$

2 阶导数的中心差商公式可写为:

$$\begin{pmatrix} u''(x_0) \\ u''(x_1) \\ \vdots \\ u''(x_{N-1}) \\ u''(x_N) \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & 1 \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} \quad (2-11)$$

(2) 若有条件给出 $u(x_0-h)$ 和 (或) $u(x_N+h)$ 的值。

1 阶导数的向前差商公式可写为:

$$\begin{pmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \frac{1}{h} \begin{pmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1 & 1 \\ & & & & -1 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} + \frac{1}{h} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ u(x_N+h) \end{pmatrix} \quad (2-12)$$

1 阶导数的向后差商公式可写为:

$$\begin{pmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \frac{1}{h} \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 & 1 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} - \frac{1}{h} \begin{pmatrix} u(x_0-h) \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad (2-13)$$

1 阶导数的中心差商公式可写为:

$$\begin{pmatrix} u'(x_0) \\ u'(x_1) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \frac{1}{2h} \begin{pmatrix} 0 & 1 & & \\ -1 & 0 & 1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} + \frac{1}{2h} \begin{pmatrix} -u(x_0-h) \\ 0 \\ \vdots \\ 0 \\ u(x_N+h) \end{pmatrix} \quad (2-14)$$

2 阶导数的中心差商公式可写为:

$$\begin{pmatrix} u''(x_0) \\ u''(x_1) \\ \vdots \\ u''(x_{N-1}) \\ u''(x_N) \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} + \frac{1}{h^2} \begin{pmatrix} u(x_0 - h) \\ 0 \\ \vdots \\ 0 \\ u(x_N + h) \end{pmatrix} \quad (2-15)$$

下面用实例说明周期函数 $u(x)=e^{\sin(\pi x)}$ 的 1 阶、2 阶导数的数值计算过程，并且与相应的解析解 $u'(x)=\pi \cos(\pi x)e^{\sin(\pi x)}$ 、 $u''(x)=\pi^2 e^{\sin(\pi x)}[\cos^2(\pi x)-\sin(\pi x)]$ 做比较，代码如下。程序输出结果如图 2-2 所示，误差最大的是 1 阶导数的向前差商公式，而 1 阶、2 阶导数的中心差商公式则更精确，这是由局部截断误差决定的。

程序 2-1

```
clear all; close all;
L=2; N=50; h=L/N;
x=h*[-N/2:N/2-1]';
u=exp(sin(pi*x));
%导数的精确解
du_exact(:,1)=pi*cos(pi*x).*u;
du_exact(:,2)=pi*cos(pi*x).*u;
du_exact(:,3)=pi^2*(cos(pi*x).^2-sin(pi*x)).*u;
%1 阶向前差商
e=ones(N,1);
D1_forword=spdiags([-e e],[0 1],N,N);
D1_forword(N,1)=1;
du(:,1)=1/h*D1_forword*u;
%1 阶中心差商
D1_center=spdiags([-e e],[-1 1],N,N);
D1_center(1,N)=-1; D1_center(N,1)=1;
du(:,2)=1/(2*h)*D1_center*u;
%2 阶中心差商
D2=spdiags([e -2*e e],[-1 0 1],N,N);
D2(1,N)=1; D2(N,1)=1;
du(:,3)=1/(h^2)*D2*u;
%误差
error=max(abs(du-du_exact));
%画图
labels={'u'(x)','u''(x)','u'''(x)'};
titles={'1 阶向前差商','1 阶中心差商','2 阶中心差商'};
```

```

for n=1:4
    subplot(2,2,n)
    if n==1
        plot(x,u,'k','LineWidth',1.5)
        xlabel x, ylabel u(x), title('u(x)=e^{sin(\pi x)}')
    else
        plot(x,du_exact(:,n-1),'k',x,du(:,n-1),'r','MarkerSize',13,'LineWidth',1.5)
        title(['Error_{max}=' num2str(error(n-1))])
        xlabel x, ylabel(labels(n-1))
        text(-1,min(du(:,n-1)),titles(n-1),'FontWeight','bold')
    end
end
end

```

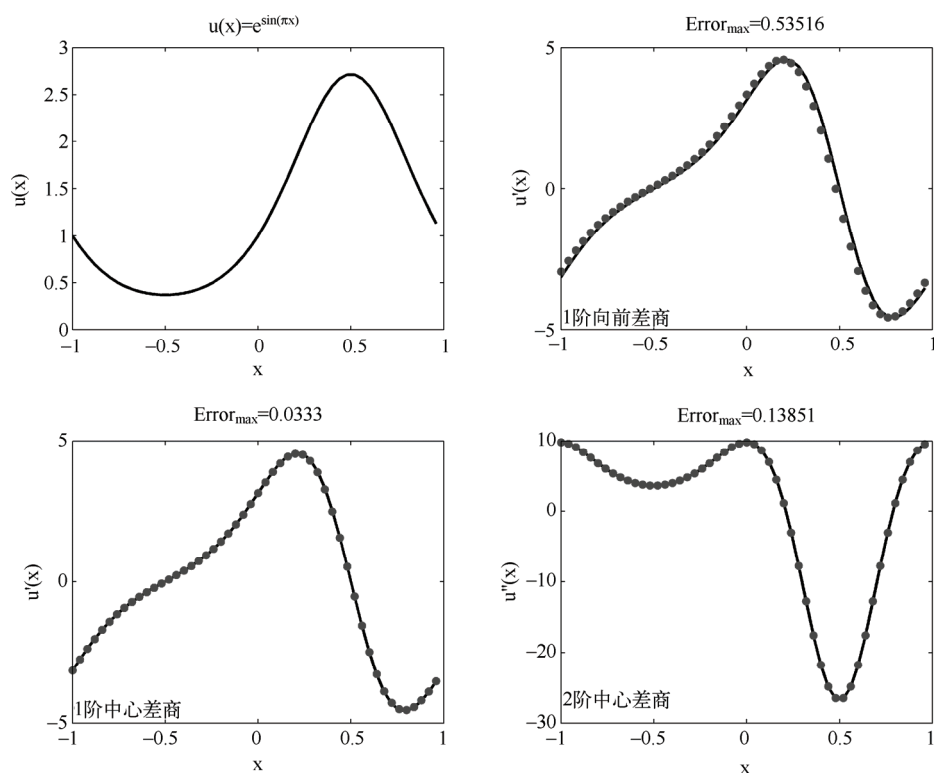


图 2-2 1 阶向前差商、1 阶中心差商、2 阶中心差商的结果（点）与解析解（曲线）

2.2 偏微分方程的差分解法

2.2.1 二维泊松方程

考虑区域 Ω 上的二维泊松问题：

$$\begin{cases} -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u = f(x, y), & (x, y) \in \Omega \\ u|_{\partial\Omega} = \varphi(x, y), \end{cases} \quad (2-16)$$

其中， $\partial\Omega$ 为区域 Ω 的边界， $f(x, y)$ 和 $\varphi(x, y)$ 为已知函数。 $u|_{\partial\Omega} = \varphi(x, y)$ 描述了函数 u 在边界上的取值，即所谓的边界条件。为简单起见，这里仅讨论 Ω 为矩形的情况，即 $a < x < b$ ， $c < y < d$ 。首先，用间隔 $h_1 = (b-a)/N$ 、 $h_2 = (d-c)/M$ 分别将 x 轴上的区间 $[a, b]$ 、 y 轴上的区间 $[c, d]$ 划分为 N 、 M 等分，得 $x_i = a + ih_1$ ， $0 \leq i \leq N$ 和 $y_j = c + jh_2$ ， $0 \leq j \leq M$ 。再根据 x_i 和 y_j 对矩形区域进行网格剖分，如图 2-3 所示。横线与竖线的交点就是网格结点，称位于边界 $\partial\Omega$ 上的结点为边界结点，其余结点为内结点。

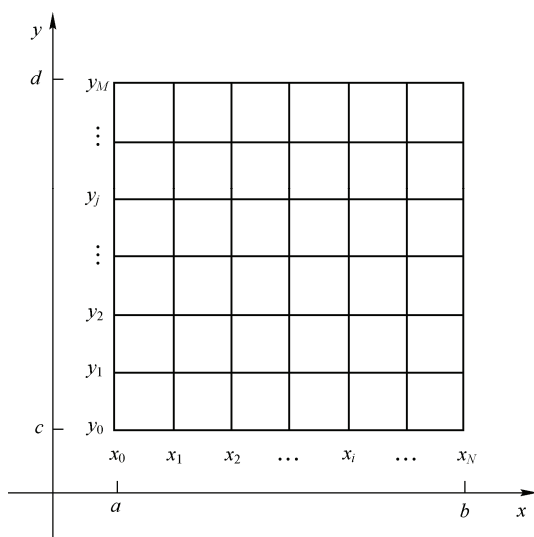


图 2-3 矩形区域的网格剖分

在内结点处考虑泊松问题：

$$-\left[\frac{\partial^2 u(x_i, y_j)}{\partial x^2} + \frac{\partial^2 u(x_i, y_j)}{\partial y^2}\right] = f(x_i, y_j), \quad 1 \leq i \leq N-1, \quad 1 \leq j \leq M-1 \quad (2-17)$$

由泰勒公式，有：

$$\frac{\partial^2 u(x_i, y_j)}{\partial x^2} = \frac{1}{h_1^2} [u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)] - \frac{h_1^2}{12} \frac{\partial^4 u(\xi_{ij}, y_j)}{\partial x^4}, \quad (2-18)$$

$$x_{i-1} < \xi_{ij} < x_{i+1}$$

$$\frac{\partial^2 u(x_i, y_j)}{\partial y^2} = \frac{1}{h_2^2} [u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1})] - \frac{h_2^2}{12} \frac{\partial^4 u(x_i, \eta_{ij})}{\partial y^4}, \quad (2-19)$$

$$y_{j-1} < \eta_{ij} < y_{j+1}$$

将以上两式代入式 (2-17), 忽略小量 $O(h_1^2+h_2^2)$, 并使用简写 $u_{ij}=u(x_i, y_j)$, 得到差分格式 (2-20)。所谓差分格式, 就是用几个相邻数值点的差商来替代方程中的导数或偏导数的近似算法。

$$-\frac{1}{h_2^2}u_{i,j-1}-\frac{1}{h_1^2}u_{i-1,j}+2\left(\frac{1}{h_1^2}+\frac{1}{h_2^2}\right)u_{i,j}-\frac{1}{h_1^2}u_{i+1,j}-\frac{1}{h_2^2}u_{i,j+1}=f(x_i, y_j), \quad (2-20)$$

$$1 \leq i \leq N-1, \quad 1 \leq j \leq M-1$$

先定义向量 $\mathbf{u}_j$:

$$\mathbf{u}_j = (u_{1j}, u_{2j}, \dots, u_{N-1,j})^T, \quad 0 \leq j \leq M \quad (2-21)$$

再将差分格式 (2-20) 写为矩阵形式:

$$\mathbf{D}\mathbf{u}_{j-1} + \mathbf{C}\mathbf{u}_j + \mathbf{D}\mathbf{u}_{j+1} = \mathbf{f}_j, \quad 1 \leq j \leq M-1 \quad (2-22)$$

其中, 矩阵 $\mathbf{C}$ 、 $\mathbf{D}$ 、向量 $\mathbf{f}_j$ 的定义如下, 注意向量 $\mathbf{f}_j$ 的首尾元素已包含了 $x=a$ 和 $x=b$ 处的边界条件。

$$\mathbf{C} = \begin{pmatrix} 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} & & & \\ -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} \\ & & & -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) \end{pmatrix} \quad (2-23)$$

$$\mathbf{D} = \begin{pmatrix} -\frac{1}{h_2^2} & & & & \\ & -\frac{1}{h_2^2} & & & \\ & & \ddots & & \\ & & & -\frac{1}{h_2^2} & \\ & & & & -\frac{1}{h_2^2} \end{pmatrix}, \quad \mathbf{f}_j = \begin{pmatrix} f(x_1, y_j) + \frac{1}{h_1^2}\varphi(x_0, y_j) \\ f(x_2, y_j) \\ \vdots \\ f(x_{N-2}, y_j) \\ f(x_{N-1}, y_j) + \frac{1}{h_1^2}\varphi(x_N, y_j) \end{pmatrix} \quad (2-24)$$

式 (2-22) 还可以进一步写成如下的矩阵形式, 注意等号右边向量的首尾元素加入了 $y=c$ 和 $y=d$ 处的边界条件。

$$\begin{pmatrix} C & D & & \\ D & C & D & \\ & \ddots & \ddots & \ddots \\ & & D & C & D \\ & & & D & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{M-2} \\ u_{M-1} \end{pmatrix} = \begin{pmatrix} f_1 - Du_0 \\ f_2 \\ \vdots \\ f_{M-2} \\ f_{M-1} - Du_M \end{pmatrix} \quad (2-25)$$

因此，矩形 Ω 上的二维泊松问题就转化为上面形如 $Au=f$ 的矩阵问题。对于这类问题，最直接的解法就是先求 A 的逆矩阵 A^{-1} ，然后 $u=A^{-1}f$ 即可，这在 Matlab 中可用左除 “\” 实现。注意式 (2-22) 只针对内结点，所以 C 、 D 均是 $N-1$ 阶方阵， A 是 $(N-1)(M-1)$ 阶方阵。下面给出一个实际例子。

$$\begin{cases} -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u = (\pi^2 - 1)e^x \sin(\pi y), & 0 < x < 2, & 0 < y < 1 \\ u(0, y) = \sin(\pi y), & u(2, y) = e^2 \sin(\pi y), & 0 \leq y \leq 1 \\ u(x, 0) = 0, & u(x, 1) = 0, & 0 < x < 2 \end{cases} \quad (2-26)$$

根据式 (2-25) 求解这个泊松问题，代码如下：

程序 2-2

```
clear all; close all;
%生成网格上的坐标
h=0.1; x=[0:h:2]'; y=[0:h:1]';
N=length(x)-1; M=length(y)-1;
[X,Y]=meshgrid(x,y);
%解析解
u_analytical=exp(X).*sin(pi*Y);
X=X(2:M,2:N); Y=Y(2:M,2:N);
%生成 f(x,y)的矩阵
f=(pi^2-1)*exp(X).*sin(pi*Y);
f(:,1)=f(:,1)+sin(pi*y(2:M))/h^2;
f(:,end)=f(:,end)+exp(2)*sin(pi*y(2:M))/h^2;
%构造矩阵 D、C、A
e=ones(N-1,1);
C=1/h^2*spdiags([-e 4*e -e],[-1 0 1],N-1,N-1);
D=-1/h^2*eye(N-1);
e=ones(M-1,1);
A=kron(eye(M-1),C)+kron(spdiags([e e],[-1 1],M-1,M-1),D);
%左除求解
f=f'; u=zeros(M+1,N+1);
u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
```

```

u(:,1)=sin(pi*y); u(:,end)=exp(2)*sin(pi*y);
%画图
figure(1), spy(A,'k',16)
figure(2), subplot(2,2,1), mesh(x,y,u)
xlabel x, ylabel y, zlabel u
subplot(2,2,2), mesh(x,y,u-u_analytical)
axis([0 2 0 1 0 0.04])
xlabel x, ylabel y, zlabel Error

```

程序输出结果如图 2-4 所示，左图为 $h_1=h_2=0.1$ 时泊松问题 (2-26) 的数值解，右图
为数值解与解析解 $u=e^x\sin(\pi y)$ 之间的误差。当然，进一步缩小 h_1 和 h_2 会以增加运算量为代
价降低这个误差。

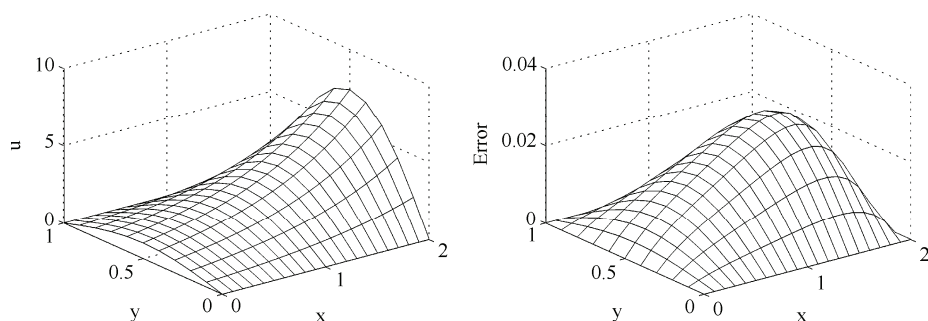


图 2-4 数值结果（左图）和误差（右图）

为了使读者产生直观的理解，图 2-5 显示了 $h_1=h_2=0.2$ 时方阵 A 的非零元素分布情况。

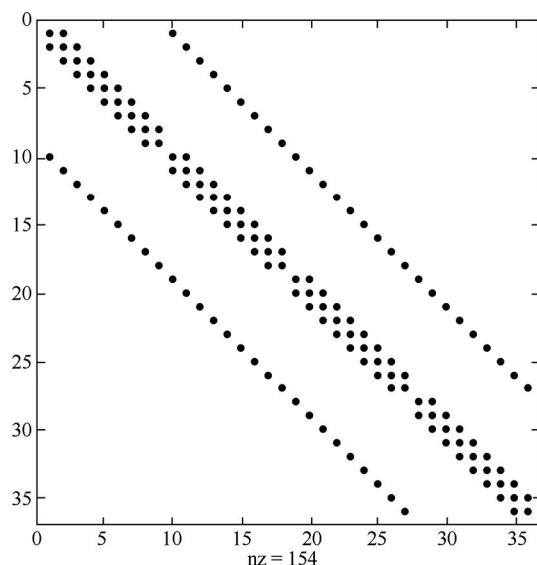


图 2-5 方阵 A 的非零元素分布

2.2.2 一维热传导方程

考虑一维热传导问题：

$$\begin{cases} \frac{\partial u}{\partial t} - a \frac{\partial^2 u}{\partial x^2} = f(x, t), & 0 < x < 1, & 0 < t \leq T \\ u(x, 0) = \varphi(x), & 0 \leq x \leq 1 \\ u(0, t) = \alpha(t), & u(1, t) = \beta(t), & 0 < t \leq T \end{cases} \quad (2-27)$$

其中 a 为正常数, $f(x, t)$ 、 $\varphi(x)$ 、 $\alpha(t)$ 和 $\beta(t)$ 为已知函数。 $u(x, 0)=\varphi(x)$ 给出了函数 u 在 $t=0$ 时刻的取值, 即初始条件。 $u(0, t)=\alpha(t)$ 和 $u(1, t)=\beta(t)$ 给出了函数 u 在边界上的取值, 即边界条件。

首先, 以空间步长 $h=1/N$ 、时间步长 $\tau=T/M$ 分别将 x 轴上区间 $[0, 1]$ 、 t 轴上区间 $[0, T]$ 分成 N 、 M 等分, 得 $x_i=ih$ 、 $0 \leq i \leq N$ 和 $t_k=k\tau$ 、 $0 \leq k \leq M$ 。再根据 x_i 和 t_k 对计算区域做网格剖分, 如图 2-6 所示。横线与竖线的交点即为结点, 称在 $t=0$ 、 $x=0$ 或 $x=1$ 上的结点为边界结点, 其余结点为内结点。

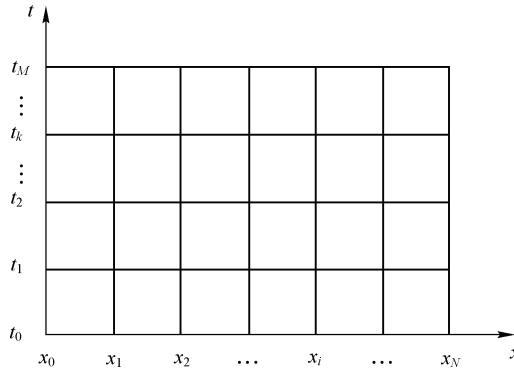


图 2-6 计算区域的网格剖分

然后在结点处考虑热传导问题：

$$\frac{\partial u(x_i, t_k)}{\partial t} - a \frac{\partial^2 u(x_i, t_k)}{\partial x^2} = f(x_i, t_k), \quad 1 \leq i \leq N-1, \quad 0 \leq k \leq M-1 \quad (2-28)$$

根据泰勒公式, 有：

$$\frac{\partial^2 u(x_i, t_k)}{\partial x^2} = \frac{1}{h^2} [u(x_{i-1}, t_k) - 2u(x_i, t_k) + u(x_{i+1}, t_k)] - \frac{h^2}{12} \frac{\partial^4 u(\xi_{ik}, t_k)}{\partial x^4}, \quad (2-29)$$

$$x_{i-1} < \xi_{ik} < x_{i+1}$$

$$\frac{\partial u(x_i, t_k)}{\partial t} = \frac{1}{\tau} [u(x_i, t_{k+1}) - u(x_i, t_k)] - \frac{\tau}{2} \frac{\partial^2 u(x_i, \eta_{ik})}{\partial t^2}, \quad t_k < \eta_{ik} < t_{k+1} \quad (2-30)$$

将上面两式代入式 (2-28), 并将小量 $O(\tau+h^2)$ 作为局部截断误差忽略掉, 同时引入步

长比 $r = \alpha\tau/h^2$ ，并使用简写 $u_i^k = u(x_i, t_k)$ ，得差分格式：

$$u_i^{k+1} = (1-2r)u_i^k + r(u_{i-1}^k + u_{i+1}^k) + \tau f(x_i, t_k), \quad 1 \leq i \leq N-1, \quad 0 \leq k \leq M-1 \quad (2-31)$$

图 2-7 显示了差分格式 (2-31) 所涉及结点的相对位置。要计算函数 u 在 (x_i, t_{k+1}) 处的取值，就需要用到其在 (x_{i-1}, t_k) 、 (x_i, t_k) 和 (x_{i+1}, t_k) 处的取值。

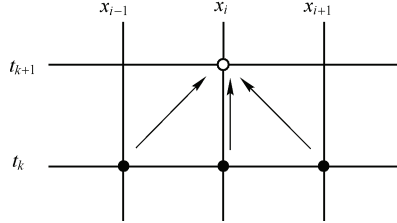


图 2-7 差分格式 (2-31) 所涉及结点的相对位置

最后，将式 (2-31) 写成矩阵形式 (2-32)，注意最右端向量的首尾元素已包含了边界条件。这样，通过一次矩阵相乘和相加就可以由 $(u_1^k, u_2^k, \dots, u_{N-2}^k, u_{N-1}^k)^T$ 得到 $(u_1^{k+1}, u_2^{k+1}, \dots, u_{N-2}^{k+1}, u_{N-1}^{k+1})^T$ 。

$$\begin{pmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \vdots \\ u_{N-2}^{k+1} \\ u_{N-1}^{k+1} \end{pmatrix} = \begin{pmatrix} 1-2r & r & & & \\ r & 1-2r & r & & \\ & \ddots & \ddots & \ddots & \\ & & r & 1-2r & r \\ & & & r & 1-2r \end{pmatrix} \begin{pmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_{N-2}^k \\ u_{N-1}^k \end{pmatrix} + \begin{pmatrix} \tau f(x_1, t_k) + ru_0^k \\ \tau f(x_2, t_k) \\ \vdots \\ \tau f(x_{N-2}, t_k) \\ \tau f(x_{N-1}, t_k) + ru_N^k \end{pmatrix} \quad (2-32)$$

需要强调的是，当步长比 $r = \alpha\tau/h^2 \leq 1/2$ 时，差分格式 (2-31) 是稳定的，反之则是不稳定的，计算结果也不可信。这一稳定性条件称为 CFL 条件，该条件是根据提出者 Courant、Friedrichs、Lewy 的名字命名的，它保证了误差和噪声在迭代的每一步中都不会放大，证明从略。下面看一个实例：

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & 0 < x < 1, \quad 0 < t \leq 1 \\ u(x, 0) = e^x, & 0 \leq x \leq 1 \\ u(0, t) = e^t, \quad u(1, t) = e^{1+t}, & 0 < t \leq 1 \end{cases} \quad (2-33)$$

根据有限差分法的矩阵形式 (2-32) 计算热传导问题 (2-33)，代码如下：

程序 2-3

```
clear all; close all;
a=1; h=0.05; x=[0:h:1];
tau=0.00125; t=[0:tau:1];
```

```

r=a*tau/h^2;
N=length(x)-1; M=length(t)-1;
%构造矩阵
e=r*ones(N-1,1);
A=spdiags([e 1-2*e e],[-1 0 1],N-1,N-1);
%设置初始条件和边界条件
u=zeros(N+1,M+1);
u(:,1)=exp(x);
u(1,:)=exp(t);
u(end,:)=exp(1+t);
%有限差分
for n=1:M
    u(2:N,n+1)=A*u(2:N,n);
    u(2,n+1)=u(2,n+1)+r*u(1,n);
    u(N,n+1)=u(N,n+1)+r*u(end,n);
end
%画图
subplot(2,2,1)
mesh(t(1:20:end),x,u(:,1:20:end))
xlabel t, ylabel x, zlabel u
subplot(2,2,2)
[T X]=meshgrid(t(1:20:end),x);
%与解析解的误差
mesh(t(1:20:end),x,exp(X+T)-u(:,1:20:end))
xlabel t, ylabel x, zlabel Error

```

程序输出如图 2-8 所示，左图为数值结果，右图为数值结果与解析解 $u=e^{x+t}$ 之间的误差，程序使用的步长比 $r=1/2$ ，所以计算过程是稳定的。

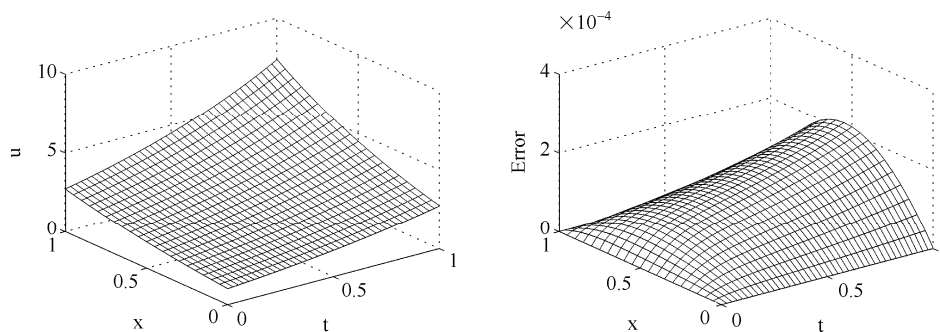


图 2-8 数值结果（左图）和误差（右图）

2.2.3 一维波动方程

考虑如下二维波动方程：

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - a^2 \frac{\partial^2 u}{\partial x^2} = f(x, t), & 0 < x < 1, \quad 0 < t \leq T \\ u(x, 0) = \varphi(x), \quad \frac{\partial u(x, 0)}{\partial t} = \psi(x), & 0 \leq x \leq 1 \\ u(0, t) = \alpha(t), \quad u(1, t) = \beta(t), & 0 < t \leq T \end{cases} \quad (2-34)$$

其中， a 为正常数， $f(x, t)$ 、 $\varphi(x)$ 、 $\psi(x)$ 、 $\alpha(t)$ 和 $\beta(t)$ 为已知函数。这里的计算区域网格剖分过程和 2.2.2 小节完全一样，就不再赘述。在结点处讨论一维波动方程，有：

$$\frac{\partial^2 u(x_i, t_k)}{\partial t^2} - a^2 \frac{\partial^2 u(x_i, t_k)}{\partial x^2} = f(x_i, t_k), \quad 1 \leq i \leq N-1, \quad 1 \leq k \leq M-1 \quad (2-35)$$

由泰勒公式，有：

$$\frac{\partial^2 u(x_i, t_k)}{\partial x^2} = \frac{1}{h^2} [u(x_{i-1}, t_k) - 2u(x_i, t_k) + u(x_{i+1}, t_k)] - \frac{h^2}{12} \frac{\partial^4 u(\xi_{ik}, t_k)}{\partial x^4}, \quad (2-36)$$

$$x_{i-1} < \xi_{ik} < x_{i+1}$$

$$\frac{\partial^2 u(x_i, t_k)}{\partial t^2} = \frac{1}{\tau^2} [u(x_i, t_{k-1}) - 2u(x_i, t_k) + u(x_i, t_{k+1})] - \frac{\tau^2}{12} \frac{\partial^4 u(x_i, \eta_{ik})}{\partial t^4}, \quad (2-37)$$

$$t_{k-1} < \eta_{ik} < t_{k+1}$$

将上两式代入式 (2-35)，并将小量 $O(\tau^2 + h^2)$ 作为局部截断误差忽略掉，同时引入步长比 $s = a\tau/h$ ，并使用简写 $u_i^k = u(x_i, t_k)$ ，得差分格式：

$$\begin{aligned} u_i^{k+1} &= s^2 u_{i-1}^k + 2(1-s^2)u_i^k + s^2 u_{i+1}^k - u_i^{k-1} + \tau^2 f(x_i, t_k) \\ 1 \leq i \leq N-1, \quad 1 \leq k \leq M-1 \end{aligned} \quad (2-38)$$

根据 CFL 条件，仅当步长比 $s \leq 1$ 时，上式才是稳定的，误差也会被控制在较小的程度，证明从略。如图 2-9 所示为此差分格式所涉及结点的相对位置。要计算函数 u 在 (x_i, t_{k+1}) 处的取值，就需要用到函数 u 在 (x_{i-1}, t_k) 、 (x_i, t_k) 、 (x_{i+1}, t_k) 及 (x_i, t_{k-1}) 处的取值。

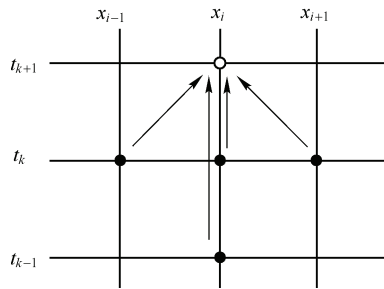


图 2-9 差分格式 (2-38) 所涉及结点的相对位置

将差分格式 (2-38) 写为矩阵形式 (2-39)，注意最后一个向量的首尾元素已包含了边界条件：

$$\begin{pmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \vdots \\ u_{N-2}^{k+1} \\ u_{N-1}^{k+1} \end{pmatrix} = \begin{pmatrix} 2(1-s^2) & s^2 & & & \\ & s^2 & 2(1-s^2) & s^2 & \\ & & \ddots & \ddots & \ddots \\ & & & s^2 & 2(1-s^2) & s^2 \\ & & & & s^2 & 2(1-s^2) \end{pmatrix} \begin{pmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_{N-2}^k \\ u_{N-1}^k \end{pmatrix} - \begin{pmatrix} u_1^{k-1} \\ u_2^{k-1} \\ \vdots \\ u_{N-2}^{k-1} \\ u_{N-1}^{k-1} \end{pmatrix} + \begin{pmatrix} \tau^2 f(x_1, t_k) + s^2 u_0^k \\ \tau^2 f(x_2, t_k) \\ \vdots \\ \tau^2 f(x_{N-2}, t_k) \\ \tau^2 f(x_{N-1}, t_k) + s^2 u_N^k \end{pmatrix}, \quad 1 \leq k \leq M-1 \quad (2-39)$$

值得一提的是，计算最开始时需要两个已知向量：\$(u_1^0, u_2^0, \dots, u_{N-2}^0, u_{N-1}^0)^T\$ 和 \$(u_1^1, u_2^1, \dots, u_{N-2}^1, u_{N-1}^1)^T\$。第一个向量可直接通过初始条件获得，即 \$u_i^0 = \varphi(x_i)\$，第二个向量通常选用欧拉法来近似估算，即 \$u_i^1 = \varphi(x_i) + \tau \psi(x_i)\$。下面给出一个实例。

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = (t^2 - x^2) \sin(xt), & 0 < x < 1, \quad 0 < t \leq 1 \\ u(x, 0) = 0, \quad \frac{\partial u(x, 0)}{\partial t} = x, & 0 \leq x \leq 1 \\ u(0, t) = 0, \quad u(1, t) = \sin t, & 0 < t \leq 1 \end{cases} \quad (2-40)$$

根据式 (2-39) 求解上式，代码如下：

程序 2-4

```
clear all; close all;
a=1; h=0.05; x=[0:h:1];
tau=0.05; t=[0:tau:1];
s=a*tau/h;
N=length(x)-1; M=length(t)-1;
[T X]=meshgrid(t,x);
%构造矩阵
e=s^2*ones(N-1,1);
A=spdiags([e 2*(1-e) e],[-1 0 1],N-1,N-1);
%设置初始条件和边界条件
u=zeros(N+1,M+1);
```

```

u(:,1)=0; u(:,2)=tau*x;
u(1,:)=0; u(end,:)=sin(t);
%有限差分
for n=2:M
    u(2:N,n+1)=A*u(2:N,n)-u(2:N,n-1)+ ...
        tau^2*(T(2:N,n).^2-X(2:N,n).^2).*sin(X(2:N,n).*T(2:N,n));
    u(2,n+1)=u(2,n+1)+s^2*u(1,n);
    u(N,n+1)=u(N,n+1)+s^2*u(end,n);
end
%画图
subplot(2,2,1)
mesh(t,x,u), view(20,40), xlabel t, ylabel x, zlabel u
subplot(2,2,2)
%与解析解的误差
mesh(t,x,u-sin(X.*T)), view(20,40), axis([0 1 0 1 0 6e-5])
xlabel t, ylabel x, zlabel Error

```

程序输出如图 2-10 所示,左图为数值解,右图为数值解与解析解 $u=\sin(xt)$ 之间的误差。代码中的步长比 $s=1$, 此时差分格式是稳定的, 误差在 10^{-5} 级。

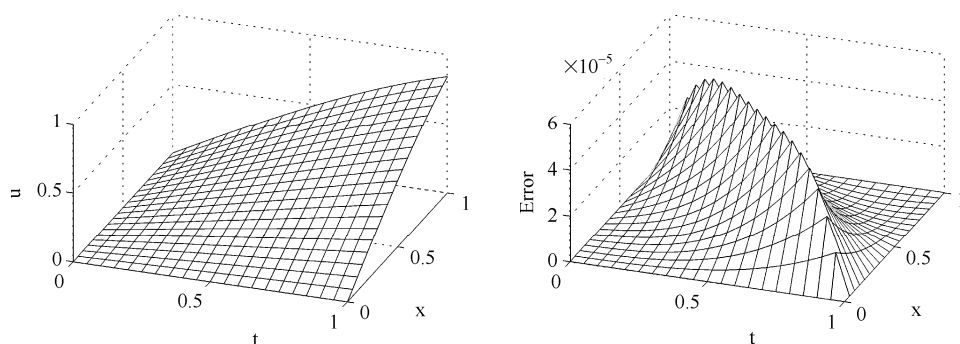


图 2-10 数值结果 (左图) 和误差 (右图)

至此, 针对三种常见偏微分方程的有限差分法原理和示例就介绍完了。实际上, 有限差分法还有更多深入的内容, 限于篇幅没有在此处提及。简单来讲, 有限差分法的核心思路就是通过差商来近似偏微分方程中的偏导数 (如: $\partial u / \partial t$ 、 $\partial^2 u / \partial t^2$ 、 $\partial^2 u / \partial x^2$), 得到可直接迭代计算的差分格式, 进而数值求解。这在过去几十年间都是处理偏微分问题的基本有效手段。

但是，有限差分法是一种局部的方法，即每个位置的导数（ $\partial u/\partial x$ 、 $\partial^2 u/\partial x^2 \dots$ ）都是由临近的几个点计算而来的。计算过程中所使用的矩阵都是包含很多 0 的稀疏矩阵也反映了这一点，因此它的精度不高。而后续章节即将讨论的谱方法是全局的算法，它使用所有已知点来计算某一处的导数，这样就大大提高了精度。

另外，在包含时间的偏微分问题中，由于 CFL 条件的限制，有限差分法不能采用较大的时间步长快速地得到结果。而在后文中，利用谱方法求解此类问题时，将使用变步长的 ode 系列函数计算 $\partial u/\partial t$ ，无需选定固定的时间步长就能在较高精度的前提下尽快得到结果。

2.3 有限元法和 Matlab 偏微分工具箱

有限差分法直接将偏微分方程离散化并根据泰勒级数展开写成差分格式近似求解，而有限元法则是把偏微分方程定解问题转化为一个等价的变分问题求解。较之有限差分法，有限元法的一个突出优点就是它可以解决在复杂几何形状上具有任意边界条件的微分问题。比如在二维情况下，有限元法使用三角形单元、四边形单元或多边形单元对求解区域进行剖分，这就能轻松适用于不规则形状的求解区域。

限于本书主旨和篇幅，本节不阐述有限元法的复杂原理，而是介绍一款 Matlab 中的基于有限元法的数值模拟工具——偏微分工具箱（PDE toolbox）的操作方法。简单来讲，偏微分工具箱给用户提供了 3 个功能：定义偏微分方程（设置计算区域、边界条件、方程参数）；数值求解偏微分方程（产生非结构网格、方程离散化、得到近似解）；将结果可视化。

在平面的有界区域 Ω 上，偏微分工具箱可处理以下 4 类问题：

（1）椭圆型（elliptic）方程：

$$-\nabla \cdot (c \nabla u) + au = f \quad (2-41)$$

（2）抛物型（parabolic）方程：

$$d \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u) + au = f \quad (2-42)$$

（3）双曲型（hyperbolic）方程：

$$d \frac{\partial^2 u}{\partial t^2} - \nabla \cdot (c \nabla u) + au = f \quad (2-43)$$

（4）特征（eigen）值问题（ λ 为未知特征值）：

$$-\nabla \cdot (c \nabla u) + au = \lambda du \quad (2-44)$$

其中， ∇ 为向量微分算符， a 、 c 、 d 、 f 和未知函数 u 均是 Ω 上的实（复）函数，对于抛物型和双曲型方程， a 、 c 、 d 、 f 也可以包含时间 t 、函数 u 、以及函数 u 的梯度。此外，偏微分工具箱还可以处理方程组的情况。

在边界 $\partial\Omega$ 处, 边界条件有3种选择:

(1) 狄利克莱 (Dirichlet) 边界条件, 它给出了函数 u 在边界上的取值:

$$hu = r \quad (2-45)$$

其中, h 和 r 可以是空间 (x 和 y)、函数 u 以及时间 t 的函数。

(2) 诺依曼 (Neumann) 边界条件, 它给出了函数 u 的变化率在边界上的情况:

$$\mathbf{n} \cdot (c\nabla u) + qu = g \quad (2-46)$$

其中, 向量 $\mathbf{n}$ 代表边界处向外的法线方向。 q 、 g 可以是空间 (x 和 y)、函数 u 以及时间 t 的函数。

(3) 混和边界条件, 即同时存在前两种边界条件 (仅限方程组的情况)。

2.3.1 基本操作

在 Matlab 中输入 `pdetool` 并回车, 可启动图形界面化的偏微分工具箱, 如图 2-11 所示。界面的上侧有一个工具栏, 供用户方便、快捷地进行常用操作, 如图 2-12 所示。工具栏中每个按钮的作用见表 2-1。

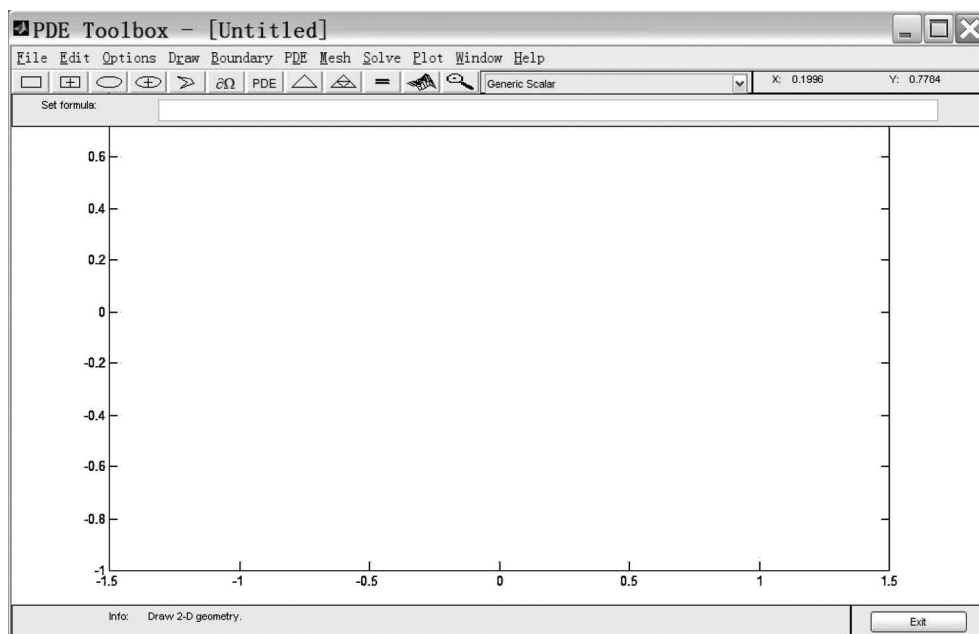


图 2-11 偏微分工具箱界面

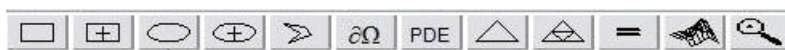


图 2-12 工具栏

表 2-1 工具栏按钮说明

	从顶点开始创建矩形/正方形。用鼠标左键点击和拖曳创建矩形。用鼠标右键（或 Ctrl+左键）点击和拖曳创建正方形
	从中心开始创建矩形/正方形。用鼠标左键点击和拖曳创建矩形。用鼠标右键（或 Ctrl+左键）点击和拖曳创建正方形
	从边缘开始创建椭圆形/圆形。用鼠标左键点击和拖曳创建椭圆形。用鼠标右键（或 Ctrl+左键）点击和拖曳创建圆形
	从中心开始创建椭圆形/圆形。用鼠标左键点击和拖曳创建椭圆形。用鼠标右键（或 Ctrl+左键）点击和拖曳创建圆形
	创建多边形。用鼠标左键点击和拖曳创建多边形的边，最后使用右键或点击多边形的起点来闭合多边形
	设置边界条件
	设置偏微分方程参数
	初始化三角网格
	加密三角网格
	解偏微分方程
	画图设置对话框
	对可视化结果进行缩放

下面以一个最简单的实例介绍偏微分工具箱的大致操作步骤。

(1) 用工具栏的前 5 个按钮创建矩形/正方形、椭圆形/圆形或者多边形，双击创建的图形可精确设置其位置和尺寸。如有需要，可使用 Draw 菜单下的 Rotate 选项对图形进行旋转操作，还可以使用 Options 菜单下的 Grid 选项加入网格以便创建图形时参考。显示区域的坐标范围在 Options 菜单下的 Axes Limits 设置，勾选 Axes Equal 可使横纵坐标的显示比例一致。矩形/正方形参数设置对话框如图 2-13 所示，设置内容包括左边、下边所在位置，以及长宽和名称。

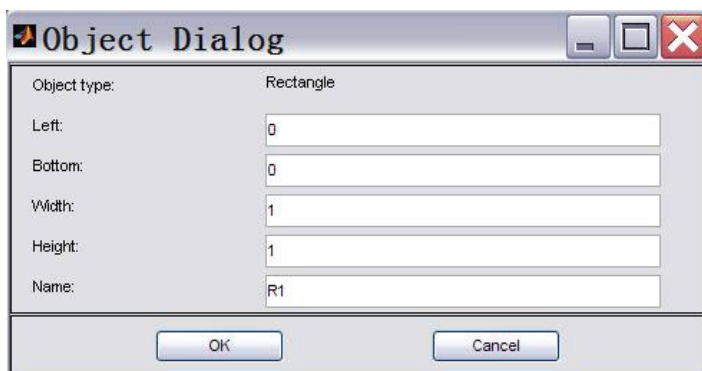


图 2-13 设置矩形/正方形参数

椭圆形参数设置对话框如图 2-14 所示，设置内容包括椭圆中心的坐标、半轴长度、旋转角度和名称。

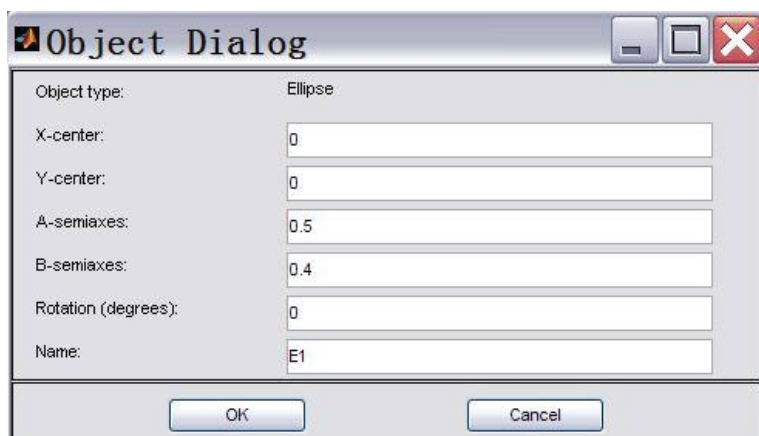


图 2-14 设置椭圆形参数

圆形参数设置对话框如图 2-15 所示，设置内容包括圆心的坐标、半径和名称。

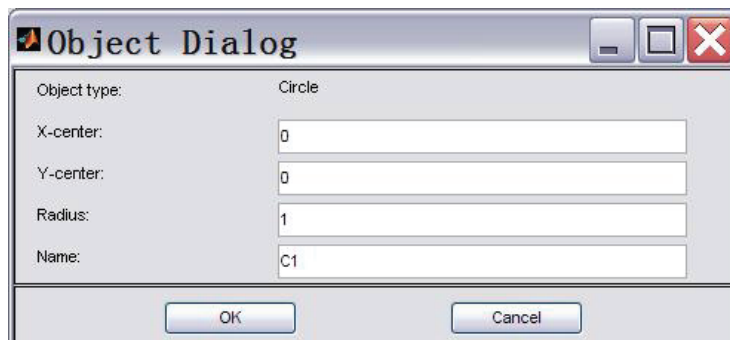


图 2-15 设置圆形参数

多边形的参数设置对话框如图 2-16 所示。设置内容为多边形每个顶点的坐标。

(2) 每创建一个图形，界面上 Set formula 后的文本框内都会出现该图形的名称，名称默认为“字母+序号”的形式。字母是形状的缩写，R 代表矩形、SQ 代表正方形，E 代表椭圆形，C 代表圆形，P 代表多边形。用加/减号连接这些图形的名称，偏微分方程的求解区域即是这些图形进行加、减的结果，比如这里设置为 $R1-SQ1-C1+E1+P1$ ，如图 2-17 所示。

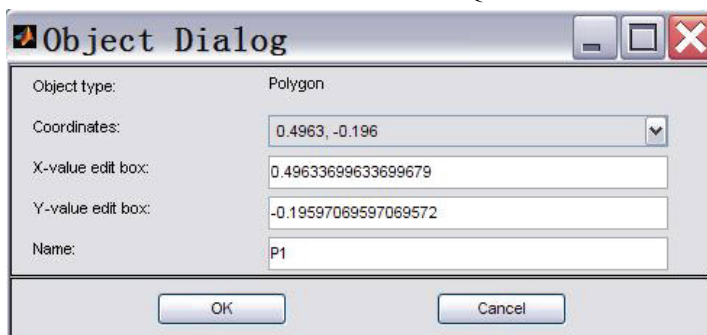


图 2-16 设置多边形参数

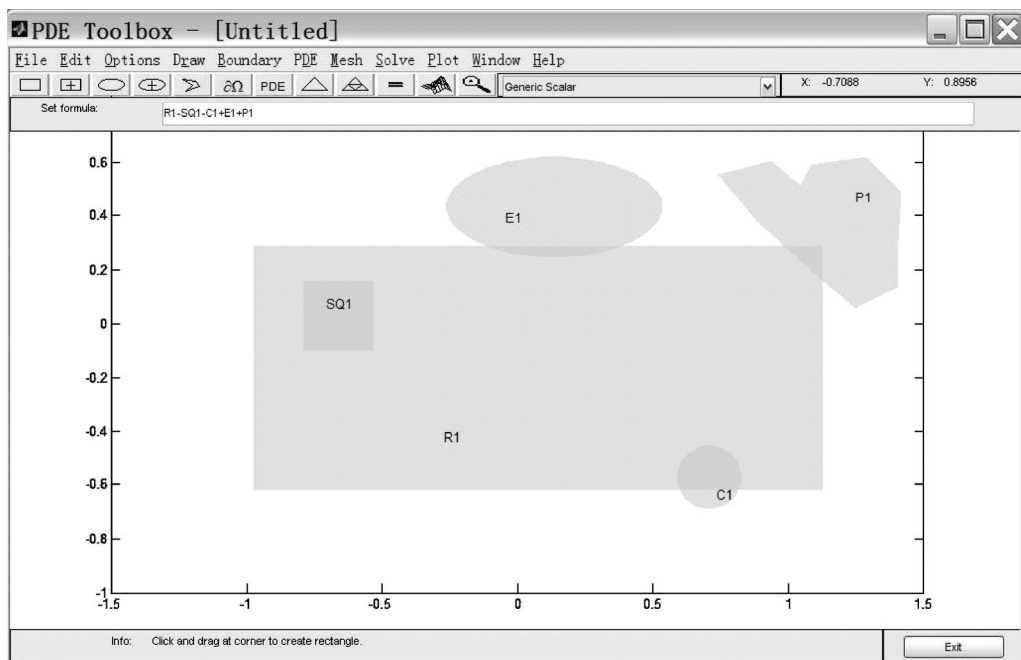


图 2-17 偏微分方程的求解区域

(3) 点击工具栏上 $\partial\Omega$ 按钮，进入边界模式，可对每一段外边界设置不同的边界条件，如图 2-18 所示。不同边界条件由颜色区分，狄利克莱边界条件为红色，诺依曼边界条件为蓝色，混和边界条件为绿色。

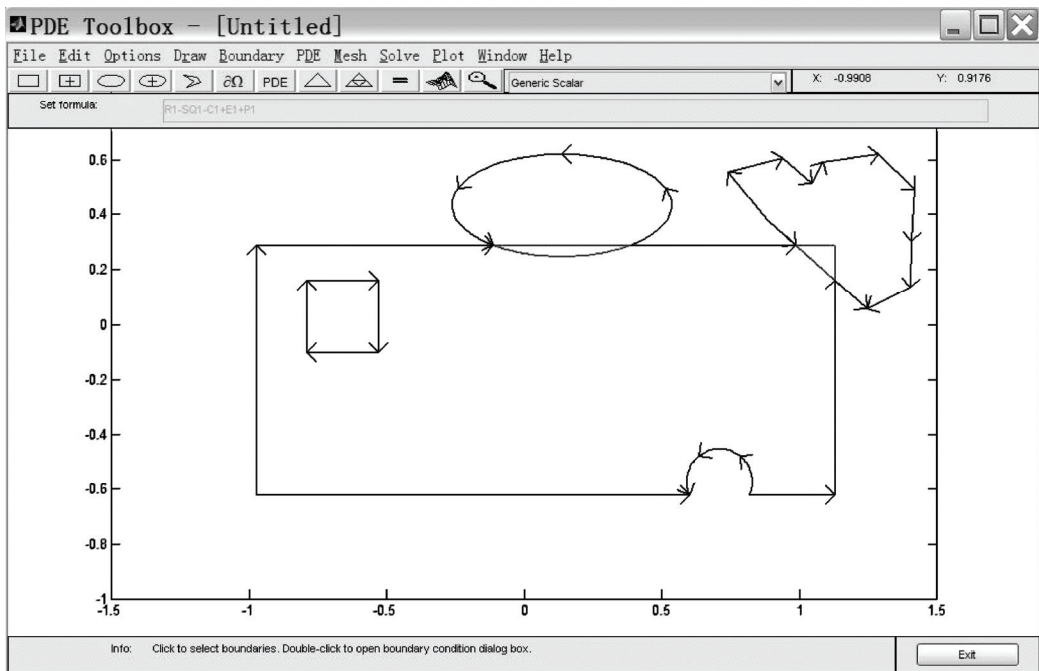


图 2-18 选择边界

双击一段外边界，弹出边界条件设置对话框，如图 2-19 所示。可选择狄利克莱“Dirichlet”或诺依曼“Neumann”边界条件，并根据边界条件的表达式（2-45）或（2-46），设置 h 、 r 或 g 、 q 的取值，式（2-46）中 c 的取值在下一步设置。

