

高等应用数学问题的 MATLAB 求解

(第三版)

薛定宇 陈阳泉 著

清华大学出版社

北 京

内 容 简 介

本书首先介绍 MATLAB 语言程序设计的基本内容,在此基础上系统介绍各个应用数学领域的问题求解,如基于 MATLAB 的微积分问题、线性代数问题的计算机求解、积分变换和复变函数问题、非线性方程与最优化问题、常微分方程与偏微分方程问题、数据插值与函数逼近问题、概率论与数理统计问题的解析解和数值解法等;还介绍了较新的非传统方法,如模糊逻辑与模糊推理、神经网络、遗传算法、小波分析、粗糙集及分数阶微积分学等领域。

本书可作为一般读者学习和掌握 MATLAB 语言的教科书,高等学校理工科各类专业的本科生和研究生学习计算机数学语言的教材或参考书,可供科技工作者、教师学习和应用 MATLAB 语言解决实际数学问题时参考,还可作为读者查询某数学问题求解方法的手册。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

MATLAB, Simulink, Symbolic Toolbox, Optimization Toolbox, Statistics Toolbox, Partial Differential Equation Toolbox, Signal Processing Toolbox, Splines Toolbox, Fuzzy Logic Toolbox, Neural Network Toolbox 等为 The MathWorks 公司的注册商标

书 名: 高等应用数学问题的 MATLAB 求解(第三版)

作 者: 薛定宇 陈阳泉 著

出版者: 清华大学出版社(北京清华大学学研大厦,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印刷者: 印刷厂

责任编辑: 王一玲

发行者: 新华书店总店北京发行所

开 本: 787×1092 1/16 印张: 字数: 720 千字

版 次: 2008 年 8 月第 2 版 2008 年 8 月第 1 次印刷

书 号: ISBN 7-302-

印 数: 15001 ~

定 价: 元

第三版前言

本书第二版出版于 2008 年的 8 月,当时最新的版本是 MATLAB R2008a 版,不过那之后一两个月内,MATLAB R2008b 就推出来了,最大的变化就是符号运算引擎从 Maple 变成了 MuPAD,这样,书中有些基于符号运算的内容,尤其是为符号变量类编写的重载函数在新版本下就全部失效了,当时一直建议采用补救与变通的方法。现在,MATLAB 的新版本的使用已经成为主流,在新推出的 MATLAB R2012b (MATLAB 8.0 版) 还出现了许多求解科学运算问题全新的方法和函数结构(如数值积分、延迟微分方程求解等),所以,亟待使用新的途径重新建立起相关问题的求解方法和机制,故此本书侧重于对符号运算方面的内容和科学运算求解的新方法等方面的更新。

很多理工科课程与科学研究都是建立在应用数学各个分支基础上的,所以科学运算问题的求解能力会从某些方面直接影响到科学研究的水平。本书根据理工科学生和学者的需求,全面介绍高等应用数学各个分支典型问题的求解。本书内容看似在介绍数学,但最终目的是期望读者在理解相关数学领域最基本概念的前提下,绕开纯数学和底层烦琐的推导过程,直接由计算机数学语言得出数学问题的解。所以学习本课程将使读者提高数学素养,掌握解决实际科学运算问题的方法,为下一步学习并实践其他课程也打下一个较好的基础。

这里所说的“绕开”纯数学,其基本思想就是用 MATLAB 语言能理解的方式将科学运算的问题描述出来,然后调用现有的函数或自编的 MATLAB 函数,将问题的解直接求出来。例如,对传统意义下看起来难以求解的非线性微分方程问题,可以编写一段代码将微分方程描述出来,以后调用相应的求解函数将其数值解求出来,再用绘图语句将得出的解绘制出来。这样的求解方法和理工科的需求完全一致,将复杂、烦琐的求解中间过程全部推给计算机去求解,这样可以把研究者从繁重的体力工作中解放出来,将精力集中到更高层次的研究中去,取得更多的成果。

本书在新版中增加了很多内容,如体视化绘图方法、区间极限、分段函数、数值积分全新解法、任意矩阵的定义与运算、数值 Laplace 变换与反变换、差分方程解析解方法、多解矩阵方程的数值求解、延迟微分方程求解方法、Mittag-Leffler 函数的数值求解、非零初值分数阶微分方程求解等,另外由于篇幅限制,舍弃了前版的一些内容,如分形问题的求解等。

本书部分新的内容融和了作者和教学团队的几位老师(尤其是东北大学潘峰博士、陈大力博士)在相关课程的教学实践与研究成果,分数阶非零初值微分方程求解部分也有博士生白鹭等人的贡献,在代码验证与课件开发等工作中,研究生郭晓静、王伟楠、刘禄等同学做了大量的工作,在此一并表示感谢。

薛定宇

2013 年 5 月

第二版前言

数学问题是科学研究中经常需要解决的问题。研究者通常将自己研究的问题用数学建模的方法建立起数学模型,然后通过求解数学模型的方法获得所研究问题的解。

本书有两个目标,其一是系统地介绍基于 MATLAB 语言的应用数学问题求解方法,这里涉及的内容涵盖理工科学生本科或研究生期间所接触到的几乎所有数学分支,而深度与广度远远超过相关数学课程的内容。对于非数学专业的读者来说,通过系统地学习本书的方法和思路,求解应用数学问题的能力会有质的提升。本书的另一个目标是作为实用数学问题求解手册供研究者参考。读者在实际研究工作中遇到数学问题的时候,完全可以套用本书的相关内容和语句直接求解,这无疑对读者会有巨大的帮助。

自本书第一版于 2004 年出版以来,作者在教学研究中又有了很多新的想法,同时得到了很多读者的反馈信息,为本书出版新版增添了新的素材。本书第二版在写作风格和格局上沿用第一版成功的套路,仍然根据系统求解数学问题的需要,组织 MATLAB 语言求解的材料,由浅入深地系统介绍数学问题的求解方法,侧重点仍然放在基于 MATLAB 的数学问题求解上。除了 MATLAB 语言版本上的更新外,本版进一步充实、完善了很多第一版的原有内容;另外添加了多重数值积分、差分方程递推求解、分形、线性矩阵不等式、多目标规划、动态规划、矩阵方程与矩阵微分方程求解、切换微分方程与随机微分方程求解、特殊函数、主成分分析、Monte Carlo 方法、径向基神经网络、粒子群优化等诸多新的主题,分数阶微积分学一节融入了作者许多新的研究成果,所以本版的内容更充实、更全面。

本书的英文版“Solving Applied Mathematical Problems with MATLAB”将由 CRC 出版社 2008 年出版,而本书第二版的内容略多于英文版的内容。本书配备的习题参考解答是配合英文版编写的,可以作为本书的习题参考。本书还配备了中、英文版的教学课件可供直接使用。

在本书新版写作过程中仍得到师长、朋友和学生的支持和建议,特别感谢东北大学徐心和教授、新加坡国立大学葛树志教授、首都师范大学赵春娜博士等。在写作过程中和同事潘峰博士、石海滨博士、陈大力博士、胡清河博士、庞哈利教授、张雪峰副教授、王斐博士等的有益讨论也为本版最终成型起了重大作用。另外,学生鄂大志、张玲敏、熊鲲、董雯彬、彭军、罗映等为本书的勘误、代码验证和辅助教学课件开发等起了重要作用,在此表示深深的感谢。

作者

2008 年 7 月

第一版前言

美国 The MathWorks 公司推出的 MATLAB 语言一直是国际科学界应用和影响最广泛的三大计算机数学语言之一。从某种意义上讲,在纯数学以外的领域中,MATLAB 语言有着其他两种计算机数学语言 Mathematica 和 Maple 无法比拟的优势和适用面。在很多领域,MATLAB 语言是科学研究者首选的计算机数学语言。目前关于 MATLAB 语言 and 应用的书籍在国际上数以千计,但从其覆盖面和应用水平来说,往往难以达到日益增长的 MATLAB 语言使用者的要求。国内外出版的著作从涵盖面及深度与广度上缺乏高层次、全面系统介绍高等应用数学问题各个分支的计算机求解的书籍^①。本书试图填补这个空白,在更高层次上系统介绍 MATLAB 语言在高等应用数学各个分支中的应用,包含的应用数学分支为微积分、线性代数、积分变换和复变函数、非线性方程与最优化、常微分方程与偏微分方程、数据插值与函数逼近、概率论与数理统计以及新的非传统方法,如模糊逻辑与模糊推理、神经网络、遗传算法、小波分析、粗糙集及分数阶微积分学等。本书不同于现有的类似于 MATLAB 手册的著作,不是 MATLAB 有什么内容就介绍什么内容,而是根据系统求解数学问题的需要,组织 MATLAB 语言求解的材料,由浅入深地介绍数学问题的求解方法。本书比作者所见识到的国内外任何一部基于 MATLAB 语言的应用数学著作都要全面、系统。

由于工作性质,作者接触过众多非数学专业的本科生、研究生、博士生,感觉大多数学生缺乏对应用数学问题的较全面了解,他们对什么问题能用数学描述,什么样的数学问题能求解不清楚,以致于在学习与研究中走了很多弯路。作者坚信,通过阅读本书可以使读者的数学能力,尤其是数学问题求解能力上一个很大的台阶。即使读者在阅读本书时对有些数学公式理解得不太透彻,只要学习本书的 MATLAB 求解方法,也能容易地求解类似的数学问题。本书的重要目标是让数学基础不深厚的读者同样能轻易地利用计算机解决较高深的应用数学问题。

本书是为东北大学自动化专业新课程“MATLAB 与数学运算”编写的教材,但内容完全脱离了自动化专业的背景,同样适用于其他理工科专业的本科生、研究生教学。本书的大部分内容在东北大学自动化专业本科生以及全校研究生选修课中讲授过,受到普遍欢迎。由于 MATLAB 语言在很多理工科专业的后续课程中有很大作用,建议有条件的学校也开设相应的课程,使学生能认识和掌握该语言,提高应用数学问题求解的水平。为此,本书配有全套的、适用于计算机辅助教学的 CAI 课件材料。

作者从 1988 年开始系统地使用 MATLAB 语言进行程序设计与科学研究,积累了丰富的第一手经验,也了解 MATLAB 语言的最新动态。作者用 MATLAB 语言编写的程序曾作为英国 Rapid Data 软件公司的商品在国际范围内发行,新近编写的几个通用程序在 The MathWorks 公司的网站上可以下载,其中反馈系统分析与设计程序 CtrlLAB 长期高居控

^①由对 The MathWorks 图书网站列出的全部相关书目及目录的分析得出的结论。

制类软件的榜首,已经用于国际上很多高校的实际教学。

多年来,作者一直在试图以最实用的方式将 MATLAB 语言介绍给国内的读者,并在清华大学出版社出版了 4 部有关 MATLAB 语言及其应用方面的著作,受到了国内外广大中文读者的普遍欢迎。其中,1996 年出版的《控制系统计算机辅助设计——MATLAB 语言与应用》一书被公认为国内关于 MATLAB 语言方面书籍中出版最早、影响最广的著作,被国内期刊文章引用近千次。

本书合作者陈阳泉博士现在美国 Utah 州立大学任教,任自组织与先进智能控制中心执行负责人、IEEE 学会高级会员,在先进智能控制、分数阶系统理论及设计、机器人导航与控制等领域均有很深的造诣和学术影响,2002 年与本人合作在清华大学出版社出版的《基于 MATLAB/Simulink 的系统仿真技术与应用》在中文读者中有很影响,并被广为引用。

本书主要介绍目前最新的 MATLAB 7.0 版,即 MATLAB Release 14,但相应的内容对 MATLAB 及相关工具箱的版本依赖程度不高,所以这里介绍的算法函数绝大部分均可以在 MATLAB 6.x 甚至更早期版本下正常运行。同时,考虑到在将来很长一段时间内两个版本可能并存,所以在很多地方也将介绍 MATLAB 6.x 的解法。

本书从使用者的角度出发,并结合作者数十年的实际编程经验和丰富的教学经验,系统地介绍 MATLAB 语言的编程技术及其在科学运算中的应用,书中融合了作者的许多编程思想和第一手材料,内容精心剪裁,相信仍然会受到读者的欢迎。

作者的一些同事、同行和朋友也先后给予作者许多建议和支持,包括东北大学信息学院的徐心和教授、东北大学信息学院院长王福利教授、北京交通大学机电学院院长朱衡君教授等,还有在互联网上交流的众多知名的和不知名的同行与朋友。本书部分内容由博士生张雪峰、潘峰编写,部分辅助程序与模型由硕士生陈大力同学编写,计算机辅助教学材料由硕士生刘莹莹同学开发,在此表示深深的谢意。

本书的出版得到了清华大学出版社欧振旭编辑细心的加工,得到清华大学出版社蔡鸿程主编的关怀,本书的出版还得到了美国 The MathWorks 公司图书计划的支持,在此表示谢意,并特别感谢 Noami Fernandez 女士、Courtney Esposito 先生为作者提供的各种帮助,感谢大连威尔思德科技发展有限公司王龙飞先生为教学网站 MATLAB 大观园提供的各种帮助。

由于作者水平所限,书中的缺点和错误在所难免,欢迎读者批评指教。

谨以此书献给我的妻子杨军和女儿薛杨。在编写本书时花费了大量本该陪伴她们的业余时间,没有她们一如既往的鼓励、支持和理解,本书不可能顺利完成。

薛定宇

2004 年 7 月 6 日于沈阳东北大学

目 录

第 1 章 计算机数学语言概述	1
1.1 数学问题计算机求解概述	1
1.1.1 为什么要学习计算机数学语言	1
1.1.2 数学问题的解析解与数值解	4
1.1.3 数学运算问题软件包发展概述	4
1.1.4 常规计算机语言的局限性	6
1.2 计算机数学语言简介	7
1.2.1 计算机数学语言的出现	7
1.2.2 有代表性的计算机数学语言	8
1.2.3 开放式免费科学运算语言简介	8
1.3 关于本书及相关内容	9
1.3.1 本书框架设计及内容安排	9
1.3.2 MATLAB 语言学习方法与资源	10
1.3.3 本课程与其他相关课程的关系	10
1.4 习 题	11
参考文献	12
第 2 章 MATLAB 语言程序设计基础	13
2.1 MATLAB 程序设计语言基础	14
2.1.1 MATLAB 语言的变量与常量	14
2.1.2 数据结构	14
2.1.3 MATLAB 的基本语句结构	16
2.1.4 冒号表达式与子矩阵提取	17
2.2 基本数学运算	17
2.2.1 矩阵的代数运算	17
2.2.2 矩阵的逻辑运算	19
2.2.3 矩阵的比较运算	20
2.2.4 解析结果的化简与变换	20
2.2.5 基本数论运算	21
2.3 MATLAB 语言的流程结构	23
2.3.1 循环结构	23
2.3.2 条件转移结构	24
2.3.3 开关结构	25

2.3.4	试探结构	26
2.4	函数编写与调试	26
2.4.1	MATLAB 语言函数的基本结构	27
2.4.2	可变输入输出个数的处理	30
2.4.3	匿名函数与 <code>inline()</code> 函数	30
2.4.4	伪代码与代码保密处理代码保密	31
2.5	二维图形绘制	31
2.5.1	二维图形绘制基本语句	31
2.5.2	多纵轴曲线的绘制	34
2.5.3	其他二维图形绘制语句	34
2.5.4	隐函数绘制及应用	35
2.5.5	图形修饰	37
2.5.6	数据文件的读取与存储	38
2.6	三维图形表示	39
2.6.1	三维曲线绘制	39
2.6.2	三维曲面绘制	40
2.6.3	等高线绘制	43
2.6.4	三维隐函数图绘制	45
2.6.5	三维图形视角设置	45
2.6.6	三维曲面的旋转	47
2.7	四维图形绘制	48
2.8	习 题	49
	参考文献	52
第 3 章	微积分问题的计算机求解	53
3.1	极限问题的解析解	53
3.1.1	单变量函数的极限	54
3.1.2	区间函数的极限运算	55
3.1.3	多变量函数的极限	56
3.2	函数导数的解析解	57
3.2.1	函数的导数和高阶导数	57
3.2.2	参数方程的导数	59
3.2.3	多元函数的偏导数	60
3.2.4	隐函数的偏导数	61
3.2.5	多元函数的 Jacobi 矩阵	62
3.2.6	Hess 偏导数矩阵	63
3.3	积分问题的解析解	63
3.3.1	不定积分的推导	64
3.3.2	定积分与无穷积分计算	65

3.3.3	多重积分问题的 MATLAB 求解	65
3.4	函数的级数展开与级数求和问题求解	66
3.4.1	Taylor 幂级数展开	66
3.4.2	Fourier 级数展开	69
3.4.3	级数求和的计算	72
3.4.4	序列求积问题	73
3.5	曲线积分与曲面积分的计算	74
3.5.1	曲线积分及 MATLAB 求解	74
3.5.2	曲面积分与 MATLAB 语言求解	76
3.6	数值微分问题	78
3.6.1	数值微分算法	78
3.6.2	中心差分方法及其 MATLAB 实现	79
3.6.3	二元函数的梯度计算	80
3.7	数值积分问题	81
3.7.1	由给定数据进行梯形求积	82
3.7.2	单变量数值积分问题求解	84
3.7.3	广义数值积分问题求解	86
3.7.4	积分函数的数值求解	87
3.7.5	双重积分问题的数值解	88
3.7.6	三重定积分的数值求解	91
3.7.7	多重积分数值求解	91
3.8	习 题	92
	参考文献	96
第 4 章	线性代数问题的计算机求解	97
4.1	特殊矩阵的输入	97
4.1.1	数值矩阵的输入	98
4.1.2	符号矩阵的输入	102
4.1.3	稀疏矩阵的输入	103
4.2	矩阵基本分析	103
4.2.1	矩阵基本概念与性质	103
4.2.2	逆矩阵与广义逆矩阵	110
4.2.3	矩阵的特征值问题	113
4.3	矩阵的基本变换与分解	116
4.3.1	矩阵的相似变换与正交矩阵	116
4.3.2	矩阵的三角分解和 Cholesky 分解	117
4.3.3	矩阵的相伴变换、对角变换和 Jordan 变换	120
4.3.4	矩阵的奇异值分解	124
4.4	矩阵方程的计算机求解	126

4.4.1	线性方程组的计算机求解	126
4.4.2	Lyapunov 方程的计算机求解	129
4.4.3	Sylvester 方程的计算机求解	132
4.4.4	Riccati 方程的计算机求解	133
4.4.5	一类线性不等式的求解	134
4.5	非线性运算与矩阵函数求值	135
4.5.1	面向矩阵元素的非线性运算	135
4.5.2	矩阵函数求值	136
4.5.3	一般矩阵函数的运算	139
4.6	习 题	141
	参考文献	144
第 5 章	积分变换与复变函数问题的计算机求解	145
5.1	Laplace 变换及其反变换	145
5.1.1	Laplace 变换及反变换的定义与性质	145
5.1.2	Laplace 变换的计算机求解	146
5.1.3	Laplace 变换问题的数值求解	149
5.2	Fourier 变换及其反变换	151
5.2.1	Fourier 变换及反变换定义与性质	151
5.2.2	Fourier 变换的计算机求解	152
5.2.3	Fourier 正弦和余弦变换	153
5.2.4	离散 Fourier 正弦、余弦变换	155
5.2.5	快速 Fourier 变换	156
5.3	其他积分变换问题及求解	157
5.3.1	Mellin 变换	157
5.3.2	Hankel 变换及求解	158
5.4	z 变换及其反变换	159
5.4.1	z 变换及反变换定义与性质	159
5.4.2	z 变换的计算机求解	160
5.4.3	双边 z 变换	161
5.4.4	有理函数 z 反变换的数值求解	161
5.5	复变函数问题的计算机求解	162
5.5.1	复数矩阵及其变换	162
5.5.2	复变函数的映射	163
5.5.3	Riemann 面绘制	163
5.5.4	留数的概念与计算	165
5.5.5	有理函数的部分分式展开	167
5.5.6	基于部分分式展开的 Laplace 反变换	170
5.5.7	封闭曲线积分问题计算	171

5.6	差分方程的求解	173
5.6.1	一般差分方程的解析求解方法	173
5.6.2	线性时变差分方程的数值解法	175
5.6.3	线性时不变差分方程的解法	176
5.6.4	一般非线性差分方程的数值求解方法	177
5.7	习 题	177
	参考文献	180
第 6 章	代数方程与最优化问题的计算机求解	181
6.1	代数方程的求解	181
6.1.1	代数方程的图解法	181
6.1.2	多项式型方程的准解析解法	183
6.1.3	一般非线性方程数值解	186
6.1.4	求解多解方程的全部解	187
6.2	无约束最优化问题求解	190
6.2.1	解析解法和图解法	191
6.2.2	基于 MATLAB 的数值解法	192
6.2.3	全局最优解与局部最优解	193
6.2.4	利用梯度求解最优化问题	195
6.2.5	带有变量边界约束的最优化问题求解	196
6.3	有约束最优化问题的计算机求解	197
6.3.1	约束条件与可行解区域	197
6.3.2	线性规划问题的计算机求解	198
6.3.3	二次型规划的求解	201
6.3.4	一般非线性规划问题的求解	201
6.4	混合整数规划问题的计算机求解	205
6.4.1	整数线性规划问题的求解	205
6.4.2	整数规划问题的穷举方法	206
6.4.3	一般非线性整数规划问题与求解	207
6.4.4	0-1 规划问题求解	210
6.5	线性矩阵不等式问题求解	212
6.5.1	线性矩阵不等式的一般描述	212
6.5.2	Lyapunov 不等式	213
6.5.3	线性矩阵不等式问题分类	215
6.5.4	线性矩阵不等式问题的 MATLAB 求解	216
6.5.5	基于 YALMIP 工具箱的最优化求解方法	217
6.6	多目标优化问题求解	219
6.6.1	多目标优化模型	219
6.6.2	无约束多目标函数的最小二乘求解	220

6.6.3	多目标问题转换为单目标问题求解	220
6.6.4	多目标优化问题的 Pareto 解集	223
6.6.5	极小极大问题求解	224
6.6.6	目标规划问题求解	226
6.7	动态规划及其在路径规划中的应用	227
6.7.1	图的矩阵表示方法	227
6.7.2	有向图的路径寻优	227
6.7.3	无向图的路径最优搜索	230
6.7.4	绝对坐标节点的最优路径规划算法与应用	231
6.8	习 题	231
	参考文献	236
第 7 章	微分方程问题的计算机求解	237
7.1	常系数线性微分方程的解析解方法	237
7.1.1	线性常系数微分方程解析解的数学描述	237
7.1.2	微分方程的解析解方法	238
7.1.3	线性状态空间方程的解析解	241
7.1.4	特殊非线性微分方程的解析解	241
7.2	微分方程问题的数值解法	242
7.2.1	微分方程问题算法概述	242
7.2.2	四阶定步长 Runge-Kutta 算法及 MATLAB 实现	244
7.2.3	一阶微分方程组的数值解	245
7.2.4	微分方程数值解的验证	249
7.3	微分方程转换	249
7.3.1	单个高阶常微分方程处理方法	249
7.3.2	高阶常微分方程组的变换方法	250
7.3.3	矩阵微分方程的变换与求解方法	254
7.4	特殊微分方程的数值解	257
7.4.1	刚性微分方程的求解	257
7.4.2	隐式微分方程求解	259
7.4.3	微分代数方程的求解	262
7.4.4	切换微分方程的求解	264
7.4.5	随机线性微分方程的求解	265
7.5	延迟微分方程求解	268
7.5.1	典型延迟微分方程的数值求解	268
7.5.2	变时间延迟微分方程的求解	270
7.5.3	中立型延迟微分方程的求解	272
7.6	边值问题的计算机求解	273
7.6.1	线性方程边值问题的打靶算法	274

7.6.2	非线性方程边值问题的打靶算法	276
7.6.3	一般边值微分方程的求解方法	277
7.7	偏微分方程求解入门	280
7.7.1	偏微分方程组求解	280
7.7.2	二阶偏微分方程的数学描述	281
7.7.3	偏微分方程的求解界面应用举例	283
7.8	基于 Simulink 的微分方程框图求解	289
7.8.1	Simulink 简介	290
7.8.2	Simulink 相关模块	290
7.8.3	微分方程的 Simulink 建模与求解	292
7.9	习 题	296
	参考文献	300
第 8 章	数据插值、函数逼近问题的计算机求解	301
8.1	插值与数据拟合	301
8.1.1	一维数据的插值问题	301
8.1.2	已知样本点的定积分计算	305
8.1.3	二维网格数据的插值问题	306
8.1.4	二维散点分布数据的插值问题	308
8.1.5	高维插值问题	311
8.1.6	基于样本数据点的离散最优化问题求解	312
8.2	样条插值与数值微积分问题求解	313
8.2.1	样条插值的 MATLAB 表示	313
8.2.2	基于样条插值的数值微积分运算	316
8.3	由已知数据拟合数学模型	318
8.3.1	多项式拟合	319
8.3.2	函数线性组合的曲线拟合方法	320
8.3.3	最小二乘曲线拟合	322
8.3.4	多变量函数的最小二乘函数拟合	324
8.4	已知函数的有理式逼近方法	325
8.4.1	给定函数的连分式展开及基于连分式的有理近似	325
8.4.2	有理式拟合 — Padé 近似	327
8.5	特殊函数及曲线绘制	330
8.5.1	Γ -函数	330
8.5.2	β -函数	331
8.5.3	Bessel 函数	331
8.5.4	Legendre 函数	333
8.5.5	Mittag-Leffler 函数	334
8.6	信号分析与数字信号处理基础	337

8.6.1	信号的相关分析	337
8.6.2	滤波技术与滤波器设计	339
8.7	习 题	342
	参考文献	344
第 9 章	概率论与数理统计问题的计算机求解	345
9.1	概率分布与伪随机数生成	345
9.1.1	概率密度函数与分布函数概述	345
9.1.2	常见分布的概率密度函数与分布函数	346
9.1.3	概率问题的求解	352
9.1.4	随机数与伪随机数	353
9.2	统计量分析	355
9.2.1	随机变量的均值与方差	355
9.2.2	随机变量的矩	356
9.2.3	多变量随机数的协方差分析	357
9.2.4	多变量正态分布的联合概率密度函数及分布函数	358
9.2.5	基于 Monte Carlo 法的数学问题求解	359
9.3	数理统计分析方法及计算机实现	360
9.3.1	参数估计与区间估计	360
9.3.2	多元线性回归与区间估计	362
9.3.3	非线性函数的最小二乘参数估计与区间估计	363
9.4	统计假设检验	366
9.4.1	统计假设检验的概念及步骤	366
9.4.2	假设检验问题求解	367
9.5	方差分析与主成分分析	369
9.5.1	方差分析	369
9.5.2	主成分分析	373
9.6	习 题	375
	参考文献	378
第 10 章	数学问题的非传统解法	379
10.1	集合论、模糊集与模糊推理	379
10.1.1	经典可枚举集合论问题及 MATLAB 求解	379
10.1.2	模糊集合与隶属度函数	381
10.1.3	模糊推理系统及其 MATLAB 求解	385
10.2	粗糙集理论与应用	389
10.2.1	粗糙集理论简介	389
10.2.2	粗糙集的基本概念	389
10.2.3	信息决策系统	390

10.2.4 粗糙集数据处理问题的 MATLAB 求解	392
10.2.5 粗糙集约简的 MATLAB 程序界面	394
10.3 人工神经网络及其在数据拟合中的应用	395
10.3.1 神经网络基础知识	395
10.3.2 径向基网络结构与应用	403
10.3.3 神经网络界面	405
10.4 进化算法及其在最优化问题中的应用	408
10.4.1 遗传算法的基本概念及 MATLAB 实现	408
10.4.2 遗传算法在求解最优化问题中的应用举例	409
10.4.3 遗传算法在有约束最优化问题中的应用	414
10.4.4 粒子群优化算法与求解	416
10.4.5 其他全局优化算法	418
10.4.6 求取精确的全局最优解	419
10.4.7 基于遗传算法的混合整数规划求解	419
10.5 小波变换及其在数据处理中的应用	420
10.5.1 小波变换及基小波波形	420
10.5.2 小波变换技术在信号处理中的应用	424
10.5.3 小波问题的程序界面	427
10.6 分数阶微积分学问题求解及应用	428
10.6.1 分数阶微积分的定义	429
10.6.2 不同分数阶微积分定义的关系与性质	430
10.6.3 分数阶微积分的计算方法	431
10.6.4 分数阶线性微分方程的求解方法	437
10.6.5 基于框图的非线性分数阶微分方程近似解法	441
10.6.6 分数阶传递函数模型与分析	446
10.7 习 题	450
参考文献	452
MATLAB 函数名索引	455
术语索引	461

第1章 计算机数学语言概述

1.1 数学问题计算机求解概述

数学问题是科学研究中经常会遇到的问题。研究者通常将自己研究的问题用数学建模的方法建立起数学模型,然后通过求解数学模型获得所研究问题的解。建立数学模型需要所研究领域的专业知识,而有了数学模型则可以采用本书介绍的通用数值方法或解析方法去求解。本章将首先对计算机数学语言给出简单介绍,通过实例介绍为什么需要学习计算机数学语言,然后介绍计算机数学语言和数学工具发展简况。本章最后将介绍本书的框架,列出涉及的数学分支并进行概述。

1.1.1 为什么要学习计算机数学语言

求解数学问题时手工推导当然是有用的,但并不是所有的问题都是能手工推导的,故需要由计算机来完成相应的任务。用计算机的方式也有两种,其一是用成型的数值分析算法、数值软件包与手工编程的方法相结合的求解方法,其二是采用国际上有影响的专门计算机语言来求解问题,这类语言包括 MATLAB、Mathematica、Maple 等,本书统一称之为**计算机数学语言**。顾名思义,用数值方法只能求解数值计算的问题,至于像公式推导等数学问题,例如求解 $x^3 + ax + c = d$ 方程的解,在 a, c, d 不是给定数值时,数值分析的方式是没有用的,必须使用计算机数学语言来求解。

本书将涉及的问题求解方法称为“数学运算”,以区别于传统意义下的“数学计算”,因为后者往往对应于数学问题的数值求解方法。本书将介绍的内容还尽可能地包括解析求解方法,如果解析解不存在则将介绍数值解方法。

在系统介绍本书的内容之前,先介绍几个例子,读者可以思考其中提出的问题,从中体会学习本书的必要性。相应的 MATLAB 语句后面还将详细介绍。

例 1-1 大学的高等数学课程学习了微分与积分的概念和数学推导方法,但实际应用中可能遇到高阶导数的问题。已知 $f(x) = \sin x / (x^2 + 4x + 3)$ 这样的简单函数,如何求解出 $d^4 f(x) / dx^4$? 当然,用手工推导是可行的,由高等数学的知识先得出 $df(t)/dx$,对结果求导得出二阶导数,对结果再求导得出三阶导数,继续进一步求导就能求出所需的 $d^4 f(x) / dx^4$,重复此方法还能求出更高阶的导数。这个过程比较机械,适合用计算机实现,用现有的计算机数学语言可以由一行语句求解问题

```
>> syms x; f=sin(x)/(x^2+4*x+3); y=diff(f,x,4)
```

上述语句得出的结果为

$$\frac{d^4 f(t)}{dx^4} = \frac{\sin x}{x^2 + 4x + 3} + 4 \frac{(2x + 4) \cos x}{(x^2 + 4x + 3)^2} - 12 \frac{(2x + 4)^2 \sin x}{(x^2 + 4x + 3)^3} + 12 \frac{\sin x}{(x^2 + 4x + 3)^2} - 24 \frac{(2x + 4)^3 \cos x}{(x^2 + 4x + 3)^4}$$

$$+48 \frac{(2x+4) \cos x}{(x^2+4x+3)^3} + 24 \frac{(2x+4)^4 \sin x}{(x^2+4x+3)^5} - 72 \frac{(2x+4)^2 \sin x}{(x^2+4x+3)^4} + 24 \frac{\sin x}{(x^2+4x+3)^3}$$

显然,若依赖手工推导,得出这样的结果需要很繁杂、细致的工作,稍有不慎就可能得出错误的结果。因此,手工推导得出结果的可信度有时是值得怀疑的。如果能采用计算机代替手工推导则会既省力,又增加可信度,故需要计算机数学语言来解决这样的问题。实践表明,利用著名的 MATLAB 语言,在 1s 内^①就可以精确地求出 $d^{100}f(x)/dx^{100}$ 。

例 1-2 在许多学科的实际应用中经常要求出多项式方程的根。著名的 Abel-Ruffini 定理已经有了定论,5 次或以上的多项式方程没有通用的解析求解方法,但在实际应用中经常要求解高次代数方程的根,故可以采用数值方法求解,如使用林士谔-Bairstrow 算法,这是数值分析中最常见的方法。考虑下面的多项式方程

$$s^6 + 9s^5 + \frac{135}{4}s^4 + \frac{135}{2}s^3 + \frac{1215}{16}s^2 + \frac{729}{16}s + \frac{729}{64} = 0$$

用林士谔-Bairstrow 算法得出的结果是

$$s_{1,2} = -1.5056 \pm j0.0032, \quad s_{3,4} = -1.5000 \pm j0.0065, \quad s_{5,6} = -1.4944 \pm j0.0032$$

将 s_1 代入原始方程,则可容易计算出方程左侧为 $-8.7041 \times 10^{-14} - j1.8353 \times 10^{-15}$ 。虽然误差不大,毕竟对这类问题来说,数值方法可能导致错误的结论。采用计算机数学语言能得出更精确的结果,即所有的根均为 -1.5 。下面列出的是本例使用的 MATLAB 求解语句

```
>> p=[1 9 135/4 135/2 1215/16 729/16 729/64]; roots(p) % 数值解
      p1=poly2sym(p); solve(p1) % 解析解
```

例 1-3 线性代数课程中介绍了求解矩阵行列式的方法,例如用代数余子式的方法可以将一个 n 阶矩阵的行列式问题化简成 n 个 $n-1$ 阶行列式问题,而 $n-1$ 阶的又可以化简为 $n-2$ 阶的问题,这样用递归的方法可以最终化简成一阶矩阵的行列式求解问题,而该问题是有解析解的,就是该一阶矩阵本身,所以数学家可以得出结论,任意阶矩阵的行列式都可以直接求解出解析解。

事实上,这样的结论忽略了复杂度或可行性问题,这样的算法计算量很大,高达 $(n-1)(n+1)!+n$,例如 $n=25$ 时,运算次数为 3.7227×10^{26} ,相当于在每秒 20 亿亿次 (2×10^{17}) 的巨型机 (2012 年世界上最快的巨型计算机) 59 年的计算量,所以虽然用代数余子式的方法可以求解,但求解是不现实的。其实在某些领域中甚至需要求解成百上千阶的问题,所以用代数余子式的方法是不可行的。

数值分析中提供了求解行列式问题的各种算法,但传统的方法对某些矩阵有时会得出错误的结果,特别是接近奇异的矩阵。考虑 Hilbert 矩阵

$$\mathbf{H} = \begin{bmatrix} 1 & 1/2 & 1/3 & \cdots & 1/n \\ 1/2 & 1/3 & 1/4 & \cdots & 1/(n+1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/(n+1) & 1/(n+2) & \cdots & 1/(2n-1) \end{bmatrix} \quad (1-1-1)$$

并假设 $n=20$,用传统的数值分析很容易得出 $\det(\mathbf{H})=0$ 的错误结论。事实上,用计算机数学语言 MATLAB 很容易在 0.4s 内得出该行列式的精确解为

^①本书中涉及的求解时间均指作者使用的 MacBook Air i5 的笔记本计算机上测出的,且 CPU 的不同运行状态下测出的时间也有差异。

225 位,因排版限制省略了中间的数字

例 1-4 考虑著名的非线性方程——Van der Pol 方程 $y'' + \mu(y^2 - 1)y' + y = 0$, 当 μ 很大时, 例如 $\mu = 1000$, 传统的数值分析方法求解可能有问题, 需要用专用的刚性方程求解算法进行求解, 而不用数值分析类课程中介绍的 Runge-Kutta 算法求解。利用 MATLAB 语言, 只需下面两行语句求解该方程, 并用图形显示方程的结果。

如果一阶微分方程可以写成 $y'(t) = -0.1y(t) + 0.2y(t-30)/[1+y^{10}(t-30)]$, 这样的方程称为延迟微分方程, 一般的数值分析教材和软件包中均不提供这种方程的数值解法, 所以只能采用计算机数学语言, 如 MATLAB 中的延迟微分方程求解函数 `dde23()` 或图形化建模仿真工具 Simulink 来求解这样的问题。在本书后面相应的内容中将介绍此方程的解法。

$$\begin{array}{ll} \min & (-2x_1 - x_2 - 4x_3 - 3x_4 - x_5) \\ \text{s.t.} & \begin{cases} 2x_2 + x_3 + 4x_4 + 2x_5 \leq 54 \\ 3x_1 + 4x_2 + 5x_3 - x_4 - x_5 \leq 62 \\ x_1, x_2 \geq 0, x_3 \geq 3.32, x_4 \geq 0.678, x_5 \geq 2.57 \end{cases} \end{array}$$

```
>> P.f=[-2 -1 -4 -3 -1]; P.Aineq=[0 2 1 4 2; 3 4 5 -1 -1];
    P.Bineq=[54 62]; P.lb=[0;0;3.32;0.678;2.57]; P.solver='linprog';
    P.options=optimset; x=linprog(P)
```

这样的求解借助数值分析或最优化方法等课程介绍的数值算法可以容易地实现。但如果再添加约束,例如需要得出该最优化问题的整数解,原来的问题就变成了整数规划问题。很少有相关书籍、软件能直接求解这样的问题。而利用计算机数学语言可以求出该整数规划问题的解为 $x_1 = 19, x_2 = 0, x_3 = 4, x_4 = 10, x_5 = 5$ 。

例 1-7 现代科学技术在其发展过程中,催生了若干新的数学分支,如模糊集合与粗糙集合、人工神经网络等,如果不借助于计算机工具,要想利用其中任何一个分支去解决实际问题都是个耗时并困难的任务。因为首先要了解相关领域的来龙去脉,弄清算法并将算法用计算机语言正确地实现。然而,利用这些新分支的数学工具解决某些特定的数学问题却是比较容易的,因为可以借助前人已经开发好的工具和框架。

很多专门的课程,如电路、电子技术、电力电子技术、电机与拖动、自动控制原理等,在

介绍原理与方法时一般采用简单的例子,刻意回避高阶的或复杂的例子。究其原因,是当时缺少高水平计算机数学语言甚至是数值分析技术的支持,所以在这些课程中很多方法不一定适合于复杂的问题求解。在实际研究中遇到稍复杂一点的问题时,只靠手工推导的方法是得不出精确结果的,所以需要特殊的专业软件或语言来解决问题,而计算机数学语言,如 MATLAB 语言,通常可以较好地解决相关问题。

从上面的例子可以看出,解决数学问题用手工推导的方法虽然有时可行,但对很多复杂问题不现实或不可靠,用传统数值分析课程甚至成型的软件包(如在国际上享有盛誉的 *Numerical Recipes* [1])得出的结果有时也是错误的,故需要学习计算机数学语言,以更好地解决以后学习和研究中遇到的问题。

1.1.2 数学问题的解析解与数值解

现代科学与工程的发展离不开数学。数学家们感兴趣的问题和其他科学家、工程技术人员所关注的问题是不同。数学家往往对数学问题的解析解,或称闭式解(closed-form solution)和解的存在性、唯一性的严格证明感兴趣,而工程技术人员一般对如何求出数学问题的解更关心。换句话说,能用某种方法获得问题的解则是工程技术人员更关心的问题。而获得这样解的最直接方法就是通过数值解法技术。

数学问题解析解不存在的情况是非常常见的。例如,定积分 $\frac{2}{\sqrt{\pi}} \int_0^a e^{-x^2} dx$ 在上限为无穷时就没有解析解。数学家可以用新的函数 $\text{erf}(a)$ 去定义这样的解,但解的值到底多大却不是一目了然的。所以,在这样的情况下,要想获得积分的值,就必须采用数值解技术。

再例如,圆周率 π 的值本身就没有解析解,中国古代的数学家、天文学家祖冲之(429–500)早在公元 480 年就算定了该值在 3.1415926 和 3.1415927 之间。在一般科学与工程应用中,取这样的值就能保证较高的精度,而对于粗略估算来说,使用公元前 250 年(?)阿基米德(前 287–前 212)的 3.1418 也未尝不可,而没有必要非去追求不存在的解析解不可。所以在这样的问题上,数值解法的优势就显示出来了。

数学问题的数值解法已经成功地应用于各个领域。例如,在力学领域,常用有限元法求解偏微分方程;在航空、航天与自动控制领域,经常用到数值线性代数与常微分方程的数值解法等解决实际问题;在工程与非工程系统的计算机仿真中,核心问题的求解也需要用到各种差分方程、常微分方程的数值解法;在高科技的数字信号处理领域,离散的快速 Fourier 变换(FFT)已经成为其不可或缺的工具。在科学工程研究中能掌握一个或多个实用的计算工具,无疑会为研究者提供解决实际问题的强有力手段。

1.1.3 数学运算问题软件包发展概述

数字计算机的出现给数值计算技术的研究注入了新的活力。在数值计算技术的早期发展中,出现了一些著名的数学软件包,如美国的基于特征值的软件包 EISPACK [2,3]和线性代数软件包 LINPACK [4],英国牛津数值算法研究组(Numerical Algorithm Group, NAG)开发的 NAG 软件包 [5]及享有盛誉的著作 *Numerical Recipes* [1]中给出的程序集等,这些都是在国际上广泛流行的、有着较高声望的软件包。

美国的 EISPACK 和 LINPACK 都是基于矩阵特征值和奇异值解决线性代数问题的专用软件包。限于当时的计算机发展状况,这些软件包大都是由 FORTRAN 语言编写的源程序组成的。例如,若想求出 N 阶实矩阵 A 的全部特征值(用 W_R 、 W_I 数组分别表示其实部和虚部)和对应的特征向量矩阵 Z ,则 EISPACK 软件包给出的子程序建议调用路径为

```
CALL BALANC(NM,N,A,IS1,IS2,FV1)
CALL ELMHES(NM,N,IS1,IS2,A,IV1)
CALL ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
CALL HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
IF (IERR.EQ.0) GOTO 99999
CALL BALBAK(NM,N,IS1,IS2,FV1,N,Z)
```

由上面的叙述可以看出,要求取矩阵的特征值和特征向量,首先要给一些数组和变量依据 EISPACK 的格式作出定义和赋值,并编写出主程序,再经过编译和连接过程,形成可执行文件,最后才能得出所需的结果。

英国的 NAG 软件包和美国学者的 *Numerical Recipes* 工具包则包括了各种各样数学问题的数值解法,二者中 NAG 的功能尤其强大。NAG 的子程序都是以字母加数字编号的形式命名的,非专业人员很难找到适合自己问题的子程序,更不用说能保证以正确的格式去调用这些子程序了。这些程序包使用起来极其复杂,谁也不能保证不发生错误,NAG 数百页的使用手册就有十几本之多!