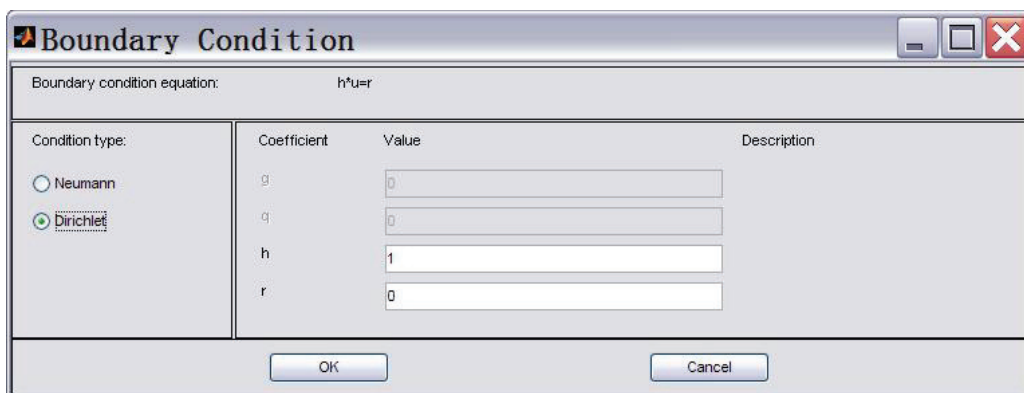


图 2-19 设置边界条件

（4）点击工具栏上  按钮，弹出偏微分方程参数设置对话框，先在左侧选择偏微分方程的类型，然后在右侧设置具体参数的取值。本例中选择椭圆型方程“Elliptic”，参数取值如图 2-20 所示。

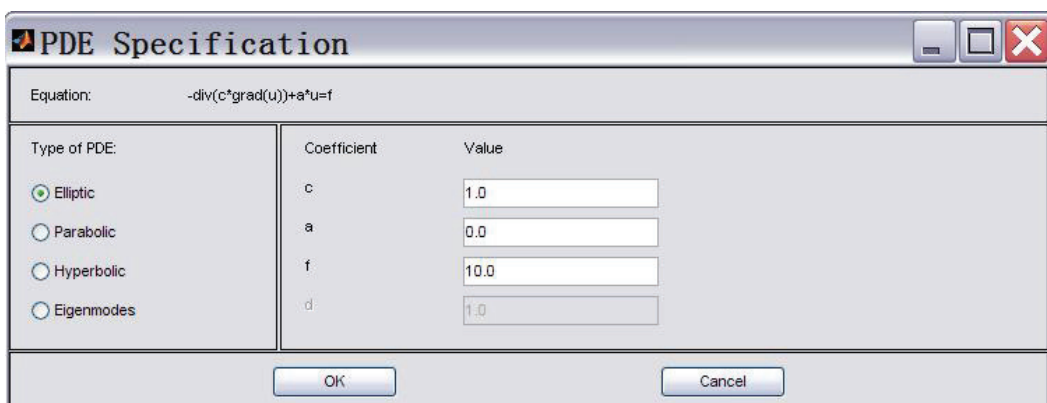


图 2-20 设置偏微分方程参数

（5）点击  按钮对求解区域进行三角网格剖分，如图 2-21 所示。

点击加密三角网格  按钮，得到更密集的网格。也可根据实际需要多次加密网格，如图 2-22 所示。

（6）点击  按钮数值求解偏微分方程并输出结果，程序默认使用颜色表示解 u 的分布，如图 2-23 所示。

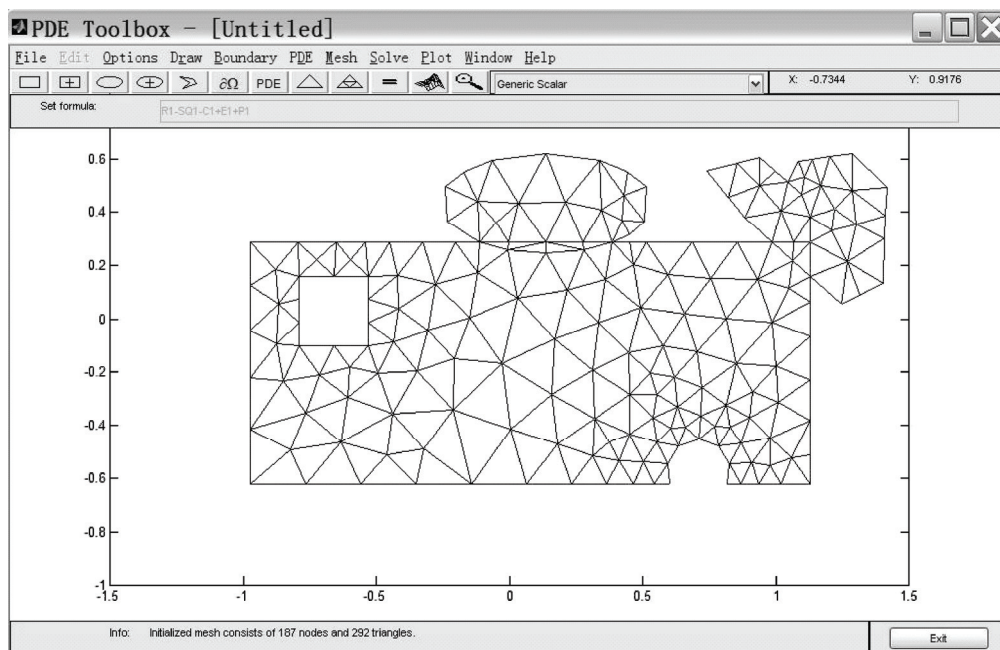


图 2-21 三角网格剖分

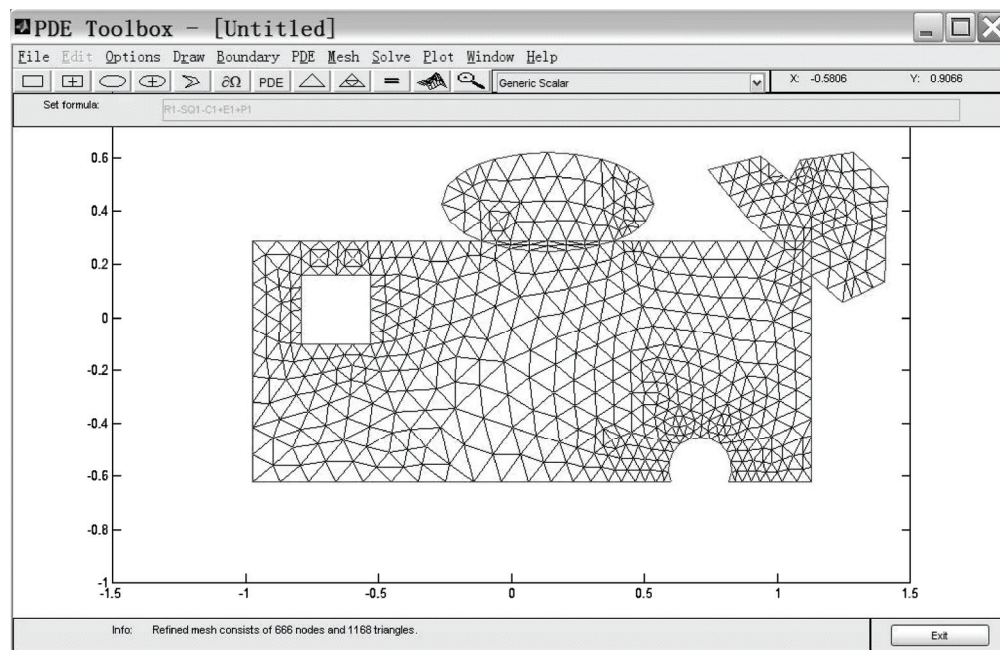


图 2-22 加密三角网格

(7) 点击  按钮，弹出画图对话框，可在这里设置输出图像的类型。除了彩色图 (Color)，还有等高线图 (Contour)、矢量场图 (Arrows)、变形网格图 (Deformed mesh)、三维图 (Height)，也可以设置是否显示坐标网格 (Plot in x-y grid)、剖分网格 (Show mesh)。

这里选择画彩色三维图并显示剖分网格，如图 2-24 所示。

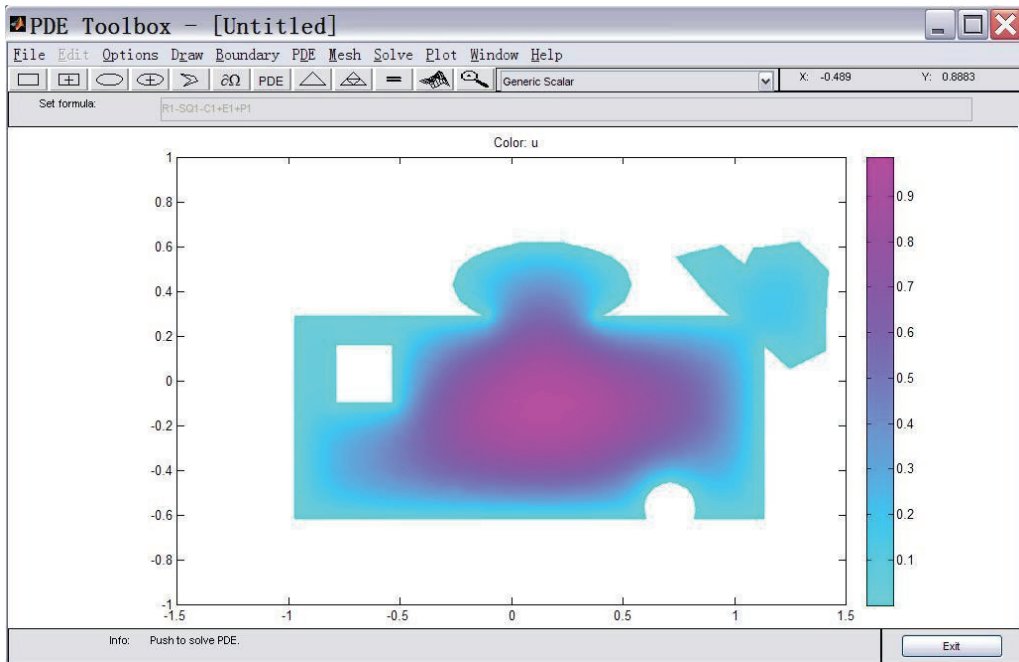


图 2-23 计算结果

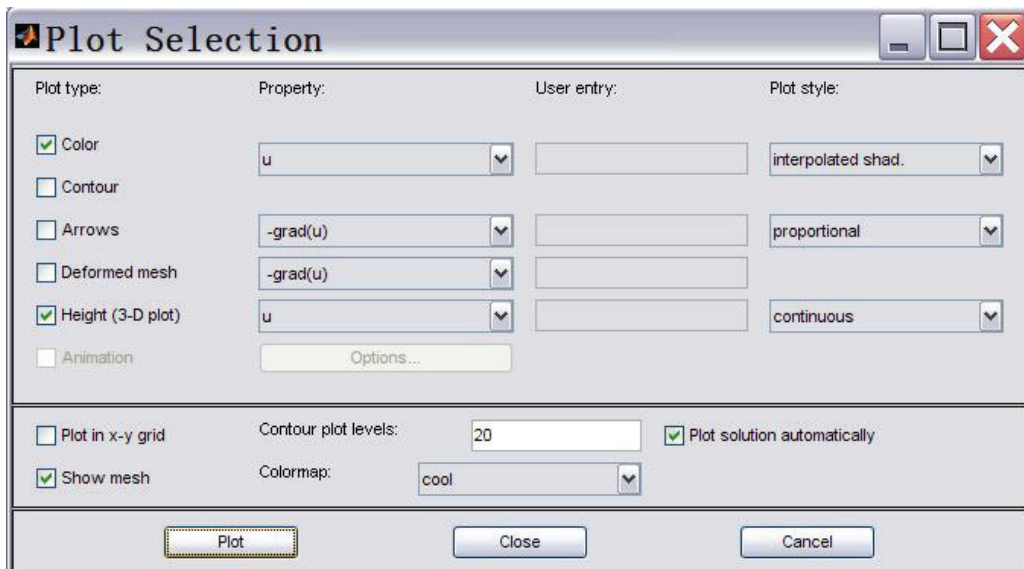


图 2-24 画图设置

点击 Plot，得到彩色三维数值结果，如图 2-25 所示。

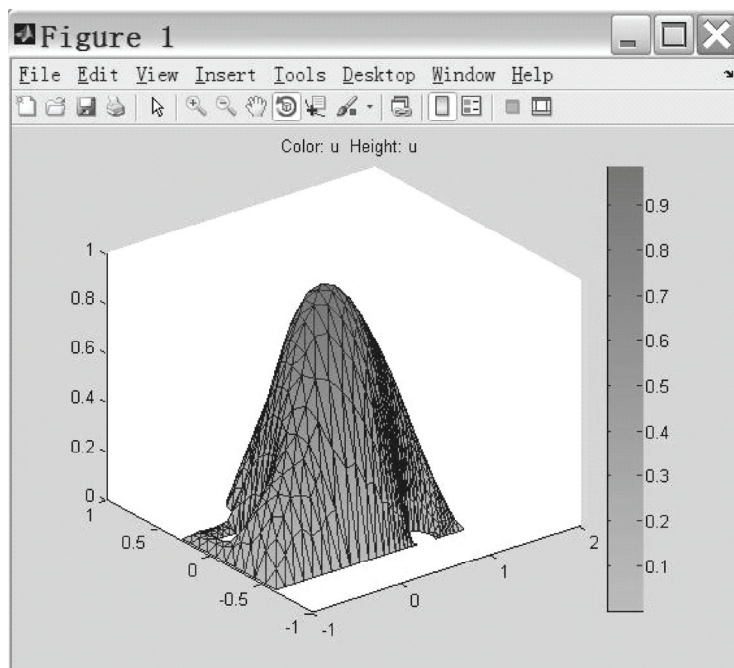


图 2-25 彩色三维数值结果

以上只是一个讲解偏微分工具箱使用步骤的简单实例。针对四种常见类型的具体偏微分方程，下面逐一给出例子。

2.3.2 二维泊松方程

考虑单位圆上的泊松问题：

$$\begin{cases} -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u = 1, & x^2 + y^2 < 1 \\ u|_{x^2+y^2=1} = 0 \end{cases} \quad (2-47)$$

其解析解为 $u=(1-x^2-y^2)/4$ ，求解步骤如下：

首先启动偏微分工具箱，点击  按钮，按住 Ctrl 键同时鼠标左键在 origin 按下并拖曳到合适位置画一个近似的单位圆。双击此圆打开图形参数设置对话框，将 X-center、Y-center、Radius 分别设置为 0、0、1，点击 OK，这样就得到了作为求解区域的单位圆。

点击  按钮进入边界模式，双击单位圆的每一段边界，弹出边界条件设置对话框，默认为 $h=1$ 、 $r=0$ 的狄利克莱边界条件 “Dirichlet”，这正是所需要的，所以不用做任何修改，点击 OK。如果若干段边界的边界条件是一样的，可按住 shift 键同时选择多段边界，双击后一起设置边界条件。

点击  按钮打开方程参数设置对话框，方程类型选椭圆型 “Elliptic”。比较式 (2-41) 和式 (2-47)，可知参数取值如下： $c=1$ 、 $a=0$ 、 $f=1$ 。填入参数后点击 OK。

点击  按钮初始化三角网格，随后可根据需要多次点击  按钮加密三角网格，程序界面的最下方将会显示网格的结点数和三角单元数。若选择 Mesh 菜单下的 Show Node Labels, 将显示结点编号, 如图 2-26 所示为初始网格及其结点编号。

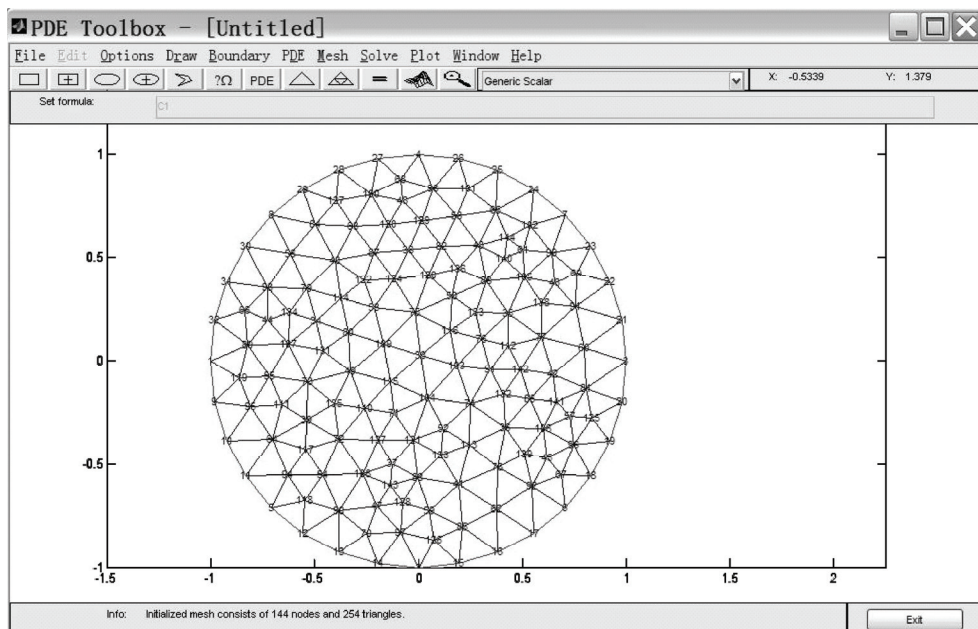


图 2-26 显示结点编号

要导出网格剖分的数据信息, 点击 Mesh 菜单下的 Export Mesh, 出现如图 2-27 所示的对话框。在其中填入用来存放点、边、三角形数据的变量名(默认为 points、edges 和 triangles 的首字母 p、e 和 t), 之后点击 OK 即可将数据导出到 Matlab 的 workspace 相应变量中。

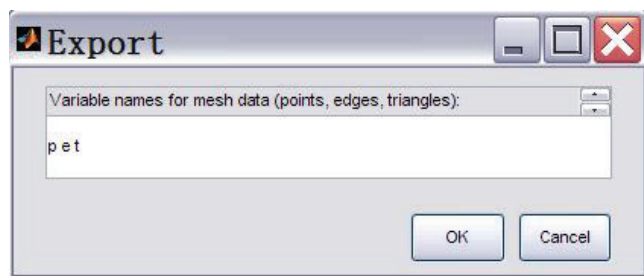


图 2-27 导出网格剖分的数据

点击  按钮数值求解方程, 方程的数值解默认以彩色二维图显示, 颜色的深浅代表函数 u 取值的大小。点击  按钮打开画图设置对话框, 勾选 Color、Height (3-D plot)、Show mesh, 点击 Plot, 得到标有剖分网格的彩色三维图, 如图 2-28 所示。当按钮  处于按下状态时, 可通过鼠标左键的拖曳来旋转三维图, 以调整最佳的观察角度。选择 File 菜单下的

Save As，可将图像以指定的图片格式保存。

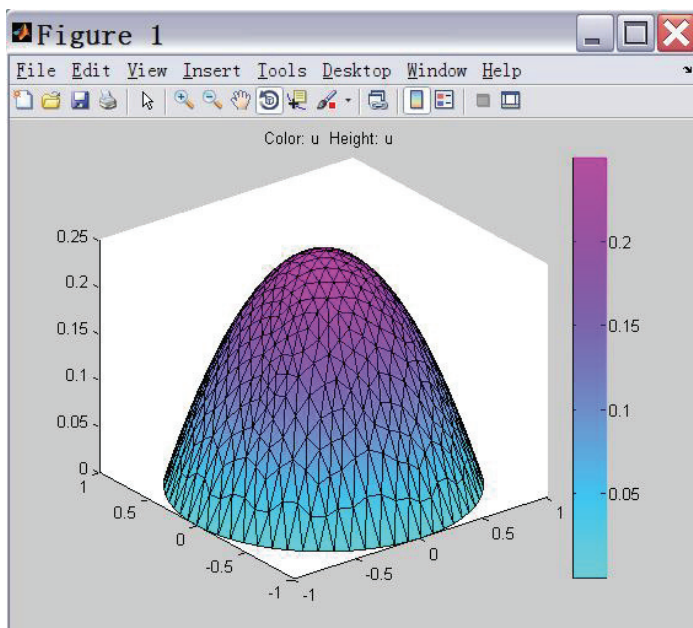


图 2-28 标有剖分网格的彩色三维图

偏微分工具箱也允许用户自定义画图的内容。点击  按钮打开画图设置对话框，勾选 Color、Height (3-D plot)、Show mesh，并在 Height (3-D plot)后面 Property 的下拉菜单中选择 user entry，其后 user entry 的文本框中填入数值解与精确解的差，即 $u-(1-x.^2-y.^2)/4$ ，如图 2-29 所示。点击 Plot，得到计算误差的彩色三维图，可以看出误差在 10^{-4} 数量级，如图 2-30 所示。

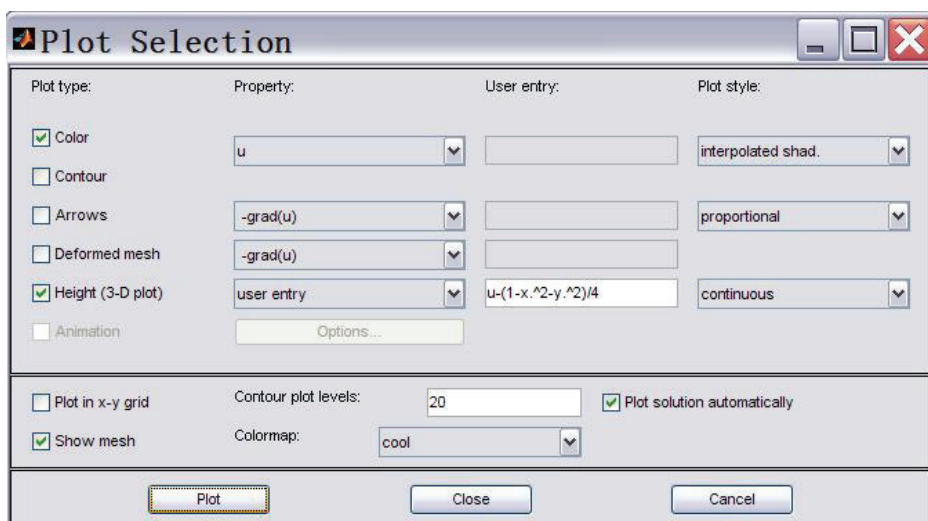


图 2-29 画误差分布图的设置

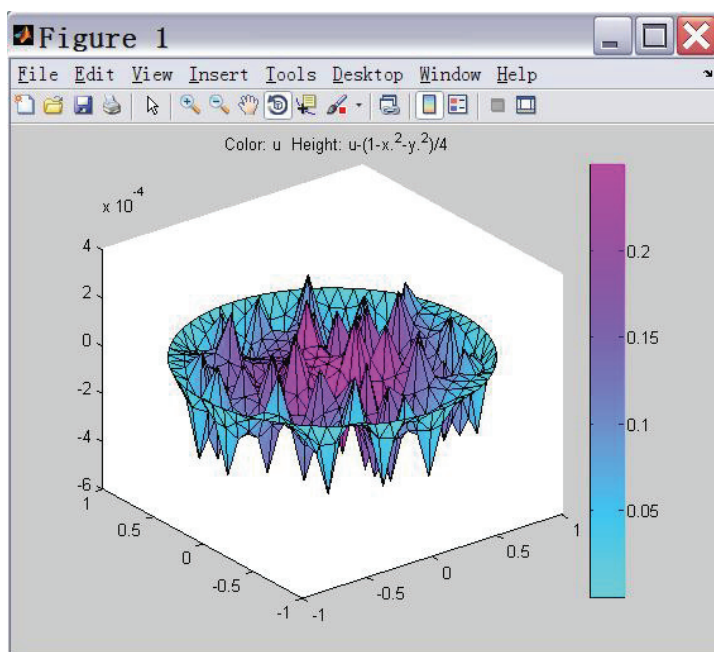


图 2-30 误差的彩色三维图

要导出数值解 u 在每个结点处的取值，点击 Solve 菜单下的 Export Solution 选项打开导出数值解的对话框，填入导出的变量名（默认为 u ），点击 OK 即可，如图 2-31 所示。

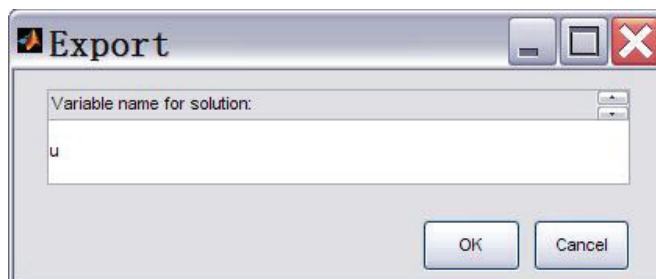


图 2-31 导出每个结点处的数值解

由于之前已经导出了三角网格的相关数据，所以结点位置（ x 、 y 坐标构成的二维数组，默认变量名为 p ，按结点序号排列）及该位置上数值解 u 的取值（默认变量名为 u ，按结点序号排列的一维数组）都出现在了 workspace 中，可供 Matlab 进行后续处理，如图 2-32 所示。

Workspace	
Stack: Base	
Name	Value
e	<7x64 double>
p	<2x541 double>
t	<4x1016 double>
u	<541x1 double>

图 2-32 workspace 中的结点数据和对应的数值解 u 的取值

2.3.3 二维热传导方程

考虑这样一个热传导问题：一个长/宽为 2×1 的金属板，中央有一个半径为 0.2 的孔。此板左边界接触一温度为 100 的恒温热源，右边界接触一温度为 200 的恒温热源，其他边界绝热。时间 $t=0$ 时金属板各处温度为 100，那么 $t=10$ 时这个金属板的温度空间分布 u 是怎样的？此问题的数学形式为：

$$\begin{cases} \frac{\partial u}{\partial t} = a \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, \\ u|_{x=-1} = 100, & u|_{x=1} = 200, \\ \frac{\partial u}{\partial n} = 0, & \text{其他边界上} \\ u|_{t=0} = 100, \end{cases} \quad (2-48)$$

取 $a=1$ ，求解步骤如下：

启动偏微分工具箱，点击  按钮，按下并拖曳鼠标左键画一个矩形。双击此矩形打开图形参数设置对话框，将 Left、Bottom、Width、Height 分别设置为 -1、-0.5、2、1，点击 OK。点击  按钮，按住 Ctrl 键同时鼠标左键在 origin 按下并拖曳到合适位置，画一个半径近似为 0.2 的圆。双击此圆打开图形参数设置对话框，将 X-center、Y-center、Radius 分别设置为 0、0、0.2，点击 OK。再将 Set formula 后的内容修改为 R1-C1，这样就设置好了求解区域。

点击  按钮进入边界模式，双击每一段边界打开边界条件设置对话框，填入边界条件并点击 OK。对于左边界，选择狄利克莱边界条件“Dirichlet”，设置 $h=1$ 、 $r=100$ 。类似地，将右边界选为狄利克莱边界条件“Dirichlet”，并令 $h=1$ 、 $r=200$ 。对于其他边界，均选择诺依曼边界条件“Neumann”，且 $q=g=0$ 。

点击  按钮打开方程参数设置对话框，方程类型选抛物型“Parabolic”。比较式 (2-42) 和式 (2-48)，可知参数取值如下： $c=1$ 、 $a=0$ 、 $f=0$ 、 $d=1$ 。填入参数后点击 OK。

点击 Solve 菜单中的 Parameters 选项，弹出求解参数设置对话框。其中，Time 一栏为求解的时间区间，填入 0:10 代表从 $t=0$ 计算到 $t=10$ 。 $u(t_0)$ 一栏为初始条件，这里填入 100。其余两栏为相对误差和绝对误差，一般采用默认设置即可，最后点击 OK。

点击  按钮初始化三角网格，随后再根据需要多次点击  按钮加密三角网格。

点击  按钮求解方程， $t=10$ 时的数值解默认将会以彩色二维图显示。点击  按钮打开画图设置对话框，勾选 Color、Contour、Arrows，点击 Plot，可得带等高线的彩色矢量场图，如图 2-33 所示。

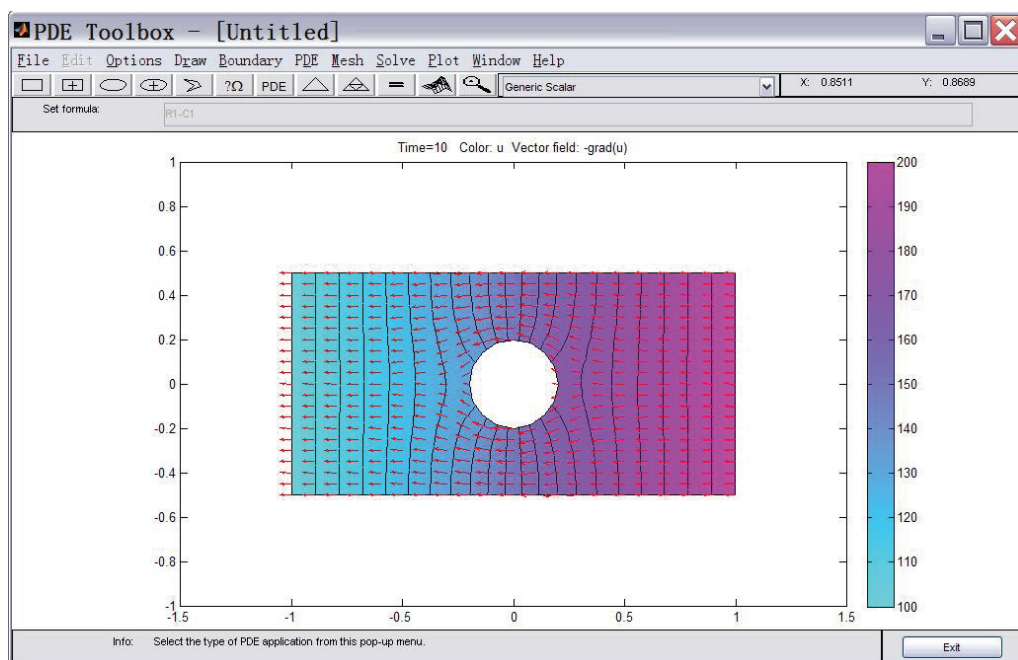


图 2-33 带等高线的彩色矢量场图

2.3.4 二维波动方程

考虑如下二维波动方程的定解问题，取 $a=0.1$ ：

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = a \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -1 < x, y < 1 \\ u|_{x=\pm 1} = u|_{y=\pm 1} = 0, \\ u|_{t=0} = e^{-20(x^2+y^2)}, & \frac{\partial u}{\partial t}|_{t=0} = 0 \end{cases} \quad (2-49)$$

可近似认为上式描述了一个四周被固定的正方形弹性薄膜在中心被拉起并突然释放之后的过程， u 为薄膜上每个点的位移。求解步骤如下：

和前面类似，先启动偏微分工具箱，点击  按钮，按住 Ctrl 键同时点击并拖曳鼠标左键画一个正方形。双击此正方形打开图形参数设置对话框，将 Left、Bottom、Width、Height 分别设置为 -1、-1、2、2，点击 OK。

点击  按钮进入边界模式，双击每一段边界，打开边界条件设置对话框，选择狄利克雷边界条件 “Dirichlet”，填入 $h=1$ 、 $r=0$ 并点击 OK。

点击  按钮打开方程参数设置对话框，方程类型选双曲型 “Hyperbolic”。比较式 (2-43) 和式 (2-49)，可知参数取值如下： $c=0.1$ 、 $a=0$ 、 $f=0$ 、 $d=1$ 。填入参数后点击 OK。

点击 Solve 菜单中的 Parameters 选项，打开求解参数设置对话框。其中，Time 一栏为求解的时间区间，填入 0:1 代表从 $t=0$ 计算到 $t=1$ 。 $u(t_0)$ 一栏为 u 的初始条件，这里填入 $\exp(-20*(x.^2+y.^2))$ 。 $u'(t_0)$ 一栏为 $\partial u/\partial t$ 的初始条件，填入 0 即可。其余两栏为相对误差和绝对误差，一般采用默认设置，最后点击 OK。

点击  按钮初始化三角网格，随后再根据需要多次点击  按钮加密三角网格。

点击  按钮打开画图设置对话框，勾选 Color、Height (3-D plot)、Show mesh，点击 Plot，可得带网格的彩色三维图，如图 2-34 所示为 $t=1$ 时的数值解。

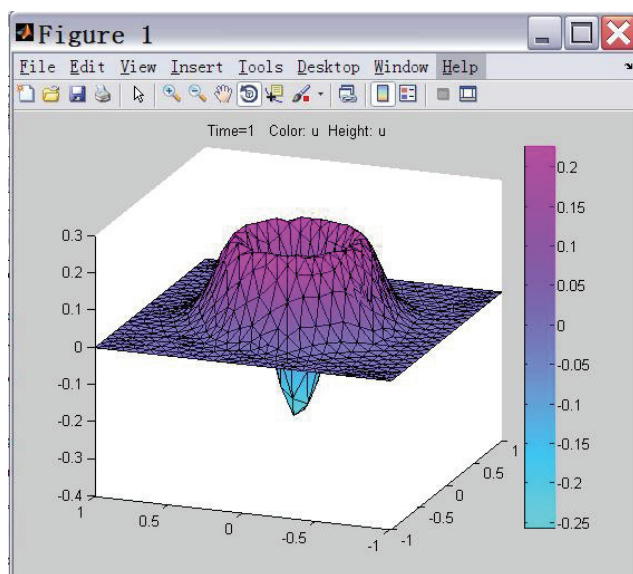


图 2-34 带网格的彩色三维图

如果在 Solve 菜单中 Parameters 选项的 Time 一栏填入 0:0.01:1，并在画图设置对话框勾选 Color、Animation，点击 Plot，将会得到数值解的彩色动画。0:0.01:1 代表在 $t=0$ 和 $t=1$ 之间，每隔 0.01 就输出一幅图作为动画的一帧。Animation 后方的 Options 选项可设置动画帧率、重复播放次数。

2.3.5 二维特征值问题

考虑单位圆内拉普拉斯算符的特征值问题：

$$\begin{cases} -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u = \lambda u, & x^2 + y^2 < 1 \\ u|_{x^2+y^2=1} = 0 \end{cases} \quad (2-50)$$

求解步骤如下：

启动偏微分工具箱，点击  按钮，按住 Ctrl 键同时鼠标左键在 origin 按下并拖曳到合

适位置画一个近似的单位圆。双击此圆打开图形参数设置对话框，将 X-center、Y-center、Radius 分别设置为 0、0、1，点击 OK。

点击  按钮进入边界模式，双击每一段边界，打开边界条件设置对话框，选择狄利克雷边界条件 “Dirichlet”，填入 $h=1$ 、 $r=0$ ，并点击 OK。

点击  按钮打开方程参数设置对话框，方程类型选特征值问题 “Eigenmodes”。比较式 (2-44) 和式 (2-50)，可知参数取值如下： $c=1$ 、 $a=0$ 、 $d=1$ 。填入参数后点击 OK。

点击 Solve 菜单中的 Parameters 选项，打开求解参数设置对话框。在此处可以设置特征值的搜索范围，这里以 0 到 100 之间的特征值为例，填入 [0 100] 并点击 OK。

点击  按钮初始化三角网格，随后再根据需要多次点击  按钮加密三角网格。

点击  按钮求解方程，程序默认显示最小的特征值和其所对应的特征函数。

点击  按钮打开画图设置对话框，勾选 Color、Height (3-D plot)，此外可以在 Eigenvalue 后的下拉菜单中选择特征值（由小到大排列），如图 2-35 所示。点击 Plot 后得到该特征值及其所对应的特征函数的彩色三维图，如图 2-36 所示为第 6 个特征值和特征函数。

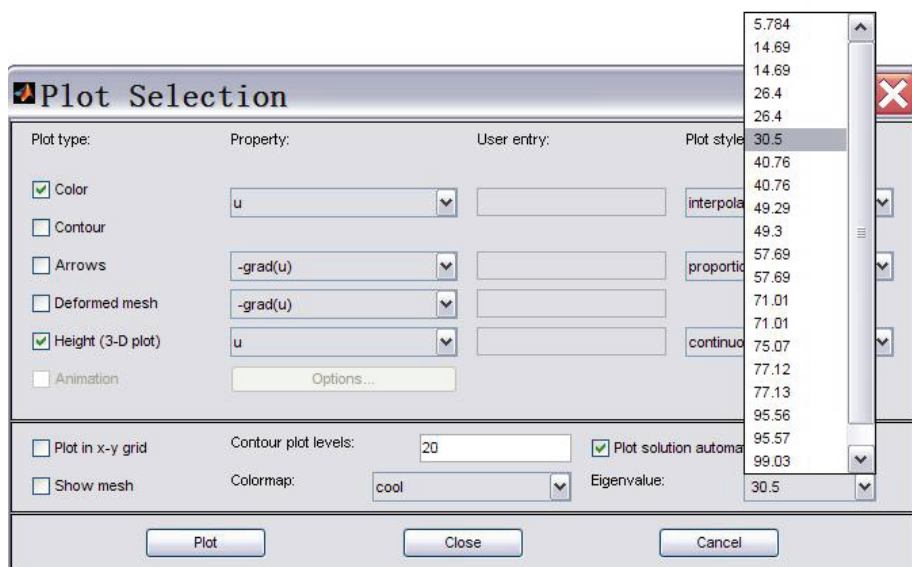


图 2-35 选择特征值

至此，利用偏微分工具箱求解四类基本偏微分方程的方法就介绍完了。实际上，偏微分工具箱还有针对性地为一些领域的具体问题提供了解决方案，如：结构力学、静电学、电磁学、热传导等，但限于篇幅，本书中就不再介绍，更多使用方法参见偏微分工具箱 Help 菜单中的内容。除了图形界面的工具之外，偏微分工具箱也提供了命令行的调用方式，可直接编写 m 文件求解偏微分方程，这使得偏微分工具箱更容易被其他程序调用并进一步处理结果。

偏微分工具箱的界面友好，且边界条件和求解区域可随意选取，这些优点毋须多言。但它依然不可避免地存在一些不足：只能求解特定类型的偏微分方程，为获得高精度必须

提高网格密度，导致运算量提高。这些不足都将在后续章节介绍的谱方法中得到改进。

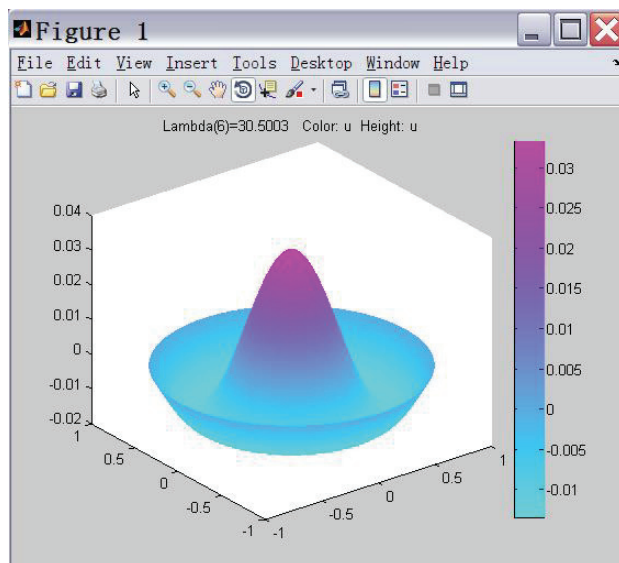


图 2-36 第 6 个特征值所对应的特征函数

中篇 周期性边界条件下的谱方法

第3章 傅里叶谱方法

本章和第4章介绍的求解偏微分方程(组)的方法都包含着周期性边界条件, 尽管周期性边界条件不属于数学物理方法中常见的传统三类边界条件, 但它并不脱离实际。某些科学问题的研究重点不受边界的影响, 如孤子之间的相互作用(非线性薛定谔方程或 KdV 方程)、各向同性的均匀湍流问题等, 周期性边界条件就可以胜任。另外, 一些科学问题本身就具有时空周期性, 如晶格振动问题、能带理论或动物表皮图案的形成问题等。在极坐标系、柱坐标系和球坐标系中, 角度(如: φ 、 θ)是具有周期性的, 对这些情况显然要采用周期性边界条件。

3.1 傅里叶谱方法的原理

3.1.1 快速傅里叶变换

对于在 $(-\infty, \infty)$ 有定义且绝对可积、并在任一有限区间上满足狄利克莱条件的函数 $u(x)$, 傅里叶变换 (Fourier transform) 及其逆变换 (inverse Fourier transform) 定义为:

$$\hat{u}(k) = \int_{-\infty}^{\infty} u(x) e^{-ikx} dx \quad (3-1)$$

$$u(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u}(k) e^{ikx} dk \quad (3-2)$$

上述定义式给出了一个傅里叶变换对 (Fourier transform pair), 本书中将它们记为 $\hat{u}(k)=F[u(x)]$ 和 $u(x)=F^{-1}[\hat{u}(k)]$ 。实际上, 傅里叶变换对的定义并不是唯一的, 两个定义式中的系数可以随意修改, 只要它们的积为 $1/2\pi$ 即可。此外, 定义式中的 e^{-ikx} 和 e^{ikx} (称为积分变换的核) 也可以互换。容易知道, $u(x)$ 先后经过傅里叶变换及其逆变换仍将得到它本身, 即: $u(x)=F^{-1}\{F[u(x)]\}$ 。在物理上, 若研究对象为空间上的信号分布 $u(x)$, x 代表空间坐标, 那么 k 就是波数 (2π 长度上波长的个数, 代表信号在空间上的变化速度), $\hat{u}(k)$ 称为波数谱; 类似地, 若将上式中的坐标 x 和波数 k 代换成时间 t

和角频率 ω ，研究对象就成了时间上的信号 $u(t)$ ，其傅里叶变换就是频谱 $\hat{u}(\omega)$ ，通过类比可知波数 k 其实就是一种空间上的频率。傅里叶变换及逆变换是空域（或时域）与频域之间的转换工具。

当函数 u 为二维函数时，将 x 方向上和 y 方向上的傅里叶变换记为 $F_x[\]$ 和 $F_y[\]$ ，并在变换后的函数上加“ $\wedge$ ”、“ $\sim$ ”进行区分，如： $\hat{u}(k_x, y) = F_x[u(x, y)]$ ， $\tilde{u}(x, k_y) = F_y[u(x, y)]$ ，其中 k_x 、 k_y 为 x 、 y 方向的波数。在二维情况下，默认 $F[\]$ 和 $F^{-1}[\]$ 分别代表二维傅里叶变换及逆变换，即 $F[\] = F_y\{F_x[\]\}$ 或 $F_x\{F_y[\]\}$ 、 $F^{-1}[\] = F_y^{-1}\{F_x^{-1}[\]\}$ 或 $F_x^{-1}\{F_y^{-1}[\]\}$ ，并有：

$$\begin{aligned}\hat{u}(k_x, k_y) &= F[u(x, y)] \\ u(x, y) &= F^{-1}[\hat{u}(k_x, k_y)]\end{aligned}\quad (3-3)$$

通常来讲，若不对“傅里叶变换”加任何限定，那么它指的就是连续傅里叶变换，也就是针对定义在无限区间内的连续函数 $u(x)$ 的傅里叶变换，这是在理想条件下的数学定义。在实际应用中，尤其是计算机的信号采样、信号处理当中，信号是离散的、有限的，离散傅里叶变换（discrete Fourier transform）就是针对这一情况提出的。在 Matlab 中，对于序列 $u_1, \dots, u_j, \dots, u_N$ 的离散傅里叶变换及逆变换定义为：

$$\hat{u}_k = \sum_{j=1}^N u_j e^{\frac{-2\pi(j-1)(k-1)i}{N}}, \quad k=1, \dots, N \quad (3-4)$$

$$u_j = \frac{1}{N} \sum_{k=1}^N \hat{u}_k e^{\frac{2\pi(j-1)(k-1)i}{N}}, \quad j=1, \dots, N \quad (3-5)$$

同样，上述定义中的归一化系数也可以有其他选择，但它们的乘积必须为 $1/N$ 。如果将序列 $u_1, \dots, u_j, \dots, u_N$ 看作等间隔空间（时间）点上的信号幅度值，那么经过离散傅里叶变换得到的序列 $\hat{u}_1, \dots, \hat{u}_k, \dots, \hat{u}_N$ 就是其相应的频谱信息。通过定义式(3-4)和(3-5)容易得到 $\hat{u}_k = \hat{u}_{k+N}$ 和 $u_j = u_{j+N}$ ，这就是说，离散傅里叶变换已经隐含了周期性边界条件，它假设待做离散傅里叶变换的离散信号在周期为 2π 的周期域上。在实际运算中，人们不直接采用上述离散傅里叶变换的定义式进行计算，而是使用快速傅里叶变换（fast Fourier transform）算法，此算法是二十世纪六十年代中期由 Cooley 和 Tukey 提出的，被誉为二十世纪最伟大的十大算法之一。离散傅里叶变换定义式的运算量（arithmetical operation）是 $O(N^2)$ ，而快速傅里叶变换可将运算量降低到 $O(N \log N)$ 。特别是当 N 很大的时候， $N \log N$ 的增长速度仅接近于 N ，可以大大地节省计算时间。但是，为了获得较高的运算速度，待变换序列的元素数量必须是 2^n 个，即其个数必须为 2、4、8、16、32、64……。

在 Matlab 中，fft 和 ifft 函数实现基于快速傅里叶变换算法的一维离散傅里叶变换及逆变换。fft 函数的一般调用语法为：

fft(u)

若 u 为一向量，则 `fft` 返回其离散傅里叶变换的结果。若 u 为一矩阵，则 `fft` 返回矩阵中每一列的离散傅里叶变换。若要计算矩阵 u 中每一行的离散傅里叶变换，只需调用 `fft(u,[],2)` 即可，它速度比 `fft(u.')` 更快一些。`ifft` 函数的调用语法与 `fft` 函数完全一样。

若输入 `fft` 函数的空域信号序列 $u_1, \dots, u_j, \dots, u_N$ 在 x 轴上的坐标间隔是 L/N ，相应的横坐标 x 可认为是：

$$-L/2, -L/2+L/N, \dots, L/2-2L/N, L/2-L/N$$

注意，`fft` 函数输出的序列 $\hat{u}_1, \dots, \hat{u}_k, \dots, \hat{u}_N$ 所对应的频率并不是按照从小到大的顺序排列的，而是非负频率对应于前半部分序列，负频率对应于后半部分序列，即：

$$0, 2\pi/L, 4\pi/L, \dots, (N-4)\pi/L, (N-2)\pi/L, -N\pi/L, -(N-2)\pi/L, \dots, -4\pi/L, -2\pi/L$$

因此 Matlab 提供了 `fftshift` 函数，用于调整 `fft` 输出序列的顺序。也就是说，`fftshift(fft(u))` 所对应的频率才是按照递增顺序排列的：

$$-N\pi/L, -(N-2)\pi/L, \dots, -2\pi/L, 0, 2\pi/L, \dots, (N-4)\pi/L, (N-2)\pi/L$$

空域坐标与频域坐标的关系如表 3-1 所示，不难发现，空域上的两点间隔和频域区间长度成反比，乘积为 2π ，频域上的两点间隔和空域区间长度的关系亦如此。

表 3-1 空域坐标与频域坐标的关系

	两点间隔	区间长度
空域坐标	L/N	L
频域坐标	$2\pi/L$	$2\pi N/L$

此外还有 `ifftshift` 函数用于进行与 `fftshift` 函数相反的操作。实际上，`fftshift` 函数的作用仅是让 `fft` 的输出结果变得更易于观察，若还需要用 `ifft` 函数将频谱信息还原回空域信号，就必须在使用 `fftshift` 之后再使用 `ifftshift` 函数将序列的顺序恢复，否则得不到正确结果。如果不是为了分析频谱信息，而是用下文即将介绍的傅里叶谱方法处理导数和微分问题，则大可不必使用 `fftshift` 和 `ifftshift` 函数调整序列的顺序，以减少代码冗余、节约时间。

基于快速傅里叶变换算法的二维离散傅里叶变换及逆变换由 `fft2`、`ifft2` 函数实现，一般调用语法为：

`fft2(u)`

它返回矩阵 u 的二维傅里叶变换，等效于 `fft(fft(u),[],2)`。`ifft2` 与 `fft2` 的调用语法一样，就不再赘述。

`fftshift` 函数的调用语法为：

fftshift(u)

若 u 为一向量，它将 u 左右两半互换并返回。若 u 为一矩阵，则它将 u 的第 1、3 象限互换以及第 2、4 象限互换并返回。此外，若要互换矩阵 u 的上下两半，可以调用 `fftshift(u,1)`。要互换矩阵 u 的左右两半，只需调用 `fftshift(u,2)`。`ifftshift` 函数与 `fftshift` 函数功能完全相反，调用语法相同，这里不再重复。

接下来以实例说明 `fft` 等函数的用法。众所周知， sinc^2 函数的傅里叶变换是 `tri` 函数（三角形函数），即：

$$F[\text{sinc}^2(ax)] = \frac{1}{\sqrt{2\pi}a^2} \text{tri}\left(\frac{k}{2\pi a}\right) \quad (3-6)$$

取 $a=1$ ，实现函数 $\text{sinc}^2(x)$ 傅里叶变换的代码如下：

程序 3-1

```
clear all; close all;
L=20; N=128;
x=L/N*[-N/2:N/2-1];
k=2*pi/L*[-N/2:N/2-1];
u=sinc(x).^2;
ut=fft(u);
subplot(2,2,1)
plot(x,u,'k','LineWidth',1.5), xlabel x, ylabel u=sinc^2(x)
subplot(2,2,2)
plot(abs(ut),'k','LineWidth',1.5)
axis([1 N 0 7]), set(gca,'xtick',[]), ylabel abs(fft(u))
subplot(2,2,3)
plot(k,abs(fftshift(ut)),'k','LineWidth',1.5)
axis([-20 20 0 7]), xlabel k, ylabel abs(fftshift(fft(u)))
subplot(2,2,4)
plot(x,ifft(ifftshift(fftshift(ut))),'k','LineWidth',1.5)
xlabel x, ylabel ifft(ifftshift(fftshift(fft(u))))
```

代码的执行结果如图 3-1 所示， sinc^2 形信号经过 `fft` 函数变换所得频谱的顺序出现颠倒，低频成分在两端，高频成分在中间。随后经过 `fftshift` 函数调整顺序，低频成分被置于频谱中间，其中的求绝对值函数 `abs` 用来得到频谱的振幅。再使用 `ifftshift` 函数还原频谱顺序并用 `ifft` 函数做傅里叶逆变换，就得到了最初的 sinc^2 函数。注意到三角形函数的幅度与式（3-6）不相符，这并不是计算错误，而是快速傅里叶变换对 `fft` 和 `ifft` 的系数选取不同造成的，这一差异丝毫不影响后面即将讨论的傅里叶谱方法。

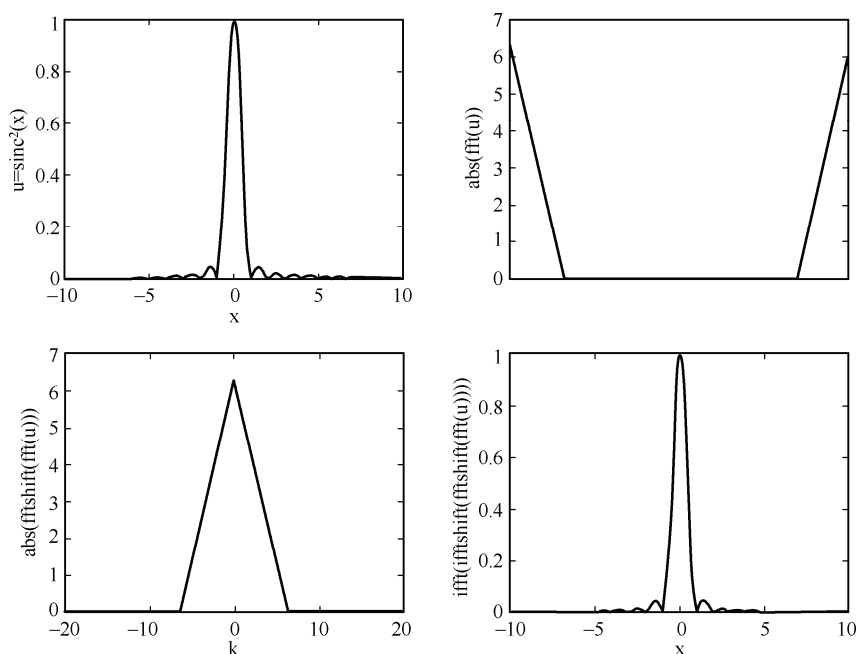


图 3-1 fft、ifft、fftshift、ifftshift 函数的用法

3.1.2 求导、积分与傅里叶谱方法

对于 $F[u'(x)]$ ，由傅里叶变换的定义和分部积分法，得到：

$$F[u'(x)] = \int_{-\infty}^{\infty} u'(x) e^{-ikx} dx = u(x) e^{-ikx} \Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} u(x) (-ik) e^{-ikx} dx \quad (3-7)$$

当 $|x| \rightarrow \infty$ 时， $u(x) \rightarrow 0$ ，则：

$$F[u'(x)] = ik \int_{-\infty}^{\infty} u(x) e^{-ikx} dx = ik F[u(x)] \quad (3-8)$$

类似地，可以得到：

$$F[u^{(n)}(x)] = (ik)^n F[u(x)] \quad (3-9)$$

其中 $u^{(n)}(x)$ 代表 $u(x)$ 的 n 阶导数。上式的意义在于：函数的求导运算在傅里叶变换的作用下，可转化为相对简单的代数运算，即： $u^{(n)}(x) = F^{-1}\{(ik)^n F[u(x)]\}$ 。正是基于此原理，傅里叶谱方法（Fourier spectral method）利用傅里叶变换将偏微分方程中空域（space domain）或时域（time domain）上的求导运算简化为频域（spectral domain）上的代数运算，求解后再通过傅里叶逆变换得到空域或时域上的结果。在代码层面上，Matlab 提供的快速傅里叶变换函数 `fft`、逆变换函数 `ifft` 以及强大的矩阵运算能力也为简洁、优雅地实现傅里叶谱方法奠定了基础。

3.1.1 小节已经提到，`fft` 输出结果在频域上的顺序是颠倒的，经过 `fft` 函数变换后的序

列在 k 轴上所对应的坐标是：

$$0, 2\pi/L, 4\pi/L, \dots, (N-4)\pi/L, (N-2)\pi/L, -N\pi/L, -(N-2)\pi/L, \dots, -4\pi/L, -2\pi/L$$

利用傅里叶谱方法求导时，在代码中通常使用顺序颠倒的频域坐标，以求序列 u 的一阶导数数值解为例： $\text{ifft}(i*k.*\text{fft}(u))$ 。其中，变量 k 是由 Matlab 语句 $(2*\pi/L)*[0:N/2-1 -N/2:-1]$ 生成的顺序颠倒的频域坐标，这样做是为了免去调用 fftshift 和 ifftshift 的步骤。

下面用实例说明傅里叶谱方法求 $u(x)=e^{\sin(\pi x)}$ 的 1、2、3 阶导数的过程，并与解析解 $u'(x)=\pi\cos(\pi x)e^{\sin(\pi x)}$ 、 $u''(x)=\pi^2 e^{\sin(\pi x)}[\cos^2(\pi x)-\sin(\pi x)]$ 、 $u'''(x)=\pi^3 e^{\sin(\pi x)}\cos(\pi x)[\cos^2(\pi x)-3\sin(\pi x)-1]$ 相比较，得到的最大误差分别在 10^{-14} 、 10^{-13} 、 10^{-11} 数量级，如图 3-2 所示。代码如下：

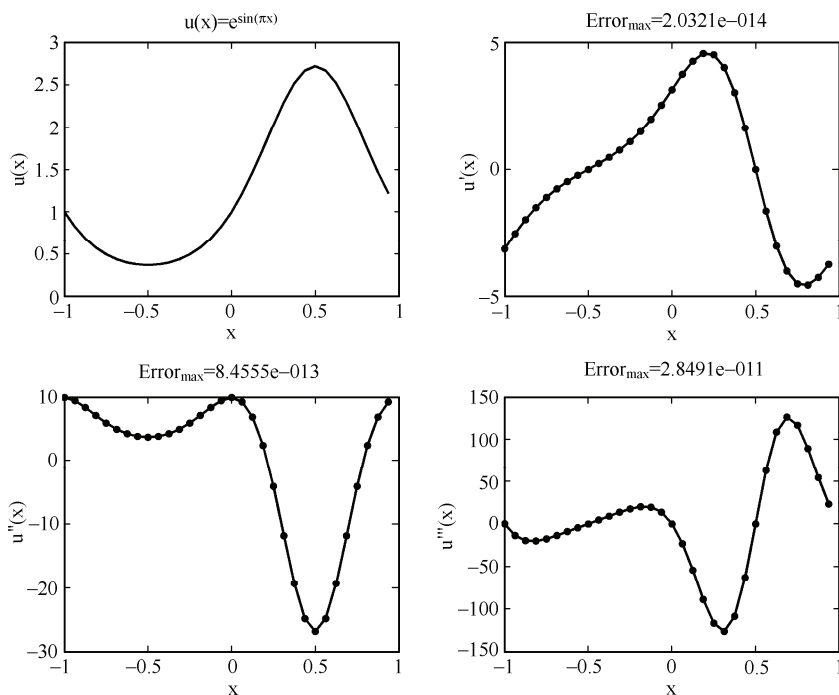


图 3-2 傅里叶谱方法求 $u(x)=e^{\sin(\pi x)}$ 的 1、2、3 阶导数并与解析解比较

程序 3-2

```
clear all; close all;
L=2; N=32;
x=L/N*[-N/2:N/2-1];
k=2*pi/L*[0:N/2-1 -N/2:-1];
u=exp(sin(pi*x)); ut=fft(u);
%导数的精确解
du_exact(:,1)=pi*cos(pi*x).*u;
du_exact(:,2)=pi^2*(cos(pi*x).^2-sin(pi*x)).*u;
```

```

du_exact(:,3)=pi^3*cos(pi*x).*(cos(pi*x).^2-3*sin(pi*x)-1).*u;
%谱方法求导
du_Fourier(:,1)=ifft(i*k.*ut);
du_Fourier(:,2)=ifft((i*k).^2.*ut);
du_Fourier(:,3)=ifft((i*k).^3.*ut);
error=max(abs(du_exact-du_Fourier));
%画图
labels={'u'(x)','u'''(x)','u''''(x)'};
for n=1:4
    subplot(2,2,n)
    if n==1
        plot(x,u,'k','LineWidth',1.5)
        xlabel x, ylabel u(x), title('u(x)=e^{sin(\pix)}')
    else
        plot(x,du_exact(:,n-1),'k',x,du_Fourier(:,n-1),'r' ...
            , 'MarkerSize',13, 'LineWidth',1.5)
        title(['Error_{max}=' num2str(error(n-1))])
        xlabel x, ylabel(labels(n-1))
    end
end
end

```

式(3-9)也可写为:

$$u(x) = F^{-1} \left\{ \frac{F[u^{(n)}(x)]}{(ik)^n} \right\} \quad (3-10)$$

若把 $u(x)$ 看成是对 $u^{(n)}(x)$ 做 n 次积分的结果, 式(3-10)就具有了求不定积分的意义。当然, 这里暂时没有考虑积分常数的问题。与式(3-9)比较可知, 求 n 阶导数相当于在频域上乘以 $(ik)^n$, 相应地, 做 n 次积分就是在频域上除以 $(ik)^n$ 。需要强调的是, 如此计算积分会出现一个问题, 那就是 k 为 0 时会导致式(3-10)出现无穷大。在编写程序时有两种解决办法:

(1) 将 k 的 0 值修改为很小的数, 如: 10^{-6} 。

(2) 在分母上加一个很小的数, 防止分母为 0, 即: $(ik)^n + \text{eps}$ 。其中, eps 是 Matlab 中的浮点相对误差限, 它的值在 10^{-16} 数量级。这样, 就通过引入微小的误差避免了分母为 0 的问题。

3.1.3 傅里叶谱方法的步骤

待求解的偏微分方程的普遍形式为:

$$\frac{\partial^n u}{\partial t^n} = Lu + N(u) \quad (3-11)$$

其中, $u(x, t)$ 为 x, t 的函数, L 代表线性算符 (linear operator), $N(u)$ 为非线性项 (nonlinear terms)。已知初始条件为 $u(x, t_0)$, 在周期性边界条件下求某一时刻的 $u(x, t)$ 。对于这样的问题, 总是可以通过变量代换的办法将等号左边对 t 的 n 阶导数降为 1 阶, 详见 1.1.6 小节。所以下面以等号左边是 $\partial u / \partial t$ 的偏微分方程为例进行讨论, 这与式 (3-11) 降阶后的偏微分方程组求解方法大同小异。

在 x 域上对以下方程做傅里叶变换:

$$\frac{\partial u}{\partial t} = Lu + N(u) \quad (3-12)$$

得到:

$$\frac{\partial \hat{u}}{\partial t} = \alpha(k) \hat{u} + F[N(u)] \quad (3-13)$$

其中, $\hat{u}(k, t)$ 代表 $u(x, t)$ 在 x 域上的傅里叶变换。为了展现对线性算符 L 及非线性项 $N(u)$ 做傅里叶变换的细节, 这里以 $L = a \cdot \partial^2 / \partial x^2 + b \cdot \partial / \partial x + c$, $N(u) = u^3 + u^3 \cdot \partial^2 u / \partial x^2 + f(x) \cdot \partial u / \partial x$ 为例进行分析, a, b 和 c 分别为常数。

对线性部分, 有 $F[Lu] = [a(ik)^2 + b(ik) + c]\hat{u}$, 则式 (3-13) 中的 $\alpha(k) = a(ik)^2 + b(ik) + c = -ak^2 + ibk + c$ 。对非线性部分, 需要将 u 、 $\partial u / \partial x$ 和 $\partial^2 u / \partial x^2$ 写为 $F^{-1}[\hat{u}]$ 、 $F^{-1}[ik\hat{u}]$ 和 $F^{-1}[(ik)^2\hat{u}]$, 则有 $F[N(u)] = F[u^3 + u^3 \cdot \partial^2 u / \partial x^2 + f(x) \cdot \partial u / \partial x] = F\{F^{-1}[\hat{u}]^3 + F^{-1}[\hat{u}]^3 \cdot F^{-1}[(ik)^2\hat{u}] + f(x) \cdot F^{-1}[ik\hat{u}]\}$ 。

这样, 式 (3-12) 中 $u(x, t)$ 对 x 的偏导数就都通过傅里叶变换及逆变换简化为式 (3-13) 中 $\hat{u}(k, t)$ 和 k 的代数运算了, 然后再将 $\hat{u}$ 和 k 离散化, 偏微分方程就简化成了常微分方程组。数值计算 $\partial \hat{u}(k, t) / \partial t$ 最直接的方法就是调用 Matlab 的 ode 系列函数, 优先选择 ode45 函数, 最后用 ifft 函数将频域上的计算结果 $\hat{u}(k, t)$ 变换回待求的 $u(x, t)$ 。

综上, 利用傅里叶谱方法数值计算偏微分方程 (组) 步骤如图 3-3 所示, 具体过程如下:

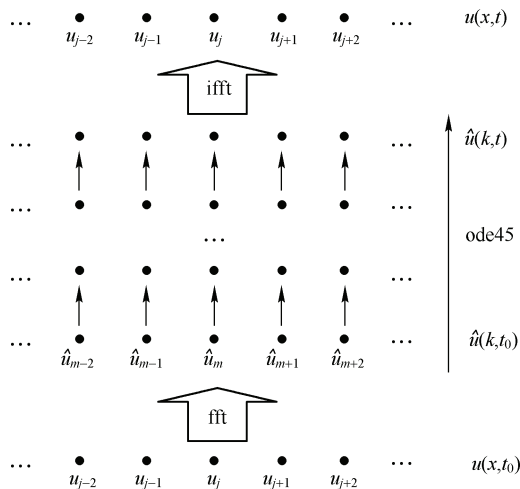


图 3-3 一维情况下的傅里叶谱方法计算过程

(1) 对于形如式 (3-11) 的偏微分方程 (组), 通过变量代换将其中的 $\partial^n u / \partial t^n$ 降为 1 阶导数, 若 $n=1$, 略过此步。

(2) 在 x 域上对偏微分方程 (组) 做傅里叶变换, 则线性项中的 $\partial^n u / \partial x^n$ 直接变为 $(ik)^n \hat{u}$, 并利用 $u = F^{-1}[\hat{u}]$ 、 $\partial^n u / \partial x^n = F^{-1}[(ik)^n \hat{u}]$ 代换非线性项中的 u 、 $\partial^n u / \partial x^n$, 得到关于 $\hat{u}(k, t)$ 的微分方程 (组)。

(3) 用时间步进法 (如龙格-库塔法等) 数值计算离散化的关于 $\hat{u}(k, t)$ 的微分方程组, 默认边界条件为周期性边界条件, 初始条件 $\hat{u}(k, t_0) = F[u(x, t_0)]$ 。

(4) 将上一步得到的结果从频域变换回空域, 即: $u(x, t) = F^{-1}[\hat{u}(k, t)]$ 。离散傅里叶变换及逆变换由 `fft`、`ifft` 函数实现。

若方程在空间上再增加一个维度, 式 (3-12) 的等号右边包含 $u(x, y, t)$ 的 $\partial^n / \partial x^n$ 和 $\partial^n / \partial y^n$, 则傅里叶谱方法的计算步骤基本不变, 如示意图 3-4 所示。唯一不同的是需要对方程做二维傅里叶变换, 使用 `fft2`、`ifft2` 函数代替 `fft`、`ifft` 函数, 并利用 $\partial^n / \partial x^n \rightarrow (ik_x)^n$ 、 $\partial^n / \partial y^n \rightarrow (ik_y)^n$ 和关系式 (3-14) 进行转化, 其中 k_x 、 k_y 为 x 、 y 方向的波数。式中 $F[\]$ 、 $F^{-1}[\]$ 代表二维傅里叶变换及逆变换, u 上加 “ $\sim$ ” 和 “ $\wedge$ ” 代表先后对函数 u 做 x 、 y 方向上的傅里叶变换 (即二维傅里叶变换)。

$$u = F^{-1}[\hat{u}], \quad \frac{\partial^n u}{\partial x^n} = F^{-1}[(ik_x)^n \hat{u}], \quad \frac{\partial^n u}{\partial y^n} = F^{-1}[(ik_y)^n \hat{u}] \quad (3-14)$$

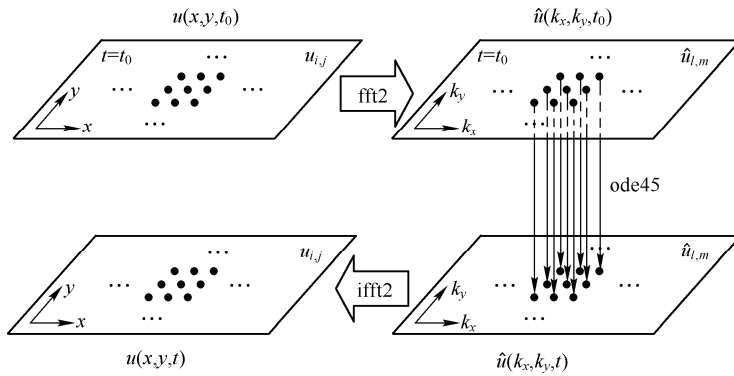


图 3-4 二维情况下的傅里叶谱方法计算过程

为了更贴近实际地说明具体计算细节, 后面章节将给出形如式 (3-11) 和 (3-12) 的偏微分方程 (组) 的求解实例。

需要强调的是, 使用傅里叶谱方法求解偏微分方程 (组) 隐含着周期性边界条件。以序列 $u_1, \dots, u_j, \dots, u_N$ 为例, 在周期性边界条件下, 可以等效认为有 $u_{mN+j} = u_j$ 的关系 (m 为任意整数), 也就是说一端边界处的函数值将对另一端边界处的函数值产生影响。周期性边界条件可以将具有时空周期性的物理问题简化为单元进行处理, 但对于一些特定的非周期性问题, 则需要修改其他条件 (计算区间的范围、参数、初始条件等) 来确保边界处的函数值恒为 0 或某一特定常数, 以排除相邻周期性的干扰, 得到正确结果。

3.1.4 滤波法

使用傅里叶谱方法时，若频域上关于 $\hat{u}$ 的偏微分方程具有刚性，那么其求解速度将大大降低。本小节介绍的滤波法可以提高一部分这种刚性方程的求解速度。

首先分析偏微分方程在傅里叶谱方法中是如何被引入刚性的，以超扩散方程 (hyper-diffusion equation) 为例，见式 (3-15)：

$$\frac{\partial u}{\partial t} = -\frac{\partial^4 u}{\partial x^4} \quad (3-15)$$

对其做傅里叶变换后，得：

$$\frac{\partial \hat{u}}{\partial t} = -(\mathrm{i}k)^4 \hat{u} = -k^4 \hat{u} \quad (3-16)$$

如果 x 域上的计算区间宽度 $L=2$ ，区间上数据点的数量 $N=256$ ，那么 $|k|$ 的最大值为 $N\pi/L \approx 400$ ，则 $(k_{\max})^4 \approx 2.6 \times 10^{10}$ ，假设 $\hat{u}$ 在高频 $k_{\max}$ 处的数量级很小，不妨令 $\hat{u}(k_{\max}, t) = 10^{-6}$ ，有：

$$\frac{\partial \hat{u}(k_{\max}, t)}{\partial t} = -k_{\max}^4 \cdot \hat{u}(k_{\max}, t) \approx -2.6 \times 10^4 \quad (3-17)$$

等号右边的数量绝对值很大，这将导致 ode45（其他变步长算法亦如此）在计算 $\partial \hat{u} / \partial t$ 时必须采用很小的步长 Δt 才能把误差控制在可接受范围内，因此显著增加了运算时间，这就是式 (3-16) 的刚性带来的计算困难。此外，采用过小的 L 、过大的 N 都会加剧这一问题。

一般地，对式 (3-12) 进行傅里叶变换得到式 (3-13) 之后，如果线性项 $\alpha(k)\hat{u}$ 中的高频部分造成了方程的刚性问题，可用滤波法解决。方法如下：偏微分方程的线性部分是可以解析求解的，因此在式 (3-13) 两端乘上积分因子 $e^{-\alpha(k)t}$ ，整理得：

$$e^{-\alpha(k)t} \cdot \frac{\partial \hat{u}}{\partial t} - \alpha(k) e^{-\alpha(k)t} \cdot \hat{u} = e^{-\alpha(k)t} F[N(u)] \quad (3-18)$$

将前两项合并，即：

$$\frac{\partial [e^{-\alpha(k)t} \hat{u}]}{\partial t} = e^{-\alpha(k)t} F[N(u)] \quad (3-19)$$

令 $\hat{w} = e^{-\alpha(k)t} \hat{u}$ ，则：

$$\begin{cases} \frac{\partial \hat{w}}{\partial t} = e^{-\alpha(k)t} F[N(u)] \\ \hat{w} = e^{-\alpha(k)t} \hat{u} \end{cases} \quad (3-20)$$

也可以写成以下形式，数值计算出式 (3-20) 或式 (3-21) 的结果再利用关系 $u = F^{-1}[e^{\alpha(k)t} \hat{w}]$ 将 $\hat{w}$ 化为 u 即可。

$$\frac{\partial \hat{w}}{\partial t} = e^{-\alpha(k)t} F \left\{ N \left\{ F^{-1} \left[e^{\alpha(k)t} \hat{w} \right] \right\} \right\} \quad (3-21)$$

这样就通过引入 $\hat{w}$ 将线性项中的刚性削弱了。然而非线性项中也可能存在刚性，以致影响运算速度，所以滤波法只是对傅里叶谱方法中的一部分刚性方程有效。实际上，较之傅里叶谱方法，滤波法对计算时间的节省只有在空域上最高阶导数的维度大于 2 的时候才能体现出来。

3.2 傅里叶谱方法求解基本偏微分方程（组）

3.2.1 一维波动方程

对于一根两端固定、没有受到任何外力的弦，若只研究其中的一段，在不太长的时间里，固定端来不及对这段弦产生影响，则可以认为固定端是不存在的，弦的长度为无限大。这种无界（ $-\infty < x < \infty$ ）弦的自由振动由式（3-22）描述。

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \quad (3-22)$$

如果保证数值计算的区间足够大，在一定时间内，弦的振动范围始终没有超出计算区间（或可以近似地这么认为），那么就能够放心地使用周期性边界条件。取 $a=1$ ，初始条件为：

$$u|_{t=0} = 2\text{sech}(x), \quad \left. \frac{\partial u}{\partial t} \right|_{t=0} = 0 \quad (3-23)$$

在数学物理方法中，无界弦的自由振动可由行波法求出解析解，即达朗贝尔公式。根据达朗贝尔公式，从 $t=0$ 开始， u 的初始状态 $2\text{sech}(x)$ 将分裂为两个 sech 形的波，分别向两边以速度 a 传播出去，即正行波和反行波。下面用傅里叶谱方法求解无界弦的自由振动问题，并与达朗贝尔公式的预测进行比较。首先引入函数 v 对式（3-22）进行降阶：

$$\begin{cases} \frac{\partial u}{\partial t} = v \\ \frac{\partial v}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} \end{cases} \quad (3-24)$$

对上式等号两边做傅里叶变换，化为偏微分方程组：

$$\begin{cases} \frac{\partial \hat{u}}{\partial t} = \hat{v} \\ \frac{\partial \hat{v}}{\partial t} = -a^2 k^2 \hat{u} \end{cases} \quad (3-25)$$

这样就可以用 ode45 求解了，详细代码如下：

程序 3-3

主程序代码如下：

```
clear all; close all;
L=80; N=256;
x=L/N*[-N/2:N/2-1];
k=(2*pi/L)*[0:N/2-1 -N/2:-1].';
%初始条件
u=2*sech(x); ut=fft(u);
vt=zeros(1,N); uvt=[ut vt];
%求解
a=1; t=0:0.5:20;
[t,uvtsol]=ode45('wave1D',t,uvt,[],N,k,a);
usol=ifft(uvtsol(:,1:N),[],2);
%画图
p=[1 11 21 41];
for n=1:4
    subplot(5,2,n)
    plot(x,usol(p(n),:),'k','LineWidth',1.5), xlabel x, ylabel u
    title(['t=' num2str(t(p(n)))]), axis([-L/2 L/2 0 2])
end
subplot(5,2,5:10)
waterfall(x,t,usol), view(10,45)
xlabel x, ylabel t, zlabel u, axis([-L/2 L/2 0 t(end) 0 2])
```

文件 wave1D.m 代码如下：

```
function duvt=wave1D(t,uvt,dummy,N,k,a)
ut=uvt(1:N); vt=uvt(N+[1:N]);
duvt=[vt; -a^2*(k).^2.*ut];
```

计算结果如图 3-5 所示，初始状态的波形分裂成两半，并分别向 x 轴正方向和负方向以速度 a 运动，这和达朗贝尔公式给出的结论是一致的。

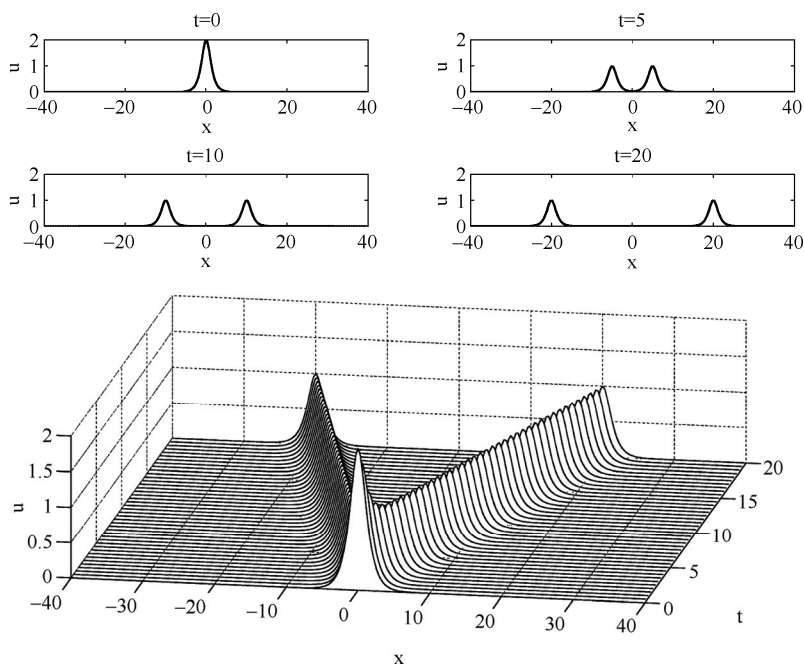


图 3-5 一维波动方程的行波解

3.2.2 二维波动方程

将 3.2.1 小节中的一维无界弦自由振动方程推广到二维空间上, 就得到了描述无界 $(-\infty < x, y < \infty)$ 弹性薄膜的波动方程:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u \quad (3-26)$$

取 $a=1$, 初始条件为:

$$u|_{t=0} = e^{-20[(x-0.4)^2 + (y+0.4)^2]} + e^{-20[(x+0.4)^2 + (y-0.4)^2]}, \quad \frac{\partial u}{\partial t} \Big|_{t=0} = 0 \quad (3-27)$$

可以这样理解上述初始条件的物理意义: 两手抓住弹性薄膜的两个位置, 分别提起, 使薄膜上形成两个峰, 在 $t=0$ 时刻突然松手。根据生活常识可以预料到, 这两个位置的薄膜将来回振动, 与此同时, 产生的波向四周传播, 而且波与波会在相遇处叠加。

为便于求解, 引入函数 v 对式 (3-26) 进行降阶, 得:

$$\begin{cases} \frac{\partial u}{\partial t} = v \\ \frac{\partial v}{\partial t} = a^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u \end{cases} \quad (3-28)$$

对上式等号两边做傅里叶变换，得到常微分方程组：

$$\begin{cases} \frac{\partial \hat{u}}{\partial t} = \hat{v} \\ \frac{\partial \hat{v}}{\partial t} = -a^2 (k_x^2 + k_y^2) \hat{u} \end{cases} \quad (3-29)$$

接下来用 ode45 求解即可，代码如下：

程序 3-4

主程序代码如下：

```
clear all; close all;
L=4; N=64;
x=L/N*[-N/2:N/2-1]; y=x;
kx=(2*pi/L)*[0:N/2-1 -N/2:-1]; ky=kx;
[X,Y]=meshgrid(x,y);
[kX,kY]=meshgrid(kx,ky);
K2=kX.^2+kY.^2;
%初始条件
u=exp(-20*((X-0.4).^2+(Y+0.4).^2))+exp(-20*((X+0.4).^2+(Y-0.4).^2));
ut=fft2(u); vt=zeros(N); uvt=[ut(:); vt(:)];
%求解
a=1; t=[0 0.25 0.5 1];
[t,uvtSol]=ode45('wave2D',t,uvt,[],N,K2(:),a);
%画图
for n=1:4
    subplot(2,2,n)
    mesh(x,y,ifft2(reshape(uvtSol(n,1:N^2),N,N))), view(10,45)
    title(['t=' num2str(t(n))]), axis([-L/2 L/2 -L/2 L/2 0 1])
    xlabel x, ylabel y, zlabel u
end
```

文件 wave2D.m 代码如下：

```
function duvt=wave2D(t,uvt,dummy,N,K2,a)
ut=uvt(1:N^2); vt=uvt(N^2+[1:N^2]);
duvt=[vt; -a^2*K2.*ut];
```

程序输出结果如图 3-6 所示，它反映了弹性薄膜上的波向四周传播的过程。

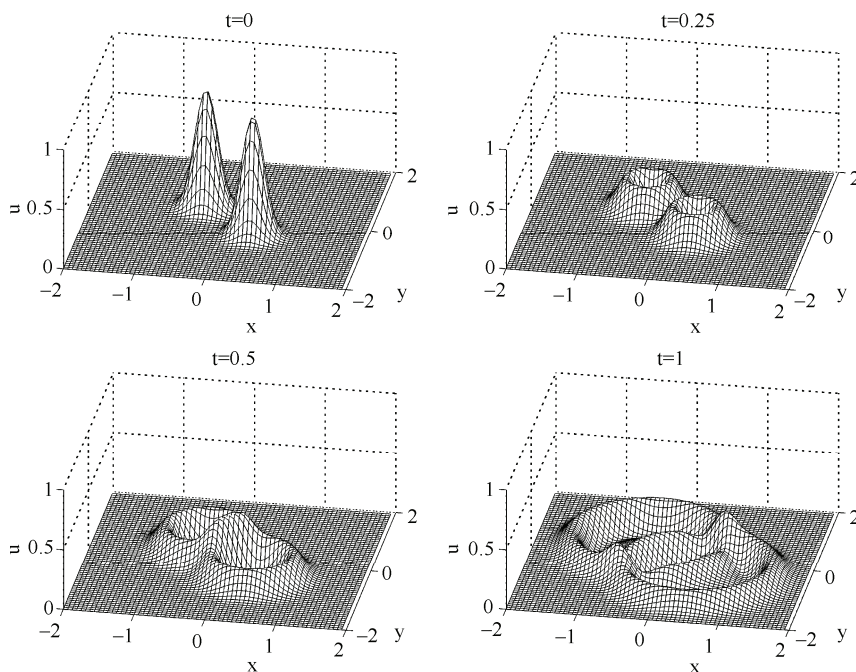


图 3-6 二维波动方程的数值解

3.2.3 一维非线性薛定谔方程

非线性薛定谔方程 (nonlinear Schrödinger equation) 在非线性物理学中的地位是举足轻重的, 它广泛应用于非线性光纤光学、等离子体物理、凝聚态物理等领域, 它的归一化形式如下:

$$\frac{\partial u}{\partial t} = \frac{i}{2} \frac{\partial^2 u}{\partial x^2} + i|u|^2 u \quad (3-30)$$

其中, u 为复振幅, x, t 代表空间坐标、时间。对式 (3-30) 做傅里叶变换, 得偏微分方程:

$$\frac{\partial \hat{u}}{\partial t} = -\frac{ik^2}{2} \hat{u} + iF \left\{ \left| F^{-1} \{ \hat{u} \} \right|^2 F^{-1} \{ \hat{u} \} \right\} \quad (3-31)$$

由逆散射方法可知, 当初始条件为 $u(x, 0) = N \operatorname{sech}(x)$ 时, 如果 $N=1$, $|u|$ 的形状在传播过程中保持不变, 如果 $N \geq 2$, $|u|$ 的形状则进行周期性的变化。 N 称为孤子的阶数, $N=1$ 对应的是基态孤子, $N \geq 2$ 对应的是高阶孤子。这里取 $N=2$ 的情况作初始条件, 使用傅里叶谱方法求解式 (3-31) 的代码如下:

程序 3-5

主程序代码如下：

```
clear all; close all;
L=20; N=256;
x=L/N*[-N/2:N/2-1];
k=(2*pi/L)*[0:N/2-1 -N/2:-1].';
%初始条件
u=2*sech(x); ut=fft(u);
%求解
t=0:0.1:5;
[t,utsol]=ode45('NLSE',t,ut,[],k);
usol=ifft(utsol,[],2);
%画图
subplot(2,2,1)
waterfall(x,t,abs(usol))
axis([-10 10 0 5 0 4]), xlabel x, ylabel t, zlabel |u|
subplot(2,2,2)
waterfall(fftshift(k),t,abs(fftshift(utsol,2)))
axis([-40 40 0 5 0 80]), xlabel k, ylabel t, zlabel |fft(u)|
```

文件 NLSE.m 代码如下：

```
function dut=NLSE(t,ut,dummy,k)
u=ifft(ut);
dut=-(i/2)*(k.^2).*ut+i*fft((abs(u).^2).*u);
```

程序输出结果如图 3-7 所示，左图为空域上的 $u(x, t)$ ，右图为频域上的 $\hat{u}(k, t)$ ，二者都在传播过程中做周期性变化，最短周期约为 $\pi/2$ ，这与逆散射法的解析结果相符。

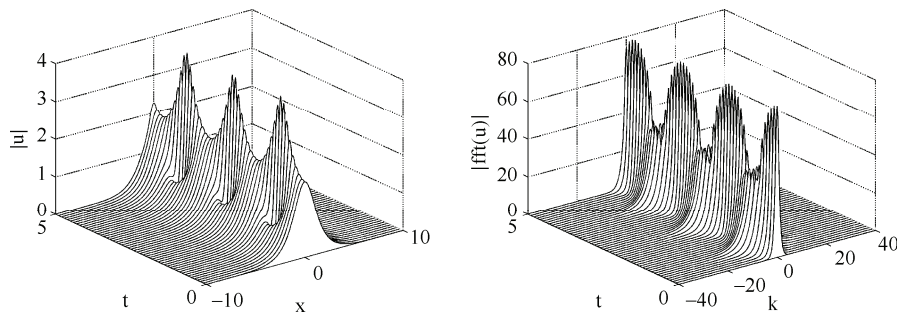


图 3-7 非线性薛定谔方程中的二阶孤子

3.3 傅里叶谱方法求解复杂偏微分方程（组）

3.3.1 一维 KdV 方程

1834 年，英国科学家、造船工程师 Scott Russel 观察到一只运行的木船船头挤出一堆水来，当船突然停下时，这堆水竟保持着它的形状，以大约 13km/h 的速度往前传播。1895 年由荷兰数学家 Korteweg 和 de Vries 共同导出了在浅水沟表面上按照一个方向传播的波的运动方程，也就是 KdV 方程 (Korteweg-de Vries equation)，它的形式可以写成：

$$\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x} - 12u \frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3} \quad (3-32)$$

u 是时间 t 和空间 x 的函数。式 (3-32) 的孤子解为 $a^2/4 \cdot \text{sech}^2\{1/2[ax-(a+a^3)t+\delta]\}$ ，其中， a 和 δ 为任意常数。对上式做傅里叶变换，得到：

$$\frac{\partial \hat{u}}{\partial t} = -ik\hat{u} - 12F\{F^{-1}[\hat{u}] \cdot F^{-1}[ik\hat{u}]\} - (ik)^3 \hat{u} \quad (3-33)$$

使用 $t=0$ 时的孤子解作为初始条件，令 $a=2$ 、 $\delta=0$ ，则 $u(x, 0)=\text{sech}^2(x)$ ，用傅里叶谱方法求解 KdV 方程的代码如下：

程序 3-6

主程序代码如下：

```
clear all; close all;
L=20; N=128;
x=L/N*[-N/2:N/2-1];
k=2*pi/L*[0:N/2-1 -N/2:-1].';
%初始条件
u=sech(x).^2; ut=fft(u);
%求解
t=0:0.05:1;
[t,utsol]=ode45('KdV',t,ut,[],k);
usol=ifft(utsol,[],2);
%画图
subplot(2,2,1)
waterfall(x,t,usol), axis([-10 10 0 1 0 1])
view(-7,35), xlabel x, ylabel t, zlabel |u|
```

```

subplot(2,2,2)
waterfall(fftshift(k),t,abs(fftshift(utsol,2)))
view(-7,30), axis([-20 20 0 1 0 15])
xlabel k, ylabel t, zlabel |fft(u)|
    
```

KdV.m 文件代码如下：

```

function dut=KdV(t,ut,dummy,k)
u=ifft(ut);
dut=-(i*k).*ut-12*fft(u.*ifft(i*k.*ut))-(i*k).^3.*ut;
    
```

程序输出结果如图 3-8 所示，左图为空域上的 $u(x, t)$ ，右图为频域上的 $\hat{u}(k, t)$ 。由孤子的解析解可知，孤子的形状保持不变，并以 5 个空间单位/1 个时间单位的速度向 x 正方向移动，相应地，频域上的振幅始终不变，这与图中的计算结果完全一致。

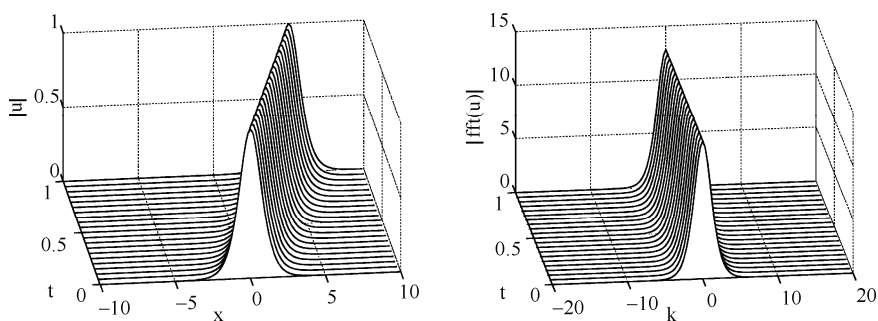


图 3-8 KdV 方程的孤子解

3.3.2 二维浅水方程组

浅水方程描述了具有自由表面、密度均匀、深度较浅的液体在重力作用下的流动过程，用于研究潮波和河流，具体形式如下：

$$\begin{cases} \frac{\partial \eta}{\partial t} = -\frac{\partial(\eta u)}{\partial x} - \frac{\partial(\eta v)}{\partial y} \\ \frac{\partial u}{\partial t} = -v \frac{\partial u}{\partial y} - u \frac{\partial u}{\partial x} - g \frac{\partial \eta}{\partial x} \\ \frac{\partial v}{\partial t} = -u \frac{\partial v}{\partial x} - v \frac{\partial v}{\partial y} - g \frac{\partial \eta}{\partial y} \end{cases} \quad (3-34)$$

其中， η 代表水深， t 为时间， x 和 y 是水平面上的坐标， u 、 v 是 x 和 y 方向上的流速， g 为重力加速度。对方程组 (3-34) 的等号两边做 x - y 空间上的二维傅里叶变换，得到偏

微分方程组:

$$\begin{cases} \frac{\partial \hat{\eta}}{\partial t} = -ik_x F \left\{ F^{-1}[\hat{\eta}] F^{-1}[\hat{u}] \right\} - ik_y F \left\{ F^{-1}[\hat{\eta}] F^{-1}[\hat{v}] \right\} \\ \frac{\partial \hat{u}}{\partial t} = -F \left\{ F^{-1}[\hat{v}] \cdot F^{-1}[ik_y \hat{u}] + F^{-1}[\hat{u}] \cdot F^{-1}[ik_x \hat{u}] \right\} - igk_x \hat{\eta} \\ \frac{\partial \hat{v}}{\partial t} = -F \left\{ F^{-1}[\hat{u}] \cdot F^{-1}[ik_x \hat{v}] + F^{-1}[\hat{v}] \cdot F^{-1}[ik_y \hat{v}] \right\} - igk_y \hat{\eta} \end{cases} \quad (3-35)$$

取 $g=1$, 初始条件为 $\eta(x, y, 0)=0.1 \cdot \exp(-x^2/10-y^2/10)+0.1$ 和 $u(x, y, 0)=v(x, y, 0)=0$, 用傅里叶谱方法计算上述方程的代码如下:

程序 3-7

主程序代码如下:

```
clear all; close all;
L=40; N=64;
x=L/N*[-N/2:N/2-1]; y=x;
kx=2*pi/L*[0:N/2-1 -N/2:-1]; ky=kx;
[X,Y]=meshgrid(x,y);
[kX,kY]=meshgrid(kx,ky);
%初始条件
e=0.1*exp(-X.^2/10-Y.^2/10)+0.1;
et=fft2(e); ut=zeros(N^2,1); vt=zeros(N^2,1);
euvt=[et(:); ut; vt;];
%求解
t=[0 5 10 25]; g=1;
[t,euvtsol]=ode45('shallow_water',t,euvt,[],kX,kY,N,g);
%画图
for n=1:4
    subplot(2,2,n)
    mesh(x,y,real(iff2(reshape(euvtsol(n,1:N^2),N,N))))
    axis([-20 20 -20 20 0.1 0.2]), title(['t=' num2str(t(n))])
    xlabel x, ylabel y, zlabel \eta, view(-80,45)
end
```

文件 shallow_water.m 代码如下:

```
function deuvt=shallow_water(t,euvt,dummy,kX,kY,N,g)
et=euvt(1:N^2); ut=euvt(N^2+[1:N^2]); vt=euvt(2*N^2+[1:N^2]);
```

```

et=reshape(et,N,N); ut=reshape(ut,N,N); vt=reshape(vt,N,N);
e=ifft2(et); u=ifft2(ut); v=ifft2(vt);
deuvt=[reshape(-i*kX.*fft2(e.*u)-i*kY.*fft2(e.*v),N^2,1);
        reshape(-fft2(v.*ifft2(i*kY.*ut)+u.*ifft2(i*kX.*ut))-g*i*kX.*et,N^2,1);
        reshape(-fft2(u.*ifft2(i*kX.*vt)+v.*ifft2(i*kY.*vt))-g*i*kY.*et,N^2,1)];

```

程序输出结果如图 3-9 所示，从 $t=0$ 时刻开始，一个三维高斯形水柱在重力的作用下坍塌，并激起了向四周传播的圆形水波。

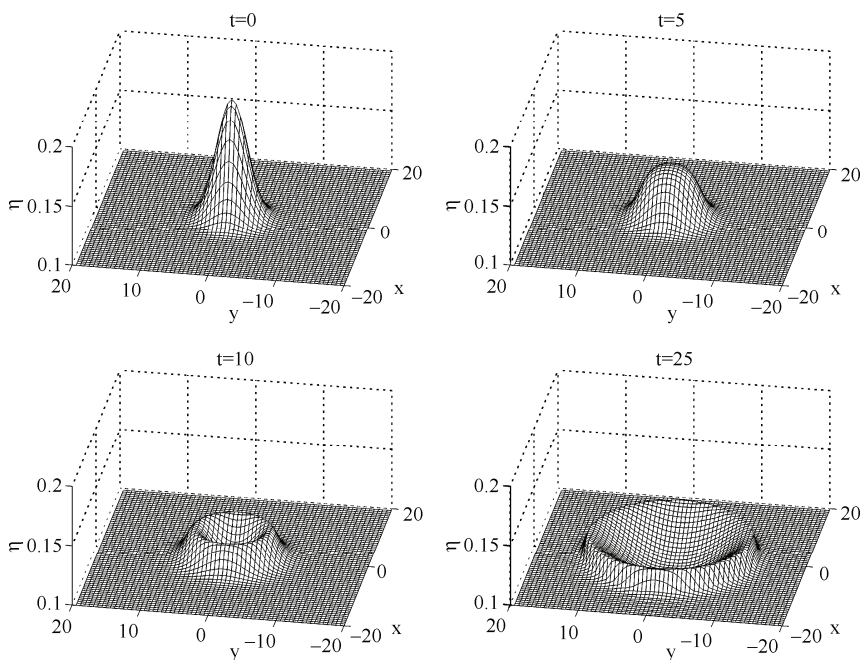


图 3-9 浅水方程的计算结果

3.3.3 二维粘性 Burgers 方程

Burgers 方程是流体力学中一个非常重要和基本的偏微分方程，它广泛地应用于空气动力学、湍流、交通流、热传导以及半导体模拟等领域。二维粘性 Burgers 方程的形式如下：

$$\frac{\partial u}{\partial t} = \nu \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u - u \left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y} \right) u \quad (3-36)$$

其中， u 代表速度， x 、 y 为空间坐标， t 为时间， ν 为粘性系数。对上式做二维傅里叶变换，得：

$$\frac{\partial \hat{u}}{\partial t} = -v(k_x^2 + k_y^2)\hat{u} - F\left\{F^{-1}[\hat{u}] \cdot F^{-1}[i(k_x + k_y)\hat{u}]\right\} \quad (3-37)$$

取 $v=0.01$ ，初始条件 $u(x, y, 0)=\text{sech}(4x^2+4y^2)$ ，傅里叶谱方法的代码如下：

程序 3-8

主程序代码如下：

```
clear all; close all;
L=4; N=64;
x=L/N*[-N/2:N/2-1]; y=x;
kx=2*pi/L*[0:N/2-1 -N/2:-1]; ky=kx;
[X,Y]=meshgrid(x,y);
[kX,kY]=meshgrid(kx,ky);
K2=kX.^2+kY.^2;
%初始条件
u=sech(4*X.^2+4*Y.^2);
ut=fft2(u);
%求解
v=0.01; t=0:0.4:1.2;
[t,utsol]=ode45('burgers',t,ut(:),[],N,kX,kY,K2,v);
%画图
for n=1:4
    subplot(2,2,n)
    mesh(x,y,real(iff2(reshape(utsol(n,:),N,N))))
    axis([-2 2 -2 2 0 1]), xlabel x, ylabel y, zlabel u
    view(46,20), title(['t=' num2str(t(n))])
end
```

burgers.m 文件代码如下：

```
function dut=burgers(t,ut,dummy,N,kX,kY,K2,v)
ut=reshape(ut,N,N); u=iff2(ut);
dut=reshape(-v*K2.*ut-fft2(u.*iff2(i*(kX+kY).*ut)),N^2,1);
```

程序执行结果如图 3-10 所示，初始波形逐渐演变成了激波，这是符合实际情况的。

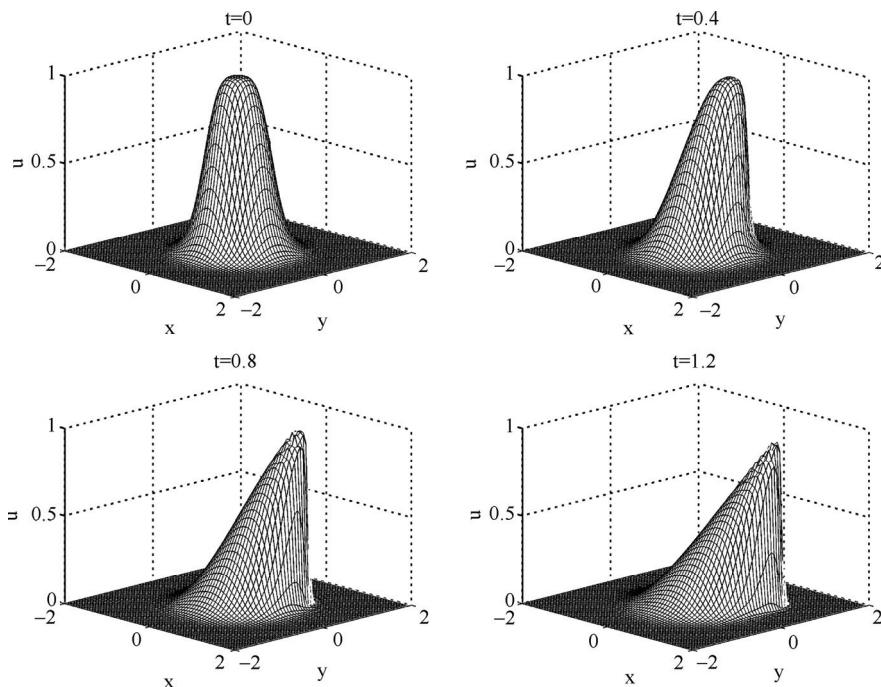


图 3-10 二维粘性 Burgers 方程的计算结果

3.3.4 二维 Schnakenberg 模型

斑图（pattern）是一类普遍存在于自然界、在时间或空间上具有某种规律的非均匀宏观结构。反应-扩散系统（reaction-diffusion system）是斑图理论中研究得最为广泛的系统，它起源于化学反应系统，但又不局限于此，还广泛应用于生物学、物理学、医学、金融学等。Schnakenberg 模型是反应-扩散系统中的一个有趣的模型，数学形式如下：

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u + \gamma (a - u + u^2 v) \\ \frac{\partial v}{\partial t} = d \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) v + \gamma (b - u^2 v) \end{cases} \quad (3-38)$$

其中， u 和 v 可看做两种化学反应物质的浓度， x 、 y 为空间坐标， t 为时间， a 、 b 、 d 、 γ 为常数。对式（3-38）做二维傅里叶变换，可将其转化为偏微分方程组：

$$\begin{cases} \frac{\partial \hat{u}}{\partial t} = -(k_x^2 + k_y^2) \hat{u} + \gamma \cdot F \left\{ a - F^{-1}[\hat{u}] + F^{-1}[\hat{u}]^2 F^{-1}[\hat{v}] \right\} \\ \frac{\partial \hat{v}}{\partial t} = -d(k_x^2 + k_y^2) \hat{v} + \gamma \cdot F \left\{ b - F^{-1}[\hat{u}]^2 F^{-1}[\hat{v}] \right\} \end{cases} \quad (3-39)$$

参数取值为： $a=0.1$ ， $b=0.8$ ， $d=26$ ， $\gamma=100$ 。为了得到靶型波，将初始条件设置为： u

在 $x-y$ 平面原点处为 1，在其他位置为 0， v 在整个 $x-y$ 平面上均为 1。利用傅里叶谱方法求解该模型的代码如下。

程序 3-9

主程序代码如下：

```
clear all; close all;
L=16; N=64;
kx=2*pi/L*[0:N/2-1 -N/2:-1]; ky=kx;
[kX,kY]=meshgrid(kx,ky);
K2=kX.^2+kY.^2;
%初始条件
u=zeros(N); u(N/2,N/2)=1; v=ones(N);
ut=fft2(u); vt=fft2(v);
uvt=[ut(:); vt(:)];
%求解
a=0.1; b=0.8; d=26; gamma=100; t=[0:0.1:0.3];
[t,uvtSol]=ode45('schnakenberg',t,uvt,[],K2,N,gamma,a,b,d);
%画图
for n=1:4
    subplot(2,2,n)
    gca=pcolor(iff2(reshape(uvtSol(n,1:N^2),N,N))); axis off
    set(gca,'LineStyle','none'), shading interp
    title(['t=' num2str(t(n))]), axis('square'), colormap('gray')
end
```

schnakenberg.m 文件代码如下：

```
function duvt=schnakenberg(t,uvt,dummy,K2,N,gamma,a,b,d)
ut=uvt(1:N^2); vt=uvt(N^2+1:end);
ut=reshape(ut,N,N); vt=reshape(vt,N,N);
u=iff2(ut); v=iff2(vt);
duvt=[reshape(-K2.*ut+gamma*fft2(a-u+(u.^2).*v),N^2,1);
      reshape(-d*K2.*vt+gamma*fft2(b-(u.^2).*v),N^2,1)];
```

程序输出的结果如图 3-11 所示，这与化学实验中观察到的靶型波一致。

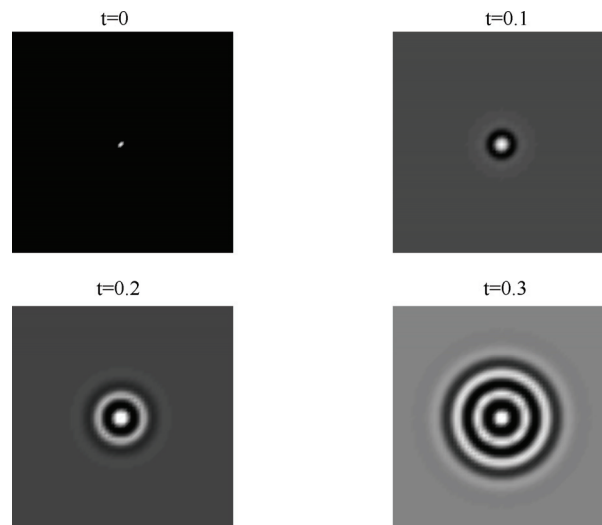


图 3-11 求解 Schnakenberg 模型得到的靶型波

第4章 谱求导矩阵

4.1 谱求导矩阵的导出和应用

在生产实践过程中,某些函数关系的具体表达式是未知的,只能通过实验测量得到该函数上的一些离散的数据点。所以,人们希望通过这些离散的数据点构造一个已知函数来近似地描述并替代未知函数。此外,尽管有些函数的表达式是已知的,但由于太过复杂繁琐,不便对其进行理论计算和数值分析,所以也有必要构造一个简单函数来替代它。

上述问题用数学语言表达为:已知在区间 $[a, b]$ 上的 N 个位置 $x_1, x_2, \dots, x_N$ 的函数值 $u_1, u_2, \dots, u_N$,求一个函数 $p(x)$ 通过所有已知点 $(x_1, u_1), (x_2, u_2), \dots, (x_N, u_N)$,即:

$$u_j = p(x_j), \quad j=1, 2, \dots, N \quad (4-1)$$

其中,用 $p(x)$ 近似未知函数的方法称为插值法(interpolation), $p(x)$ 称为插值函数。下面介绍用谱方法确定插值函数 $p(x)$ 的过程,并将其用于计算离散数据的各阶导数、偏微分方程(组)的数值求解。

4.1.1 谱方法插值

考虑在区间 $[0, 2\pi]$ 上、具有周期性边界条件的插值问题。在此区间上间距为 h 的 N 个位置 $x_1, x_2, \dots, x_N$ 对应的函数值为 $u_1, u_2, \dots, u_N$ 。其中, $x_j = jh$, $h = 2\pi/N$,即:

$$\frac{\pi}{h} = \frac{N}{2} \quad (4-2)$$

在推导前先做3点说明:

(1) 选择区间 $[0, 2\pi]$ 是为了方便讨论,在该区间上得出的结论同样适用于对其平移得到的其他区间(如 $[-\pi, \pi]$),并可以通过乘以缩放因子方便地将结论转化到其他长度非 2π 的任意区间上(如 $[-L, L]$)。