Numerical Recipes 一书中给出的一系列算法语言源程序也是一个在国际上广泛应用的软件包。该书中的子程序有 C、FORTRAN 和 Pascal 等版本,适合于科学研究者和工程技术人员直接应用。该书的程序包由 200 多个高效、实用的子程序构成,这些子程序一般有良好的数值特性,比较可靠,为各国的研究者所信赖。

具有 FORTRAN 和 C 等高级计算机语言知识的读者可能已经注意到,如果用它们去进行程序设计,尤其当涉及矩阵运算或画图时,编程会很麻烦。例如,若想求解一个线性代数方程,用户得首先去编写一个主程序,然后编写一个子程序去读入各个矩阵的元素,之后再编写一个子程序,求解相应的方程(如使用 Gauss 消去法),最后输出计算结果。如果选择的计算子程序不是很可靠,则所得的计算结果往往会出现问题。如果没有标准的子程序可以调用,则用户往往要将自己编好的子程序逐条地输入计算机,然后进行调试,最后进行计算。这样一个简单的问题往往需要用户编写 100 条左右的源程序,输入与调试程序也是很费事的,并无法保证所输入的程序完全可靠。求解线性方程组这样一个简单的功能需要 100 条源程序,其他复杂的功能往往要求有更多条语句,如采用双步 QR 法求取矩阵特征值的子程序则需要 500 多条源程序,其中任何一条语句有毛病,甚至调用不当(如数组维数不匹配)都可能导致错误结果的出现。

尽管如此,数学软件包仍在继续发展,其发展方向是采用国际上最先进的数值算法,提供更高效、更稳定、更快速、更可靠的数学软件包。例如,在线性代数计算领域,全新的 LaPACK 已经成为当前最有影响的软件包,但它们的目的是似乎已经不再为一般用户提供解决问题的方法,而是为数学软件提供底层的支持。新版的 MATLAB 语言以及自由软件 Scilab 等著名的计算机数学语言都已经放弃了一直使用的 LINPACK 和 EISPACK,而采用

LaPACK 为其底层支持软件包。

一些数学的专门分支也出现了相关的数学程序库,支持 FORTRAN、C++ 等语言直接调用与编程。在互联网上同样有大量的 MATLAB 语言和其他计算机数学语言的数学工具箱,所以遇到典型问题的数学求解时,可以直接利用相关的工具箱来求解,因为其中大部分工具箱毕竟还是在相应领域有影响的专家编写的,得出的结果比外行自己查阅书籍、论文编写的可信度要高得多。

1.1.4 常规计算机语言的局限性

人们有时习惯用其他计算机语言,如 C 和 FORTRAN,解决实际问题。毋庸置疑,这些计算机语言在数学与工程问题求解中起过很大的作用,而且它们曾经是实现 MATLAB 这类高级语言的底层计算机语言。然而,对于一般科学研究者来说,利用 C 这类语言去求解数学问题是不够的。首先,一般程序设计者无法编写出符号运算和公式推导类程序,只能编写数值计算程序;其次,常规数值算法往往不是求解数学问题的最好方法;另外,除了上述局限性外,采用底层计算机语言编程,由于程序冗长难以验证,即使得出结果也不敢相信与依赖该结果。所以应该采用更可靠、更简洁的专门计算机数学语言来进行科学研究,因为这样可以将研究者从烦琐的底层编程中解放出来,更好地把握要求解的问题,避免“只见树木、不见森林”的现象,这无疑受到更多研究者认可的方式。

例 1-8 已知 Fibonacci 数列的前两个元素为 $a_1=a_2=1$,随后的元素可以由 $a_k = a_{k-1} + a_{k-2}$, $k = 3, 4, \dots$ 递推地计算出来。试用计算机列出该数列的前 100 项。

解 C 语言在编写程序之前需要首先给变量选择数据类型,如此问题需要的是整数,所以很自然地选择 int 或 long 来表示数列的元素,若选择数据类型为 int,则可以编写出如下 C 程序

```
main()
{ int a1, a2, a3, i;
  a1=1; a2=1; printf("%d %d ",a1,a2);
  for (i=3; i<=100; i++)
  { a3=a1+a2; printf("%d ",a3); a1=a2; a2=a3;
  }}
```

只用了上面几条语句,问题就看似轻易地被解决了。然而该程序是错误的!运行该程序会发现,该数列打印到第 24 项突然会出现负数,而再显示下几项会发现时正时负。显然,上面的程序出了问题。问题出在 int 整型变量的选择上,因为该数据类型能表示数值的范围为 $(-32767, 32767)$,超出此范围则会导致错误的结果。即使采用 long 整型数据定义,也只能保留 31 位二进制数值,即保留 9 位十进制有效数字,超过这个数仍然返回负值。可见,采用 C 语言,如果某些细节考虑不到,则可能得出完全错误的结论。故可以说 C 这类语言得出的结果有时不大令人信服。用 MATLAB 语言则不必考虑这些烦琐的问题。

```
>> a=[1 1]; for i=3:100, a(i)=a(i-1)+a(i-2); end; a
```

另外,由于 long 整型数据只能保持 9 位有效数字,而 double 型只能保留 15 位有效数字,如果得出的结果超出此范围,则精度将存在局限性。采用 MATLAB 的符号运算则可以避免这类问题,只

需将第一个语句修改成 $a=\text{sym}([1,1])$ 就可以得出 a_{100} 的值为 354224848179261915075, 甚至在 15 分钟内得出 a_{5000} 的全部 1044 位有效数字, 这个结果是采用任何数值计算语言无法得出的。

例 1-9 试编写出两个矩阵 A 和 B 相乘的 C 语言通用程序。

解 如果 A 为 $n \times p$ 矩阵, B 为 $p \times m$ 矩阵, 则由线性代数理论, 可以得出 C 矩阵, 其元素为
$$c_{ij} = \sum_{k=1}^p a_{ik}b_{kj}, i=1, \dots, n, j=1, \dots, m.$$
 分析上面的算法, 容易编写出 C 语言程序, 其核心部分为三重循环结构

```
for (i=0; i<n; i++){for (j=0; j<m; j++){
    c[i][j]=0; for (k=0; k<p; k++) c[i][j]+=a[i][k]*b[k][j]; }}
```

看起来这样一个通用程序通过这几条语句就解决了。事实不然, 这个程序有个致命的漏洞, 就是没考虑两个矩阵是不是可乘。通常我们知道, 两个矩阵可乘, 则 A 矩阵的列数应该等于 B 矩阵的行数, 所以很自然地想到应该加一个判定语句

if A 的列数不等于 B 的行数, 给出错误信息

其实这样的判定也有漏洞, 因为若 A 或 B 为标量, 则 A 和 B 无条件可乘, 而增加上面的 if 语句反而会给出错误信息。这样在原来的基础上还应该增加判定 A 或 B 是否为标量的语句。

其实即使考虑了上面所有的内容, 程序还不是通用的程序, 因为并未考虑矩阵为复数矩阵的情况。这也需要特殊的语句处理。

从这个例子可见, 用 C 这类语言处理某类标准问题时需要特别细心, 否则难免会有漏洞, 致使程序出现错误, 或其通用性受到限制, 甚至可能得出有误导性的结果。在 MATLAB 语言中则没有必要考虑这样的琐碎问题, 因为 A 和 B 矩阵的积由 $A*B$ 直接求取, 若可乘则得出正确结果, 如不可乘则给出出现问题的原因。

当然, 在实时性与实时控制等领域, C 语言也有它的优势。虽然 MATLAB 的代码也可以自动翻译成 C 语言程序, 但这不是本书叙述的范围。

1.2 计算机数学语言简介

1.2.1 计算机数学语言的出现

1984 年正式推出的 MATLAB 语言为数学问题的计算机求解, 特别是控制系统的仿真和 CAD 发展起到了巨大的推动作用。1978 年美国 New Mexico 大学计算机科学系的主任 Cleve Moler 教授认为用当时最先进的 EISPACK 和 LINPACK 软件包求解线性代数问题过程过于烦琐, 所以构思一个名为 MATLAB (MATrix LABoratory, 即矩阵实验室) 的交互式计算机语言。该语言一开始为免费版本, 1984 年 The MathWorks 公司成立, 并推出了 1.0 版。该语言的出现正赶上控制界基于状态空间的控制理论蓬勃发展的阶段, 所以很快就引起了控制界学者的关注, 出现了用 MATLAB 语言编写的控制系统工具箱, 在控制界产生了巨大的影响, 成为控制界的标准计算机语言。后来由于控制界及相关领域提出的各种各样要求, MATLAB 语言得到了持续发展, 使得其功能越来越强大。可以说, MATLAB 语言是由计算数学专家首创的, 但是由控制界学者“捧红”的新型计算机语言。目前大部分工具箱

都是面向控制和相关学科的,但随着 MATLAB 语言的不断发展,目前也在其他领域开始使用。稍后出现的 Mathematica 及 Maple 等语言也是当前应用广泛的计算机数学语言。

1.2.2 有代表性的计算机数学语言

目前在国际上有三种计算机数学语言最有影响: The MathWorks 公司的 MATLAB 语言、Wolfram Research 公司的 Mathematica 语言和 Waterloo Maplesoft 公司的 Maple 语言。这三种语言各有特色,其中 MATLAB 长于数值运算,其程序结构类似于其他计算机语言,因而编程很方便。Mathematica 和 Maple 有强大的解析运算和数学公式推导、定理证明的功能,相应的数值计算能力比 MATLAB 要弱,这两种语言更适合于纯数学领域的计算机求解。此外,德国 MuPAD 也是较好的计算机数学语言。

和 Mathematica 及 Maple 相比, MATLAB 语言的数值运算功能是很出色的。除此之外,更有一个另两种语言不可替代的优势,就是 MATLAB 语言对各种各样领域均有专业领域专家编写的工具箱,可以高效、可靠地解决各种各样的问题。早期 MATLAB 版本的符号运算工具箱利用 Maple 作为其符号运算引擎,能直接求解常用的符号运算问题。另外, MATLAB 提供了对 Maple 全部函数的接口,无需安装 Maple 就可以调用 Maple 所有的数学函数,这大大地增强了 MATLAB 的符号运算功能,使之在这方面的功能也不逊色于 Mathematica 和 Maple。新版本的 MATLAB 符号运算工具箱采用 MuPAD 为其符号运算引擎,有些符号运算的能力较以前版本有改善,也有很多功能不如早期版本,本书将针对具体问题建议采用合适的 MATLAB 版本。

本书采用 MATLAB 语言为主要计算机数学语言,系统地介绍其在数学及一般科学运算问题求解中的应用。掌握了该语言将提高读者求解数学问题的能力,提高数学水平,拓广知识面,使得原来看起来无从下手的高深应用数学问题的实际求解变得轻而易举。

1.2.3 开放式免费科学运算语言简介

尽管 MATLAB、Maple 和 Mathematica 等语言有着强大的科学运算功能,但它们还是有局限性的。首先,它们都是昂贵的商品软件;另外,其内核部分的源程序是不可见的,所以一些开放式的科学运算语言还是很受欢迎的。有影响的开放式自由软件包括:

(1) **Scilab**。Scilab 是法国国家计算机科学与控制研究院 (INRIA) 开发的类似于 MATLAB 的软件^[6],该语言是 1989 年正式推出的,其源代码完全公开,且为免费传播的自由软件。该语言的主要应用背景是控制与信号处理。Scilab 下的 Scicos 是类似于 Simulink 的基于框图的仿真工具。从总体上看,除了其本身独有的个别工具箱外,在语言档次和工具箱的深度与广度上与 MATLAB 尚有极大差距,但其源代码公开与产品免费这两大特点足以使其成为科学运算研究领域的一种有影响的计算机语言。Scilab 主页地址为 <http://www.scilab.org/>。

(2) **Octave**。Octave 语言是构思于 1988 年,并于 1993 年正式推出的一种数值计算语言,其出发点和 MATLAB 一样都是数值线性代数的计算。该语言早期的目标是为教学提供支持。该语言目前较广泛地用于教学和其他应用。Octave 网站的地址为

<http://www.gnu.org/software/octave/>。

(3) 此外,小型的、基于矩阵运算的数值语言如 Freemat 和 SpeQ 都是优秀的软件工具,它们的下载地址分别为 http://freemat.sourceforge.net/wiki/index.php/Main_Page 和 <http://www.speqmath.com/index.php?id=1>。然而,从科学运算问题求解角度看,这些语言和 MATLAB 语言相比还是有很大差距的。

1.3 关于本书及相关内容

本书相应的课程是一门新型的课程。此前一些高校陆续开出了相应课程,如“数学实验”^[7]课程简略介绍了计算机数学语言在一些应用数学分支中的最基本的分析方法,但缺乏如何利用实用的计算机数学语言系统、深入地与各数学分支的数学问题求解有机结合。另一门相关的课程“MATLAB 语言及应用”更侧重于 MATLAB 语言的编程内容,对数学问题求解介绍也不全面。从作者本人多年一线教学经验看,如果能找出一种中间途径,既介绍 MATLAB 编程的基本方法,又能全面系统地介绍其在应用数学各个分支的问题求解中的应用,无疑将会对读者大有裨益,这就是编写本书的初衷。

本书的第一版和第二版在本科生、研究生及研修班实际教学中已经使用多届,配备了较好的交互性计算机辅助教学材料,该书还被众多高校学生与科研人员用作提高数学问题求解能力的参考资料及参考手册。在长期教学与研究中,作者整理各方面的反馈,融合更新的知识,编写了本书的新版本,以期发挥更大的作用。

1.3.1 本书框架设计及内容安排

本书各章的安排如下:

第 1 章综述 MATLAB 等计算机数学语言的发展概况,介绍学习本课程的必要性及本课程与其他课程之间的关系。

第 2 章“MATLAB 语言程序设计基础”,以比较简洁的形式对 MATLAB 语言编程、科学绘图等方面进行介绍,为学习本课程打下必要的基础。

第 3 章“微积分问题的计算机求解”,包括极限和基本微积分问题的解析解法、函数的级数展开、级数求和与序列求积、曲线积分和曲面积分、数值微分、数值积分,其解析解部分基本涵盖了高等数学课程的全部计算内容。

第 4 章“线性代数问题的计算机求解”,包括矩阵基本分析、矩阵基本变换、线性方程组的计算机求解、矩阵函数的求解等,这部分远比传统线性代数课程的内容更广泛。

第 5 章“积分变换与复变函数问题的计算机求解”,包括 Laplace 变换、Fourier 变换、 z 变换及反变换问题等问题的计算机直接推导,复变函数的留数、部分分式展开计算。还介绍差分方程解析与数值求解方法、复平面映射等内容。

第 6 章“代数方程与最优化问题的计算机求解”,包括非线性方程的解析解与数值解、无约束最优化、有约束最优化、整数规划等内容。本章还引入了线性矩阵不等式问题的求解、多目标规划和动态规划问题求解等新内容。

第 7 章“微分方程问题的计算机求解”，包括微分方程的解析解法、常微分方程数值解概述、常微分方程组初值问题的 MATLAB 求解、特殊微分方程与延迟微分方程的求解、边值问题的求解、偏微分方程求解入门。还将介绍基于 Simulink 框图的微分方程数值解方法。

第 8 章“数据插值与函数逼近问题的计算机求解”，包括插值与数据拟合、样条插值函数及基于样条插值的数值微积分运算、曲线拟合与平滑、特殊函数计算、数字信号处理与快速 Fourier 变换技术、滤波技术与滤波器设计等。

第 9 章“概率论与数理统计问题的计算机求解”，包括概率分布与随机数生成、统计量分析、数理统计方法、统计假设检验、方差分析和主成分分析等。

第 10 章“数学问题的非传统解法”，包括模糊逻辑与模糊推理、人工神经网络在数据拟合中的应用、遗传算法在最优化求解中的应用、小波理论在数据处理中的应用、粗糙集理论与应用及分数阶微积分理论与计算等。这里给出相关领域的入门知识，读者可以由此为起点，利用 MATLAB 语言提供的工具直接求解相关的问题。

本书内容看似在介绍数学，但最终目的是期望读者在理解相关数学领域最基本概念的前提下，绕开纯数学和底层烦琐的推导过程，直接由计算机数学语言得出数学问题的解。所以学习本课程将使读者提高数学素养，掌握解决实际科学运算问题的方法，为下一步学习并实践其他课程也打下一个较好的基础。

1.3.2 MATLAB 语言学习方法与资源

学好 MATLAB 语言，可以将 30 字的学习准则作为座右铭，即“带着问题学，活学活用，学用结合，急用先学，立竿见影，在用字上狠下工夫”。

本书系统深入地介绍了 MATLAB 语言在各个应用数学分支中的应用，然而，再厚的一本书也不可能包括所有的内容和解答，用户在学习使用 MATLAB 语言时应该充分地利用各种资源。例如，The MathWorks 公司网站 <http://www.mathworks.com> 上免费提供了全套 MATLAB 语言及工具箱手册的 HTML 版和 PDF 版电子文档，和本书相关的手册参见文献 [8~20]。

The MathWorks 公司的网站还提供了 File Exchange 子网站，公布用户开发的各种实用程序进行交流，此外，强大的用户组能为 MATLAB 语言的学习与应用提供各种帮助。

联机帮助系统是 MATLAB 提供的重要帮助手段，读者应该学会灵活使用联机帮助系统。联机帮助信息可以由 MATLAB 命令窗口的 Help 菜单获得，该菜单将打开如图 1-1(a) 所示的菜单项。选择其中的 Using the Desktop 菜单项将打开联机帮助窗口，如图 1-1(b) 所示。帮助信息可以由该窗口得出。

也可以在 MATLAB 命令窗口下键入 `help` 命令或 `doc` 直接显示帮助信息。还可以使用 `lookfor` 命令查询某关键词，获得帮助信息。

1.3.3 本课程与其他相关课程的关系

本书对应的课程不是数值分析类课程的 MATLAB 版本介绍，应该理解成是从更高层次、采用更有效的方法，解决实际中可能遇到的数学问题的方法论。数值分析课程更侧重于

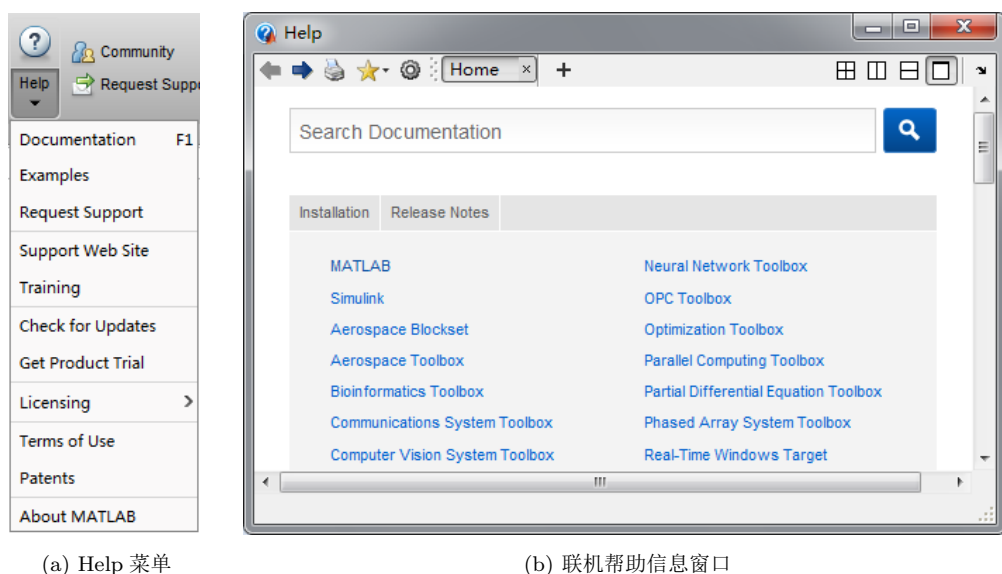


图 1-1 联机帮助信息查询

介绍经典的算法,侧重于介绍原始的、能充分显示问题来龙去脉的算法,而在实际问题求解中这些方法通常是不实用的,甚至是根本不使用的。例如,在求解常微分方程初值问题时,数值分析课程最侧重介绍的是四阶定步长 Runge-Kutta 算法,但从实际求解的实践来看,采用定步长算法是有问题的。其一是由于算法不如变步长算法高效,有时在求解中可能花费难以接受的时间;另一个方面,也是更重要的,定步长算法在求解过程中对解的正确性没有检测环节,在求解过程中出现误差也无从知晓,故得出的结果可靠性存在问题,而采用变步长算法能根据误差自动选择计算步长,保证求解的正确性。另外很多内容,如在实际应用中经常遇到是延迟微分方程、微分代数方程等的求解在传统数值分析课程中也是不介绍的。

高等数学和各类应用数学的计算问题均可以由介绍的方法直接求解,但这并不意味着高等数学类课程的理论不重要。读者可以在学好高等数学、应用数学理论的基础上更好地理解问题,更容易地解决问题。

本书对各种数学问题一般采用 3 个阶段进行描述,第一个阶段是用纯数学方式描述数学问题,解释数学表达式的物理意义;第二个阶段简要介绍求解数学问题的算法,第三个阶段介绍解决该类数学问题的 MATLAB 语句或语句组。前两个阶段为支持第三个阶段学习的背景材料,对于数学基础不是特别好的读者来说,可以在理解数学问题描述的基础上直接学习第三个阶段的内容,学会求解数学问题的手段。本书重点侧重于用 MATLAB 语言直接求解数学问题的方法论,而不是该问题的数学理论。

1.4 习 题

1. 在你的机器上安装 MATLAB 语言环境,并键入 `demo` 命令,由给出的菜单系统和对话框原型演示程序,领略 MATLAB 语言在求解数学问题方面的能力与方法。

2. 假设已知 Lyapunov 方程如下

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \mathbf{X} + \mathbf{X} \begin{bmatrix} 16 & 4 & 1 \\ 9 & 3 & 1 \\ 4 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

试利用 `lookfor lyapunov` 命令查询和关键词 `lyapunov` 有关的函数名,并用 `doc` 命令获得相关函数的进一步调用信息,观察是否能得出该方程的解,并检验得出解的精度。

3. 利用联机帮助命令 `doc symbolic/diff` 查询符号运算工具箱中的求导函数 `diff`,在了解所提供帮助信息的基础上试着求解例 1.1 中给出的问题,并和该例比较得出的结果。

参考文献

- [1] Press W H, Flannery B P, Teukolsky S A, et al. Numerical recipes, the art of scientific computing. Cambridge: Cambridge University Press, 1986
- [2] Garbow B S, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide extension, Lecture notes in computer sciences, volume 51. New York: Springer-Verlag, 1977
- [3] Smith B T, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide. Lecture notes in computer sciences. New York: Springer-Verlag, second edition , 1976
- [4] Dongarra J J, Bunsh J R, Molor C B. LINPACK user's guide. Philadelphia: Society of Industrial and Applied Mathematics, 1979
- [5] Numerical Algorithm Group. NAG FORTRAN library manual, 1982
- [6] 胡包钢,赵星,康孟珍. 科学计算自由软件 SCILAB 教程. 北京: 清华大学出版社, 2003
- [7] 何文章,桂占吉,贾敬. 大学数学实验. 哈尔滨: 哈尔滨工程大学出版社, 1999
- [8] The MathWorks Inc. MATLAB getting started
- [9] The MathWorks Inc. MATLAB mathematics
- [10] The MathWorks Inc. MATLAB graphics
- [11] The MathWorks Inc. MATLAB image processing toolbox, user's manual
- [12] The MathWorks Inc. MATLAB symbolic toolbox user's manual
- [13] The MathWorks Inc. MATLAB optimization toolbox user's manual
- [14] The MathWorks Inc. MATLAB partial differentiation toolbox user's manual
- [15] The MathWorks Inc. MATLAB splines toolbox user's manual
- [16] The MathWorks Inc. MATLAB statistics toolbox user's manual
- [17] The MathWorks Inc. MATLAB fuzzy logic toolbox user's manual
- [18] The MathWorks Inc. MATLAB neural network toolbox user's manual
- [19] The MathWorks Inc. MATLAB genetic algorithm and direct search toolbox user's manual
- [20] The MathWorks Inc. MATLAB wavelet toolbox user's manual

第2章 MATLAB 语言程序设计基础

MATLAB 语言是当前国际上自动控制领域的首选计算机语言,也是很多理工科专业最适合的计算机数学语言。本书以 MATLAB 语言为主要计算机语言,系统、全面地介绍在数学运算问题中 MATLAB 语言的应用。掌握该语言不但有助于更深入地理解和掌握数学问题的求解思路,提高求解数学问题的能力,而且还可以充分利用该语言,在其他专业课程的学习中得到积极的帮助。

和其他程序设计语言相比,MATLAB 语言有如下的优势:

(1) **简洁高效性**。MATLAB 程序设计语言集成度高,语句简洁,往往用 C/C++ 等程序设计语言编写的数百条语句,用 MATLAB 语言一条语句就能解决问题,其程序可靠性高、易于维护,可以大大提高解决问题的效率和水平。

(2) **科学运算功能**。MATLAB 语言以矩阵为基本单元,可以直接用于矩阵运算。另外,最优化问题、数值微积分问题、微分方程数值解问题、数据处理问题等都能直接用 MATLAB 语言求解。

(3) **绘图功能**。MATLAB 语言可以用最直观的语句将实验数据或计算结果用图形的方式显示出来,并可以将以往难以显示出来的隐函数直接用曲线绘制出来。MATLAB 语言还允许用户用可视的方式编写图形用户界面,其难易程度和 Visual Basic 相仿,这使得用户可以很容易地利用该语言编写通用程序。

(4) **庞大的工具箱与模块集**。MATLAB 是被控制界的学者“捧红”的,是控制界通用的计算机语言,在应用数学及控制领域几乎所有的研究方向均有自己的工具箱,而且由专业领域内知名专家编写,可信度比较高。随着 MATLAB 的日益普及,在其他工程领域也出现了工具箱,这也大大促进了 MATLAB 语言在诸多领域的应用。

(5) **强大的动态系统仿真功能**。Simulink 提供的面向框图的仿真及概念性仿真功能,使得用户能容易地建立复杂系统模型,准确地对其进行仿真分析。Simulink 的概念性仿真模块集允许用户在一个框架下对含有控制环节、机械环节和电子、电机环节的机电一体化系统进行建模与仿真,这是目前其他计算机语言无法做到的。

本章 2.1 节将介绍 MATLAB 语言编程的最基本内容,包括数据结构、基本语句结构和重要的冒号表达式与子矩阵提取方法。2.2 节将介绍 MATLAB 语言中矩阵的基本数学运算,包括代数运算、逻辑运算、比较运算及简单的数论运算函数。2.3 节将介绍 MATLAB 语言的基本编程结构,如循环语句结构、条件转移结构、开关结构和试探结构,介绍各种结构在程序设计中的应用。2.4 节介绍 MATLAB 语言编程中最重要的程序结构——M-函数的结构与程序编写技巧。2.5 节、2.6 节将分别介绍基于 MATLAB 语言的二维、三维图形绘制的方法,如各种二维曲线绘制、隐函数的曲线绘制、三维图形绘制及视角设置等,并将介绍图形修饰方法。2.7 节还将给出四维图形的绘制方法,包括基于时间的三维动画和体视化方法。

限于本书的篇幅,本章只能介绍 MATLAB 语言最基础的入门知识。初步掌握该语言的基本功能,就能更好地理解和使用该语言研究数学问题求解的内容,还可以为其他相关后续课程的学习打下良好的基础。

2.1 MATLAB 程序设计语言基础

2.1.1 MATLAB 语言的变量与常量

MATLAB 语言变量名应该由一个字母引导,后面可以跟字母、数字、下划线等。例如,MYvar12, MY_Var12 和 MyVar12_ 均为有效的变量名,而 12MyVar 和 _MyVar12 为无效的变量名。在 MATLAB 中变量名是区分大小写的,也就是说,Abc 和 ABc 两个变量名表达的是不同的变量,在使用 MATLAB 语言编程时一定要注意。

在 MATLAB 语言中还为特定常数保留了一些名称,虽然这些常量都可以重新赋值,但建议在编程时应尽量避免对这些量重新赋值。

(1) **eps**。机器的浮点运算误差限。PC 机上 **eps** 的默认值为 2.2204×10^{-16} ,若某个量的绝对值小于 **eps**,则可以认为这个量为 0。

(2) **i** 和 **j**。若 **i** 或 **j** 量不被改写,则它们表示纯虚数量 **j**。但在 MATLAB 程序编写过程中经常事先改写这两个变量的值,如在循环过程中常用这两个变量来表示循环变量,所以应该确认使用这两个变量时没有被改写。如果想恢复该变量,则可以用语句 $i = \text{sqrt}(-1)$ 设置,即对 $\sqrt{-1}$ 。

(3) **Inf**。无穷大量 $+\infty$ 的 MATLAB 表示,也可以写成 **inf**。同样地, $-\infty$ 可以表示为 **-Inf**。在 MATLAB 程序执行时,即使遇到了以 0 为除数的运算,也不会终止程序的运行,而只给出一个“除 0”警告,并将结果赋成 **Inf**,这样的定义方式符合 IEEE 的标准。从数值运算编程角度看,这样的实现形式明显优于 C 这样的非专业语言。

(4) **NaN**。不定式(not a number),通常由 $0/0$ 运算、**Inf/Inf**、 $0 \times \text{Inf}$ 及其他可能的运算得出。**NaN** 是一个很奇特的量,如 **NaN** 与 **Inf** 的乘积仍为 **NaN**。

(5) **pi**。圆周率 π 的双精度浮点表示。

(6) **lasterr**。存放最新一次的错误信息。此变量为字符串型,如果在本次执行过程中没出现过错误,则此变量为空字符串。

(7) **lastwarn**。存放最新的警告信息。若未出现过警告,则此变量为空字符串。

2.1.2 数据结构

1. 数值型数据

强大方便的数值运算功能是 MATLAB 语言最显著的特色。为保证较高的计算精度, MATLAB 语言中最常用的数值量为双精度浮点数,占 8 个字节(64 位),遵从 IEEE 记数法,有 11 个指数位、52 位尾数及一个符号位,值域的近似范围为 $-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$,其 MATLAB 表示为 **double()**。考虑到一些特殊的应用,比如图像处理, MATLAB 语言还引入了无符号的 8 位整形数据类型,其 MATLAB 表示为 **uint8()**,其值域为 $0 \sim 255$,这样可以大大地节省 MATLAB 的存储空间,提高处理速度。此外,在 MATLAB 中还可以使用

其他的数据类型,如 `int8()`、`int16()`、`int32()`、`uint16()`、`uint32()` 等,每一个类型后面的数字表示其位数,其含义不难理解。

2. 符号型

MATLAB 还定义了“符号”型变量,以区别于常规的数值型变量,可以用于公式推导和数学问题的解析解法。进行解析运算前需要首先将采用的变量申明为符号变量,这需要用 `syms` 命令来实现。该语句具体的用法为 `syms var_list var_props`,其中, `var_list` 给出需要申明的变量列表,可以同时申明多个变量,中间用空格分隔,而不能用逗号等分隔。如果需要,还可以进一步申明变量的类型 `var_props`,可以使用的类型为 `real`、`positive` 等。如果需要将 a, b 均定义为符号变量,则可以用 `syms a b` 语句声明,该命令还支持对符号变量具体形式的设定,如 `syms a real`。

符号变量的类型可以由 `assumptions()` 函数读出,例如,若用 `syms a real` 语句申明变量 a ,则 `assumptions(a)` 将返回 $a \in \mathbb{R}$ 。

符号型数值可以通过变精度算法函数 `vpa()` 以任意指定的精度显示出来。该函数的调用格式为 `vpa(A)`,或 `vpa(A, n)`,其中, A 为需要显示的表达式或矩阵, n 为指定的有效数字位数,前者以默认的 32 位十进制位数显示结果。

例 2-1 试显示出圆周率 π 的前 300 位有效数字。

解 使用符号运算工具箱中提供的 `vpa()` 函数可以按任意精度显示符号变量的值,故题中要求的结果可以用下面语句立即显示出来

```
>> vpa(pi,300)
```

这样可以显示出 π 的值为 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127。若不指定 n ,则 `vpa(pi)` 命令将得出结果为 $\pi = 3.1415926535897932384626433832795$ 。

3. 其他数据结构

除了用于数学运算的数值数据结构外, MATLAB 还支持下面的数据结构:

(1) **字符串型数据**。MATLAB 支持字符串变量,可以用它来存储相关的信息。和 C 语言等程序设计语言不同, MATLAB 字符串是用单引号括起来的,而不是用双引号。

(2) **多维数组**。三维数组是一般矩阵的直接拓展,可以这样理解,三维数组可以直接用于彩色数字图像的描述,在控制系统的分析上也可以直接用于多变量系统的表示上。在实际编程中还可以使用维数更高的数组。

(3) **单元数组**。单元数组是矩阵的直接扩展,其存储格式类似于普通的矩阵,而矩阵的每个元素不是数值,可以认为能存储任意类型的信息,这样每个元素称为“单元”(cell),例如, $A\{i, j\}$ 可以表示单元数组 A 的第 i 行、第 j 列的内容。

(4) **类与对象**。MATLAB 允许用户自己编写包含各种复杂信息的变量,亦即类变量,该变量可以包含各种下级的信息,还可以重新对类定义其计算,这在很多领域都特别有用,后面遇到的时候将通过例子介绍相关的编程方法。

2.1.3 MATLAB 的基本语句结构

MATLAB 的语句有两种最基本的结构——直接赋值结构和函数调用结构。

(1) **直接赋值语句**。直接赋值语句的基本结构为 **赋值变量 = 赋值表达式**，这一过程把等号右边的表达式直接赋给左边的赋值变量，并返回到 MATLAB 的工作空间。如果赋值表达式后面没有分号，则将在 MATLAB 命令窗口中显示表达式的运算结果。若不想显示运算结果，则应该在赋值语句的末尾加一个分号。如果省略了赋值变量和等号，则表达式运算的结果将赋给保留变量 **ans**。所以说，保留变量 **ans** 将永远存放最近一次无赋值变量语句的运算结果。

例 2-2 试在 MATLAB 工作空间中输入矩阵

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

解 在 MATLAB 语言中表示一个矩阵是很容易的事，可以由下面的 MATLAB 语句将该矩阵直接输入到工作空间中

```
>> A=[1,2,3; 4 5,6; 7,8 0]
```

该语句将矩阵赋给变量 **A**，同时，在命令窗口中按照下面的格式显示该矩阵。为阅读方便，本书后续内容将不再给出 MATLAB 格式的显示，而直接给出数学格式的显示。其中的 **>>** 为 MATLAB 的提示符，由机器自动给出，在提示符下可以输入各种各样的 MATLAB 命令。矩阵的内容由方括号括起来的部分表示，在方括号中的分号表示矩阵的换行，逗号或空格表示同一行矩阵元素间的分隔。给出了上面的命令，就可以在 MATLAB 的工作空间中建立一个 **A** 变量了。如果不想显示中间结果，则应该在语句末尾加一个分号，如

```
>> A=[1,2,3; 4 5,6; 7,8 0]; % 不显示结果,但进行赋值
```

```
A=[[A; [1 2 3]], [1;2;3;4]]; % 矩阵维数动态变化
```

例 2-3 试在 MATLAB 环境中输入复数矩阵

$$\mathbf{B} = \begin{bmatrix} 1+9j & 2+8j & 3+7j \\ 4+6j & 5+5j & 6+4j \\ 7+3j & 8+2j & 0+j \end{bmatrix}$$

解 复数矩阵的输入同样也是很简单的，在 MATLAB 环境中定义了两个记号 **i** 和 **j**，可以用来直接输入复数矩阵，这样可以通过下面的 MATLAB 语句对复数矩阵直接进行赋值

```
>> B=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i]
```

(2) **函数调用语句**。函数调用的基本结构为 **[返回变量列表]=函数名(输入变量列表)**，其中，函数名的要求和变量名的要求是一致的，一般函数名应该对应在 MATLAB 路径下的一个文件。例如，函数名 **my_fun** 应该对应于 **my_fun.m** 文件。当然，还有一些函数名需对应于 MATLAB 内核中的内核函数 (built-in function)，如 **inv()** 函数等。

返回变量列表和输入变量列表均可以由若干个变量名组成，它们之间应该分别用逗号分隔。返回变量还允许用空格分隔，例如 **[U S V] = svd(X)**，该函数对给定的 **X** 矩阵进行奇异值分解，所得的结果由 **U, S, V** 这 3 个变量返回。

2.1.4 冒号表达式与子矩阵提取

冒号表达式是 MATLAB 中很有用的表达式,在向量生成、子矩阵提取等很多方面都是特别重要的。冒号表达式的格式为 $\mathbf{v} = s_1:s_2:s_3$,该函数将生成一个行向量 \mathbf{v} ,其中 s_1 为向量的起始值, s_2 为步距,该向量将从 s_1 出发,每隔步距 s_2 取一个点,直至不超过 s_3 的最大值就可以构成一个向量。若省略 s_2 ,则步距取默认值 1。

例 2-4 试探不同的步距,从 $t \in [0, \pi]$ 区间取出一些点构成向量。

解 先试一下步距 0.2,这样可以用下面的语句生成一个向量

```
>> v1=0: 0.2: pi % 注意,最终取值为 3 而不是  $\pi$ 
```

该语句将生成行向量 $\mathbf{v}_1 = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4, 2.6, 2.8, 3]$ 。

下面还将尝试冒号表达式不同的写法,并得出如下的结果

```
>> v2=0: -0.1: pi, v3=0:pi, v4=pi:-1:0
```

这样产生的 \mathbf{v}_2 向量为 1×0 空矩阵, $\mathbf{v}_3 = [0, 1, 2, 3]$, $\mathbf{v}_4 = [3.1416, 2.1416, 1.1416, 0.1416]$ 。

提取子矩阵的具体方法是 $\mathbf{B} = \mathbf{A}(\mathbf{v}_1, \mathbf{v}_2)$,其中, \mathbf{v}_1 向量表示子矩阵要保留的行号构成的向量, \mathbf{v}_2 表示要保留的列号构成的向量,这样从 \mathbf{A} 矩阵中提取有关的行和列,就可以构成子矩阵 \mathbf{B} 了。若 \mathbf{v}_1 为 $:$,则表示要提取所有的行, \mathbf{v}_2 亦有相应的处理结果。关键词 **end** 表示最后一行(或列,取决于其位置)。

例 2-5 下面将列出若干命令,并加以解释,读者可以自己用一个矩阵测试这些语句,体会子矩阵提取的方法。

```
>> A=[1,2,3; 4 5,6; 7,8 0]; % 矩阵输入
```

```
B1=A(1:2:end, :) % 提取  $\mathbf{A}$  矩阵全部奇数行、所有列
```

```
B2=A([3,2,1],[1,1,1]) % 提取  $\mathbf{A}$  矩阵 3,2,1 行、反复三次由首列构成子矩阵
```

```
B3=A(:,end:-1:1) % 将  $\mathbf{A}$  矩阵左右翻转,即最后一列排在最前面
```

上述的语句将生成下面的各个矩阵

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 0 \end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix} 7 & 7 & 7 \\ 4 & 4 & 4 \\ 1 & 1 & 1 \end{bmatrix}, \mathbf{B}_3 = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 0 & 8 & 7 \end{bmatrix}$$

2.2 基本数学运算

2.2.1 矩阵的代数运算

如果一个矩阵 \mathbf{A} 有 n 行、 m 列元素,则称 \mathbf{A} 矩阵为 $n \times m$ 矩阵;若 $n = m$,则矩阵 \mathbf{A} 又称为方阵。MATLAB 语言中定义了下面各种矩阵的基本代数运算:

(1) **矩阵转置**。在数学公式中一般把一个矩阵的转置记作 \mathbf{A}^T ,假设 \mathbf{A} 矩阵为一个 $n \times m$ 矩阵,则其转置矩阵 \mathbf{B} 的元素定义为 $b_{ji} = a_{ij}$, $i = 1, \dots, n$, $j = 1, \dots, m$,故 \mathbf{B} 为 $m \times n$ 矩阵。如果 \mathbf{A} 矩阵含有复数元素,则对之进行转置时,其转置矩阵 \mathbf{B} 的元素定义为 $b_{ji} = a_{ij}^*$, $i = 1, \dots, n$, $j = 1, \dots, m$,亦即首先对各个元素进行转置,然后再逐项求取其共轭复数值。这种转置方式又称为 Hermite 转置,其数学记号为 $\mathbf{B} = \mathbf{A}^*$ 。MATLAB 中用 \mathbf{A}' 可以求出 \mathbf{A} 矩阵的 Hermite 转置,矩阵的转置则可以由 $\mathbf{A}.'$ 求出。

(2) **加减法运算**。假设在 MATLAB 工作环境下有两个矩阵 A 和 B , 则可以由 $C=A+B$ 和 $C=A-B$ 命令执行矩阵加减法。若 A 和 B 矩阵的维数相同, 它会自动地将 A 和 B 矩阵的相应元素相加减, 从而得出正确的结果, 并赋给 C 变量。若二者之一为标量, 则应该将其遍加(减)于另一个矩阵。在其他情况下, MATLAB 将自动地给出错误信息, 提示用户两个矩阵的维数不匹配。

(3) **矩阵乘法**。假设有两个矩阵 A 和 B , 其中 A 矩阵的列数与 B 矩阵的行数相等, 或其一为标量, 则称 A 、 B 矩阵是可乘的, 或称 A 和 B 矩阵的维数是相容的。假设 A 为 $n \times m$ 矩阵, 而 B 为 $m \times r$ 矩阵, 则 $C=AB$ 为 $n \times r$ 矩阵, 其各个元素为 $c_{ij} = \sum_{k=1}^m a_{ik}b_{kj}$, 其中 $i=1, 2, \dots, n, j=1, 2, \dots, r$ 。MATLAB 语言中两个矩阵的乘法由 $C=A*B$ 直接求出, 且这里并不需要指定 A 和 B 矩阵的维数。若 A 和 B 矩阵的维数相容, 则可以准确无误地获得乘积矩阵 C ; 如果二者的维数不相容, 则将给出错误信息, 通知用户两个矩阵不可乘。

(4) **矩阵的左除**。MATLAB 中用“\”运算符表示两个矩阵的左除, $A \setminus B$ 为方程 $AX=B$ 的解 X 。若 A 为非奇异方阵, 则 $X=A^{-1}B$ 。如果 A 矩阵不是方阵, 也可以求出 $X=A \setminus B$, 这时将使用最小二乘解法来求取 $AX=B$ 中的 X 矩阵。

(5) **矩阵的右除**。MATLAB 中定义了“/”符号, 用于表示两个矩阵的右除, 相当于求方程 $XA=B$ 的解。 A 为非奇异方阵时 B/A 为 BA^{-1} , 但在计算方法上存在差异, 更精确地, 有 $B/A=(A' \setminus B')'$ 。

(6) **矩阵翻转**。MATLAB 提供了一些矩阵翻转处理的特殊命令, 如 $B=\text{fliplr}(A)$ 命令将矩阵 A 进行左右翻转再赋给 B , 亦即 $b_{ij}=a_{i,n+1-j}$, 而 $C=\text{flipud}(A)$ 命令将 A 矩阵进行上下翻转并将结果赋给 C , 亦即 $c_{ij}=a_{m+1-i,j}$ 。 $D=\text{rot90}(A)$ 将 A 矩阵逆时针旋转 90° 后赋给 D , 亦即 $d_{ij}=a_{j,n+1-i}$ 。 $\text{rot90}(A,k)$ 还可以旋转该矩阵 $90k^\circ$ 。

(7) **矩阵乘方运算**。一个矩阵的乘方运算可以在数学上表述成 A^x , 而前提条件是要求 A 矩阵为方阵。如果 x 为正整数, 则乘方表达式 A^x 的结果可以将 A 矩阵自乘 x 次得出。如果 x 为负整数, 则可以将 A 矩阵自乘 $-x$ 次, 然后对结果进行求逆运算就可以得出该乘方结果。如果 x 是一个分数, 例如 $x=n/m$, 其中 n 和 m 均为整数, 则相当于将 A 矩阵自乘 n 次, 然后对结果再开 m 次方。在 MATLAB 中统一表示成 $F=A^x$ 。

(8) **点运算**。MATLAB 中定义了一种特殊的运算, 即所谓的点运算。两个矩阵之间的点运算是它们对应元素的直接运算。例如, $C=A.*B$ 表示 A 和 B 矩阵的相应元素之间直接进行乘法运算, 然后将结果赋给 C 矩阵, 即, $c_{ij}=a_{ij}b_{ij}$ 。这种点乘积运算又称为 Hadamard 乘积。注意, 点乘积运算要求 A 和 B 矩阵的维数相同。可以看出, 这种运算和普通乘法运算是不同的。

点运算在 MATLAB 中起着很重要的作用。例如, 当 x 是一个向量时, 则求取数值 $[x_i^5]$ 时不能直接写成 x^5 , 而必须写成 $x.^5$ 。在进行矩阵的点运算时, 同样要求运算的两个矩阵的维数一致, 或其中一个变量为标量。其实一些特殊的函数, 如 $\sin()$ 也是由点运算的形式进行的, 因为它要对矩阵的每个元素求取正弦值。

矩阵点运算不只可以用于点乘积运算,还可以用于其他运算的场合。例如对前面给出的 \mathbf{A} 矩阵作 $\mathbf{A}.\wedge\mathbf{A}$ 运算,则新矩阵的第 (i,j) 元素为 $a_{ij}^{a_{ij}}$,这样可以得出下面的结果

```
>> A=[1,2,3; 4 5,6; 7,8 0]; B=A.^A
```

该语句将计算并生成如下的矩阵

$$\mathbf{B} = \begin{bmatrix} 1^1 & 2^2 & 3^3 \\ 4^4 & 5^5 & 6^6 \\ 7^7 & 8^8 & 0^0 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 27 \\ 256 & 3125 & 46656 \\ 823543 & 16777216 & 1 \end{bmatrix}$$

例 2-6 重新考虑例 2-2 中的 \mathbf{A} 矩阵,试求出其全部三次方根并检验结果。

解 由 \wedge 运算可以容易地得出原矩阵的一个三次方根

```
>> A=[1,2,3; 4,5,6; 7,8,0]; C=A^(1/3), e=norm(A-C^3)
```

具体表示如下,经检验误差为 $e = 1.0145 \times 10^{-14}$,比较精确。

$$\mathbf{C} = \begin{bmatrix} 0.77179 + j0.6538 & 0.48688 - j0.015916 & 0.17642 - j0.2887 \\ 0.88854 - j0.072574 & 1.4473 + j0.47937 & 0.52327 - j0.49591 \\ 0.46846 - j0.64647 & 0.66929 - j0.6748 & 1.3379 + j1.0488 \end{bmatrix}$$

事实上,矩阵的三次方根应该有三个结果,而上面只得出其中的一个。对该方根进行两次旋转,即计算 $\mathbf{C}e^{j2\pi/3}$ 和 $\mathbf{C}e^{j4\pi/3}$,则将得出另外两个根,经检验得出的根是正确的。

```
>> j1=exp(sqrt(-1)*2*pi/3); A1=C*j1, A2=C*j1^2
```

```
e1=norm(A-A1^3), e2=norm(A-A2^3)
```

这样可以得出另外两个根如下,误差都是 10^{-14} 级别。

$$\mathbf{A}_1 = \begin{bmatrix} -0.9521 + j0.34149 & -0.22966 + j0.42961 & 0.16181 + j0.29713 \\ -0.38142 + j0.80579 & -1.1388 + j1.0137 & 0.16784 + j0.70112 \\ 0.32563 + j0.72893 & 0.24974 + j0.91702 & -1.5772 + j0.63425 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} 0.18031 - j0.99529 & -0.25722 - j0.41369 & -0.33823 - j0.008436 \\ -0.50712 - j0.73321 & -0.3085 - j1.4931 & -0.69111 - j0.20521 \\ -0.79409 - j0.082464 & -0.91904 - j0.24222 & 0.23934 - j1.6831 \end{bmatrix}$$

2.2.2 矩阵的逻辑运算

早期版本的 MATLAB 语言并没有定义专门的逻辑变量。在 MATLAB 语言中,如果一个数的值为 0,则可以认为它为逻辑 0,否则为逻辑 1。新版本支持逻辑变量,且上面的定义仍有效。假设矩阵 \mathbf{A} 和 \mathbf{B} 均为 $n \times m$ 矩阵,则在 MATLAB 下定义了如下的逻辑运算:

(1) **矩阵的与运算**。在 MATLAB 下用 $\&$ 号表示矩阵的与运算。例如, $\mathbf{A} \& \mathbf{B}$ 表示两个矩阵 \mathbf{A} 和 \mathbf{B} 相应元素的与运算。如果两个矩阵相应元素均非 0 则该结果元素的值为 1。否则,该元素为 0。

(2) **矩阵的或运算**。在 MATLAB 下用 $|$ 号表示矩阵的或运算,如果两个矩阵相应元素存在非 0 值,则该结果元素的值为 1。否则,该元素为 0。

(3) **矩阵的非运算**。在 MATLAB 下用 \sim 号表示矩阵的非运算。若矩阵元素为 0,则结果为 1,否则为 0。

(4) 矩阵的异或运算。MATLAB 下矩阵 A 和 B 的异或运算可以表示成 $\text{xor}(A, B)$ 。若相应的两个数一个为 0, 一个非 0, 则结果为 1, 否则为 0。

2.2.3 矩阵的比较运算

MATLAB 语言定义了各种比较关系, 如 $C = A > B$, 当 A 和 B 矩阵满足 $a_{ij} > b_{ij}$ 时, $c_{ij} = 1$, 否则 $c_{ij} = 0$ 。MATLAB 语言还支持等于关系, 用 $==$ 表示, 大于等于关系, 用 $>=$ 表示, 还支持不等于 \sim 关系, 其意义是很明显的, 可以直接使用。