(2) 所谓周期性边界条件,是指 N 个函数值 $u_1, u_2, \dots, u_N$ 可等效地看做是无穷多个函数值 $\dots, u_{-1}, u_0, u_1, \dots, u_{N-1}, u_N, u_{N+1}, \dots$ 的一部分,并存在 $u_{j+mN} = u_j$ 的关系(m 为任意整数)。这里把讨论的函数当做周期为 2π 的周期函数处理。

(3) N 的奇偶会导致接下来的推导细节有所差异,但过程是相似的,所以本书只分析 N 为偶数的情况。

若对序列 $u_1, u_2, \dots, u_N$ 做离散傅里叶变换,那么空域(时域)上的间隔 h 决定了频域上的区间为 $[-\pi/h, \pi/h]$ (见表3-1)。由式(4-2),该区间也可写为 $[-N/2, N/2]$ 。本章中的离散傅里叶变换对定义为:

$$\hat{u}_k = h \sum_{j=1}^N e^{-ikx_j} u_j, \quad k = -\frac{N}{2} + 1, \dots, \frac{N}{2} \quad (4-3)$$

$$u_j = \frac{1}{2\pi} \sum_{k=-N/2+1}^{N/2} e^{ikx_j} \hat{u}_k, \quad j = 1, \dots, N \quad (4-4)$$

这与第 3 章给出的 Matlab 中离散傅里叶变换对的定义略有不同，但实质是一样的。注意到式 (4-4) 中的频率分量并不是完全对称的， k 中有 $0, \pm 1, \pm 2, \dots, \pm(N/2-1), N/2$ ，却没有 $-N/2$ ，这在求解插值函数时会引起一些小小的问题。所以，令 $\hat{u}_{-N/2} = \hat{u}_{N/2}$ ，并重新定义离散傅里叶逆变换为：

$$u_j = \frac{1}{2\pi} \sum_{k=-N/2}^{N/2} e^{ikx_j} \hat{u}_k, \quad j = 1, \dots, N \quad (4-5)$$

其中，“ Σ' ”代表求和时在 $k=\pm N/2$ 的项上乘以 $1/2$ 。需要强调的是，式 (4-3) 和式 (4-4) 仍然是离散傅里叶变换对的定义，式 (4-5) 仅是用于确定插值函数 $p(x)$ 的。求 $p(x)$ 时需要把式 (4-5) 中的 $x_j=jh$ 推广到 $[0, 2\pi]$ 上的任意实数 x ，即：

$$p(x) = \frac{1}{2\pi} \sum_{k=-N/2}^{N/2} e^{ikx} \hat{u}_k, \quad x \in [0, 2\pi] \quad (4-6)$$

确定插值函数的步骤是这样的：先用式 (4-3) 将序列 $u_1, u_2, \dots, u_N$ 变换为 $\hat{u}_{-N/2+1}, \hat{u}_{-N/2+2}, \dots, \hat{u}_{N/2}$ ，令 $\hat{u}_{-N/2} = \hat{u}_{N/2}$ ，再根据序列 $\hat{u}_{-N/2}, \hat{u}_{-N/2+1}, \dots, \hat{u}_{N/2}$ 通过式 (4-6) 得到 $p(x)$ 。这样得到的插值函数 $p(x)$ 可用来求序列 $u_1, u_2, \dots, u_N$ 在 x_j 处的各阶导数，即：

$$\left. \frac{\partial^n p(x)}{\partial x^n} \right|_{x=x_j} \quad (4-7)$$

上面介绍的是通过谱方法计算插值函数 $p(x)$ 来估算序列 $u_1, u_2, \dots, u_N$ 在 x_j 处的导数的基本原理。接下来，为了把上述过程转化为方便的矩阵运算，采用如下思路：首先求出周期 δ 函数的插值函数 $S_N(x)$ ，然后将任意序列 $u_1, u_2, \dots, u_N$ 写为周期 δ 函数的线性组合，进而可把它的插值函数 $p(x)$ 写为 $S_N(x)$ 的线性组合，最后找到 $p(x)$ 和 $S_N(x)$ 在 $x_1, x_2, \dots, x_N$ 处导数的关系并写为矩阵形式，给出针对任意序列 $u_1, u_2, \dots, u_N$ 的谱求导矩阵。

4.1.2 谱求导矩阵

周期 δ 函数定义为：

$$\delta_j = \begin{cases} 1, & (j \% N = 0) \\ 0, & (j \% N \neq 0) \end{cases} \quad (4-8)$$

其中，百分号“ $\%$ ”代表求余运算，周期 δ 函数在 $j=mN$ (m 为任意整数) 时取值为 1，其他情况为 0，它所对应的横坐标为 $x_j=jh$ 。利用式 (4-3) 求周期 δ 函数的离散傅里叶变换，结果为一常数 h ：

$$\hat{\delta}_k = h \sum_{j=1}^N e^{-ikx_j} \delta_j = h, \quad k = -\frac{N}{2} + 1, \dots, \frac{N}{2} \quad (4-9)$$

再利用式 (4-6) 求周期 δ 函数的插值函数:

$$\begin{aligned}
 p(x) &= \frac{h}{2\pi} \sum_{k=-N/2}^{N/2} e^{ikx} \\
 &= \frac{h}{2\pi} \left(\frac{1}{2} \sum_{k=-N/2}^{N/2-1} e^{ikx} + \frac{1}{2} \sum_{k=-N/2+1}^{N/2} e^{ikx} \right) \\
 &= \frac{h}{2\pi} \left(\frac{1}{2} e^{-\frac{i}{2}x} \sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx} + \frac{1}{2} e^{\frac{i}{2}x} \sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx} \right) \\
 &= \frac{h}{2\pi} \cos(x/2) \sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx} \\
 &= \frac{h}{2\pi} \cos(x/2) \frac{e^{i(-N/2+1/2)x} - e^{i(N/2+1/2)x}}{1 - e^{ix}} \\
 &= \frac{h}{2\pi} \cos(x/2) \frac{e^{-i(N/2)x} - e^{i(N/2)x}}{e^{-ix/2} - e^{ix/2}} \\
 &= \frac{h}{2\pi} \cos(x/2) \frac{\sin(Nx/2)}{\sin(x/2)}
 \end{aligned} \tag{4-10}$$

由式 (4-2), 最终得到的周期 δ 函数的插值函数为周期 sinc 函数 S_N :

$$S_N(x) = \frac{\sin(\pi x/h)}{(2\pi/h)\tan(x/2)} \tag{4-11}$$

可以证明, $S_N(x)|_{x \rightarrow 0} = 1$ 。图 4-1 显示了周期 δ 函数以及它的插值函数——周期 sinc 函数 $S_N(x)$, 二者的周期均为 2π , 无论 N 取值为 4 还是 16, 后者都精确、巧妙地经过了前者的所有离散点。

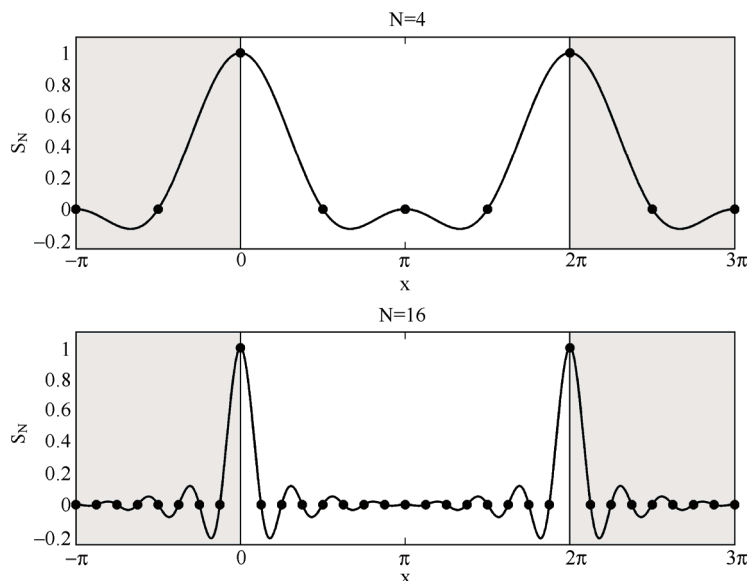


图 4-1 曲线为周期 sinc 函数, 黑点为周期 δ 函数, 上下两图分别对应 $N=4$ 和 $N=16$

若将区间 $[0, 2\pi]$ 上的序列 $u_1, u_2, \dots, u_N$ 写为周期 δ 函数的线性叠加，即：

$$u_j = \sum_{m=1}^N u_m \delta_{j-m} \quad (4-12)$$

那么，将其中的周期 δ 函数替换为周期 sinc 函数 S_N ，就得到序列 $u_1, u_2, \dots, u_N$ 的插值函数 $p(x)$ ：

$$p(x) = \sum_{m=1}^N u_m S_N(x - x_m) \quad (4-13)$$

注意到 $p(x)$ 就是 $S_N(x)$ 的线性组合，如果用 $p(x)$ 来估算序列 $u_1, u_2, \dots, u_N$ 在 $x_j = jh$ 处的 1 阶导数，则有：

$$p'(x_j) = \sum_{m=1}^N u_m S'_N(x_j - x_m) \quad (4-14)$$

其中， $x_j - x_m = (j - m)h$ 。将上式写为矩阵形式：

$$\begin{pmatrix} p'(x_1) \\ p'(x_2) \\ p'(x_3) \\ \vdots \\ p'(x_N) \end{pmatrix} = \begin{pmatrix} S'_N(0) & S'_N(-h) & S'_N(-2h) & \cdots & S'_N((1-N)h) \\ S'_N(h) & S'_N(0) & S'_N(-h) & & \\ S'_N(2h) & S'_N(h) & S'_N(0) & & \\ \vdots & & & \ddots & \\ S'_N((N-1)h) & & & & \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} \quad (4-15)$$

因此，只需在向量 $(u_1, u_2, \dots, u_N)^T$ 上乘以一个 N 阶方阵即可得到由它的谱方法插值函数 $p(x)$ 的导数组成的向量 $(p'(x_1), p'(x_2), \dots, p'(x_N))^T$ ，这个 N 阶方阵就是谱求导矩阵（spectral differentiation matrix）。下文用 $\mathbf{D}_N$ 来表示 1 阶 $N \times N$ 谱求导矩阵， $\mathbf{D}_N^{(n)}$ 表示 n 阶 $N \times N$ 谱求导矩阵。

周期 sinc 函数 $S_N(x)$ 在 $x_j = jh$ 处的 1 阶导数为：

$$S'_N(x_j) = \begin{cases} 0, & (j \% N = 0) \\ (-1)^j / 2 \cdot \cot(jh/2), & (j \% N \neq 0) \end{cases} \quad (4-16)$$

将其代入到式（4-15）中的 N 阶方阵，得到 1 阶谱求导矩阵：

$$\mathbf{D}_N = \begin{pmatrix} 0 & & & -\frac{\cot(1h/2)}{2} \\ -\frac{\cot(1h/2)}{2} & \ddots & & \frac{\cot(2h/2)}{2} \\ \frac{\cot(2h/2)}{2} & & \ddots & -\frac{\cot(3h/2)}{2} \\ -\frac{\cot(3h/2)}{2} & & & \vdots \\ \vdots & \ddots & & \frac{\cot(1h/2)}{2} \\ \frac{\cot(1h/2)}{2} & & & 0 \end{pmatrix} \quad (4-17)$$

类似地，还可以得到高阶谱求导矩阵。周期 sinc 函数 $S_N(x)$ 在 $x_j = jh$ 处的 2 阶导数为：

$$S_N''(x_j) = \begin{cases} -\frac{\pi^2}{3h^2} - \frac{1}{6}, & (j \% N = 0) \\ -\frac{(-1)^j \csc^2(jh/2)}{2}, & (j \% N \neq 0) \end{cases} \quad (4-18)$$

那么, 2 阶谱求导矩阵为:

$$\mathbf{D}_N^{(2)} = \begin{pmatrix} \ddots & & \vdots & & & \\ & \ddots & -\frac{1}{2} \csc^2\left(\frac{2h}{2}\right) & & & \\ & & \frac{1}{2} \csc^2\left(\frac{1h}{2}\right) & & & \\ & & -\frac{\pi^2}{3h^2} - \frac{1}{6} & & & \\ & & \frac{1}{2} \csc^2\left(\frac{1h}{2}\right) & & \ddots & \\ & & -\frac{1}{2} \csc^2\left(\frac{2h}{2}\right) & & \ddots & \\ & & \vdots & & \ddots & \ddots \end{pmatrix} \quad (4-19)$$

周期 sinc 函数 $S_N(x)$ 在 $x_j=jh$ 处的 3 阶导数为:

$$S_N'''(x_j) = \begin{cases} 0, & (j \% N = 0) \\ (-1)^j \cot\left(\frac{jh}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{jh}{2}\right) - \frac{\pi^2}{2h^2} \right], & (j \% N \neq 0) \end{cases} \quad (4-20)$$

同样得到 3 阶谱求导矩阵:

$$\mathbf{D}_N^{(3)} = \begin{pmatrix} 0 & & & -\cot\left(\frac{1h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{1h}{2}\right) - \frac{\pi^2}{2h^2} \right] \\ -\cot\left(\frac{1h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{1h}{2}\right) - \frac{\pi^2}{2h^2} \right] & \ddots & \ddots & \cot\left(\frac{2h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{2h}{2}\right) - \frac{\pi^2}{2h^2} \right] \\ \cot\left(\frac{2h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{2h}{2}\right) - \frac{\pi^2}{2h^2} \right] & & \ddots & -\cot\left(\frac{3h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{3h}{2}\right) - \frac{\pi^2}{2h^2} \right] \\ -\cot\left(\frac{3h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{3h}{2}\right) - \frac{\pi^2}{2h^2} \right] & & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \cot\left(\frac{1h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{1h}{2}\right) - \frac{\pi^2}{2h^2} \right] \\ \cot\left(\frac{1h}{2}\right) \left[\frac{3}{4} \csc^2\left(\frac{1h}{2}\right) - \frac{\pi^2}{2h^2} \right] & & & & 0 \end{pmatrix} \quad (4-21)$$

构造 n 阶谱求导矩阵 $\mathbf{D}_N^{(n)}$ 的一般方法为：

(1) 求周期 sinc 函数 $S_N(x)$ 在 $x_j=jh$ 处的 n 阶导数 $S_N^{(n)}(x_j)$ 。

(2) $\mathbf{D}_N^{(n)}$ 的第 1 列为 $(S_N^{(n)}(x_N), S_N^{(n)}(x_1), S_N^{(n)}(x_2), \dots, S_N^{(n)}(x_{N-1}))^T$, 第 2 列为 $(S_N^{(n)}(x_{N-1}), S_N^{(n)}(x_N), S_N^{(n)}(x_1), \dots, S_N^{(n)}(x_{N-2}))^T$, 第 3 列为 $(S_N^{(n)}(x_{N-2}), S_N^{(n)}(x_{N-1}), S_N^{(n)}(x_N), \dots, S_N^{(n)}(x_{N-3}))^T \dots \dots$ 依此类推。

此外, 周期函数 $S_N^{(n)}(x)$ 的奇偶性是由 n 的奇偶决定的, 这在构造 $\mathbf{D}_N^{(n)}$ 时可以产生一些便利:

$$\left. \frac{\partial^n S_N(x)}{\partial x^n} \right|_{x=x_j} = (-1)^n \left. \frac{\partial^n S_N(x)}{\partial x^n} \right|_{x=x_{N-j}} \quad (4-22)$$

当 n 为奇数时, 必有:

$$\left. \frac{\partial^n S_N(x)}{\partial x^n} \right|_{x=x_j} = 0, \quad j \% N = 0 \quad (4-23)$$

当 n 为偶数时, $S_N^{(n)}(x)$ 在 $x=x_j$ 处 ($j \% N = 0$) 的取值是无穷小/无穷小的形式, 需要用洛必达法则求它的极限, 即:

$$\left. \frac{\partial^n S_N(x)}{\partial x^n} \right|_{x=x_j} = \lim_{x \rightarrow 0} \frac{\partial^n S_N(x)}{\partial x^n}, \quad j \% N = 0 \quad (4-24)$$

人工计算 $S_N(x)$ 的 n 阶导数的工作量随着阶数 n 的增大而显著增加, 比较省时省力的方法是利用 Matlab 的符号运算功能, 调用 `diff` 函数和 `limit` 函数求导、求极限, 并结合 `toeplitz` 函数, 可实现程序自动生成 $\mathbf{D}_N^{(n)}$, 有兴趣的读者可以自行尝试。

由于谱求导矩阵 $\mathbf{D}_N^{(n)}$ 是在计算区间长度为 2π 的前提下构造的, 所以若实际的计算区间长度为 L , 则必须在 $\mathbf{D}_N^{(n)}$ 上乘以缩放系数才可保证结果正确, 即: $(2\pi/L)^n \mathbf{D}_N^{(n)}$ 。

下面给出使用谱求导矩阵 $\mathbf{D}_N$ 、 $\mathbf{D}_N^{(2)}$ 、 $\mathbf{D}_N^{(3)}$ 对 $u(x)=e^{\sin(\pi x)}$ 求 1、2、3 阶导数, 并与精确解 $u'(x)=\pi \cos(\pi x)e^{\sin(\pi x)}$ 、 $u''(x)=\pi^2 e^{\sin(\pi x)}[\cos^2(\pi x)-\sin(\pi x)]$ 、 $u'''(x)=\pi^3 e^{\sin(\pi x)}\cos(\pi x)[\cos^2(\pi x)-3\sin(\pi x)-1]$ 比较的实例。代码如下:

程序 4-1

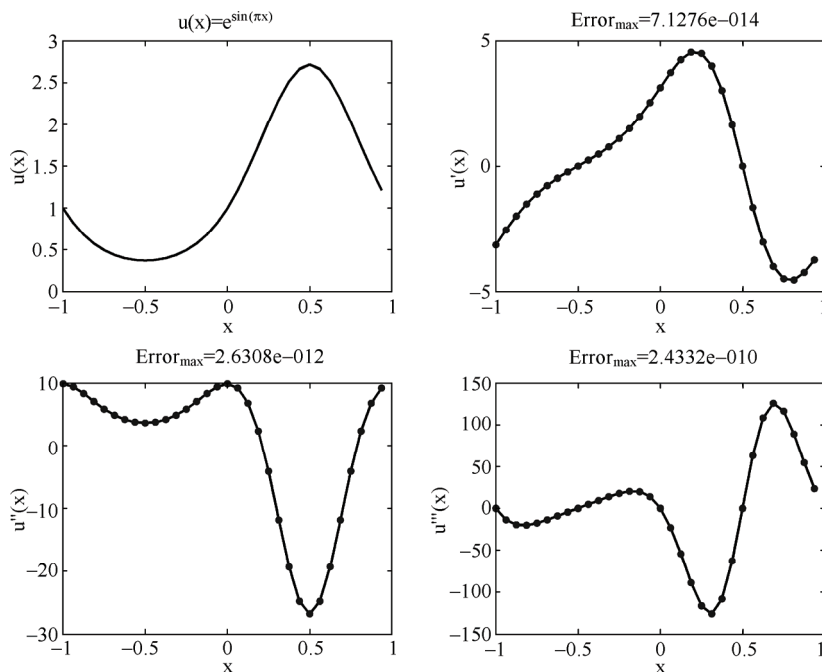
```
clear all; close all;
L=2; N=32;
x=L/N*[-N/2:N/2-1]';
%构造谱求导矩阵
h=2*pi/N; column=[0 0.5*(-1).^(1:N-1).*cot((1:N-1)*h/2)]';
D=(2*pi/L)*toeplitz(column,column([1 N:-1:2]));
```

```

column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
column=[0 (-1).^(1:N-1).*cot((1:N-1)*h/2).*( ...
    -pi^2/(2*h^2)+3/4*csc((1:N-1)*h/2).^2)]';
D3=(2*pi/L)^3*toeplitz(column,column([1 N:-1:2]));
%导数的精确解
u=exp(sin(pi*x));
du_exact(:,1)=pi*cos(pi*x).*u;
du_exact(:,2)=pi^2*(cos(pi*x).^2-sin(pi*x)).*u;
du_exact(:,3)=pi^3*cos(pi*x).*(cos(pi*x).^2-3*sin(pi*x)-1).*u;
%谱方法求导
du_Fourier(:,1)=D*u;
du_Fourier(:,2)=D2*u;
du_Fourier(:,3)=D3*u;
error=max(abs(du_exact-du_Fourier));
%画图
labels={'u'(x)','u''(x)','u'''(x)'};
for n=1:4
    subplot(2,2,n)
    if n==1
        plot(x,u,'k','LineWidth',1.5)
        xlabel x, ylabel u(x), title('u(x)=e^{sin(\pix)}')
    else
        plot(x,du_exact(:,n-1),'k',x,du_Fourier(:,n-1),'r' ...
            , 'MarkerSize',13,'LineWidth',1.5)
        title(['Error_{max}=' num2str(error(n-1))])
        xlabel x, ylabel(labels(n-1))
    end
end
end

```

如图 4-2 所示，曲线为精确的 $u(x)$ 、 $u'(x)$ 、 $u''(x)$ 和 $u'''(x)$ ，点为利用谱求导矩阵计算得到的 $u'(x_j)$ 、 $u''(x_j)$ 和 $u'''(x_j)$ 。在 N 的取值仅为 32 的情况下，用谱求导矩阵计算 $u(x)=e^{\sin(\pi x)}$ 的 1、2、3 阶导数与精确解的最大误差分别在 10^{-14} 、 10^{-12} 、 10^{-10} 数量级。


 图 4-2 用谱求导矩阵计算 $u(x)=e^{\sin(\pi x)}$ 的 1、2、3 阶导数（点）并与精确解（曲线）比较

4.1.3 用谱求导矩阵求解偏微分方程的步骤

与第 3 章一样，待求解的偏微分方程的普遍形式为：

$$\frac{\partial^n u}{\partial t^n} = Lu + N(u) \quad (4-25)$$

其中， $u(x, t)$ 为 x 、 t 的函数， L 代表线性算符， $N(u)$ 为非线性项。通过函数代换可将等号左边的 n 阶导数 $\partial^n / \partial t^n$ 降到 1 阶，所以下面分析式 (4-26) 的求解过程，这与式 (4-25) 降阶后得到的方程组的解法是一样的。

$$\frac{\partial u}{\partial t} = Lu + N(u) \quad (4-26)$$

用谱求导矩阵数值求解式 (4-26) 的步骤如下：

(1) 在 x 轴上的计算区间内，将 x 离散化为 N 个等间距的位置 $\mathbf{x}=(x_1, x_2, \dots, x_N)^T$ ，相应地，将函数 u 离散化为 N 维向量 $\mathbf{u}=(u_1, u_2, \dots, u_N)^T$ 。

(2) 在等号右边，将线性算符 L 和非线性项 $N(u)$ 里所有对函数 u 的求导运算替换为谱求导矩阵与向量 $\mathbf{u}$ 的乘运算，即： $\partial^n u / \partial x^n \rightarrow \mathbf{D}_N^{(n)} \mathbf{u}$ ，并将 L 和 $N(u)$ 都写成矩阵形式，得到形如式 (4-27) 的微分方程组。注意，若计算区间长度不是 2π ，则必须在 $\mathbf{D}_N^{(n)}$ 上乘以缩放系数。

(3) 用时间步进法（欧拉法、龙格-库塔法等）数值计算式 (4-27) 等号左边的 $\partial / \partial t$ ，在周期性边界条件下，得到不同 t 处的向量 $\mathbf{u}$ 。

$$\frac{\partial}{\partial t} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} & L_{13} & \cdots & L_{1N} \\ L_{21} & L_{22} & L_{23} & \cdots & L_{2N} \\ L_{31} & L_{32} & L_{33} & \cdots & L_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & L_{N3} & \cdots & L_{NN} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} + N \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} \quad (4-27)$$

针对步骤2, 这里以 $L = a \cdot \partial^2 / \partial x^2 + b \cdot \partial / \partial x + c$, $N(u) = u^3 + u^3 \cdot \partial^2 u / \partial x^2 + f(x) \cdot \partial u / \partial x$ 为例进行说明, a 、 b 和 c 分别为常数。对线性算符 L , 除了将 $\partial^n / \partial x^n$ 替换为 $\mathbf{D}_N^{(n)}$ 之外, 还必须在线性算符的常数 c 上乘以 N 阶单位矩阵 $\mathbf{I}_N$, 得到 N 阶方阵 $\mathbf{L}$, 即: $\mathbf{L} = a\mathbf{D}_N^{(2)} + b\mathbf{D}_N + c\mathbf{I}_N$ 。注意: Matlab 对矩阵与常数之间的加法是有特殊定义的, 如果忘记在 c 上乘以 $\mathbf{I}_N$ 会导致错误, 因为线性算符矩阵 $\mathbf{L}$ 与向量 $\mathbf{u}$ 相乘 $\mathbf{Lu} = (a\mathbf{D}_N^{(2)} + b\mathbf{D}_N + c\mathbf{I}_N)\mathbf{u} = a\mathbf{D}_N^{(2)}\mathbf{u} + b\mathbf{D}_N\mathbf{u} + c\mathbf{u} \neq (a\mathbf{D}_N^{(2)} + b\mathbf{D}_N + c)\mathbf{u}$ 。对非线性项 $N(u)$, 替换 $\partial^n / \partial x^n$ 为 $\mathbf{D}_N^{(n)}$ 时, $\mathbf{D}_N^{(n)}$ 与向量 $\mathbf{u}$ 之间的乘法是矩阵乘法, 在 Matlab 中用 “*” 表示。而 u^3 应理解为向量 $\mathbf{u}$ 中的每个元素的立方, 这在 Matlab 中用 “.^3” 表示。类似地, $f(x) \cdot (\mathbf{D}_N\mathbf{u})$ 也应处理为 $f(x)$ 和向量 $\mathbf{D}_N\mathbf{u}$ 中的每个对应元素的相乘, 在 Matlab 中用 “.*” 表示。所以此时 $\mathbf{Lu} + N(u)$ 在 Matlab 中应写为:

$$(a*\mathbf{D2} + b*\mathbf{D1} + c*\mathbf{I}) * \mathbf{u} + \mathbf{u}.^3 + \mathbf{u}.^3 .* (\mathbf{D2} * \mathbf{u}) + \mathbf{f} .* (\mathbf{D1} * \mathbf{u})$$

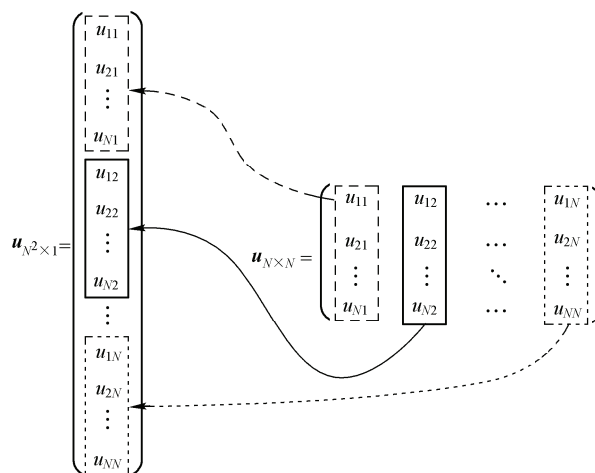
换句话说, 只有谱求导矩阵 $\mathbf{D}_N^{(n)}$ 与向量 $\mathbf{u}$ 之间的乘法是矩阵运算, 其余的运算均是在矩阵或向量的元素之间进行的。但也有一个例外, 由于线性算符 L 中的常数 c 在矩阵化时乘了单位矩阵 $\mathbf{I}_N$, 所以实际上 $c\mathbf{I}_N$ 与向量 $\mathbf{u}$ 之间也是矩阵乘法, 这样才能将其与线性算符 L 中的其他项相加: $\mathbf{L} = a\mathbf{D}_N^{(2)} + b\mathbf{D}_N + c\mathbf{I}_N$ 。

若方程式 (4-26) 再增加一个维度, 等号右边包含了 $u(x, y, t)$ 的 $\partial^n / \partial x^n$ 和 $\partial^n / \partial y^n$, 那么求解步骤中的一些细节就需要推广到二维空间上。在 x 轴、 y 轴上的计算区间内分别取等间距的 N 个位置 $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$ 以及 $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$, 于是在 x - y 平面上的计算区域内就得到了 N^2 个位置的坐标 $(x_1, y_1), (x_1, y_2), \dots, (x_1, y_N), (x_2, y_1), \dots, (x_2, y_N), \dots, (x_N, y_N)$ 。相应地, 函数 u 可以被离散化为 N 阶方阵 (第一种形式) 或 N^2 维列向量 (第二种形式), 如式 (4-28)、式 (4-29) 所示:

$$\mathbf{u}_{N \times N} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1N} \\ u_{21} & u_{22} & \cdots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NN} \end{pmatrix} \quad (4-28)$$

$$\mathbf{u}_{N^2 \times 1} = (u_{11}, u_{21}, \dots, u_{N1}, u_{12}, u_{22}, \dots, u_{N2}, \dots, u_{NN})^T \quad (4-29)$$

式 (4-28) 中, 第 i 行 j 列的元素 u_{ij} 对应的坐标是 (x_j, y_i) 。列向量 (4-29) 的第 1 到第 N 个元素对应于式 (4-28) 的第 1 列, 第 $N+1$ 到第 $2N$ 个元素对应于式 (4-28) 的第 2 列……依此类推, 如图 4-3 所示。另外, 可以通过 Matlab 中的 reshape 函数在二者间进行转换。


 图 4-3 函数 u 的两种离散形式及它们的对应关系

针对函数 u 的两种离散形式， $\partial^n u / \partial x^n$ 和 $\partial^n u / \partial y^n$ 的计算方法也大为不同。对于第一种形式，有：

$$\frac{\partial^n u}{\partial y^n} \rightarrow \mathbf{D}_N^{(n)} \mathbf{u}_{N \times N} \quad (4-30)$$

$$\frac{\partial^n u}{\partial x^n} \rightarrow \left(\mathbf{D}_N^{(n)} (\mathbf{u}_{N \times N})^T \right)^T = \mathbf{u}_{N \times N} \left(\mathbf{D}_N^{(n)} \right)^T \quad (4-31)$$

对于第二种形式，有：

$$\frac{\partial^n u}{\partial y^n} \rightarrow \left(\mathbf{I}_N \otimes \mathbf{D}_N^{(n)} \right) \mathbf{u}_{N^2 \times 1} \quad (4-32)$$

$$\frac{\partial^n u}{\partial x^n} \rightarrow \left(\mathbf{D}_N^{(n)} \otimes \mathbf{I}_N \right) \mathbf{u}_{N^2 \times 1} \quad (4-33)$$

其中 $\mathbf{I}_N$ 为 N 阶单位矩阵，“ $\otimes$ ”代表克罗内克积（Kronecker product），它是两个任意大小矩阵间的运算，可用 Matlab 中的 kron 函数实现。若 $k \times l$ 矩阵与 $m \times n$ 矩阵做克罗内克积运算，结果将是 $km \times ln$ 矩阵，并可以被分成 $k \times l$ 块，比如：

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix} \quad (4-34)$$

为帮助读者理解，下面给出 $n=2$ 、 $N=3$ 时式 (4-32) 和式 (4-33) 的一个例子，设 $\mathbf{D}_3^{(2)}$ 为：

$$\mathbf{D}_3^{(2)} = \begin{pmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{pmatrix} \quad (4-35)$$

那么, $\partial^2 u / \partial y^2$ 为:

$$\left(\mathbf{I}_3 \otimes \mathbf{D}_3^{(2)} \right) \mathbf{u}_{9 \times 1} = \left(\begin{array}{ccc|ccc|ccc} D_{11} & D_{12} & D_{13} & & & & & & \\ D_{21} & D_{22} & D_{23} & & & & & & \\ D_{31} & D_{32} & D_{33} & & & & & & \\ \hline & & & D_{11} & D_{12} & D_{13} & & & \\ & & & D_{21} & D_{22} & D_{23} & & & \\ & & & D_{31} & D_{32} & D_{33} & & & \\ \hline & & & & & & D_{11} & D_{12} & D_{13} \\ & & & & & & D_{21} & D_{22} & D_{23} \\ & & & & & & D_{31} & D_{32} & D_{33} \end{array} \right) \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{pmatrix} \quad (4-36)$$

其中, 空白元素均为 0, 下同。 $\partial^2 u / \partial x^2$ 为:

$$\left(\mathbf{D}_3^{(2)} \otimes \mathbf{I}_3 \right) \mathbf{u}_{9 \times 1} = \left(\begin{array}{ccc|ccc|ccc} D_{11} & & & D_{12} & & & D_{13} & & \\ & D_{11} & & & D_{12} & & & D_{13} & \\ & & D_{11} & & & D_{12} & & & D_{13} \\ \hline D_{21} & & & D_{22} & & & D_{23} & & \\ & D_{21} & & & D_{22} & & & D_{23} & \\ & & D_{21} & & & D_{22} & & & D_{23} \\ \hline D_{31} & & & D_{32} & & & D_{33} & & \\ & D_{31} & & & D_{32} & & & D_{33} & \\ & & D_{31} & & & D_{32} & & & D_{33} \end{array} \right) \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{pmatrix} \quad (4-37)$$

最后, 比较函数 u 在二维平面上的两种离散形式 (式 (4-28) 和式 (4-29)) 以及相应的数值求导过程。从使用的难易度和运算量来讲, 前者是远远优于后者的。因为前者最多只出现 N 阶方阵之间的乘法运算, 而后者出现了 N^2 阶方阵与 N^2 维向量的乘法运算。但是在做某些求导过程的逆运算时, 就不得不采用第二种离散形式。例如, 在 x - y 二维平面上函数 u 和 v 有如下关系:

$$\mathbf{v} = \Delta \mathbf{u} \quad (4-38)$$

Δ 为拉普拉斯算符, 根据式 (4-32) 和式 (4-33), 拉普拉斯算符可以表示为 N^2 阶方阵:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \rightarrow \mathbf{D}_N^{(2)} \otimes \mathbf{I}_N + \mathbf{I}_N \otimes \mathbf{D}_N^{(2)} \quad (4-39)$$

若已知 $\mathbf{v}$, 求 $\mathbf{u}$ 只需:

$$\mathbf{u}_{N^2 \times 1} = \left(\mathbf{D}_N^{(2)} \otimes \mathbf{I}_N + \mathbf{I}_N \otimes \mathbf{D}_N^{(2)} \right)^{-1} \mathbf{v}_{N^2 \times 1} \quad (4-40)$$

类似地, 还可以用此方法求诸如 $a \partial^n / \partial x^n + b \partial^m / \partial y^m$ 等导数的逆运算。然而, 对于第一种离散形式, 有 $\mathbf{v}_{N \times N} = \mathbf{D}_N^{(2)} \mathbf{u}_{N \times N} + \mathbf{u}_{N \times N} (\mathbf{D}_N^{(2)})^T$, 无法直观、方便地利用 $\mathbf{D}_N^{(2)}$ 和 $\mathbf{v}_{N \times N}$

求得 $\mathbf{u}_{N \times N}$ 。

此外，在二维的特征值问题中，也必须采用第二种离散形式才能求解。

4.2 利用谱求导矩阵求解基本偏微分方程（组）

4.2.1 一维线性谐振子的定态薛定谔方程

一维谐振子的特征值问题是量子力学中一个重要的基本问题，分子的振动、晶格的振动、原子核表面的振动和辐射场的振动等都可以分解为若干彼此独立的一维谐振动。若选取自然平衡位置为坐标 x 的原点和势能零点，则一维线性谐振子的势能可以表示为 $U(x)=Kx^2/2$ ，其中 K 是描述简谐作用力强度的参数。采用自然单位，一维线性谐振子的特征值问题有如下形式：

$$\left(-\frac{\partial^2}{\partial x^2} + x^2\right)u = \lambda u \quad (4-41)$$

其中， u 为波函数， λ 为特征值，且 $u \neq 0$ 。此问题存在解析解，特征值 $\lambda_n=2n+1$ ，其中 $n=0, 1, 2, \dots$ ，相应的特征函数为：

$$u_n = e^{-\frac{x^2}{2}} H_n(x) \quad (4-42)$$

其中， $H_n(x)$ 为埃尔米特多项式（Hermite polynomials）。由于特征函数随着 $|x|$ 的增加而迅速衰减，所以，若计算区间足够宽的话，就能保证特征函数在边界处的值总是近似为 0，这样即使用周期性边界条件来处理此特征值问题也无妨。或者，也可以认为这里的势场是周期势场。

将 x 和 u 离散化为： $\mathbf{x}=(x_1, x_2, \dots, x_N)^T$ ， $\mathbf{u}=(u_1, u_2, \dots, u_N)^T$ ，那么式（4-41）可写为矩阵形式：

$$\left[-\mathbf{D}_N^{(2)} + \text{diag}(x_1^2, x_2^2, \dots, x_N^2)\right] \mathbf{u} = \lambda \mathbf{u} \quad (4-43)$$

用 Matlab 中的 eig 函数可方便地求解这种矩阵形式的特征值问题，具体代码如下：

程序 4-2

```
clear all; close all;
L=20; N=64;
x=L/N*[-N/2:N/2-1];
%构造谱求导矩阵
h=2*pi/N;
```

```

column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%求特征值和特征函数
[V,D]=eig(-D2+diag(x.^2));
eigenvalues=diag(D);
for n=1:25
    subplot(5,5,n), plot(x,V(:,n),'k','LineWidth',1.5)
    title(num2str(eigenvalues(n)))
end

```

结果如图 4-4 所示, 前 25 个特征值、特征函数与解析解基本一致。随着特征值的增加, 特征函数的分布范围也越来越大。如果特征函数的分布范围超出了计算区间, 那么周期性边界条件会导致错误, 所以一定要保证计算区间足够大。注意, 每个特征值对应的特征函数并不是唯一的, 如果 u_n 是特征值为 λ_n 的特征函数, 那么 ku_n 也必是它的特征函数, k 为非零常数。

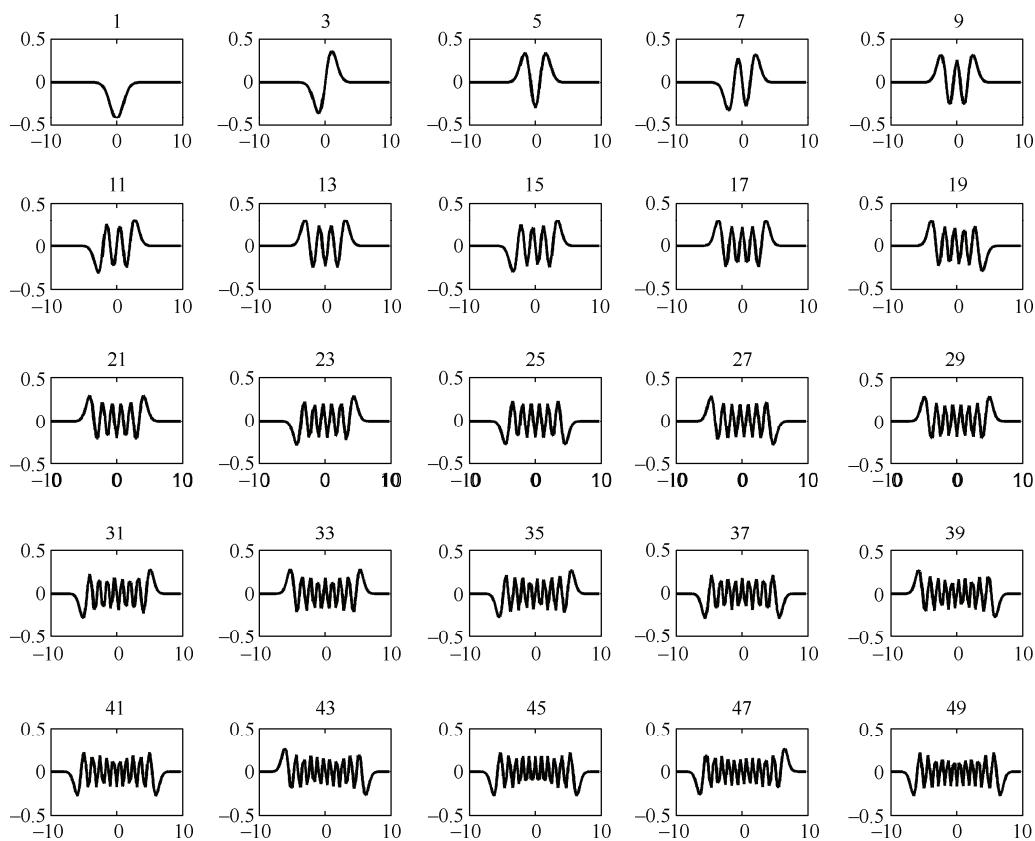


图 4-4 一维线性谐振子的前 25 个特征值和特征函数

4.2.2 二维线性谐振子的定态薛定谔方程

二维线性谐振子的特征值问题有如下形式：

$$\left(-\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2} + x^2 + y^2 \right) u = \lambda u \quad (4-44)$$

其中， x 、 y 为空间坐标， u 为波函数。特征值的解析解为：

$$\lambda_{m,n} = \lambda_m + \lambda_n \quad (4-45)$$

其中， $\lambda_m = 2m+1$ ， $\lambda_n = 2n+1$ ， $m=0, 1, 2, \dots$ ， $n=0, 1, 2, \dots$ 。与 4.2.1 小节类似，因为特征函数随着 $|x|$ 和 $|y|$ 的增大而快速衰减，所以在计算范围足够大的前提下可以采用周期性边界条件。

将 x 、 y 离散化为： $\mathbf{x}=(x_1, x_2, \dots, x_N)^T$ 以及 $\mathbf{y}=(y_1, y_2, \dots, y_N)^T$ ，再将 u 离散化为 N^2 维列向量：

$$\mathbf{u}_{N^2 \times 1} = (u_{11}, u_{21}, \dots, u_{N1}, u_{12}, u_{22}, \dots, u_{N2}, \dots, u_{NN})^T \quad (4-46)$$

那么， $(-\partial^2/\partial x^2 - \partial^2/\partial y^2 + x^2 + y^2)$ 的矩阵形式为：

$$-\mathbf{D}_N^{(2)} \otimes \mathbf{I}_N - \mathbf{I}_N \otimes \mathbf{D}_N^{(2)} + \text{diag}(x_1^2, \dots, x_N^2) \otimes \mathbf{I}_N + \mathbf{I}_N \otimes \text{diag}(y_1^2, \dots, y_N^2) \quad (4-47)$$

同样利用 eig 函数求解此矩阵的特征值和特征函数，代码如下：

程序 4-3

```
clear all; close all;
L=10; N=32;
x=L/N*[-N/2:N/2-1]; y=x;
%构造谱求导矩阵
h=2*pi/N;
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%求特征值和特征函数
I=eye(N); L=kron(D2,I)+kron(I,D2);
x2=kron(diag(x.^2),I); y2=kron(I,diag(y.^2));
[V,D]=eig(-L+x2+y2); eigenvalues=diag(D);
%画图
[py,px]=meshgrid(0.75:-0.25:0,0:0.25:0.75);
```

```

figure(1)
for n=1:16
    subplot('position',[px(n)+0.04 py(n)+0.04 0.17 0.17])
    mesh(x,y,reshape(V(:,n),N,N))
    axis([-4 4 -4 4 -0.1 0.1]), title(num2str(eigenvalues(n)))
end
figure(2)
for n=1:16
    subplot('position',[px(n)+0.04 py(n)+0.04 0.17 0.17])
    contour(x,y,reshape(V(:,n),N,N),[-0.03 0.03])
    axis([-4 4 -4 4]), axis square, title(num2str(eigenvalues(n)))
end

```

结果如图 4-5（三维图）、图 4-6（等高线图）所示：

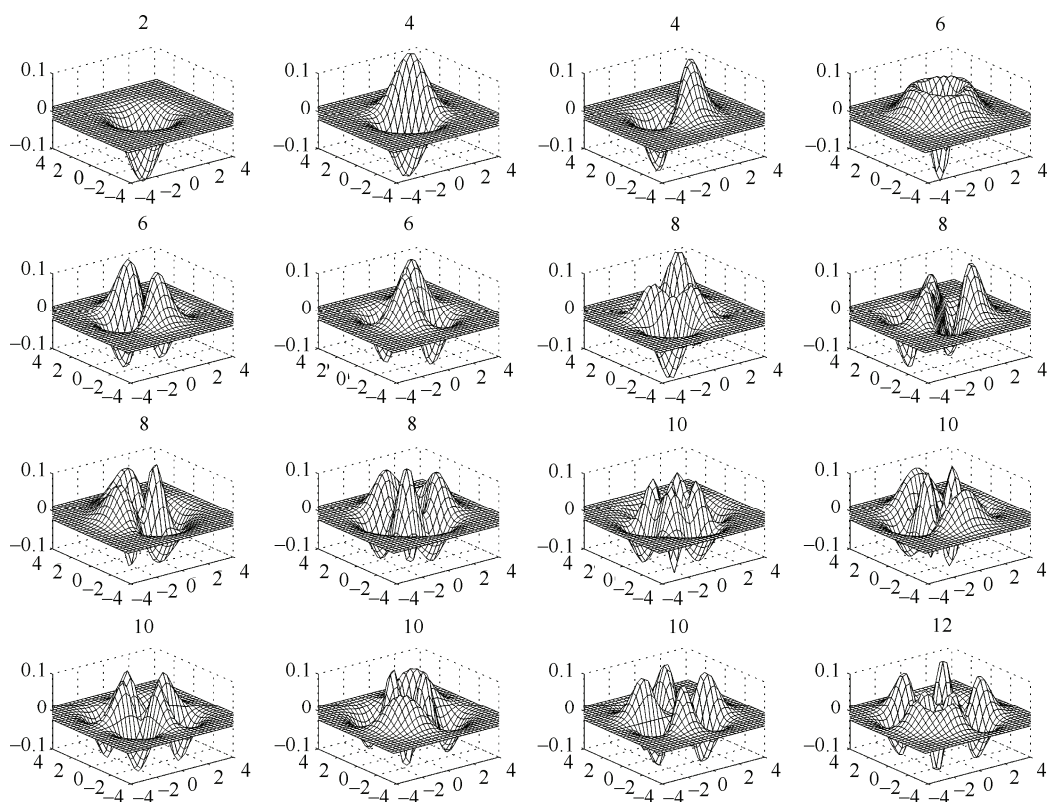


图 4-5 二维线性谐振子的前 16 个特征值和特征函数

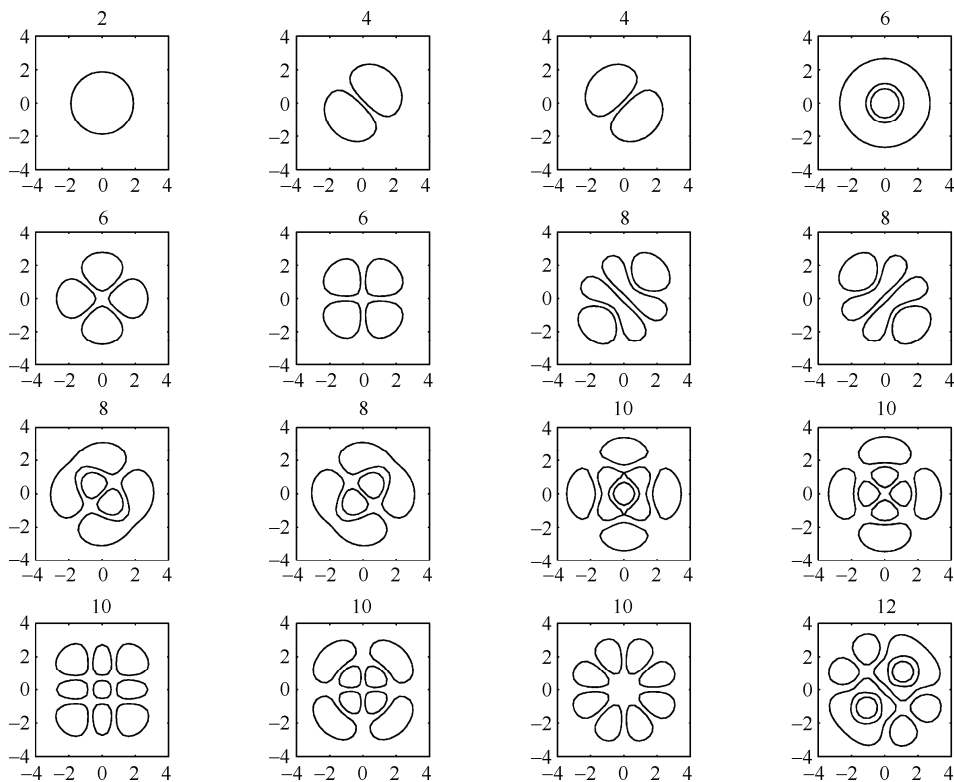


图 4-6 二维线性谐振子的前 16 个特征值和特征函数（等高线图）

4.2.3 一维波动方程

对于一根两端固定、细长柔软的弦线，若其平衡位置为 x 轴， $u(x, t)$ 表示弦上坐标为 x 的点在时刻 t 垂直于 x 方向的位移，则弦的自由振动方程为：

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \quad (4-48)$$

实际上，杆的纵振动方程、理想传输线的电报方程也是此方程。对于无界 ($-\infty < x < \infty$) 弦的自由振动，如果计算区间足够大，那么就可以采用周期性边界条件。这里取 $a=1$ ，初始条件为：

$$u|_{t=0} = 2\operatorname{sech}(x+10) - 2\operatorname{sech}(x-10), \quad \frac{\partial u}{\partial t}|_{t=0} = 0 \quad (4-49)$$

在数学物理方法中，给定初始条件的无界弦自由振动方程存在解析解，称为达朗贝尔公式。对于初始条件 (4-49)，弦各处的初始速度为 0，弦的初始波形包含了 $x=\pm 10$ 处的 2 个 sech 函数。根据达朗贝尔公式，这 2 个 sech 函数将分别分裂成两半，以速

度 a 沿着 x 轴的正方向和负方向传播, 即正行波和反行波, 这些正行波和反行波的叠加就给出了弦的总位移。下面给出数值解法的代码, 并与达朗贝尔公式的预测进行比较。

程序 4-4

主程序代码如下:

```
clear all; close all;
L=80; N=160;
x=L/N*[-N/2:N/2-1];
%构造谱求导矩阵
h=2*pi/N;
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%初始条件
u=2*sech(x+10)-2*sech(x-10);
v=zeros(1,N); uv=[u v];
%数值求解
a=1; t=0:0.5:20;
[t,uvsol]=ode45('wave1D',t,uv,[],N,D2,a);
%画图
p=[1 11 21 41];
for n=1:4
    subplot(5,2,n)
    plot(x,uvsol(p(n),1:N),'k','LineWidth',1.5), xlabel x, ylabel u
    title(['t=' num2str(t(p(n)))]), axis([-L/2 L/2 -2 2])
end
subplot(5,2,5:10)
waterfall(x,t,uvsol(:,1:N)), view(10,45)
xlabel x, ylabel t, zlabel u, axis([-L/2 L/2 0 t(end) -2 2])
```

wave1D.m 文件代码如下:

```
function duv=wave1D(t,uv,dummy,N,D2,a)
u=uv(1:N); v=uv(N+[1:N]);
duv=[v; a^2*D2*u];
```

程序输出结果如图 4-7 所示，上图是弦在 $t=0, 5, 10, 20$ 时的位移 u ，下图是位移 u 随位置 x 、时间 t 的变化情况。初始条件中的 sech 函数在 $t=0$ 之后分裂为正行波和反行波，正行波和反行波的振幅为分裂前 sech 函数振幅的一半，并分别以速度 $a=1$ 向两边传播。在 $t=10$ 时，其中的一对正行波和反行波叠加为 0，所以它们暂时消失了。总之，数值计算的结果与达朗贝尔公式的预测完全一致。

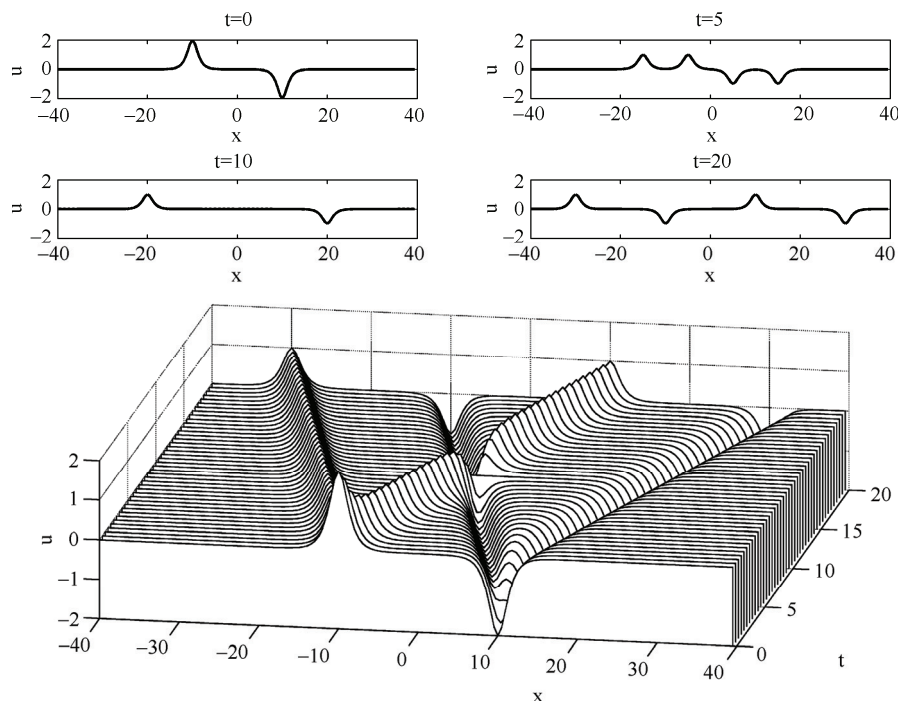


图 4-7 一维波动方程数值解

4.2.4 二维波动方程

将式 (4-48) 推广到二维情况得到：

$$\frac{\partial^2 u}{\partial t^2} = a^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u \quad (4-50)$$

它是薄膜的波动方程。其中， $u(x, y, t)$ 为 t 时刻位置 (x, y) 处薄膜的位移。

取 $a=1$ ，初始条件为：

$$u|_{t=0} = e^{-20[(x-0.4)^2 + (y+0.4)^2]} - e^{-20[(x+0.4)^2 + (y-0.4)^2]}, \quad \frac{\partial u}{\partial t}|_{t=0} = 0 \quad (4-51)$$

在计算区域内将 u 离散化为式 (4-28) 的形式，那么根据式 (4-30) 和式 (4-31)， $(\partial^2/\partial x^2 + \partial^2/\partial y^2)u$ 可表为：

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u \rightarrow \mathbf{u}_{N \times N} \left(\mathbf{D}_N^{(2)}\right)^T + \mathbf{D}_N^{(2)} \mathbf{u}_{N \times N} \quad (4-52)$$

式(4-50)数值求解过程与4.2.3小节类似,代码如下:

程序4-5

主程序代码如下:

```
clear all; close all;
L=4; N=60;
x=L/N*[-N/2:N/2-1]; y=x;
[X,Y]=meshgrid(x,y);
%构造谱求导矩阵
h=2*pi/N;
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%初始条件
u=exp(-20*((X-0.4).^2+(Y+0.4).^2))-exp(-20*((X+0.4).^2+(Y-0.4).^2));
v=zeros(N); uv=[u(:) v(:)];
%数值求解
a=1; t=[0 0.25 0.5 1];
[t,uvsol]=ode45('wave2D',t,uv,[],N,D2,a);
%画图
for n=1:4
    subplot(2,2,n)
    mesh(x,y,reshape(uvsol(n,1:N^2),N,N))
    title(['t=' num2str(t(n))]), xlabel x, ylabel y, zlabel u
    axis([-L/2 L/2 -L/2 L/2 -1 1])
end
```

wave2D.m 文件代码如下:

```
function duv=wave2D(t,uv,dummy,N,D2,a)
u=reshape(uv(1:N^2),N,N); v=uv(N^2+[1:N^2]);
duv=[v; reshape(a^2*D2*u+a^2*u*D2',N^2,1)];
```

数值计算结果如图4-8所示,薄膜上的波动逐渐向四周传播。

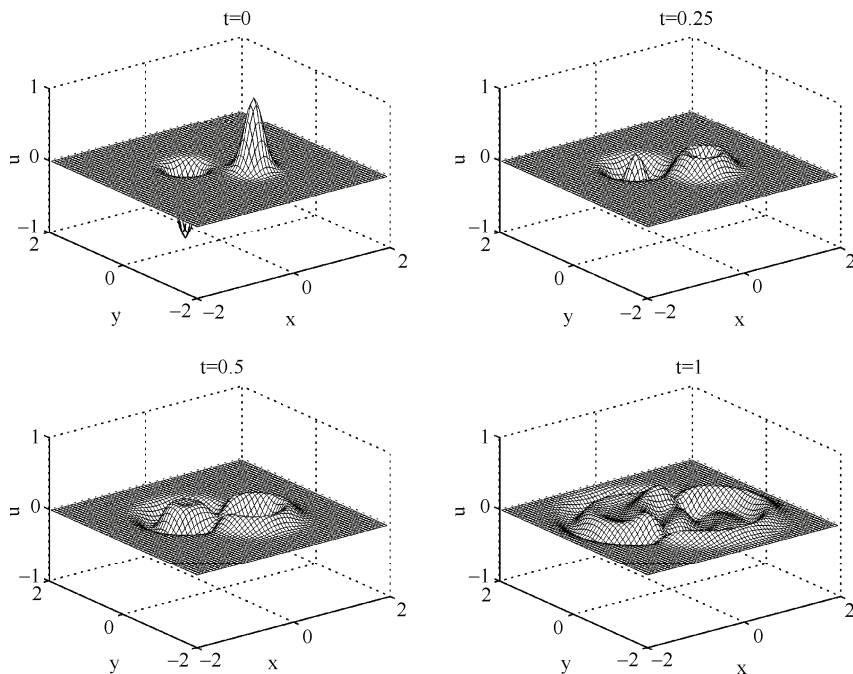


图 4-8 二维波动方程的数值计算结果

4.3 利用谱求导矩阵求解复杂偏微分方程（组）

4.3.1 Ginzburg-Landau 方程

Ginzburg-Landau 方程在超导理论、流体力学以及光学中有着重要的应用。用于输出超短脉冲的锁模[⊖]激光器就是由 Ginzburg-Landau 方程描述的（在光学中也称为主方程，即 master equation），这里讨论如下形式的 Ginzburg-Landau 方程：

$$\frac{\partial u}{\partial z} = -\frac{i}{2}(\beta_2 + ig_0\tau)\frac{\partial^2 u}{\partial t^2} + i\gamma|u|^2u + \frac{1}{2}\left(g_0 - \frac{\alpha_0}{1+|u|^2/P_0}\right)u \quad (4-53)$$

其中， u 、 t 和 z 分别代表光的复振幅、时间和光在激光器内传播的距离， i 为虚数单位。其他参数取值为 $\beta_2=-1$ ， $g_0=1$ ， $\tau=0.2$ ， $\gamma=1$ ， $\alpha_0=1.2$ ， $P_0=1$ 。与此前所讨论的方程不同，式（4-53）属于复数偏微分方程，但在利用 Matlab 求解过程中，它的处理与实数偏微分方程并无太大差异，代码如下：

程序 4-6

主程序代码如下：

```
clear all; close all;
```

[⊖] 锁模是光学里一种用于产生极短时间激光脉冲的技术。

```

L=20; N=100;
t=L/N*[-N/2:N/2-1];
%构造谱求导矩阵
h=2*pi/N;
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%初始条件
u=sech(t/2)*0.3+0.5*rand(1,N);
%数值求解
beta2=-1; gamma=1; g0=1; tau=0.2;
alpha0=1.2; P0=1; z=0:30;
[z,usol]=ode45('master',z,u,[],D2,beta2,gamma,g0,tau,alpha0,P0);
%画图
waterfall(t,z,abs(usol)), xlabel t, ylabel z, zlabel |u|

```

master.m 文件代码如下:

```

function du=master(z,u,dummy,D2,beta2,gamma,g0,tau,alpha0,P0)
du=-i/2*(beta2+i*g0*tau)*D2*u+i*gamma*abs(u).^2.*u+...
    1/2*(g0-alpha0./(1+abs(u).^2/P0)).*u;

```

程序输出结果如图 4-9 所示。尽管光的初始条件包含了很强的噪声，但由于多种物理效应的共同作用，光在锁模激光器内的传播过程当中形成了稳定的脉冲，这与锁模激光器的实际情况是一致的，称作自启动。

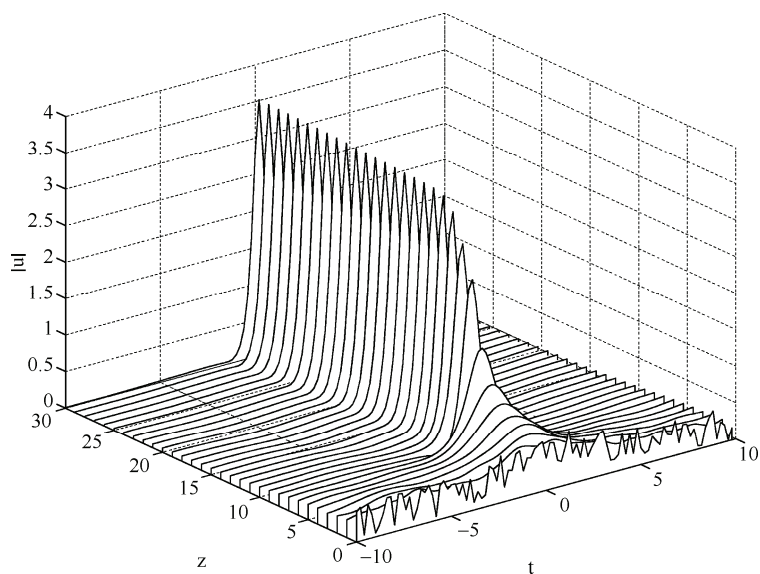


图 4-9 锁模激光器的自启动过程

4.3.2 耦合非线性薛定谔方程组

对于某些基于耦合效应和非线性效应的锁模激光器（如：用波导阵列做锁模器件的激光器），理论模型就需要用耦合非线性薛定谔方程组（coupled nonlinear Schrödinger equations）来描述：

$$\begin{cases} \frac{\partial u_1}{\partial z} = ik_{12}u_2 - \frac{i}{2}(\beta_2 + ig\tau)\frac{\partial^2 u_1}{\partial t^2} + i\gamma|u_1|^2 u_1 + \frac{1}{2}(g - \alpha_1)u_1 \\ \frac{\partial u_2}{\partial z} = i(k_{12}u_1 + k_{23}u_3) - \frac{\alpha_2}{2}u_2 \\ \frac{\partial u_3}{\partial z} = ik_{23}u_2 - \frac{\alpha_3}{2}u_3 \end{cases} \quad (4-54)$$

其中， u_1 、 u_2 和 u_3 为 3 个波导内的光复振幅， t 、 z 为时间和光在波导内传播的距离， i 为虚数单位。其他参数取值： $k_{12}=0.3$ ， $k_{23}=0.3$ ， $\beta_2=1$ ， $g=0.3$ ， $\tau=2$ ， $\gamma=1$ ， $\alpha_1=0.01$ ， $\alpha_2=0.1$ ， $\alpha_3=1$ 。由于式（4-54）中的 3 个方程是耦合的，所以需要同时对它们求解，代码如下：

程序 4-7

主程序代码如下：

```
clear all; close all;
L=100; N=100;
t=L/N*[-N/2:N/2-1];
%构造谱求导矩阵
h=2*pi/N;
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%初始条件
u1=0.3*sech(t/4)+0.6*rand(1,N);
u2=zeros(1,N); u3=zeros(1,N);
u=[u1 u2 u3];
%数值求解
k12=0.3; k23=k12; beta2=1; gamma=1; g=0.3;
tau=2; a1=0.01; a2=0.1; a3=1; z=0:4:200;
[z,usol]=ode45('CNLSE',z,u,[],N,D2,k12,k23,beta2,gamma,g,tau,a1,a2,a3);
%画图
for n=1:3
    subplot(2,2,n)
```

```

waterfall(t,z,abs(usol(:,(n-1)*N+[1:N])))
xlabel t, ylabel z, zlabel(['|u_' num2str(n) '|'])
end

```

CNLSE.m 文件代码如下:

```

function du=CNLSE(z,u,dummy,N,D2,k12,k23,beta2,gamma,g,tau,a1,a2,a3)
u1=u(1:N); u2=u(N+[1:N]); u3=u(2*N+[1:N]);
du=[i*k12*u2-i/2*(beta2+i*g*tau)*D2*u1+...
    i*gamma*abs(u1).^2.*u1+1/2*(g-a1).*u1;
    i*(k12*u1+k23*u3)-a2/2*u2;
    i*k23*u2-a3/2*u3];

```

类似于 4.3.1 小节的结果, 这种基于波导阵列的锁模激光器同样可以实现自启动锁模, 如图 4-10 所示, 3 个波导内的光均从噪声初始条件开始逐渐形成了稳定的脉冲。

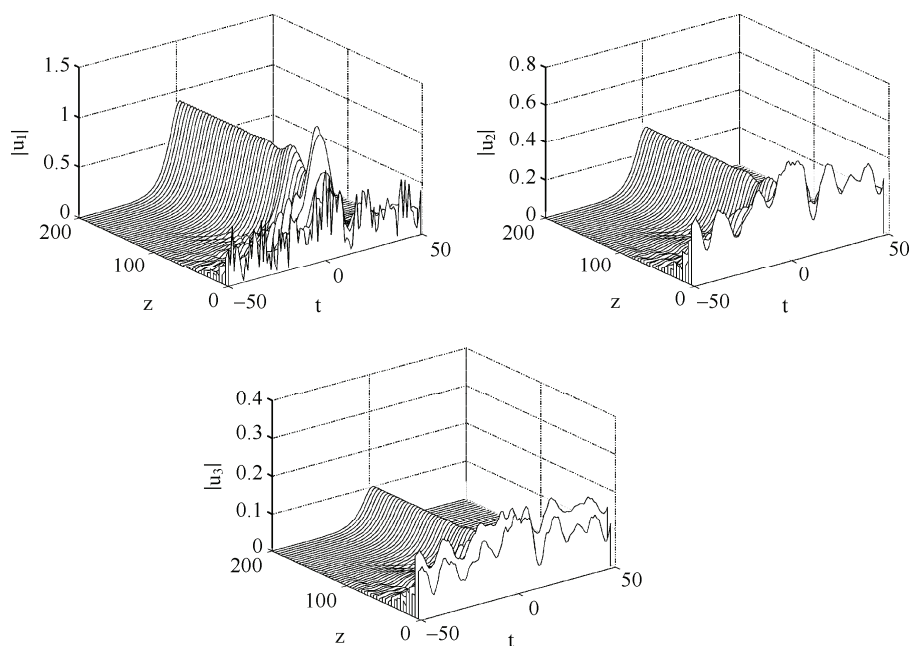


图 4-10 耦合非线性薛定谔方程数值计算结果

4.3.3 二维 Schnakenberg 模型

1952 年, 被称为计算机理论之父的英国科学家图灵提出了一个新颖的想法: 一些动物 (如: 老虎、金钱豹) 身上之所以会长着有规律的条纹斑点图案 (斑图), 是因为在细胞中存在一种化学物质, 由于反应-扩散系统的特性而自组织形成了斑图。这种想法逐渐被后来

的实验和理论所证实。本小节以反应-扩散系统中的 Schnakenberg 模型为例介绍计算斑图的方法，它的形式如下：

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u + \gamma (a - u + u^2 v) \\ \frac{\partial v}{\partial t} = d \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) v + \gamma (b - u^2 v) \end{cases} \quad (4-55)$$

其中， u 和 v 可看做两种化学反应物质的浓度， x 、 y 为空间坐标， t 为时间，参数取值如下： $a=0.1$ 、 $b=0.8$ 、 $d=26$ 、 $\gamma=100$ 。由于动物表皮为闭合曲面，因此使用周期性边界条件是合理的。

若把 u 离散化为方阵 $\mathbf{u}_{N \times N}$ ，则方程中的 $(\partial^2/\partial x^2 + \partial^2/\partial y^2)u$ 可表示为 $\mathbf{u}_{N \times N}(\mathbf{D}_N^{(2)})^T + \mathbf{D}_N^{(2)} \mathbf{u}_{N \times N}$ ， $(\partial^2/\partial x^2 + \partial^2/\partial y^2)v$ 同理。等号左边的 $\partial/\partial t$ 交给 ode23 函数计算即可，具体代码如下：

程序 4-8

主程序代码如下：

```
clear all; close all;
L=4; N=60;
%构造谱求导矩阵
h=2*pi/N;
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%初始条件
u=rand(N)*0.5+0.5; v=0.5*ones(N);
uv=[u(:); v(:)];
%数值求解
gamma=100; a=0.1; b=0.8; d=26;
t=0:0.1:0.3;
[t, uvsol]=ode23('schnakenberg', t, uv, [], D2, N, gamma, a, b, d);
%画图
for n=1:4
    subplot(2,2,n)
    gca=pcolor(reshape(uvsol(n,1:N^2),N,N)); axis off
    set(gca,'LineStyle','none'), shading interp
    title(['t=' num2str(t(n))]), axis square
end
```

schnakenberg.m 文件代码如下：

```

function duv=schnakenberg(t,uv,dummy,D2,N,gamma,a,b,d)
u=uv(1:N^2); v=uv(N^2+1:end);
u=reshape(u,N,N); v=reshape(v,N,N);
duv=[reshape((D2*u+u*D2')+gamma*(a-u+u.^2.*v),N^2,1);
      reshape(d*(D2*v+v*D2')+gamma*(b-u.^2.*v),N^2,1)];

```

程序输出的结果如图 4-11 所示, $t=0$ 时的初始条件设置为噪声, 随后图形逐渐开始自组织。在 $t=0.3$ 时, 斑图已经初具雏形, 十分类似动物皮毛上的图案。

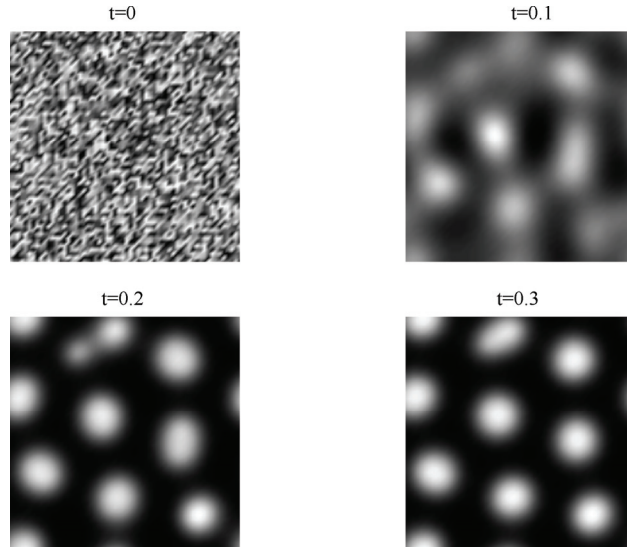


图 4-11 计算二维 Schnakenberg 模型得到的斑图

4.3.4 二维平流-扩散方程

二维平流-扩散方程 (advection-diffusion equation) 描述了大气的准二维运动, 它的形式如下:

$$\begin{cases} \frac{\partial \omega}{\partial t} = \nu \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \omega + \frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \psi = \omega \end{cases} \quad (4-56)$$

其中, ψ 、 ω 分别为流函数 (stream function) 和涡量 (vorticity), x 、 y 为空间坐标, t 为时间, ν 为一常数。式 (4-56) 中的第一式同时包含了平流部分—— $\frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y}$, 以及扩散部分—— $\nu(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})\omega$ 。将 ψ 和 ω 离散化为方阵 $\boldsymbol{\psi}_{N \times N}$ 和 $\boldsymbol{\omega}_{N \times N}$ 之后, 等号右边可写为 $\nu[\boldsymbol{\omega}_{N \times N}(\mathbf{D}_N^{(2)})^T + \mathbf{D}_N^{(2)}\boldsymbol{\omega}_{N \times N}] + \mathbf{D}_N \boldsymbol{\psi}_{N \times N} \boldsymbol{\omega}_{N \times N}(\mathbf{D}_N)^T - \boldsymbol{\psi}_{N \times N}(\mathbf{D}_N)^T \cdot \mathbf{D}_N \boldsymbol{\omega}_{N \times N}$, 等号左边的 $\partial/\partial t$ 使用 ode45 函数计算。

流函数 ψ 可看作涡量 ω 的一个参量，计算涡量 ω 的过程中需要先用式 (4-56) 中的第二式求解流函数 ψ ，这是一个对 ω 的积分运算，由于产生了积分常数 c ，所以 ψ 并不是唯一的，即： $\psi=\psi_0+c$ 。但是注意到式 (4-56) 中的 ψ 都是以导数的形式存在，所以积分常数 c 并不会影响 ω 的计算结果，可不做考虑。计算流函数 ψ 有两种方法：

(1) 直接对 $\partial^2/\partial x^2 + \partial^2/\partial y^2$ 的矩阵形式求逆得到积分矩阵，对 ω 进行积分：

$$\psi_{N^2 \times 1} = \left(\mathbf{D}_N^{(2)} \otimes \mathbf{I}_N + \mathbf{I}_N \otimes \mathbf{D}_N^{(2)} \right)^{-1} \omega_{N^2 \times 1} \quad (4-57)$$

(2) 利用第 3 章的方法，对方程做二维傅里叶变换，将积分运算转化为在频域上的除运算（注意用技巧避免分母为 0）：

$$\hat{\psi} = -\frac{\hat{\omega}}{k_x^2 + k_y^2} \quad (4-58)$$

因为方法一中存在对 N^2 阶方阵的求逆运算，运算量较大，所以这里采用方法二。

使用周期性边界条件，初始条件 $\omega(x, y, 0) = \text{sech}(x^2 + y^2/20)$ ，取 $\nu=0.001$ ，数值计算二维平流-扩散方程的代码如下：

程序 4-9

主程序代码如下：

```
clear all; close all;
L=20; N=128;
x=L/N*[-N/2:N/2-1]; y=x;
kx=(2*pi/L)*[0:N/2-1 -N/2:-1];
kx(1)=1e-6; ky=kx;
[X,Y]=meshgrid(x,y);
[kX,kY]=meshgrid(kx,ky);
K2=kX.^2+kY.^2;
%构造谱求导矩阵
h=2*pi/N; column=[0 0.5*(-1).^(1:N-1).*cot((1:N-1)*h/2)]';
D=(2*pi/L)*toeplitz(column,column([1 N:-1:2]));
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2];
D2=(2*pi/L)^2*toeplitz(column);
%初始条件
w=sech(X.^2+Y.^2/20);
%数值求解
v=0.001; t=0:10:30;
[t,wsol]=ode45('advection_diffusion',t,w(:),[],N,D,D2,K2,v);
%画图
for n=1:4
```

```

subplot(2,2,n)
gca=pcolor(reshape(wsol(n,:),N,N)); axis off
set(gca,'LineStyle','none'), shading interp
title(['t=' num2str(t(n))]), axis square
end

```

advection_diffusion.m 文件代码如下:

```

function dw=advection_diffusion(t,w,dummy,N,D,D2,K2,v)
w=reshape(w,N,N); wt=fft2(w);
psi=ifft2(wt./(-K2));
dw=reshape(v*(D2*w+w*D2')-(psi*D').*(D*w)+(D*psi).*(w*D'),N^2,1);

```

程序输出结果如图 4-12 所示,随着时间 t 的推移,涡量 ω 的分布出现了逆时针的旋转。如果将平流部分变号,旋转方向将变为顺时针,读者可自行尝试。

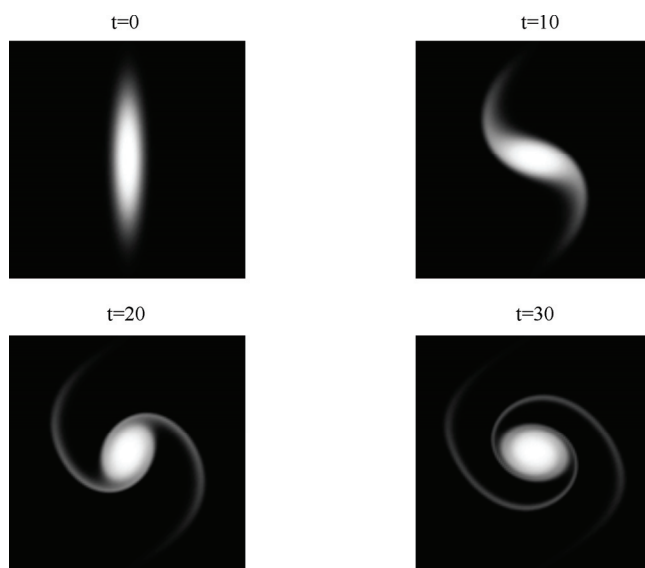


图 4-12 二维平流-扩散方程计算结果

下篇 第一类、第二类和第三类边界 条件下的谱方法

第5章 切比雪夫谱方法

第3、4章在进行数值求导时用到了离散傅里叶变换或周期插值函数，导致这些方法都暗含着周期性边界条件，所以在处理常见偏微分问题时有一定的局限。针对这一弊端，本章介绍适用于第一、二、三类边界条件的切比雪夫谱方法（Chebyshev spectral method）。简单来讲，前面章节把数学问题放在有界（bounded）、周期的（periodic）区间上来处理，而本章则是在有界、非周期的（nonperiodic）区间内解决数学问题。

5.1 切比雪夫求导矩阵的导出

5.1.1 吉布斯现象和龙格现象

首先，比较以下3种对有限区间内非周期函数插值的方法：

- (1) 用第4章的方法在周期性边界条件下对函数进行插值。
- (2) 在区间内等间距的点上对函数进行代数多项式插值。
- (3) 在区间内的切比雪夫点（Chebyshev points）上对函数进行代数多项式插值。

方法1采用了周期性边界条件，如果该函数在计算区间的边界附近迅速衰减到0或趋于某一特定值，那么就可以在周期性边界条件下对其精确插值（前面章节有这样的实例）。但一般来讲，方法1相当于将该函数强制转化为在边界处间断、周期为 L 的周期函数（ L 为区间宽度），且认为其插值函数的形式是周期 sinc 函数的线性叠加。如此得到的插值函数会在间断点附近产生震荡，带来误差。若增加插值所使用的离散点个数 N ，震荡将向间断点处靠近，且震荡幅度趋于一个常数，约等于间断点处跳变值的9%，这种现象称为吉布斯现象（Gibbs phenomenon）。

例如：对定义在 $[-\pi, \pi]$ 上的 $y=e^x$ 函数进行扩展，得到在边界处间断、周期为 2π 的周期函数，此周期函数的插值函数为：

$$p(x) = \sum_{j=1}^N y_j S_N(x - x_j) \quad (5-1)$$

其中, x_j 为区间 $[-\pi, \pi]$ 上间隔 $2\pi/N$ 的 N 个点 (即: $-\pi, -\pi+2\pi/N, -\pi+4\pi/N, \dots, \pi-2\pi/N$), y_j 为 x_j 处的函数值, S_N 为周期 sinc 函数, N 取 64。如图 5-1 所示, 插值函数 $p(x)$ 在间断点处出现了振荡, 极大地影响了插值精确度。实现上述过程的代码如下:

程序 5-1

```
clear all; close all;
L=2*pi; N=64; h=2*pi/N;
x=L/N*[-N/2:N/2-1]; y=exp(x);
subplot(2,1,1)
plot(x-L,y,'k',x,y,'k',x+L,y,'k','LineWidth',1.5)
axis([-10 10 -5 25]), xlabel x, ylabel y
subplot(2,1,2)
x2=-3/2*L:0.01:3/2*L; interpolant=0;
for n=1:length(x)
    interpolant=interpolant+ ...
        y(n)*sin(pi*(x2-x(n))/h)/((2*pi/h)*tan((x2-x(n))/2));
end
plot(x2,interpolant,'k','LineWidth',1.5)
axis([-10 10 -5 25]), xlabel x, ylabel p(x)
```

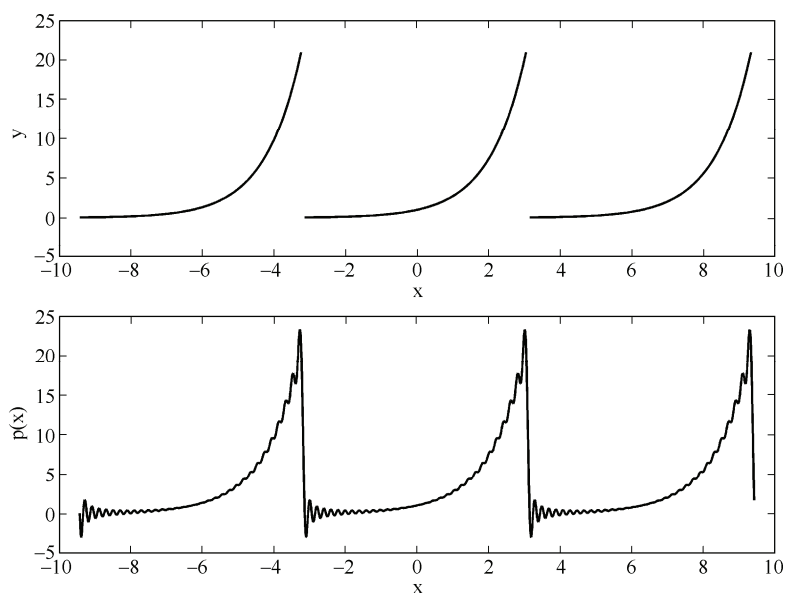


图 5-1 上图: 由 $[-\pi, \pi]$ 上的 $y=e^x$ 函数扩展得到的在边界处间断、周期为 2π 的周期函数;

下图: 插值函数出现了吉布斯现象

方法 2 是一种比较容易想到的方法：在宽为 L 的区间内均匀地选取 $N+1$ 个位置（即： $-L/2, -L/2+L/N, -L/2+2L/N, \dots, L/2-2L/N, L/2-L/N, L/2$ ），用 N 次代数多项式在这些位置对函数进行插值，插值函数的形式为：

$$p(x) = a_0 + a_1x + \dots + a_Nx^N \quad (5-2)$$

此方法没有使用周期性边界条件，避免了吉布斯现象的出现。但是， N 次代数多项式必然存在 $N-1$ 个极值，它将导致插值函数 $p(x)$ 在区间的边界处出现震荡，并随着 N 的增大而变得更加严重，这被称为龙格现象（Runge's phenomenon）。由于震荡带来误差，所以方法 2 也是行不通的，后面会用实例说明。

方法 3 在区间内选取了不均匀的 $N+1$ 个位置——切比雪夫点，在区间 $[-1, 1]$ 内的切比雪夫点的位置为：

$$x_j = \cos(j\pi/N), \quad j = 0, 1, \dots, N \quad (5-3)$$

可以把这些切比雪夫点理解为上半个单位圆上等间距的点在横轴的投影， $N=8$ 和 $N=16$ 的情况如图 5-2 所示。注意切比雪夫点是从右向左排序的。

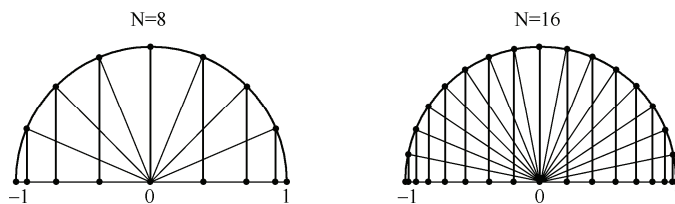


图 5-2 切比雪夫点是上半个单位圆上等间距的点在横轴的投影。左图： $N=8$ ，右图： $N=16$

下面这个例子使用方法 2 和方法 3 对函数 $y=1/(1+25x^2)$ 进行代数多项式插值，并比较了插值函数，代码如下：

程序 5-2

```
clear all; close all;
L=2;
for m=1:2
    N=16*m;
    for n=1:2
        if n==1
            x=L/N*[-N/2:N/2];
        else
            x=L/2*cos(pi*[0:N]/N);
        end
        y=1./(1+25*x.^2); x2=-L/2:0.01:L/2;
```

```

y2=polyval(polyfit(x,y,N),x2);
error=max(abs(y2-1./(1+25*x2.^2)));
subplot(2,2,n+2*(m-1))
plot(x2,y2,'k',x,y,'r','MarkerSize',15,'LineWidth',1.5)
axis([-1.1 1.1 -0.1 1.1]), xlabel x, ylabel y
title(['N=' num2str(N) ', Error_{max}=' num2str(error)])
end
end

```

程序输出结果如图 5-3 所示，在均匀点上对函数进行代数多项式插值会导致在区间边界附近出现很大误差，而在切比雪夫点上对函数进行代数多项式插值则几乎没有这个问题。随着 N 的增大，前者的误差会越来越大，而后者的误差约正比于 2^{-N} 。

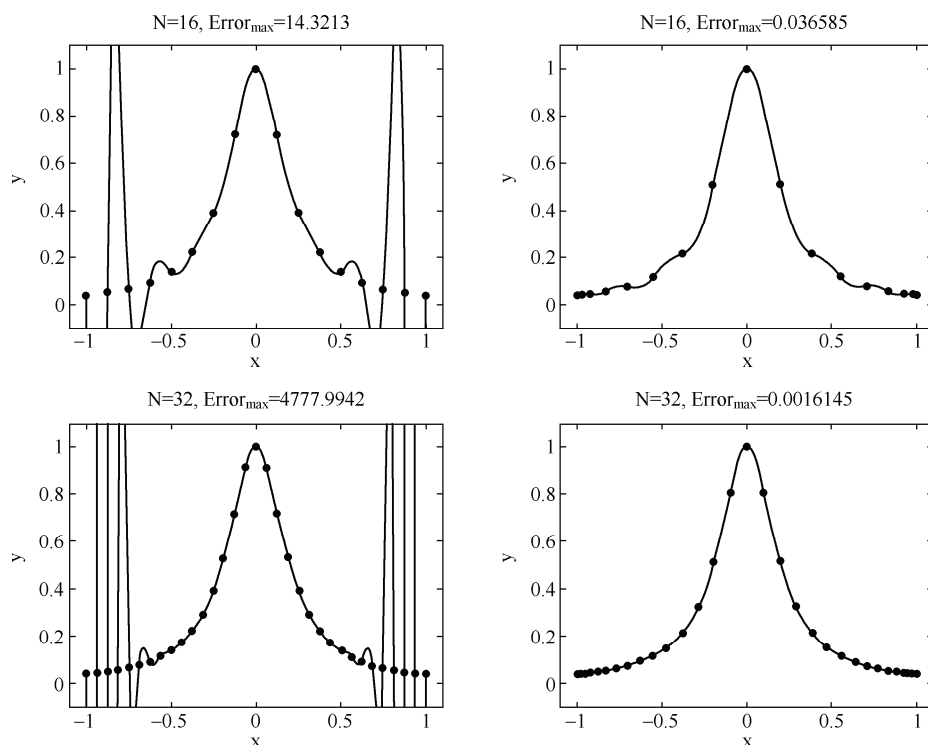


图 5-3 根据离散数据（点）得到插值函数（曲线）。左边两图：使用均匀点插值；
右边两图：使用切比雪夫点插值

通过上面的分析可知，由于方法 3 没有使用周期性边界条件，不存在间断点，从而杜绝了吉布斯现象，同时又通过切比雪夫点划分区间，避免了龙格现象，是一种针对有限区间内非周期函数的理想插值方法。

实际上，为了消除龙格现象，有很多非均匀划分区间的方法，使用切比雪夫点划分的

方法只是其中之一。这些方法都符合以下条件：当 $N \rightarrow \infty$ 时，点的密度分布满足

$$d \sim \frac{N}{\pi\sqrt{1-x^2}} \quad (5-4)$$

证明从略。

5.1.2 切比雪夫求导矩阵

根据 5.1.1 小节，对有限区间内的非周期函数进行数值求导的最佳方法是这样的：首先在计算区间内确定 $N+1$ 个切比雪夫点 $x_0, x_1, \dots, x_N$ ，在这些点上对函数进行代数多项式插值，得到最高次幂小于或等于 N 的插值函数 $p(x)$ ，然后求插值函数在切比雪夫点处的导数 $p'(x_0), p'(x_1), \dots, p'(x_N)$ 。由于上述过程是线性的，可以写成矩阵形式，所以本小节的目标就是总结其中的规律并构造切比雪夫求导矩阵，将求导运算转化为矩阵运算。

选取计算区间为 $[-1, 1]$ ，以便讨论。坐标被离散化为 $x_j = \cos(j\pi/N)$ ，其中 $j=0, 1, \dots, N$ ，用 $\mathbf{x}$ 表示 $N+1$ 维向量 $(x_0, x_1, \dots, x_N)^T$ 。与第 4 章不同，这里的 N 是任意正整数而不仅是正偶数，区间内的离散点数为 $N+1$ 个而不是 N 个，坐标 $x_0, x_1, \dots, x_N$ 是由大到小而不是由小到大，请注意区分。用 $\mathbf{u}$ 代表这些位置上的函数值组成的 $N+1$ 维向量 $(u_0, u_1, \dots, u_N)^T$ ， $p(x)$ 代表它的插值函数。定义 $\mathbf{D}_N$ 为 $(N+1) \times (N+1)$ 切比雪夫求导矩阵，并使得下式成立：

$$\mathbf{p}'(\mathbf{x}) = \mathbf{D}_N \mathbf{u} \quad (5-5)$$

先来看 $N=1$ 和 $N=2$ 的情况，然后推广到一般情况。 $N=1$ 时，只有 2 个插值点 $x_0=1$ 和 $x_1=-1$ ，以及相应的 2 个函数值 u_0 和 u_1 ，则插值函数为：

$$p(x) = \frac{1}{2}(1+x)u_0 + \frac{1}{2}(1-x)u_1 \quad (5-6)$$

对其求导，得：

$$p'(x) = \frac{1}{2}u_0 - \frac{1}{2}u_1 \quad (5-7)$$

为了满足式 (5-5)，构造 2×2 切比雪夫求导矩阵 $\mathbf{D}_1$ 为：

$$\mathbf{D}_1 = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \quad (5-8)$$

$N=2$ 时，有 3 个插值点 $x_0=1$ 、 $x_1=0$ 和 $x_2=-1$ ，以及相应的 3 个函数值 u_0 、 u_1 和 u_2 ，则插值函数为：

$$p(x) = \frac{1}{2}x(1+x)u_0 + (1+x)(1-x)u_1 + \frac{1}{2}x(x-1)u_2 \quad (5-9)$$

对其求导，得：

$$p'(x) = \left(x + \frac{1}{2}\right)u_0 - 2xu_1 + \left(x - \frac{1}{2}\right)u_2 \quad (5-10)$$

代入 $x_0=1$ 、 $x_1=0$ 和 $x_2=-1$ 就可以得到该位置的导数值。根据式 (5-5)， 3×3 切比雪夫求导矩阵 $\mathbf{D}_2$ 为：

$$\mathbf{D}_2 = \begin{pmatrix} \frac{3}{2} & -2 & \frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 2 & -\frac{3}{2} \end{pmatrix} \quad (5-11)$$

若继续如此计算更高阶的 $\mathbf{D}_N$ ，将会找到其中的规律。下面略过这个过程，直接给出对于任意 N 的切比雪夫求导矩阵 $\mathbf{D}_N$ 中每个元素的表达式：

$$\begin{aligned} (\mathbf{D}_N)_{00} &= \frac{2N^2+1}{6}, & (\mathbf{D}_N)_{NN} &= -\frac{2N^2+1}{6}, \\ (\mathbf{D}_N)_{jj} &= \frac{-x_j}{2(1-x_j^2)}, & j &= 1, \dots, N-1, \\ (\mathbf{D}_N)_{ij} &= \frac{c_i}{c_j} \frac{(-1)^{i+j}}{x_i - x_j}, & i \neq j, \quad i, j &= 0, \dots, N, \end{aligned} \quad (5-12)$$

其中 $(\mathbf{D}_N)_{ij}$ 代表切比雪夫求导矩阵 $\mathbf{D}_N$ 中第 $i+1$ 行第 $j+1$ 列的元素。且：

$$c_i = \begin{cases} 2, & i=0, N \\ 1, & i=1, \dots, N-1 \end{cases} \quad (5-13)$$

下式直观地给出了切比雪夫求导矩阵 $\mathbf{D}_N$ 的结构：

$$\mathbf{D}_N = \begin{pmatrix} \frac{2N^2+1}{6} & 2\frac{(-1)^j}{1-x_j} & \frac{1}{2}(-1)^N \\ \vdots & \frac{(-1)^{i+j}}{x_i-x_j} & \vdots \\ -\frac{1}{2}\frac{(-1)^i}{1-x_i} & \frac{-x_j}{2(1-x_j^2)} & \frac{1}{2}\frac{(-1)^{N+i}}{1+x_i} \\ \frac{(-1)^{i+j}}{x_i-x_j} & \vdots & \vdots \\ -\frac{1}{2}(-1)^N & -2\frac{(-1)^{N+j}}{1+x_j} & -\frac{2N^2+1}{6} \end{pmatrix} \quad (5-14)$$

由于后面的程序将反复用到切比雪夫求导矩阵 $\mathbf{D}_N$ 和相应的切比雪夫点，所以下面给

出函数 `cheb` 的定义文件 `cheb.m`，只需输入 N 的值就能返回 $\mathbf{D}_N$ 和相应的切比雪夫点，以便其他程序调用。

程序 5-3

`cheb.m` 文件代码如下：

```
function [D,x]=cheb(N)
if N==0, D=0; x=1; return, end
x=cos(pi*(0:N)/N)';
c=[2; ones(N-1,1); 2].*(-1).^(0:N)';
X=repmat(x,1,N+1);
dX=X-X';
D=(c*(1./c))./(dX+(eye(N+1)));
D=D-diag(sum(D'));
```

实际上，`cheb.m` 并不是严格按照式 (5-12) 来计算 $\mathbf{D}_N$ 的，对角线上的元素是由对角线以外的元素来计算的，即：

$$(\mathbf{D}_N)_{ii} = -\sum_{\substack{j=0 \\ j \neq i}}^N (\mathbf{D}_N)_{ij} \quad (5-15)$$

这样就给出了存在舍入误差的情况下更具稳定性的 $\mathbf{D}_N$ 。试想这个情形：对于离散函数值 $\mathbf{u}=(1, 1, \dots, 1)^T$ ，它的插值函数为一常数 $p(x)=1$ ，所以 $p'(x)=0$ 。这就要求 $\mathbf{D}_N$ 与 $(1, 1, \dots, 1)^T$ 的乘积必须是 $(0, 0, \dots, 0)^T$ ，于是就得到了式 (5-15)。

下面是 $N=1, 2, 3, 4, 5$ 时 `cheb` 输出的结果，注意到 $\mathbf{D}_N$ 是“中心对称”的，即：
 $(\mathbf{D}_N)_{ij} = -(\mathbf{D}_N)_{N-i, N-j}$

```
>> cheb(1)
```

```
ans =
```

```
0.5000    -0.5000
0.5000    -0.5000
```

```
>> cheb(2)
```

```
ans =
```

```
1.5000    -2.0000    0.5000
```

```

0.5000      0   -0.5000
-0.5000  2.0000  -1.5000

```

```
>> cheb(3)
```

```
ans =
```

```

3.1667  -4.0000   1.3333  -0.5000
1.0000  -0.3333  -1.0000   0.3333
-0.3333   1.0000   0.3333  -1.0000
0.5000  -1.3333   4.0000  -3.1667

```

```
>> cheb(4)
```

```
ans =
```

```

5.5000  -6.8284   2.0000  -1.1716   0.5000
1.7071  -0.7071  -1.4142   0.7071  -0.2929
-0.5000   1.4142         0  -1.4142   0.5000
0.2929  -0.7071   1.4142   0.7071  -1.7071
-0.5000   1.1716  -2.0000   6.8284  -5.5000

```

```
>> cheb(5)
```

```
ans =
```

```

8.5000 -10.4721   2.8944  -1.5279   1.1056  -0.5000
2.6180  -1.1708  -2.0000   0.8944  -0.6180   0.2764
-0.7236   2.0000  -0.1708  -1.6180   0.8944  -0.3820
0.3820  -0.8944   1.6180   0.1708  -2.0000   0.7236
-0.2764   0.6180  -0.8944   2.0000   1.1708  -2.6180
0.5000  -1.1056   1.5279  -2.8944  10.4721  -8.5000

```

$\mathbf{D}_N$ 是用于求 1 阶导数的切比雪夫求导矩阵, 要得到用于求 2 阶导数的切比雪夫求导矩阵, 一般有两种思路: (1) 用精确的表达式计算。(2) 直接对 $\mathbf{D}_N$ 平方。为了简便, 本书采用第 2 种思路, 同样, 还可以得到用于求高阶导数的切比雪夫求导矩阵:

$$\begin{aligned}
\frac{\partial}{\partial x} &\rightarrow \mathbf{D}_N \\
\frac{\partial^2}{\partial x^2} &\rightarrow \mathbf{D}_N^2 \\
\frac{\partial^3}{\partial x^3} &\rightarrow \mathbf{D}_N^3 \\
\frac{\partial^4}{\partial x^4} &\rightarrow \mathbf{D}_N^4
\end{aligned} \tag{5-16}$$

因为 $\mathbf{D}_N$ 的表达式是在区间 $[-1, 1]$ 上得到的，所以，在对其他区间上的函数求导前，还需要对 $\mathbf{D}_N$ 进行缩放，即： $\partial/\partial x \rightarrow (2/L)\mathbf{D}_N$ ， $\partial^n/\partial x^n \rightarrow [(2/L)\mathbf{D}_N]^n$ 。

下面的代码用切比雪夫求导矩阵计算 $u(x)=\text{sech}(x)$ 和 $u(x)=e^{\sin(2x)}$ 在 $[-1, 1]$ 上的 1、2 阶导数，并与精确解进行比较。

程序 5-4

```

clear all; close all;
L=2; N=32;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
D2=D^2; x=L/2*x;
for m=1:2
    if m==1
        u=sech(x);
        %导数的精确解
        du_exact(:,1)=-u.*tanh(x);
        du_exact(:,2)=u-2*u.^3;
    else
        u=exp(sin(2*x));
        %导数的精确解
        du_exact(:,1)=2*cos(2*x).*u;
        du_exact(:,2)=4*u.*(cos(2*x).^2-sin(2*x));
    end
    %切比雪夫谱方法求导
    du_cheb(:,1)=D*u;
    du_cheb(:,2)=D2*u;
    error=max(abs(du_cheb-du_exact));
    titles={'u(x)=sech(x)', 'u(x)=e^{sin(2x)}'};
    labels={'u'(x)', 'u''(x)'};

```

```

subplot(3,2,m)
plot(x,u,'k',x,u,'r','MarkerSize',13,'LineWidth',1.5)
title(titles(m)), xlabel x, ylabel u(x)
for n=1:2
    subplot(3,2,2*n+m)
    plot(x,du_exact(:,n),'k',x,du_cheb(:,n), ...
        'r','MarkerSize',13,'LineWidth',1.5)
    title(['Error_{max}=' num2str(error(n))])
    xlabel x, ylabel(labels(n))
end
end
end

```

程序输出结果如图 5-4 所示，曲线为精确的 $u(x)$ 、 $u'(x)$ 和 $u''(x)$ ，点分别为 $u(x_j)$ 及利用切比雪夫求导矩阵计算得到的 $u'(x_j)$ 和 $u''(x_j)$ 。可以看到，在 $N=32$ 的条件下，误差在 10^{-10} 至 10^{-14} 数量级。

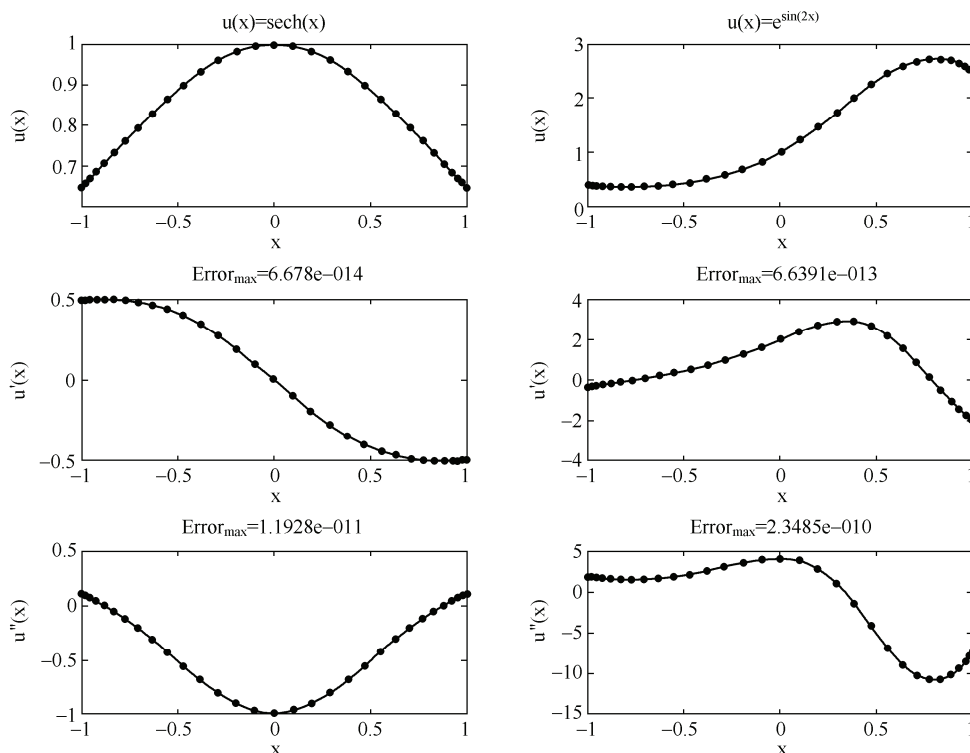


图 5-4 用切比雪夫求导矩阵计算 $u(x)=\text{sech}(x)$ 和 $u(x)=e^{\sin(2x)}$ 在 $[-1, 1]$ 上的 1、2 阶导数的数值解（点）并与精确解（曲线）比较

对于二维问题，先用切比雪夫点分别在 x 、 y 方向上划分区间 $[-1, 1]$ ，即 $\mathbf{x}=(x_0, x_1, \dots,$

$\mathbf{x}_N^T$ 和 $\mathbf{y}=(y_0, y_1, \dots, y_N)^T$ 。于是在 x - y 平面的 $-1 \leq x, y \leq 1$ 区域内得到了 $(N+1)^2$ 个点: $(x_0, y_0), (x_0, y_1), \dots, (x_0, y_N), (x_1, y_0), \dots, (x_1, y_N), \dots, (x_N, y_N)$, 如图 5-5 所示, 左图中 $N=8$, 右图中 $N=16$ 。

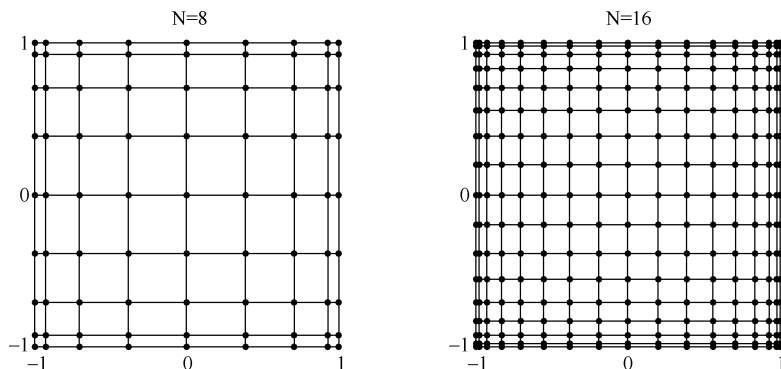


图 5-5 用切比雪夫点划分二维区域

用 $N+1$ 阶方阵或 $(N+1)^2$ 维向量 $\mathbf{u}$ 表示这些点上的函数值 (可用 Matlab 的 reshape 函数在二者之间互相转换):

$$\mathbf{u}_{(N+1) \times (N+1)} = \begin{pmatrix} u_{00} & u_{01} & \cdots & u_{0N} \\ u_{10} & u_{11} & \cdots & u_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N0} & u_{N1} & \cdots & u_{NN} \end{pmatrix} \quad (5-17)$$

$$\mathbf{u}_{(N+1)^2 \times 1} = (u_{00}, u_{10}, \dots, u_{N0}, u_{01}, u_{11}, \dots, u_{N1}, \dots, u_{NN})^T \quad (5-18)$$