MATLAB 还提供了一些特殊的函数, 在编程中也是很实用的。其中, $\text{find}()$ 函数可以查询出满足某关系的数组下标。例如, 若想查出矩阵 C 中数值等于 1 的元素的下标, 则可以给出 $\text{find}(C==1)$ 命令如下

```
>> A=[1,2,3; 4 5,6; 7,8 0]; % 输入实数矩阵
    find(A>=5) % 找出矩阵元素大于等于 5 的下标
```

这样找出的下标 $i = [3, 5, 6, 8]$ 。可以看出, 该函数相当于先将 A 矩阵按列构成列向量, 然后再判断哪些元素大于或等于 5, 返回其下标。而 $\text{find}(\text{isnan}(A))$ 函数将查出 A 变量中为 NaN 的各元素的下标。还可以用下面的格式同时返回行和列坐标

```
>> [i,j]=find(A>=5)
```

这样得出的双下标向量分别为 $i = [3, 2, 3, 2]^T$, $j = [1, 2, 2, 3]^T$ 。此外, $\text{all}()$ 和 $\text{any}()$ 函数也是很实用的查询函数

```
>> a1=all(A>=5), a2=any(A>=5)
```

前一个命令当 A 矩阵的某列元素全都大于或等于 5 时, 相应元素为 1, 否则为 0。而后者在某列中含有大于或等于 5 时, 相应元素为 1, 否则为 0。故而得出的向量分别为 $a_1 = [0, 0, 0]$, $a_2 = [1, 1, 1]$ 。例如若想判定一个矩阵 A 是否元素均大于或等于 5, 则可以简单地写成 $\text{all}(A(:) >= 5)$ 。

2.2.4 解析结果的化简与变换

符号运算工具箱可以用于推导数学公式, 但其结果往往不是最简形式, 或不是用户期望的格式, 所以需要对结果进行化简处理。MATLAB 中最常用的化简函数是 $\text{simple}()$ 函数, 该函数尝试各种化简函数, 最终得出计算机认为最简的结果。该函数的调用格式如下

```
s1 = simple(s) % 从各种方法中自动选择最简格式
[s1,how] = simple(s) % 化简并返回实际采用的化简方法
```

其中, s 为原始表达式, s_1 为化简后表达式, how 为实际采用的化简方式, 为字符串变量。除了 $\text{simple}()$ 函数外, 还有其他专门的化简函数, 如 $\text{collect}()$ 函数可以合并同类项, $\text{expand}()$ 可以展开多项式, $\text{factor}()$ 可以进行因式分解, $\text{numden}()$ 可以提取多项式的分子和分母等。这些函数的信息与调用格式可以由 help 命令得出。

例 2-7 假设已知含有因式的多项式 $P(s) = (s+3)^2(s^2+3s+2)(s^3+12s^2+48s+64)$, 试用各种化简函数对之进行处理, 并理解得出的变换结果。

解 首先应该定义符号变量 s , 这样就可以表示该多项式了。有了多项式, 则先尝试 MATLAB 认为的最简形式。

```
>> syms s; P=(s+3)^2*(s^2+3*s+2)*(s^3+12*s^2+48*s+64) % P 保持原状
P1=simple(P), [P2,m]=simple(P) % 两种化简方法得出同样最简结果
P3=expand(P) % 多项式展开方法
```

这里, 化简后的多项式为分解的因式形式, $P_1 = P_2 = (s+3)^2(s+2)(s+1)(s+4)^3$, 还将返回 `simple()` 函数, 最终采用的化简方法为 'factor'。该多项式展开后的结果为

$$P_3 = s^7 + 21s^6 + 185s^5 + 883s^4 + 2454s^3 + 3944s^2 + 3360s + 1152$$

符号运算工具箱中有一个很有用的变量替换函数 `subs()`, 其格式为

```
f1 = subs(f, x1, x1*) % 变量简单替换, 相当于点运算
f1 = subs(f, {x1, x2, ..., xn}, {x1*, x2*, ..., xn*}) % 同时替换多个变量
```

其中, f 为原表达式。该函数的目的是将其中的 x_1 替换成 x_1^* , 生成新的表达式 f_1 。后一种格式表示可以一次性替换多个变量。

符号运算工具箱的结果可以通过 `latex()` 函数转换成科学排版语言 \LaTeX 能支持的字符串, 该字符串可以直接嵌入 \LaTeX 文档^[1], 得出更好的科学排版效果。

例 2-8 考虑例 2-7 中给出的多项式 $P(s)$, 试用 $s = (z-1)/(z+1)$ 对原式进行双线性变换, 化简得出的结果并得出其 \LaTeX 排版格式。

解 下面语句可以直接完成双线性变换, 并得出结果的最简表达式

```
>> syms s z; P=(s+3)^2*(s^2+3*s+2)*(s^3+12*s^2+48*s+64);
P1=simple(subs(P,s,(z-1)/(z+1))), latex(P1)
```

该语句将得出如下的字符串

```
\frac{8\, z\, \left(2\, z + 1\right)^2\, \left(3\, z + 1\right)\, \left(5\, z + 3\right)^3}{\left(z + 1\right)^7}
```

而该字符串在 \LaTeX 排版语言下可以显示为

$$P_1(z) = 8 \frac{(2z+1)^2 z (3z+1) (5z+3)^3}{(z+1)^7}$$

对于非科技文献排版工具, 如 Microsoft Word 等, 则没有直接的转换程序。

2.2.5 基本数论运算

MATLAB 语言还提供了一组简单的数据变换和基本数论函数, 如表 2-1 所示。下面将演示其中若干函数的应用。读者还可以自己选定矩阵对其他函数实际调用, 观察得出的结果, 以便更好地体会这些函数。

例 2-9 考虑一组数据 $-0.2765, 0.5772, 1.4597, 2.1091, 1.191, -1.6187$, 试用不同的取整方法观察所得出的结果, 并进一步理解取整函数。

解 可以用下面的语句将数据用向量表示, 调用取整函数则得出如下的结果

表 2-1 基本数据变换和数论函数表

函数名	调用格式	函数说明
<code>floor()</code>	<code>n = floor(x)</code>	将 x 中元素按 $-\infty$ 方向取整,即取不足整数,得出 n ,数学上记作 $n = [x]$
<code>ceil()</code>	<code>n = ceil(x)</code>	将 x 中元素按 $+\infty$ 方向取整,即取过剩整数,得出 n
<code>round()</code>	<code>n = round(x)</code>	将 x 中元素按最近的整数取整,亦即四舍五入,得出 n
<code>fix()</code>	<code>n = fix(x)</code>	将 x 中元素按离 0 近的方向取整,得出 n
<code>rat()</code>	<code>[n,d] = rat(x)</code>	将 x 中元素变换成最简有理数, n 和 d 分别为分子和分母矩阵
<code>rem()</code>	<code>B = rem(A,C)</code>	A 中元素对 C 中元素求模得出的余数
<code>gcd()</code>	<code>k = gcd(n,m)</code>	求取两个整数 n 和 m 的最大公约数
<code>lcm()</code>	<code>k = lcm(n,m)</code>	求取两个整数 n 和 m 的最小公倍数
<code>factor()</code>	<code>v = factor(n)</code>	对 n 进行质因数分解,其各个质因数由向量 v 返回
<code>isprime()</code>	<code>v1 = isprime(v)</code>	判定向量 v 中的各个整数值是否为质数,若是则 v_1 向量相应的值置 1,否则为 0

```
>> A=[-0.2765,0.5772,1.4597,2.1091,1.191,-1.6187];
```

```
v1=floor(A), v2=ceil(A), v3=round(A), v4=fix(A) % 不同取整函数
```

采用不同的取整函数将得出 $v_1 = [-1, 0, 1, 2, 1, -2]$, $v_2 = [0, 1, 2, 3, 2, -1]$, $v_3 = [0, 1, 1, 2, 1, -2]$, $v_4 = [0, 0, 1, 2, 1, -1]$ 。

例 2-10 假设 3×3 的 Hilbert 矩阵可以由 $A=\text{hilb}(3)$ 定义,试对其进行有理数变换。

解 用下面的语句可以进行所需变换,并得出所需结果

```
>> A=hilb(3); [n,d]=rat(A)
```

这时得出的两个整数矩阵分别为

$$n = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, d = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

例 2-11 试求 1856120、1483720 的最大公约数与最小公倍数,并得出最小公倍数的质因数分解。

解 由于数值较大,不适合用 MATLAB 的数值形式显示,所以有必要将其转换成符号变量,并由下面的语句直接解出所需的结果

```
>> m=sym(1856120); n=sym(1483720); gcd(m,n), lcm(m,n), factor(lcm(n,m))
```

亦即其最大公约数为 1960,最小公倍数为 1405082840。其最小公倍数的质因数分解可以由下面的语句求出为 $(2)^3(5)(7)^2(757)(947)$ 。

这里使用的 `gcd()` 和 `lcm()` 函数只能用于求解两个整数的相应运算。如果想求多个数值的最大公约数与最小公倍数,则可以嵌套使用这些函数,如 `gcd(gcd(m,n),k)`。

例 2-12 试列出 1~1000 之间的全部质数。

解 用下面的语句就可以立即求出所有满足条件的质数。这里以表 2-2 的形式给出可读性更好的结果。在实际求解过程中,用 `isprime(A)` 测出每个整数是否为质数,最后用下标提取的方式将这些质数提取出来。该结构比较特殊,起的作用是将向量 A 中下标为 1 的那些位保留下来。

```
>> A=1:1000; B=A(isprime(A))
```

表 2-2 1000 以内的质数表

2	19	47	79	109	151	191	229	269	311	353	397	439	479	523	577	617	659	709	757	811	857	907	953
3	23	53	83	113	157	193	233	271	313	359	401	443	487	541	587	619	661	719	761	821	859	911	967
5	29	59	89	127	163	197	239	277	317	367	409	449	491	547	593	631	673	727	769	823	863	919	971
7	31	61	97	131	167	199	241	281	331	373	419	457	499	557	599	641	677	733	773	827	877	929	977
11	37	67	101	137	173	211	251	283	337	379	421	461	503	563	601	643	683	739	787	829	881	937	983
13	41	71	103	139	179	223	257	293	347	383	431	463	509	569	607	647	691	743	797	839	883	941	991
17	43	73	107	149	181	227	263	307	349	389	433	467	521	571	613	653	701	751	809	853	887	947	997

2.3 MATLAB 语言的流程结构

作为一种程序设计语言，MATLAB 提供了循环语句结构、条件语句结构、开关语句结构以及与众不同的试探语句。本节将介绍各种语句结构。

2.3.1 循环结构

循环结构可以由 `for` 或 `while` 语句引导，用 `end` 语句结束，在这两个语句之间的部分称为循环体。这两种语句结构的示意图分别如图 2-1 (a)、(b) 所示。

(1) `for` 语句的一般结构 `for i = v, 循环结构体, end`。

在 `for` 循环结构中，`v` 为一个向量，循环变量 `i` 每次从 `v` 向量中取一个数值，执行一次循环体的内容，如此下去，直至执行完 `v` 向量中所有的分量，将自动结束循环体的执行。由此可见，这样的格式比 C 语言的相应格式灵活得多。如果 `v` 是矩阵，则每次 `i` 取一个列向量。

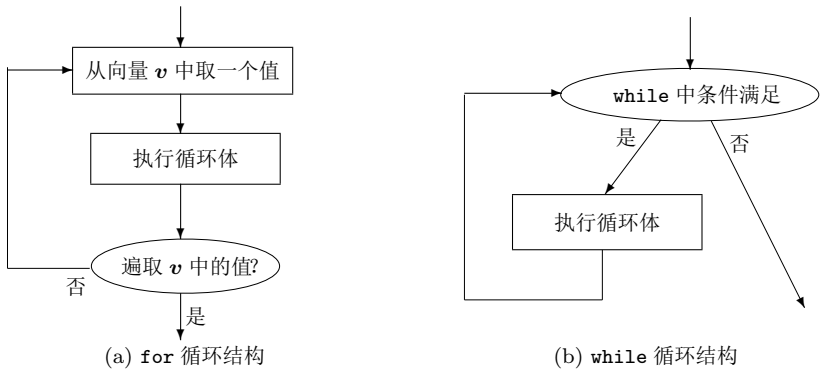


图 2-1 循环结构的示意图

(2) `while` 循环语句的基本结构 `while (条件式), 循环结构体, end`。

`while` 循环中的条件式是一个逻辑表达式，若其值为真（非零）则将自动执行循环体的结构，执行完后再判定“条件式”的真伪，为真则仍然执行结构体，否则将退出循环体结构。

`while` 与 `for` 循环结构是不同的，下面将通过例子演示它们的区别及适用场合。

例 2-13 用循环结构求解 $\sum_{i=1}^{100} i$ 。

解 利用循环语句中的 for 结构和 while 结构,可以按下面的语句分别编程,并得出相同的结果,即 $s_1 = s_2 = 5050$

```
>> s1=0; for i=1:100, s1=s1+i; end, s1
      s2=0; i=1; while (i<=100), s2=s2+i; i=i+1; end, s2
```

其中,for 结构的编程稍简单些。事实上,前面的求和用 sum(1:100) 就能够得出所需的结果,这样做借助了 MATLAB 的 sum() 函数对整个向量进行直接操作,故程序更简单了。

例 2-14 求出满足 $\sum_{i=1}^m i > 10000$ 的最小 m 值。

解 这样的问题用 for 循环结构就不便求解,而应该用 while 结构来求出所需的 m 值。具体的语句如下,得出的结果为 $s = 10011, m = 141$,该结果也可以通过 sum(1:m) 命令检验

```
>> s=0; m=0; while (s<=10000), m=m+1; s=s+m; end, s, m
```

循环语句在 MATLAB 语言中是可以嵌套使用的,也可以在 for 下使用 while,或相反使用。另外,在循环语句中如果使用 break 语句,则可以结束上一层的循环结构。

在 MATLAB 程序中,循环结构的执行速度较慢。所以在实际编程过程中,如果能对整个矩阵进行运算时,尽量不要采用循环结构,这样可以提高代码的效率。下面将通过例子演示循环与向量化编程的区别。

例 2-15 求解级数求和问题 $S = \sum_{i=1}^{100000} \left(\frac{1}{2^i} + \frac{1}{3^i} \right)$ 。

解 用循环语句和向量化方式的执行时间分别可以用 tic, toc 命令测出,可见对这个问题来说,向量化所需的时间相当于循环结构的 37.5%,故用向量化的方法可以节省时间。

```
>> tic, s=0; for i=1:100000, s=s+1/2^i+1/3^i; end; toc
      tic, i=1:100000; s=sum(1./2.^i+1./3.^i); toc
```

2.3.2 条件转移结构

条件转移结构是一般程序设计语言都支持的结构。MATLAB 下的最基本的转移结构是 if ... end 型的,也可以和 else 语句和 elseif 语句扩展转移语句。该语句的示意图如图 2-2 所示,其一般结构为

```
if (条件 1)      %如果条件 1 满足,则执行下面的段落 1
    语句组 1      %这里也可以嵌套下级的 if 结构
elseif (条件 2)  %否则如果满足条件 2,则执行下面的段落 2
    语句组 2
    :
    %可以按照这样的结构设置多种转移条件
else            %上面的条件均不满足时,执行下面的段落
    语句组 n + 1
end
```

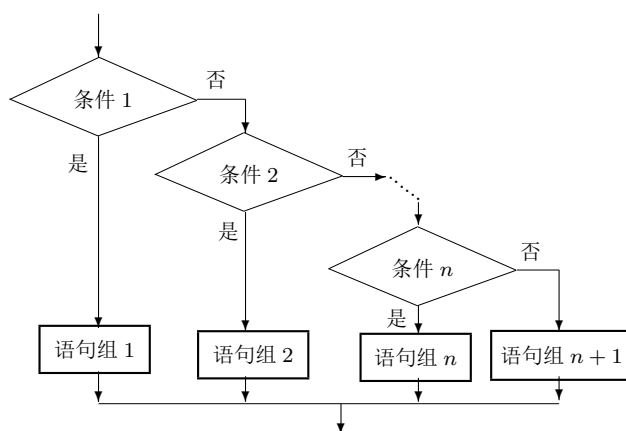


图 2-2 转移结构的示意图

例 2-16 用 for 循环和 if 语句的形式求解例 2-14 的问题。

解 例 2-14 中提及只用 for 循环结构不便于实现求出和式大于 10000 的最小 i 值, 利用该结构必须配合 if 语句结构才能实现

```
>> s=0; for i=1:10000, s=s+i; if s>10000, break; end, end
```

可见, 这样的结构较烦琐, 不如直接使用 while 结构直观、方便。

2.3.3 开关结构

开关语句的示意图如图 2-3 所示。该语句的基本结构为

```
switch 开关表达式
case 表达式 1, 语句段 1
case {表达式 2, 表达式 3, ..., 表达式 m}, 语句段 2
    ⋮
otherwise, 语句段 n
end
```

其中, 开关语句的关键是对“开关表达式”值的判断, 当开关表达式的值等于某个 case 语句后面的条件时, 程序将转移到该组语句中执行, 执行完成后程序转出开关体继续向下执行。

在使用开关语句结构时应该注意下面几点:

(1) 当开关表达式的值等于表达式 1 时, 将执行语句段 1, 执行完语句段 1 后将转出开关体, 而无需像 C 语言那样在下一个 case 语句前加 break 语句。所以本结构在这点上和 C 语言是不同的。

(2) 当需要在开关表达式满足若干个表达式之一时执行某一程序段, 则应该把这样的一些表达式用大括号括起来, 中间用逗号分隔。事实上, 这样的结构是 MATLAB 语言定义的单元结构。

(3) 当前面枚举的各个表达式均不满足时, 则将执行 otherwise 语句后面的语句段, 此

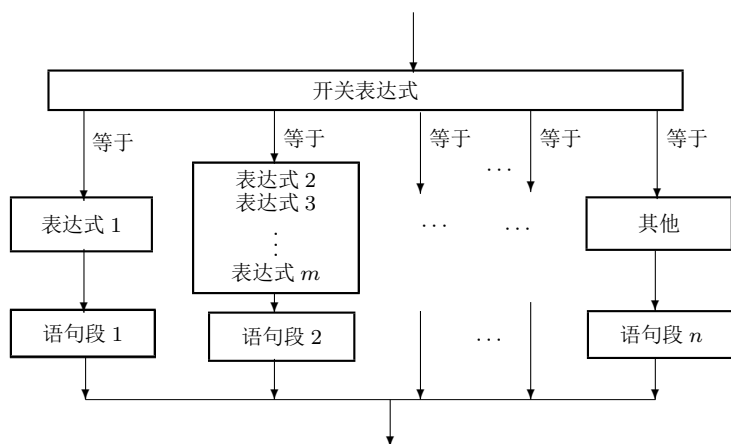


图 2-3 开关结构的示意图

语句等价于 C 语言中的 `default` 语句。

(4) 程序的执行结果和各个 `case` 语句的次序是无关的。当然这也不是绝对的, 当两个 `case` 语句中包含同样的条件时, 执行结果则和这两个语句的顺序有关。

(5) 在 `case` 语句引导的各个表达式中, 不要用重复的表达式, 否则列在后面的开关通路将永远也不能执行。

2.3.4 试探结构

MATLAB 语言提供了一种新的试探式语句结构, 其调用格式如下

```
try, 语句段 1, catch, 语句段 2, end
```

本语句结构首先试探性地执行语句段 1, 如果在此段语句执行过程中出现错误, 则将错误信息赋给保留的 `lasterr` 变量, 并终止这段语句的执行, 转而执行语句段 2 中的语句。这种新的语句结构是 C 等语言中所没有的。试探性结构在实际编程中还是很实用的, 例如可以将一段不保险但速度快的算法放到 `try` 段落中, 而将一个保险的程序放到 `catch` 段落中, 这样就能保证原始问题的求解更加可靠, 且可能使程序高速执行。该结构的另外一种应用是, 在编写通用程序时, 某算法可能出现失效的现象, 这时在 `catch` 语句段说明错误的原因。

2.4 函数编写与调试

MATLAB 下提供了两种源程序文件格式。其中一种是普通的 ASCII 码构成的文件, 在这样的文件中包含一族由 MATLAB 语言所支持的语句, 它类似于 DOS 下的批处理文件, 这种文件称作 M-脚本文件 (M-script, 本书中将其简称为 M-文件), 它的执行方式很简单, 用户只需在 MATLAB 的提示符 `>>` 下键入该 M-文件的文件名, 这样 MATLAB 就会自动执行该 M 文件中的各条语句。M-文件只能对 MATLAB 工作空间中的数据进行处理, 文件中所有语句的执行结果也完全返回到工作空间中。M-文件格式适用于用户所需要立即得到结果的小规模运算。

例 2-17 考虑一个实际的例子。在例 2-14 中编写一个简单的程序,可以求出和式大于 10000 的最小 m ,所以若想分别求出大于 20000, 30000 的 m_i 值,分别改变程序的限制值 10000,将其设置成 20000、30000 就可以满足要求,但这样做还是很繁杂的。如果能建立一种机制,或建立一个程序模块,给它输入 20000 的值就能返回满足它的 m_i 值,无疑这样的要求是很合理的。

在实际的 MATLAB 程序设计中,前面的一种修改程序本身的方法为 M-文件的方法,而后一种方法为函数的基本功能。后面将继续介绍函数的编写与应用。

文件名的命名规则与变量名一致,必须由字母引导,且现在的版本区分大小写。这里有两点必须引起注意:为了避免以后的麻烦甚至 MATLAB 函数冲突,建议在拟起函数名之前先用 `which` 命令查一下有无此名字的函数,如果有则需要重新起名。此外,应该尽量避免起过于简单的名字,如 `i.m`、`A.m` 等,因为起这样的名字可能和以后用的变量名冲突。

M-函数格式是 MATLAB 程序设计的主流,在实际编程中,不建议使用 M-脚本文件格式编程。本节将着重介绍 MATLAB 函数的编写方法与技巧。

2.4.1 MATLAB 语言函数的基本结构

MATLAB 的 M-函数是由 `function` 语句引导的,其基本结构如下

```
function [返回变量列表] = 函数名(输入变量列表)
    注释说明语句段,由百分号%引导
    输入、返回变量格式的检测
    函数体语句
```

这里输入和返回变量的实际个数分别由 `nargin` 和 `nargout` 两个 MATLAB 保留变量来给出,只要进入该函数, MATLAB 就将自动生成这两个变量。

返回变量如果多于 1 个,则应该用方括号将它们括起来,否则可以省去方括号。多个输入变量或返回变量之间用逗号分隔。注释语句段的每行语句都应该由百分号%引导,百分号后面的内容不执行,只起注释作用。用户采用 `help` 命令则可以显示出注释语句段的内容。此外,从规范编程的角度看,输入变量的个数与类型检测也是必要的。如果输入或返回变量格式不正确,则应该给出相应的提示。

从系统的角度来说, MATLAB 函数是一个变量处理单元,它从主调函数接收变量,对之进行处理后,将结果返回到主调函数中。除了输入和输出变量外,其他在函数内部产生的所有变量都是局部变量,在函数调用结束后这些变量均将消失。这里将通过下面的例子来演示函数编程的格式与方法。

例 2-18 先考虑例 2-17 中要求的 M-函数实现。根据要求,可以选择实际的输入变量为 k ,返回的变量为 m 和 s ,其中 s 为 m 项的和,这样就可以编写出该函数为

```
function [m,s]=findsum(k)
s=0; m=0; while (s<=k), m=m+1; s=s+m; end
```

编写了函数,就可以将其存为 `findsum.m` 文件,这样就可以在 MATLAB 环境中对不同的 k 值调用该函数了。例如,若想求出大于 145323 的最小 m 值,则可以得出如下命令,这时得出的结果为

$m_1 = 539, s_1 = 145530$

```
>> [m1,s1]=findsum(145323)
```

可见,这样的调用格式很灵活,无需修改程序本身就可以很容易地调用函数,得出所需的结果,所以建议采用这样的方法进行编程。

例 2-19 假设想编写一个函数生成 $n \times m$ 阶的 Hilbert 矩阵,它的第 i 行第 j 列的元素值为 $h_{i,j} = 1/(i+j-1)$ 。想在编写的函数中实现下面几点:

- (1) 如果只给出一个输入参数,则会自动生成一个方阵,即令 $m = n$;
- (2) 在函数中给出合适的帮助信息,包括基本功能、调用方式和参数说明;
- (3) 检测输入和返回变量的个数,如果有错误则给出错误信息。

解 其实在编写程序时详细给出注释语句,养成一个好的习惯,无论对程序设计者还是对程序的维护者、使用者都是大有裨益的。根据上面的要求,可以编写一个 MATLAB 函数 myhilb(), 文件名为 myhilb.m, 并应该放到 MATLAB 的路径下。

```
function A=myhilb(n, m)
%MYHILB 本函数用来演示 MATLAB 语言的函数编写方法
%      A=MYHILB(N, M) 将产生一个 N 行 M 列的 Hilbert 矩阵 A
%      A=MYHILB(N) 将产生一个 NxN 的方 Hilbert 阵 A
%See also: HILB

% Designed by Professor Dingyu XUE, Northeastern University, PRC
%      5 April, 1995, Last modified by DYX at 30 July, 2001
if nargin>1, error('Too many output arguments.');
```

在这段程序中,由%引导的部分是注释语句,通常用来给出一段说明性的文字来解释程序段落的功能和变量含义等。由前面的第(1)点要求,首先测试输入的参数个数,如果个数为 1 (即 nargin 的值为 1),则将矩阵的列数 m 赋成 n 的值,从而产生一个方阵。如果输入或返回变量个数不正确,则函数前面的语句将自动检测,并显示出错误信息。后面的双重 for 循环语句依据前面给出算法来生成一个 Hilbert 矩阵。给出如下命令

```
>> help myhilb
```

此函数的联机帮助信息可以显示如下

```
MYHILB 本函数用来演示 MATLAB 语言的函数编写方法
      A=MYHILB(N, M) 将产生一个 N 行 M 列的 Hilbert 矩阵 A
      A=MYHILB(N) 将产生一个 NxN 的方 Hilbert 阵 A
See also: HILB
```

注意,这里只显示了程序及调用方法,而没有把该函数中有关作者的信息显示出来。对照前面的函数可以立即发现,因为在作者信息的前面给出了一个空行,所以可以容易地得出结论,如果想使一段信息可以用 help 命令显示出来,在它前面不应该加空行,即使想在 help 中显示一个空行,这个空

行也应该由%来引导。

有了函数之后,可以采用下面的各种方法来调用它,并产生出所需的结果

```
>> A1=myhilb(3,4), A2=myhilb(4) % 不同的调用格式产生不同的结果矩阵
```

这样可以得出

$$A_1 = \begin{bmatrix} 1 & 0.5 & 0.33333 \\ 0.5 & 0.33333 & 0.25 \\ 0.33333 & 0.25 & 0.2 \\ 0.25 & 0.2 & 0.16667 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 0.5 & 0.33333 & 0.25 \\ 0.5 & 0.33333 & 0.25 & 0.2 \\ 0.33333 & 0.25 & 0.2 & 0.16667 \\ 0.25 & 0.2 & 0.16667 & 0.14286 \end{bmatrix}$$

例 2-20 MATLAB 函数是可以递归调用的,亦即在函数的内部可以调用函数自身。试用递归调用的方式编写一个求阶乘 $n!$ 的函数。

解 考虑求阶乘 $n!$ 的例子。由阶乘定义可见 $n! = n(n-1)!$, 这样, n 的阶乘可以由 $n-1$ 的阶乘求出, 而 $n-1$ 的阶乘可以由 $n-2$ 的阶乘求出, 依次类推, 直到计算到已知的 $1! = 0! = 1$, 从而能建立起递归调用的关系。为了节省篇幅起见, 这里略去了注释行段落。

```
function k=my_fact(n)
if nargin~=1, error('Error: Only one input variable accepted'); end
if abs(n-floor(n))>eps | n<0 % judge whether n is a non-negative integer
    error('n should be a non-negative integer');
end
if n>1, k=n*my_fact(n-1); % 若 n>1, 则采用递归调用
elseif any([0 1]==n), k=1; end % 0!=1!=1, 函数的出口
```

可以看出, 该函数首先判定 n 是否为非负整数, 如果不是则给出错误信息, 如果是, 则在 $n > 1$ 时递归调用该程序自身, 若 $n = 1$ 或 0 时则直接返回 1 。由 `my_fact(11)` 格式调用该函数则立即可以得出阶乘 $11! = 39916800$ 。

其实 MATLAB 提供了求取阶乘的函数 `factorial()`, 其核心算法为 `prod(1:n)`, 从结构上更简单、直观, 速度也更快。

例 2-21 试比较递归算法和循环算法在 Fibonacci 数列中应用的优劣。

解 递归算法无疑是解决一类问题的有效算法, 但不宜滥用。现在考虑一个反例, 考虑 Fibonacci 数列, $a_1 = a_2 = 1$, 第 k 项 ($k = 3, 4, \dots$) 可以写成 $a_k = a_{k-1} + a_{k-2}$, 这样很自然想到使用递归调用算法编写相应的函数, 该函数设置 $k = 1, 2$ 时出口为 1 , 这样函数清单如下

```
function a=my_fibo(k)
if k==1 | k==2, a=1; else, a=my_fibo(k-1)+my_fibo(k-2); end
```

该函数中略去了检测 k 是否为正整数的语句。如果想得到第 25 项, 则需要给出如下的语句, 同时测出运行该函数所运行的时间为 0.99 s, MATLAB 早期版本耗时将比新版本多得多。

```
>> tic, my_fibo(25), toc
```

如果用递归方法求 $k = 30$ 的运算时间将达到 10.26 s, 求解 $k = 40$ 问题则需数小时的时间, 计算量呈几何级数增长。现在改用循环语句结构求解 $k = 100$ 时的项, 耗时仅 0.0002 s。

```
>> tic, a=[1,1]; for k=3:100, a(k)=a(k-1)+a(k-2); end, toc
```

可见, 一般循环方法用极短的时间就能算出递归调用不可能解决的问题, 所以在实际应用时

应该注意不能滥用递归调用格式。进一步观察结果可见,由于该序列的值过大,用上述的双精度算法并不能得出整个序列的精确结果,所以应该采用符号运算数据类型,例如将 $a = [1 \ 1]$ 修改成 $a = \text{sym}([1 \ 1])$,这样可以得出数值解难以达到的精度,如 $a_{100} = 354224848179261915075$ 。

2.4.2 可变输入输出个数的处理

下面将介绍单元数组的一个重要应用——如何建立起无限个输入或返回变量的函数调用格式。应该指出的是,当前很多 MATLAB 语言函数均采用本方法编写。

例 2-22 MATLAB 提供的 `conv()` 函数可以用来求两个多项式的乘积。对于多个多项式的连乘,则不能直接使用此函数,而需要用该函数嵌套使用,这样在表示很多多项式连乘时相当麻烦。试编写一个 MATLAB 函数,使得它能直接处理任意多个多项式的乘积问题。

解 可以用单元数组的形式来编写一个函数 `convs()`,专门解决多个多项式连乘的问题

```
function a=convs(varargin)
a=1; for i=1:length(varargin), a=conv(a,varargin{i}); end
```

这时,所有的输入变量列表由单元变量 `varargin` 表示。相应地,如有需要,也可以将返回变量列表用一个单元变量 `varargout` 表示。在这样的表示下,理论上就可以处理任意多个多项式的连乘问题了。例如可以用下面的格式调用该函数

```
>> P=[1 2 4 0 5]; Q=[1 2]; F=[1 2 3]; D=convs(P,Q,F)
E=conv(conv(P,Q),F) % 若采用 conv() 函数,则需要嵌套调用
G=convs(P,Q,F,[1,1],[1,3],[1,1])
```

可以得出 $D=E=[1, 6, 19, 36, 45, 44, 35, 30]^T$, $G=[1, 11, 56, 176, 376, 578, 678, 648, 527, 315, 90]^T$ 。

2.4.3 匿名函数与 `inline()` 函数

有时为了描述某个数学函数的方便,可以用匿名函数来直接编写该数学函数,形式相当于前面介绍的 M-函数,但无需编写一个真正的 M-文件。匿名函数的基本格式为

```
f=@(变量列表)函数内容, 例如, f=@(x,y)sin(x.^2+y.^2)
```

此外,该函数还允许直接使用 MATLAB 工作空间中的变量。例如,若在 MATLAB 工作空间内已经定义了 a, b 变量,则匿名函数可以用 $f=@(x,y)a*x.^2+b*y.^2$ 的格式定义数学关系式 $f(x,y)=ax^2+by^2$,这样无需将 a, b 作为附加参数在输入变量里表示出来,所以使得数学函数的定义更加方便。注意,在匿名函数定义时, a, b 的值以当前 MATLAB 工作空间中的数值为主,在定义该匿名函数后, a, b 的值再发生变化,则在匿名函数中的值将不随着改变,所以使用工作空间变量时应该多加小心。

早期版本中的 `inline()` 函数功能类似于匿名函数,但现在看来其使用不方便,也不支持 MATLAB 工作空间中变量的直接使用,运行效率也远远低于匿名函数,所以这里只给出其调用格式 `fun=inline('函数内容',自变量列表)`。例如, $f(x,y)=\sin(x^2+y^2)$ 可以用 `f=inline('sin(x.^2+y.^2)','x','y')` 直接定义。

2.4.4 伪代码与代码保密处理

MATLAB 的伪代码 (pseudo code) 技术的目的有两个: 一是能提高程序的执行速度, 因为采用了伪代码技术, MATLAB 将 .m 文件转换成能立即执行的代码, 所以在程序实际执行时, 省去了再转换的过程, 从而能使得程序的速度加快。由于 MATLAB 本身的转换过程也很快, 所以在一般程序执行时速度加快的效果并不是很明显。然而当执行较复杂的图形界面程序时, 伪代码技术的应用便能很明显地加快程序执行的速度。二是伪代码技术能把可读的 ASCII 码构成的 .m 文件转换成一种二进制代码, 从而使得其他用户无法读取其中的语句, 从而对源代码起到某种保密作用。

MATLAB 提供了 `pcode` 命令来将 .m 文件转换成伪代码文件, 伪代码文件后缀名为 .p。如果想把某文件 `mytest.m` 转换成伪代码文件, 则可以使用 `pcode mytest` 命令; 若想让生成的 .p 文件也位于和原 .m 文件相同的目录下, 则可以使用 `pcode mytest -inplace` 命令。如果想把整个目录下的 .m 文件全转换为 .p 文件, 则首先用 `cd` 命令进入该目录, 然后输入 `pcode *.m`, 若原文件无语法错误, 就可以在本目录下将 .m 文件全部转换为 .p 文件; 若存在语法错误, 则将中止转换, 并给出错误信息。用户可以通过这样的方法发现自己程序中存在的所有语法错误。如果同时存在同名的 .m 文件和 .p 文件, 则 .p 文件的执行优先。

用户一定要在安全的位置保留 .m 源文件, 不能轻易删除, 因为 .p 文件是不可逆的。

2.5 二维图形绘制

图形绘制与可视化是 MATLAB 语言的一大特色。MATLAB 提供了一系列直观、简单的二维图形和三维图形绘制命令与函数, 可以将实验结果和仿真结果用可视的形式显示出来。本节将介绍各种各样的图形绘制方法。

2.5.1 二维图形绘制基本语句

假设用户已经获得了一些实验数据。例如, 已知各个时刻 $t = t_1, t_2, \dots, t_n$ 和在这些时刻的函数值 $y = y_1, y_2, \dots, y_n$, 则可以将这些数据输入到 MATLAB 环境中, 构成向量 $t = [t_1, t_2, \dots, t_n]$ 和 $y = [y_1, y_2, \dots, y_n]$, 如果用户想用图形的方式表示二者之间的关系, 则给出 `plot(t, y)` 即可绘制二维图形。可以看出, 该函数的调用是相当直观的。这样绘制出的“曲线”实际上是给出各个数值点间的折线, 如果这些点足够密, 则看起来就是曲线了, 故以后将称之为曲线。在实际应用中, `plot()` 函数的调用格式还可以进一步扩展:

(1) t 仍为向量, 而 y 为矩阵给出如下, 则将在同一坐标系下绘制 m 条曲线, 每一行和 t 之间的关系将绘制出一条曲线。注意, 这时要求 y 矩阵的列数应该等于 t 的长度。

$$y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix}$$

(2) t 和 y 均为矩阵, 且假设 t 和 y 矩阵的行和列数均相同, 则将绘制出 t 矩阵每行和

y 矩阵对应行之间关系的曲线。

(3) 假设有对这样的向量或矩阵, $(t_1, y_1), (t_2, y_2), \cdots, (t_m, y_m)$, 则可以用下面的语句直接绘制出各自对应的曲线

```
plot(t1, y1, t2, y2, ⋯, tm, ym)
```

(4) 曲线的性质, 如线型、粗细、颜色等, 还可以使用下面的命令进行指定

```
plot(t1, y1, 选项 1, t2, y2, 选项 2, ⋯, tm, ym, 选项 m)
```

其中, “选项” 可以按表 2-3 中说明的形式给出, 其中的选项可以进行组合。例如, 若想绘制红色的点划线, 且每个转折点上用五角星表示, 则选项可以使用 `'r-.pentagram'` 组合形式。

表 2-3 MATLAB 绘图命令的各种选项

曲线线型		曲线颜色		标记符号	
选项	意义	选项	意义	选项	意义
'-'	实线	'b'	蓝色	'c'	蓝绿色
'--'	虚线	'g'	绿色	'k'	黑色
'.'	点线	'm'	红紫色	'r'	红色
'-.'	点划线	'w'	白色	'y'	黄色
'none'	无线				
				'*'	星号
				'.'	点号
				'x'	叉号
				'v'	▽
				'^'	△
				'>'	▷
				'<'	◁
				'pentagram'	五角星
				'o'	圆圈
				'square'	□
				'diamond'	◇
				'hexagram'	六角星

绘制完二维图形后, 还可以用 `grid on` 命令在图形上添加网格线, 用 `grid off` 命令取消网格线; 另外用 `hold on` 命令可以保护当前的坐标系, 使得以后再使用 `plot()` 函数时将新的曲线叠印在原来的图上, 用 `hold off` 则可以取消保护状态; 用户可以使用 `title()` 函数在绘制的图形上添加标题, 还可以用 `xlabel()`、`ylabel()` 函数给 x 、 y 坐标轴添加标注。

例 2-23 试绘制出显函数方程 $y = \sin(\tan x) - \tan(\sin x)$ 在 $x \in [-\pi, \pi]$ 区间内的曲线。

解 解决这种问题的最简捷方法是采用下面的语句直接绘制

```
>> x=[-pi : 0.05: pi]; % 以 0.05 为步距构造自变量向量
y=sin(tan(x))-tan(sin(x)); plot(x,y) % 求出并绘制各个点上的函数值
```

这些语句可以绘制出该函数的曲线, 如图 2-4(a) 所示。

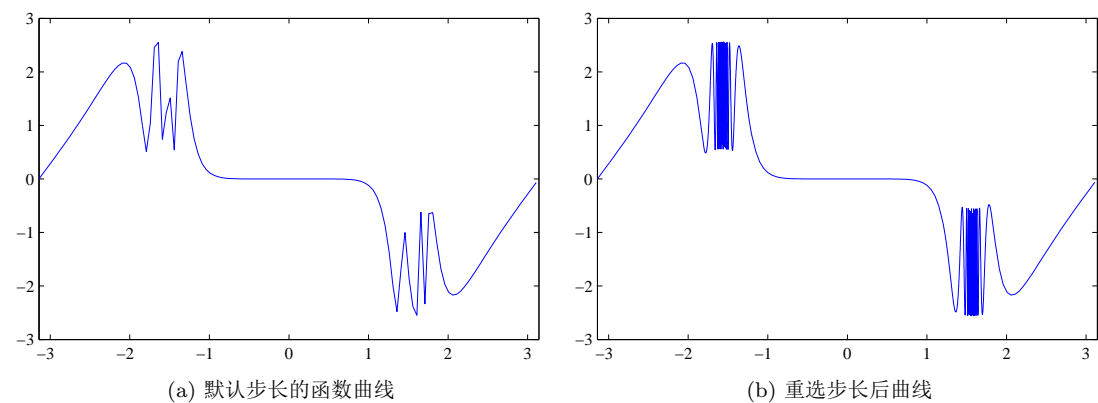


图 2-4 给定函数的曲线表示

从得出的曲线可以看出,在 $x \in (-1.8, -1.2)$ 及 $x \in (1.2, 1.8)$ 两个子区间内图形较粗糙,应该在
这些区间加密自变量选择点,这样可以将上述的语句修改为

```
>> x=[-pi:0.05:-1.8,-1.799:.001:-1.2, -1.2:0.05:1.2,...  
      1.201:0.001:1.8, 1.81:0.05:pi]; % 以变步距方式构造自变量向量  
y=sin(tan(x))-tan(sin(x)); plot(x,y) % 求出并绘制各个点上的函数值
```

这样将得出如图 2-4 (b) 所示的曲线。可见,这样得出的曲线在快变化区域内表现良好。

例 2-24 绘制出饱和非线性特性函数 $y = \begin{cases} 1.1 \operatorname{sign}(x), & |x| > 1.1 \\ x, & |x| \leq 1.1 \end{cases}$ 的曲线。

解 当然用 if 语句可以很容易求出各个 x 点上的 y 值。但这里将考虑另外一种有效的实现方法。如果构造了 x 向量,则关系表达式 $x > 1.1$ 将生成一个和 x 一样长的向量,在满足 $x_i > 1.1$ 的点上,生成向量的对应值为 1,否则为 0,根据这样的想法,则可以用下面的语句绘制出分段函数的曲线,如图 2-5 所示

```
>> x=[-2:0.02:2]; y=1.1*sign(x).*(abs(x)>1.1) + x.*(abs(x)<=1.1); plot(x,y)
```

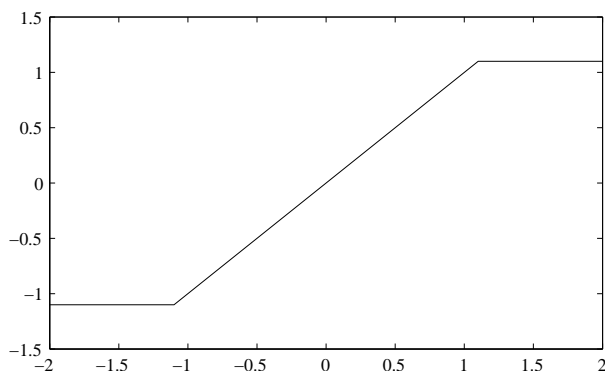


图 2-5 分段函数曲线绘制

在这样的分段模型描述中,注意不要将某个区间重复表示。例如,不能将给出的语句中最后一个条件表示成 $1.1*(x \geq 1.1)$,否则因为第 2 项中也有 $x_i = 1.1$ 的选项,将使得 $x_i = 1.1$ 点函数求取重复,得出错误的结果。

另外,由于 plot() 函数只将给定点用直线连接起来,分段线性的非线性曲线可以由有限的几个转折点来表示,该语句能得出和图 2-5 完全一致的结果

```
>> plot([-2,-1.1,1.1,2],[-1.1,-1.1,1.1,1.1])
```

在 MATLAB 绘制的图形中,每条曲线是一个对象,坐标轴是一个对象,而图形窗口还是一个对象,每个对象都有不同的属性,用户可以通过 set() 函数设置对象的属性,还可以用 get() 函数获得对象的某个属性。这两个语句广泛应用于图形用户界面的设计,它们的调用格式为

```
set(句柄,'属性名 1',属性值 1,'属性名 2',属性值 2,...)  
v = get(句柄,'属性名')
```

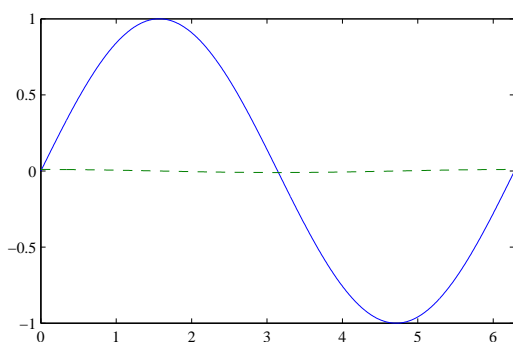
2.5.2 多纵轴曲线的绘制

假设有两组数据,如果它们幅值相差比较悬殊,尽管可以将它们在一个坐标系下绘制出来,这样绘制会使得幅值小的曲线可读性较差,这时可以考虑使用 `plotyy()` 函数将它们绘制出来。下面通过例子演示这样的曲线绘制方法。

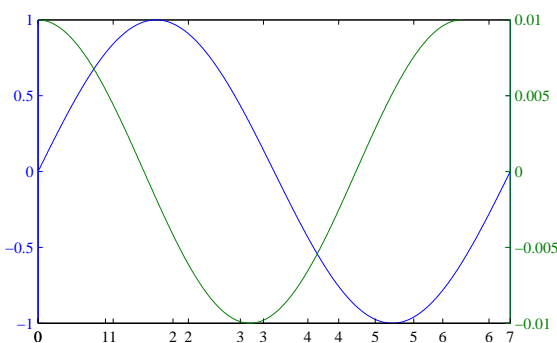
例 2-25 试将 $y_1 = \sin x$ 与 $y_2 = 0.01 \cos x$ 绘制出来。

解 直接采用下面语句可以绘制出两条函数曲线,如图 2-6(a) 所示。由于两条曲线的幅值相差太悬殊, y_2 曲线的可读性很差,所以不宜采用这样的绘制方法

```
>> x=0:0.01:2*pi; y1=sin(x); y2=0.01*cos(x); plot(x,y1,x,y2,'--')
```



(a) 不恰当的直接绘制结果



(b) 双纵轴曲线的绘制

图 2-6 两条幅值悬殊曲线的绘制

对这样的问题应该采用 `plotyy()` 函数绘制出双纵轴曲线,如图 2-6(b) 所示

```
>> plotyy(x,y1,x,y2)
```

在某些特殊的应用中可能还需要绘制三、四纵轴的曲线,可以考虑下载 MathWorks 的 File Exchange 网站下相应的实用程序,如 `plotyyy()` [2]、`plot4y()` [3] 等,利用 `plotxx()` 函数还可以绘制双 x 轴的曲线 [4]。

2.5.3 其他二维图形绘制语句

除了标准的二维曲线绘制之外, MATLAB 还提供了具有各种特殊意义的图形绘制函数,其常用调用格式如表 2-4 所示。其中,参数 x 、 y 分别表示横、纵坐标绘图数据, c 表示颜色选项, y_m 、 y_M 表示误差图的上下限向量。当然,随着输入参数个数及类型的不同,各个函数的绘图形式也有所区别。下面将通过例子来演示各个绘图函数的效果。

例 2-26 试用极坐标绘制函数 `polar()` 绘制出 $\rho = 5 \sin(4\theta/3)$ 和 $\rho = 5 \sin(\theta/3)$ 的极坐标曲线。

解 由极坐标方程的数学表达式可以立即得出结论,这两个函数的周期均为 6π ,所以若想绘制极坐标曲线,则应该先构造一个 θ 向量,然后求出 ρ 向量,调用 `polar()` 函数就可以立即绘制出所需的极坐标曲线,分别如图 2-7(a)、(b) 所示

```
>> theta=0:0.01:6*pi; rho=5*sin(4*theta/3); polar(theta,rho)
figure; rho=5*sin(theta/3); polar(theta,rho)
```

表 2-4 MATLAB 提供的特殊二维曲线绘制函数

函数名	意义	常用调用格式	函数名	意义	常用调用格式
bar()	二维条形图	bar(<i>x,y</i>)	comet()	彗星状轨迹图	comet(<i>x,y</i>)
compass()	罗盘图	compass(<i>x,y</i>)	errorbar()	误差限图形	errorbar(<i>x,y,y_m,y_M</i>)
feather()	羽毛状图	feather(<i>x,y</i>)	fill()	二维填充函数	fill(<i>x,y,c</i>)
hist()	直方图	hist(<i>y,n</i>)	loglog()	对数图	loglog(<i>x,y</i>)
polar()	极坐标图	polar(<i>x,y</i>)	quiver()	引力线图	quiver(<i>x,y</i>)
stairs()	阶梯图形	stairs(<i>x,y</i>)	stem()	火柴杆图	stem(<i>x,y</i>)
semilogx()	<i>x</i> -半对数图	semilogx(<i>x,y</i>)	semilogy()	<i>y</i> -半对数图	semilogy(<i>x,y</i>)

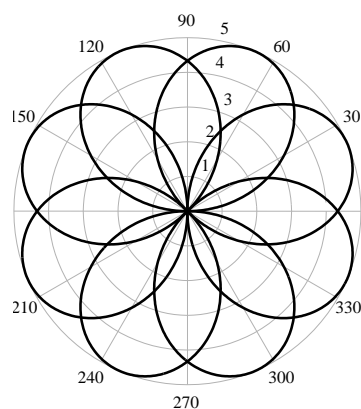
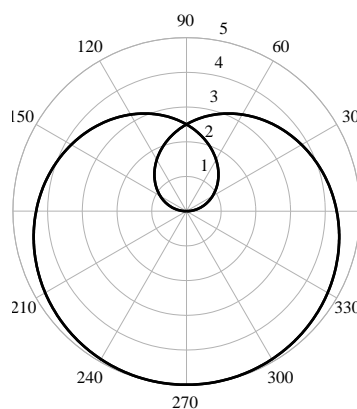
(a) $\rho = 5 \sin(4\theta/3)$ (b) $\rho = 5 \sin(\theta/3)$

图 2-7 极坐标曲线

例 2-27 以正弦数据为例,试在同一窗口的不同区域用不同的绘图方式绘制出相应的曲线。

解 可以用下面的各种语句绘制出如图 2-8 所示的曲线。其中,subplot() 函数可以将图形窗口分为若干块,在某一块内绘制图形。在函数调用时,第 1 个 2 表示将窗口分为 2 行,第 2 个 2 表示将窗口分为 2 列,第 3 个参数指定绘图的位置。

```
>> t=0:.2:2*pi; y=sin(t);           % 先生成绘图用数据
subplot(2,2,1), stairs(t,y)          % 分割窗口,在左上角绘制阶梯曲线
subplot(2,2,2), stem(t,y)            % 火柴杆曲线绘制
subplot(2,2,3), bar(t,y)             % 直方图绘制
subplot(2,2,4), semilogx(t,y)        % 横坐标为对数的曲线
```

2.5.4 隐函数绘制及应用

隐函数即满足 $f(x,y) = 0$ 方程的 x,y 之间的关系式。用前面介绍的曲线绘制方法显然会有问题。例如,很多隐函数无法求出 x,y 之间的显式关系,所以无法先定义一个 x 向量再求出相应的 y 向量,从而不能采用 plot() 函数来绘制曲线。另外,即使能求出 x,y 之间的显式关系,但不是单值函数,则绘制起来也是很麻烦的。MATLAB 提供的 ezplot() 函数可

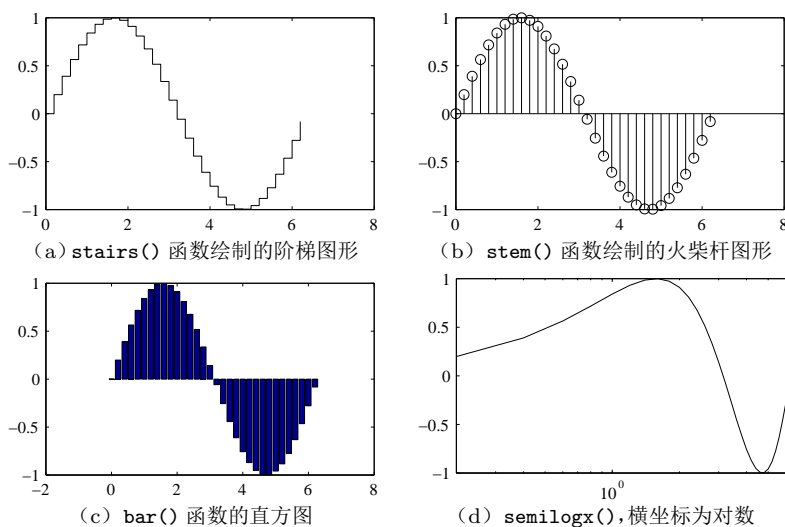


图 2-8 不同的二维曲线绘制函数

以直接绘制隐函数曲线,该函数的调用格式为 `ezplot(隐函数表达式)`。下面将通过例子来演示该函数的使用方法。

例 2-28 试绘制出隐函数 $f(x, y) = x^2 \sin(x + y^2) + y^2 e^{x+y} + 5 \cos(x^2 + y) = 0$ 的曲线。

解 从给出的函数可见,无法用解析的方法写出该函数,所以不能用前面给出的 `plot()` 函数绘制出该函数的曲线。对这样的隐函数,可以给出如下的 MATLAB 命令,并将得出如图 2-9(a) 所示的隐函数曲线

```
>> ezplot('x^2 *sin(x+y^2) +y^2*exp(x+y)+5*cos(x^2+y)')
```

上面的语句将自动选择 x 轴的范围,亦即函数的定义域,如果想改变定义域,则可以用下面的语句给出命令,并得出如图 2-9(b) 所示的隐函数曲线

```
>> ezplot('x^2 *sin(x+y^2) +y^2*exp(x+y)+5*cos(x^2+y)', [-10 10])
```

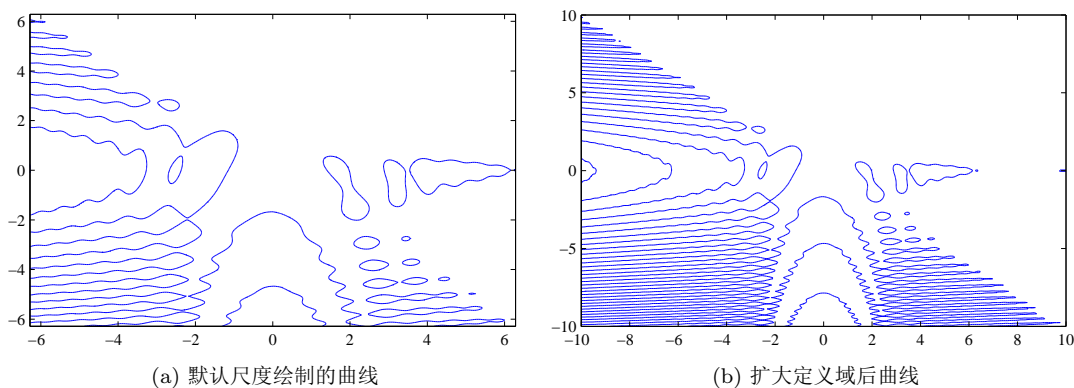
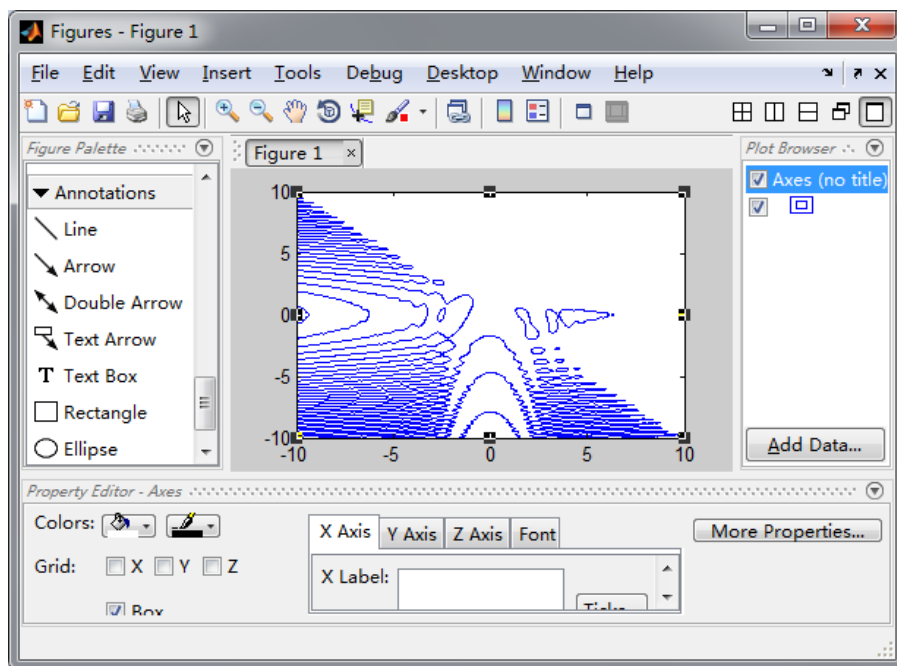


图 2-9 隐函数曲线绘制

2.5.5 图形修饰

MATLAB 提供了强大的图形修饰功能,其编辑窗口如图 2-10(a) 所示,其工具栏允许用户在图形上添加箭头、文字、双向箭头、椭圆、方框等新的标记,大大提高了图形修饰的功能。此外还可以对图形进行局部放大、三维图形的旋转等。





(a) 图形编辑窗口



(b) 新 MATLAB 版本下的图形编辑工具栏

图 2-10 MATLAB 的图形编辑界面

图形编辑主要有 3 方面的内容,图形窗口左侧的部分对应于 View 菜单下的 Figure Palette,其中用户可以选择这里的工具在图形上添加箭头、各类文字及椭圆等修饰,还可以添加二维、三维坐标系。图形窗口下面的窗口对应于该菜单的 Property Editor,允许修改选中对象的颜色、线形、字体等属性。右侧的窗口对应于 View 菜单的 Plot Browser,允许用户从图上选择图形元素进行编辑,还允许用户添加新的数据,在现有的图形上叠印新的图形。如果使用新版本的 MATLAB,将给出图形编辑工具栏,如图 2-10(b) 所示。在新版本下并不给出如图 2-10(a) 所示的编辑界面,只能用工具栏提供的工具修饰图形。

图形窗口的工具栏提供了用鼠标选择图形上点坐标的按钮,可以用其代替早期版本的 `ginput()` 函数,读出并显示曲线上点坐标的信息,该功能更适合于数学问题图解方法的实现。单击工具栏的图形旋转按钮,则可以将二维图形用三维图形表示,如图 2-11 所示。

如果单击 Text Box 工具,则用鼠标在图形上单击则可以确定文字添加的位置,然后直

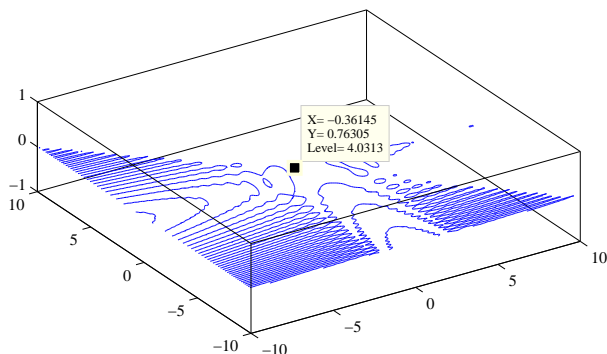


图 2-11 二维图形的三维表示

接输入字符串即可。字符串可以用普通的字母和文字表示,也可以用 L^AT_EX 的格式描述数学公式。单击 Line、Arrow 工具还可以在图形上叠印线段和箭头等。

L^AT_EX 是一个著名的科学文档排版系统, MATLAB 支持的只是其中一个子集,这里简单介绍在 MATLAB 图形窗口中添加 L^AT_EX 描述的数学公式的方法:

(1) 特殊符号是由 \ 引导的命令定义的, MATLAB 支持的特殊符号在表 2-5 中给出。

(2) 上下标分别用 ^ 和 _ 表示,例如 $a_2^2 + b_2^2 = c_2^2$ 表示 $a_2^2 + b_2^2 = c_2^2$ 。如果需要表示多个上标,则需要用大括号括起,表示段落,例如 $a^A bc$ 命令表示 $a^A bc$,其中 A 为上标。如果想将 abc 均表示成 a 的上标,则需要给出命令 $a^{\{abc\}}$ 。

(3) 很多 L^AT_EX 常用命令是 MATLAB 图形窗口下不支持的,例如显示分式的 \frac 命令,所以在排版时建议采用 overpic 宏包,在图形上叠印 L^AT_EX 命令,得到最好的排版效果,并使得图形上公式的字体与正文保持一致。

L^AT_EX 科技文献排版系统是当今学术界最广泛使用的排版系统,具有 Word 类排版系统无可比拟的优越性,感兴趣的读者可以进一步阅读文献 [1] 等。

2.5.6 数据文件的读取与存储

MATLAB 提供了 save 和 load 命令,可以将工作空间中的变量存入指定文件,或从文件将数据读入工作空间。在 MATLAB 命令窗口给出 save 命令,则将当前 MATLAB 工作空间中所有的数据直接存入默认的 matlab.mat 文件,该文件是二进制文件,只能用 load 命令读出。如果想将工作空间中的 A、B、C 以默认的二进制形式存入 mydat.mat 文件,则可以直接给出 `save mydat A B C` 命令。

如果想将这三个变量以可读的 ASCII 码(纯文本文件)存入 mydat.dat 文件,则需要给出 `save /ascii mydat.dat A B C` 命令。

如果使用了长文件名或路径名,则采用 load 命令会出现问题,这时可以调用 load() 函数即可,可以调用 `X = load(文件名)` 将文件中的数据读入 X 变量。

MATLAB 还支持与 Excel 文件之间的数据交互,由 xlsread() 可以读入相关数据,其调用格式为 `X = xlsread(文件名,区域)`,其中“区域”为所需的矩形区域标记,如 'B5:C67'。另外,由 xlswrite() 函数可以将变量写入 Excel 文件。

表 2-5 图形窗口下可以直接使用的 T_EX 命令表

类别	c	T _E X 命令	c	T _E X 命令	c	T _E X 命令	c	T _E X 命令
小写 希腊 字符	α	\alpha	β	\beta	γ	\gamma	δ	\delta
	ϵ	\epsilon	ε	\varepsilon	ζ	\zeta	η	\eta
	θ	\theta	ϑ	\vartheta	ι	\iota	κ	\kappa
	λ	\lambda	μ	\mu	ν	\nu	ξ	\xi
	o	o	π	\pi	ϖ	\varpi	ρ	\rho
	ι	\iota	κ	\kappa	ϱ	\varrho	σ	\sigma
	ς	\varsigma	τ	\tau	υ	\upsilon	ϕ	\phi
	φ	\varphi	χ	\chi	ψ	\psi	ω	\omega
大写 希腊 字符	Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
	Ξ	\Xi	Π	\Pi	Σ	\Sigma	Υ	\Upsilon
	Φ	\Phi	Ψ	\Psi	Ω	\Omega		
常用 数学 符号	\aleph	\aleph	\prime	\prime	\forall	\forall	\exists	\exists
	\wp	\wp	\Re	\Re	\Im	\Im	∂	\partial
	∞	\infty	∇	\nabla	\surd	\surd	\angle	\angle
	\neg	\neg	\int	\int	\clubsuit	\clubsuit	\diamondsuit	\diamondsuit
	\heartsuit	\heartsuit	\spadesuit	\spadesuit				
二元 运算 符号	\pm	\pm	\cdot	\cdot	\times	\times	\div	\div
	\circ	\circ	\bullet	\bullet	\cup	\cup	\cap	\cap
	\vee	\vee	\wedge	\wedge	\otimes	\otimes	\oplus	\oplus
关系 数学 符号	\leq	\leq	\geq	\geq	\equiv	\equiv	\sim	\sim
	\subset	\subset	\supset	\supset	\approx	\approx	\subseteq	\subseteq
	\supseteq	\supseteq	\in	\in	\ni	\ni	\propto	\propto
	$ $	\mid	\perp	\perp				
箭头 符号	\leftarrow	\leftarrow	\uparrow	\uparrow	\Leftarrow	\Leftarrow	\Uparrow	\Uparrow
	\rightarrow	\rightarrow	\downarrow	\downarrow	\Rightarrow	\Rightarrow	\Downarrow	\Downarrow
	\leftrightarrow	\leftrightarrow	\updownarrow	\updownarrow				

例 2-29 Excel 文件 census.xls 包含某省的年度人口数, 其中第 B 列为年度, 第 C 列为人口数。有效数据从第 5 行到第 67 行。试将年度信息读入 t 向量, 并将人口数读入 p 向量。

解 由上述已知条件可以看出, 该文件的有效数据区域为第 B、C 列, 第 5 到 67 行, 故矩形数据区域可以表示成 'B5:C67', 由下面语句可以直接将所需数据读入 MATLAB 工作空间, 这样用列向量提取的方法则可以得出所需的 t 和 p 向量, 得出的年度人口曲线如图 2-12 所示。

```
>> X=xlsread('census.xls','B5:C67'); t=X(:,1); p=X(:,2); plot(t,p)
```

2.6 三维图形表示

2.6.1 三维曲线绘制

二维曲线绘制函数 plot() 可以扩展到三维曲线的绘制中。这时可以用 plot3() 函数绘制三维曲线。该函数的调用格式为

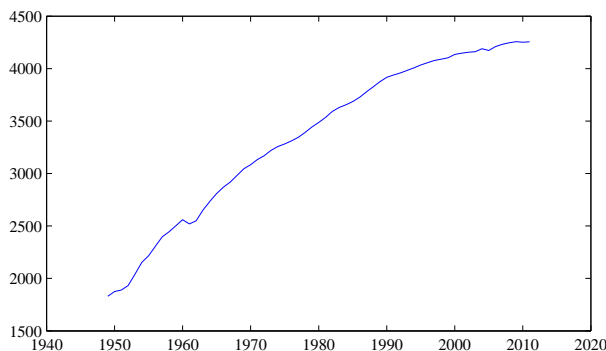


图 2-12 某省人口的年度曲线(单位:万人)

```
plot3(x,y,z)
```

```
plot3(x1,y1,z1,选项 1,x2,y2,z2,选项 2,...,xm,ym,zm,选项 m)
```

其中“选项”和二维曲线绘制的完全一致,如表 2-3 所示。相应地,类似于二维曲线绘制函数, MATLAB 还提供了其他的三维曲线绘制函数,如 `stem3()` 可以绘制三维火柴杆型曲线, `fill3()` 可以绘制三维的填充图形, `bar3()` 可以绘制三维的直方图等。

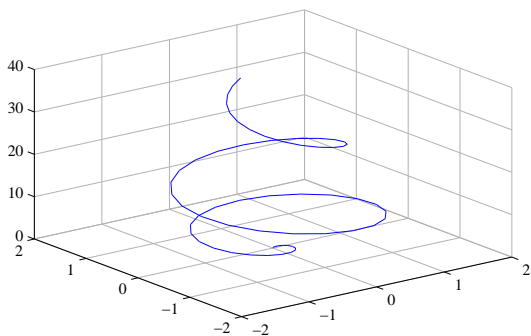
例 2-30 试绘制参数方程 $x(t) = t^3 e^{-t} \sin 3t$, $y(t) = t^3 e^{-t} \cos 3t$, $z = t^2$ 的三维曲线。

解 若想绘制该参数方程的曲线,可以先定义一个时间向量 t ,由其计算出 x , y , z 向量,并用函数 `plot3()` 绘制出三维曲线,如图 2-13(a) 所示。注意,这里应该采用点运算

```
>> t=0:.1:2*pi;           % 构造 t 向量,注意下面的点运算
    x=t.^3.*exp(-t).*sin(3*t); y=t.^3.*exp(-t).*cos(3*t); z=t.^2;
    plot3(x,y,z), grid % 三维曲线绘制
```

如果用 `stem3()` 函数绘制出火柴杆形曲线,如图 2-13(b) 所示

```
>> stem3(x,y,z); hold on; plot3(x,y,z), grid
```



(a) 三维曲线绘制

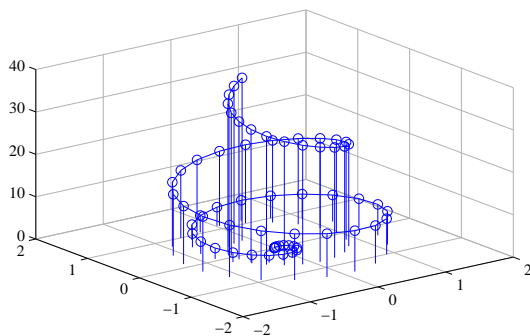
(b) `stem3()` 函数绘制的三维图形

图 2-13 三维曲线的绘制

2.6.2 三维曲面绘制

如果已知二元函数 $z = f(x, y)$,则可以绘制出该函数的三维曲面图。在绘制三维图之

前,应该先调用 `meshgrid()` 函数生成网格矩阵数据 x 和 y ,这样就可以按函数公式用点运算的方式计算出 z 矩阵,之后就可以用 `mesh()` 或 `surf()` 等函数进行三维图形绘制了。具体的函数调用格式为

```
[x,y] = meshgrid(v1,v2)      % 生成网格数据
z = ... ,如 z = x.*y        % 计算二元函数的 z 矩阵
surf(x,y,z) 或 mesh(x,y,z)  % mesh() 绘制网格图,surf() 绘制表面图
```

其中, v_1 和 v_2 为 x 轴和 y 轴的分隔方式。`surf()` 函数还可以返回曲面的句柄,这样就可以对得出的曲面进行进一步操作处理了。

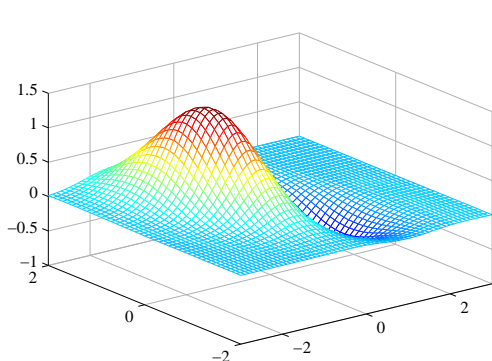
三维曲面还可以由其他函数绘制,如 `surfc()` 函数和 `surfl()` 函数可以分别绘制带有等高线和光照下的三维曲面,`waterfall()` 函数可以绘制瀑布形三维图形。在 MATLAB 下还提供了等高线绘制的函数,如 `contour()` 函数和三维等高线函数 `contour3()`,这里将通过例子介绍三维曲面绘制方法与技巧。

MATLAB 还提供了更简洁的简易绘图函数,如 `ezsurf()`、`ezmesh()`、`ezcontour()`、`ezcontourf()` 等函数,这些函数只需给出数学表达式即可绘制出所需图形,这些函数更利于绘制给定函数的三维图形。

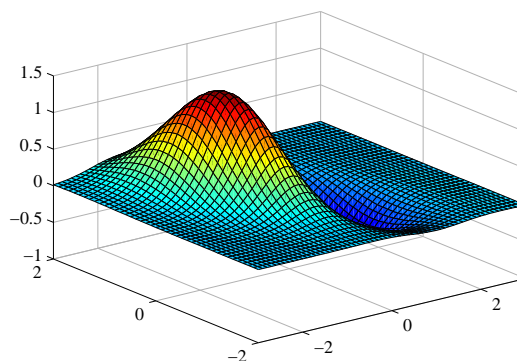
例 2-31 给出二元函数 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$, 试在 x - y 平面内选择一个区域,并绘制出该函数三维表面图形。

解 首先可以调用 `meshgrid()` 函数生成 x - y 平面的网格表示。该函数的调用意义十分明显,即可以产生一个横坐标起始于 -3 ,中止于 2 ,步距为 0.1 ,纵坐标起始于 -2 ,中止于 2 ,步距为 0.1 的网格分割。然后由上面的公式计算出曲面的 z 矩阵。最后调用 `mesh()` 函数来绘制曲面的三维表面网格图形,如图 2-14(a) 所示

```
>> [x,y]=meshgrid(-3:0.1:2,-2:0.1:2);
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); mesh(x,y,z)
```



(a) `mesh()` 函数绘制的网格图



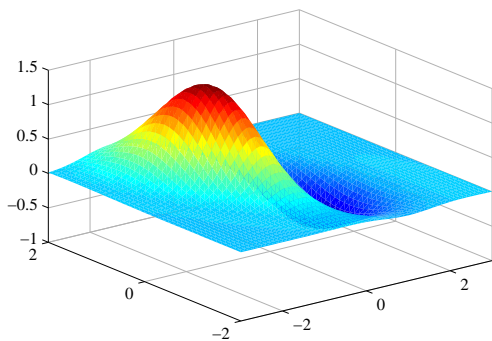
(b) `surf()` 函数绘制的表面图

图 2-14 给定函数的三维图表示

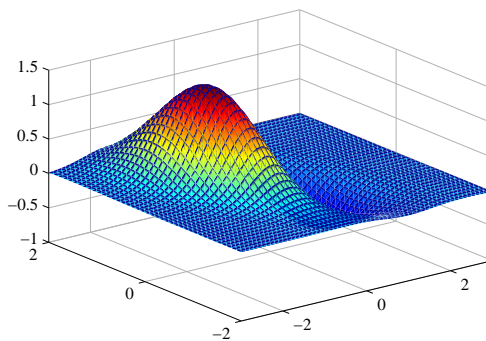
若用 `surf()` 函数取代 `mesh()` 函数,则可以得出如图 2-14(b) 所示的表面图

```
>> surf(x,y,z) % 绘制三维表面图
```

三维表面图可以用 `shading` 命令修饰其显式形式,该命令可以带三种不同的选项, `flat` (每个网格块用同样颜色着色的没有网格线的表面图,效果如图 2-15 (a) 所示)、`interp` (插值的光滑表面图,效果如图 2-15 (b) 所示) 和 `faceted` (不同于 `flat`,有网格线的,本选项是默认的,效果如图 2-14 (b) 所示)。



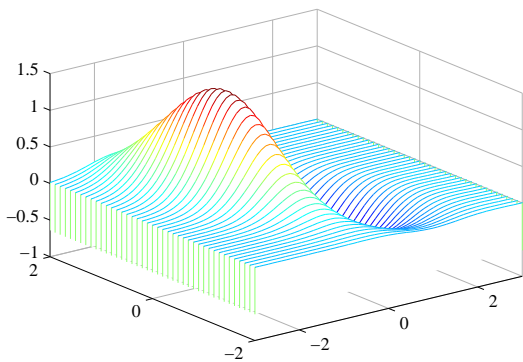
(a) shading flat



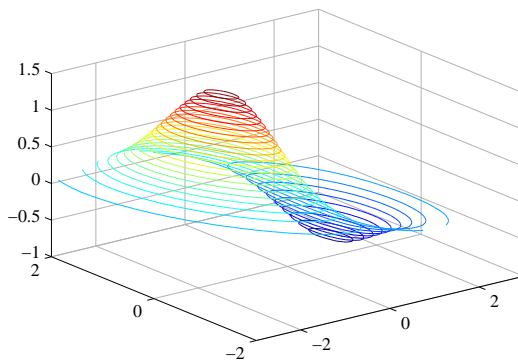
(b) shading interp

图 2-15 shading 命令修饰的三维图

MATLAB 还提供了其他的三维图形绘制函数。如 `waterfall(x,y,z)` 命令可以绘制出瀑布形图形,如图 2-16 (a) 所示,而 `contour3(x,y,z,30)` 命令可以绘制出如图 2-16 (b) 所示的三维等高线图形,其中,30 为用户选定的等高线条数,当然可以不给出该参数,那样将默认地设置等高线条数,对这个例子来说显得过于稀疏。



(a) waterfall() 函数的绘制曲面



(b) contour3() 绘制的三维等高线图

图 2-16 其他三维图形表示

```
>> ezsurf('x^2-2*x)*exp(-x^2-y^2-x*y)', [-3 2 -2 2])
figure; ezcontour('x^2-2*x)*exp(-x^2-y^2-x*y)', [-3 2 -2 2])
```

例 2-32 试绘制出二元函数 $z = f(x, y) = \frac{1}{\sqrt{(1-x)^2 + y^2}} + \frac{1}{\sqrt{(1+x)^2 + y^2}}$ 。

解 可以用下面的语句绘制出三维图,如图 2-17 (a) 所示

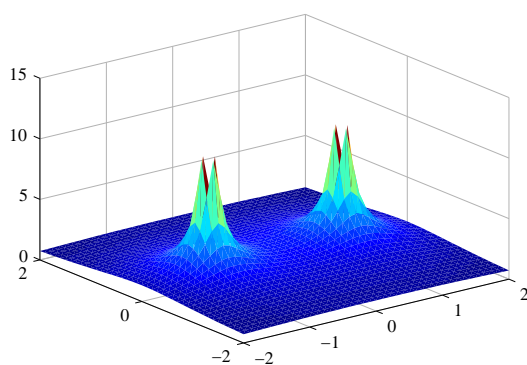
```
>> [x,y]=meshgrid(-2:.1:2);
z=1./(sqrt((1-x).^2+y.^2))+1./(sqrt((1+x).^2+y.^2));
```



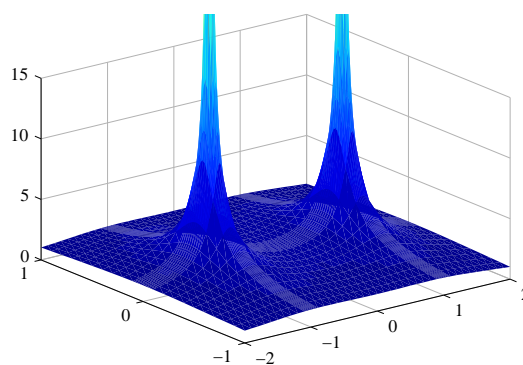
```
surf(x,y,z), shading flat
```

事实上,这样得出的图形有点问题,在 $(\pm 1, 0)$ 点处出现 ∞ 值,所以应该在该区域减小步距,采用变步距的方式,最终读出如图 2-17(b)所示的图形,为了便于比较,这里仍选择 z -轴的范围和图 2-17(a)的一致。注意在 $(\pm 1, 0)$ 处的值趋于无穷大。

```
>> xx=[-2:.1:-1.2, -1.1:0.02:-0.9, -0.8:0.1:0.8, 0.9:0.02:1.1, 1.2:0.1:2];
yy=[-1:0.1:-0.2, -0.1:0.02:0.1, 0.2:.1:1]; [x,y]=meshgrid(xx,yy);
z=1./(sqrt((1-x).^2+y.^2))+1./(sqrt((1+x).^2+y.^2));
surf(x,y,z), shading flat; zlim([0,15])
```



(a) 等步距



(b) 变步距

图 2-17 不同网格选择下的三维图

例 2-33 假设某概率密度函数由下面分段函数表示^[5]

$$p(x_1, x_2) = \begin{cases} 0.5457 \exp(-0.75x_2^2 - 3.75x_1^2 - 1.5x_1), & x_1 + x_2 > 1 \\ 0.7575 \exp(-x_2^2 - 6x_1^2), & -1 < x_1 + x_2 \leq 1 \\ 0.5457 \exp(-0.75x_2^2 - 3.75x_1^2 + 1.5x_1), & x_1 + x_2 \leq -1 \end{cases}$$

试以三维曲面的形式来表示这一函数。

解 选择 $x = x_1, y = x_2$, 用循环结构和条件转移结构可以求取该函数的函数值, 但结构将很烦琐, 所以类似于前面介绍的分段函数求取方法, 可以利用比较表达式来求此二维函数的值

```
>> [x,y]=meshgrid(-1:.04:1,-2:.04:2);
z= 0.5457*exp(-0.75*y.^2-3.75*x.^2-1.5*x).*(x+y>1)+...
    0.7575*exp(-y.^2-6*x.^2).*((x+y>-1) & (x+y<=1))+...
    0.5457*exp(-0.75*y.^2-3.75*x.^2+1.5*x).*(x+y<=-1);
h=surf(x,y,z), shading flat
```

这样将得出如图 2-18 所示的三维表面图。此外, 由于这里 surf() 函数返回了句柄 h, 可以给出命令 delete(h) 删除得出的三维曲面。后面还将演示对曲面的旋转处理等进一步操作。

2.6.3 等高线绘制

如果已知三维网格数据 x, y, z , 则可以通过 contour() 函数绘制三维数据的等高线, 该函数的调用格式为 `contour(x,y,z,n)`, 其中 n 为等高线的条数, 该函数的一种调用格

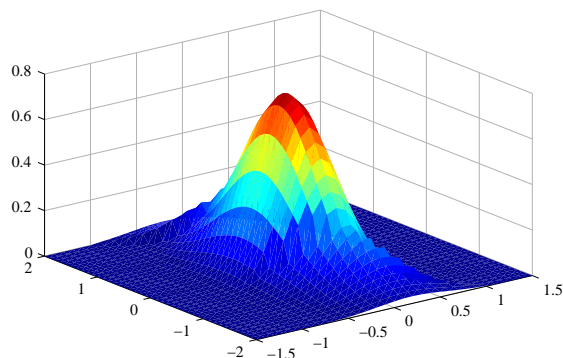


图 2-18 分段二维函数曲面绘制

式为 $[C,h] = \text{contour}(x,y,z,n)$ ，该函数返回的变量 h 为等高线的句柄， C 为等高线高度信息。若有了这些信息，则 $\text{clabel}(C,h)$ 函数可以在等高线上叠印出等高线信息。

函数 $\text{contourf}()$ 可以绘制出填充的等高线图，而 $\text{contour3}()$ 函数可以绘制出三维等高线图，它们的调用格式分别为 $\text{contourf}(x,y,z,n)$ 或 $\text{contour3}(x,y,z,n)$ 。

例 2-34 考虑例 2-33 中给出的分段函数，试绘制其等高线图。

解 可以仿照前面语句得出绘图的数据，等高线图可以由 $\text{contour}()$ 函数直接绘制，得出的结果如图 2-19(a) 所示。该函数除了绘图之外还将返回等高线的句柄 h 和数据 C ，依赖这两个变量即可以在原来的等高线图上叠印等高线的数值，如图 2-19(b) 所示。

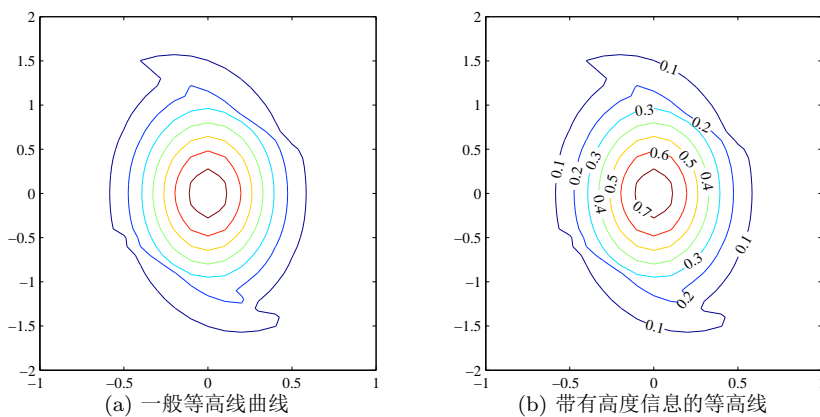


图 2-19 分段函数的等高线

```
>> [x,y]=meshgrid(-1:1:1,-2:1:2);
z= 0.5457*exp(-0.75*y.^2-3.75*x.^2-1.5*x).*(x+y>1)+...
    0.7575*exp(-y.^2-6*x.^2).*((x+y>-1) & (x+y<=1))+...
    0.5457*exp(-0.75*y.^2-3.75*x.^2+1.5*x).*(x+y<=-1);
[C,h]=contour(x,y,z); clabel(C,h)
```

用下面的语句可以直接绘制出填充的等高线图和三维等高线图，如图 2-20(a)、(b) 所示，其中，后一个语句中的 30 是指定等高线的条数，如果不给出此参数，对本例来说等高线将过于稀疏。


```
>> contourf(x,y,z); figure; contour3(x,y,z,30)
```

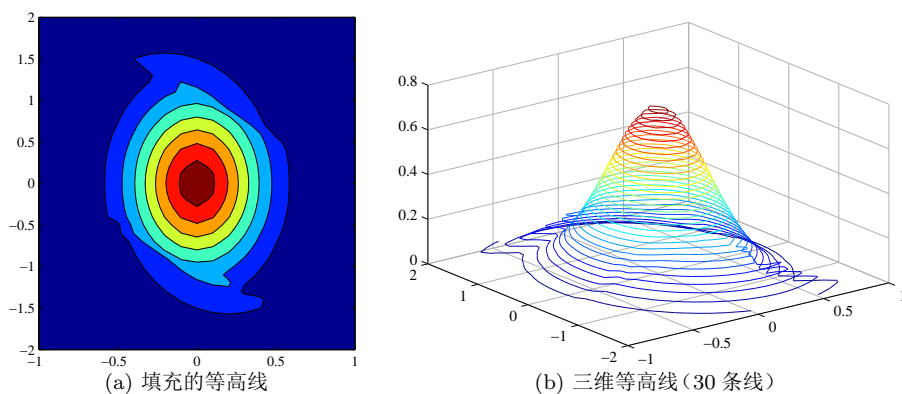


图 2-20 填充等高线和三维等高线

2.6.4 三维隐函数图绘制

前面介绍的 `ezplot3()` 等函数只能绘制三维显函数曲线。如果某三维曲面由隐函数 $g(x, y, z) = 0$ 表示,则可以调用 `ezimplot3()` 函数绘制其曲面图形^[6],该函数的调用格式为 `ezimplot3(fun, [xm, xM, ym, yM, zm, zM])`,其中, `fun` 可以为匿名函数、字符串、M-函数,也可以是符号表达式,其中字符串既可以表示 M-函数的文件名,也可以直接描述隐函数。坐标轴范围向量 $x_m, x_M, y_m, y_M, z_m, z_M$ 的默认值为 $\pm 2\pi$ 。如果只给出一对上下限 x_m, x_M ,则表示三个坐标轴均同样设置。该函数的核心部分是等高面绘制函数。

例 2-35 假设某三维曲线由隐函数 $x(x, y, z) = x \sin(y+z^2) + y^2 \cos(x+z) + zx \cos(z+y^2) = 0$ 表示,且感兴趣的区域为 $x, y, z \in (-1, 1)$,试绘制其三维曲面。

解 用字符串或匿名函数的方式都可以描述原始的隐函数,二者作用相同

```
>> f='x*sin(y+z^2)+y^2*cos(x+z)+z*x*cos(z+y^2)';  
f=@(x,y,z)x*sin(y+z^2)+y^2*cos(x+z)+z*x*cos(z+y^2);
```

这样,用下面语句就可以直接绘制出该隐函数的三维曲面图,如图 2-21(a) 所示。

```
>> ezimplot3(f, [-1 1])
```

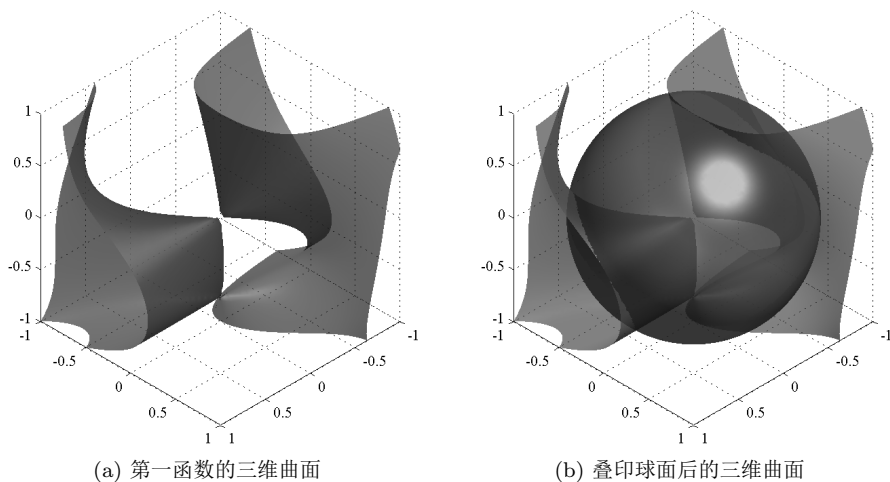
如果使用下面的语句还可以在原曲面上叠印单位球面 $x^2 + y^2 + z^2 = 1$,如图 2-21(b) 所示。

```
>> f1='x^2+y^2+z^2'; ezimplot3(f1, [-1 1]);
```

2.6.5 三维图形视角设置

MATLAB 三维图形显示中提供了修改视角的功能,允许用户从任意的角度观察三维图形,实现视角转换有两种方法。其一是使用图形窗口工具栏中提供的三维图形转换按钮来可视地对图形进行旋转,其二是用 `view()` 函数有目的地进行旋转。

MATLAB 三维图形视角的定义如图 2-22 (a) 所示。其中有两个角度就可以唯一地确定视角,方位角 α 定义为视点在 x - y 平面投影点与 y 轴负方向之间的夹角,默认值为

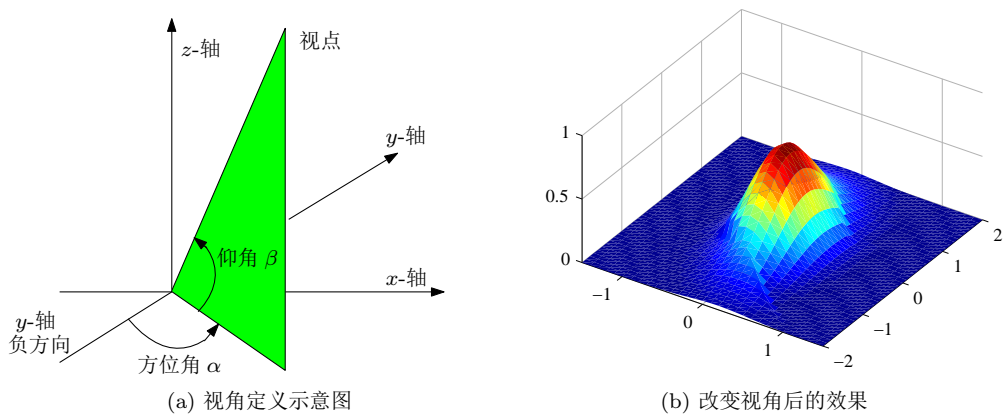


(a) 第一函数的三维曲面

(b) 叠印球面后的三维曲面

图 2-21 隐函数三维曲面绘制

$\alpha = -37.5^\circ$, 仰角 β 定义为视点和 x - y 平面的夹角, 默认值为 $\beta = 30^\circ$ 。



(a) 视角定义示意图

(b) 改变视角后的效果

图 2-22 三维图形的视角及设置

如果想改变视角来观察曲面, 则可以给出 `view(α , β)` 命令。例如, 俯视图可以由函数 `view(0,90)` 设置, 正视图由 `view(0,0)` 设置, 侧视图可以由 `view(90,0)` 来设定。

例如, 对图 2-18 中给出的三维网格图进行处理, 设方位角为 $\alpha = 80^\circ$, 仰角为 $\beta = 10^\circ$, 则下面的 MATLAB 语句将得出如图 2-22 (b) 所示的三维曲面。

```
>> view(80,10), xlim([-1.5 1.5])
```

例 2-36 试在同一图形窗口上绘制例 2-31 中函数曲面的三视图。

解 用下面的语句可以容易地绘制出三维图, 并用相应的语句设置不同的视角, 则可以最终得出如图 2-23 所示的各个视图。

```
>> [x,y] = meshgrid(-3:0.1:3,-2:0.1:2);
    z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); subplot(224), surf(x,y,z)
    subplot(221), surf(x,y,z), view(0,90); % 俯视图
```

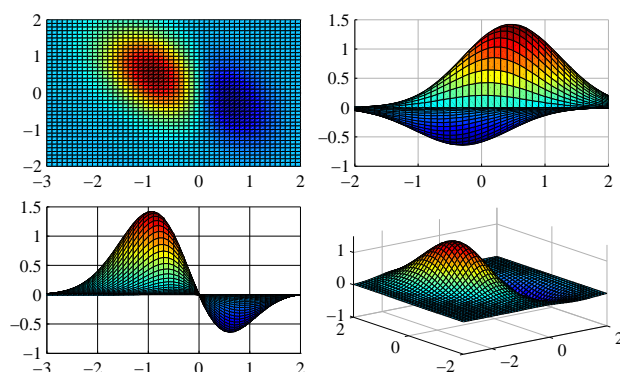


图 2-23 二元函数的三视图

```
subplot(222), surf(x,y,z), view(90,0); % 侧视图
subplot(223), surf(x,y,z), view(0,0); % 正视图
```