其中, 元素 u_{ij} 对应的坐标是 (x_j, y_i) 。可将 $\partial^n u / \partial y^n$ 和 $\partial^n u / \partial x^n$ 写为矩阵形式:

$$\frac{\partial^n u}{\partial y^n} \rightarrow \mathbf{D}_N^n \mathbf{u}_{(N+1) \times (N+1)} \quad (5-19)$$

$$\frac{\partial^n u}{\partial x^n} \rightarrow \left(\mathbf{D}_N^n \left(\mathbf{u}_{(N+1) \times (N+1)} \right)^T \right)^T = \mathbf{u}_{(N+1) \times (N+1)} \left(\mathbf{D}_N^n \right)^T \quad (5-20)$$

或者写为:

$$\frac{\partial^n u}{\partial y^n} \rightarrow \left(\mathbf{I}_{N+1} \otimes \mathbf{D}_N^n \right) \mathbf{u}_{(N+1)^2 \times 1} \quad (5-21)$$

$$\frac{\partial^n u}{\partial x^n} \rightarrow \left(\mathbf{D}_N^n \otimes \mathbf{I}_{N+1} \right) \mathbf{u}_{(N+1)^2 \times 1} \quad (5-22)$$

其中, $\mathbf{I}_{N+1}$ 为 $N+1$ 阶单位矩阵。

5.2 狄利克莱边界条件（第一类边界条件）

狄利克莱边界条件（Dirichlet boundary condition），又称为第一类边界条件，它给出了未知函数在边界上的值。即： $u|_{\partial\Omega}=g$ ，其中， $\partial\Omega$ 为计算区域 Ω 的边界， g 为已知函数。

5.2.1 一维泊松方程

在狄利克莱边界条件下，考虑一维泊松问题：

$$u'' = f(x), \quad -1 < x < 1, \quad u(\pm 1) = 0 \quad (5-23)$$

其中， $f(x)$ 为已知函数， $u(x)$ 为待求函数。将横轴上的区间 $[-1, 1]$ 离散化为向量 $\mathbf{x}=(x_0, x_1, \dots, x_N)^T$ ，相应地， $u(x)$ 被离散化为向量 $\mathbf{u}=(u_0, u_1, \dots, u_N)^T$ 。则式（5-23）可以写为矩阵形式：

$$\mathbf{D}_N^2 \mathbf{u} = \mathbf{f}(\mathbf{x}), \quad u_0 = u_N = 0 \quad (5-24)$$

式（5-24）代表了含有 $N+1$ 个方程的方程组，其中未知数有 $N-1$ 个： $u_1, u_2, \dots, u_{N-1}$ ，比方程个数少 2 个。实际上，只需要 $N-1$ 个方程就能求解这些未知数，不妨忽略边界上的方程，即：删去矩阵 $\mathbf{D}_N^2$ 的首尾行以及向量 $\mathbf{f}(\mathbf{x})$ 的首尾元素。又因为 $u_0=u_N=0$ ，矩阵 $\mathbf{D}_N^2$ 的首尾列与其相乘的结果也是 0，所以删去矩阵 $\mathbf{D}_N^2$ 的首尾列和向量 $\mathbf{u}$ 的首尾元素。于是就得到了在狄利克莱边界条件 $u(\pm 1)=0$ 下的 2 阶切比雪夫求导矩阵——将矩阵 $\mathbf{D}_N^2$ 的首尾行、首尾列删除后的 $N-1$ 阶方阵，如图 5-6 所示，此矩阵将在后续内容中反复用到。

图 5-6 在狄利克莱边界条件 $u(\pm 1)=0$ 下修改切比雪夫求导矩阵

如果这里用“~”表示删除矩阵首尾行和首尾列、删除向量首尾元素的操作，那么求解式（5-23），只需：

$$\tilde{\mathbf{u}} = (\tilde{\mathbf{D}}_N^2)^{-1} \tilde{\mathbf{f}}(\tilde{\mathbf{x}}) \quad (5-25)$$

注意，必须先对 $\mathbf{D}_N$ 平方，再删除其首尾行和首尾列，这个顺序不能颠倒。此外，通过式（5-25）得到 $N-1$ 维向量 $\tilde{\mathbf{u}}$ 后，一定要在其首尾补 0。

通过在切比雪夫求导矩阵上乘以缩放因子，上述方法可以从区间 $[-1, 1]$ 推广到任意区间，下面的泊松方程为例：

$$u'' = x^2 - x, \quad -2 < x < 2, \quad u(\pm 2) = 0 \quad (5-26)$$

它的精确解为 $u(x) = x^4/12 - x^3/6 + 2x/3 - 4/3$ 。计算数值解并与精确解比较的代码如下：

程序 5-5

```
clear all; close all;
L=4; N=10;
[D,x]=cheb(N); D=D/(L/2);
D2=D^2; D2=D2(2:N,2:N); x=L/2*x;
f=x(2:N).^2-x(2:N);
u=D2\f; u=[0;u;0];
error=x.^4/12-x.^3/6+2*x/3-4/3-u;
X=-L/2:0.1:L/2;
exact=X.^4/12-X.^3/6+2*X/3-4/3;
subplot(2,2,1)
plot(X,exact,'k',x,u,'r','MarkerSize',16,'LineWidth',1.5)
title(['Error_{max}=' num2str(max(abs(error)))]), xlabel x, ylabel u
subplot(2,2,2)
plot(x,error,'r','MarkerSize',16)
xlabel x, ylabel Error
```

程序输出结果如图 5-7 所示。 $N=10$ 时，精确解与数值解吻合得很好，最大误差在 10^{-15} 数量级，而且边界处的误差最小。

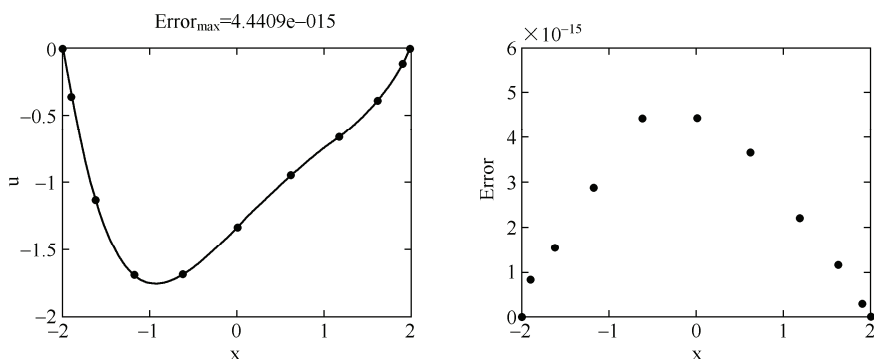


图 5-7 左图：精确解（曲线）与数值解（点）的最大误差在 10^{-15} 数量级。右图：误差分布

如果泊松方程的狄利克莱边界条件更具一般性，比如：

$$u'' = x^2 - x, \quad -2 < x < 2, \quad u(2) = 2, \quad u(-2) = -1 \quad (5-27)$$

那么，有两种方法解决这一问题。

方法 1:

式 (5-27) 的解可写为 $u(x)=u_0(x)+c_1x+c_2$ 的形式, 其中 $u_0(x)$ 是满足式 (5-26) 的特殊解。选取合适的 c_1 、 c_2 使多项式 c_1x+c_2 满足式 (5-27) 的边界条件, 即: $(c_1x+c_2)|_{x=2}=2$, $(c_1x+c_2)|_{x=-2}=-1$, 得到 $c_1=0.75$, $c_2=0.5$ 。毫无疑问, $u_0(x)+0.75x+0.5$ 必是式 (5-27) 的解。

方法 2:

如图 5-8 所示, 将矩阵 $\mathbf{D}_N^2$ 的首尾行分别修改为 $1\ 0\ 0\ \dots\ 0$ 和 $0\ \dots\ 0\ 0\ 1$, 以使得向量 $\mathbf{u}$ 和向量 $\mathbf{f}(\mathbf{x})$ 的首尾元素相等。然后将向量 $\mathbf{f}(\mathbf{x})$ 的首尾元素分别改为 u_0 和 u_N (函数 $u(x)$ 在边界的取值), 最后通过下式计算向量 $\mathbf{u}$:

$$\mathbf{u} = (\hat{\mathbf{D}}_N^2)^{-1} \hat{\mathbf{f}}(\mathbf{x}) \quad (5-28)$$

$$\begin{pmatrix} u_0 \\ f(x_1) \\ \vdots \\ f(x_{N-1}) \\ u_N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ & & & & \mathbf{D}_N^2 & & & \\ & & & & & & & \\ 0 & \dots & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}$$

图 5-8 在更具一般性的狄利克莱边界条件下修改切比雪夫求导矩阵

其中, 矩阵和向量上的“ $\wedge$ ”代表对矩阵、向量进行上述修改。在方法 2 中, 不但利用了 $N-1$ 个方程求解未知数 $u_1, u_2, \dots, u_{N-1}$, 还额外加了两个方程 (对应于矩阵 $\mathbf{D}_N^2$ 的首尾行), 以确保 $\mathbf{u}$ 的首尾元素必满足边界条件。

用方法 1 和方法 2 求解式 (5-27) 并与精确解 $u(x)=x^4/12-x^3/6+17x/12-5/6$ 比较的代码如下:

程序 5-6

```
clear all; close all;
L=4; N=10;
[D,x]=cheb(N); D=D/(L/2); x=L/2*x;
%方法 1
D2=D^2; D2=D2(2:N,2:N);
f=x(2:N).^2-x(2:N);
u1=D2\f; u1=[0;u1;0];
u1=u1+0.75*x+0.5;
%方法 2
D2=D^2; D2([1 N+1],:)=0;
D2(1,1)=1; D2(N+1,N+1)=1;
f=x(2:N).^2-x(2:N);
```

```

f=[2;f;-1]; u2=D2\f;
%误差
exact=x.^4/12-x.^3/6+17*x/12-5/6;
error1=abs(exact-u1);
error2=abs(exact-u2);
subplot(2,2,1)
plot(x,u1,'or',x,u2,'+b',x,exact,'k','MarkerSize',10,'LineWidth',1.5)
title(['Error1_{max}= ' num2str(max(error1))]; ...
      ['Error2_{max}= ' num2str(max(error2))]);
xlabel x, ylabel u, legend('方法 1','方法 2',2)
subplot(2,2,2)
plot(x,error1,'or',x,error2,'+b','MarkerSize',10,'LineWidth',1.5)
xlabel x, ylabel Error, legend('方法 1','方法 2',2)

```

程序输出结果如图 5-9 所示，在 $N=10$ 的情况下，方法 1 的计算结果和方法 2 的计算结果均与解析解吻合较好，误差分别在 10^{-15} 和 10^{-16} 数量级。此外，方法 1 在各处的误差要普遍高于方法 2 在各处的误差。

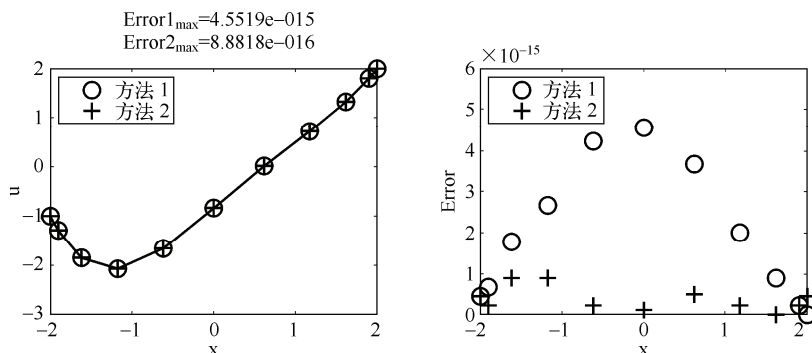


图 5-9 左图：方法 1 的计算结果 (○)、方法 2 的计算结果 (+)、解析解 (曲线)；右图：方法 1 的误差 (○)、方法 2 的误差 (+)

5.2.2 二维泊松方程

对于含有边界条件的二维泊松方程 (5-29)，不能直接使用式 (5-21) 和式 (5-22) 求解，还需要根据边界条件对矩阵和向量进行相应修改。

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u = f(x, y), \quad -1 < x, y < 1, \quad u|_{|x|=1} = u|_{|y|=1} = 0 \quad (5-29)$$

考虑到函数 u 在边界上的取值为 0，所以将图 5-6 的思路推广到二维平面上来。用 $N-1$

阶方阵或 $(N-1)^2$ 维向量 $\tilde{\mathbf{u}}$ 表示除边界外所有位置的函数值，即：

$$\tilde{\mathbf{u}}_{(N-1) \times (N-1)} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1(N-1)} \\ u_{21} & u_{22} & \cdots & u_{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ u_{(N-1)1} & u_{(N-1)2} & \cdots & u_{(N-1)(N-1)} \end{pmatrix} \quad (5-30)$$

$$\tilde{\mathbf{u}}_{(N-1)^2 \times 1} = (u_{11}, u_{21}, \dots, u_{(N-1)1}, u_{12}, u_{22}, \dots, u_{(N-1)2}, \dots, u_{(N-1)(N-1)})^T \quad (5-31)$$

那么，可在 $u|_{|x|=1}=u|_{|y|=1}=0$ 的条件下将式 (5-29) 中的 $\partial^2 u / \partial y^2$ 和 $\partial^2 u / \partial x^2$ 写为矩阵形式：

$$\frac{\partial^2 u}{\partial y^2} \rightarrow (\mathbf{I}_{N-1} \otimes \tilde{\mathbf{D}}_N^2) \tilde{\mathbf{u}}_{(N-1)^2 \times 1} \quad (5-32)$$

$$\frac{\partial^2 u}{\partial x^2} \rightarrow (\tilde{\mathbf{D}}_N^2 \otimes \mathbf{I}_{N-1}) \tilde{\mathbf{u}}_{(N-1)^2 \times 1} \quad (5-33)$$

则拉普拉斯算符可写为：

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \rightarrow \tilde{\mathbf{L}} = \tilde{\mathbf{D}}_N^2 \otimes \mathbf{I}_{N-1} + \mathbf{I}_{N-1} \otimes \tilde{\mathbf{D}}_N^2 \quad (5-34)$$

求解式 (5-29)，只需：

$$\tilde{\mathbf{u}}_{(N-1)^2 \times 1} = \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{f}}_{(N-1)^2 \times 1} \quad (5-35)$$

其中，向量 $\mathbf{f}$ 为 $f(x, y)$ 在各个离散点的取值，上面的“~”代表对其进行删除所有边界值的操作， $\mathbf{D}_N^2$ 上面的“~”代表对其进行删除首尾行、首尾列的操作。在这些过程中， $(N+1)^2$ 维向量变为 $(N-1)^2$ 维向量， $N+1$ 阶方阵变为 $N-1$ 阶方阵。当然，通过式 (5-35) 得到 $\tilde{\mathbf{u}}$ 之后，还需要在边界对应的位置上补 0。

图 5-10 分别显示了在不考虑边界条件情况下 (式 (5-21) 和式 (5-22)) 和函数在边界取值为 0 的情况下 (式 (5-32) 和式 (5-33))， $\partial^2 / \partial x^2$ 、 $\partial^2 / \partial y^2$ 和 $\partial^2 / \partial x^2 + \partial^2 / \partial y^2$ 的矩阵形式，每个点表示一个非零元素。 $N=5$ 时，上三图均为 6^2 阶方阵，下三图均为 4^2 阶方阵。与有限差分法所构造的二维求导矩阵相比，二维切比雪夫求导矩阵要更稠密。此外，较之前者，后者的精度更高，所以只需较小的 N 就可以计算出令人满意的结果，大大地减小了运算量。代码如下：

程序 5-7

```
clear all; close all;
L=2; N=5;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
I=eye(N+1); LA=kron(I,D2)+kron(D2,I);
```

```

subplot(2,3,1), spy(kron(I,D2),'k',8)
title('kron(I_{N+1},D^2_N)')
subplot(2,3,2), spy(kron(D2,I),'k',8)
title('kron(D^2_N,I_{N+1})')
subplot(2,3,3), spy(LA,'k',8)
title('kron(I_{N+1},D^2_N)+kron(D^2_N,I_{N+1})')
D2=D2(2:N,2:N);
I=eye(N-1); LA=kron(I,D2)+kron(D2,I);
subplot(2,3,4), spy(kron(I,D2),'k',10)
title('kron(I_{N-1},D^2_N(2:N,2:N))')
subplot(2,3,5), spy(kron(D2,I),'k',10)
title('kron(D^2_N(2:N,2:N),I_{N-1})')
subplot(2,3,6), spy(LA,'k',10)
title(['kron(I_{N-1},D^2_N(2:N,2:N)); ...
    [' +kron(D^2_N(2:N,2:N),I_{N-1})']])
    
```

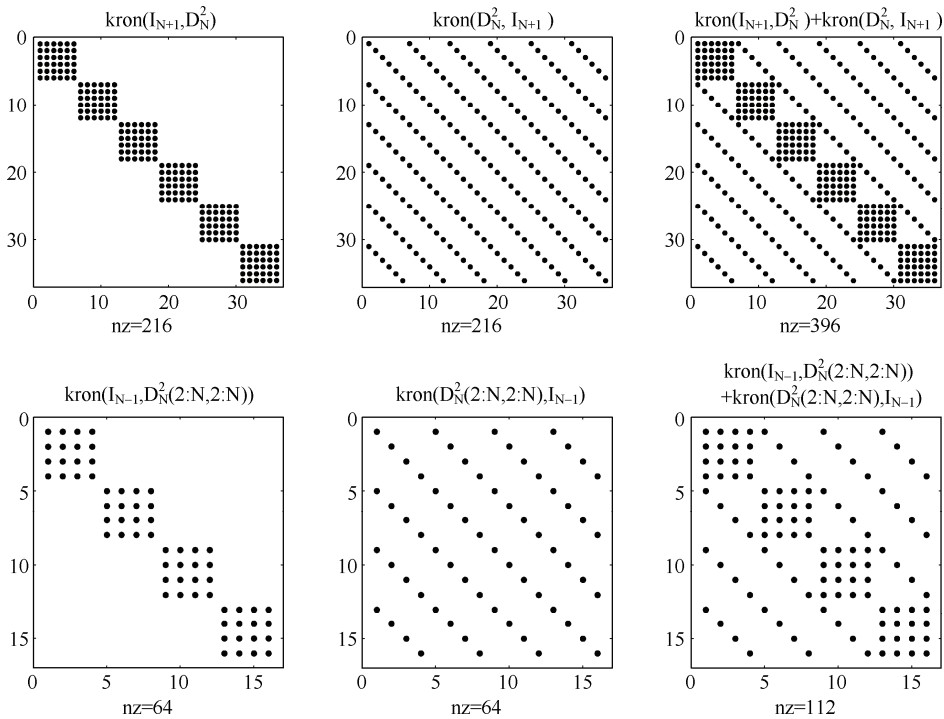


图 5-10 不考虑边界条件（上三图）和函数在边界上取值为 0（下三图）时， $\partial^2/\partial x^2$ 、 $\partial^2/\partial y^2$ 以及 $\partial^2/\partial x^2 + \partial^2/\partial y^2$ 的矩阵形式， $N=5$

接下来以二维泊松方程（5-36）为例给出具体求解过程，代码如下：

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u = 6(x+1)(y+0.1)^3, \quad -1 < x, y < 1, \quad u|_{|x|=1} = u|_{|y|=1} = 0 \quad (5-36)$$

程序 5-8

```

clear all; close all;
L=2; N=20;
%构造拉普拉斯算符矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
D2=D2(2:N,2:N);
I=eye(N-1); LA=kron(I,D2)+kron(D2,I);
x=L/2*x; y=x; [x,y]=meshgrid(x,y);
X=x(2:N,2:N); Y=y(2:N,2:N);
f=6*(X(:)+1).*(Y(:)+0.1).^3;
%求解
u=zeros(N+1);
u(2:N,2:N)=reshape(LA\f,N-1,N-1);
%画图
[x2,y2]=meshgrid(-L/2:0.04:L/2,-L/2:0.04:L/2);
u2=interp2(x,y,u,x2,y2,'cubic');
mesh(x2,y2,u2), xlabel x, ylabel y, zlabel u
view(-60,30), axis([-1 1 -1 1 -0.3 0.1])

```

因为 N 仅为 20, 且原点附近的网格比较稀疏, 所以直接画出来的解的曲面将不够光滑。为解决这一问题, 本程序使用了二维数据内插值函数 `interp2`, 使最终结果建立在稠密、均匀的网格上, 如图 5-11 所示。

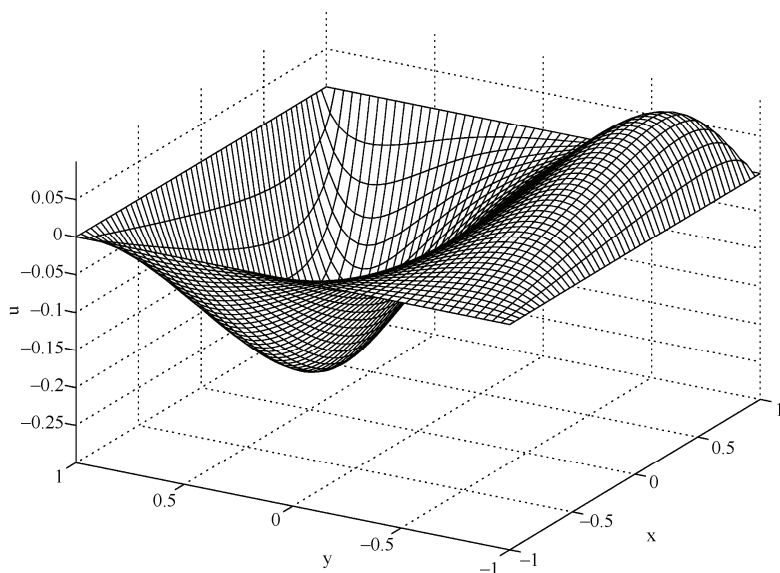


图 5-11 二维泊松方程的解

再来考虑具有更复杂边界条件的二维泊松方程，如式（5-37）所示（其中的 $\text{sinc}(x)$ 函数定义为 $\sin(\pi x)/\pi x$ ）：

$$\begin{cases} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u = 6(x+1)(y+0.1)^3, & -1 < x, y < 1 \\ u|_{x=1, |y| \leq 1/2} = 1/2 - |y| \\ u|_{y=1} = \text{sinc}(4x)/2 \\ u|_{x=-1} = u|_{y=-1} = u|_{x=1, |y| > 1/2} = 0 \end{cases} \quad (5-37)$$

针对这种更具一般性的二维边界条件，可将图 5-8 所示的思路推广到二维空间上来，先写出不含边界条件的拉普拉斯算符的矩阵形式（注意与式（5-34）区别）：

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \rightarrow \mathbf{L} = \mathbf{D}_N^2 \otimes \mathbf{I}_{N+1} + \mathbf{I}_{N+1} \otimes \mathbf{D}_N^2 \quad (5-38)$$

然后对矩阵 $\mathbf{L}$ 进行修改，代码如下：

程序 5-9

```
clear all; close all;
L=2; N=5;
%构造拉普拉斯算符矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
I=eye(N+1); LA=kron(I,D2)+kron(D2,I);
x=L/2*x; y=x; [x,y]=meshgrid(x,y);
X=x(:); Y=y(:);
subplot(2,3,1), spy(LA,'k',8)
title('kron(I_{N+1},D^2_N)+kron(D^2_N,I_{N+1})')
%修改拉普拉斯算符矩阵
bound=find(abs(X)==L/2|abs(Y)==L/2);
LA(bound,:)=0;
subplot(2,3,2), spy(LA,'k',8)
title('L(bound,:)=0')
LA(bound,bound)=eye(4*N);
subplot(2,3,3), spy(LA,'k',8)
title('L(bound,bound)=eye(4*N)')
```

其中，语句“`bound=find(abs(X)==L/2|abs(Y)==L/2);`”表示：找到矩阵 $\mathbf{L}$ 中所有对应于边界的行的序号，之后将这些行的所有元素替换为 0，最后把这些行中第 i 个元素设置为 1（ i 是该行的序号）。程序输出的结果显示了这一过程，如图 5-12 所示。

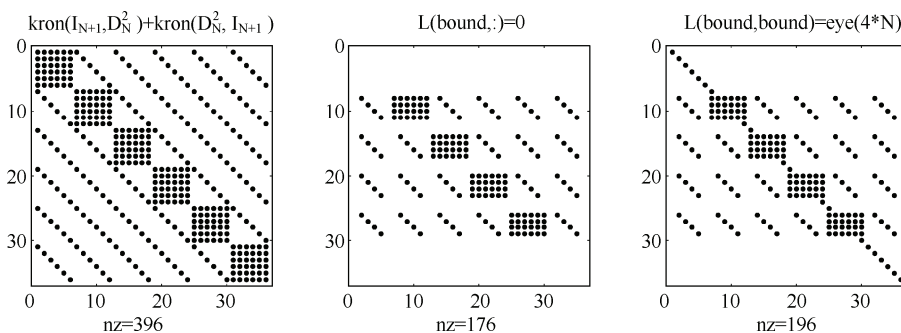


图 5-12 修改拉普拉斯算符矩阵的过程

在 L 上加 “ $\wedge$ ” 表示修改后的拉普拉斯算符矩阵。对于式 (5-39)，容易知道：在边界处，向量 f 的元素和向量 u 的元素相等；在其他位置，向量 f 的元素等于 $\partial^2 u / \partial x^2 + \partial^2 u / \partial y^2$ 在该处的值。

$$f_{(N+1)^2 \times 1} = \hat{L} u_{(N+1)^2 \times 1} \quad (5-39)$$

类似于式 (5-28)，求解式 (5-37) 只需：

$$u_{(N+1)^2 \times 1} = \hat{L}^{-1} f_{(N+1)^2 \times 1} \quad (5-40)$$

其中， f 上的 “ $\wedge$ ” 表示：在边界处， f 的取值与 $u(x, y)$ 的边界条件一样，而在其他位置的取值为 $f(x, y) = 6(x+1)(y+0.1)^3$ 。具体代码如下：

程序 5-10

```
clear all; close all;
L=2; N=40;
%构造拉普拉斯算符矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
I=eye(N+1); LA=kron(I,D2)+kron(D2,I);
x=L/2*x; y=x; [x,y]=meshgrid(x,y);
X=x(:); Y=y(:);
%修改拉普拉斯算符矩阵
bound=find(abs(X)==L/2|abs(Y)==L/2);
LA(bound,:)=0; LA(bound,bound)=eye(4*N);
%给 f 加入边界条件
f=6*(X+1).*(Y+0.1).^3;
f(bound)=(Y(bound)==L/2).*sinc(4*X(bound))/2+...
(X(bound)==L/2).*max(0,(0.5-abs(Y(bound)))));
%求解
u=reshape(LA\f,N+1,N+1);
%画图
```

```
x2=-L/2:0.04:L/2; y2=x2;
u2=interp2(x,y,u,x2,y2','cubic');
mesh(x2,y2,u2)
view(-60,30), axis([-1 1 -1 1 -0.25 0.5])
xlabel x, ylabel y, zlabel u
```

程序输出结果如图 5-13 所示：

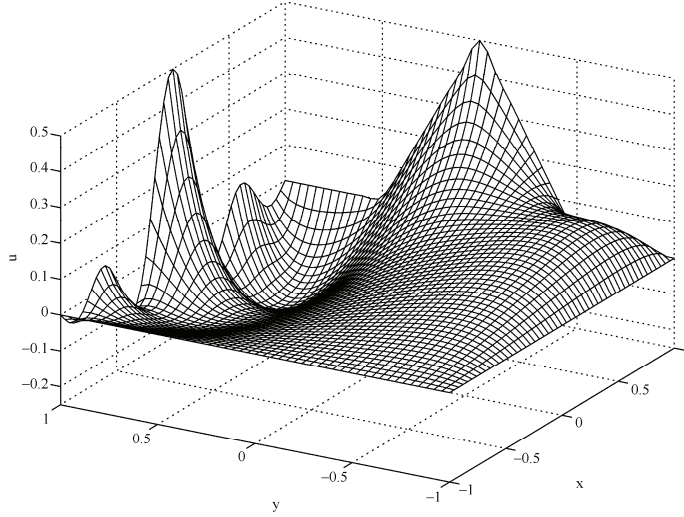


图 5-13 复杂边界条件下二维泊松方程的解

5.2.3 Allen-Cahn 方程

Allen-Cahn 方程的形式为：

$$\begin{cases} \frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + u - u^3, & -1 < x < 1 \\ u|_{x=\pm 1} = \pm 1 \\ u|_{t=0} = 0.53x - 0.47 \sin(1.5\pi x) \end{cases} \quad (5-41)$$

离散化 x 和 $u(x, t)$ 之后，式 (5-41) 中的 $\partial^2/\partial x^2$ 可用矩阵 $\mathbf{D}_N^2$ 计算， $\partial/\partial t$ 可用 ode45 函数计算。但需要把边界条件转化为 ode45 函数能直接处理的形式，所以将上式写为：

$$\begin{cases} \frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + u - u^3, & -1 < x < 1 \\ \frac{\partial u}{\partial t} \Big|_{x=\pm 1} = 0 \\ u|_{t=0} = 0.53x - 0.47 \sin(1.5\pi x) \end{cases} \quad (5-42)$$

容易知道， $u(\pm 1, 0)=\pm 1$ 及 $\partial u/\partial t|_{x=\pm 1}=0$ 等价于 $u(\pm 1, t)=\pm 1$ ，所以式 (5-41) 和式 (5-42)

是等价的 ($u(\pm 1, 0) = \pm 1$ 是隐含在初始条件里的)。当 $\varepsilon = 0.01$ 时, 求解式 (5-42) 的代码如下:

程序 5-11

主程序代码如下:

```
clear all; close all;
L=2; N=20;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
x=L/2*x; D2=D^2;
%初始条件
u=0.53*x-0.47*sin(1.5*pi*x);
%求解
t=[0:2:100]; epsilon=0.01;
[t,usol]=ode45('allen_cahn',t,u,[],D2,epsilon);
%画图
subplot('position',[0.15 0.58 0.7 0.4])
mesh(x,t,usol), view(-60,55)
xlabel x, ylabel t, zlabel u
axis([-1 1 0 100 -1 round(max(usol(:)))])
subplot('position',[0.15 0.08 0.7 0.4])
X=L/2:-0.02:-L/2;
u=interp2(x,t,usol,X,t,'cubic');
mesh(X,t,u), view(-60,55)
xlabel x, ylabel t, zlabel u
axis([-1 1 0 100 -1 round(max(u(:)))])
```

文件 allen_cahn.m 代码如下:

```
function du=allen_cahn(t,u,dummy,D2,epsilon)
du=epsilon*D2*u+u-u.^3;
du([1 end])=0;
```

程序输出结果如图 5-14 所示, 由于计算范围中部的网格比边缘的稀疏, 而且切比雪夫求导矩阵的计算精度高, 无需较大 N 值, 所以曲面在 u 变化快的地方就不够光滑 (上图)。用 interp2 函数插值得到的密集、均匀网格上的结果更准确、美观 (下图)。

若 Allen-Cahn 方程的边界条件包含时间 t , 比如:

$$\begin{cases} \frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + u - u^3, & -1 < x < 1 \\ u|_{x=1} = 1 + \sin^2(0.2t) \\ u|_{x=-1} = -1 \\ u|_{t=0} = 0.53x - 0.47 \sin(1.5\pi x) \end{cases} \quad (5-43)$$

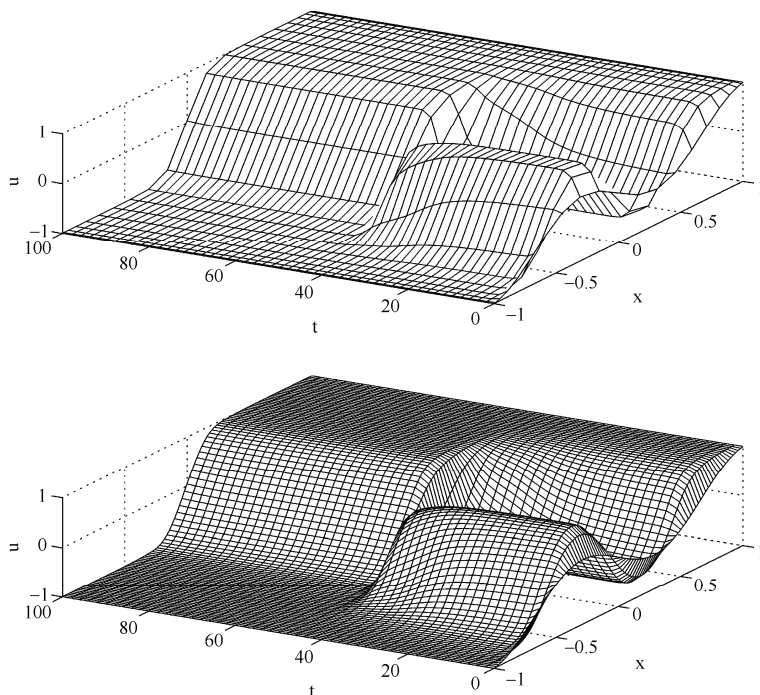


图 5-14 上图：计算 Allen-Cahn 方程的结果；下图：利用 interp2 插值并加密、均匀化网格之后的结果

类似地，可将其写为 ode45 函数能直接求解的等价形式：

$$\left\{ \begin{array}{l} \frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + u - u^3, \quad -1 < x < 1 \\ \frac{\partial u}{\partial t} \Big|_{x=1} = 0.2 \sin(0.4t) \\ \frac{\partial u}{\partial t} \Big|_{x=-1} = 0 \\ u \Big|_{t=0} = 0.53x - 0.47 \sin(1.5\pi x) \end{array} \right. \quad (5-44)$$

只需对程序 5-11 中的 allen_cahn.m 文件稍加改动就能实现式 (5-44) 的求解，代码如下：

程序 5-12

文件 allen_cahn.m 代码如下：

```
function du=allen_cahn(t,u,dummy,D2,epsilon)
du=epsilon*D2*u+u-u.^3;
du(1)=sin(2*t/5)/5;
du(end)=0;
```

程序输出结果如图 5-15 所示。本例和上例的问题均来源于文献[2]，得到的结果与文

献[2]完全相同，但这里给出的方法更简洁、精确，有兴趣的读者可以比较一下。

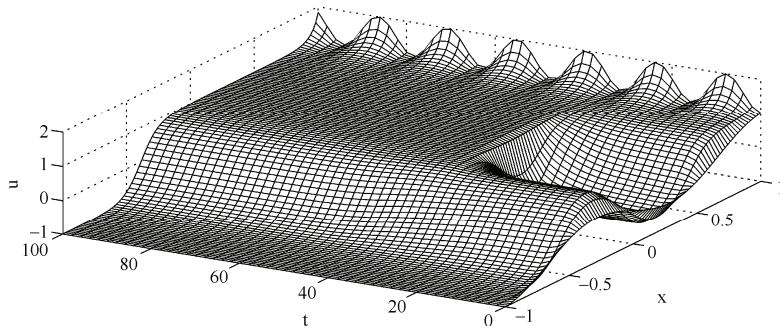


图 5-15 Allen-Cahn 方程的数值结果

5.2.4 二维热传导方程

考虑如下二维热传导方程：

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -2 < x, y < 2 \\ u|_{|x|=2} = u|_{|y|=2} = 0 \\ u|_{t=0} = 1 - \sqrt{x^2 + y^2}, & x^2 + y^2 \leq 1 \\ u|_{t=0} = 0, & x^2 + y^2 > 1 \end{cases} \quad (5-45)$$

其中， t 代表时间， x, y 代表空间坐标， $u(x, y, t)$ 代表 t 时刻区域内的温度分布，边界条件给出了边界处的温度。用方阵 $\tilde{\mathbf{u}}_{(N-1) \times (N-1)}$ 表示除边界以外位置的离散化函数值（定义见式 (5-30)），则有：

$$\frac{\partial^2 u}{\partial y^2} \rightarrow \tilde{\mathbf{D}}_N^2 \tilde{\mathbf{u}}_{(N-1) \times (N-1)} \quad (5-46)$$

$$\frac{\partial^2 u}{\partial x^2} \rightarrow \left(\tilde{\mathbf{D}}_N^2 \left(\tilde{\mathbf{u}}_{(N-1) \times (N-1)} \right)^T \right)^T = \tilde{\mathbf{u}}_{(N-1) \times (N-1)} \left(\tilde{\mathbf{D}}_N^2 \right)^T \quad (5-47)$$

为了避免出现 $(N-1)^2$ 阶大型方阵，所以此处不用式 (5-34) 的形式表示拉普拉斯算符。只有在迫不得已的情况下才可以使用式 (5-34)，如：特征值问题、拉普拉斯算符的逆运算等。

根据式 (5-46)、式 (5-47) 处理二维热传导方程等号右边的两项，用 `ode45` 计算左边的 $\partial/\partial t$ ，代码如下：

程序 5-13

主程序代码如下：

```
clear all; close all;
```

```

L=4; N=40;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
D2=D2(2:N,2:N); x=L/2*x; y=x;
[X,Y]=meshgrid(x(2:N),y(2:N));
%初始条件
u=max(0,1-sqrt(X.^2+Y.^2));
%求解
t=[0 0.02 0.1 0.5];
[t,usol]=ode45('heat',t,u(:),[],N,D2);
%画图
for n=1:4
    subplot(2,2,n)
    u=zeros(N+1);
    u(2:N,2:N)=reshape(usol(n,:),N-1,N-1);
    surf(x,y,u), shading interp
    axis([-2 2 -2 2 0 1])
    xlabel x, ylabel y, zlabel u
    title(['t=' num2str(t(n))]);
end

```

文件 heat.m 代码如下：

```

function du=heat(t,u,dummy,N,D2)
u=reshape(u,N-1,N-1);
du=reshape(D2*u+u*D2',(N-1)^2,1);

```

程序的输出结果如图 5-16 所示，热量迅速地从中央高温区域向四周扩散，温差越小，扩散速度越慢，直到各个位置的温度都趋于 0。

下面分析更复杂边界条件下的二维热传导方程：

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -1 < x, y < 1 \\ u|_{x=1} = 1 - |y| \\ u|_{x=-1} = u|_{|y|=1} = 0 \\ u|_{r=0} = 1 - \sqrt{(x-1)^2 + y^2}, & (x-1)^2 + y^2 \leq 1 \\ u|_{r=0} = 0, & (x-1)^2 + y^2 > 1 \end{cases} \quad (5-48)$$

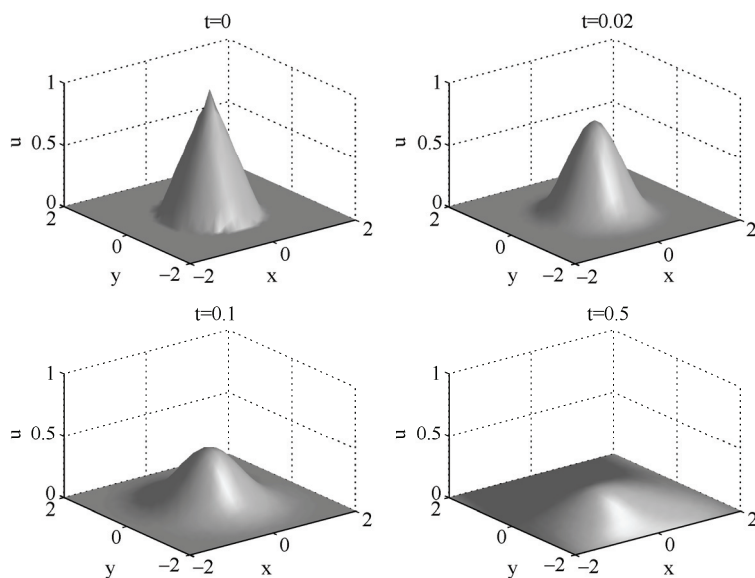


图 5-16 二维热传导方程的计算结果

可将其等价写为：

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -1 < x, y < 1 \\ \frac{\partial u}{\partial t} \Big|_{|x|=1} = \frac{\partial u}{\partial t} \Big|_{|y|=1} = 0 \\ u|_{t=0} = 1 - \sqrt{(x-1)^2 + y^2}, & (x-1)^2 + y^2 \leq 1 \\ u|_{t=0} = 0, & (x-1)^2 + y^2 > 1 \end{cases} \quad (5-49)$$

这种形式可直接用 ode45 函数计算，具体代码如下：

程序 5-14

主程序代码如下：

```
clear all; close all;
L=2; N=20;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
D2=D^2; x=L/2*x; y=x;
[X,Y]=meshgrid(x,y);
%初始条件
u=max(0,1-sqrt((X-1).^2+Y.^2));
%求解
t=[0 0.001 0.02 0.5];
```

```

[t,usol]=ode45('heat2',t,u(:),[],N,D2);
%画图
for n=1:4
    subplot(2,2,n)
    surf(x,y,reshape(usol(n,:),N+1,N+1))
    axis([-1 1 -1 1 0 1]), shading interp
    xlabel x, ylabel y, zlabel u
    title(['t=' num2str(t(n))]);
end

```

文件 heat2.m 代码如下：

```

function du=heat2(t,u,dummy,N,D2)
u=reshape(u,N+1,N+1);
du=D2*u+u*D2';
du([1 N+1],:)=0; du(:,[1 N+1])=0;
du=du(:);

```

程序输出结果如图 5-17 所示，从 $t=0$ 时刻开始，温度分布曲面逐渐趋于光滑，从 $x=1$ 边界进入的热量被另外三个边界吸收，稳定之后就成为了右下图的形态。

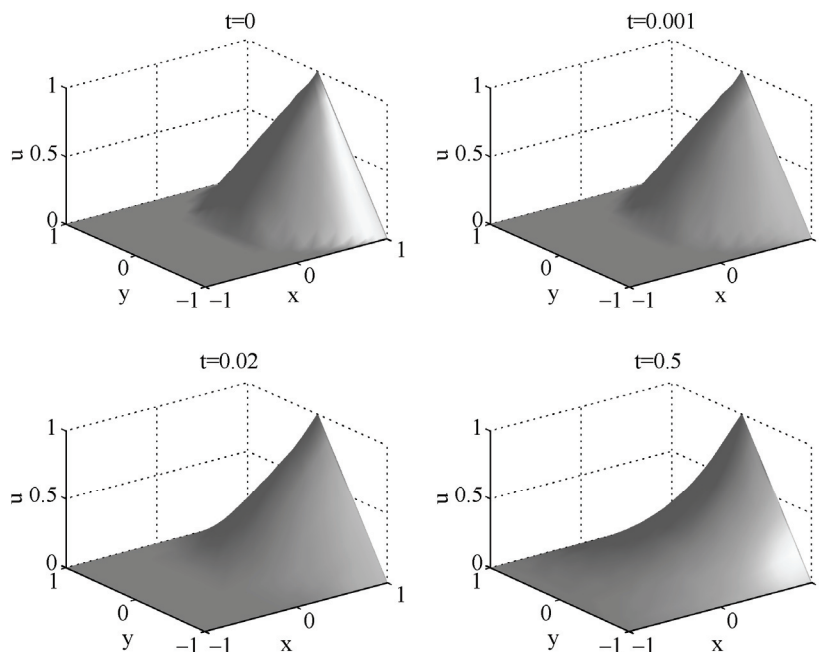


图 5-17 二维热传导方程的计算结果

5.2.5 一维特征值问题

本小节分析一维无限深方势阱中粒子的波函数分布问题。宽为 L 的无限深方势阱表达式为：

$$V(x) = \begin{cases} 0, & |x| < L/2 \\ \infty, & |x| > L/2 \end{cases} \quad (5-50)$$

由于势垒无限高，粒子不能穿过阱壁，所以阱外和阱壁处的波函数必为 0。在势阱内，采用自然单位的一维定态薛定谔方程为（取 $L=2$ ）：

$$u'' = \lambda u, \quad -1 < x < 1, \quad u(\pm 1) = 0 \quad (5-51)$$

其中， u 代表波函数， λ 代表能量特征值。上式的精确解为： $\lambda_n = -\pi^2 n^2/4$ ， $u_n = \sin[n\pi(x+1)/2]$ ， $n=1, 2, 3, \dots$ 。

用 $N+1$ 维向量 $\mathbf{u}$ 代表切比雪夫点上的波函数取值，删去其首尾元素得到 $N-1$ 维向量 $\tilde{\mathbf{u}}$ 。则式（5-51）的矩阵形式为：

$$\tilde{\mathbf{D}}_N^2 \tilde{\mathbf{u}}_{(N-1) \times 1} = \lambda \tilde{\mathbf{u}}_{(N-1) \times 1} \quad (5-52)$$

调用 Matlab 的 eig 函数求解矩阵的特征值，代码如下：

程序 5-15

```
clear all; close all;
L=2; N=16;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
D2=D2(2:N,2:N); x=L/2*x;
%求解
[V,E]=eig(D2); E=diag(E);
[eigenvalues,i]=sort(E,'descend');
V=[zeros(1,N-1);V(:,i);zeros(1,N-1)];
%画图
x2=-1:0.01:1;
for m=1:12
    subplot(4,3,m)
    plot(x2,polyval(polyfit(x,V(:,m),N),x2),'k', ...
        x,V(:,m),'r','MarkerSize',15,'LineWidth',1.5)
    n=sqrt(-eigenvalues(m)*4/pi^2);
    title(['-(' num2str(n) '\pi)^2/4']);
```

end

这里 $N=16$ ，所以在每个特征函数上只获得了 17 个数据点。当特征函数的变化比较剧烈时，这些有限的数点就不足以描绘出特征函数的本来模样。但实际上这些数据点包含了更多的信息，因为本章使用代数多项式作为插值函数，所以利用 `polyval`、`polyfit` 函数（多项式曲线拟合的 Matlab 函数）可以精确地获取插值函数在其他位置的值，以便画出准确的特征函数图像。如图 5-18 所示，点为计算结果，曲线为多项式曲线拟合的结果，每幅图上上面都标有计算得到的特征值。

需要强调的是，在特征函数的每个周期内，至少有 2 个数据点才能保证其正确性。比如，对于同样的 N ，切比雪夫点在原点的密度为等间距点在原点密度的 $2/\pi$ ，在 d 维空间中，这一比例变为 $(2/\pi)^d$ 。根据式 (5-51) 的精确解表达式，特征函数的周期为 $4/n$ 。所以，在原点处，若特征函数每周期内的数据点多于 2 个，有： $2/\pi \cdot N/2 \cdot 4/n > 2$ ，可化为： $n < 2N/\pi \approx 10$ 。这即是说，当 N 取 16 时，计算结果中 $n \geq 10$ 的特征值和特征函数都是不足为信的，这与图 5-18 一致。与精确解相比，第 7、8、9 个特征值虽有误差，但还比较小，第 10、11、12 个特征值就很不可信了，特征函数亦如此。

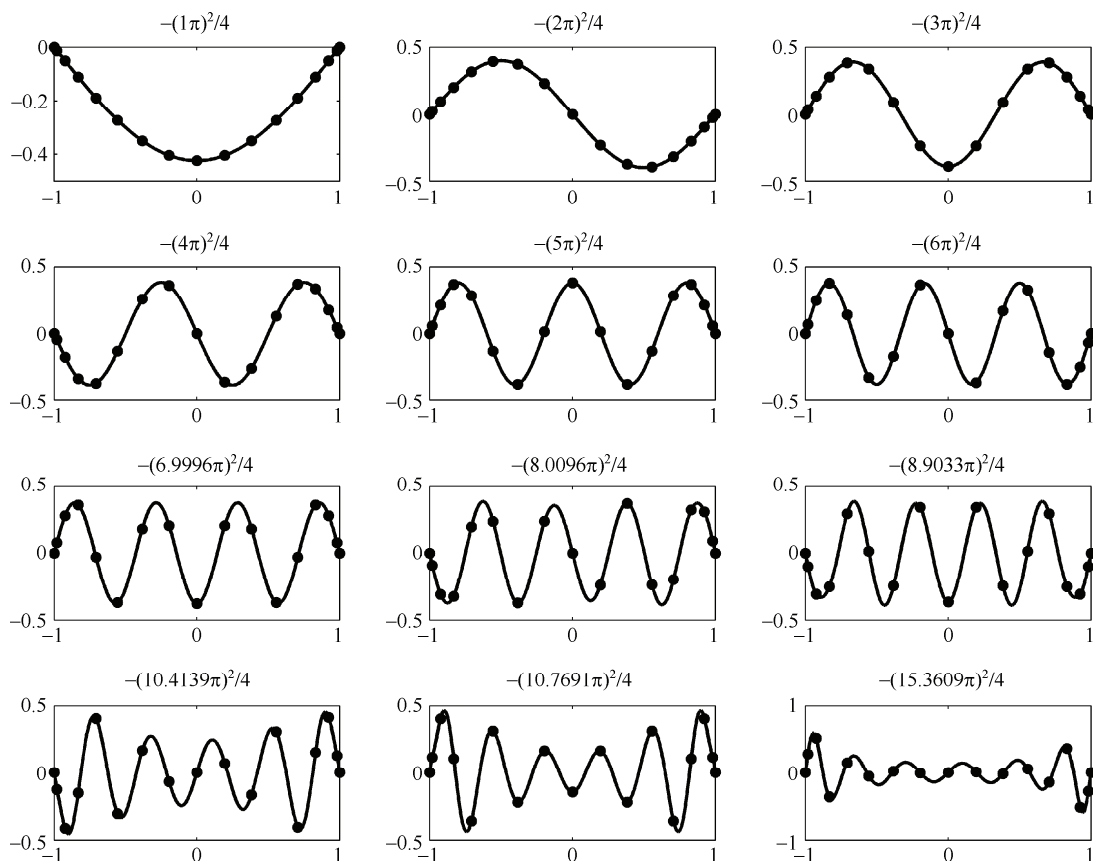


图 5-18 一维无限深方势阱中粒子波函数的特征值和特征函数

5.2.6 二维特征值问题

将式(5-51)拓展到二维空间,二维无限深方势阱中的定态薛定谔方程为(采用自然单位):

$$\begin{cases} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u = \lambda u, & -1 < x, y < 1 \\ u|_{x=\pm 1} = u|_{y=\pm 1} = 0 \end{cases} \quad (5-53)$$

它的特征值和特征函数的精确解为 ($n_1, n_2=1, 2, 3, \dots$):

$$\begin{aligned} \lambda_{n_1, n_2} &= -\frac{\pi^2}{4} (n_1^2 + n_2^2) \\ u_{n_1, n_2} &= \sin\left(\frac{n_1\pi}{2}(x+1)\right) \sin\left(\frac{n_2\pi}{2}(y+1)\right) \end{aligned} \quad (5-54)$$

根据边界条件,式(5-53)的矩阵形式可写为:

$$\tilde{\mathbf{L}} \tilde{\mathbf{u}}_{(N-1)^2 \times 1} = \lambda \tilde{\mathbf{u}}_{(N-1)^2 \times 1} \quad (5-55)$$

其中, $(N-1)^2$ 维向量 $\tilde{\mathbf{u}}$ 由式(5-31)定义,矩阵形式的拉普拉斯算符由式(5-34)定义。计算该矩阵的特征值和特征函数,代码如下:

程序 5-16

```
clear all; close all;
L=2; N=20;
%构造拉普拉斯算符矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
D2=D2(2:N,2:N); x=L/2*x; y=x;
I=eye(N-1); LA=kron(I,D2)+kron(D2,I);
%求解
[V,E]=eig(LA); E=diag(E);
[eigenvalues,i]=sort(real(E),'descend'); V=V(:,i);
%画图
x2=-L/2:0.08:L/2; y2=x2;
[py,px]=meshgrid(0.75:-0.25:0,0:0.25:0.75);
for n=1:16
    subplot('position',[px(n)+0.04 py(n)+0.01 0.17 0.22]);
    v=zeros(N+1); v(2:N,2:N)=reshape(V(:,n),N-1,N-1);
    v2=interp2(x,y,v,x2,y2,'cubic');
```

```

mesh(x2,y2,v2), hold on, view(35,20), axis off
axis([-1.1 1.1 -1.1 1.1 -0.2 0.13]);
contour3(x2,y2,v2-0.2,[-0.2-eps -0.2+eps])
m=-eigenvalues(n)*4/pi^2;
text(0,-2.4,0.3,['-' num2str(m) '\pi^2/4']);
end