2.6.6 三维曲面的旋转

前面介绍的视角变换并未改变曲面的本身,只是通过重新设置视角来调整观察角度。MATLAB 还提供了曲面本身的旋转变换方法,旋转变换可以采用 `rotate()` 函数实现,该函数的调用格式为 `rotate(h,v,α)`,其中, `h` 为曲面的句柄,该句柄可以由 `surf()` 函数直接返回,也可以在图形编辑状态下单击选中曲面,然后由 `h = gco` 命令提取。`v` 为旋转的基线,它是 1×3 的向量,存储一个三维空间点,旋转基线是坐标轴原点与该空间点之间的连线。 α 是旋转的角度(单位为“度”)。如果想绕 x 轴正方向旋转,则 $v = [1,0,0]$,如果想让其绕 x 轴负方向旋转,则 $v = [-1,0,0]$ 。

例 2-37 重新考虑例 2-33 中给出的分段函数曲面,试旋转得出的曲面。

解 用下面的语句可以重新绘制原分段函数的三维曲面,如图 2-18 所示。

```
>> [x,y]=meshgrid(-1:.04:1,-2:.04:2);
z= 0.5457*exp(-0.75*y.^2-3.75*x.^2-1.5*x).*(x+y>1)+...
    0.7575*exp(-y.^2-6*x.^2).*((x+y>-1) & (x+y<=1))+...
    0.5457*exp(-0.75*y.^2-3.75*x.^2+1.5*x).*(x+y<=-1); h=surf(x,y,z);
```

该函数返回了曲面句柄 `h`,如果想将原曲面沿 x 轴逆时针旋转 15° ,则可以给出下面的语句,旋转后的曲面如图 2-24(a) 所示。

```
>> rot_ax=[1,0,0]; rotate(h,rot_ax,15)
```

如果想让原曲面绕原点与空间点 $(1,1,1)$ 之间的连线旋转 15° ,则可以得出下面的语句,这样可以得出如图 2-24(b) 所示的旋转效果。

```
>> h=surf(x,y,z); rot_ax=[1,1,1]; rotate(h,rot_ax,15)
```

下面可以用循环结构给出原曲面沿 x 轴旋转一周的动画演示(每 0.02s 旋转 1°)。这里使用了 `axis tight` 保证旋转过程中坐标轴的尺度固定不变,另外值得注意的是,旋转角度应该填写 1,而不

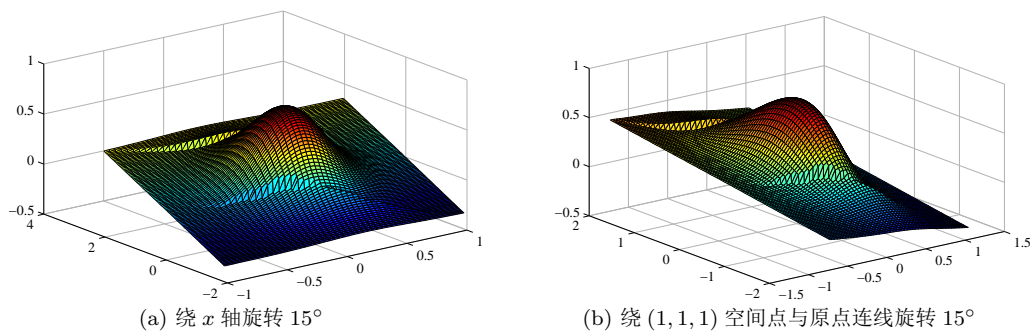


图 2-24 曲面旋转的效果

能写成 i , 因为每循环一步都在原来的基础上旋转 1° 。

```
>> figure; h=surf(x,y,z); r_ax=[1 0 0]; axis tight % 保证坐标轴尺度不变
    for i=0:360, rotate(h,r_ax,1); pause(0.02), end
```

2.7 四维图形绘制

前面介绍的三维图形绘制主要描述的是二元函数 $z = S(x, y)$ 在三维空间内的图形, 如果某三元函数的数学表达式为 $v = V(x, y, z)$, 则需要绘制该三元函数的体视化 (volume visualization) 图形。三元函数在实际应用中有很多例子, 例如固体内部的温度、流体的流速、液体的浓度分布等, 这用普通的三维图是表现不出来的, 而直接绘制四维图是不可能的, 所以只能用特殊三维空间图形来表示, 再辅以任何角度的切面观察三维问题内部函数的值。这里的方法又称为体视化方法。CT 成像是用切面观察三维问题内部结构的很好的例子。当然, 前面给出的三维动画演示也可以理解成一种四维的图形, 即三维曲面随第 4 维——时间的变化动画。

可以用 `meshgrid()` 函数生成三维网格数据 x, y, z , 再将三元函数的体数据 V 求出来 (注意应该采用点运算计算体数据), 然后再调用 `slice()` 函数绘制出感兴趣的切面。该函数的调用格式为 `slice(x, y, z, V, x1, y1, z1)`, 其中, x, y, z, V 为体视化数据, x_1, y_1, z_1 为描述切面的数据, 如果为常数向量则表示垂直于该坐标轴的切面, 当然这些切面也可以设置为旋转得出的平面甚至曲面, 具体使用方法将通过例子演示。

例 2-38 已知某三元函数 $V(x, y, z) = \sqrt{x^x + y^{(x+y)/2} + z^{(x+y+z)/3}}$, 试用体视化的方法观察该三元函数, 并给出切面观察该函数的性质。

解 由于涉及求平方根, 所以 x, y, z 应该取非负值, 可以通过如下命令构造网格数据, 然后计算出体视化数据 V 。分别选择三组平行于坐标轴平面的切面, 例如, 第一组切面定位于 $x = 1, x = 2$, 第二组定位于 $y = 1, y = 2$, 第三组设置于 $z = 0, z = 1$, 这样可以得出如图 2-25 (a) 所示的切面图。

```
>> [x,y,z]=meshgrid(0:0.1:2); V=sqrt(x.^x+y.^((x+y)/2)+z.^((x+y+z)/3));
    slice(x,y,z,V,[1 2],[1 2],[0 1]);
```

利用前面介绍的方法, 先构造一个普通平面 $z = 1$, 再沿 x 轴旋转 45° 构造切面, 这样即可由该

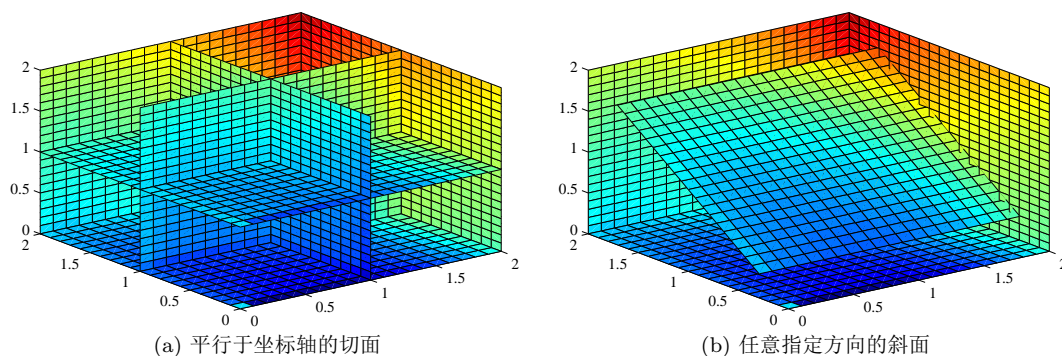


图 2-25 切面图的效果

切面提取 x_1, y_1, z_1 数据, 则可以由 `slice()` 函数得出所需的切面图, 如图 2-25(b) 所示。

```
>> [x0,y0]=meshgrid(0:0.1:2); z0=ones(size(x0));
    h=surf(x0,y0,z0); rotate(h,[1,0,0],45);
    x1=get(h,'XData'); y1=get(h,'YData'); z1=get(h,'ZData');
    slice(x,y,z,V,x1,y1,z1), hold on, slice(x,y,z,V,2,2,0)
```

为更方便地观察切面图, 我们编写了一个简易的图形用户界面 `vol_visual4d()`, 使用此界面之前应该在 MATLAB 工作空间中建立体视化数据 x, y, z, V , 这样就可以用下面格式调用此函数 `vol_visual4d(x,y,z,V)`, 然后利用界面上的控件直接处理各个切面。

例 2-39 可以用下面的语句直接生成例 2-38 中的数据, 然后调用 `vol_visual4d()` 函数, 则可以直接起动此界面。对图形的属性稍加处理即可以得出如图 2-26 所示的切面显示。

```
>> [x,y,z]=meshgrid(0:0.1:2); V=sqrt(x.^x+y.^((x+y)/2)+z.^((x+y+z)/3));
    vol_visual4d(x,y,z,V);
```

2.8 习 题

1. 启动 MATLAB 环境, 并给出语句

```
tic, A=rand(500); B=inv(A); norm(A*B-eye(500)), toc
```

试运行该语句, 观察得出的结果, 并利用 `help` 或 `doc` 命令对你不熟悉的语句进行帮助信息查询, 逐条给出上述程序段与结果的解释。

2. 试用符号元素工具箱支持的方式表达多项式 $f(x) = x^5 + 3x^4 + 4x^3 + 2x^2 + 3x + 6$, 并令 $x = (s-1)/(s+1)$, 将 $f(x)$ 替换成 s 的函数。
3. 试求出无理数 $\sqrt{2}$ 、 $\sqrt[6]{11}$ 、 $\sin 1^\circ$ 、 e^2 、 $\ln(21)$ 的前 200 位有效数字。
4. 已知数学函数 $f(x) = \frac{x \sin x}{\sqrt{x^2 + 2}(x+5)}$, $g(x) = \tan x$, 试求出复合函数 $f(g(x))$ 和 $g(f(x))$ 。
5. 由于双精度数据结构有一定的位数限制, 大数的阶乘很难保留足够的精度。试用数值方法和符号运算的方法计算并比较 C_{50}^{10} , 其中 $C_m^n = \frac{m!}{n!(m-n)!}$ 。

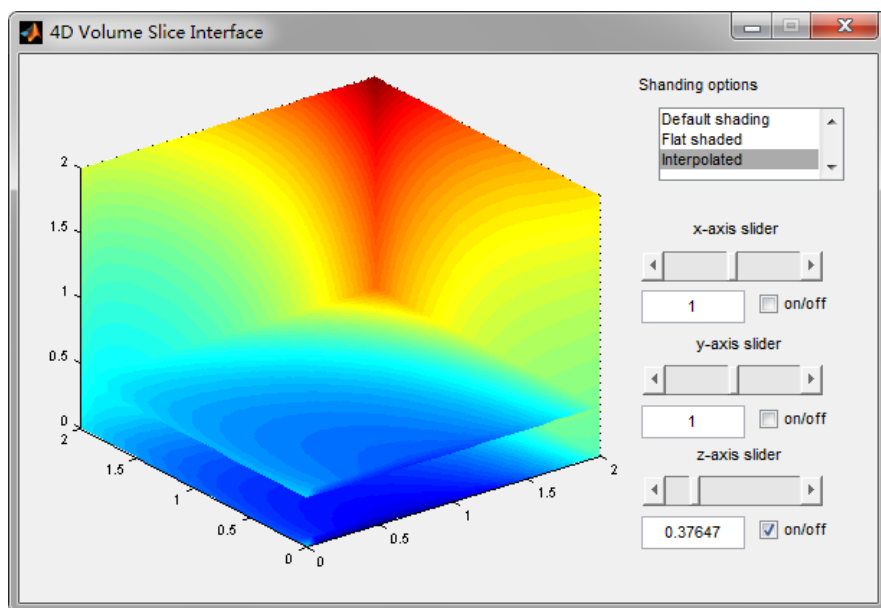


图 2-26 切面界面的效果显示

6. 用 MATLAB 语句输入矩阵 A 和 B

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 2 & 3 & 4 & 1 \\ 3 & 2 & 4 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1+4j & 2+3j & 3+2j & 4+1j \\ 4+1j & 3+2j & 2+3j & 1+4j \\ 2+3j & 3+2j & 4+1j & 1+4j \\ 3+2j & 2+3j & 4+1j & 1+4j \end{bmatrix}$$

前面给出的是 4×4 矩阵, 如果给出 $A(5,6) = 5$ 命令将得出什么结果?

7. 假设已知矩阵 A , 试给出相应的 MATLAB 命令, 将其全部偶数行提取出来, 赋给 B 矩阵, 用 $A = \text{magic}(8)$ 命令生成 A 矩阵, 用上述的命令检验一下结果是不是正确。

8. 用 MATLAB 语言实现下面的分段函数 $y = f(x) = \begin{cases} h, & x > D \\ h/Dx, & |x| \leq D \\ -h, & x < -D \end{cases}$ 。

9. 用数值方法可以求出 $S = \sum_{i=0}^{63} 2^i = 1 + 2 + 4 + 8 + \cdots + 2^{62} + 2^{63}$, 试不采用循环的形式求出和式的数值解。由于数值方法采用 `double` 形式进行计算, 难以保证有效位数字, 所以结果不一定精确。试采用符号运算的方法求该和式的精确值。

10. 编写一个矩阵相加函数 `mat.add()`, 使其具体的调用格式为 $A = \text{mat.add}(A_1, A_2, A_3, \cdots)$, 要求该函数能接受任意多个矩阵进行加法运算。

11. 自己编写一个 MATLAB 函数, 使它能自动生成一个 $m \times m$ 的 Hankel 矩阵, 并使其调用格式为 $v = [h_1, h_2, h_m, h_{m+1}, \cdots, h_{2m-1}]$; $H = \text{myhankel}(v)$ 。

12. 已知 Fibonacci 数列可以由式 $a_k = a_{k-1} + a_{k-2}$, $k = 3, 4, \cdots$ 生成, 其中初值为 $a_1 = a_2 = 1$, 试编写出生成某项 Fibonacci 数值的 MATLAB 函数, 要求:

(1) 函数格式为 $y = \text{fib}(k)$, 给出 k 即能求出第 k 项 a_k 并赋给 y 向量;

(2) 编写适当语句, 对输入输出变量进行检验, 确保函数能正确调用;

(3) 利用递归调用的方式编写此函数。

13. 已知某迭代序列 $x_{n+1} = \frac{x_n}{2} + \frac{3}{2x_n}$, $x_1 = 1$, 并已知该序列当 n 足够大时将趋于某个固定的常数, 试选择合适的 n , 找出该序列的稳态值 (达到精度要求 10^{-14}), 并找出其精确的数学表示。

14. 试用循环结构找出 1000 以下所有的质数。

15. 若某个三位数, 每位数字的三次方的和等于其本身, 则称其为水仙花数, 试找出所有水仙花数。

16. 试求 $S = \prod_{n=1}^{\infty} \left(1 + \frac{2}{n^2}\right)$, 使计算精度达到 $\epsilon = 10^{-12}$ 级。

17. 已知 $\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$ 。取 $x = 1$, 则立即得出下面的计算式

$$\pi \approx 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right)$$

试利用循环累加方法计算出 π 的近似值, 要求精度达到 10^{-6} 。

18. 试用下面两种方法求解代数方程 $f(x) = x^2 \sin(0.1x + 2) - 3 = 0$ 。

(1) **二分法**。若在某个区间 (a, b) 内, $f(a)f(b) < 0$, 则该区间内存在方程的根。取中点 $x_1 = (a + b)/2$, 则可以根据 $f(x_1)$ 和 $f(a)$ 、 $f(b)$ 的关系确定根的范围, 用这样的方法可以将区间的长度减半。重复这样的过程, 直至区间长度小于预先指定的 ϵ , 则可以认为得出的区间端点是方程的解。令 $\epsilon = 10^{-10}$, 试用二分法求区间 $(-4, 0)$ 内方程的解。

(2) **Newton-Raphson 迭代法**。假设该方程解的某个初始猜测点为 x_n , 则由梯度法可以得出下一个近似点 $x_{n+1} = x_n - f(x_n)/f'(x_n)$ 。若两个点足够近, 即 $|x_{n+1} - x_n| < \epsilon$, 其中 ϵ 为预先指定的误差限, 则认为 x_{n+1} 是方程的解, 否则将 x_{n+1} 设置为初值继续搜索, 直至得出方程的解。令 $x_0 = -4$, $\epsilon = 10^{-12}$, 试用 Newton-Raphson 迭代法求解上面的方程。

19. 由矩阵理论可知, 如果一个矩阵 M 可以写成 $M = A + BCB^T$, 并且其中 A , B , C 为相应阶数的矩阵, 则 M 矩阵的逆矩阵可以由下面的算法求出

$$M^{-1} = (A + BCB^T)^{-1} = A^{-1} - A^{-1}B(C^{-1} + B^T A^{-1}B)^{-1} B^T A^{-1}$$

试根据上面的算法用 MATLAB 语句编写一个函数对矩阵 M 进行求逆, 并通过一个小例子来检验该程序, 并和直接求逆方法进行精度上的比较。

20. 已知迭代模型 $\begin{cases} x_{k+1} = 1 + y_k - 1.4x_k^2, \\ y_{k+1} = 0.3x_k \end{cases}$, 试写出求解该模型的 M-函数。如果取迭代初值为

$x_0 = 0$, $y_0 = 0$, 那么请进行 30000 次迭代求出一组 x 和 y 向量, 然后在所有的 x_k 和 y_k 坐标处点亮一个点 (注意不要连线), 最后绘制出所需的图形。**提示:** 这样绘制出的图形又称为 Henon 引力线图, 它将迭代出来的随机点吸引到一起, 最后得出貌似连贯的引力线图。

21. 著名的 Mittag-Leffler 函数的基本定义为 $f_{\alpha}(x) = \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(\alpha k + 1)}$, 其中 $\Gamma(x)$ 为 Γ 函数, 可以由 `gamma(x)` 函数直接计算。试编写出 MATLAB 函数, 使得其调用格式为 `f = mymittag(alpha, z, epsilon)`, ϵ 为用户允许的误差限, 其默认值为 $\epsilon = 10^{-6}$, z 为已知数值向量。利用该函数分别绘制出 $\alpha = 1$ 和 $\alpha = 0.5$ 的曲线。
22. 用 MATLAB 语言的基本语句显然可以立即绘制一个正三角形。试结合循环结构, 编写一个小程序, 在同一个坐标系下绘制出该正三角形绕其中心旋转后得出的一系列三角形, 还可以调整旋转步距观察效果。
23. 选择合适的步距绘制出图形 $\sin 1/t$, 其中 $t \in (-1, 1)$ 。
24. 分别选取合适的 θ 范围, 绘制出下列极坐标图形:
- (1) $\rho = 1.0013\theta^2$, (2) $\rho = \cos 7\theta/2$, (3) $\rho = \sin \theta/\theta$, (4) $\rho = 1 - \cos^3 7\theta$
25. 用图解的方式求解下面联立方程的近似解:
- (1) $\begin{cases} x^2 + y^2 = 3xy^2 \\ x^3 - x^2 = y^2 - y \end{cases}$ (2) $\begin{cases} e^{-(x+y)^2 + \pi/2} \sin(5x + 2y) = 0 \\ (x^2 - y^2 + xy)e^{-x^2 - y^2 - xy} = 0 \end{cases}$
26. Lambert W 函数是一个常用的函数, 其数学形式为 $W(z)e^{W(z)} = z$, 试绘制其函数曲线。
27. 请分别绘制出 xy 和 $\sin xy$ 的三维图和等高线。
28. 在图形绘制语句中, 若函数值为不定式 `NaN`, 则相应的部分不绘制出来。试利用该规律绘制 $z = \sin xy$ 的表面图, 并剪切下 $x^2 + y^2 \leq 0.5^2$ 的部分。
29. 试绘制出 $(x^2 + xy + xz)e^{-z} + z^2yx + \sin(x + y + z^2) = 0$ 函数的曲面。
30. 试绘制下面三元函数的体视化切面图。
- (1) $V(x, y, z) = \sqrt{e^x + e^{(x+y)-xy} + e^{(x+y+z)/3-xyz}}$, (2) $V(x, y, z) = e^{-x^2 - y^2 - z^2}$

参考文献

- [1] Lamport L. \LaTeX : a document preparation system — user's guide and reference manual. Reading MA: Addison-Wesley Publishing Company, second edition, 1994
- [2] Gilbert D. Extended plotyy to three y-axes. MATLAB Central File ID: # 1017, 2001
- [3] Bodin P. PLOTYY4 support for four y axes. MATLAB Central File ID: # 4425, 2004
- [4] Gilbert D. PLOTXX create graphs with two x axes. MATLAB Central File ID: # 317, 1999
- [5] Atherton D P, Xue D. The analysis of feedback systems with piecewise linear nonlinearities when subjected to Gaussian inputs. Kozin F, Ono T, eds., Control systems, topics on theory and application, 23–38. Tokyo: Mita Press, 1991
- [6] Morales G. Ezimplot3: implicit 3D functions plotter. MATLAB Central File ID #23623

第3章 微积分问题的计算机求解

Issac Newton (1643 – 1727) 和 Gottfried Wilhelm Leibniz (1646 – 1716) 创立的微积分学是很多科学分支的基础。单变量与多变量函数微积分、函数极限、级数求和、Taylor 级数展开、Fourier 级数展开、常微分方程等问题直接求解是微积分学的重要内容。MATLAB 的符号运算工具箱可以直接求解这样问题的解析解。本章 3.1 节中给出基于 MATLAB 符号运算工具箱中函数的单边、多边极限问题及多变量函数极限问题的求解方法, 3.2 节介绍各种微分问题的计算机求解方法, 3.3 节介绍各种积分问题的解析求解方法。3.4 节将介绍给定单变量函数与多变量函数的 Taylor 幂级数展开、给定函数的 Fourier 级数逼近方法, 并利用 MATLAB 的绘图功能研究有限项拟合的拟合效果和适用范围; 还介绍一般级数的求和与求积方法等。3.5 节中将介绍的两类曲线积分和两类曲面积分及其 MATLAB 求解方法补充了微积分学的计算机求解方式, 这部分内容大部分均应该是解析求解和解析推导, 属于计算机代数研究的领域, 用传统的数值分析方法是不能求解的。对不熟悉计算机代数系统开发的读者来说, 用 C 这样的底层语言直接进行解析解推导有极大难度, 而必须使用计算机数学语言 (如 MATLAB 语言) 完成这类问题的分析与求解。

在实际科学与工程研究中, 微积分问题解析求解有时也面临困难。例如, 若函数本身未知, 只有由科学实验测出的一些实验数据, 则无法用推导的方式通过数据对其代表的函数求导或求积分, 而需要通过数值的方式进行数值微分与数值积分的运算等。3.6 节中将介绍实用的中心差分数值微分算法及其 MATLAB 语言实现, 还将介绍多变量函数的偏微分求解方法与实例。在实际应用中还有很多函数积分的解析解不存在, 所以需要通过数值积分的算法进行近似。3.7 节中将介绍用数值算法求取函数积分及重积分问题的求解方法。若给出的数据点较稀疏, 则基于数据直接求取数值微积分会有很大的误差, 但可以结合第 8 章中介绍的样条插值方法对其求解。具体内容请参见第 8 章 8.2 节。

对本章内容进一步拓展, 如果微积分的阶次可以选择为非整数, 则可以引入一个新的学科——非整数阶微积分学, 或更常用的, 分数阶微积分学。本书第 10 章 10.6 节将系统介绍分数阶微积分学问题的 MATLAB 求解方法。

3.1 极限问题的解析解

应用 MATLAB 语言的符号运算工具箱, 可以很容易地求解极限问题、微分问题、积分问题等微积分基本问题。利用本节和后面两节介绍的方法, 读者应该能立即具备依赖 MATLAB 语言及其符号运算工具箱中提供的强大函数直接求解一般微积分运算问题的能力。本节主要侧重各种极限问题的求解方法, 包括单变量极限、单边极限和多重极限问题。

3.1.1 单变量函数的极限

假设已知函数 $f(x)$, 则极限问题的一般描述为

$$L = \lim_{x \rightarrow x_0} f(x) \quad (3-1-1)$$

其中, x_0 可以是一个确定的值, 也可以是无穷大, 例如 $x \rightarrow \infty$ 。对某些函数来说, 还可以如下定义单边极限 (或称左右极限) 问题。

$$L_1 = \lim_{x \rightarrow x_0^-} f(x), \text{ 或 } L_2 = \lim_{x \rightarrow x_0^+} f(x) \quad (3-1-2)$$

前者表示 x 从左侧趋近于 x_0 点, 所以又称为左极限, 后者相应地称为右极限。极限问题在 MATLAB 符号运算工具箱中可以使用 `limit()` 函数直接求出, 该函数的调用格式为

```
L = limit(f, x, x0) % 求极限
L = limit(f, x, x0, 'left' 或 'right') % 求单边极限
```

在求解之前应该先申明自变量 x , 再用符号表达式的形式定义原函数 f , 若 x_0 为 ∞ , 则可以用 `inf` 直接表示。如果要求解左右极限问题, 还需要给出 'left' 或 'right' 选项。下面将通过例子演示 MATLAB 求解极限的方法。

例 3-1 试求解极限问题 $\lim_{x \rightarrow \infty} x \left(1 + \frac{a}{x}\right)^x \sin \frac{b}{x}$ 。

解 利用 MATLAB 语言, 应该首先申明 a, b 和 x 为符号变量, 然后定义函数或序列表达式, 最后调用 `limit()` 函数求出给定函数的极限, 得出的极限为 $e^a b$ 。

```
>> syms x a b; f=x*(1+a/x)^x*sin(b/x); L=limit(f,x,inf)
```

例 3-2 试求解单边极限问题 $\lim_{x \rightarrow 0^+} \frac{e^{x^3} - 1}{1 - \cos \sqrt{x} - \sin x}$ 。

解 利用 MATLAB 语言的 `limit()` 函数, 可以容易地求出单边极限为 12。

```
>> syms x; f=(exp(x^3)-1)/(1-cos(sqrt(x)-sin(x))); limit(f,x,0,'right')
```

用下面的语句还可以绘制出 $(-0.1, 0.1)$ 区间的函数曲线, 如图 3-1 所示。

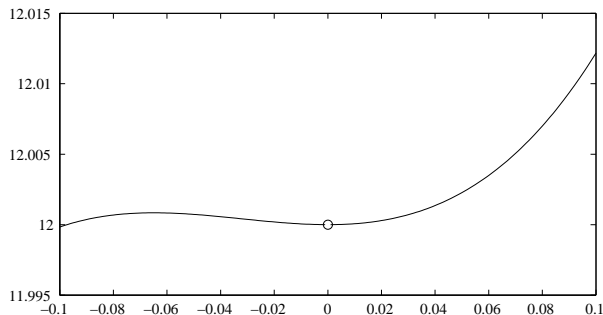


图 3-1 $x = 0$ 附近的曲线

```
>> ezplot(f, [-0.01, 0.01]); hold on; plot([0], [12], 'o')
```

可见,对这个例子来说,即使使用 `limit(f,x,0)` 命令也能求出函数极限值是 12。

回顾原始问题,其中采用 $x \rightarrow 0^+$ 是因为它可以保证根号内的值为非负数。事实上,即使是负数, $\cos j\alpha = (e^{\alpha} + e^{-\alpha})/2$ 也是有定义的,且其结果为实数。这对本问题没有影响。但对某些分段函数来说,单边极限是不同的。

若关于某点 a , 函数 $f(t)$ 的左右极限相同,则该点称为第一类间断点,否则称为第二类间断点。下面的例子演示一个简单的第二类间断点问题。

例 3-3 试分别求出 $\tan t$ 函数关于 $\pi/2$ 点处的左右极限。

解 由下面命令可以分别求出函数的左右极限,分别为 $L_1 = \infty$ 和 $L_2 = -\infty$ 。

```
>> syms t; f=tan(t); L1=limit(f,t,pi/2,'left'), L2=limit(f,t,pi/2,'right')
```

例 3-4 试求出序列的极限 $\lim_{n \rightarrow \infty} \frac{\sqrt[3]{n^2} \sin n!}{n+1}$ 。

解 序列极限的求解方法与函数极限完全一致:先声明符号变量,然后用符号表达式描述序列,最后调用 `limit()` 函数直接求解。由下面的语句可以得出此序列的极限等于 0。

```
>> syms n; f=n^(2/3)*sin(factorial(n))/(n+1); F=limit(f,n,inf)
```

例 3-5 试求出极限 $\lim_{n \rightarrow \infty} n \arctan \left(\frac{1}{n(x^2+1)+x} \right) \tan^n \left(\frac{\pi}{4} + \frac{x}{2n} \right)$ 。

解 该极限表达式既包括序列又包括函数,但这丝毫未给求解带来任何困难,可以申明两个符号变量—— n 和 x ,这样用下面语句可以直接得出问题的极限为 $e^x/(x^2+1)$ 。

```
>> syms x n; f=n*atan(1/(n*(x^2+1)+x))*tan(pi/4+x/2/n)^n; limit(f,n,inf)
```

3.1.2 区间函数的极限运算

在引入区间函数概念之前先看一下下面的例子。

例 3-6 试求出 $\lim_{x \rightarrow \infty} x^n$ 和 $\lim_{n \rightarrow \infty} x^n$ 。

解 早期的符号运算工具箱没有办法解决此问题,新版的 MATLAB 符号运算工具箱由于支持分段函数,所以可以较好地解决此类问题,具体的语句如下

```
>> syms x n real; f=x^n; L1=limit(f,n,inf), L2=limit(f,x,inf)
```

得出的结果均为分段函数,其中 L_2 的描述为 `piecewise([n == 0,1],[0 < n,Inf],[n < 0,0])`,这两个极限的结果可以解读成(其中 L_1 结果最末一个条件似乎有误,应该包括 $x=0$,即 $-1 < x < 1$)

$$L_1 = \begin{cases} 1, & x = 1 \\ \infty, & x > 1 \\ \text{无极限}, & x < -1 \\ 0, & 0 < x < 1 \text{ 或 } -1 < x < 0 \end{cases} \quad L_2 = \begin{cases} 1, & n = 0 \\ \infty, & n > 0 \\ 0, & n < 0 \end{cases}$$

有些函数,如 $\sin x$,在 $x \rightarrow \infty$ 的极限是不存在的,但可以通过 MuPAD 函数的选项求取其极限范围,可以通过下面的语句直接调用 MuPAD 底层函数求解相关问题

```
L = feval(symengine,'limit',f,'x=infinity','Intervals')
```

其中 `feval()` 函数可以直接调用 MuPAD 的底层函数 `limit()`,其变元为 MuPAD 写法。

例 3-7 假设 $a, b > 0$, 试求出 $f(t) = a \sin 8x^2 + b \cos(2x - 2)$ 函数在 $x \rightarrow \infty$ 时极限的区间。

解 利用底层的 MuPAD 命令可以解出该极限的区间为 $(-a - b, a + b)$ 。

```
>> syms a b positive, syms x; f=a*sin(8*x^2)+b*cos(-2*x+2);
    L=feval(symengine,'limit',f,'x=infinity','Intervals')
```

MATLAB 符号运算工具箱的新引擎——MuPAD 支持分段函数的使用, 分段函数的描述需要通过底层 MuPAD 语句来实现, 这对一般 MATLAB 使用者来说可能不方便, 所以我们编写了一个接口函数 `piecewise()` 来定义分段函数

```
function f=piecewise(varargin), str=[];
try
    for i=1:2:length(varargin),
        str=[str,[' ',varargin{i},' ',' ',varargin{i+1}'],' '];
    end
catch, error('Input arguments should be given in pairs.'), end
f=feval(symengine,'piecewise',str(1:end-1));
```

该函数的调用格式为 `f = piecewise(var1,var2,...)`, 其中输入变量 `var` 应该成对出现, 都应该由字符串给出, 前面一个是条件, 后面一个是该条件下的函数表达式, 其中条件中的逻辑运算应该由关键词 `and`、`or` 和 `not` 来表示。

该函数使用了 `try`, `catch` 结构, 确保输入变量成对出现, 否则将给出错误信息。由于生成的字符串最后一个字符是多余的逗号, 所以使用 `end-1` 下标舍弃它。

例 3-8 考虑例 2-24 的饱和非线性函数 $y = \begin{cases} 1.1 \operatorname{sign}(x), & |x| > 1.1 \\ x, & |x| \leq 1.1 \end{cases}$, 试绘制其曲线。

解 可以首先描述分段函数, 然后绘制该函数曲线, 其结果与例 2-24 中的完全一致。值得指出的是, 由于符号运算本身的局限性, 分段函数定义的符号变量不能用 `ezplot()` 函数直接绘制。

```
>> f=piecewise('abs(x)>1.1','1.1*sign(x)','abs(x)<=1.1','x');
    syms x; x0=-3:0.01:3; f1=subs(f,x,x0); plot(x0,f1)
```

如果 $|x| \leq 1.1$ 在数学上表示成 $-1.1 \leq x \leq 1.1$, 也可以将其理解成 $x \geq -1.1$ 且 $x \leq 1.1$, 这时相应的字符串表示应该为 `'x >= -1.1 and x <= 1.1'`。

3.1.3 多变量函数的极限

多变量函数的极限分为两类极限问题, 一类是累极限, 另一类是重极限。假设有二元函数 $f(x, y)$, 该函数的累极限定义为

$$L_1 = \lim_{x \rightarrow x_0} \left[\lim_{y \rightarrow y_0} f(x, y) \right], \text{ 或 } L_2 = \lim_{y \rightarrow y_0} \left[\lim_{x \rightarrow x_0} f(x, y) \right] \quad (3-1-3)$$

其中, x_0, y_0 既可以是数值也可以是函数。在 MATLAB 下, 函数的累极限可以通过下面的语句直接求出, 该函数嵌套地使用了 `limit()` 函数 `$L_1 = \text{limit}(\text{limit}(f, x, x_0), y, y_0)$` 或 `$L_1 = \text{limit}(\text{limit}(f, y, y_0), x, x_0)$` 。

例 3-9 试求出二元函数累极限 $\lim_{y \rightarrow \infty} \left[\lim_{x \rightarrow 1/\sqrt{y}} e^{-1/(y^2+x^2)} \frac{\sin^2 x}{x^2} \left(1 + \frac{1}{y^2}\right)^{x+a^2 y^2} \right]$ 。

解 由于涉及 \sqrt{y} , 在 MATLAB 下应该假设 y 为正数 (早期版本无需指出), 所以本例中的问题可以用下面的语句直接解出, 其极限值为 e^{a^2} 。

```
>> syms x a; syms y positive;
f=exp(-1/(y^2+x^2))*sin(x)^2/x^2*(1+1/y^2)^(x+a^2*y^2);
L=limit(limit(f,x,1/sqrt(y)),y,inf)
```

除了累极限之外, 还可以定义出函数的重极限

$$L = \lim_{\substack{x \rightarrow x_0 \\ y \rightarrow y_0}} f(x, y) \quad (3-1-4)$$

在一般情况下, 如果两个累极限的值相等, 则函数的重极限很可能等于这个值。另外, 也可能出现这两个语句都可以执行且得出不同结果的情形, 或二者相同但双重极限不存在的情形, 使用时应慎重, 可以尝试从不同的方向趋近目标, 观察是否能得出一致结论。

例 3-10 试求二重极限 $\lim_{\substack{x \rightarrow \infty \\ y \rightarrow \infty}} \left(\frac{xy}{x^2 + y^2} \right)^{x^2}$ 。

解 该函数的两个累极限可以由下面语句求出, 均为 0, 一般可以认为原函数的二重极限也为 0。为慎重起见, 当然还可以加入其他方向的极限来验证, 如 $x \rightarrow y^2$ 、 $y \rightarrow x^2$ 等, 都将给出一致的结论。

```
>> syms x y; f=(x*y/(x^2+y^2))^(x^2);
L1=limit(limit(f,x,inf),y,inf), L2=limit(limit(f,y,inf),x,inf)
L3=limit(limit(f,x,y^2),y,inf), L4=limit(limit(f,y,x^2),x,inf)
```

3.2 函数导数的解析解

3.2.1 函数的导数和高阶导数

如果函数和自变量都已知, 且均为符号变量, 则可以用 `diff()` 函数解出给定函数的各阶导数。`diff()` 函数的调用格式为 $f_1 = \text{diff}(f, x, n)$, 其中, f 为给定函数, x 为自变量, 这两个变量均应该为符号型的, n 为导数的阶次, 若省略 n 则将自动求取一阶导数; 如果 f 表达式中只有一个符号变量, 还可以省略变量 x 。

例 3-11 给定函数 $f(x) = \frac{\sin x}{x^2 + 4x + 3}$, 试求出 $\frac{d^4 f(x)}{dx^4}$ 。

解 这是本书开始时给出的第一个例子。函数的导数和高阶导数问题可以很容易地用 MATLAB 符号运算工具箱语句求解。可以首先申明 x 为符号变量, 再用 MATLAB 语句描述原函数, 然后调用 `diff()` 函数就能直接得出函数的一阶导数。

```
>> syms x; f=sin(x)/(x^2+4*x+3); f1=diff(f)
```

可以得出如下的结果 $f_1(x) = \frac{\cos x}{x^2 + 4x + 3} - \frac{(2x + 4) \sin x}{(x^2 + 4x + 3)^2}$ 。

由 `ezplot()` 函数可以直接绘制出原函数与得出一阶导数函数的曲线, 如图 3-2 所示。

```
>> ezplot(f,[0,5]), hold on; ezplot(f1,[0,5])
```

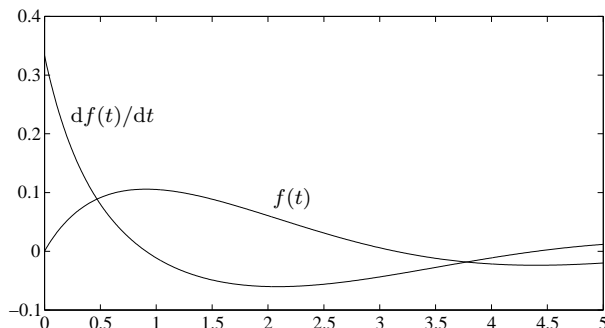


图 3-2 函数及其一阶导数图示

原函数的 4 阶导数可以直接由下面的语句求出

```
>> f4=diff(f,x,4); latex(f4) % 求解四阶导数
```

得出的结果比较冗长,由 L^AT_EX 可以显示出更好的效果如下

$$\begin{aligned} & \frac{\sin x}{x^2+4x+3} + 4 \frac{(2x+4) \cos x}{(x^2+4x+3)^2} - 12 \frac{(2x+4)^2 \sin x}{(x^2+4x+3)^3} + 12 \frac{\sin x}{(x^2+4x+3)^2} - 24 \frac{(2x+4)^3 \cos x}{(x^2+4x+3)^4} \\ & + 48 \frac{(2x+4) \cos x}{(x^2+4x+3)^3} + 24 \frac{(2x+4)^4 \sin x}{(x^2+4x+3)^5} - 72 \frac{(2x+4)^2 \sin x}{(x^2+4x+3)^4} + 24 \frac{\sin x}{(x^2+4x+3)^3} \end{aligned}$$

从化简的结果看,单纯采用 `simple()` 函数得出的化简结果不一定是令人满意的最简结果,而需要根据具体问题选择合适的化简方法。仔细分析上述结果可以发现,若按照 $\sin x$ 或 $\cos x$ 单独进行处理,则可能得出最简的结果。这样,分别运行 `collect(simple(f4),cos(x))` 和 `collect(simple(f4),sin(x))`,则可以得出下面给出的更简洁的结果

$$\begin{aligned} & 8(x^5 + 10x^4 + 26x^3 - 4x^2 - 99x - 102) \frac{\cos x}{(x^2 + 4x + 3)^4} + \\ & (x^8 + 16x^7 + 72x^6 - 32x^5 - 1094x^4 - 3120x^3 - 3120x^2 + 192x + 1581) \frac{\sin x}{(x^2 + 4x + 3)^5} \end{aligned}$$

MATLAB 现成的 `diff()` 函数还适合于求解给定函数更高阶的导数。例如,下面给出的命令一般可以在 10s 内获得该函数的 100 阶导函数 (MATLAB R2008a 及早期版本所需时间不到 1s)。

```
>> tic, diff(f,x,100); toc
```

例 3-12 试推导函数 $F(t) = t^2 f(t) \sin t$ 的 3 阶导函数公式,并得出 $f(t) = e^{-t}$ 时的 3 阶导数,将这样得出的结果与直接求导的结果相比较。

解 用 `sym('f(t)')` 函数可以定义出函数表达式 $f(t)$,这样由下面的语句可以直接推导出 $F(t)$ 函数的 3 阶导函数公式

```
>> syms t; y=sym('f(t)'); G=simple(diff(t^2*y*sin(t),t,3))
```

得出的结果为

$$\begin{aligned} \frac{d^3 F(t)}{dt^3} = & \left[\frac{d^3 f(t)}{dt^3} \sin t + 3 \frac{d^2 f(t)}{dt^2} \cos t - 3 \frac{df(t)}{dt} \sin t - f(t) \cos t \right] t^2 \\ & + \left[6 \frac{d^2 f(t)}{dt^2} \sin t + 12 \frac{df(t)}{dt} \cos t - 6 f(t) \sin t \right] t + 6 \frac{df(t)}{dt} \sin t + 6 f(t) \cos t \end{aligned}$$

下面语句则可以直接推导出当 $f(t) = e^{-t}$ 时原函数的 3 阶导数, 与直接求导结果完全一致。得出的导函数为 $y_1(t) = 2e^{-t}(t^2 \cos t + t^2 \sin t - 6t \cos t + 3 \cos t - 3 \sin t)$ 。

```
>> y1=simple(subs(G,y,exp(-t))), simple(diff(t^2*sin(t)*exp(-t),3)-y1)
```

例 3-13 假设矩阵函数如下, 试求出其三阶导数矩阵

$$\mathbf{H}(x) = \begin{bmatrix} 4 \sin 5x & e^{-4x^2} \\ 3x^2 + 4x + 1 & \sqrt{4x^2 + 2} \end{bmatrix}$$

解 MATLAB 语言的 `diff()` 函数可以直接用于已知矩阵函数 $\mathbf{H}(x)$ 的导数计算, 即对 $\mathbf{H}(x)$ 的每个元素 $h_{i,j}(x)$ 直接求导, 构成新的导数矩阵 $\mathbf{N}(x)$ 。

```
>> syms x; H=[4*sin(5*x), exp(-4*x^2); 3*x^2+4*x+1, sqrt(4*x^2+2)]
H1=diff(H,x,3)
```

这样得出的三阶导数矩阵为

$$\mathbf{H}_1(x) = \frac{d}{dx} \mathbf{H}(x) = \begin{bmatrix} -500 \cos 5x & 192x e^{-4x^2} - 512x^3 e^{-4x^2} \\ 0 & \frac{24\sqrt{2}x^3}{(2x^2+1)^{5/2}} - \frac{12\sqrt{2}x}{(2x^2+1)^{3/2}} \end{bmatrix}$$

3.2.2 参数方程的导数

若已知参数方程 $y = f(t), x = g(t)$, 则 $\frac{d^n y}{dx^n}$ 可以由递推公式求出

$$\begin{aligned} \frac{dy}{dx} &= \frac{f'(t)}{g'(t)} \\ \frac{d^2 y}{dx^2} &= \frac{d}{dt} \left(\frac{f'(t)}{g'(t)} \right) \frac{1}{g'(t)} = \frac{d}{dt} \left(\frac{dy}{dx} \right) \frac{1}{g'(t)} \\ &\vdots \\ \frac{d^n y}{dx^n} &= \frac{d}{dt} \left(\frac{d^{n-1} y}{dx^{n-1}} \right) \frac{1}{g'(t)} \end{aligned} \quad (3-2-1)$$

MATLAB 并没有提供可以直接用于参数方程的高阶导数求取的函数, 所以我们应该自己编写一个通用函数来完成这项工作。由前面的计算公式可见, 用递归函数的格式编程比较合适。

```
function result=paradiff(y,x,t,n)
if mod(n,1)~=0, error('n should positive integer, please correct')
else, if n==1, result=diff(y,t)/diff(x,t);
      else, result=diff(paradiff(y,x,t,n-1),t)/diff(x,t);
end, end
```

例 3-14 已知参数方程 $y = \frac{\sin t}{(t+1)^3}, x = \frac{\cos t}{(t+1)^3}$, 试求 $\frac{d^3 y}{dx^3}$ 。

解 由前面给出的函数调用格式, 可以立即得出所需的高阶导数。

```
>> syms t; y=sin(t)/(t+1)^3; x=cos(t)/(t+1)^3; f=simple(paradiff(y,x,t,3))
```

得出如下的结果

$$\frac{d^3 y}{dx^3} = \frac{-3(t+1)^7 [(t^4 + 4t^3 + 6t^2 + 4t - 23) \cos t - (4t^3 + 12t^2 + 32t + 24) \sin t]}{(t \sin t + \sin t + 3 \cos t)^5}$$

3.2.3 多元函数的偏导数

MATLAB 的符号运算工具箱中并未提供求取偏导数的专门函数, 这些偏导数仍然可以通过 `diff()` 函数直接实现。假设已知二元函数 $f(x, y)$, 若想求 $\partial^{m+n} f / (\partial x^m \partial y^n)$, 则可以用下面的函数嵌套地求出

`f1 = diff(diff(f, x, m), y, n)`, 或 `f1 = diff(diff(f, y, n), x, m)`

例 3-15 试求出二元函数 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$ 的一阶偏导数, 并用图形表示。

解 用下面的语句可直接求出 $\partial z / \partial x$ 与 $\partial z / \partial y$

```
>> syms x y; z=(x^2-2*x)*exp(-x^2-y^2-x*y); zx=simple(diff(z,x)), zy=diff(z,y)
```

其数学形式分别为

$$\frac{\partial z(x, y)}{\partial x} = -e^{-x^2 - y^2 - xy}(-2x + 2 + 2x^3 + x^2y - 4x^2 - 2xy)$$

$$\frac{\partial z(x, y)}{\partial y} = -x(x - 2)(2y + x)e^{-x^2 - y^2 - xy}$$

在 $x \in (-3, 2), y \in (-2, 2)$ 区域内生成网格, 则可以分别得出原函数及其偏导数的数值解。这样, 可以直接用下面的语句绘制出原函数的三维曲面, 如图 3-3(a) 所示

```
>> [x0,y0]=meshgrid(-3:.2:2,-2:.2:2); z0=subs(z,{x,y},{x0,y0});
surf(x0,y0,z0), zlim([-0.3 1.3]) % 直接绘制三维曲面
```

既然计算出了对两个自变量的一阶偏导数, 则可以调用 `quiver()` 函数绘制出引力线, 该引力线可以叠印在由 `contour()` 函数绘制出的等值线上, 如图 3-3(b) 所示。其中, 引力线绘制函数的详细信息可以由 `doc quiver` 命令进一步列出。

```
>> contour(x0,y0,z0,30), hold on; zx0=subs(zx,{x,y},{x0,y0});
zy0=subs(zy,{x,y},{x0,y0}); quiver(x0,y0,-zx0,-zy0) % 负梯度表示从高到低
```

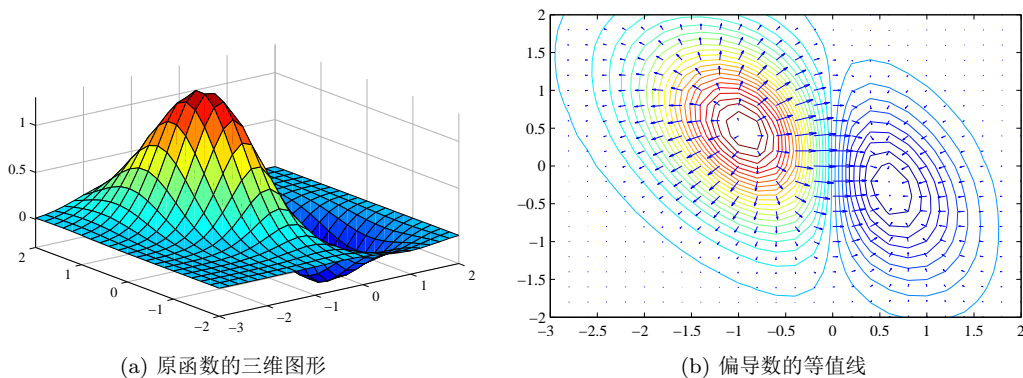


图 3-3 二元函数及其偏导数的图形表示

例 3-16 已知三元函数 $f(x, y, z) = \sin(x^2y)e^{-x^2y-z^2}$, 试求出偏导数 $\frac{\partial^4 f(x, y, z)}{\partial x^2 \partial y \partial z}$ 。

解 由下面的语句申明自变量及函数, 则可以用 MATLAB 语句立即得出所需的偏导函数

```
>> syms x y z; f=sin(x^2*y)*exp(-x^2*y-z^2);
    df=diff(diff(diff(f,x,2),y),z); F=simple(df)
```

得出的结果很冗长, 其数学表示为

$$F = -4ze^{-x^2y-z^2} \left[\cos(x^2y) - 10 \cos(x^2y) yx^2 + 4x^4 \sin(x^2y) y^2 + 4 \cos(x^2y) x^4 y^2 - \sin(x^2y) \right]$$

3.2.4 隐函数的偏导数

已知隐函数的数学表达式为 $f(x_1, x_2, \dots, x_n) = 0$, 则可以通过隐函数对相关变量的偏导数求出自变量之间的偏导数。具体可以用下面的公式求出 $\partial x_i / \partial x_j$

$$\frac{\partial x_i}{\partial x_j} = - \frac{\partial f(x_1, x_2, \dots, x_n) / \partial x_j}{\partial f(x_1, x_2, \dots, x_n) / \partial x_i} \quad (3-2-2)$$

由于 f 对 x_i, x_j 的偏导数可以分别由 `diff()` 函数求出, 故整个偏导数可以由它们的除法获得, 所以这样的问题可以由 `F1 = -diff(f, x_j)/diff(f, x_i)` 直接得出。

对二元函数 $f(x, y)$ 来说, 如果求出了 $\partial y / \partial x = F_1(x, y)$ (这里仍使用偏导数记号, 以便于此公式最终推导到一般多元函数), 则可以很容易地推导出其二阶导数的计算

$$F_2(x, y) = \frac{\partial^2 y}{\partial x^2} = \frac{\partial F_1(x, y)}{\partial x} + \frac{\partial F_1(x, y)}{\partial y} F_1(x, y) \quad (3-2-3)$$

更高阶的偏导数可以由下式递推求出

$$F_n(x, y) = \frac{\partial^n y}{\partial x^n} = \frac{\partial F_{n-1}(x, y)}{\partial x} + \frac{\partial F_{n-1}(x, y)}{\partial y} F_1(x, y) \quad (3-2-4)$$

上述命令用 MATLAB 语言可以很容易地实现, 后面将通过例子演示。此外, 上述方法可以直接推广到多元函数高阶导数的直接求取。根据这里给出的算法可以容易地编写出隐函数 f 的 n 偏导数函数 $f_1 = \partial^n y / \partial x^n$, 该函数调用格式为 `f1 = impldiff(f, x, y, n)`

```
function dy=impldiff(f,x,y,n)
if mod(n,1)~=0, error('n should positive integer, please correct')
else, F1=-simple(diff(f,x)/diff(f,y)); dy=F1;
    for i=2:n, dy=simple(diff(dy,x)+diff(dy,y)*F1);
end, end
```

例 3-17 考虑例 3-15 中给出的二元函数 $f(x, y) = (x^2 - 2x)e^{-x^2-y^2-xy} = 0$, 试求 $\frac{\partial y}{\partial x}$ 和 $\frac{\partial^3 y}{\partial x^3}$ 。

解 根据式 (3-2-2) 可以直接求解所需偏导数 $\partial y / \partial x$

```
>> syms x y; f=(x^2-2*x)*exp(-x^2-y^2-x*y); F1=impldiff(f,x,y,1)
```

可以得出偏导数为 $\frac{\partial y}{\partial x} = F_1(x, y) = \frac{-3x^3 + 6x^2 + 4x - 4}{2x(x + 2y)(x - 2)} - \frac{1}{2}$ 。在此基础上可以由下面公式求 $\frac{\partial^2 y}{\partial x^2}$ 。

利用前面介绍的递推公式还可以直接求出函数的二阶、三阶导数

```
>> F2=impldiff(f,x,y,2), F3=impldiff(f,x,y,2),[n,d]=numden(F3), simple(n)
```

这些语句可以求出 y 的高阶导数

$$F_2(x, y) = \frac{\partial^2 y}{\partial x^2} = -\frac{3x^4 - 12x^3 + 16x^2 - 8x + 8}{2x^2(x+2y)(x-2)^2} - \frac{(-3x^3 + 6x^2 + 4x - 4)^2}{2x^2(x+2y)^3(x-2)^2}$$

$$F_3(x, y) = \frac{\partial^3 y}{\partial x^3} = -\frac{-54x^9 + (324 - 54y)x^8 + (-54y^2 + 324y - 482)x^7 + (324y^2 - 616y - 408)x^6 + (-552y^2 + 264y + 1164)x^5 + (128y^3 + 72y^2 + 432y + 128)x^4 + (64y^4 - 384y^3 + 816y^2 - 416y - 888)x^3 + (-192y^4 + 768y^3 - 672y^2 + 384y + 96)x^2 + (384y^4 - 512y^3 + 384y^2 - 192y + 288)x - (256y^4 + 192y^2 + 96)}{x^3(x+2y)^5(x-2)^3}$$

例 3-18 试求出隐函数 $x^2 + xy + y^2 = 3$ 的各阶导数^[1]。

解 利用下面的语句可以直接求出函数的各阶导数, 另外, 由于 $x^2 + xy + y^2 = 3$, 可以将该条件带入得出的结果, 化简求出的函数的各阶导数。

```
>> f=x^2+x*y+y^2-3; F1=impldiff(f,x,y,1)
f2=impldiff(f,x,y,2); F2=subs(f2,x^2+x*y+y^2,3)
f3=impldiff(f,x,y,3); F3=subs(f3,x^2+x*y+y^2,3)
f4=impldiff(f,x,y,4); F4=subs(f4,x^2+x*y+y^2,3)
```

上面的命令可以得出

$$F_1 = -\frac{2x+y}{x+2y}, F_2 = -\frac{18}{(x+2y)^3}, F_3 = -\frac{162x}{(x+2y)^5}, F_4 = -\frac{648(4x^2+xy+y^2)}{(x+2y)^7}$$

其中, 使用的 `subs()` 命令有时似乎替换得不完全, F_4 还可以手工替换为 $F_4 = -\frac{1944(x^2+1)}{(x+2y)^7}$ 。

3.2.5 多元函数的 Jacobi 矩阵

假设有 n 个自变量的 m 个函数定义为

$$\begin{cases} y_1 = f_1(x_1, x_2, \dots, x_n) \\ y_2 = f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ y_m = f_m(x_1, x_2, \dots, x_n) \end{cases} \quad (3-2-5)$$

将相应的 y_i 对 x_j 求偏导, 则得出矩阵

$$\mathbf{J} = \begin{bmatrix} \partial y_1 / \partial x_1 & \partial y_1 / \partial x_2 & \cdots & \partial y_1 / \partial x_n \\ \partial y_2 / \partial x_1 & \partial y_2 / \partial x_2 & \cdots & \partial y_2 / \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial y_m / \partial x_1 & \partial y_m / \partial x_2 & \cdots & \partial y_m / \partial x_n \end{bmatrix} \quad (3-2-6)$$

该矩阵又称为 Jacobi 矩阵, 它在图像处理、机器人等诸多领域中均是很很有用的概念。Jacobi 矩阵可以由 MATLAB 的符号运算工具箱中的 `jacobian()` 函数直接求得, 其调用格式为 $\mathbf{J} = \text{jacobian}(\mathbf{y}, \mathbf{x})$, 其中, \mathbf{x} 是自变量构成的向量, \mathbf{y} 是由各个函数构成的向量。

例 3-19 已知球面坐标到直角坐标的变换公式为 $x = r \sin \theta \cos \phi, y = r \sin \theta \sin \phi, z = r \cos \theta$, 试求出函数向量 $[x, y, z]$ 对自变量向量 $[r, \theta, \phi]$ 的 Jacobi 矩阵。

解 可以先申明符号变量并描述 3 个函数, 这样可以用下面的语句求解出其 Jacobi 矩阵

```
>> syms r theta phi; x=r*sin(theta)*cos(phi); y=r*sin(theta)*sin(phi);
    z=r*cos(theta); J=jacobian([x; y; z],[r theta phi])
```

可以得出 Jacobi 矩阵为

$$\mathbf{J} = \begin{bmatrix} \sin \theta \cos \phi & r \cos \theta \cos \phi & -r \sin \theta \sin \phi \\ \sin \theta \sin \phi & r \cos \theta \sin \phi & r \sin \theta \cos \phi \\ \cos \theta & -r \sin \theta & 0 \end{bmatrix}$$

3.2.6 Hess 偏导数矩阵

对一个给定的 n 元标量函数 $f(x_1, x_2, \dots, x_n)$, 其 Hess 矩阵的定义为

$$\mathbf{H} = \begin{bmatrix} \partial^2 f / \partial x_1^2 & \partial^2 f / \partial x_1 \partial x_2 & \cdots & \partial^2 f / \partial x_1 \partial x_n \\ \partial^2 f / \partial x_2 \partial x_1 & \partial^2 f / \partial x_2^2 & \cdots & \partial^2 f / \partial x_2 \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f / \partial x_n \partial x_1 & \partial^2 f / \partial x_n \partial x_2 & \cdots & \partial^2 f / \partial x_n^2 \end{bmatrix} \quad (3-2-7)$$

可见, 该 Hess 矩阵实际上就是标量函数 $f(x, y)$ 的二阶偏导数矩阵。新版 MATLAB 提供了 `hessian()` 函数可以直接求出原函数的 Hess 矩阵, 调用格式为 $\mathbf{H} = \text{hessian}(f, \mathbf{x})$, 其中向量 $\mathbf{x} = [x_1, x_2, \dots, x_n]$ 。早期版本的 MATLAB 符号运算工具箱并未提供 `hessian()` 函数, 可以由 $\mathbf{H} = \text{jacobian}(\text{jacobian}(f, \mathbf{x}), \mathbf{x})$ 直接求解。

例 3-20 重新考虑例 3-15 中给出的二元函数, 试求其 Hess 矩阵。

解 下面语句可以直接求取该函数的 Hess 矩阵

```
>> syms x y; f=(x^2-2*x)*exp(-x^2-y^2-x*y); H=simple(hessian(f,[x,y]))
```

得出的结果 (或早期版本嵌套调用 `jacobian()` 函数) 为

$$\mathbf{H} = e^{-x^2-y^2-xy} \begin{bmatrix} 4x - 2(2x-2)(2x+y) - 2x^2 - (2x-x^2)(2x+y)^2 + 2 \\ 2x - (2x-2)(x+2y) - x^2 - (2x-x^2)(x+2y)(2x+y) \\ 2x - (2x-2)(x+2y) - x^2 - (2x-x^2)(x+2y)(2x+y) \\ x(x-2)(x^2+4xy+4y^2-2) \end{bmatrix}$$

3.3 积分问题的解析解

在微积分学中, 积分问题几种常用的表示方法为

$$F(x) = \int f(x) dx, \quad \int_a^b f(x) dx, \quad \int \cdots \int f(x_1, x_2, \dots, x_n) dx_n \cdots dx_2 dx_1 \quad (3-3-1)$$

其中函数 $f(\cdot)$ 称为被积函数。第一个积分表达式称为不定积分, 函数 $F(x)$ 称为原函数。第二个积分式称为定积分。第三个积分称为多重积分。在传统微积分学课程中, 求解不定积分问题通常需要灵活熟练地掌握和运用各种不同的积分方法, 如变量替换积分法和分部积分法等, 求解积分问题是否成功通常在很大程度上取决于用户的经验和技巧。本节侧重于介绍基于 MATLAB 的积分问题客观求解方法。

3.3.1 不定积分的推导

MATLAB 符号运算工具箱中提供了一个 `int()` 函数,可以直接用来求取符号函数的不定积分。该函数的调用格式为 `F = int(f,x)`。如果被积函数 f 中只有一个变量,则调用语句中的 x 可以省略。值得指出的是,该函数得出的结果 $F(x)$ 是积分原函数,实际的不定积分应该是 $F(x) + C$ 构成的函数族,其中, C 是任意常数。

对于可积的函数, MATLAB 符号运算工具箱提供的 `int()` 函数可以用计算机代替繁重的手工推导,立即得出原始问题的解。而对于不可积的函数来说, MATLAB 也是无能为力的。下面将通过例子介绍该函数的使用方法及应用。

例 3-21 考虑例 3-11 中给出的问题,用 `diff()` 函数可以直接求 $f(x)$ 函数的一阶导数。现在对得出的导数再进行积分,试检验是否可以得出一致的结果。

解 先定义原函数并对其求导,然后再对导数进行积分,则

```
>> syms x; y=sin(x)/(x^2+4*x+3); y1=diff(y); y0=int(y1)
```

得出的结果为 $y_0 = \frac{\sin x}{2(x+1)} - \frac{\sin x}{2(x+3)}$ 。现在对原函数求 4 阶导数,再对结果进行 4 次积分,则可以用下面语句判定正确性。由于得出的结果为 $\frac{\sin x}{(x+1)(x+3)}$,和原函数完全一致,故说明对给定的例子来说, MATLAB 得出的结果是正确的。

```
>> y4=diff(y,4); y0=int(int(int(int(y4)))); simple(y0)
```

例 3-22 试证明 $\int x^3 \cos^2 ax \, dx = \frac{x^4}{8} + \left(\frac{x^3}{4a} - \frac{3x}{8a^3}\right) \sin 2ax + \left(\frac{3x^2}{8a^2} - \frac{3}{16a^4}\right) \cos 2ax + C$ 。

解 用 MATLAB 语言的符号运算工具箱可以直接得出下面的化简结果

```
>> syms a x; f=simple(int(x^3*cos(a*x)^2,x))
```

得出的结果为

$$f = \frac{1}{16a^4} \left(2a^4 x^4 - 3 \cos 2ax + 6a^2 x^2 \cos 2ax + 4a^3 x^3 \sin 2ax - 6ax \sin 2ax + 3 \right)$$

然而,从得出的结果很难看出它是否和等式右侧完全一致,这就需要将等式右侧的表达式也输入到 MATLAB 工作空间,将二者相减并进行化简,从而得出其差为 $-3/(16a^4)$

```
>> f1=x^4/8+(x^3/(4*a)-3*x/(8*a^3))*sin(2*a*x)+...
```

```
(3*x^2/(8*a^2)-3/(16*a^4))*cos(2*a*x);
```

```
simple(f-f1) % 求两个结果的差
```

可见,二者并非完全相等,幸好得出的差为一个常数项,即使两种方法得出的积分原函数有差距,但因为形成积分函数族时需要加一个任意常数 C ,故可以认为题中的等式得证。

例 3-23 考虑两个不可积问题 $f(x) = e^{-x^2/2}$ 与 $g(x) = x \sin(ax^4)e^{x^2/2}$ 的积分问题求解。

解 首先考虑 $f(x) = e^{-x^2/2}$ 的不定积分求解。用 MATLAB 语言可以给出下面的语句

```
>> syms x; int(exp(-x^2/2))
```

得出的解为 $\sqrt{\pi/2} \operatorname{erf}(x/\sqrt{2})$ 。该积分虽然不可积,但可以用数学方法定义一个特殊符号函数

$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$,这样似乎可以写出定积分的解析表达式。事实上,这样的结果在工程中是不

能用的,必须得出相应的数值解,如得出给定 x 时可以由 `vpa()` 函数求取其具体数值。

再考虑一个真正不可积的函数 $g(x) = x \sin(ax^4)e^{x^2/2}$, 用 MATLAB 语句可以尝试对其直接求积分, 得出如下不可积的信息

```
>> syms a x; int(x*sin(a*x^4)*exp(x^2/2))
```

该函数将给出错误信息“Warning:Explicit integral could not be found”, 说明解析解不存在。

3.3.2 定积分与无穷积分计算

如果求出了某个函数 $f(x)$ 的不定积分为 $F(x) + C$, 则其在 (a, b) 区间内的定积分等于 $I = F(b) - F(a)$ 。在实际应用中, 有些函数不定积分可能不存在, 但仍然需要求取它的具体定积分值或无穷积分的值。例如, 前面定义的特殊函数 $\text{erf}(x)$ 虽然不定积分不可直接求解, 但需要得出 $\text{erf}(1.5)$ 的值, 则需要进行定积分的数值运算。

在 MATLAB 语言中仍然可以使用 `int()` 函数来求解定积分或无穷积分问题, 该函数的具体调用格式为 `I = int(f, x, a, b)`, 其中, x 为自变量, (a, b) 为定积分的积分区间, 求解无穷积分时, 允许将 a, b 设置成 `-Inf` 或 `Inf`, 如果得出的结果不是确切的数值, 还可以试着用 `vpa()` 函数得出定积分的解。有时, 定积分可以依赖定积分公式直接求解。如果 $f(x)$ 的不定积分原函数为 $F(x)$, 则其在 (a, b) 区间的定积分可以由 $F(b) - F(a)$ 求出。

例 3-24 仍考虑 $f(x) = e^{-x^2/2}$ 的定积分问题, 试求出当 $a = 0, b = 1.5$ 或 ∞ 时的定积分值。

解 若要求解该问题, 需要给出如下的 MATLAB 语句

```
>> syms x; I1=int(exp(-x^2/2),x,0,1.5), vpa(I1)
I2=int(exp(-x^2/2),x,0,inf)
```

得出 $I_1 = \sqrt{\pi/2} \text{erf}(3\sqrt{2}/4)$, 其高精度数值解为 $I_1 = 1.0858533176660165697024190765423$ 。无穷积分问题的解析解为 $I_2 = \sqrt{\pi/2}$ 。

例 3-25 试求解函数边界的定积分问题 $I(t) = \int_{\cos t}^{e^{-2t}} \frac{-2x^2 + 1}{(2x^2 - 3x + 1)^2} dx$ 。

解 MATLAB 提供的 `int()` 函数还可以求解函数积分区域的定积分问题, 题中的定积分可以由下面的 MATLAB 语句直接求解。对本例来说直接使用 `int()` 函数求解定积分好像有问题, 只好先求出不定积分, 再用定积分公式求出结果。

```
>> syms x t; f=(-2*x^2+1)/(2*x^2-3*x+1)^2; I1=int(f)
I=subs(I1,x,exp(-2*t))-subs(I1,x,cos(t))
```

得出的结果为 $I(t) = \frac{1}{2 \cos t - 1} - \frac{1}{\cos t - 1} - \frac{1}{2e^{-2t} - 1} + \frac{1}{e^{-2t} - 1}$ 。

3.3.3 多重积分问题的 MATLAB 求解

多重积分问题也可以在 MATLAB 语言环境中直接求解, 但需要根据实际情况先选择积分顺序, 可积的部分作为内积分, 然后再处理外积分。每步积分均采用 `int()` 函数处理, 如果交换积分顺序后仍然不能积出解析解, 则说明原积分问题没有解析解, 而需要采用数值方法求解原始的积分问题。多重积分的数值解法将在 3.7.5 节中介绍。

例 3-26 已知下面的三元函数 $F(x, y, z)$, 试求出 $\int \cdots \int F(x, y, z) dx^2 dy dz$

$$F(x, y, z) = -4ze^{-x^2y-z^2} (\cos x^2y - 10yx^2 \cos x^2y + 4x^4y^2 \sin x^2y + 4x^4y^2 \cos x^2y - \sin x^2y)$$

解 事实上, 此函数是例 3-16 中给出的 $f(x, y, z)$ 经偏导运算得出的, 故需要对求导过程进行逆向运算, 还原回原函数的结果。

对该函数进行积分。先对 z 积分一次, 对 y 积分一次, 再连续对 x 积分两次, 经过化简, 则得出结果为 $f_1 = e^{-x^2y-z^2} \sin x^2y$, 该结果完全还原例 3-16 中给出的原函数。

```
>> syms x y z; f0=-4*z*exp(-x^2*y-z^2)*(cos(x^2*y)-10*cos(x^2*y)*y*x^2+...
    4*sin(x^2*y)*x^4*y^2+4*cos(x^2*y)*x^4*y^2-sin(x^2*y));
    f1=int(f0,z); f1=int(f1,y); f1=int(f1,x); f1=simple(int(f1,x))
```

改变积分求解顺序, 变成 $z \rightarrow x \rightarrow x \rightarrow y$, 仍可以得出一致的结果。

```
>> f2=int(f0,z); f2=int(f2,x); f2=int(f2,x); f2=simple(int(f2,y))
```

例 3-27 试求解三重定积分问题 $\int_0^2 \int_0^\pi \int_0^\pi 4xz e^{-x^2y-z^2} dz dy dx$ 。

解 用如下的定积分求解语句可以立即计算出所需三重积分

```
>> syms x y z; int(int(int(4*x*z*exp(-x^2*y-z^2),x,0,2),y,0,pi),z,0,pi)
```

这时得出的结果为 $-(e^{-\pi^2} - 1)(\gamma + \ln(4\pi) - \text{Ei}(-4\pi))$, 其中, γ 为 Euler 常数, $\text{Ei}(z)$ 为指数积分, 即 $\text{Ei}(z) = \int_{-\infty}^z e^{-t} t^{-1} dt$ 。该函数虽然解析不可积, 但可以求出其数值解。这样, 原始问题的精确数值解可以由 `vpa(ans)` 得出, 其结果为 3.1080794020854127228346146476714。

3.4 函数的级数展开与级数求和问题求解

本节将介绍给定的单变量函数与多变量函数的 Taylor 幂级数展开、各种函数的 Fourier 级数展开、有穷级数与无穷级数求和、序列乘积等问题的计算机求解方法。

3.4.1 Taylor 幂级数展开

1. 单变量函数的 Taylor 幂级数展开

如果在 $x = 0$ 点附近进行 Taylor 幂级数展开, 则得出

$$f(x) = a_1 + a_2x + a_3x^2 + \cdots + a_kx^{k-1} + o(x^k) \quad (3-4-1)$$

其中, 系数 a_i 可以由下面的公式求出

$$a_i = \frac{1}{(i-1)!} \lim_{x \rightarrow 0} \frac{d^{i-1}}{dx^{i-1}} f(x), \quad i = 1, 2, 3, \cdots \quad (3-4-2)$$

该幂级数展开又称为 Maclaurin 级数, 若关于 $x = a$ 点进行展开, 则可以得出

$$f(x) = b_1 + b_2(x-a) + b_3(x-a)^2 + \cdots + b_k(x-a)^{k-1} + o[(x-a)^k] \quad (3-4-3)$$

其中,各个系数 b_i 可以如下求出

$$b_i = \frac{1}{(i-1)!} \lim_{x \rightarrow a} \frac{d^{i-1}}{dx^{i-1}} f(x), \quad i = 1, 2, 3, \dots \quad (3-4-4)$$

Taylor 幂级数展开可由符号运算工具箱的 `taylor()` 函数直接导出,其调用格式为

`F = taylor(f, x, a, 'Order', k)`, % 关于 $x = a$ 点进行 k 次 Taylor 幂级数展开

其中, f 为函数的符号表达式, x 为自变量,若函数只有一个自变量,则 x 可以省略。 k 为需要展开的项数,默认值为 6 项。如果不给出 a 则可以求出 $a = 0$ 的 Taylor 级数展开。早期版本 MATLAB 的 `taylor()` 函数调用格式与此不同,为 `F = taylor(f, x, k, a)`。下面将通过例子演示 Taylor 幂级数展开的方法。

例 3-28 仍考虑例 3-11 中给出的函数 $f(x) = \sin x / (x^2 + 4x + 3)$, 试求出该函数的 Maclaurin 幂级数展开的前 9 项,并关于 $x = 2$ 和 $x = a$ 分别进行原函数的 Taylor 幂级数展开。

解 先用下面的语句输入已知的函数,这样就可以调用 `taylor()` 函数导出其 Maclaurin 幂级数展开的前 9 项为

$$y(x) \approx -\frac{386459}{918540}x^8 + \frac{515273}{1224720}x^7 - \frac{3067}{7290}x^6 + \frac{4087}{9720}x^5 - \frac{34}{81}x^4 + \frac{23}{54}x^3 - \frac{4}{9}x^2 + \frac{1}{3}x$$

```
>> syms x; f=sin(x)/(x^2+4*x+3); y=taylor(f,x,'Order',9)
```

在传统微积分教材中,因为缺少必要的计算机支持,所以遗留了很大的缺陷,即若用有限项级数展开去逼近一个给定函数,逼近的效果如何? 在哪个区间适用,哪个区间不适用? 当然,有了 MATLAB 语言,这些简单的问题就可以轻而易举地解决了。图 3-4(a) 中给出了 9 项 Maclaurin 幂级数对原函数在 $(-1, 1)$ 区间的拟合效果,显然,当 x 较大时拟合不理想。

```
>> ezplot(f, [-1, 1]), hold on; ezplot(y, [-1, 1])
```

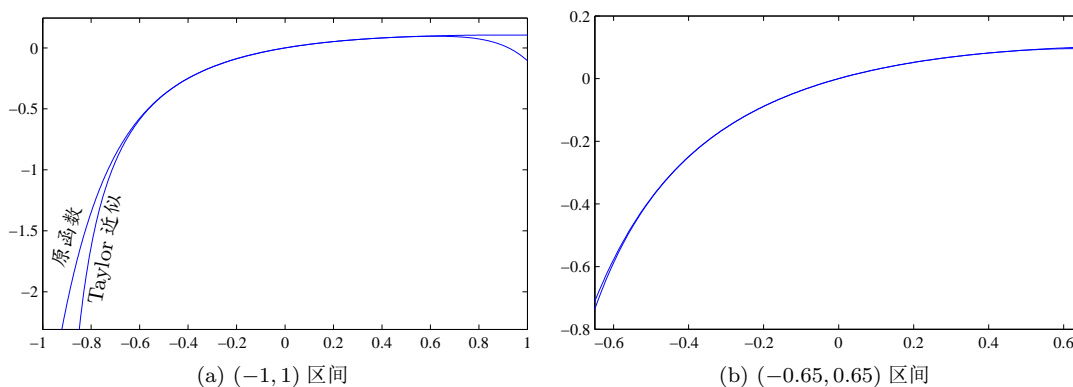


图 3-4 有限项 Maclaurin 幂级数近似效果

如果整个拟合区间缩减到 $[-0.65, 0.65]$, 则可以得出如图 3-4(b) 所示的拟合效果,可见拟合效果明显改观。由本例可见,利用 MATLAB 的绘图功能,拟合效果可以马上观察出来。

关于 $x = 2$ 的 Taylor 幂级数展开前 9 项可以使用如下语句直接导出

```
>> F=taylor(f,x,2,'Order',9) % 结果冗长,不全部列出
```

展开的前 4 项为

$$\frac{\sin 2}{15} + \left(\frac{\cos 2}{15} - \frac{8 \sin 2}{225} \right) (x-2) - \left(\frac{127 \sin 2}{6750} + \frac{8 \cos 2}{225} \right) (x-2)^2 + \left(\frac{23 \cos 2}{6750} + \frac{628 \sin 2}{50625} \right) (x-2)^3$$

若想导出关于某一点 $x = a$ 的 Taylor 幂级数展开,则可以给出如下语句

```
>> syms a; taylor(f,x,a,'Order',5)
```

考虑到篇幅,这里只显示展开表达式的前3项为

$$\frac{\sin a}{a^2 + 3 + 4a} + \left[\frac{\cos a}{a^2 + 3 + 4a} - \frac{(4 + 2a) \sin a}{(a^2 + 3 + 4a)^2} \right] (x - a) + \left[-\frac{\sin a}{(a^2 + 3 + 4a)^2} - \frac{\sin a}{2(a^2 + 3 + 4a)} - \frac{(a^2 \cos a + 3 \cos a + 4a \cos a - 4 \sin a - 2a \sin a)(4 + 2a)}{(a^2 + 3 + 4a)^3} \right] (x - a)^2$$

例 3-29 试对正弦函数 $y = \sin x$ 进行 Taylor 幂级数展开,观察不同阶次下的近似效果。

解 根据要求,可以给出如下的 MATLAB 语句,用循环的形式得出各次 Taylor 幂级数展开,得到如图 3-5 所示的拟合曲线。若拟合的阶次较低,则拟合效果较好的区间较小。增大拟合阶次,则拟合较好的区域将明显增大。对本例来说,若选择 $n = 16$,则在 $(-2\pi, 2\pi)$ 区间内的拟合效果将很理想。

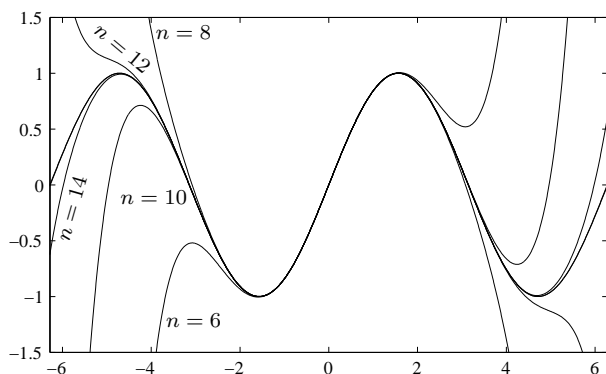


图 3-5 正弦函数的 Taylor 幂级数近似比较

```
>> syms x; y=sin(x); ezplot(y), hold on
```

```
for n=[6:2:16], p=taylor(y,x,'Order',n), ezplot(p), end
```

其中,16 阶 Taylor 幂级数展开式为

$$\sin x \approx x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362880} - \frac{x^{11}}{39916800} + \frac{x^{13}}{6227020800} - \frac{x^{15}}{1307674368000}$$

2. 多变量函数的 Taylor 幂级数展开

多变量函数 $f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$ 的 Taylor 幂级数展开可以写成

$$\begin{aligned} f(\mathbf{x}) = f(\mathbf{a}) &+ \left[(x_1 - a_1) \frac{\partial}{\partial x_1} + \dots + (x_n - a_n) \frac{\partial}{\partial x_n} \right] f(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{a}} + \\ &\frac{1}{2!} \left[(x_1 - a_1) \frac{\partial}{\partial x_1} + \dots + (x_n - a_n) \frac{\partial}{\partial x_n} \right]^2 f(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{a}} + \dots + \\ &\frac{1}{k!} \left[(x_1 - a_1) \frac{\partial}{\partial x_1} + \dots + (x_n - a_n) \frac{\partial}{\partial x_n} \right]^k f(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{a}} + \dots \end{aligned} \quad (3-4-5)$$

其中, $\mathbf{a} = [a_1, a_2, \dots, a_n]$ 为 Taylor 幂级数展开的中心点。新版 MATLAB 的符号运算工具箱的 `taylor()` 函数可以直接进行多变量函数 Taylor 幂级数展开。该函数的调用格式为

$F = \text{taylor}(f, [x_1, x_2, \dots, x_n], [a_1, a_2, \dots, a_n], 'Order', k)$

其中, $k-1$ 为展开的最高阶次, f 为原多变量函数。

例 3-30 试求例 3-15 中给出函数 $f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$ 的各种 Taylor 幂级数展开。

解 使用给出的函数就可以立即得出关于原点的 Taylor 幂级数展开

```
>> syms x y; f=(x^2-2*x)*exp(-x^2-y^2-x*y);
    F=taylor(f,[x,y],[0,0],'Order',8); collect(F,x)
```

其数学表示形式如下

$$F(x, y) \approx \frac{x^7}{3} + \left(y + \frac{1}{2}\right)x^6 + (2y^2 + y - 1)x^5 + \left(\frac{7y^3}{3} + \frac{3y^2}{2} - 2y - 1\right)x^4 \\ + (2y^4 + y^3 - 3y^2 - y + 2)x^3 + \left(y^5 + \frac{y^4}{2} - 2y^3 - y^2 + 2y + 1\right)x^2 + \left(\frac{y^6}{3} - y^4 + 2y^2 - 2\right)x$$

现在求取关于 $x=1, y=a$ 的幂级数展开, 则需要给出语句

```
>> syms a; F=taylor(f,[x,y],[1,a],'Order',3), F1(x)=simple(F)
```

便可以得出如下的幂级数展开式和化简表达式

$$F(x, y) \approx -e^{-a^2-a-1} \left[\left(\frac{a}{2} + 1\right)(a+2) - 2 \right] (x-1)^2 - e^{-a^2-a-1} (2a+1)(a-y) \\ - e^{-a^2-a-1} (a-y)^2 \left[(2a+1) \left(a + \frac{1}{2}\right) - 1 \right] + e^{-a^2-a-1} (a+2)(x-1) - e^{-a^2-a-1} \\ + e^{-a^2-a-1} (a-y)(x-1) \left[(2a+1) \left(\frac{a}{2} + 1\right) + (a+2) \left(a + \frac{1}{2}\right) - 1 \right] \\ F_1(x) = -\frac{1}{2} e^{-a^2-a-1} (4a^4 - 4a^3x - 8a^3y + 8a^3 + a^2x^2 + 4a^2xy - 12a^2x + 4a^2y^2 - 12a^2y \\ + 14a^2 + 4ax^2 + 10axy - 12ax + 4ay^2 - 12ay + 10a + 2xy - 4x - y^2 - 4y + 6)$$

3.4.2 Fourier 级数展开

给定周期性数学函数 $f(x)$, 其中, $x \in [-L, L]$, 且周期为 $T = 2L$, 可以人为地对该函数在其他区间上进行周期延拓, 使得 $f(x) = f(kT + x)$, k 为任意整数, 这样可以根据需要将其写成下面的级数形式

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi}{L}x + b_n \sin \frac{n\pi}{L}x \right) \quad (3-4-6)$$

其中,
$$\begin{cases} a_n = \frac{1}{L} \int_{-L}^L f(x) \cos \frac{n\pi x}{L} dx, & n = 0, 1, 2, \dots \\ b_n = \frac{1}{L} \int_{-L}^L f(x) \sin \frac{n\pi x}{L} dx, & n = 1, 2, 3, \dots \end{cases} \quad (3-4-7)$$

该级数称为 Fourier 级数, 而 a_n, b_n 又称为 Fourier 系数。若 $x \in (a, b)$, 则可以计算出周期 $L = (b-a)/2$, 引入新变量 \hat{x} , 使得 $x = \hat{x} + L + a$, 则可以将 $f(\hat{x})$ 映射成 $(-L, L)$ 区间上的函数, 可以对之进行 Fourier 级数展开, 再将 $\hat{x} = x - L - a$ 映射回 x 的函数即可。

MATLAB 语言未直接提供求解 Fourier 系数与级数的现成函数。其实由上述公式不难编写出解析或数值的 Fourier 级数求解函数。其中解析函数如下

```
function [F,A,B]=fseries(f,x,varargin)
[p,a,b]=default_vals({6,-pi,pi},varargin{:});
L=(b-a)/2; if a+b, f=subs(f,x,x+L+a); end
A=int(f,x,-L,L)/L; B=[]; F=A/2;
for n=1:p
    an=int(f*cos(n*pi*x/L),x,-L,L)/L; bn=int(f*sin(n*pi*x/L),x,-L,L)/L;
    A=[A,an]; B=[B,bn]; F=F+an*cos(n*pi*x/L)+bn*sin(n*pi*x/L);
end
if a+b, F=subs(F,x,x-L-a); end
```

我们为该函数编写了一个下级支持函数 `default_vals()`, 可以用于读取默认值。这个函数后面还将用到。该函数的内容为

```
function varargout=default_vals(vals,varargin)
if nargin~=length(vals), error('number of arguments mismatch');
else, nn=length(varargin)+1;
    varargout=varargin; for i=nn:nargout, varargout{i}=vals{i};
end, end, end
```

该函数的调用格式为 `[F,A,B]=fseries(f,x,p,a,b)`, 其中, f 为给定函数, x 为自变量, p 为展开项数, 默认值为 6, a, b 为 x 的区间, 可以省略取其默认值 $[-\pi, \pi]$, 得出的 A, B 为 Fourier 系数向量, F 为展开式。

例 3-31 试求给定函数 $y = x(x - \pi)(x - 2\pi)$, $x \in (0, 2\pi)$ 的 Fourier 级数展开。

解 上述给定函数的 Fourier 级数展开可以很自然地用下面的语句得出

```
>> syms x; f=x*(x-pi)*(x-2*pi); [F,A,B]=fseries(f,x,12,0,2*pi); F
```

这样, 可以得出前 12 项的 Fourier 级数展开为

$$f(x) = 12 \sin x + \frac{3}{2} \sin 2x + \frac{4}{9} \sin 3x + \frac{3}{16} \sin 4x + \frac{12}{125} \sin 5x + \frac{1}{18} \sin 6x + \frac{12}{343} \sin 7x \\ + \frac{3}{128} \sin 8x + \frac{4}{243} \sin 9x + \frac{3}{250} \sin 10x + \frac{12}{1331} \sin 11x + \frac{1}{144} \sin 12x$$

其实, 该展开的解析表达式为 $f(x) = \sum_{n=1}^{\infty} \frac{12}{n^3} \sin nx$ 。

由下面的语句可以得出 12 项 Fourier 级数展开对原函数的拟合情况, 如图 3-6 (a) 所示, 可见, 函数的拟合效果是很理想的。

```
>> ezplot(f,[0,2*pi]), hold on, ezplot(F,[0,2*pi])
```

如果想比较更大区间内的拟合效果, 如 $x \in (-\pi, 3\pi)$, 则可以给出下面的语句

```
>> ezplot(f,[-pi,3*pi]), hold on, ezplot(F,[-pi,3*pi])
```

这时的拟合效果如图 3-6 (b) 所示。可见, 在 $(0, 2\pi)$ 区间内拟合效果仍然很理想, 然而在其他区间内, Fourier 级数因为是定义在周期延拓基础上的, 所以和原函数完全不同。

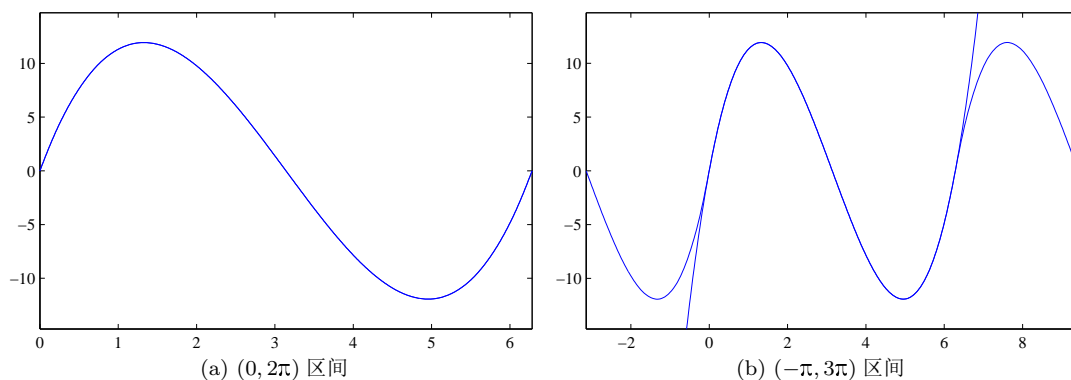


图 3-6 有限项 Fourier 级数近似效果比较

例 3-32 考虑 $(-\pi, \pi)$ 区间的方波信号, 假设 $x \geq 0$ 时 $y = 1$, 否则 $y = -1$, 试对该方波信号进行 Fourier 级数拟合, 并观察用多少项能有较好的拟合效果。

解 给定的函数可以由 $f(x) = |x|/x$ 表示, 故由下面语句可以容易地生成 x -轴数据点, 将其中的零值用 $[-\epsilon, \epsilon]$ 取代并重新排序, 则可以求出理论的方波数值。再用不同阶次的 Fourier 级数展开去拟合原来的方波函数, 得出的曲线如图 3-7 (a) 所示。

```
>> syms x; f=abs(x)/x; % 定义方波信号
xx=-pi:pi/200:pi; xx=xx(xx~=0); xx=sort([xx,-eps,eps]); % 剔除零值
yy=subs(f,x,xx); plot(xx,yy), % 绘制出理论值并保持坐标系
for n=1:20, f1=fseries(f,x,n); y1=subs(f1,x,xx); line(xx,y1); end
```

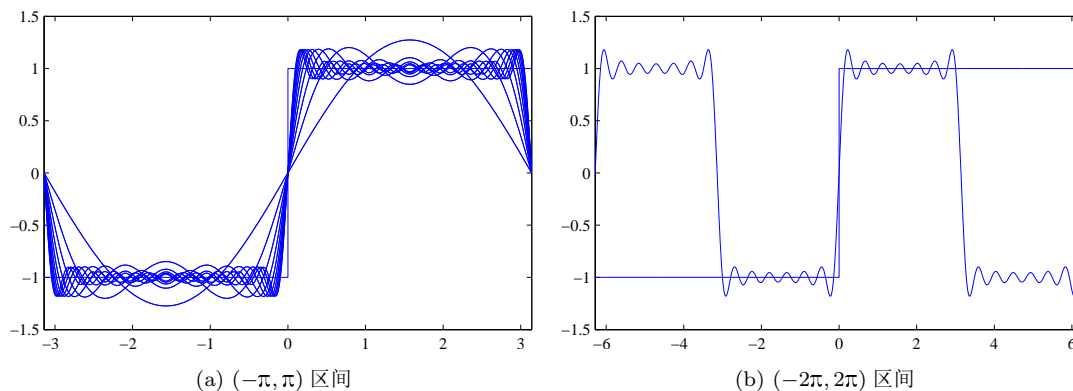


图 3-7 方波信号的 Fourier 级数逼近

从得出的结果看, 当阶次等于 10 左右就能得出较好的拟合, 再增加阶次也不会有显著的改善效果。取 $n = 14$, 则 Taylor 幂级数展开可以由下面的语句具体得出

```
>> f1=fseries(f,x,14)
```

可以得出 $f(x) \approx 4 \frac{\sin x}{\pi} + \frac{4 \sin 3x}{3\pi} + \frac{4 \sin 5x}{5\pi} + \frac{4 \sin 7x}{7\pi} + \frac{4 \sin 9x}{9\pi} + \frac{4 \sin 11x}{11\pi} + \frac{4 \sin 13x}{13\pi}$ 。从该结果可以总结出一般的展开公式为 $f(x) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2k-1)x}{2k-1}$ 。

同样地,如果比较区间扩展到 $(-2\pi, 2\pi)$,则由下面语句可以得出这时的拟合比较,如图 3-7(b) 所示。由于 Fourier 级数周期延拓区间的定义,它在指定区间以外与原函数无关

```
>> xx=[-2*pi:pi/200:2*pi]; xx=xx(xx~=0); xx=sort([xx,-eps,eps]);
yy=subs(f,x,xx); plot(xx,yy), y1=subs(f1,x,xx); line(xx,y1)
```

3.4.3 级数求和的计算

符号运算工具箱中提供的 `symsum()` 可以用于已知通项的有穷或无穷级数求和。该函数调用格式为 `S = symsum(f_k, k, k_0, k_n)`, 其中, f_k 为级数的通项, k 为级数自变量, k_0 和 k_n 为级数求和的起始项与终止项,它们也可以是无穷量 `inf`。该函数可以得出

$$S = \sum_{k=k_0}^{k_n} f_k \quad (3-4-8)$$

如果给出的 f_k 表达式中只含有一个变量,则在函数调用时可以省略 k 量。

例 3-33 计算有限项级数求和 $S = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \cdots + 2^{62} + 2^{63} = \sum_{i=0}^{63} 2^i$ 。

解 用数值计算方法可以由下面语句得出结果为 $1.844674407370955 \times 10^{19}$

```
>> format long; sum(2.^[0:63])
```

由于数值计算中使用了 `double` 数据类型,至多只能保留 16 位有效数字,所以得出的结果不是很精确。对这样的问题应该采用符号运算工具箱的 `symsum()` 函数,或至少将 2 定义为符号量,就可以用 `sum()` 函数求解。对原始问题稍扩展一步,一直到第 201 项的级数求和可以用下面的语句精确求出为 3213876088517980551083924184682325205044405987565585670602751,这是用数值算法无法精确做到的。

```
>> sum(sym(2).^[0:200]) % 或 syms k; symsum(2^k,0,200)
```

例 3-34 试求解无穷级数的和 $S = \frac{1}{1 \times 4} + \frac{1}{4 \times 7} + \frac{1}{7 \times 10} + \cdots + \frac{1}{(3n-2)(3n+1)} + \cdots$ 。

解 如果想借助 MATLAB 的符号运算工具箱,则可以立即得出结果为 $1/3$ 。

```
>> syms n; s=symsum(1/((3*n-2)*(3*n+1)),n,1,inf)
```