```

前 16 个计算结果如图 5-19 所示：每个特征函数下面都有对应于 $u=0$ 的等高线图，每幅小图的左上角都标有相应的特征值，这些计算结果与精确解一致。

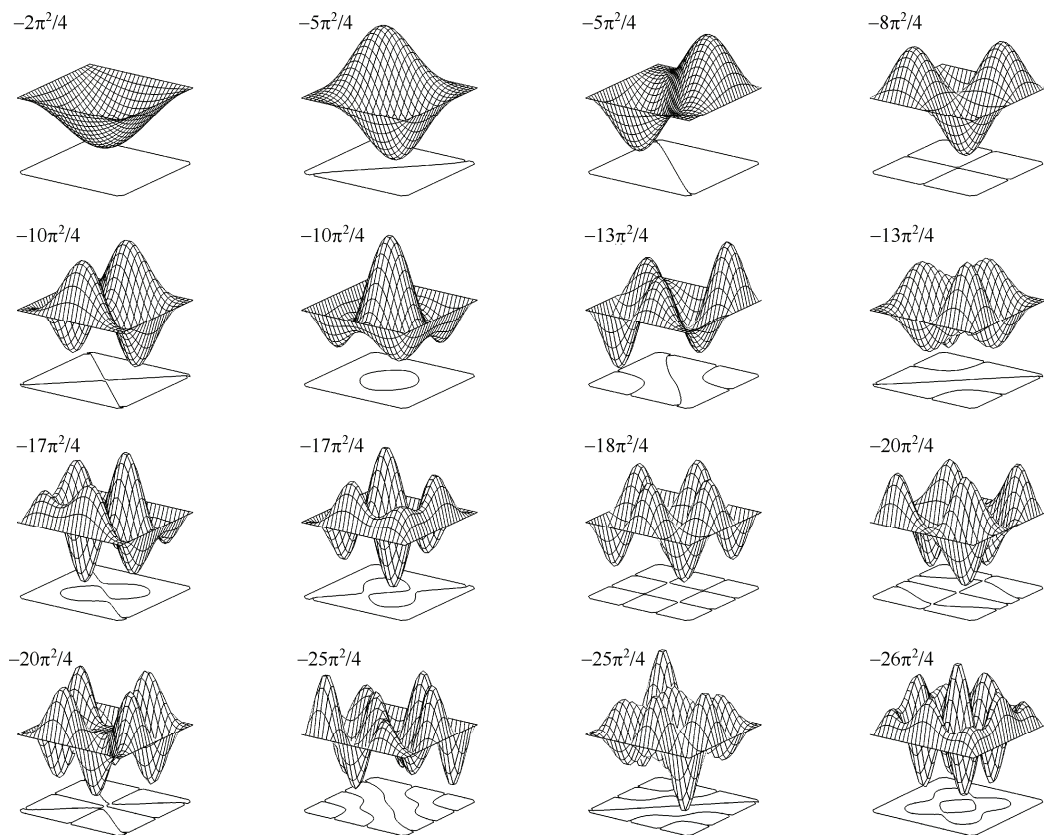


图 5-19 二维无限深方势阱中粒子波函数的特征值和特征函数

5.3 诺依曼边界条件（第二类边界条件）

诺依曼边界条件 (Neumann boundary condition)，又称为第二类边界条件，它给出了未知函数在边界上的法线方向的导数值。即： $\partial u / \partial n|_{\partial \Omega} = g$ ，其中， $\partial \Omega$ 为计算区域 Ω 的边界， n 为边界的法向， g 为已知函数。

5.3.1 一维泊松方程

考虑有如下边界条件的一维泊松方程：

$$u'' = f(x), \quad -1 < x < 1, \quad u'(1) = u(-1) = 0 \quad (5-56)$$

将 x 和 u 分别离散化为 $N+1$ 维向量 $\mathbf{x}=(x_0, x_1, \dots, x_N)^T$ 和 $\mathbf{u}=(u_0, u_1, \dots, u_N)^T$ 。注意 $x_N=-1$ 、 $x_0=1$ ，因为切比雪夫点的坐标是由大到小排列的。针对在左边界的狄利克莱边界条件 $u_N=0$ ，可用图 5-6 所示的方法修改矩阵 $\mathbf{D}_N^2$ 。针对在右边界的诺依曼边界条件，可将矩阵 $\mathbf{D}_N^2$ 的第一行替换为 $\mathbf{D}_N$ 的第一行，如图 5-20 所示。修改后的矩阵 $\mathbf{D}_N^2$ 与向量 $\mathbf{u}$ 相乘所得向量的第一个元素将是 $u'(x_0)$ 。

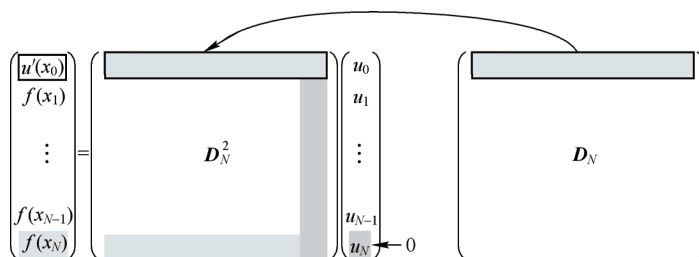


图 5-20 在狄利克莱边界条件 $u(-1)=0$ 和诺依曼边界条件 $u'(1)=0$ 下修改切比雪夫求导矩阵

在 $\mathbf{D}_N^2$ 上加 “-” 代表经过上述修改的切比雪夫求导矩阵，在 $\mathbf{f}$ 上加 “-” 代表删去向量 $\mathbf{f}=(f(x_0), f(x_1), \dots, f(x_N))^T$ 的最后一个元素（狄利克莱边界条件）并将其第一个元素替换为函数在右边界的导数值 $u'(x_0)=u'(1)=0$ （诺依曼边界条件）。求解式 (5-56)，只需：

$$\bar{\mathbf{u}}_{N \times 1} = (\bar{\mathbf{D}}_N^2)^{-1} \bar{\mathbf{f}}_{N \times 1} \quad (5-57)$$

最后在 N 维向量 $\bar{\mathbf{u}}$ 的末尾补 0。

在切比雪夫求导矩阵上乘以缩放因子可将上述方法推广至任意区间，以下面的一维泊松方程为例：

$$u'' = e^x, \quad -2 < x < 2, \quad u'(2) = u(-2) = 0 \quad (5-58)$$

其精确解为： $u=e^x-e^2x-e^{-2}-2e^2$ 。计算数值解并与精确解比较的代码如下：

程序 5-17

```
clear all; close all;
L=4; N=17;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2); x=L/2*x;
D2=D^2; D2(1,:)=D(1,:);
```

```

D2=D2(1:N,1:N);
%求解
f=exp(x(1:N)); f(1)=0;
u=D2\f; u=[u;0];
exact=exp(x)-exp(2)*x-exp(-2)-2*exp(2);
error=abs(exact-u);
%画图
subplot(2,2,1)
plot(x,exact,'k',x,u,'r','MarkerSize',16,'LineWidth',1.5)
title(['Error_{max}= ' num2str(max(error))]), xlabel x, ylabel u
subplot(2,2,2)
plot(x,error,'r','MarkerSize',16)
xlabel x, ylabel Error

```

程序输出结果如图 5-21 所示，数值结果与精确解基本吻合，最大误差在 10^{-13} 数量级。

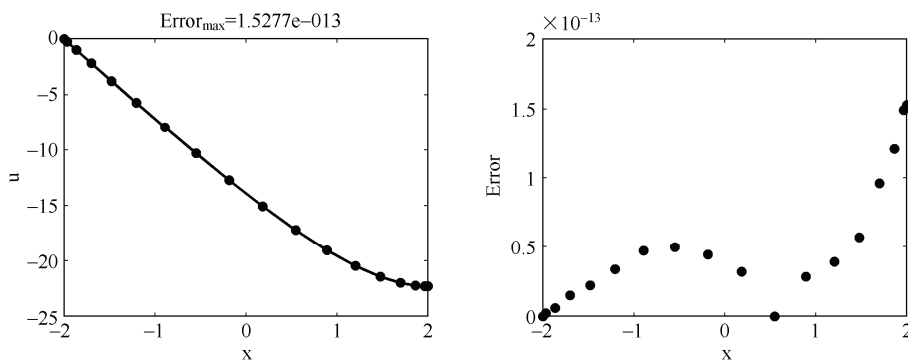


图 5-21 左图：一维泊松方程的数值解（点）和精确解（曲线），右图：误差分布

5.3.2 二维泊松方程

考虑如下二维泊松方程：

$$\begin{cases} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u = -\sin \left[(x+2)^2 (y+1) \right], & -2 < x, y < 2 \\ \frac{\partial u}{\partial x} \Big|_{x=2} = \frac{\partial u}{\partial y} \Big|_{y=2} = u \Big|_{y=-2} = 0 \\ u \Big|_{x=-2} = \frac{\sin(\pi y)}{10} \end{cases} \quad (5-59)$$

将函数 $u(x, y)$ 离散化为 $(N+1)^2$ 维向量 $\mathbf{u}$:

$$\mathbf{u}_{(N+1)^2 \times 1} = (u_{00}, u_{10}, \dots, u_{N0}, u_{01}, u_{11}, \dots, u_{N1}, \dots, u_{NN})^T \quad (5-60)$$

针对在 $x=-2$ 边界和 $y=-2$ 边界处的狄利克莱边界条件, 可用图 5-12 所示的方法修改拉普拉斯算符矩阵 $\mathbf{L}$ (式 (5-61)) 中对应于这两个边界的行。

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \rightarrow \mathbf{L} = \mathbf{D}_N^2 \otimes \mathbf{I}_{N+1} + \mathbf{I}_{N+1} \otimes \mathbf{D}_N^2 \quad (5-61)$$

针对在 $x=2$ 边界和 $y=2$ 边界处的诺依曼边界条件, 需要将 5.3.1 小节的方法推广到二维空间。先将 $\partial/\partial x$ 、 $\partial/\partial y$ 写为矩阵形式 $\mathbf{H}_x$ 、 $\mathbf{H}_y$:

$$\frac{\partial}{\partial x} \rightarrow \mathbf{H}_x = \mathbf{D}_N \otimes \mathbf{I}_{N+1} \quad (5-62)$$

$$\frac{\partial}{\partial y} \rightarrow \mathbf{H}_y = \mathbf{I}_{N+1} \otimes \mathbf{D}_N \quad (5-63)$$

并把 $\mathbf{H}_x$ 中对应于 $x=2$ 边界的行覆盖到拉普拉斯算符矩阵 $\mathbf{L}$ 相应的位置, 类似地, 把 $\mathbf{H}_y$ 中对应于 $y=2$ 边界的行覆盖到拉普拉斯算符矩阵 $\mathbf{L}$ 相应的位置。在拉普拉斯算符矩阵 $\mathbf{L}$ 上加 “-” 代表其经过了上述修改, 则对于下式:

$$\mathbf{f}_{(N+1)^2 \times 1} = \bar{\mathbf{L}} \mathbf{u}_{(N+1)^2 \times 1} \quad (5-64)$$

可知向量 $\mathbf{f}$ 中对应于 $x=-2$ 边界和 $y=-2$ 边界的元素, 与向量 $\mathbf{u}$ 相应位置的元素相等, 向量 $\mathbf{f}$ 中对应于 $x=2$ 边界和 $y=2$ 边界的元素, 分别与向量 $\mathbf{H}_x \mathbf{u}$ 和 $\mathbf{H}_y \mathbf{u}$ 在该位置的元素相等。这样, 求解式 (5-59), 只需:

$$\mathbf{u}_{(N+1)^2 \times 1} = \bar{\mathbf{L}}^{-1} \bar{\mathbf{f}}_{(N+1)^2 \times 1} \quad (5-65)$$

其中, 向量 $\mathbf{f}$ 为离散化的 $f(x, y) = -\sin[(x+2)^2(y+1)]$, 它上面的 “-” 代表对其做如下修改: 对应于 $x=-2$ 边界和 $y=-2$ 边界的元素分别取 $\sin(\pi y)/10$ 和 0 (狄利克莱边界条件), 对应于 $x=2$ 边界和 $y=2$ 边界的元素取 0 (诺依曼边界条件)。

具体代码如下:

程序 5-18

```
clear all; close all;
L=4; N=50;
%构造拉普拉斯算符矩阵
[D,x]=cheb(N); x=L/2*x; y=x;
[X,Y]=meshgrid(x,y);
X=X(:); Y=Y(:); D=D/(L/2); D2=D^2;
I=eye(N+1); LA=kron(I,D2)+kron(D2,I);
```

```

%修改拉普拉斯算符矩阵
Hx=kron(D,I); Hy=kron(I,D);
bound1=find(X==-L/2|Y==-L/2);
bound2=find(X==L/2|Y==L/2);
LA(bound1,:)=0; LA(bound1,bound1)=eye(2*N+1);
LA(bound2,:)=repmat(X(bound2)==L/2,1,(N+1)^2).*Hx(bound2,:) ...
    +repmat(Y(bound2)==L/2,1,(N+1)^2).*Hy(bound2,:);
%边界条件
f=-sin((X+2).^2.*(Y+1));
f([bound1;bound2])=0;
f(bound1)=(X(bound1)==-L/2).*sin(pi*Y(bound1))/10;
%求解
u=LA\f; u=reshape(u,N+1,N+1);
%画图
x2=-L/2:0.05:L/2; y2=x2;
u2=interp2(x,y,u,x2,y2,'cubic');
mesh(x2,y2,u2), view(-25,45)
xlabel x, ylabel y, zlabel u
axis([-2 2 -2 2 -0.1 0.3])

```

程序运行结果如图 5-22 所示。

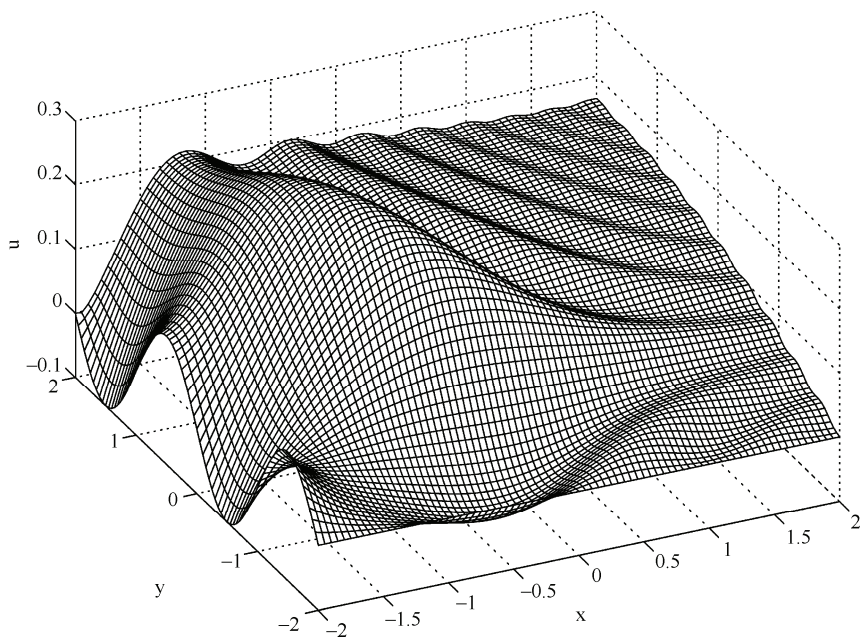


图 5-22 二维泊松方程的解

5.3.3 一维热传导方程

设函数 $u(x)$ 在切比雪夫点 $\mathbf{x}=(x_0, x_1, \dots, x_N)^T$ 处的取值为向量 $\mathbf{u}=(u_0, u_1, \dots, u_N)^T$ 。如果诺依曼边界条件仅约束了 $u'(1)$ 的值 ($x=-1$ 处为其他边界条件, 如狄利克莱边界条件), 有:

$$u'(1) = \begin{pmatrix} (\mathbf{D}_N)_{00} & (\mathbf{D}_N)_{01} & \cdots & (\mathbf{D}_N)_{0N} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = (\mathbf{D}_N)_{00} u_0 + \begin{pmatrix} (\mathbf{D}_N)_{01} & (\mathbf{D}_N)_{02} & \cdots & (\mathbf{D}_N)_{0N} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} \quad (5-66)$$

在 $u'(1)=0$ 的边界条件下, $u_0=u(1)$ 与 $(u_1, u_2, \dots, u_N)^T$ 的关系为:

$$u_0 = -\frac{1}{(\mathbf{D}_N)_{00}} \begin{pmatrix} (\mathbf{D}_N)_{01} & (\mathbf{D}_N)_{02} & \cdots & (\mathbf{D}_N)_{0N} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} \quad (5-67)$$

类似地, 如果诺依曼边界条件同时约束了 $u'(\pm 1)$ 的值, 有:

$$\begin{pmatrix} u'(1) \\ u'(-1) \end{pmatrix} = \begin{pmatrix} (\mathbf{D}_N)_{00} & (\mathbf{D}_N)_{01} & \cdots & (\mathbf{D}_N)_{0N} \\ (\mathbf{D}_N)_{N0} & (\mathbf{D}_N)_{N1} & \cdots & (\mathbf{D}_N)_{NN} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} \quad (5-68)$$

$$= \begin{pmatrix} (\mathbf{D}_N)_{00} & (\mathbf{D}_N)_{0N} \\ (\mathbf{D}_N)_{N0} & (\mathbf{D}_N)_{NN} \end{pmatrix} \begin{pmatrix} u_0 \\ u_N \end{pmatrix} + \begin{pmatrix} (\mathbf{D}_N)_{01} & (\mathbf{D}_N)_{02} & \cdots & (\mathbf{D}_N)_{0(N-1)} \\ (\mathbf{D}_N)_{N1} & (\mathbf{D}_N)_{N2} & \cdots & (\mathbf{D}_N)_{N(N-1)} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix}$$

在 $u'(\pm 1)=0$ 条件下, $(u_0, u_N)^T$ 与 $(u_1, u_2, \dots, u_{N-1})^T$ 关系为:

$$\begin{pmatrix} u_0 \\ u_N \end{pmatrix} = -\begin{pmatrix} (\mathbf{D}_N)_{00} & (\mathbf{D}_N)_{0N} \\ (\mathbf{D}_N)_{N0} & (\mathbf{D}_N)_{NN} \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{D}_N)_{01} & (\mathbf{D}_N)_{02} & \cdots & (\mathbf{D}_N)_{0(N-1)} \\ (\mathbf{D}_N)_{N1} & (\mathbf{D}_N)_{N2} & \cdots & (\mathbf{D}_N)_{N(N-1)} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} \quad (5-69)$$

其中, $u_0=u(1)$, $u_N=u(-1)$ 。

式 (5-67) 和式 (5-69) 的意义在于: 将边界条件 $u'(1)=0$ 或 $u'(\pm 1)=0$ 转化为对 $u(1)$ 或 $u(\pm 1)$ 的约束条件, 这样就可以直接处理单边界或双边界上的诺依曼边界条件了。以一维热传导方程为例:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & -1 < x < 1 \\ \frac{\partial u}{\partial x} \Big|_{x=\pm 1} = 0 \\ u \Big|_{t=0} = 1 + \cos(\pi x) \end{cases} \quad (5-70)$$

上式描述了均匀细杆上的热传导过程， u 代表温度分布， t 代表时间， x 代表杆上的坐标，诺依曼边界条件 $\partial u / \partial x|_{x=\pm 1} = 0$ 代表边界是绝热的。可以预料，若时间 t 足够长，杆上的温度将处处相等。

容易知道，边界条件可等价写为：

$$\frac{\partial u}{\partial x} \Big|_{x=\pm 1} = 0 \Leftrightarrow \begin{cases} \frac{\partial}{\partial t} \left(\frac{\partial u}{\partial x} \right) \Big|_{x=\pm 1} = 0 \\ \frac{\partial u}{\partial x} \Big|_{x=\pm 1, t=0} = 0 \end{cases} \quad (5-71)$$

u 在边界处的二阶混合偏导数 $\partial(\partial u / \partial x) / \partial t|_{x=\pm 1}$ 显然是连续的，且由物理意义可知 $\partial u / \partial t|_{x=\pm 1}$ 也是连续的，所以 u 的二阶混合偏导数的求导次序可以在边界处交换（证明从略）：

$$\frac{\partial}{\partial t} \left(\frac{\partial u}{\partial x} \right) \Big|_{x=\pm 1} = \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t} \right) \Big|_{x=\pm 1} = 0 \quad (5-72)$$

那么，式（5-70）可等价写为式（5-73）。初始条件已经包含 $\partial u / \partial x|_{x=\pm 1, t=0} = 0$ ，这里无需重复。

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & -1 < x < 1 \\ \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t} \right) \Big|_{x=\pm 1} = 0 \\ u \Big|_{t=0} = 1 + \cos(\pi x) \end{cases} \quad (5-73)$$

上式中的 $\partial / \partial t$ 可用 `ode45` 函数计算， $\partial^2 / \partial x^2$ 用矩阵 $\mathbf{D}_N^2$ 计算，根据式（5-69）处理诺依曼边界条件（处理时将 $\partial u / \partial t$ 看做 u ），代码如下：

程序 5-19

主程序代码如下：

```
clear all; close all;
L=2; N=20;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
```

```

D2=D^2; x=L/2*x;
%初始条件
u=1+cos(pi*x);
%诺依曼边界条件
BC=-D([1 N+1],[1 N+1])\D([1 N+1],2:N);
%求解
t=0:0.01:0.4;
[t,usol]=ode45('heat1D',t,u,[],N,D2,BC);
%画图
X=L/2:-0.05:-L/2;
u=interp2(x,t,usol,X,t,'cubic');
waterfall(X,t,u)
xlabel x, ylabel t, zlabel u
axis([-1 1 0 0.4 0 2])

```

文件 heat1D.m 代码如下:

```

function du=heat1D(t,u,dummy,N,D2,BC)
du=D2*u;
du([1 N+1])=BC*du(2:N);

```

输出结果如图 5-23 所示: $t=0$ 时刻杆上的温度分布极不均匀, 随着 t 的增加, 热量从高温部分扩散到低温部分, 杆上各处温度趋于某一定值。由于两端绝热 ($\partial u / \partial x|_{x=\pm 1}=0$), u 对 x 在 $[-1, 1]$ 上的积分结果不随 t 变化。

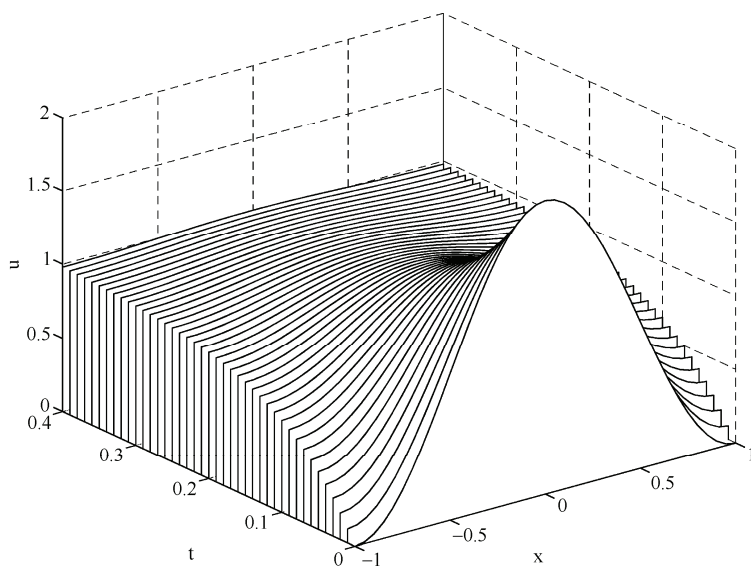


图 5-23 两端绝热条件下一维热传导方程的解

5.3.4 二维波动方程

考虑如下的二维波动方程：

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -3 < x < 3, -1 < y < 1 \\ \frac{\partial u}{\partial y} \Big|_{|y|=1} = 0, & u \Big|_{x=-3} = u \Big|_{x=3} \\ u \Big|_{t=0} = e^{-8[(x+1.5)^2 + y^2]}, & \frac{\partial u}{\partial t} \Big|_{t=0} = -\frac{\partial u}{\partial x} \Big|_{t=0} \end{cases} \quad (5-74)$$

可认为上式描述了水的波动过程， $u(x, y, t)$ 是水面高度的分布， x, y 是空间坐标， t 是时间。在 $|x|=3$ 边界处采用了周期性边界条件，这代表从 $x=3$ 边界流出的水将从 $x=-3$ 边界流入，反之亦然。在 $|y|=1$ 边界处采用了诺依曼边界条件， $\partial u / \partial y|_{|y|=1} = 0$ 代表水不能从 $|y|=1$ 处流入或流出。

由初始条件可知 $\partial u / \partial y|_{|y|=1, t=0} \approx 0$ ，因此可将边界条件 $\partial u / \partial y|_{|y|=1} = 0$ 等价地写为 $\partial(\partial u / \partial y) / \partial t|_{|y|=1} = 0$ 。又由它的连续性，所以可交换求导顺序，等价写为 $\partial(\partial u / \partial t) / \partial y|_{|y|=1} = 0$ 。然后，引入函数 $v(x, y, t) = \partial u(x, y, t) / \partial t$ ，将式 (5-74) 中的 $\partial^2 / \partial t^2$ 降为 $\partial / \partial t$ ，得到：

$$\begin{cases} \frac{\partial u}{\partial t} = v, & -3 < x < 3, -1 < y < 1 \\ \frac{\partial v}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u \\ \frac{\partial v}{\partial y} \Big|_{|y|=1} = 0, & u \Big|_{x=-3} = u \Big|_{x=3} \\ u \Big|_{t=0} = e^{-8[(x+1.5)^2 + y^2]}, & v \Big|_{t=0} = -\frac{\partial u}{\partial x} \Big|_{t=0} \end{cases} \quad (5-75)$$

用 `ode45` 计算上式中的 $\partial / \partial t$ 。此外，由于 $|x|=3$ 处存在周期性边界条件，所以可使用谱求导矩阵计算 x 方向上的导数。对于 $|y|=1$ 边界处的诺依曼边界条件，这里使用切比雪夫求导矩阵计算 y 方向上的导数，并根据式 (5-69) 处理 $\partial v / \partial y|_{|y|=1} = 0$ 。代码如下：

程序 5-20

主程序代码如下：

```
clear all; close all;
%x 方向
Lx=6; Nx=50;
x=Lx/Nx*[-Nx/2:Nx/2-1]';
h=2*pi/Nx; column=[0 0.5*(-1).^(1:Nx-1).*cot((1:Nx-1)*h/2)]';
```

```

Dx=(2*pi/Lx)*toeplitz(column,column([1 Nx:-1:2]));
column=[-pi^2/(3*h^2)-1/6 -0.5*(-1).^(1:Nx-1)./sin(h*(1:Nx-1)/2).^2];
D2x=(2*pi/Lx)^2*toeplitz(column);
%y 方向
Ly=2; Ny=20; [Dy,y]=cheb(Ny);
Dy=Dy/(Ly/2); D2y=Dy^2; y=Ly/2*y;
%诺依曼边界条件
BCy=-Dy([1 Ny+1],[1 Ny+1])\Dy([1 Ny+1],2:Ny);
%初始条件
[X,Y]=meshgrid(x,y);
u=exp(-8*((X+1.5).^2+Y.^2));
v=-u*Dx';
uv=[u(:); v(:)];
%求解
t=0:2:4;
[t,uvsol]=ode45('wave_tank',t,uv,[],Nx,Ny,D2x,D2y,BCy);
%画图
for n=1:3
subplot(3,1,n)
u=reshape(uvsol(n,1:end/2),Ny+1,Nx);
mesh(x,y,u), view(-10,60), grid off
title(['t=' num2str(t(n))])
xlabel x, ylabel y, zlabel u
axis([-3 3 -1 1 -0.15 1])
end

```

文件 wave_tank.m 代码如下:

```

function duv=wave_tank(t,uv,dummy,Nx,Ny,D2x,D2y,BCy)
u=reshape(uv(1:end/2),Ny+1,Nx);
v=reshape(uv(end/2+1:end),Ny+1,Nx);
%诺依曼边界条件
v([1 Ny+1],:)=BCy*v(2:Ny,:);
duv=[v(:); reshape(u*D2x'+D2y*u,Nx*(Ny+1),1)];

```

计算结果如图 5-24 所示, 边界 $|y|=1$ 像“水池壁”一样将“水”挡住。方程式 (5-74) 来源于文献[2], 这里使用了比文献[2]更精确的方法, 有兴趣的读者可以比较一下两种方法的差异。

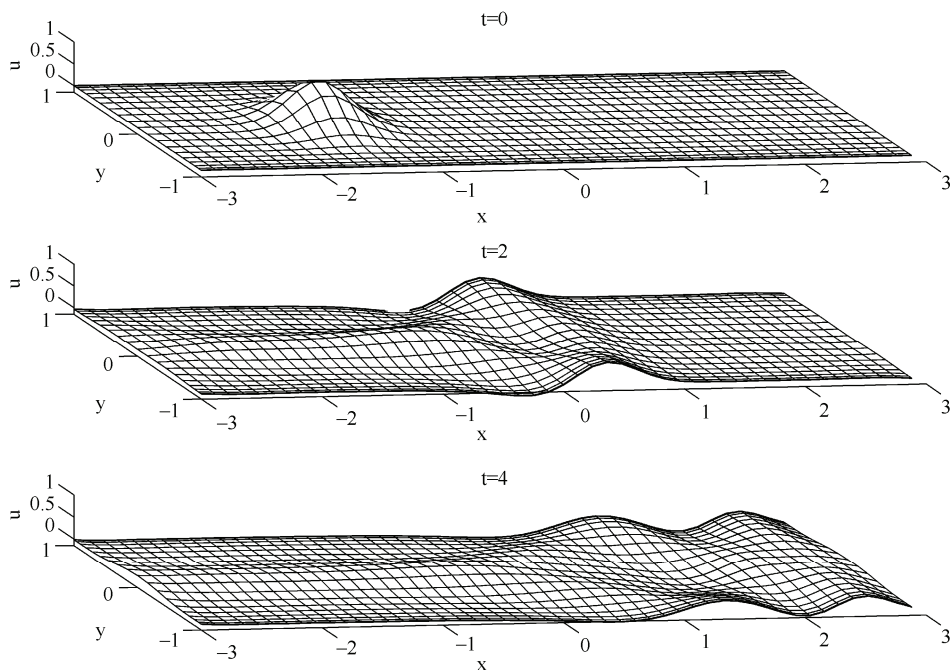


图 5-24 二维波动方程的解, $|x|=3$ 边界处采用了周期性边界条件, $|y|=1$ 边界处采用了诺依曼边界条件 $\partial u / \partial y|_{|y|=1} = 0$

5.3.5 一维四阶问题

一维双调和问题 (biharmonic problem) 的形式如下:

$$u'''' = f(x), \quad -1 < x < 1, \quad u(\pm 1) = u'(\pm 1) = 0 \quad (5-76)$$

求解上式的难点在于边界条件既规定了 $u(\pm 1) = 0$ 同时又要求 $u'(\pm 1) = 0$ 。为了简化这一复杂的边界条件, 将 $u(x)$ 的插值函数 $p(x)$ 写为:

$$p(x) = (1 - x^2)q(x), \quad -1 < x < 1 \quad (5-77)$$

其中, $q(x)$ 在边界处取值为 0, 即 $q(\pm 1) = 0$ 。容易知道, 上式定义的 $p(x)$ 必满足 $p(\pm 1) = p'(\pm 1) = 0$ 。这样就将插值函数 $p(x)$ 的复杂边界条件简化为 $p(x)$ 的狄利克莱边界条件。

对式 (5-77) 求 4 阶导数并代入 $q(x) = p(x)/(1 - x^2)$, 得:

$$\begin{aligned} p''''(x) &= (1 - x^2)q''''(x) - 8xq'''(x) - 12q''(x) \\ &= \left[(1 - x^2) \cdot d^4/dx^4 - 8x \cdot d^3/dx^3 - 12 \cdot d^2/dx^2 \right] \cdot q(x) \\ &= \left[(1 - x^2) \cdot d^4/dx^4 - 8x \cdot d^3/dx^3 - 12 \cdot d^2/dx^2 \right] / (1 - x^2) \cdot p(x) \end{aligned} \quad (5-78)$$

那么，对 $p(x)$ 求 4 阶导数的运算可写为矩阵形式：

$$d^4/dx^4 \rightarrow \tilde{L}_2 = [\text{diag}(1 - \tilde{x}^2) \tilde{D}_N^4 - 8\text{diag}(\tilde{x}) \tilde{D}_N^3 - 12\tilde{D}_N^2] \text{diag}(1/(1 - \tilde{x}^2)) \quad (5-79)$$

其中，函数 diag 表示将向量转化为对角矩阵，向量 $\mathbf{x}$ 代表 $(x_0, x_1, \dots, x_N)^T$ 。此外，向量 $1 - \mathbf{x}^2$ 代表 $(1 - x_0^2, 1 - x_1^2, \dots, 1 - x_N^2)^T$ ，向量 $1/(1 - \mathbf{x}^2)$ 代表 $(1/(1 - x_0^2), 1/(1 - x_1^2), \dots, 1/(1 - x_N^2))^T$ 。“ $\sim$ ”表示删除向量的首尾元素以及删除矩阵首尾行、首尾列。根据式 (5-79)，可将式 (5-76) 写为矩阵形式：

$$\tilde{L}_2 \tilde{\mathbf{u}} = f(\tilde{\mathbf{x}}) \quad (5-80)$$

求解上式只需求解：

$$\tilde{\mathbf{u}} = \tilde{L}_2^{-1} f(\tilde{\mathbf{x}}) \quad (5-81)$$

并在得到的向量 $\tilde{\mathbf{u}}$ 的首尾补 0。

以下面的双调和方程为例：

$$u'''' = \sin(\pi x), \quad -1 < x < 1, \quad u(\pm 1) = u'(\pm 1) = 0 \quad (5-82)$$

接下来求它的数值解，并与其精确解 $u = \sin(\pi x)/\pi^4 + (x^3 - x)/2\pi^3$ 比较。代码如下：

程序 5-21

```
clear all; close all;
L=2; N=24;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
D4=(diag(1-x.^2)*D^4-diag(8*x)*D^3-12*D^2)*diag([1./(1-x.^2)]);
D4=D4(2:N,2:N);
%求解
f=sin(pi*x(2:N)); u=[0;D4\f;0];
%画图
exact=sin(pi*x)/pi^4+(x.^3-x)/(2*pi^3);
error=abs(exact-u);
subplot(2,2,1)
plot(x,exact,'k',x,u,'r','MarkerSize',15,'LineWidth',1.5)
title(['Error_{max}= ' num2str(max(error))])
xlabel x, ylabel u
subplot(2,2,2)
plot(x,error,'r','MarkerSize',15)
xlabel x, ylabel Error
```

结果如图 5-25 所示， $N=24$ 时的最大误差在 10^{-15} 数量级。

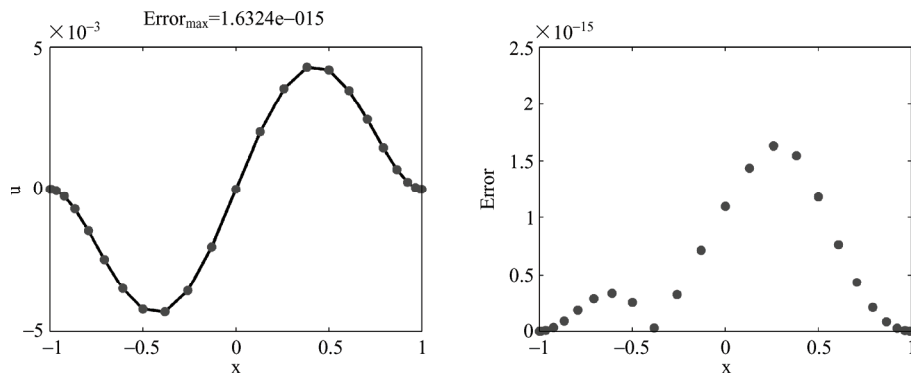


图 5-25 左图：双调和方程的数值解（点）和解析解（线）；右图：误差分布

类似地，若双调和问题（5-76）的计算区间为 $[-L/2, L/2]$ ，只需将式（5-77）中的 $1-x^2$ 改为 $L^2/4-x^2$ ，并对 D_N 做相应的缩放即可。

5.3.6 二维四阶问题

考虑二维双调和算符的特征值问题：

$$\begin{cases} \Delta^2 u = \lambda u, & -1 < x, y < 1 \\ u|_{|x|=1} = u|_{|y|=1} = 0 \\ \frac{\partial u}{\partial x}|_{|x|=1} = \frac{\partial u}{\partial y}|_{|y|=1} = 0 \end{cases} \quad (5-83)$$

其中，二维双调和算符 $\Delta^2 = (\partial_x^2 + \partial_y^2)^2 = \partial_x^4 + 2\partial_x^2 \partial_y^2 + \partial_y^4$ 。由于上式的边界条件与式（5-76）的边界条件一致，所以可直接将式（5-79）应用到二维双调和算符的矩阵形式中，有：

$$\Delta^2 \rightarrow \tilde{L}_2 \otimes I_{N-1} + 2(\tilde{D}_N^2 \otimes I_{N-1})(I_{N-1} \otimes \tilde{D}_N^2) + I_{N-1} \otimes \tilde{L}_2 \quad (5-84)$$

用 eig 函数求解该矩阵的特征值和特征函数的代码如下：

程序 5-22

```
clear all; close all;
L=2; N=25;
%二维双调和算符矩阵
[D,x]=cheb(N); x=L/2*x; y=x;
D=D/(L/2); D2=D^2; D2=D2(2:N,2:N);
D4=(diag(1-x.^2)*D^4-diag(8*x)*D^3-12*D^2)*diag([1./(1-x.^2)]);
D4=D4(2:N,2:N); I=eye(N-1);
LA2=kron(D4,I)+2*kron(D2,I)*kron(I,D2)+kron(I,D4);
```

```

%求解
[V,D]=eig(LA2); D=diag(real(D));
[eigenvalues,i]=sort(D); V=real(V(:,i));
%画图
x2=-1.08:0.08:1.08; y2=x2;
[py,px]=meshgrid(0.8:-0.2:0,0:0.2:0.8);
for n=1:25
    subplot('position',[px(n)+0.02 py(n)+0.01 0.15 0.2]);
    v=zeros(N+1,N+1); v(2:N,2:N)=reshape(V(:,n),N-1,N-1);
    v2=interp2(x,y,v,x2,y2,'cubic');
    mesh(x2,y2,v2), hold on, axis off
    contour3(x2,y2,v2-0.25,[-0.25-eps -0.25+eps])
    text(0.7,0,0.13,sprintf('%0.1f',eigenvalues(n)))
    axis([-1 1 -1 1 -0.25 0.1])
end

```

程序输出结果如图 5-26 所示, 每幅小图的右上角都标有特征值, 并给出了相应的特征函数的立体图, 其下方是 $u=0$ 的等高线图。

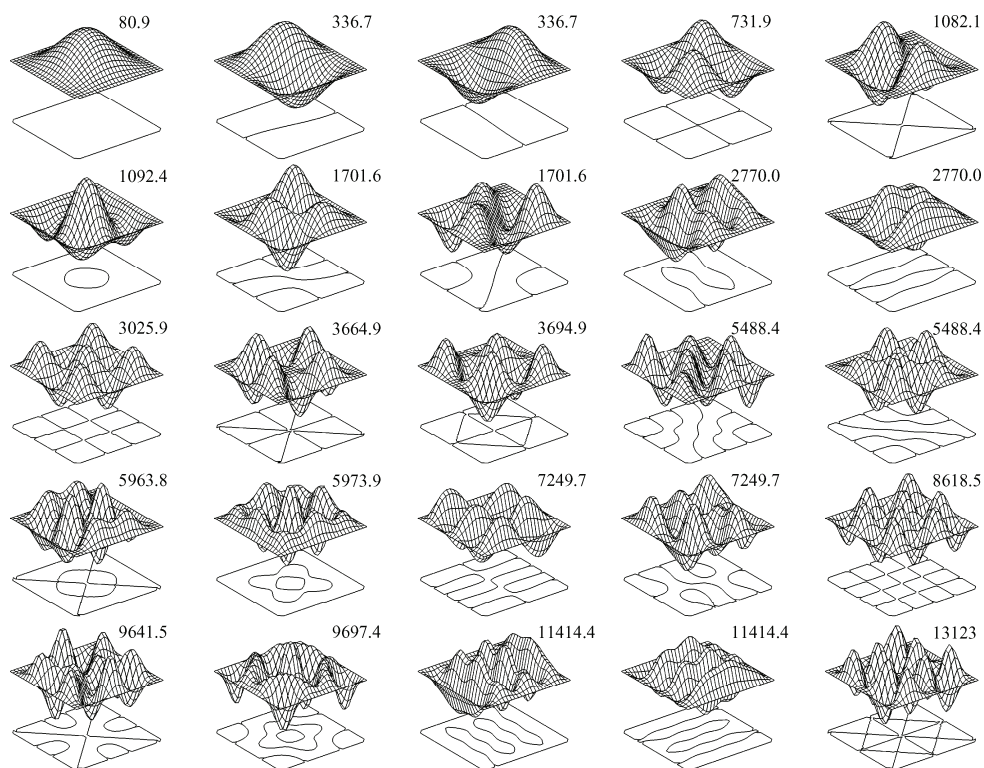


图 5-26 二维双调和算符的前 25 个特征值和特征函数

5.4 洛平边界条件（第三类边界条件）

洛平边界条件（Robin boundary condition），又称为第三类边界条件或混合边界条件，它给出了在边界处未知函数和它在法线方向上导数的线性组合的值。即： $(u+h\partial u/\partial n)|_{\partial\Omega}=g$ ，其中， $\partial\Omega$ 为计算区域 Ω 的边界， n 为边界的法向， g 为已知函数。

5.4.1 一维泊松方程

考虑具有洛平边界条件的一维泊松问题：

$$u'' = f(x), \quad -1 < x < 1, \quad u(\pm 1) + hu'(\pm 1) = g(\pm 1) \quad (5-85)$$

先将 x 和 u 分别离散化为 $N+1$ 维向量 $\mathbf{x}=(x_0, x_1, \dots, x_N)^T$ 和 $\mathbf{u}=(u_0, u_1, \dots, u_N)^T$ 。针对两端边界的洛平边界条件，需要综合图 5-8 及图 5-20 所示的方法修改矩阵 $\mathbf{D}_N^2$ 。如图 5-27 所示：取出矩阵 $h\mathbf{D}_N$ 的首（尾）行，在其首（尾）元素上加 1，然后替换到矩阵 $\mathbf{D}_N^2$ 的首（尾）行处。这样修改后的矩阵 $\mathbf{D}_N^2$ 与向量 $\mathbf{u}$ 相乘所得向量的首尾元素将是 $u_0+hu'(x_0)$ 和 $u_N+hu'(x_N)$ 。

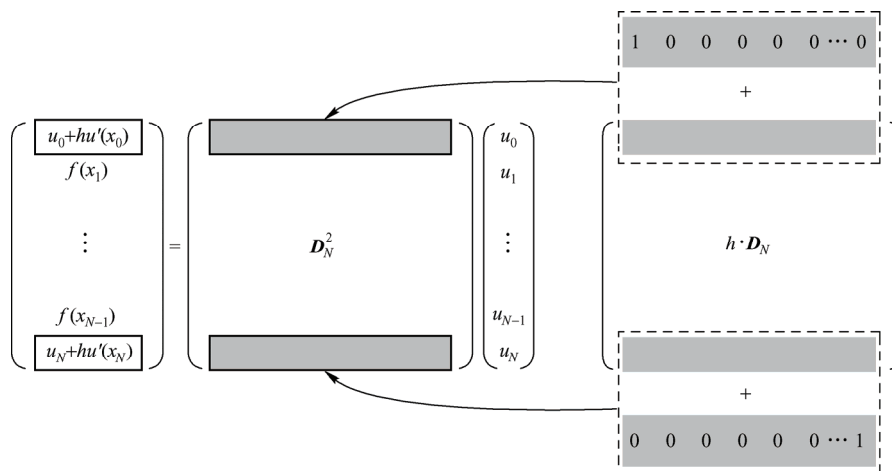


图 5-27 在洛平边界条件下修改切比雪夫求导矩阵

在 $\mathbf{D}_N^2$ 上加两个“-”代表经过上述修改的切比雪夫求导矩阵，在 $\mathbf{f}$ 上加两个“-”代表将向量 $\mathbf{f}=(f(x_0), f(x_1), \dots, f(x_N))^T$ 的首尾元素分别替换为 $u_0+hu'(x_0)$ 和 $u_N+hu'(x_N)$ 的值（洛平边界条件）。那么，求解式（5-85），只需：

$$\mathbf{u}_{N \times 1} = (\bar{\mathbf{D}}_N^2)^{-1} \bar{\mathbf{f}}_{N \times 1} \quad (5-86)$$

在切比雪夫求导矩阵上乘以缩放因子可将上述方法推广至任意区间，以下面的一维泊

松方程为例：

$$u'' = e^x, \quad -2 < x < 2, \quad (u + 2u')|_{|x|=2} = 1 \quad (5-87)$$

根据式 (5-86) 求解，并与其解析解 $u = e^x + 3(e^{-2} - e^2)x/4 - 3e^{-2} + 1$ 比较，代码如下：

程序 5-23

```
clear all; close all;
L=4; N=18; h=2;
%构造切比雪夫求导矩阵并修改
[D,x]=cheb(N); D=D/(L/2); x=L/2*x;
D2=D^2; I=eye(N+1);
D2([1 N+1],:)=h*D([1 N+1],:)+I([1 N+1],:);
%求解
f=exp(x(1:N+1)); f([1 N+1])=1; u=D2\f;
%画图
exact=exp(x)+3/4*(exp(-2)-exp(2))*x+1-3*exp(-2);
error=abs(exact-u);
subplot(2,2,1)
plot(x,exact,'k',x,u,'r','MarkerSize',16,'LineWidth',1.5)
title(['Error_{max}=' num2str(max(error))]), xlabel x, ylabel u
subplot(2,2,2)
plot(x,error,'r','MarkerSize',16)
xlabel x, ylabel Error
```

程序输出结果如图 5-28 所示：左图中，点表示计算得到的数值解，线表示解析解，二者吻合得很好。右图为误差分布， $N=18$ 时，最大误差仅在 10^{-13} 级。

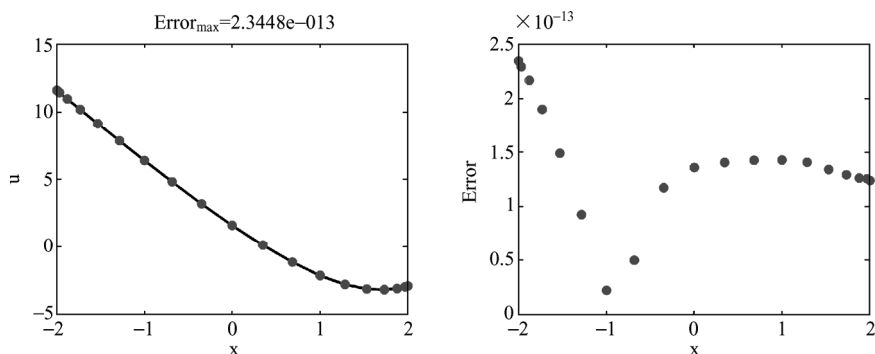


图 5-28 左图：洛平边界条件下一维泊松问题的数值解（点）和解析解（线），右图：误差分布

5.4.2 二维泊松方程

考虑如下二维泊松方程：

$$\begin{cases} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u = \sin[(x+5)(y+2)], & -2 < x, y < 2 \\ \left(u + \frac{1}{10} \frac{\partial u}{\partial x} \right) \Big|_{|x|=2} = \left(u + \frac{1}{10} \frac{\partial u}{\partial y} \right) \Big|_{|y|=2} = 1 \end{cases} \quad (5-88)$$

将函数 $u(x, y)$ 离散化为 $(N+1)^2$ 维向量 $\mathbf{u}$ ：

$$\mathbf{u}_{(N+1)^2 \times 1} = (u_{00}, u_{10}, \dots, u_{N0}, u_{01}, u_{11}, \dots, u_{N1}, \dots, u_{NN})^T \quad (5-89)$$

洛平边界条件的矩阵形式 $\mathbf{R}_x$ 和 $\mathbf{R}_y$ 如下，其中 $\mathbf{I}_N$ 为 N 阶单位矩阵：

$$u + h \frac{\partial u}{\partial x} \rightarrow \mathbf{R}_x \mathbf{u}_{(N+1)^2 \times 1} = \left[\mathbf{I}_{(N+1)^2} + h(\mathbf{D}_N \otimes \mathbf{I}_{N+1}) \right] \mathbf{u}_{(N+1)^2 \times 1} \quad (5-90)$$

$$u + h \frac{\partial u}{\partial y} \rightarrow \mathbf{R}_y \mathbf{u}_{(N+1)^2 \times 1} = \left[\mathbf{I}_{(N+1)^2} + h(\mathbf{I}_{N+1} \otimes \mathbf{D}_N) \right] \mathbf{u}_{(N+1)^2 \times 1} \quad (5-91)$$

在此需要把 5.4.1 小节的方法推广到二维情况。即：将矩阵 $\mathbf{R}_x$ 和 $\mathbf{R}_y$ 中分别对应于边界 $|x|=2$ 和边界 $|y|=2$ 的行替换到拉普拉斯算符矩阵（式（5-92））中相应的位置。

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \rightarrow \mathbf{L} = \mathbf{D}_N^2 \otimes \mathbf{I}_{N+1} + \mathbf{I}_{N+1} \otimes \mathbf{D}_N^2 \quad (5-92)$$

在 $\mathbf{L}$ 上面加 2 个 “-” 代表修改后的拉普拉斯算符矩阵，那么，求解式（5-88）只需：

$$\mathbf{u}_{(N+1)^2 \times 1} = \bar{\mathbf{L}}^{-1} \bar{\mathbf{f}}_{(N+1)^2 \times 1} \quad (5-93)$$

其中，向量 $\mathbf{f}$ 为离散化的 $f(x, y) = \sin[(x+5)(y+2)]$ ，它上面的 2 个 “-” 代表对其做如下修改：在边界 $|x|=2$ 和边界 $|y|=2$ 处的元素取 1（根据洛平边界条件）。

具体代码如下：

程序 5-24

```
clear all; close all;
L=4; N=40; h=0.1;
%构造拉普拉斯算符矩阵
[D,x]=cheb(N); x=L/2*x; y=x;
[X,Y]=meshgrid(x,y);
X=X(:); Y=Y(:); D=D/(L/2); D2=D^2;
I=eye(N+1); LA=kron(I,D2)+kron(D2,I);
```

```

%修改拉普拉斯算符矩阵
Hx=kron(D,I); Hy=kron(I,D);
bound1=find(X==L/2|X==-L/2);
bound2=find(Y==L/2|Y==-L/2);
I=eye((N+1)^2);
LA(bound1,:)=I(bound1,:)+h*Hx(bound1,:);
LA(bound2,:)=I(bound2,:)+h*Hy(bound2,:);
%求解
f=sin((X+5).*(Y+2));
f([bound1;bound2])=1;
u=LA\f; u=reshape(u,N+1,N+1);
%画图
x2=-L/2:0.05:L/2; y2=x2;
u2=interp2(x,y,u,x2,y2,'cubic');
mesh(x2,y2,u2), view(-25,45)
xlabel x, ylabel y, zlabel u
%边界误差
Ex=u(:)+h*Hx*u(:)-1; max(abs(Ex(bound1)))
Ey=u(:)+h*Hy*u(:)-1; max(abs(Ey(bound2)))

```

计算结果如图 5-29 所示。程序最后输出数值解在边界 $|x|=2$ 和边界 $|y|=2$ 处的最大误差分别为 1.3778×10^{-11} 和 1.5851×10^{-11} 。

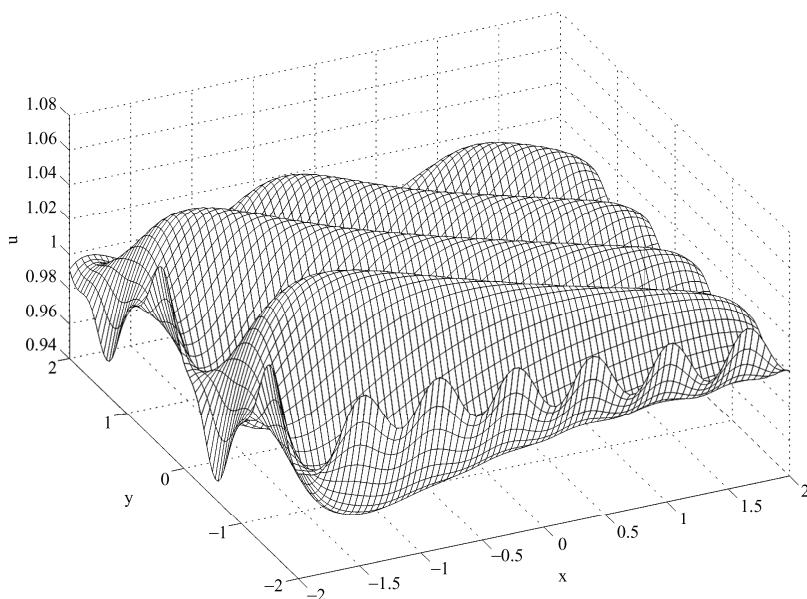


图 5-29 洛平边界条件下二维泊松方程的解

5.4.3 一维热传导方程

在分析一维热传导方程的解法前，先把洛平边界条件写为矩阵形式。这里以洛平边界条件 $(u \pm hu')|_{x=\pm 1}=0$ 为例，其中的正号对应右边界，负号对应左边界。那么可将其写为：

$$\begin{pmatrix} u(1) \\ u(-1) \end{pmatrix} + h \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} u'(1) \\ u'(-1) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (5-94)$$