此级数求和亦可以用数值方法求得。假设求前 10 000 000 项的和,这时可以求出级数的和,结果为 0.33333332222165。但可以看出,得出无穷级数的和与解析解间存在很大差异,这个差异就是 `double` 数据类型引起的,它不能保留任意多小数位。

```
>> m=1:10000000; s1=sum(1./((3*m-2).*(3*m+1))); format long; s1
```

可见,即使选择的累加项数极多,耗时很长,得出的结果仍然有较大误差,达到 10^{-6} 级。从通项上看,当 $m = 10^7$ 时,通项值 10^{-15} 级,从表面上可能得出结论,似乎累加的结果误差不会太大。其实不然,由于双精度数值的有效位数有限,只有 16 位,所以计算通项时 16 位后的数字加到累加量上就消失了,这就是数值分析中经常所说的“大数吃小数”的现象,所以采用纯数值的方法,即使取再多的位数也不能精确得出正确的结果。

例 3-35 试求解含有变量的无穷级数的和 $J = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)(2x+1)^{2n+1}}$ 。

解 前面介绍的例子都是数值的例子,直接采用累加的方式就可以近似求出结果。这里给出的求和问题中含有变量 x ,所以仅靠数值运算的方式不可能得出该级数的和,而必须采用符号运算工具箱求解该问题,这需要给出下面的命令,最简结果为 $2 \operatorname{atanh}(1/(2x+1))$,并给出收敛条件 $x > 0$ 或 $x < -1$ 。早期版本的结果是 $\ln[(x+1)/x]$ 。可以用 `ezplot()` 函数验证二者是完全等效的。

```
>> syms n x; s1=symsum(2/((2*n+1)*(2*x+1)^(2*n+1)),n,0,inf); simple(s1)
ezplot(2*atanh(1/(2*x+1))), hold on, ezplot(log((x+1)/x))
```

例 3-36 试求解级数与极限综合问题 $\lim_{n \rightarrow \infty} \left[\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} \right) - \ln n \right]$ 。

解 前面介绍了级数求和,还介绍了极限的求解方法,所以这里给出的综合问题仍然能用 MATLAB 语言的符号运算工具箱直接求解。从题中给出的式子可见,其中包含级数求和项可以表示成 `symsum(1/m,1,n)`,这样原始的问题可以直接由下面的 MATLAB 语句求解

```
>> syms m n; limit(symsum(1/m,m,1,n)-log(n),n,inf)
```

该语句得出的结果为 Euler 常数 γ ,其值可以由 MATLAB 精确地显示出来,如使用 `vpa(ans,70)` 命令,这时 0.5772156649015328606065120900824024310421593359399235988057672348848677。

注意,求解该问题不能先求解无穷级数的和,然后再减去 $\ln n$,再求极限,这样做前后两项均为无穷大,求极限的结果将是不定式 NaN。

例 3-37 试求解下面的综合问题

$$S = \lim_{n \rightarrow \infty} \left[\left(1 + \frac{1}{n^2} \right) \sin \frac{\pi}{n^2} + \left(1 + \frac{2}{n^2} \right) \sin \frac{2\pi}{n^2} + \cdots + \left(1 + \frac{n-1}{n^2} \right) \sin \frac{(n-1)\pi}{n^2} \right]$$

解 从上面给出的问题可见,级数的通项公式为 $a_k = (1+k/n^2) \sin(k\pi/n^2)$,且 $k=1, 2, \cdots, n-1$,这样原始问题的解可以用下面语句直接求出,为 $S = \pi/2$ 。

```
>> syms n k; S=simple(limit(symsum((1+k/n^2)*sin(k*pi/n^2),k,1,n-1),n,inf))
```

3.4.4 序列求积问题

序列乘积的数学表示为

$$P = \prod_{n=a}^b f_n \quad (3-4-9)$$

新版的 MATLAB 符号运算工具箱提供了求解函数 `symprod()` 直接求取序列求积问题,其语句格式为 `P = symprod(f_n, n, a, b)`。

例 3-38 试求出序列的有限项乘积 $P_n = \prod_{k=1}^n \left(1 + \frac{1}{k^3} \right)$ 和无穷项乘积。

解 由下面的语句可以立即得出该序列的有限项乘积与无穷乘积为

```
>> syms k n; P1=symprod(1+1/k^3,k,1,n); P1=simple(P1)
P2=symprod(1+1/k^3,k,1,inf); P2=simple(P2) % 得出的结果如下
```

$$P_1 = \frac{\cosh\left(\frac{\sqrt{3}}{2}\pi\right) \Gamma\left(n + \frac{1-\sqrt{3}i}{2}\right) \Gamma\left(n + \frac{1+\sqrt{3}i}{2}\right) (n+1)^3}{\pi \Gamma(n+2)^2}, \quad P_2 = \frac{\cosh\left(\frac{\sqrt{3}}{2}\pi\right)}{\pi}$$

例 3-39 试求出下面无穷级数的和

$$S = 1 - \frac{1}{2} + \frac{1 \times 3}{2 \times 4 \times 6} - \frac{1 \times 3 \times 5}{2 \times 4 \times 6} + \frac{1 \times 3 \times 5 \times 7}{2 \times 4 \times 6 \times 8} - \frac{1 \times 3 \times 5 \times 7 \times 9}{2 \times 4 \times 6 \times 8 \times 10} + \cdots$$

解 这个问题是级数求和问题,其通项公式为 $(-1)^n \prod_{k=1}^n [(2k-1)/(2k)]$, 且 $n = 0, 1, \cdots, \infty$, 故由下面的语句可以直接得出原问题的解为 $S = \sqrt{2}/2$

```
>> syms k n, S=symsum((-1)^n*symprod((2*k-1)/(2*k),k,1,n),n,0,inf)
```

例 3-40 试求出 $P = \prod_{n=1}^{\infty} \left(1 + \frac{x}{n}\right) e^{-x/n}$ 。

解 下面语句可以直接得出原问题的解

```
>> syms n x; P=symprod((1+x/n)*exp(-x/n),n,1,inf)
```

得出解的分段函数如下

$$P = \begin{cases} 0, & x \text{ 为负整数} \\ e^{-\gamma x} / \Gamma(x+1), & \text{其他, 其中 } \gamma \text{ 为 Euler 常数} \end{cases}$$

3.5 曲线积分与曲面积分的计算

MATLAB 语言并未直接提供曲线积分和曲面积分的现成函数。本节将介绍两类曲线、曲面积分的概念,引入它们转换成一般积分问题的算法,并介绍利用 MATLAB 语言的符号运算工具箱直接求解曲线、曲面积分的解析解方法。

3.5.1 曲线积分及 MATLAB 求解

1. 第一类曲线积分

曲线积分在高等数学中一般分为第一类曲线积分和第二类曲线积分。其中,第一类曲线积分问题起源于对不均匀分布的空间曲线总质量的求取^[2]。假设在空间曲线 l 上的密度函数为 $f(x, y, z)$, 则其总质量,亦即第一类曲线积分的值可以由下面的式子直接求出

$$I_1 = \int_l f(x, y, z) ds \quad (3-5-1)$$

其中, s 为曲线上某点的弧长,所以这类曲线积分又称为对弧长的曲线积分。若 x, y, z 均由参数方程 $x = x(t), y = y(t), z = z(t)$ 给出,则可以将这些量直接调入 $f(\cdot)$ 函数,而弧长可以表示成

$$ds = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2} dt, \text{ 简记作 } ds = \sqrt{x_t^2 + y_t^2 + z_t^2} dt \quad (3-5-2)$$

则可以将这类曲线积分也变换成对参数 t 的普通定积分问题

$$I = \int_{t_m}^{t_M} f[x(t), y(t), z(t)] \sqrt{x_t^2 + y_t^2 + z_t^2} dt \quad (3-5-3)$$

若被积函数 $f(x, y)$ 为二元函数, 也可以用相应的转换方法将其转换成普通积分问题, 故用 MATLAB 语言可以求出第一类曲线积分的值。

例 3-41 试求 $\int_l \frac{z^2}{x^2 + y^2} ds$, 其中 l 为螺线, $x = a \cos t, y = a \sin t, z = at, (0 \leq t \leq 2\pi, a > 0)$ 。

解 用下面的语句可以立即得出曲线积分值为 $I = 8\sqrt{2}\pi^3 a/3$

```
>> syms t; syms a positive; x=a*cos(t); y=a*sin(t); z=a*t;
I=int(z^2/(x^2+y^2)*sqrt(diff(x,t)^2+diff(y,t)^2+diff(z,t)^2),t,0,2*pi)
```

例 3-42 试求 $\int_l (x^2 + y^2) ds$, 其中 l 曲线为 $y = x$ 与 $y = x^2$ 围成的正向曲线。

解 应该用下面的指令绘制出给定的两条曲线, 如图 3-8 所示。

```
>> x=0:.001:1.2; y1=x; y2=x.^2; plot(x,y1,x,y2), hold on, ii=find(x<=1);
xx=[x(ii),x(ii(end):-1:1)]; yy=[y2(ii), y1(ii(end):-1:1)]; fill(xx,yy,'g')
```

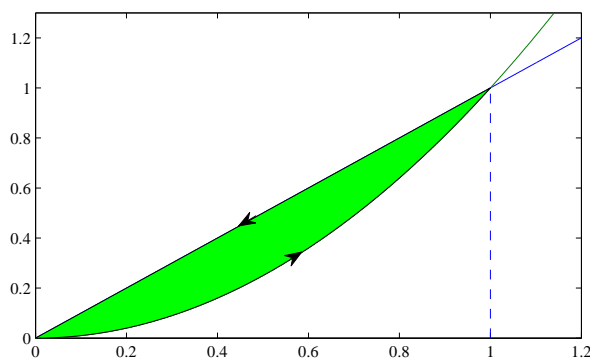


图 3-8 积分曲线示意图

可见, 可以将原来的积分问题化成两段曲线的积分问题来求解。故应该给出如下的指令, 求解出两段曲线的积分值, 将其相加则得出原问题的解为 $I = -\frac{2}{3}\sqrt{2} + \frac{349}{768}\sqrt{5} + \frac{7}{512}\ln(-2 + \sqrt{5})$

```
>> syms x; y1=x; y2=x^2; I1=int((x^2+y2^2)*sqrt(1+diff(y2,x)^2),x,0,1);
I2=int((x^2+y1^2)*sqrt(1+diff(y1,x)^2),x,1,0); I=I2+I1
```

2. 第二类曲线积分

第二类曲线积分问题又称为对坐标的曲线积分, 它起源于变力 $\vec{f}(x, y, z)$ 沿曲线 l 移动时做功的研究。这类曲线积分的数学表达式为

$$I_2 = \int_l \vec{f}(x, y, z) \cdot d\vec{s} \quad (3-5-4)$$

其中, $\vec{f}(x, y, z)$ 为向量, 可以写成 $\vec{f} = [P(x, y, z), Q(x, y, z), R(x, y, z)]$, 曲线 $d\vec{s}$ 亦为向量, 若曲线可以由参数方程表示成 t 的函数, 记作 $x(t), y(t), z(t)$, 则可以将 $d\vec{s}$ 表示成

$$d\vec{s} = \left[\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt} \right]^T dt \quad (3-5-5)$$

则两个向量的点乘可以由这两个向量直接得出, 这样就可以利用 MATLAB 语言求出第二类曲线积分的值。

例 3-43 试求出曲线积分 $\int_l \frac{x+y}{x^2+y^2} dx - \frac{x-y}{x^2+y^2} dy$, l 为正向圆周 $x^2 + y^2 = a^2$ 。

解 若想按圆周曲线进行积分, 则可以写出参数方程 $x = a \cos t, y = a \sin t, (0 \leq t \leq 2\pi)$, 这样, 用下面的方法可以直接求出曲线积分为 2π

```
>> syms t; syms a positive; x=a*cos(t); y=a*sin(t);
F=[(x+y)/(x^2+y^2), -(x-y)/(x^2+y^2)]; ds=[diff(x,t);diff(y,t)];
I=int(F*ds,t,2*pi,0) % 正向圆周
```

例 3-44 试求出曲线积分的值 $\int_l (x^2 - 2xy)dx + (y^2 - 2xy)dy$, l 为抛物线 $y = x^2 (-1 \leq x \leq 1)$ 。

解 其实, 曲线给出的方程已经是关于 x 的参数方程, 且 x 对 x 的导数显然为 1, 故可以用下面的语句求出曲线积分的值为 $-14/15$

```
>> syms x; y=x^2; F=[x^2-2*x*y, y^2-2*x*y]; ds=[1; diff(y,x)];
I=int(F*ds,x,-1,1)
```

3.5.2 曲面积分与 MATLAB 语言求解

1. 第一类曲面积分

第一类曲面积分的数学定义为

$$I = \iint_S \phi(x, y, z) dS \quad (3-5-6)$$

其中, dS 为小区域的面积, 故这类积分又称为对面积的曲面积分。曲面 S 由 $z = f(x, y)$ 给出, 则该积分可以转换成 $x-y$ 平面的二重积分为

$$I = \iint_{\sigma_{xy}} \phi[x, y, f(x, y)] \sqrt{1 + f_x^2 + f_y^2} dx dy \quad (3-5-7)$$

其中, σ_{xy} 为积分区域。

例 3-45 试求出 $\iint_S xyz dS$, 其中积分曲面 S 是由四个平面 $x=0, y=0, z=0$ 和 $x+y+z=a$ 围成的外侧面, 且 $a > 0$ 。

解 记这四个平面为 S_1, S_2, S_3, S_4 , 则原积分可以由 $\iint_S = \iint_{S_1} + \iint_{S_2} + \iint_{S_3} + \iint_{S_4}$ 求出。考虑 S_1, S_2, S_3 平面, 由于被积函数的值为 0, 故这些积分也为 0, 所以只需研究 S_4 的曲线积分。 S_4 平面的数学表示为 $z = a - x - y$, 故由下面的语句可以求出曲面积为 $I = \sqrt{3}a^5/120$ 。


```
>> syms x y; syms a positive; z=a-x-y;
I=int(int(x*y*z*sqrt(1+diff(z,x)^2+diff(z,y)^2),y,0,a-x),x,0,a)
```

若曲面由参数方程

$$x = x(u, v), \quad y = y(u, v), \quad z = z(u, v) \quad (3-5-8)$$

给出, 则曲面积分可以由下面的公式求出

$$I = \iint_{\Sigma} \phi[x(u, v), y(u, v), z(u, v)] \sqrt{EG - F^2} du dv \quad (3-5-9)$$

$$\text{式中, } E = x_u^2 + y_u^2 + z_u^2, \quad F = x_u x_v + y_u y_v + z_u z_v, \quad G = x_v^2 + y_v^2 + z_v^2 \quad (3-5-10)$$

例 3-46 试求出曲面积分 $\iint (x^2 y + z y^2) dS$, 其中 S 为螺旋曲面 $x = u \cos v, y = u \sin v, z = v$ 的 $0 \leq u \leq a, 0 \leq v \leq 2\pi$ 部分。

解 由上述公式可以立即得出积分结果为 $I = \frac{1}{8}\pi^2 \left(2a(a^2 + 1)^{3/2} - a\sqrt{a^2 + 1} - \operatorname{arcsinh} a \right)$ 。

```
>> syms u v; syms a positive;
x=u*cos(v); y=u*sin(v); z=v; f=x^2*y+z*y^2;
E=simple(diff(x,u)^2+diff(y,u)^2+diff(z,u)^2);
F=diff(x,u)*diff(x,v)+diff(y,u)*diff(y,v)+diff(z,u)*diff(z,v);
G=simple(diff(x,v)^2+diff(y,v)^2+diff(z,v)^2);
I=int(int(f*sqrt(E*G-F^2),u,0,a),v,0,2*pi)
```

2. 第二类曲面积分

第二类曲面积分又称为对坐标的曲面积分。其数学定义为

$$I = \iint_{S^+} \vec{\Gamma} \cdot d\vec{V} = \iint_{S^+} P(x, y, z) dy dz + Q(x, y, z) dx dz + R(x, y, z) dx dy \quad (3-5-11)$$

其中, 正向曲面 S^+ 由 $z = f(x, y)$ 给出, 被积函数 $\vec{\Gamma} = [P, Q, R]$ 为行向量, 而 $d\vec{V} = [dy dz, dx dz, dx dy]^T$ 为列向量。这类曲面积分问题可以转换成第一类曲面积分

$$I = \iint_{S^+} [P(x, y, z) \cos \alpha + Q(x, y, z) \cos \beta + R(x, y, z) \cos \gamma] dS \quad (3-5-12)$$

其中, z 由 $f(x, y)$ 代替, 且

$$\cos \alpha = \frac{-f_x}{\sqrt{1 + f_x^2 + f_y^2}}, \quad \cos \beta = \frac{-f_y}{\sqrt{1 + f_x^2 + f_y^2}}, \quad \cos \gamma = \frac{1}{\sqrt{1 + f_x^2 + f_y^2}} \quad (3-5-13)$$

这样, 分母上的 $\sqrt{1 + f_x^2 + f_y^2}$ 正好和式 (3-5-7) 中的项抵消, 故整个曲面积分可以写成

$$I = \iint_{\sigma_{xy}} -P f_x dx dy - Q f_y dx dz + R dy dz \quad (3-5-14)$$

若曲面由参数方程式(3-5-8)给出,则可以由下面的方程求出

$$\cos \alpha = \frac{A}{\sqrt{A^2+B^2+C^2}}, \quad \cos \beta = \frac{B}{\sqrt{A^2+B^2+C^2}}, \quad \cos \gamma = \frac{C}{\sqrt{A^2+B^2+C^2}} \quad (3-5-15)$$

其中, $A = y_u z_v - z_u y_v$, $B = z_u x_v - x_u z_v$, $C = x_u y_v - y_u x_v$ 。这样,由得出的第一类曲面积分转换成二重积分会发现,式(3-5-15)的分母正好和 $\sqrt{EG-F^2}$ 抵消。这时整个曲面积分可以简化成

$$I = \iint_{S^+} [AP(u, v) + BQ(u, v) + CR(u, v)] du dv \quad (3-5-16)$$

例 3-47 试求出曲面积分 $\iint_S (xy+z) dy dz$, 其中 S 是椭球面 $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$ 的上半部, 且积分沿椭球面的上面。

解 可以引入参数方程 $x = a \sin u \cos v$, $y = b \sin u \sin v$, $z = c \cos u$, 且 $0 \leq u \leq \frac{\pi}{2}$, $0 \leq v \leq 2\pi$, 这样,原始曲面积分问题可以转换为一般双重积分问题

$$\int_0^{2\pi} \int_0^{\pi} CR du dv, \quad \text{其中 } R = xy + z, C = x_u y_v - y_u x_v$$

可以用下面的语句求出所需的曲面积为 $2abc\pi/3$ 。

```
>> syms u v; syms a b c positive;
x=a*sin(u)*cos(v); y=b*sin(u)*sin(v); z=c*cos(u);
C=diff(x,u)*diff(y,v)-diff(y,u)*diff(x,v); R=x*y+z
I=int(int(R*C,u,0,pi/2),v,0,2*pi)
```

3.6 数值微分问题

前面介绍了已知原型函数,可以通过 `diff()` 函数求取各阶导数解析解的方法,并得出结论,高达 100 阶的导数也可以用 MATLAB 语言在几秒钟的时间内直接求出。应该指出,前面介绍的解析解方法的前提是原型函数为已知的。如果函数表达式未知,只有实验数据,在实际应用中经常也有求导的要求,这样的问题就不能用前面的方法获得问题的解析解。要求解这样的问题,需要引入数值算法得出所需问题的解。由于在 MATLAB 语言中没有现成的数值微分函数,所以本节将先介绍数值微分算法,介绍其中较好算法的 MATLAB 实现,最后将通过例子演示数值微分程序。

3.6.1 数值微分算法

假设已经等间隔地测出了一组数据 (t_i, y_i) , 且已知时间间隔为 Δt , 由高等数学中导数的定义可知,若 $\Delta t \rightarrow 0$, 则相邻两点的差值除以间隔 Δt 就是该点处的导数,由此可以引入前向差分公式

$$y'_i = \frac{\Delta y_i}{\Delta t} = \frac{y_{i+1} - y_i}{\Delta t} \quad (3-6-1)$$

类似地,还可以引入后向差分公式

$$y'_i = \frac{\Delta y_i}{\Delta t} = \frac{y_i - y_{i-1}}{\Delta t} \quad (3-6-2)$$

遗憾的是,这两种微分算法的精度都是 $o(\Delta t)$ 级的,当 Δt 稍大时,产生的误差会很大。经实践检验,利用基于前向和后向差分的数值微分算法求取高阶微分时的精度一般都是很低的,所以这里只介绍两种中心差分的算法。首先定义一阶微分为

$$y'_i = \frac{\Delta y_i}{\Delta t} = \frac{y_{i+1} - y_{i-1}}{2\Delta t} \quad (3-6-3)$$

记
$$\tilde{f}'(x) = \frac{f(x + \Delta t) - f(x - \Delta t)}{2\Delta t} \quad (3-6-4)$$

由 Taylor 级数展开可以将上式进一步写成

$$\begin{aligned} \tilde{f}(x) &= \frac{f(x) + \Delta t f'(x) + \Delta t^2 f''(x)/2! + \Delta t^3 f'''(\xi)/3! + o(\Delta t^4)}{2\Delta t} - \\ &\quad \frac{f(x) - \Delta t f'(x) + \Delta t^2 f''(x)/2! - \Delta t^3 f'''(\xi)/3! + o(\Delta t^4)}{2\Delta t} = f'(x) + \frac{\Delta t^3}{3!} f'''(\xi) \end{aligned} \quad (3-6-5)$$

可见这种中心差分的算法精度为 $o(\Delta t^2)$ 。

这里给出一组比此算法精度更高的高阶中心差分算法,这些算法的精度为 $o(\Delta t^4)$

$$\begin{aligned} y'_i &= \frac{-y_{i+2} + 8y_{i+1} - 8y_{i-1} + y_{i-2}}{12\Delta t} \\ y''_i &= \frac{-y_{i+2} + 16y_{i+1} - 30y_i + 16y_{i-1} - y_{i-2}}{12\Delta t^2} \\ y'''_i &= \frac{-y_{i+3} + 8y_{i+2} - 13y_{i+1} + 13y_{i-1} - 8y_{i-2} + y_{i-3}}{8\Delta t^3} \\ y^{(4)}_i &= \frac{-y_{i+3} + 12y_{i+2} - 39y_{i+1} + 56y_i - 39y_{i-1} + 12y_{i-2} - y_{i-3}}{6\Delta t^4} \end{aligned} \quad (3-6-6)$$

3.6.2 中心差分方法及其 MATLAB 实现

从前面的介绍可知,式(3-6-6)中给出的微分算法有 $o(\Delta t^4)$ 级精度,因而即使 Δt 不趋于 0 时,仍能得出较好的近似微分。所以这里采用该公式为所选算法,可以编写出一个 MATLAB 函数,其内容为

```
function [dy,dx]=diff_ctr(y,Dt,n)
yx1=[y 0 0 0 0]; yx2=[0 y 0 0 0]; yx3=[0 0 y 0 0];
yx4=[0 0 0 y 0]; yx5=[0 0 0 0 y]; yx6=[0 0 0 0 0 y];
switch n
case 1, dy=(-diff(yx1)+7*diff(yx2)+7*diff(yx3)-diff(yx4))/(12*Dt); L0=3;
case 2, dy=(-diff(yx1)+15*diff(yx2)-15*diff(yx3)+diff(yx4))/(12*Dt^2);L0=3;
case 3, dy=(-diff(yx1)+7*diff(yx2)-6*diff(yx3)-6*diff(yx4)+...
          7*diff(yx5)-diff(yx6))/(8*Dt^3); L0=5;
case 4, dy=(-diff(yx1)+11*diff(yx2)-28*diff(yx3)+28*diff(yx4)-...
          11*diff(yx5)+diff(yx6))/(6*Dt^4);L0=5;
end
dy=dy(L0+1:end-L0); dx=(1:length(dy))+L0-2-(n>2))*Dt;
```

这样编写的 M-函数调用格式为 $[d_y, d_x] = \text{diff_ctr}(y, \Delta t, n)$, 其中, y 为给定的等间距的实测数据构成的向量, Δt 为自变量的间距, n 为所需的导数阶次。向量 d_y 为得出的导数向量, 而 d_x 为相应的自变量向量。注意这两个向量的长度比 y 短。

例 3-48 这里仍采用例 3-11 中给出的函数(这里给出函数原型是为了精度检验)。由于原型函数已知, 所以可以求出导数的解析解, 从中求出精确的值。试用数值微分求取原函数的 1~4 阶导数, 并和解析解比较精度。

解 生成一个横坐标点组成的向量 x 。另外由已知的原型函数, 可以立即得出函数各阶导数的解析解, 并将已知的横坐标点代入, 即可以得出各阶导数精确的数值解, 可以用于对照

```
>> h=0.05; x=0:h:pi; syms x1; y=sin(x1)/(x1^2+4*x1+3);
yy1=diff(y); f1=subs(yy1,x1,x); yy2=diff(yy1); f2=subs(yy2,x1,x);
yy3=diff(yy2); f3=subs(yy3,x1,x); yy4=diff(yy3); f4=subs(yy4,x1,x);
```

假设由该原型函数可以生成一些数据点 y_i , 由这些点可以拟合出曲线的 1~4 阶导数。下面的语句可以通过数值的方法获得已知数据点处的各阶导数, 还可以绘制出曲线, 将数值导数和由解析解计算出来的导数在相同的坐标系下绘制出来, 如图 3-9 所示。可以看出, 由中心差分算法获得的导数是很精确的, 其误差从图上是看不出来的。

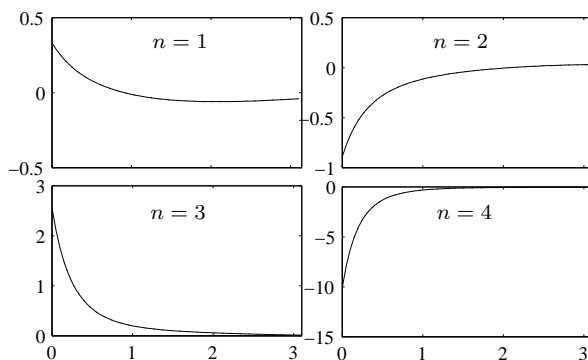


图 3-9 各阶导数比较

```
>> y=sin(x)./(x.^2+4*x+3); [y1,dx1]=diff_ctr(y,h,1);
subplot(221), plot(x,f1,dx1,y1,':'); [y2,dx2]=diff_ctr(y,h,2);
subplot(222), plot(x,f2,dx2,y2,':'); [y3,dx3]=diff_ctr(y,h,3);
subplot(223), plot(x,f3,dx3,y3,':'); [y4,dx4]=diff_ctr(y,h,4);
subplot(224), plot(x,f4,dx4,y4,':')
```

下面定量地分析得出的误差, 考虑计算得出的 4 阶导数向量, 其长度比原始对照向量 f_4 短, 所以两个向量取同样多点进行比较, 就可以得出数值方法的相对误差最大值为 3.5×10^{-4} , 亦即 0.035%。由此可见, 这里的数值方法还是很精确的。

```
>> norm((y4-f4(4:60))./f4(4:60))
```

3.6.3 二元函数的梯度计算

如果给定二元函数的函数值矩阵 z , 其中 z 为网格数据, 则可以由 `gradient()` 函数求

取二元函数的梯度。该函数的调用格式为 $[f_x, f_y] = \text{gradient}(z)$ 。其实,这样计算出来的 f_x 与 f_y 不是真正的梯度,这里尚未考虑 x, y 坐标的情况。如果得到的 z 矩阵是建立在等间距的形式生成网格基础上的,则实际的梯度值可以由 $f_x = f_x/\Delta x$ 和 $f_y = f_y/\Delta y$ 求出,其中, Δx 和 Δy 分别为 x, y 生成网格的步距。

例 3-49 考虑例 3-15 中的问题,假设已经得出网格数据,试用数值方法由该数据解出梯度值。

解 现在重新生成数据,则可以由这些数据直接计算出该函数的梯度,而无需再从原函数直接计算梯度值。由下面的语句还能绘制出带有等值线的引力线图,和图 3-3(b) 中的图形完全一致。

```
>> [x,y]=meshgrid(-3:.2:3,-2:.2:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
    [fx,fy]=gradient(z); fx=fx/0.2; fy=fy/0.2;
    contour(x,y,z,30); hold on; quiver(x,y,fx,fy)
```

下面的语句将绘制出误差的曲面,如图 3-10 所示。可见大部分区域内误差还是较小的,但在某些小的区域内误差较大,这说明原来网格的间距较大,使得简单的梯度函数难以精确求解。

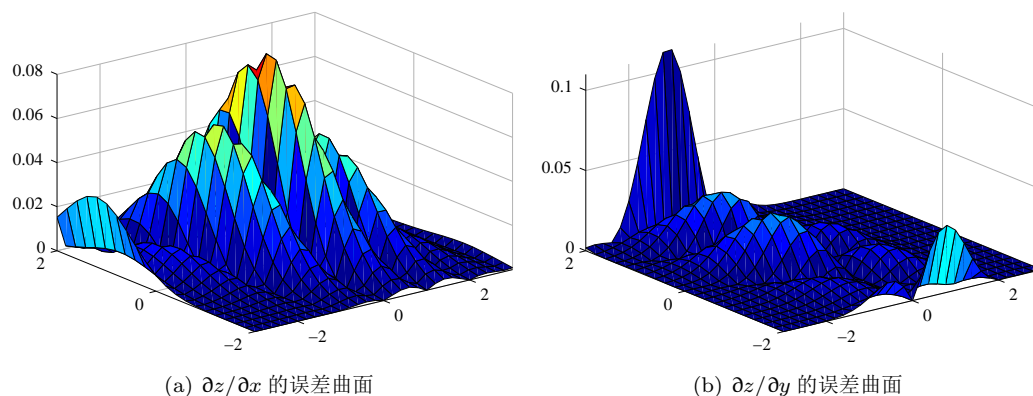


图 3-10 二元函数数值梯度的误差曲面

```
>> zx=-exp(-x.^2-y.^2-x.*y).*(-2*x+2+2*x.^3+x.^2.*y-4*x.^2-2*x.*y);
    zy=-x.*(x-2).*(2*y+x).*exp(-x.^2-y.^2-x.*y);
    surf(x,y,abs(fx-zx)); axis([-3 3 -2 2 0,0.08])
    figure; surf(x,y,abs(fy-zy)); axis([-3 3 -2 2 0,0.11])
```

如果将网格加密一倍,则可以由下面的语句计算数值梯度,得出的结果和其与理论值之间的误差也可以绘制出来,如图 3-11 所示,可见这时误差显著减小。

```
>> [x,y]=meshgrid(-3:.1:3,-2:.1:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
    [fx,fy]=gradient(z); fx=fx/0.1; fy=fy/0.1;
    zx=-exp(-x.^2-y.^2-x.*y).*(-2*x+2+2*x.^3+x.^2.*y-4*x.^2-2*x.*y);
    zy=-x.*(x-2).*(2*y+x).*exp(-x.^2-y.^2-x.*y);
    surf(x,y,abs(fx-zx)); axis([-3 3 -2 2 0,0.02])
    figure; surf(x,y,abs(fy-zy)); axis([-3 3 -2 2 0,0.06])
```

3.7 数值积分问题

数值积分问题是传统数值分析课程中的重要内容。本节将分几种情况介绍数值积分问

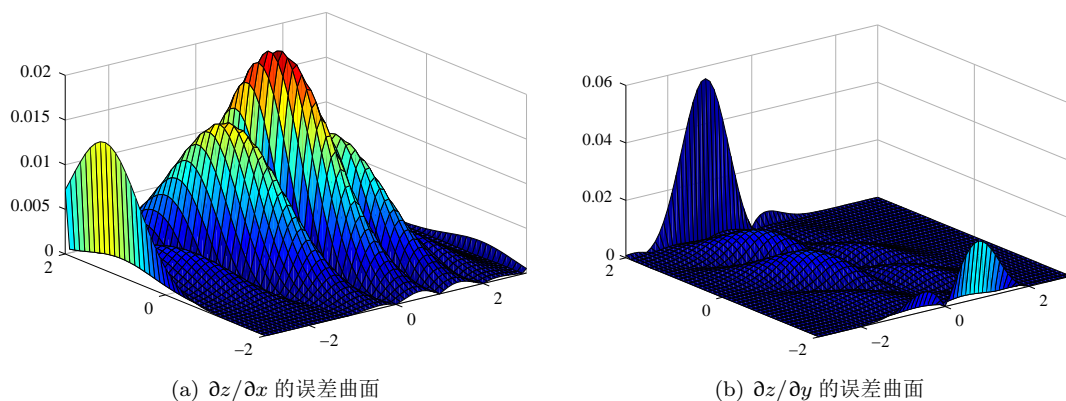


图 3-11 网格加密后二元函数数值梯度的误差曲面

题的求解方法。首先,如果被积函数的数学表达式未知,则需要由实测数据通过梯形算法求出积分的近似值;如果被积函数已知,则将分别介绍一元函数积分、一元函数广义积分、二重积分以及多重积分问题。采用前面介绍的解析解方法和 `vpa()` 函数,可以得出任意一元函数的积分值,所以若安装了符号运算工具箱,则没有太大必要采用本节介绍的纯数值方法;对于重积分问题来说,如果内重积分是解析不可积的,则解析解方法是不能得出积分值的,必须采用数值积分方法。

3.7.1 由给定数据进行梯形求积

一元函数定积分的数学表示为

$$I = \int_a^b f(x)dx \quad (3-7-1)$$

在被积函数 $f(x)$ 理论上不可积时,即使有强大的计算机数学语言帮忙,也不能够求出该积分的解析解,所以往往要采用数值方法来求解。求解定积分的数值方法是多种多样的,如简单的梯形法、Simpson 法、Romberg 法等算法都是数值分析课程中经常介绍的方法。它们的基本思想都是将整个积分空间 $[a, b]$ 分割成若干个子空间 $[x_i, x_{i+1}]$, $i = 1, 2, \dots, N$, 其中 $x_1 = a$, $x_{N+1} = b$ 。这样整个积分问题就分解为下面的求和形式

$$\int_a^b f(x)dx = \sum_{i=1}^N \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{i=1}^N \Delta f_i \quad (3-7-2)$$

而在每一个小的子空间上都可以近似地求解出来,当然最简单的求每一个小的子空间的积分方法是采用梯形近似的方法。梯形方法还可以应用于已知数据样本点的数值积分问题求解。假设在实验中测得一组数据 $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_N, y_N)$, 且 x_i 为严格单调递增的数值,直接求取这些点对应曲线的数值积分最直观的方法就是用梯形方法,用直线将这些点连接起来,则积分可以近似为该折线与 x -轴之间围成的面积。

假设已经建立起向量 $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$, $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$, 则由 MATLAB 的 `trapz()` 函数可以直接用梯形法求解积分问题,该函数调用格式为 `S = trapz(x, y)`, 其

中, x 可以为行向量或列向量, y 的行数应该等于 x 向量的元素数。如果 y 由多列矩阵给出, 则用该函数可以得出若干个函数的积分值。

例 3-50 试用梯形法求出 $x \in (0, \pi)$ 区间内, 函数 $\sin x, \cos x, \sin(x/2)$ 的定积分值。

解 生成区间内横坐标向量, 用上述的算法可以求出各个函数的数值积分值为 1.9982, 0.0000, 1.9995, 而这些理论值分别为 2, 0, 2。

```
>> x1=[0:pi/30:pi]'; y=[sin(x1) cos(x1) sin(x1/2)]; S=trapz(x1,y)
```

由于选择的步距较大, 为 $h = \pi/30 \approx 0.1$, 故得出的结果有较大的误差。在 8.1.2 节中将积分问题与样条插值技术相结合, 给出一个能精确计算数值积分的 MATLAB 函数, 并演示其在更大步距下的有效性和精度。

例 3-51 请用定步长方法求解积分 $\int_0^{3\pi/2} \cos 15x \, dx$ 。

解 求解问题之前, 首先用下面的 MATLAB 语句绘制出被积函数的曲线, 如图 3-12 所示。可见, 在求解区域内被积函数有很强的振荡。

```
>> x=linspace(0,3*pi/2,30); y=cos(15*x); plot(x,y)
```

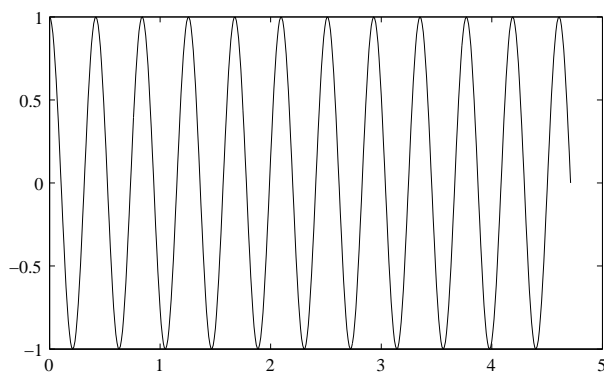


图 3-12 被积函数 $f(x) = \cos 15x$ 的曲线

对不同的步距 $h = 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001$, 可以用下面的语句求出采用不同步距的积分近似结果。为了有更好的表示, 特将结果用表 3-1 列出。

```
>> syms x, A=int(cos(15*x),0,3*pi/2); h0=10.^[-1:-1:-6]; v=[];
for h=h0, tic
    x=[0:h:3*pi/2, 3*pi/2]; y=cos(15*x); I=trapz(x,y); v=[v; h,I,1/15-I];
toc, end
```

表 3-1 步距选择与计算结果

步长	得出积分值	误差	步长	得出积分值	误差
0.1	0.05389175150075948	0.0127749152	0.0001	0.06666665416666881	$1.24999978 \times 10^{-8}$
0.01	0.0665416954658383	0.0001249712	10^{-5}	0.06666666654166685	$1.24999816 \times 10^{-10}$
0.001	0.06666541668003727	1.2499866×10^{-6}	10^{-6}	0.06666666666541621	$1.25045807 \times 10^{-12}$

可见,随着步距 h 的减小,计算精度逐渐增加。例如,当 $h = 10^{-6}$ 时可以保留小数点后 11 位精确数字,但这时求解的时间也将成倍增加,达到 0.25 s —— 此函数执行效率较早期版本有明显的改善。如果想进一步增加计算精度,还得再减小步长,这样内存将耗尽,程序不能继续执行下去。

3.7.2 单变量数值积分问题求解

单变量函数的数值积分还可以采用一般数值分析中介绍的其他算法进行求解。例如,可以采用下面给出的 Simpson 方法求解出 $[x_i, x_{i+1}]$ 上的积分 Δf_i 的近似值为

$$\Delta f_i \approx \frac{h_i}{12} \left[f(x_i) + 4f\left(x_i + \frac{h_i}{4}\right) + 2f\left(x_i + \frac{h_i}{2}\right) + 4f\left(x_i + \frac{3h_i}{4}\right) + f(x_i + h_i) \right] \quad (3-7-3)$$

式中, $h_i = x_{i+1} - x_i$ 。

MATLAB 8.0 版开始引入新的自适应变步长数值积分求取函数 `integral()`, 其调用格式为 `I = integral(f,a,b,属性设置对)`, 其中, f 用于描述被积函数, 它可以是一个 `Fun.m` 函数文件名(由 `@Fun` 或 `'Fun'` 给出), 该函数的一般格式为 $y = \text{Fun}(x)$, 还可以用匿名函数或 `inline()` 函数; a 、 b 分别为定积分的上限和下限。该函数还允许给出“属性设置对”来设置积分控制选项, 如 `RelTol` 选项和相对误差限的值来指定计算精度。这样的属性还包括绝对误差限 `AbsTol`、向量积分控制标志 `ArrayValued` —— 允许一次计算若干个函数的积分。下面将通过例子演示数值积分的求解。

早期 MATLAB 版本可以使用底层函数 `quad()`、`quadl()`、`quadgk()` 和 `quadv()` 计算数值积分的值, 其调用格式与 `integral()` 类似, 使用早期版本的读者只需将这里的 `integral()` 函数替换成相应的底层函数即可, 必要时可以由 `doc` 命令获得函数的帮助。

例 3-52 考虑不可积数学函数 $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$, 试用数值方法来求解该积分。

解 在求取数值解之前, 需要首先描述一下被积函数。描述被积函数有三种方法, 其一是建立一个 MATLAB 函数并将其存成文件, 其内容为

```
function y=c3ffun(x), y=2/sqrt(pi)*exp(-x.^2);
```

这样, 可以将上述内容存入一个 `c3ffun.m` 文件。由于自变量每次读入的可以是一组 x 的值, 所以函数内部应该使用点运算来计算每个自变量取值处的函数值 y 。

第二种方法是建立匿名函数, 其格式为

```
>> f=@(x)2/sqrt(pi)*exp(-x.^2);
```

这种方法的特点是可以动态地描述需要求解的问题, 而无需建立一个单独的文件, 所以这样的方法更适合于简单问题的直接应用。

类似于匿名函数的方法, 第三种方式是用 `inline()` 函数定义被积函数, 可以给出下面的语句

```
>> f=inline('2/sqrt(pi)*exp(-x.^2)','x');
```

同样, 这种方法也无需建立一个单独的 MATLAB 文件。相比之下, `inline()` 函数的第一个输入变量为被积函数本身, 和 MATLAB 函数描述格式完全相同, 第 2 个输入变量为自变量, 当然还可以带有多个自变量。

定义了被积函数, 可以调用 `integral()` 函数直接求解出定积分值为 0.966105146475311。


```
>> f=@(x)2/sqrt(pi)*exp(-x.^2); y=integral(f,0,1.5)
```

用两种方法得出的结果完全相同。其实,对这样简单的一元数值积分问题来说,用符号运算工具箱可以求解出更精确的解 $I_0 = 0.966105146475311$ 。可见,该函数在双精度意义下还是相当精确的。

```
>> syms x, I0=vpa(int(2/sqrt(pi)*exp(-x^2),0,1.5))
```

虽然前面介绍的三种方法均可以用于描述被积函数,但它们各有特点。M 函数的方法可以描述带有中间变量的问题,而后两种方法则不能。在后面将涉及到的返回多个变量的问题也不适合采用匿名函数与 `inline()` 函数。从计算速度看,使用匿名函数的速度要明显快于另两种方法。匿名函数和 `inline()` 函数在功能上是重叠的,只是匿名函数在 MATLAB 7.0 版才开始引入,所以很多早期的程序仍在使用另两种方法。本书将尽量采用匿名函数,如需返回多个变量或涉及中间变量时将采用 M 函数。

例 3-53 试求解下面分段函数的积分问题

$$I = \int_0^4 f(x)dx, \text{ 其中 } f(x) = \begin{cases} e^{x^2}, & 0 \leq x \leq 2 \\ \frac{80}{4 - \sin(16\pi x)}, & 2 < x \leq 4 \end{cases}$$

解 用曲线绘制函数不难绘制出分段函数,这里为减小视觉上的误差,在端点和间断点处采用了特殊处理,故可以得出如图 3-13 所示的填充图形。可见,在 $x = 2$ 点处有跳跃。

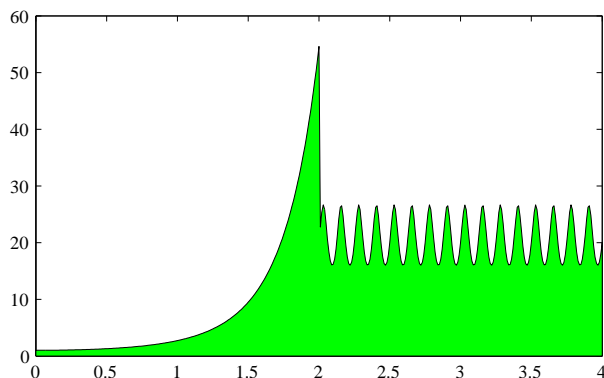


图 3-13 被积区域填充示意图

```
>> x=[0:0.01:2, 2+eps:0.01:4,4];
y=exp(x.^2).*(x<=2)+80./(4-sin(16*pi*x)).*(x>2);
y(end)=0; x=[eps, x]; y=[0,y]; fill(x,y,'g')
```

利用关系表达式可以描述出被积函数,调用积分函数 `integral()` 就可以求解出原始定积分,得出 $I_1 = 57.764450125048512$ 。

```
>> f=@(x)exp(x.^2).*(x<=2)+80./(4-sin(16*pi*x)).*(x>2); I=integral(f,0,4)
```

不过从得出的结果看,二者有很大的差异。其实,可以将原来的积分问题转换成 $\int_0^2 + \int_2^4$ 的问题,用积分问题解析求解函数 `int()` 可以得出原始问题的精确解为 $I = 57.76445012505301033315$ 。

```
>> syms x; I0=vpa(int(exp(x^2),0,2)+int(80/(4-sin(16*pi*x)),2,4))
```

此问题的解析解是已知的,当然可以和解析解对比,看得出的解精度如何,而实际应用中解析解是未知的,如何检验得出的解是否正确呢?考虑设置一下更严格的相对误差限 RelTol,看看能否得出一致的结果,如果不能则再设置更小的误差限。例如,本问题选择误差限 10^{-20} 即可得出精确的结果 $I_2 = 57.764450125053017$ 。

```
>> I2=integral(f,0,4,'RelTol',1e-20)
```

利用符号变量的分段函数表示方法,可以给出下面语句计算积分,结果和前面的完全一致。

```
>> f=piecewise('x<=2','exp(x^2)','x>2','80/(4-sin(16*pi*x))');
syms x; I=vpa(int(f,x,0,4))
```

例 3-54 试用 integral() 函数求解例 3-51 中的定积分问题, $I = \int_0^{3\pi/2} \cos 15x \, dx$ 。

解 从例 3-51 中演示的定步长方法看,只有步长选得极小,才能准确得出 11 位有效数字,且耗时较长。其实,用变步长数值积分函数可以轻而易举地求出该定积分问题的解为 $S = 0.066666666666667$,所需时间只需 0.002s,使用的时间也大大地减少了。

```
>> f=@(x)cos(15*x); tic, S=integral(f,0,3*pi/2,'RelTol',1e-20), toc
```

所以,由此可以得出结论:求解变化不均匀的函数的积分不宜采用传统数值分析类课程介绍的定步长积分算法,因为用该算法精度难以保证;而若要使用小步长,则计算量将极大,且仍然无法保证计算精度。采用变步长算法可以很容易地得出原问题的解。