设函数 $u(x)$ 在切比雪夫点 $\mathbf{x}=(x_0, x_1, \dots, x_N)^T$ 处的取值为向量 $\mathbf{u}=(u_0, u_1, \dots, u_N)^T$ 。则有：

$$\begin{pmatrix} u(1) \\ u(-1) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_N \end{pmatrix} \quad (5-95)$$

将式 (5-95) 和式 (5-68) 代入式 (5-94)，整理可得 $(u_0, u_N)^T$ 与 $(u_1, u_2, \dots, u_{N-1})^T$ 的关系：

$$\begin{pmatrix} u_0 \\ u_N \end{pmatrix} = -h \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + h \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} (\mathbf{D}_N)_{00} & (\mathbf{D}_N)_{0N} \\ (\mathbf{D}_N)_{N0} & (\mathbf{D}_N)_{NN} \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} (\mathbf{D}_N)_{01} & (\mathbf{D}_N)_{02} & \cdots & (\mathbf{D}_N)_{0(N-1)} \\ (\mathbf{D}_N)_{N1} & (\mathbf{D}_N)_{N2} & \cdots & (\mathbf{D}_N)_{N(N-1)} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} \quad (5-96)$$

至此，洛平边界条件 $(u \pm hu')|_{x=\pm 1}=0$ 被转化成了对 u_0 和 u_N 的约束条件 (5-96)，该式可用程序容易地计算。处理 $(u \pm hu')|_{x=\pm 1} \neq 0$ 的情况与之类似，不再赘述。

下面讨论一维热传导方程：

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & -1 < x < 1 \\ \left(u \pm h \frac{\partial u}{\partial x} \right) \Big|_{x=\pm 1} = 0 \\ u|_{t=0} = 1 + \cos(\pi x) \end{cases} \quad (5-97)$$

上式描述了均匀细杆上的热传导过程， u 代表温度分布， t 代表时间， x 代表杆上的坐标，洛平边界条件代表杆在边界处是自由冷却的。所谓自由冷却，是指杆的两端与周围环境根据牛顿冷却定律交换热量，即单位时间单位横截面积从边界处释放的热量与边界和外界的温度差成正比。由此可以得到 $(u \pm h \partial u / \partial x)|_{\text{边}} = v$ ，其中 v 为外界温度， h 为一常数，正负号分别对应于右边界和左边界。可以预料，若时间 t 足够长，杆上的温度将处

处趋于 v 。

根据方程的物理意义可知, $\partial u/\partial t|_{\text{边}}$ 和 $\partial(\partial u/\partial x)/\partial t|_{\text{边}}$ 均是连续变化的, 则 u 对 x 和 t 的二阶混合偏导数可以交换求导顺序 (证明从略), 即 $\partial(\partial u/\partial x)/\partial t = \partial(\partial u/\partial t)/\partial x$ 。所以式 (5-97) 的边界条件可等价写为:

$$\left(u \pm h \frac{\partial u}{\partial x}\right)\bigg|_{x=\pm 1} = 0 \Leftrightarrow \begin{cases} \left[\frac{\partial u}{\partial t} \pm h \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t}\right)\right]\bigg|_{x=\pm 1} = 0 \\ \left(u \pm h \frac{\partial u}{\partial x}\right)\bigg|_{\substack{t=0 \\ x=\pm 1}} = 0 \end{cases} \quad (5-98)$$

则式 (5-97) 可被转化为如下形式:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & -1 < x < 1 \\ \left[\frac{\partial u}{\partial t} \pm h \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t}\right)\right]\bigg|_{x=\pm 1} = 0 \\ u|_{t=0} = 1 + \cos(\pi x) \end{cases} \quad (5-99)$$

接下来就可以利用 `ode45` 函数计算上式, 根据式 (5-96) 处理其中的边界条件 (处理时需要将 $\partial u/\partial t$ 看做 u)。取 $h=0.1$, 代码如下:

程序 5-25

主程序代码如下:

```
clear all; close all;
L=2; N=20; h=0.1;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
D2=D^2; x=L/2*x;
%初始条件
u=1+cos(pi*x);
%洛平边界条件
A=h*[1 0;0 -1];
BC=-(A*D([1 N+1],[1 N+1])+[1 0;0 1])*(A*D([1 N+1],2:N));
%求解
t=0:0.03:1.5;
[t,usol]=ode45('heat1D',t,u,[],N,D2,BC);
```

```

%画图
X=L/2:-0.05:-L/2;
u=interp2(x,t,usol,X,t,'cubic');
waterfall(X,t,u)
xlabel x, ylabel t, zlabel u
axis([-1 1 0 1.5 0 2]), view(-130,30)
%误差
E=usol+h*(D*usol)'; max(abs(E(:,1)))

```

文件 heat1D.m 代码如下：

```

function du=heat1D(t,u,dummy,N,D2,BC)
du=D2*u;
du([1 N+1])=BC*du(2:N);

```

输出结果如图 5-30 所示，热量迅速在杆上扩散开来，并通过边界传递到外界，杆两端与外界的温差越小，向外界传递热量的速度也就越慢，最终整个杆的温度将趋于 0。数值解在边界处的最大误差为 3.5111×10^{-15} 。

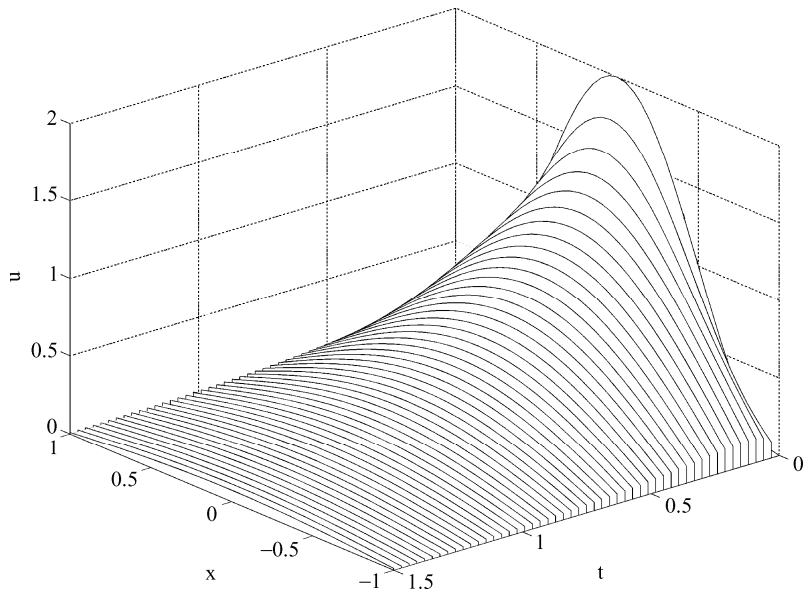


图 5-30 两端边界自由冷却的一维热传导问题的解

5.4.4 二维热传导方程

若两个边界分别采用洛平边界条件 $(u+hu')|_{x=-1}=0$ ($h \neq 0$) 和诺依曼边界条件 $u'|_{x=-1}=0$ ，即：

$$\begin{pmatrix} u(1) \\ 0 \end{pmatrix} + h \begin{pmatrix} u'(1) \\ u'(-1) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (5-100)$$

设函数 $u(x)$ 在切比雪夫点 $\mathbf{x}=(x_0, x_1, \dots, x_N)^T$ 处的取值为向量 $\mathbf{u}=(u_0, u_1, \dots, u_N)^T$ 。与 5.4.3 小节的过程类似，整理可得 $(u_0, u_N)^T$ 与 $(u_1, u_2, \dots, u_{N-1})^T$ 的关系：

$$\begin{pmatrix} u_0 \\ u_N \end{pmatrix} = -h \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + h \begin{pmatrix} (\mathbf{D}_N)_{00} & (\mathbf{D}_N)_{0N} \\ (\mathbf{D}_N)_{N0} & (\mathbf{D}_N)_{NN} \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{D}_N)_{01} & (\mathbf{D}_N)_{02} & \cdots & (\mathbf{D}_N)_{0(N-1)} \\ (\mathbf{D}_N)_{N1} & (\mathbf{D}_N)_{N2} & \cdots & (\mathbf{D}_N)_{N(N-1)} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} \quad (5-101)$$

这样就把边界条件 $(u+hu')|_{x=1}=0$ 和 $u'|_{x=-1}=0$ 转化为对 u_0 和 u_N 的约束条件，后面将用到这一关系。

式 (5-102) 描述了二维热传导问题，边界 $x=1$ 处的洛平边界条件代表该边界是自由冷却的，边界 $x=-1$ 和边界 $|y|=1$ 处的诺依曼边界条件代表这些边界是绝热的。

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -1 < x, y < 1 \\ \left(u + h \frac{\partial u}{\partial x} \right) \Big|_{x=1} = 0, & \frac{\partial u}{\partial x} \Big|_{x=-1} = \frac{\partial u}{\partial y} \Big|_{|y|=1} = 0 \\ u|_{t=0} = [1 + \cos(\pi x)][1 + \cos(\pi y)] \end{cases} \quad (5-102)$$

与一维热传导方程类似，为了能用 ode45 函数处理此问题，将上式等价写为：

$$\begin{cases} \frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u, & -1 < x, y < 1 \\ \left[\frac{\partial u}{\partial t} + h \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t} \right) \right] \Big|_{x=1} = 0, & \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t} \right) \Big|_{x=-1} = \frac{\partial}{\partial y} \left(\frac{\partial u}{\partial t} \right) \Big|_{|y|=1} = 0 \\ u|_{t=0} = [1 + \cos(\pi x)][1 + \cos(\pi y)] \end{cases} \quad (5-103)$$

根据式 (5-101) 处理边界 $x=\pm 1$ 处的洛平边界条件和诺依曼边界条件，根据式 (5-69) 处理边界 $y=\pm 1$ 处的诺依曼边界条件（处理时需要将 $\partial u / \partial t$ 看做 u ）。取 $h=0.1$ ，代码如下：

程序 5-26

主程序代码如下：

```
clear all; close all;
```

```
L=2; N=20; h=0.1;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
D2=D^2; x=L/2*x; y=x;
%初始条件
[X,Y]=meshgrid(x,y);
u=(1+cos(pi*X)).*(1+cos(pi*Y));
%边界条件
BCx=-(h*D([1 N+1],[1 N+1])+[1 0;0 0])\ (h*D([1 N+1],2:N));
BCy=-D([1 N+1],[1 N+1])\D([1 N+1],2:N);
%求解
t=[0 0.05 0.2 4];
[t,usol]=ode45('heat2D',t,u(:,:),N,D2,BCx,BCy);
%画图
for n=1:4
    subplot(2,2,n)
    u=reshape(usol(n,:),N+1,N+1);
    surf(x,y,u), shading interp
    axis([-1 1 -1 1 0 4]), view(15,15)
    xlabel x, ylabel y, zlabel u
    title(['t=' num2str(t(n))])
    %误差
    E=h*u*D'+u; max(abs(E(:,1)))
end
```

文件 heat2D.m 代码如下：

```
function du=heat2D(t,u,dummy,N,D2,BCx,BCy)
u=reshape(u,N+1,N+1);
du=D2*u+u*D2';
du([1 N+1],:)=BCy*du(2:N,:);
du(:,[1 N+1])=du(:,2:N)*BCx';
du=du(:);
```

计算结果如图 5-31 所示，热量从中央的高温部分流向四周的低温部分。因为热量只能通过边界 $x=1$ 传递到外界，而其他 3 个绝热边界处都积累了热量，所以温度普遍比边界 $x=1$ 处高。若时间 t 足够长，各处温度将趋于 0。此外，数值解在边界 $x=1$ 处的最大误差在 10^{-14} 数量级。

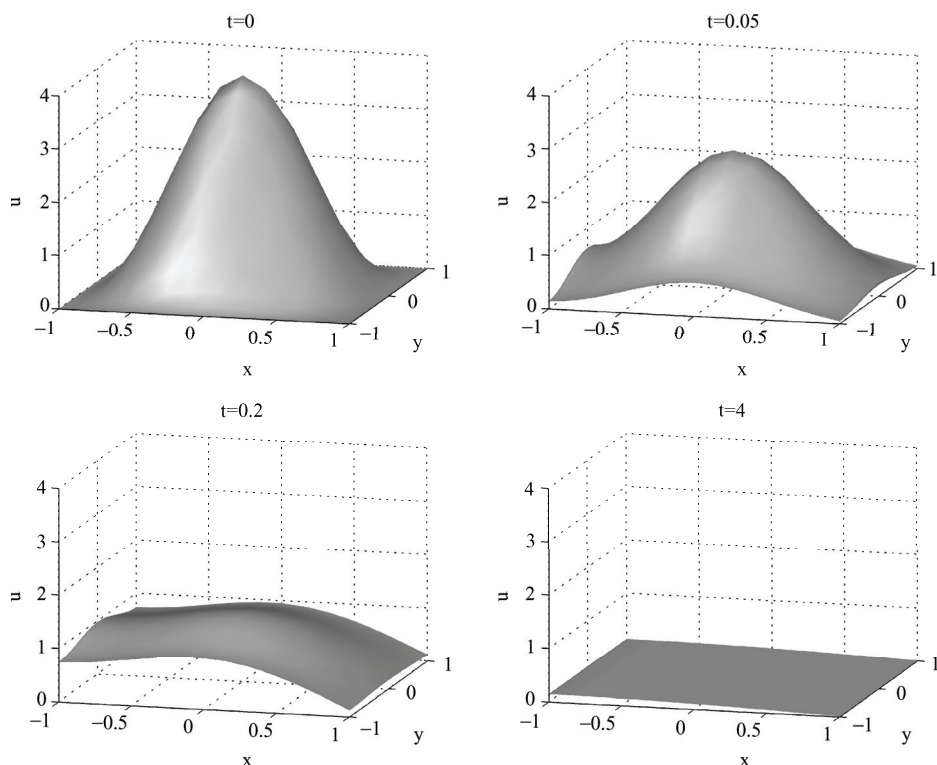


图 5-31 1 个边界自由冷却、其他 3 个边界绝热的二维热传导问题的解

5.5 利用切比雪夫谱方法求解复杂偏微分方程（组）

5.5.1 广义特征值问题

标准的特征值问题形如下式：

$$Au = \lambda u \quad (5-104)$$

其中， A 为已知方阵， u 为未知非零向量， λ 为未知常数。使上式成立的 λ 称为特征值，相应的向量 u 称为特征向量。可用 Matlab 的 `eig` 函数求解上述问题的特征值和特征向量，调用形式为 `eig(A)`。

而广义特征值问题的形式为：

$$Au = \lambda Bu \quad (5-105)$$

其中， A 和 B 为已知方阵， u 为未知非零向量， λ 为未知常数。使上式成立的 λ 称为特征值，相应的向量 u 称为特征向量。若方阵 B 可逆，广义特征值问题可化为式 (5-104) 的形式，即：

$$\mathbf{B}^{-1}\mathbf{A}u = \lambda u \quad (5-106)$$

但这么做既不简单也不实用，况且方阵 $\mathbf{B}$ 也未必是可逆的。求解广义特征值问题通常选用 QZ 法（QZ method），在 Matlab 中的调用形式为 `eig(A, B)`。以下面的特征值问题为例：

$$u'' = \lambda xu, \quad -2 < x < 2, \quad u(\pm 2) = 0 \quad (5-107)$$

用 $N+1$ 维向量 $\mathbf{u}_{(N+1) \times 1}$ 代表函数 $u(x)$ 在区间 $[-2, 2]$ 上的切比雪夫点 $\mathbf{x} = (x_0, x_1, \dots, x_N)^T$ 处的取值，删去其首尾元素得到 $N-1$ 维向量 $\tilde{\mathbf{u}}_{(N-1) \times 1}$ 。则式（5-107）的矩阵形式为：

$$\tilde{\mathbf{D}}_N^2 \tilde{\mathbf{u}}_{(N-1) \times 1} = \lambda \cdot \text{diag}(x_1, \dots, x_{N-1}) \tilde{\mathbf{u}}_{(N-1) \times 1} \quad (5-108)$$

对于这样的广义特征值问题，求解代码如下：

程序 5-27

```
clear all; close all;
L=4; N=40;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2);
D2=D^2; D2=D2(2:N,2:N); x=L/2*x;
%求解
[V,E]=eig(D2,diag(x(2:N))); E=diag(E);
i=find(E>0); E=E(i); V=V(:,i);
[eigenvalues,i]=sort(E); V=V(:,i);
%画图
x2=-2:0.01:2;
for n=1:9
    subplot(3,3,n)
    plot(x2,polyval(polyfit(x,[0;V(:,n);0],N),x2),'k','LineWidth',1.5)
    title(num2str(eigenvalues(n)))
    axis([-2 2 -1.1 1.1]), xlabel x, ylabel u
end
```

程序输出了前 9 个正特征值及相应的特征向量，如图 5-32 所示。

5.5.2 二维 Barkley 模型

在前面章节关于含时问题的实例中，使用了 `ode45` 处理函数 u 对时间 t 的导数 $\partial u / \partial t$ 。`ode45` 是基于 4、5 阶龙格-库塔法的变步长算法，可以根据所计算方程变化的快慢自行

调整步长，在保证误差符合要求的前提下尽可能快地求解问题。但需要将 u 的边界条件转化为针对 $\partial u / \partial t$ 的边界条件，这往往要求 u 的二阶混合偏导数 $\partial^2 u / \partial t \partial x$ 可以交换求导顺序。实际上，为避免这一不便，还可以使用欧拉法处理导数 $\partial u / \partial t$ ，但同时带来了如下问题：运算量大，精度低，代码略繁，需要人为选取合适的步长。变步长的 4、5 阶龙格-库塔法和欧拉法各有利弊，选取哪种方法应根据问题具体情况、计算机配置和个人喜好来决定。

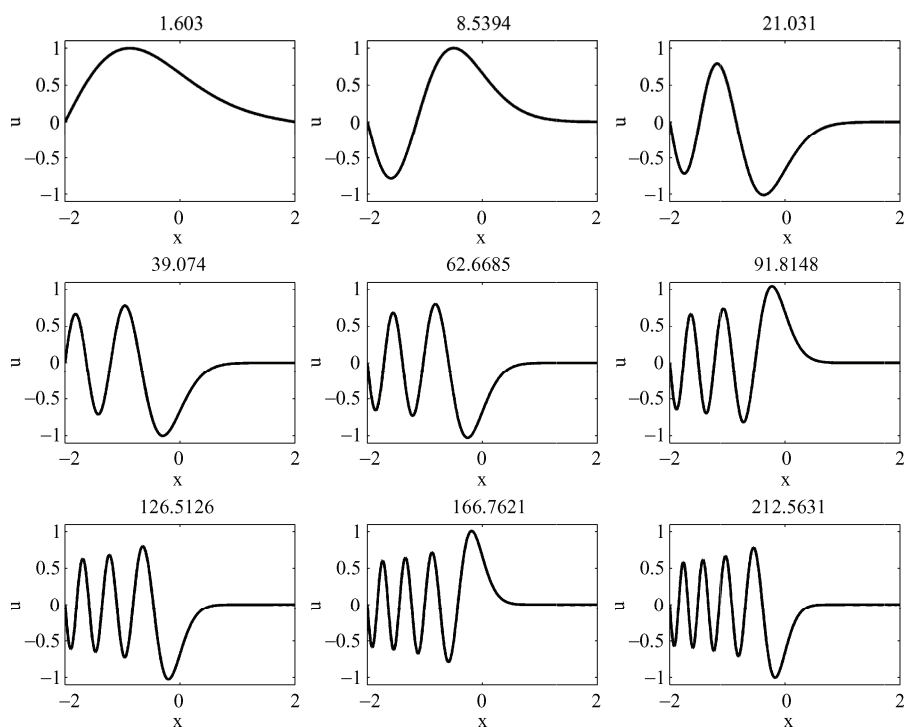


图 5-32 广义特征值问题的前 9 个特征值和特征向量

下面以数值计算反应-扩散系统中的螺旋波为例讲解如何用欧拉法解决含时问题。螺旋波是系统远离平衡态时由于系统自组织形成的一类特殊斑图。在 B-Z 反应、正在聚集的粘性霉菌、铂金表面的一氧化碳氧化以及心脏中均能观测到螺旋波的存在。描述螺旋波的 Barkley 模型为：

$$\begin{cases} \frac{\partial u}{\partial t} = d \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u + \frac{u(1-u)}{\varepsilon} \left(u - \frac{b+v}{a} \right) \\ \frac{\partial v}{\partial t} = u - v \end{cases} \quad (5-109)$$

其中， u 和 v 是 2 种能够相互转化的化学物质的浓度， t 为时间， x 、 y 为空间坐标， d 为扩散系数， a 、 b 和 ε 为系统参数。

根据欧拉法， $\partial u / \partial t$ 和 $\partial v / \partial t$ 表示为：

$$\begin{aligned}\frac{\partial u}{\partial t} &\rightarrow \frac{u(t+\Delta t)-u(t)}{\Delta t} \\ \frac{\partial v}{\partial t} &\rightarrow \frac{v(t+\Delta t)-v(t)}{\Delta t}\end{aligned}\quad (5-110)$$

在二维区域的切比雪夫点上将函数 u 、 v 离散化为 $N+1$ 阶方阵 $\mathbf{u}_{(N+1)\times(N+1)}$ 、 $\mathbf{v}_{(N+1)\times(N+1)}$ 。利用切比雪夫求导矩阵处理方程中的扩散项 $(\partial^2/\partial x^2 + \partial^2/\partial y^2)u$ ，即：

$$\begin{aligned}\frac{\partial^2 u}{\partial y^2} &\rightarrow \mathbf{D}_N^2 \mathbf{u}_{(N+1)\times(N+1)} \\ \frac{\partial^2 u}{\partial x^2} &\rightarrow \left(\mathbf{D}_N^2 \left(\mathbf{u}_{(N+1)\times(N+1)} \right)^T \right)^T = \mathbf{u}_{(N+1)\times(N+1)} \left(\mathbf{D}_N^2 \right)^T\end{aligned}\quad (5-111)$$

选取计算区域 Ω 为： $-60 < x, y < 60$ 。由于化学物质在边界处没有进出，所以边界条件为 $\partial u/\partial n|_{\partial\Omega} = \partial v/\partial n|_{\partial\Omega} = 0$ (n 代表边界 $\partial\Omega$ 处的法向)，可根据式 (5-69) 处理该诺依曼边界条件。参数取值为： $a=0.5$ ， $b=0.01$ ， $d=1.6$ ， $\varepsilon=0.02$ 。为产生螺旋波，需要将两种化学物质浓度 u 、 v 的初始状态设置为在空间上具有 3 个不同值 0、0.5、1 的浓度梯度的状态，具体参见代码。

程序 5-28

```
clear all; close all;
L=120; N=90;
%构造切比雪夫求导矩阵
[D,x]=cheb(N); D=D/(L/2); D2=D^2;
x=L/2*x; y=x;
%边界条件
BCx=-(D([1 N+1],2:N'))/(D([1 N+1],[1 N+1]))';
BCy=-D([1 N+1],[1 N+1])\D([1 N+1],2:N);
%初始条件
u_old=zeros(N+1,N+1); v_old=u_old;
u_old(N/2,1:N/2)=0.5; v_old(N/2,1:N/2)=0.5;
u_old(N/2-1,1:N/2)=1; v_old(N/2+1,1:N/2)=1;
usol(:,,1)=u_old; vsol(:,,1)=v_old;
%求解
dt=0.002; epsilon=0.02; a=0.5; b=0.01; d=1.6;
for n=1:3
    for m=1:2500
        u=u_old+dt*(d*(u_old*(D2')+D2*u_old)+ ...
            1/epsilon*u_old.*(1-u_old).*(u_old-(b+v_old)/a));
        v=v_old+dt*(u_old-v_old);
        u([1 N+1],:)=BCy*u(2:N,:); u(:,[1 N+1])=u(:,2:N)*BCx;
```

```

v([1 N+1],:)=BCy*v(2:N,:); v(:,[1 N+1])=v(:,2:N)*BCx;
u_old=u; v_old=v;
end
usol(:,n+1)=u; vsol(:,n+1)=v;
end
%画图
for n=1:4
    subplot(2,2,n)
    gca=pcolor(x,y,usol(:,n));
    set(gca,'LineStyle','none'), axis off
    shading interp, axis square
end

```

代码中选取的时间步长为 0.002，每前进一步，都要根据边界条件修正 u 、 v 在边界处的取值。程序每隔 5 个单位时间输出一幅小图，如图 5-33 所示。



图 5-33 螺旋波的产生过程

5.5.3 二维平流-扩散方程

第 4 章给出了在周期性边界条件下求解二维平流-扩散方程的方法，本小节将在狄利克莱边界条件下求解该方程，有兴趣的读者可以比较它们之间的差异。

$$\begin{cases} \frac{\partial \omega}{\partial t} = v \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \omega + \frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \psi = \omega \end{cases} \quad (5-112)$$

上式为二维平流-扩散方程，其中， ψ 、 ω 分别是流函数和涡量， x 、 y 为空间坐标， t 是时间， ν 为常数。计算区域 Ω 为： $-10 < x, y < 10$ ，边界条件为 $\psi|_{\partial\Omega} = \omega|_{\partial\Omega} = 0$ 。用切比雪夫点在计算区域 Ω 内将函数 ψ 、 ω 离散化为 $N+1$ 阶方阵 $\boldsymbol{\psi}_{(N+1) \times (N+1)}$ 、 $\boldsymbol{\omega}_{(N+1) \times (N+1)}$ 。式 (5-112) 的第一式等号左边的 $\partial\omega/\partial t$ 可用 ode45 函数计算，等号右边可表示为矩阵形式（“~”代表删除首尾行首尾列的操作）：

$$\begin{aligned} & \nu \left[\tilde{\boldsymbol{\omega}}_{(N-1) \times (N-1)} \left(\tilde{\boldsymbol{D}}_N^2 \right)^T + \tilde{\boldsymbol{D}}_N^2 \tilde{\boldsymbol{\omega}}_{(N-1) \times (N-1)} \right] + \tilde{\boldsymbol{D}}_N \tilde{\boldsymbol{\psi}}_{(N-1) \times (N-1)} \cdot \tilde{\boldsymbol{\omega}}_{(N-1) \times (N-1)} \left(\tilde{\boldsymbol{D}}_N \right)^T \\ & - \tilde{\boldsymbol{\psi}}_{(N-1) \times (N-1)} \left(\tilde{\boldsymbol{D}}_N \right)^T \cdot \tilde{\boldsymbol{D}}_N \tilde{\boldsymbol{\omega}}_{(N-1) \times (N-1)} \end{aligned} \quad (5-113)$$

将函数 ψ 、 ω 的离散形式由方阵写为向量，并根据式 (5-34) 所示的拉普拉斯算符矩阵形式，式 (5-112) 的第二式可表为：

$$\tilde{\boldsymbol{\psi}}_{(N-1)^2 \times 1} = \left(\tilde{\boldsymbol{D}}_N^2 \otimes \boldsymbol{I}_{N-1} + \boldsymbol{I}_{N-1} \otimes \tilde{\boldsymbol{D}}_N^2 \right)^{-1} \tilde{\boldsymbol{\omega}}_{(N-1)^2 \times 1} \quad (5-114)$$

取 $\nu=0.001$ ，初始条件 $\omega(x, y, 0) = \text{sech}[(x+2)^2 + y^2/20] + \text{sech}[(x-2)^2 + y^2/20]$ ，求解式 (5-112) 代码如下：

程序 5-29

主程序代码如下：

```
clear all; close all;
L=20; N=60;
%构造拉普拉斯算符矩阵并求逆
[D,x]=cheb(N); x=L/2*x; y=x;
[X,Y]=meshgrid(x(2:N),y(2:N));
D=D/(L/2); D2=D^2;
D=D(2:N,2:N); D2=D2(2:N,2:N);
I=eye(N-1); LA=kron(I,D2)+kron(D2,I);
LA_inv=inv(LA);
%初始条件
w=sech((X+2).^2+Y.^2/20)+sech((X-2).^2+Y.^2/20);
%求解
v=0.001; t=0:5:15;
[t,wsol]=ode113('advection_diffusion',t,w(:),[],N,D,D2,LA_inv,v);
%画图
x2=-L/2:0.2:L/2; y2=x2;
for n=1:4
    subplot(2,2,n)
    w=zeros(N+1);
```

```

w(2:N,2:N)=reshape(wsol(n,:),N-1,N-1);
w=interp2(x,y,w,x2,y2,'cubic');
gca=pcolor(w); set(gca,'LineStyle','none'), shading interp
title(['t=' num2str(t(n))]), axis('square'), axis off
end

```

文件 advection_diffusion.m 代码如下：

```

function dw=advection_diffusion(t,w,dummy,N,D,D2,LA_inv,v)
psi=reshape(LA_inv*w,N-1,N-1);
w=reshape(w,N-1,N-1);
dw=reshape(v*(D2*w+w*D2')-(psi*D').*(D*w)+(D*psi).*(w*D'),(N-1)^2,1);

```

程序中的拉普拉斯算符矩阵为 $(N-1)^2$ 阶方阵，它的求逆运算将耗费相当多 CPU 和内存资源。为节约时间，主程序仅对其做一次求逆运算，每次调用 `advection_diffusion` 函数时，将其逆矩阵作为参数传递进去。即使这样，执行本程序的时间仍然比本书中的其他程序略多一些。此外，因为计算时删除了方阵 $\psi_{(N+1) \times (N+1)}$ 、 $\omega_{(N+1) \times (N+1)}$ 在边界处的值，所以最后要在相应位置补 0。又因为依照切比雪夫点划分计算区域 Ω 导致计算结果在空间上的不均匀分布，所以最后使用了 `interp2` 函数对结果进行插值使其均匀化。最终结果如图 5-34 所示。

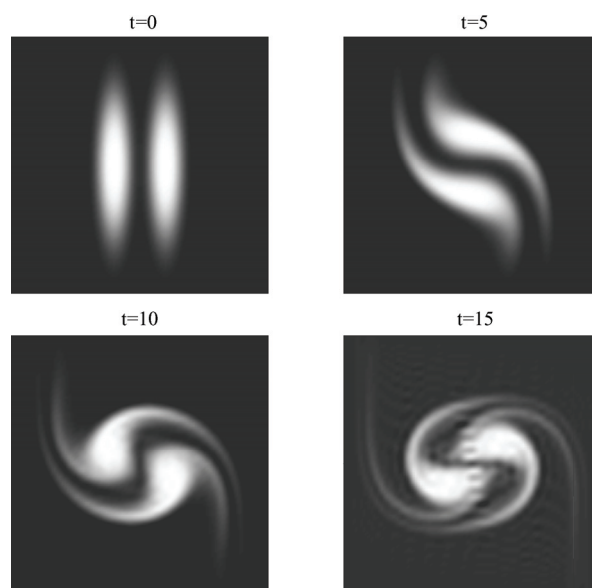


图 5-34 狄利克莱边界条件下的二维平流-扩散方程的计算结果

附录

附录 A Matlab 主要符号和函数

Matlab 中的内置符号和函数众多，限于篇幅这里不能全部涉及。本附录只将书中主要用到的一些符号和函数的语法和说明罗列出来，以便读者查询。

A.1 运算符、操作符和常量

A.1.1 算术运算符

算术运算符用来进行常见的数学运算，用法见表 A-1。

表 A-1 算术运算符

运 算 符	说 明	运 算 符	说 明
+	矩阵加法/矩阵与标量加法	-	矩阵减法/矩阵与标量减法
.*	矩阵对应元素相乘	*	矩阵乘法
./	矩阵对应元素的右除法	/	矩阵右除法
.\	矩阵对应元素的左除法	\	矩阵左除法
.^	矩阵对应元素的乘方	^	矩阵乘方
.'	矩阵转置	'	矩阵共轭转置

矩阵的加/减就是矩阵对应元素的加/减，因此要求两个矩阵的行数、列数相同。标量是指 1×1 矩阵，矩阵与标量之间的加/减法就是矩阵每一个元素与标量的加/减。

矩阵右除 A/B 相当于 $A * \text{inv}(B)$ ，矩阵左除 $A \backslash B$ 相当于 $\text{inv}(A) * B$ 。

A.1.2 关系运算符

关系运算符用来进行大小的比较，返回值为 1 或 0（表示真或假），说明见表 A-2。

表 A-2 关系运算符

运 算 符	说 明	运 算 符	说 明
>	大于	>=	大于或等于
<	小于	<=	小于或等于
==	相等	~=	不等

A.1.3 逻辑运算符

除了用表 A-3 中的运算符实现“与”、“或”、“非”的操作，还可以等效使用 `and`、`or`、`not` 函数。

表 A-3 逻辑运算符

运 算 符	说 明	运 算 符	说 明
<code>&</code>	逻辑与	<code>~</code>	逻辑非
<code> </code>	逻辑或	<code>xor</code>	逻辑异或

A.1.4 常用操作符

1. “[” 和 “]” 用于创建向量和矩阵。

2. “,” 和 “;” 在创建向量、矩阵和写代码时均会用到。创建向量和矩阵时，在同一行的元素用 “,” 或空格隔开，不同行的元素用 “;” 隔开。写每一行代码时，可分别用 “,” 或 “;” 结尾，前者将在 Matlab 的 Command Window 中显示语句的计算结果，后者则不显示。

3. “:” 可以用来产生向量，也可以用来访问矩阵的特定元素。如：

(1) `a:h:b` 产生间隔为 `h` 从 `a` 到 `b` 的等差序列。如果 `a` 和 `b` 的差不是 `h` 的整数倍，那么输出结果将不包含 `b`。另外，`a:b` 产生间隔为 1 从 `a` 到 `b` 的等差序列。

(2) `A(n,:)`、`A(:,n)` 分别返回矩阵 `A` 的第 `n` 行、第 `n` 列元素。另外，`A(n:m,:)`、`A(:,n:m)` 分别返回矩阵 `A` 的第 `n` 到 `m` 行、第 `n` 到 `m` 列的元素。

(3) `A(k,n:m)`、`A(n:m,k)` 分别返回矩阵 `A` 第 `k` 行的第 `n` 到 `m` 个元素、第 `k` 列的第 `n` 到 `m` 个元素。另外，`A(n:m,j:k)` 返回矩阵 `A` 中在第 `n` 到 `m` 行且在第 `j` 到 `k` 列的元素。

(4) 值得一提的是，可使用角标 `end` 调用向量或矩阵的最后一个元素。如 `A(end,end)` 代表最后一行的最后一个元素，`A(end,n:m)` 代表最后一行第 `n` 到 `m` 个元素。

(5) `A(:)` 将矩阵 `A` 的所有元素组成一列返回。

部分有代表性的示例如下：

```
>> A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```

1     2     3
4     5     6
7     8     9
```

```
>> A(1:2,2:3)
```

```
ans =
```

```

2     3
5     6
```

```
>> A(:)
```

```
ans =
```

```
1
4
7
2
5
8
3
6
9
```

- 4. “'” 用于创建字符串。
- 5. “...” 为续行号，可把本行和下一行连起来执行，一般用于一行特别长语句的情况。
- 6. “%” 用于注释其后的一行内容，注释的内容不会执行。

A.1.5 专用常量

专用常量见表 A-4。

表 A-4 专用常量

常 量	说 明	常 量	说 明
ans	未赋值的结果存储变量	eps	浮点数的精度
pi	圆周率 π	i 或 j	虚数单位
Inf	无穷大 ∞	NaN	非数 (Not a Number)，代表 0/0 等
realmax	最大正浮点数	realmin	最小正浮点数

A.2 矩阵、图形窗口相关函数

A.2.1 特殊矩阵的生成

表 A-5 列出了生成特殊矩阵的函数。

表 A-5 生成特殊矩阵的函数

函 数	说 明	语 法	
zeros	生成全 0 矩阵	zeros(n)	生成 n 阶全 0 方阵
		zeros(m,n)	生成 m×n 全 0 矩阵
ones	生成全 1 矩阵	ones(n)	生成 n 阶全 1 方阵
		ones(m,n)	生成 m×n 全 1 矩阵
eye	生成单位矩阵	eye(n)	生成 n 阶单位矩阵
magic	生成魔方矩阵	magic(n)	生成 n×n 魔方矩阵
rand	生成 0 至 1 均匀分布随机数	rand(n)	生成 n 阶均匀分布随机数方阵
		rand(m,n)	生成 m×n 均匀分布随机数矩阵

(续)

函 数	说 明	语 法
randn	生成均值为 0 方差为 1 的正态分布随机数	randn(n) randn(m,n) 生成 $m \times n$ 正态分布随机数矩阵
diag	生成对角矩阵或返回矩阵对角元素	diag(v) diag(A) 将向量 v 放在返回矩阵的主对角线上以向量形式返回矩阵 A 的主对角线
spdiags	生成带状稀疏矩阵	spdiags(B,d,m,n) 将矩阵 B 的每一列放在 $m \times n$ 稀疏矩阵的向量 d 所指定的对角线上
linspace	生成等间隔向量	linspace(a,b,n) 把区间 $[a,b]$ 均分成 n 个数并以向量返回

部分有代表性的示例如下：

```
>> zeros(2)
```

```
ans =
```

```
0    0
0    0
```

```
>> ones(2,3)
```

```
ans =
```

```
1    1    1
1    1    1
```

```
>> eye(3)
```

```
ans =
```

```
1    0    0
0    1    0
0    0    1
```

```
>> magic(3)
```

```
ans =
```

```
8    1    6
3    5    7
4    9    2
```

```
>> rand(3,4)
```

```
ans =

    0.8147    0.9134    0.2785    0.9649
    0.9058    0.6324    0.5469    0.1576
    0.1270    0.0975    0.9575    0.9706
```

```
>> diag([1 2 3])
```

```
ans =

     1     0     0
     0     2     0
     0     0     3
```

```
>> spdiags(ones(4,2),[-1 1],4,4)
```

```
ans =

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

```
>> linspace(1,2,4)
```

```
ans =

    1.0000    1.3333    1.6667    2.0000
```

A.2.2 常用矩阵相关函数

表 A-6 列出了一些常用的矩阵相关函数

表 A-6 常用的矩阵相关函数

函数	说 明	语 法
sort	从小到大排序	sort(v) 给向量 v 排序
		sort(A) 给矩阵 A 的每一列排序
max	返回最大值	max(v) 返回向量 v 的最大值
		max(A) 返回矩阵 A 每一列的最大值
min	返回最小值	min(v) 返回向量 v 的最小值
		min(A) 返回矩阵 A 每一列的最小值
kron	计算克罗内克积	kron(A,B) 返回矩阵 A、B 的克罗内克积

(续)

函数	说 明	语 法
eig	计算特征值、特征向量	eig(A) [V,D]=eig(A) 返回矩阵 A 的所有特征值 返回矩阵 A 所有特征值构成的对角矩阵 D、特征向量构成的矩阵 V
inv	计算逆矩阵	inv(A) 返回方阵 A 的逆矩阵
det	计算行列式	det(A) 返回方阵 A 的行列式的值
real	返回实部	real(A) 返回矩阵 A 所有元素的实部
imag	返回虚部	imag(A) 返回矩阵 A 所有元素的虚部
abs	返回绝对值/模	abs(A) 返回矩阵 A 所有元素的绝对值/模
conj	返回共轭复数	conj(A) 返回矩阵 A 所有元素的共轭复数
spy	用图形表示稀疏矩阵	spy(A) 返回稀疏矩阵 A 的示意图
ndims	返回矩阵的维数	ndims(A) 返回矩阵 A 的维数
numel	返回矩阵的元素个数	numel(A) 返回矩阵 A 的元素个数
length	返回矩阵行、列数的最大值	length(A) 返回矩阵 A 行数、列数的最大值
size	返回矩阵的行数、列数	[m,n]=size(A) 返回矩阵 A 的行数 m、列数 n
reshape	将矩阵改变成指定形状	reshape(A,m,n) 从列方向上将矩阵 A 改变成 m×n 矩阵
clear	清除 workspace 中指定的变量和函数，释放内存	clear A clear all 清除变量 A 清空 workspace，用来避免之前执行的代码可能造成的影响

部分有代表性的示例如下：

```
>> A=magic(3)
```

```
A =
```

```

8     1     6
3     5     7
4     9     2
```

```
>> sort(A)
```

```
ans =
```

```

3     1     2
4     5     6
8     9     7
```

```
>> max(A)
```

```
ans =
```

```

      8      9      7

>> min(A)

ans =

      3      1      2

>> eig(A)

ans =

    15.0000
     4.8990
    -4.8990

>> inv(A)

ans =

    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028

>> det(A)

ans =

   -360

>> A=[1:5]+[6:10]*i

A =

Columns 1 through 3

    1.0000 + 6.0000i    2.0000 + 7.0000i    3.0000 + 8.0000i

Columns 4 through 5

    4.0000 + 9.0000i    5.0000 +10.0000i

>> real(A)

```

```
ans =  
  
    1    2    3    4    5  
  
>> imag(A)  
  
ans =  
  
    6    7    8    9   10  
  
>> abs(A)  
  
ans =  
  
    6.0828    7.2801    8.5440    9.8489   11.1803  
  
>> conj(A)  
  
ans =  
  
Columns 1 through 3  
  
    1.0000 - 6.0000i    2.0000 - 7.0000i    3.0000 - 8.0000i  
  
Columns 4 through 5  
  
    4.0000 - 9.0000i    5.0000 -10.0000i  
  
>> ndims(ones(3))  
  
ans =  
  
    2  
  
>> numel(ones(2,4))  
  
ans =  
  
    8  
  
>> length(ones(2,4))
```

```
ans =
```

```
4
```

```
>> size(ones(2,4))
```

```
ans =
```

```
2    4
```

```
>> reshape(eye(4),2,8)
```

```
ans =
```

```
1    0    0    0    0    1    0    0
0    0    1    0    0    0    0    1
```

A.2.3 图形窗口控制函数

表 A-7 列出了常用的图形窗口控制函数。

表 A-7 图形窗口控制函数

函 数	说 明	语 法
clc	清空 Command Window	clc 清空 Command Window 中的内容
figure	创建图形窗口	figure figure(h) 创建一个图形窗口 创建句柄为 h 的图形窗口，若此句柄已存在，则将其设为当前图形窗口
clf	清除图形	clf clf(h) 清除当前窗口的图形 清除句柄为 h 的窗口的图形
close	关闭图形窗口	close close all 关闭当前图形窗口 关闭所有图形窗口
gcf	返回当前图形窗口句柄	gcf 当前图形窗口句柄
subplot	画子图形	subplot(m,n,p) 将图形窗口划分为 m×n 个子图，并将第 p 个子图作为当前图形
hold	保留图形	hold on hold off 保留当前图形，使后一个图形可在其上叠加 退出上述模式
title	加图形标题	title('string') 将 string 作为图形标题
xlabel	给 x 轴加标签	xlabel x 给 x 轴加标签 x
ylabel	给 y 轴加标签	ylabel y 给 y 轴加标签 y
zlabel	给 z 轴加标签	zlabel z 给 z 轴加标签 z
legend	给数据加图形标签	legend('s1',...) 给数据加图形标签 s1……
view	设置观察角度	view(az,el) az 为方位角，el 为仰角，单位为度
axis	设置坐标轴范围	axis([a b c d]) x 轴范围为[a, b]，y 轴范围为[c, d]
drawnow	强制刷新图形窗口	drawnow 强制刷新图形窗口

附录 B 将计算结果制作成 gif 动画

求解包含时间的偏微分方程（组）将得到随着时间变化的数值结果，把这样的数据制作成 gif 动画并结合到幻灯片中，在毕业答辩、小组讨论、课堂教学等场合有着广泛的应用。生动的彩色 gif 动画具有很强的表现力，令人刮目相看，大大提高了报告人所讲述理论结果的直观性、生动性、观赏性。

生成 gif 动画主要用到 4 个函数：getframe、frame2im、rgb2ind、imwrite。

(1) getframe 函数的一般调用形式为：F=getframe(h)，其作用是截取句柄为 h 的窗口内的一帧图像。

(2) frame2im 函数的作用是把一帧截图转为图像数据。

(3) rgb2ind 函数的作用是将 RGB 图像转换为索引图像，一般调用形式为：[X,map]=rgb2ind(RGB,n)。其中，X、map 分别为转换后的图像数据和颜色表数据，RGB 为转换前的图像数据，n 指定 map 中的颜色数。

(4) imwrite 函数的作用是将图像数据写入图像文件，一般调用形式为：imwrite(X,map,filename,fmt,Param1,Val1,Param2,Val2...)。其中，X、map 意义同上，filename 为文件名，fmt 为文件格式，Param1,Val1,Param2,Val2... 为若干可选参数及其取值。如：参数 LoopCount 为动画的循环播放次数，这里设为 inf，即无穷大。参数 DelayTime 为每帧间隔时间，单位秒。参数 WriteMode 为写入文件的模式，有覆盖 overwrite（默认）和追加 append 两种选择。

生成 gif 动画的示例代码如下：

程序 B-1

主程序代码如下：

```
clear all; close all;
x=-1:0.02:1; y=x;
[X,Y]=meshgrid(x,y);
filename='test.gif';
for a=1:10
    u=a*exp(-10*(X.^2+Y.^2));
    mesh(x,y,u), axis([-1 1 -1 1 0 10]), drawnow
    im=frame2im(getframe(gcf));
    [A,map]=rgb2ind(im,256);
    if a==1
        %先以覆盖模式写入指定的 gif 文件
        imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',0.1);
    else
```

```
%再以追加模式将每一帧写入 gif 文件
imwrite(A,map,filename,'gif','WriteMode','append','DelayTime',0.1);
end
end
```

运行代码之后在当前目录下生成 gif 文件，缩略图如图 B-1 所示，该动画显示了一个三维高斯函数的峰值逐渐增大的过程。

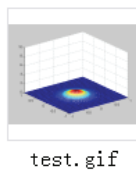


图 B-1 生成 gif 动画的缩略图

参 考 文 献

- [1] J Nathan Kutz. Practical Scientific Computing[OL]. <http://courses.washington.edu/amath581/581.pdf>.
- [2] Lloyd N Trefethen. Spectral Methods in MATLAB[M]. 北京：清华大学出版社，2011.
- [3] 崔国华，许如初. 计算方法[M]. 北京：电子工业出版社，2002.
- [4] 孙志忠. 偏微分方程数值解法[M]. 北京：科学出版社，2005.
- [5] 陆君安，尚涛，谢进，谷平. 偏微分方程的 MATLAB 解法[M]. 武汉：武汉大学出版社，2001.
- [6] 姚端正，梁家宝. 数学物理方法[M]. 武汉：武汉大学出版社，1997.
- [7] Cleve B Moler. MATLAB 数值计算[M]. 喻文健，译. 北京：机械工业出版社，2006.

跋

谱方法的本质就是将函数近似写成光滑函数的有限级数展开式，因此，针对越是光滑的函数，级数展开式的项数就越少，谱方法的精度也就越高。例如对于一个存在 p 阶导数的函数，用谱求导矩阵求出的 v 阶导数将具有 $O(h^{p-v})$ 的精度。精度高是谱方法最大的优势，对光滑函数的微分问题，它一般可达逾 10 位有效数字的精度，但有限差分法和有限元法通常只有 4 位或 5 位有效数字的精度。这是不难理解的，由于谱方法是一种全局的算法，它在计算某一处的导数时用到了所有已知点。而有限差分法作为一种局部算法，某一处的导数仅由临近的几个点运算得到，相比之下精度大打折扣。有限元法也是一种局部算法，其精度受到剖分单元的几何形状、剖分密度影响，还受制于计算机有限的计算速度和存储量，提高精度需要很大代价。

因为傅里叶谱方法无需构造大型矩阵，仅依赖于运算复杂度为 $O(M\log N)$ 的 FFT 算法，所以它在保证高精度的同时，运算量也大大低于有限差分法和有限元法，且只需要很少的计算机内存，它的不足之处就是仅适用于周期性边界条件下的微分问题。切比雪夫谱方法可以处理第一、二、三类边界条件下的偏微分问题，但只能在规则区域内求解（直角坐标系下为矩形，极坐标系下为圆形）。这正是由谱方法的全局性造成的，全局性在带来高精度的同时，也限制了求解区域的一般性。仅在求解区域这一点上，谱方法不及有限元法的适用性强。

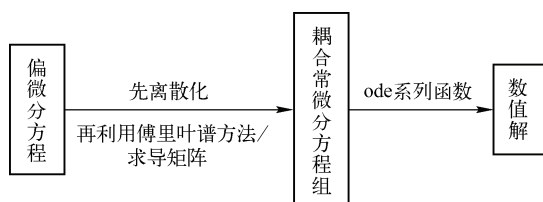
总结本书中处理微分方程的一般思路：

(1) 针对只含空间坐标 $(x, y, \dots)$ 不含时间 t 的微分方程（椭圆型方程、特征值问题等）。求解思路是先将函数 u 和变量 x, y 等离散化，再利用求导矩阵将微分方程转换为矩阵的基本问题。如椭圆型方程就成了形如 $Au=f$ 的矩阵问题，数值解即为 $u=A^{-1}f$ 。再如特征值问题可写为 $Au=\lambda u$ 的矩阵形式，数值解即为矩阵 A 的特征值和特征向量。

(2) 针对形如下式的包含空间坐标 x 和时间 t 的偏微分方程（抛物型方程、非线性方程等）或方程组：

$$\frac{\partial u}{\partial t} = f\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \dots, x, t\right)$$

求解思路为先将函数 u 和变量 x 离散化，再利用傅里叶谱方法或求导矩阵处理等号右边的 $\partial u/\partial x, \partial^2 u/\partial x^2, \dots$ ，这样就可以将偏微分方程转换为耦合常微分方程组。最后使用时间步进法（推荐 Matlab 的 ode 系列函数）数值计算此耦合常微分方程组，得到指定时刻的数值解。上述过程见如下示意图。



而对于等号左边为高阶导数 $\partial^n u / \partial t^n$ （双曲型方程）或者等号右边有 u 在多个空间维度（ $x, y \cdots$ ）上的偏导数的情况，求解方法也大同小异。

在实际问题中遇到的规则区域上的微分方程（组）基本都可以用上述思路又快又好地数值求解。

在线互动交流平台

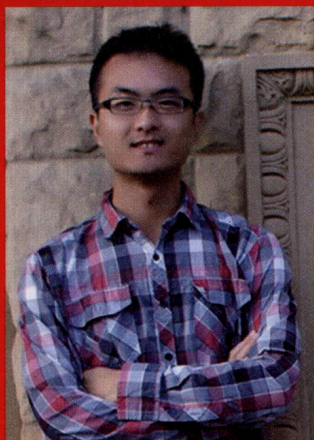
官方微博: <http://weibo.com/cmpjsj>

豆瓣网: <http://site.douban.com/139085/>

读者信箱: cmp_itbook@163.com

Matlab 微分方程高效解法: 谱方法原理与实现

作者简介



张晓, 85 后, 博士。曾以联合培养博士生身份赴美国华盛顿大学研习数值计算方法及其在光学中的应用, 目前在清华大学物理系做博士后研究工作。研究方向包括锁模激光技术、光学相干层析成像和高速全光信号处理。爱好计算机网络技术, 出版图书有《黑客攻防入门与进阶全程实录》、《骇客入侵秘辛与防御过程全攻略》, 并为网络安全期刊撰稿多年。



书中代码可在

<http://www.cmpbook.com> 下载。

地址: 北京市百万庄大街22号

邮政编码: 100037

电话服务

服务咨询热线: 010-88361066

读者购书热线: 010-68326294

010-88379203

网络服务

机工官网: www.cmpbook.com

机工官博: weibo.com/cmp1952

金书网: www.golden-book.com

教育服务网: www.cmpedu.com

封面无防伪标均为盗版



机械工业出版社
微信公众号



计算机分社微信服务号

上架建议 计算机/Matlab

ISBN 978-7-111-51623-1

策划编辑◎车 忱 / 封面设计◎



智识文化
Zhi Shi Culture

ISBN 978-7-111-51623-1



9 787111 516231 >

定价: 43.00 元