例 3-55 试求解复数积分问题 $\int_2^{6-j5} e^{-x^2-jx} \sin(7+j2)x \, dx$ 。

解 复函数的积分问题可以由下面的语句直接求解,得出的积分值为 $I = -0.9245 + j25.792$ 。采用理论值求解方法可以验证,前面得出的数值积分是准确的。

```
>> f=@(x)exp(-x.^2-1i*x).*sin((7+2i)*x); I=integral(f,2,6-5i,'RelTol',1e-20)
syms x; i=sqrt(-1); F=exp(-x^2-i*x)*sin((7+2i)*x); I0=vpa(int(F,2,6-5i))
```

例 3-56 重新考虑例 3-51 中的振荡函数积分问题,若积分区间为 $[0, 100]$,试求出其数值积分。

解 由于积分区间过大且被积函数一直在震荡,所以早期版本的 quadl() 函数将失效,quadgk() 函数可以使用。利用新版本的数值积分函数可以得出积分的值为 $I_1 = -0.066260130460299$,采用解析解方法验证了该积分的精确值为 $I = \sin(1500)/15 \approx -0.066260130460443564274$ 。

```
>> f=@(x)cos(15*x); I1=integral(f,0,100,'RelTol',1e-20)
syms x; I=int(cos(15*x),x,0,100), vpa(I)
```

3.7.3 广义数值积分问题求解

前面介绍的 integral() 可以直接用于广义积分的求取,但该函数只能在 MATLAB 8.0 及后续版本下运行。早期版本中可以采用 quadgk() 函数求无穷积分,该函数采用了 Gauss-Kronrod 算法^[3]。函数的调用格式与前面介绍的完全一致,直接在积分限位置给出 -inf 或 inf 即可。下面通过例子演示该函数的应用。

例 3-57 试求出无穷积分 $\int_0^{\infty} e^{-x^2} \, dx$ 。

解 由数值积分函数 `integral()` 可以直接得出所需的无穷积分为 $I = 0.886226925452758$, 与理论值 $I_1 = \sqrt{\pi}/2 \approx 0.88622692545275801365$ 相当接近, 误差达到 10^{-16} 级别。

```
>> f=@(x)exp(-x.^2); I=integral(f,0,inf,'RelTol',1e-20)
syms x; I1=int(exp(-x^2),0,inf), vpa(I1)
```

例 3-58 已知 $I(\alpha) = \int_0^{\infty} e^{-\alpha x^2} \sin(\alpha^2 x) dx$, 试绘制出 $I(\alpha)$ 与 α 的关系曲线, 其中 $\alpha \in (0, 4)$ 。

解 前面介绍的积分都是某个单个函数的定积分, 而这里需要求解的是对一系列 α 值的定积分问题, 使用应该采用向量函数积分的方法。下面的语句可以直接求取原问题的积分, 得出的函数曲线如图 3-14 所示。早期版本求解此问题需要采用循环结构。

```
>> a=0:0.1:4; f=@(x)exp(-a*x.^2).*sin(a.^2*x);
I=integral(f,0,inf,'RelTol',1e-20,'ArrayValued',true); plot(a,I)
```

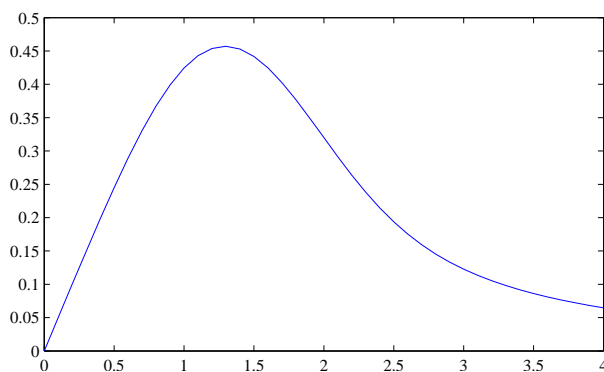


图 3-14 积分 $I(\alpha)$ 与 α 的关系曲线

3.7.4 积分函数的数值求解

本节前面介绍的内容都是求出 (a, b) 区间的定积分的方法, 如何绘制出函数的积分函数曲线是这里要探讨的问题。积分函数在 a 点的值为 0 (即 (a, a) 区间的定积分为 0), 所以可以编写如下的函数

```
function [x,f1]=intfunc(f,a,b,n)
if nargin<=3, n=100; end; x=linspace(a,b,n); f1=0; f0=0;
for i=2:n, f2=f0+integral(f,x(i-1),x(i)); f1=[f1, f2]; f0=f2; end
```

该函数的调用格式为 `[x, f1] = intfunc(f, a, b, n)`, n 的默认值为 100。

例 3-59 试绘制出例 3-53 中分段函数的积分曲线。

解 由于分段函数中 e^{x^2} 是不可积的函数, 所以不能用解析解方法绘制出其积分函数曲线, 求解这样的问题只能用数值方法。先用匿名函数定义出被积函数, 则可以调用 `intfunc()` 函数直接求解原问题, 得出的积分函数曲线如图 3-15 所示。可见, 例 3-53 得出的定积分只是其右侧端点的函数值。

```
>> f=@(x)exp(x.^2).*(x<=2)+80./(4-sin(16*pi*x)).*(x>2);
[x1,f1]=intfunc(f,0,4,100); plot(x1,f1,x1(end),f1(end),'o'), f1(end)
```

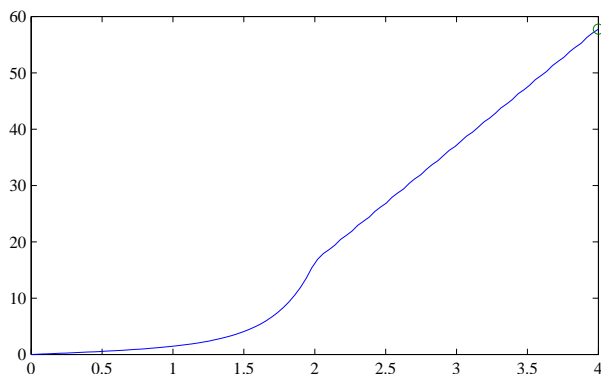


图 3-15 给出函数的积分曲线

3.7.5 双重积分问题的数值解

考虑下面的双重定积分问题

$$I = \int_{y_m}^{y_M} \int_{x_m}^{x_M} f(x, y) dx dy \quad (3-7-4)$$

使用 MATLAB 提供的 `integral2()` 函数就可以直接求出上述双重定积分的数值解。该函数的调用格式为 `I = integral2(f, x_m, x_M, y_m, y_M, 属性参数对)`，其中，“属性参数对”的用法与 `integral()` 函数完全一致，早期版本应该使用 `dblquad()`。

例 3-60 试求出双重定积分 $J = \int_{-1}^1 \int_{-2}^2 e^{-x^2/2} \sin(x^2 + y) dx dy$ 。

解 用匿名函数表示被积函数，并选择 x 和 y 的积分范围分别为 $[-2, 2]$ 、 $[-1, 1]$ ，这样就可以通过下面的 MATLAB 语句求出被积函数的双重定积分值为 1.574498159218786。

```
>> f=@(x,y)exp(-x.^2/2).*sin(x.^2+y); J=integral2(f,-2,2,-1,1,'RelTol',1e-20)
```

仿照图 3-15 的思路，可以编写出等间距矩形子区域的积分函数数值解的 MATLAB 函数，并绘制出积分函数曲面，尽管被积函数可能不可积。该函数的调用格式为

```
[x,y,f1] = intfunc2(f,x_m,x_M,y_m,y_M,n,m)
```

其中， f 为匿名函数或 M-函数， (x_m, x_M) 和 (y_m, y_M) 为积分的矩形区域， n, m 为 x, y 轴的分段数，默认值为 50。返回变量 $f_1(\text{end}, \text{end})$ 即为定积分的值。

```
function [xv,yv,f1]=intfunc2(f,xm,xM,ym,yM,varargin)
[n,m]=default_vals(50,50,varargin{:}); f1=[]; fy0=0;
xv=linspace(xm,xM,n); yv=linspace(ym,yM,m); d=yv(2)-yv(1);
for i=1:m,
    y0=yv(i); fnew=@(x)f(x,y0)*d; [x1,y1]=intfunc(fnew,xm,xM,n);
    fy2=fy0+y1; f1=[f1; fy2]; fy0=fy2;
end
```

例 3-61 求解例 3-60 被积函数在矩形区域内的积分曲面。

解 下面语句可以先用匿名函数定义被积函数,这样就可以求出二元函数的积分曲面,如图 3-16 所示,曲面左上角的值为例 3-60 求出的近似定积分值 $I = 1.5949$ 。

```
>> f=@(x,y)exp(-x.^2/2).*sin(x.^2+y);
[x,y,z]=intfunc2(f,-2,2,-1,1); surf(x,y,z), I=z(end,end)
```

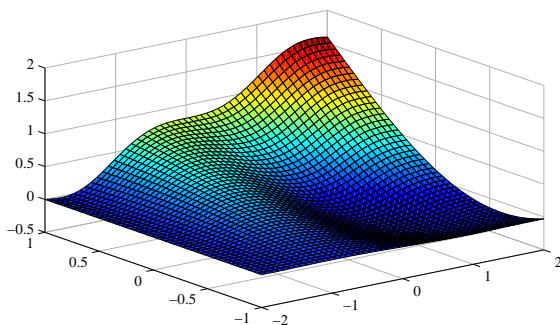


图 3-16 给出二元函数的积分曲面

遗憾的是,在 MATLAB 中并没有提供求解更一般的双重积分问题的函数

$$I = \int_{y_m}^{y_M} \int_{x_m(y)}^{x_M(y)} f(x, y) dx dy \quad (3-7-5)$$

好在美国学者 Howard Wilson 与 Bryce Gardner 开发了数值积分工具箱 (Numerical Integration Toolbox, NIT) [4] 可以解决这样的问题。该工具箱可以在 MathWorks 公司的网站上免费下载,该工具箱中的函数 quad2dggen() 可以直接求解式 (3-7-5) 的双重积分问题。该函数的调用格式为 $J = \text{quad2dggen}(f, f_{xm}, f_{xM}, y_m, y_M, \epsilon)$, 其中, ϵ 为误差限。误差限越小,计算应该越精确,但计算量也将增大,此函数默认的误差限为 $\epsilon = 10^{-3}$ 。该函数还涉及 3 个 MATLAB 函数,即被积函数和上下限函数。

对先对 y 再对 x 积分的问题,可以令 $\hat{x} = y$, $\hat{y} = x$, 则式 (3-7-5) 可以等效地变换为

$$I = \int_{\hat{x}_m}^{\hat{x}_M} \int_{\hat{y}_m(\hat{x})}^{\hat{y}_M(\hat{x})} f(\hat{y}, \hat{x}) d\hat{y} d\hat{x} \quad (3-7-6)$$

这样,最简单的方法就是互换原函数 $f(x, y)$ 中变量的次序,而不必修改其他的部分,将被积函数定义成 $f = @(y, x)$ 即可。下面将通过一个具体例子来演示双重积分的运算。

例 3-62 试求出双重定积分 $J = \int_{-1/2}^1 \int_{-\sqrt{1-x^2/2}}^{\sqrt{1-x^2/2}} e^{-x^2/2} \sin(x^2 + y) dy dx$ 。

解 这里的例子是先 y 后 x , 可以先构造出 $y_m(x)$ 和 $y_M(x)$ 上下界函数,再调用相应的函数求解原问题,其结果为 0.411929546176295。注意,这里应该交换被积函数积分变量次序。

```
>> fh=@(x)sqrt(1-x.^2/2); fl=@(x)-sqrt(1-x.^2/2);
f=@(y,x)exp(-x.^2/2).*sin(x.^2+y); y=quad2dggen(f,fl,fh,-1/2,1,1e-20)
```

下面的解析解方法不能得出原问题的解析解,但可以得出其数值解为 0.41192954617629512。

```
>> syms x y % 现在考虑解析解方法
i1=int(exp(-x^2/2)*sin(x^2+y),y,-sqrt(1-x^2/2),sqrt(1-x^2/2));
int(i1,x,-1/2,1), vpa(ans) % 求取解析解时得出警告信息,但数值解可得出
```

例 3-63 前面由 quad2dggen() 函数得出数值解也是很精确的。本问题可以求出高精度数值解的主要原因是被积函数的内积分是可积的。如果不可积,则无从得出整个双重积分,这样只能采用数值解。例如,若积分问题变成

$$J = \int_{-1}^1 \int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} e^{-x^2/2} \sin(x^2 + y) dx dy$$

则对 x 是不可积的,故调用解析解方法不会得出任何结果,而数值解求解不受此影响。对这里给出的问题来说,由于积分顺序是先 x 后 y ,所以无需修改被积函数本身,这时可以由下面语句直接得出其数值解为 0.536860382697953。

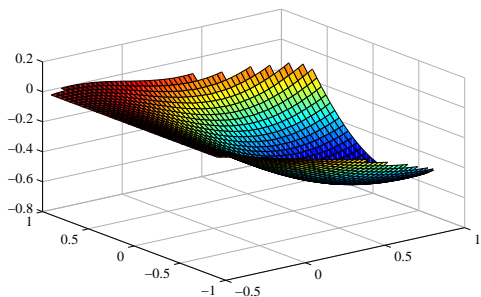
```
>> fh=@(y)sqrt(1-y.^2); fl=@(y)-sqrt(1-y.^2);
f=@(x,y)exp(-x.^2/2).*sin(x.^2+y); I=quad2dggen(f,fl,fh,-1,1,1e-20)
```

积分函数曲面仍然可以利用 intfunc2() 函数求出,然后将积分区域之外部分的函数值设置成 NaN 即可。

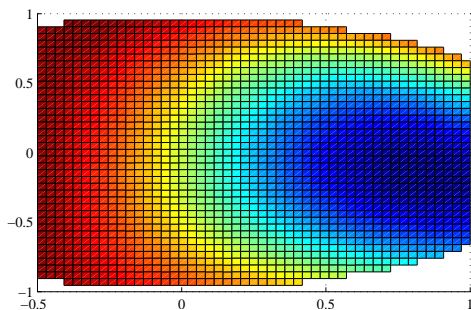
例 3-64 试求取例 3-62 中的积分曲面。

解 可以由前面介绍的方法先求出矩形区域的积分函数数值解,然后对得到的解逐列扫描,将积分区域之外的点函数值设置为 NaN,这样可以通过下面语句绘制积分函数的曲面,如图 3-17(a) 所示。使用 view(0,90) 则可以得出积分区域的表示,如图 3-17(b) 所示。

```
>> f=@(x,y)exp(-x.^2/2).*sin(x.^2+y); fh=@(x)sqrt(1-x.^2/2);
fl=@(x)-sqrt(1-x.^2/2); [x,y,z]=intfunc2(f,-1/2,1,-1.2,1.2);
for i=1:length(x), x0=x(i); xx=sort([fl(x0), fh(x0)]);
ii=find(y>xx(2) | y<xx(1)); z(ii,i)=NaN;
end
surf(x,y,z), figure, surf(x,y,z), view(0,90)
```



(a) 二元函数积分曲面



(b) 曲面俯视图

图 3-17 二元函数的积分曲面

3.7.6 三重定积分的数值求解

长方体区域的三重定积分

$$I = \int_{x_m}^{x_M} \int_{y_m}^{y_M} \int_{z_m}^{z_M} f(x, y, z) dz dy dx \quad (3-7-7)$$

可以由 MATLAB 提供的 `integral3()` 函数得出。该函数调用格式为

`I = integral3(f, x_m, x_M, y_m, y_M, z_m, z_M, 属性参数对)`

其中, f 描述三元被积函数, 同样可以用 M-函数、匿名函数或 `inline()` 函数定义。“属性参数对”的内容与 `integral()` 函数完全一致, 早期版本可用 `triplequad()` 函数求解。

例 3-65 用数值方法求解例 3-27 中的三重定积分问题 $\int_0^2 \int_0^\pi \int_0^\pi 4xz e^{-x^2 y - z^2} dz dy dx$ 。

解 用匿名函数描述被积函数, 该被积函数有 x, y, z 三个自变量, 通过下面的语句立即可以求出三重定积分值, 其近似解为 3.108079402085465。

```
>> f=@(x,y,z)4*x.*z.*exp(-x.*x.*y-z.*z);
tic, I=integral3(f,0,2,0,pi,0,pi,'RelTol',1e-20), toc
```

3.7.7 多重积分数值求解

NIT 工具箱还可以解决多重超长方体边界的定积分问题, 例如, 使用 `quadndg()` 函数, 但对一般积分区域来说则没有现成的求解函数。积分区域为超长方体的多重积分问题可以表示为

$$I = \int_{x_{1m}}^{x_{1M}} \int_{x_{2m}}^{x_{2M}} \cdots \int_{x_{pm}}^{x_{pM}} f(x_1, x_2, \cdots, x_p) dx_p \cdots dx_2 dx_1 \quad (3-7-8)$$

该问题的求解语句为 `I = quadndg(f, [x1m, x2m, ..., xpm], [x1M, x2M, ..., xpM], ε)`, 其中, f 为描述被积函数的 M-函数, ϵ 为误差容限, 可以忽略。

例 3-66 试用多重积分的求解函数重新求例 3-65 中的三重积分问题 $\int_0^2 \int_0^\pi \int_0^\pi 4xz e^{-x^2 y - z^2} dz dy dx$ 。

解 令 $x_1 = x, x_2 = y, x_3 = z$, 则原问题的被积函数可以重新改写成 $f(\mathbf{x}) = 4x_1 x_3 e^{-x_1^2 x_2 - x_3^2}$, 可以用下面的匿名函数直接描述被积函数, 然后调用求解函数求解原积分问题, 得出原问题的解为 $I = 3.108079402085409$, 该结果与例 3-65 一致, 但由于 `quadndg()` 算法效率高于 `integral3()`, 所以求解的时间大约为例 3-65 的 1/10。

```
>> f=@(x)4*x(1)*x(3)*exp(-x(1)^2*x(2)-x(3)^2);
tic, I=quadndg(f,[0 0 0],[2,pi,pi]), toc
```

例 3-67 用数值和解析解方法求解下面的 5 重定积分问题。

$$I = \int_0^5 \int_0^4 \int_0^1 \int_0^2 \int_0^3 \sqrt[3]{v} \sqrt{w} x^2 y^3 z \, dz dy dx dw dv$$

解 对这样的特殊问题来说, 其解析解是可以求出的, 积分值为 $120\sqrt[3]{5}$ 。

```
>> syms x y z w v; F=v^(1/3)*sqrt(w)*x^2*y^3*z;
I=int(int(int(int(int(F,z,0,3),y,0,2),x,0,1),w,0,4),v,0,5)
```

事实上,大部分高维积分问题解析解是不存在的,所以应该采用数值方法去求解。令 $x_1 = v$, $x_2 = w$, $x_3 = x$, $x_4 = y$, $x_5 = z$, 则被积函数可以改写成

$$f(\mathbf{x}) = \sqrt[3]{x_1} \sqrt{x_2} x_3^2 x_4^3 x_5$$

所以此积分问题的被积函数可以由匿名函数表示,这样可以给出下面的求解语句,得出 $I = 205.2205 \approx 120\sqrt[3]{5}$ 。由于这里采用的算法是非向量型的,所以运算速度较向量型算法慢很多,本例所需时间大约 20s。

```
>> f=@(x)(x(1))^(1/3)*sqrt(x(2))*x(3)^2*x(4)^3*x(5);
tic, I=quadndg(f,[0 0 0 0 0],[5,4,1,2,3]), toc
```

例 3-68 用数值方法求解下面的 5 重定积分问题。

$$I = \int_0^5 \int_0^4 \int_0^1 \int_0^2 \int_0^3 \left(e^{-\sqrt[3]{v}} \sin \sqrt{w} + e^{-x^2 y^3 z} \right) dz dy dx dw dv$$

解 这里给出的例子是不能解析求解的,必须借助数值方法求解原始问题。仍旧令 $x_1 = v$, $x_2 = w$, $x_3 = x$, $x_4 = y$, $x_5 = z$, 则被积函数可以改写成

$$f(\mathbf{x}) = e^{-\sqrt[3]{x_1}} \sin \sqrt{x_2} + e^{-x_3^2 x_4^3 x_5}$$

此积分问题的被积函数可以由匿名函数表示,这样可以给出下面的求解语句,得出多重积分的值为 $I = 113.60574122$ 。尽管被积函数比上例的复杂很多,但两者的计算时间相差无几。

```
>> f=@(x)exp(-(x(1))^(1/3))*sin(sqrt(x(2)))+exp(-x(3)^2*x(4)^3*x(5));
tic, I=quadndg(f,[0 0 0 0 0],[5,4,1,2,3]), toc
```

3.8 习 题

1. 试求出如下极限:

$$(1) \lim_{x \rightarrow \infty} (3^x + 9^x)^{\frac{1}{x}}, \quad (2) \lim_{x \rightarrow \infty} \frac{(x+2)^{x+2}(x+3)^{x+3}}{(x+5)^{2x+5}}, \quad (3) \lim_{x \rightarrow a} \left(\frac{\tan x}{\tan a} \right)^{\cot(x-a)},$$

$$(4) \lim_{x \rightarrow 0} \left[\frac{1}{\ln(x + \sqrt{1+x^2})} - \frac{1}{\ln(1+x)} \right],$$

$$(5) \lim_{x \rightarrow \infty} \left[\sqrt[3]{x^3 + x^2 + x + 1} - \sqrt{x^2 + x + 1} \frac{\ln(e^x + x)}{x} \right].$$

2. 试求出下面的累极限 $\lim_{x \rightarrow a} \left[\lim_{y \rightarrow b} f(x, y) \right]$ 和 $\lim_{y \rightarrow b} \left[\lim_{x \rightarrow a} f(x, y) \right]$:

$$(1) f(x, y) = \sin \frac{\pi x}{2x+y}, a = \infty, b = \infty, \quad (2) f(x, y) = \frac{1}{xy} \tan \frac{xy}{1+xy}, a = 0, b = \infty.$$

3. 试求下面的双重极限:

$$(1) \lim_{\substack{x \rightarrow -1 \\ y \rightarrow 2}} \frac{x^2 y + x y^3}{(x+y)^3}, \quad (2) \lim_{\substack{x \rightarrow 0 \\ y \rightarrow 0}} \frac{xy}{\sqrt{xy+1}-1}, \quad (3) \lim_{\substack{x \rightarrow 0 \\ y \rightarrow 0}} \frac{1 - \cos(x^2 + y^2)}{(x^2 + y^2) e^{x^2 + y^2}}.$$

4. 求出下面函数的导数:

$$(1) y(x) = \sqrt{x \sin x \sqrt{1 - e^x}}, \quad (2) y = \frac{1 - \sqrt{\cos ax}}{x(1 - \cos \sqrt{ax})},$$

$$(3) \operatorname{atan} \frac{y}{x} = \ln(x^2 + y^2), \quad (4) y(x) = -\frac{1}{na} \ln \frac{x^n + a}{x^n}, \quad n > 0.$$

5. 试求出 $y(t) = \sqrt{\frac{(x-1)(x-2)}{(x-3)(x-4)}}$ 函数的 4 阶导数。

6. 在高等数学中, 求解分子和分母均同时为 0 或 ∞ 的分式极限时可使用 L'Hôpital 法则, 即对分子分母分别求导数, 再由比值得出。试用该法则求 $\lim_{x \rightarrow 0} \frac{\ln(1+x) \ln(1-x) - \ln(1-x^2)}{x^4}$, 并和直接求出的极限结果相比较。

7. 已知参数方程 $\begin{cases} x = \ln \cos t \\ y = \cos t - t \sin t \end{cases}$, 试求出 $\frac{dy}{dx}$ 和 $\frac{d^2y}{dx^2} \Big|_{t=\pi/3}$ 。

8. 假设 $u = \arccos \sqrt{\frac{x}{y}}$, 试验证 $\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial^2 u}{\partial y \partial x}$ 。

9. 设 $\begin{cases} xu + yv = 0 \\ yu + xv = 1 \end{cases}$, 试求解 $\frac{\partial^2 u}{\partial x \partial y}$ 。

10. 假设 $f(x, y) = \int_0^{xy} e^{-t^2} dt$, 试求 $\frac{x}{y} \frac{\partial^2 f}{\partial x^2} - 2 \frac{\partial^2 f}{\partial x \partial y} + \frac{\partial^2 f}{\partial y^2}$ 。

11. 试由下面参数方程求出 dy/dx , d^2y/dx^2 和 d^3y/dx^3 :

$$(1) x = e^{2t} \cos^2 t, y = e^{2t} \sin^2 t, \quad (2) x = \arcsin \frac{t}{\sqrt{1+t^2}}, y = \arccos \frac{t}{\sqrt{1+t^2}}.$$

12. 若 $x^2 - xy + 2y^2 + x - y - 1 = 0$, 求出 dy/dx , d^2y/dx^2 和 d^3y/dx^3 在 $x = 0, y = 1$ 时的值。

13. 假设已知函数矩阵 $\mathbf{f}(x, y, z) = \begin{bmatrix} 3x + e^{yz} \\ x^3 + y^2 \sin z \end{bmatrix}$, 试求出其 Jacobi 矩阵。

14. 若 $u = x - y + x^2 + 2xy + y^2 + x^3 - 3x^2y - y^3 + x^4 - 4x^2y^2 + y^4$, 试求 $\frac{\partial^4 u}{\partial x^4}$, $\frac{\partial^4 u}{\partial x^3 \partial y}$, $\frac{\partial^4 u}{\partial x^2 \partial y^2}$ 。

15. 若 $u = \ln \frac{1}{\sqrt{(x-\xi)^2 + (y-\eta)^2}}$, 试求 $\frac{\partial^4 u}{\partial x \partial y \partial \xi \partial \eta}$ 。

16. 若 $z = \psi(x^2 + y^2)$, 试求 $y \frac{\partial z}{\partial x} - x \frac{\partial z}{\partial y}$ 。

17. 若 $u = x\phi(x+y) + y\psi(x+y)$, 试求 $\frac{\partial^2 u}{\partial x^2} - 2 \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 u}{\partial y^2}$ 。

18. 试求解下面的不定积分问题:

$$(1) I(x) = -\int \frac{3x^2 + a}{x^2(x^2 + a)^2} dx, \quad (2) I(x) = \int \frac{\sqrt{x(x+1)}}{\sqrt{x} + \sqrt{1+x}} dx,$$

$$(3) I(x) = \int x e^{ax} \cos bx dx, \quad (4) I(t) = \int e^{ax} \sin bx \sin cx dx.$$

19. 试求出下面的定积分或无穷积分:

$$(1) I = \int_0^{\infty} \frac{\cos x}{\sqrt{x}} dx, \quad (2) I = \int_0^1 \frac{1+x^2}{1+x^4} dx, \quad (3) \int_{e^{-2\pi n}}^1 \left| \cos \left(\ln \frac{1}{x} \right) \right| dx.$$

20. 试求解下面的定积分:

$$(1) \int_0^{0.75} \frac{1}{(x+1)\sqrt{x^2+1}} dx, \quad (2) \int_0^1 \frac{\arcsin \sqrt{x}}{\sqrt{x(1-x)}} dx, \quad (3) \int_0^{\pi/4} \left(\frac{\sin x - \cos x}{\sin x + \cos x} \right)^{2n+1} dx.$$

21. 试求出下面的不定积分:

$$(1) \int \frac{\sin^2 x - 4 \sin x \cos x + 3 \cos^2 x}{\sin x + \cos x} dx, \quad (2) \int \frac{\sin^2 x - \sin x \cos x + 2 \cos^2 x}{\sin x + 2 \cos x} dx.$$

22. 假设 $f(x) = e^{-5x} \sin(3x + \pi/3)$, 试求出积分函数 $R(t) = \int_0^t f(x)f(t+x)dx$.

23. 试求出下面重积分:

$$(1) \int_0^{\pi} \int_0^{\pi} |\cos(x+y)| dx dy, \quad (2) \int_0^1 \int_{-1}^{1-x} \arcsin(x+y) dy dx, \\ (3) \iint_{|x|+|y| \leq 1} (|x|+|y|) dx dy, \quad (4) \iint_{\pi^2 \leq x^2+y^2 \leq 4\pi^2} \sin \sqrt{x^2+y^2} dx dy.$$

24. 对 a 的不同取值试求出 $I = \int_0^{\infty} \frac{\cos ax}{1+x^2} dx$.

25. 试证明对任何函数 $f(t)$, $\int_a^b f(t) dt = - \int_b^a f(t) dt$.

26. 试求解下述的多重积分问题:

$$(1) \int_0^2 \int_0^{\sqrt{4-x^2}} \sqrt{4-x^2-y^2} dy dx, \quad (2) \int_0^3 \int_0^{3-x} \int_0^{3-x-y} xyz dz dy dx, \\ (3) \int_0^2 \int_0^{\sqrt{4-x^2}} \int_0^{\sqrt{4-x^2-y^2}} z(x^2+y^2) dz dy dx, \quad (4) \int_0^1 \int_0^x \int_0^y \int_0^z xyzue^{6-x^2-y^2-z^2-u^2} du dz dy dx.$$

27. 试对下面函数进行 Fourier 幂级数展开:

$$(1) f(x) = (\pi - |x|) \sin x, \quad -\pi \leq x < \pi, \quad (2) f(x) = e^{|x|}, \quad -\pi \leq x < \pi, \\ (3) f(x) = \begin{cases} 2x/l, & 0 < x < l/2 \\ 2(l-x)/l, & l/2 < x < l \end{cases}, \text{ 且 } l = \pi.$$

28. 试求出下面函数的 Taylor 幂级数展开:

$$(1) \int_0^x \frac{\sin t}{t} dt, \quad (2) \ln \left(\frac{1+x}{1-x} \right), \quad (3) \ln \left(x + \sqrt{1+x^2} \right), \quad (4) (1+4.2x^2)^{0.2}, \\ (5) e^{-5x} \sin(3x + \pi/3) \text{ 分别关于 } x=0, x=a \text{ 的幂级数展开}.$$

29. 试求出下面多变量函数的 Taylor 幂级数展开:

$$(1) f(x, y) = e^x \cos y \text{ 关于 } x=0, y=0 \text{ 点和 } x=a, y=b \text{ 点的展开}, \\ (2) f(x, y) = \ln(1+x)\ln(1+y) \text{ 关于 } x=0, y=0 \text{ 和 } x=a, y=b \text{ 的展开}.$$

30. 对 $f(x, y) = \frac{1 - \cos(x^2 + y^2)}{(x^2 + y^2)e^{x^2 + y^2}}$ 关于 $x = 1, y = 0$ 点进行二维 Taylor 幂级数展开。

31. 试求下面级数的前 n 项及无穷项的和:

$$(1) \frac{1}{1 \times 6} + \frac{1}{6 \times 11} + \cdots + \frac{1}{(5n-4)(5n+1)} + \cdots,$$

$$(2) \left(\frac{1}{2} + \frac{1}{3}\right) + \left(\frac{1}{2^2} + \frac{1}{3^2}\right) + \cdots + \left(\frac{1}{2^n} + \frac{1}{3^n}\right) + \cdots,$$

$$(3) \frac{1}{3} \left(\frac{x}{2}\right) + \frac{1 \times 4}{3 \times 6} \left(\frac{x}{2}\right)^2 + \frac{1 \times 4 \times 7}{3 \times 6 \times 9} \left(\frac{x}{2}\right)^3 + \frac{1 \times 4 \times 7 \times 10}{3 \times 6 \times 9 \times 12} \left(\frac{x}{2}\right)^4 + \cdots.$$

32. 试求下面无穷级数之和:

$$(1) \sum_{n=0}^{\infty} \frac{\sin^2 n\alpha \sin nx}{n}, \quad (0 < \alpha < \frac{\pi}{2}), \quad (2) \sum_{n=0}^{\infty} \frac{(-1)^n n^3}{(n+1)!} x^n, \quad (3) \sum_{n=0}^{\infty} \frac{x^{4n+1}}{4n+1},$$

$$(4) \frac{1}{3} \frac{x}{2} + \frac{1 \times 4}{3 \times 6} \left(\frac{x}{2}\right)^2 + \frac{1 \times 4 \times 7}{3 \times 6 \times 9} \left(\frac{x}{2}\right)^3 + \frac{1 \times 4 \times 7 \times 10}{3 \times 6 \times 9 \times 12} \left(\frac{x}{2}\right)^4 + \cdots.$$

33. 已知序列通项 a_n , 试求出无穷级数的和:

$$(1) a_n = (\sqrt{1+n} - \sqrt{n})^p \ln \frac{n-1}{n+1}, \quad (2) a_n = \frac{1}{n^{1+k/\ln n}}.$$

34. 试求出下面序列的和:

$$(1) \sum_{n=1}^{\infty} \frac{x^n}{(1+x)(1+x^2)\cdots(1+x^n)}, \quad (2) \sum_{n=2}^{\infty} \frac{(-1)^n}{n^2 + n - 2}, \quad (3) \sum_{n=2}^{\infty} \frac{1}{n^2(n+1)^2(n+2)^2}.$$

35. 试求出下面的极限:

$$(1) \lim_{n \rightarrow \infty} \left(\frac{1}{2^2 - 1} + \frac{1}{4^2 - 1} + \frac{1}{6^2 - 1} + \cdots + \frac{1}{(2n)^2 - 1} \right),$$

$$(2) \lim_{n \rightarrow \infty} n \left(\frac{1}{n^2 + \pi} + \frac{1}{n^2 + 2\pi} + \frac{1}{n^2 + 3\pi} + \cdots + \frac{1}{n^2 + n\pi} \right).$$

36. 试证明 $\cos \theta + \cos 2\theta + \cdots + \cos n\theta = \frac{\sin(n\theta/2) \cos[(n+1)\theta/2]}{\sin \theta/2}$.

37. 试求出下面的无穷序列乘积:

$$(1) \prod_{n=1}^{\infty} \frac{(2n+1)(2n+7)}{(2n+3)(2n+5)}, \quad (2) \prod_{n=1}^{\infty} \frac{9n^2}{(3n-1)(3n+1)}, \quad (3) \prod_{n=1}^{\infty} a^{(-1)^n/n}, a > 0.$$

38. 试求出以下的曲线积分:

$$(1) \int_l (x^2 + y^2) ds, l \text{ 为曲线 } x = a(\cos t + t \sin t), y = a(\sin t - t \cos t), (0 \leq t \leq 2\pi),$$

$$(2) \int_l (yx^3 + e^y) dx + (xy^3 + xe^y - 2y) dy, \text{ 其中 } l \text{ 为 } a^2 x^2 + b^2 y^2 = c^2 \text{ 正向上半椭圆},$$

$$(3) \int_l y dx - x dy + (x^2 + y^2) dz, l \text{ 为曲线 } x = e^t, y = e^{-t}, z = at, 0 \leq t \leq 1, a > 0,$$

$$(4) \int_l (e^x \sin y - my) dx + (e^x \cos y - m) dy, \text{ 其中 } l \text{ 为由 } (a, 0) \text{ 点到 } (0, 0) \text{ 再经 } x^2 + y^2 = ax \text{ 上正向半圆周构成的曲线}.$$

39. 若曲面 S 为半球 $z = \sqrt{R^2 - x^2 - y^2}$ 的底部, 试求下面曲面积分:

$$(1) \int_S xyz^3 ds, \quad (2) \int_S (x + yz^3) dxdy.$$

40. 试对下面数值描述的函数求取各阶数值微分, 并用梯形法求取定积分。

x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1	1.2
y_i	0	2.2077	3.2058	3.4435	3.241	2.8164	2.311	1.8101	1.3602	0.98172	0.67907	0.4473	0.27684

41. 由下表给出的数据计算函数的梯度。已知这些数据是由函数 $f(x, y) = 4 - x^2 - y^2$ 生成的, 试将得出的梯度曲面和理论值进行比较。

0	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
0	4	3.96	3.84	3.64	3.36	3	2.56	2.04	1.44	0.76	0
0.2	3.96	3.92	3.8	3.6	3.32	2.96	2.52	2	1.4	0.72	-0.04
0.4	3.84	3.8	3.68	3.48	3.2	2.84	2.4	1.88	1.28	0.6	-0.16
0.6	3.64	3.6	3.48	3.28	3	2.64	2.2	1.68	1.08	0.4	-0.36
0.8	3.36	3.32	3.2	3	2.72	2.36	1.92	1.4	0.8	0.12	-0.64
1	3	2.96	2.84	2.64	2.36	2	1.56	1.04	0.44	-0.24	-1
1.2	2.56	2.52	2.4	2.2	1.92	1.56	1.12	0.6	0	-0.68	-1.44
1.4	2.04	2	1.88	1.68	1.4	1.04	0.6	0.08	-0.52	-1.2	-1.96
1.6	1.44	1.4	1.28	1.08	0.8	0.44	0	-0.52	-1.12	-1.8	-2.56
1.8	0.76	0.72	0.6	0.4	0.12	-0.24	-0.68	-1.2	-1.8	-2.48	-3.24
2	0	-0.04	-0.16	-0.36	-0.64	-1	-1.44	-1.96	-2.56	-3.24	-4

42. 试用数值方法求出定积分 $\int_0^\pi (\pi - t)^{1/4} f(t) dt$, 其中, $f(t) = e^{-t} \sin(3t + 1)$ 。如果采样点选为 $t = 0.1, 0.2, \dots, \pi$, 试用数值方法求出各个采样点处的积分函数值 $F(t) = \int_0^t (\pi - \tau)^{1/4} f(\tau) d\tau$, 并绘制出 $F(t)$ 曲线。

43. 试用数值积分方法求出下面的多重积分值。值得指出的是, 下面积分的解析解均不存在, 所以应该验证得出的结果是否正确。

$$(1) \int_0^2 \int_0^{e^{-x^2/2}} \sqrt{4 - x^2 - y^2} e^{-x^2 - y^2} dy dx, \quad (2) \int_0^2 \int_0^2 \int_0^2 z(x^2 + y^2) e^{-x^2 - y^2 - z^2 - xz} dz dy dx,$$

$$(3) \int_0^{7/10} \int_0^{4/5} \int_0^{9/10} \int_0^1 \int_0^{11/10} \sqrt{6 - x^2 - y^2 - z^2 - w^2 - u^2} dw du dz dy dx.$$

参考文献

- [1] 吉米多维奇著, 李荣涑译. 数学分析习题集. 北京: 人民教育出版社, 1979
- [2] 陈传璋, 金福临, 朱学炎等. 数学分析. 北京: 人民教育出版社, 1979
- [3] Shampine L F. Vectorized adaptive quadrature in MATLAB. Journal of Computational and Applied Mathematics, 2008, 211(2):131-140
- [4] Wilson H, Gardner B. Numerical integration toolbox (NIT)

第4章 线性代数问题的计算机求解

线性代数问题是科学技术中最常见的数学问题,很多理论和应用都是建立在线性代数基础上的,因此解决线性代数问题是很有意义的。然而经典线性代数课程中介绍的手工推导的方法,不适合高阶矩阵的分析与计算,所以需要计算机数学语言来解决这些高阶问题。

很多计算机数学语言,如 MATLAB 语言,都起源于对线性代数问题的研究。早期的线性代数计算问题侧重于数值解法,很多数学软件包也都是从线性代数的计算开始的。例如,国际上最著名的 EISPACK 是求解矩阵特征值问题的软件包, LINPACK 是求解一般线性代数问题的软件包,目前最新的 LAPACK 也是解决线性代数计算的软件包。随着计算机科学的发展,当前能解决矩阵分析与运算问题的计算机数学语言已经不局限于数值线性代数方法了,逐渐也可以求解解析解问题。Mathematica 和 Maple 等大型计算机数学语言都已经能直接求解线性代数的解析解问题。MATLAB 语言的符号运算工具箱可以调用 Maple 的各种解析运算功能,可以很好地解决线性代数的解析解运算问题。

本章 4.1 节中将介绍矩阵的输入方法,可以用简单的函数直接输入如零矩阵、幺矩阵、单位矩阵、随机数矩阵、对角矩阵、Hilbert 矩阵、相伴矩阵、Vandermonde 矩阵及 Hankel 矩阵等特殊矩阵的 MATLAB 函数,并介绍用 MATLAB 语言的符号运算工具箱语句编写输出符号矩阵的方法、稀疏矩阵的输入方法等,为解决线性代数问题的求解打下良好的基础。4.2 节将介绍矩阵分析的基本概念及求解函数,对矩阵进行数值解与解析解分析,例如矩阵的行列式、迹、秩、范数、特征多项式、逆矩阵和广义逆矩阵、特征值与特征向量等,为矩阵的初步分析做准备。4.3 节介绍各种各样的矩阵分解方法,例如矩阵的相似变换基本概念、矩阵的正交分解、三角分解、对称矩阵的 Cholesky 分解、一般矩阵的伴随分解、Jordan 变换、奇异值分解等,利用矩阵分解的方法可以简化矩阵分析。4.4 节首先分析了线性代数方程可解的条件,分别对唯一解、无穷解和无解等问题进行处理,给出了基于 MATLAB 语言的无穷解的基础解系与通解求取方法,还介绍了无解方程的最小二乘求解方法等,并介绍其他形式的矩阵方程的解析解与数值解方法与结果检验方法,包括 Lyapunov 方程、Sylvester 方程的解析解和数值解法、Riccati 方程的数值解法将在本节中给出,并给出一般 Sylvester 方程解析解的求解程序。4.5 节将研究矩阵元素的非线性运算及矩阵函数求解的问题,给出求解指数矩阵、三角函数矩阵以及一般矩阵函数和复合矩阵函数的解析解应用程序。从理论上说,本书提供的矩阵函数解析解程序可以用于任意复杂的矩阵函数解析运算。

4.1 特殊矩阵的输入

MATLAB 语言中固然可以通过最底层的语句逐行输入一个矩阵,但这样的方法对具有某种特殊结构的矩阵来说显得很烦琐。例如,想输入单位矩阵,再采用逐个元素输入的方法

式是很耗时的,故应该考虑采用 MATLAB 支持的现成函数 `eye()` 来输入特殊矩阵。下面将介绍一些特殊矩阵的输入方法。

4.1.1 数值矩阵的输入

1. 零矩阵、幺矩阵及单位矩阵

在一般的矩阵理论中,把所有元素都为零的矩阵定义为零矩阵,把元素全为 1 的矩阵称为幺矩阵,把主对角线元素均为 1,而其他元素全部为 0 的方阵称为单位矩阵。这里进一步扩展单位矩阵的定义,使其为 $m \times n$ 的矩阵。零矩阵、幺矩阵和扩展单位矩阵的 MATLAB 生成函数分别为

```
A = zeros(n), B = ones(n), C = eye(n)    % 生成 n × n 方阵
A = zeros(m,n); B = ones(m,n); C = eye(m,n) % 生成 m × n 矩阵
A = zeros(size(B))    % 生成和矩阵 B 同样维数的矩阵
```

例 4-1 下面的语句可以生成一个 3×8 的零矩阵 A ,并可以生成一个和 A 维数相同的扩展单位阵 B 。可见,这样特殊矩阵的输入还是很容易的。

```
>> A=zeros(3,8), B=eye(size(A))    % 单位矩阵输入
```

可以将下面两个矩阵输入 MATLAB 工作空间

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

函数 `zeros()` 和 `ones()` 还可用于多维数组的生成,例如, `zeros(3,4,5)` 将生成一个 $3 \times 4 \times 5$ 的三维数组,其元素全部为 0。

2. 随机元素矩阵

顾名思义,随机元素矩阵的各个元素是随机产生的。如果矩阵的随机元素满足 $[0, 1]$ 区间上的均匀分布,则可以由 MATLAB 函数 `rand()` 来生成,该函数通常的调用格式为

```
A = rand(n)    % 生成 n × n 阶标准均匀分布伪随机数方阵
A = rand(n,m)  % 生成 n × m 阶标准均匀分布伪随机数矩阵
```

函数 `rand()` 还可以用于多维数组的生成。满足标准正态分布的随机数矩阵可以由 `randn()` 函数获得,当然也可以使用 `B = rand(size(A))` 形式调用该函数。

这里的随机数实际上是“伪随机数”。所谓伪随机数,就是通过某种数学公式生成的、满足某些随机指标的数据。这样的随机数是可以重复的,与某些用电子方法获得的不可重复的随机数是不同的。

更一般地,如果想生成 (a, b) 区间上均匀分布的随机数,则可以先用 `V = rand(n,m)` 命令生成一个 $(0, 1)$ 上均匀分布的随机数矩阵 V ,再用 $V_1 = a + (b - a) * V$ 语句则可以生成满足需要的矩阵 V_1 。如果想生成满足 $N(\mu, \sigma^2)$ 的正态分布的随机数,则可以先用 `V = randn(n,m)` 命令生成标准正态分布的随机数矩阵 V ,再用 $V_1 = \mu + \sigma * V$ 命令就可以转换成所需的矩阵。

本书第 9 章还将介绍满足特殊分布的伪随机数生成函数与方法。

3. 对角元素矩阵

对角矩阵是一种特殊的矩阵,这种矩阵的主对角线元素可以为 0 或非 0 元素,而非对角线元素的值均为 0。对角矩阵的数学描述方法为 $\text{diag}(\alpha_1, \alpha_2, \dots, \alpha_n)$,其中对角矩阵的矩阵表示为

$$\text{diag}(\alpha_1, \alpha_2, \dots, \alpha_n) = \begin{bmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_n \end{bmatrix} \quad (4-1-1)$$

MATLAB 提供了对角矩阵的生成函数 $\text{diag}()$ 。该函数的调用格式为

```
A = diag(v)      % 已知向量生成对角矩阵
v = diag(A)      % 已知矩阵提取对角元素列向量
A = diag(v,k)   % 生成主对角线上第 k 条对角线为 v 的矩阵
```

例 4-2 MATLAB 中的 $\text{diag}()$ 函数是很有特色的,其不同方式执行不同的任务。例如

```
>> C=[1 2 3]; V=diag(C), V1=diag(V), V2=diag(C,2), V3=diag(C,-1)
```

则可以依次生成下面的矩阵

$$V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, V_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, V_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, V_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix}.$$

在实际应用中还可以取 k 为负值,表示主对角线下数的第 k 条对角线。利用这样的性质,可以容易地构造出三对角矩阵。

```
>> V=diag([1 2 3 4])+diag([2 3 4],1)+diag([5 4 3],-1)
```

如果有若干个子矩阵 A_1, A_2, \dots, A_n ,可以编写一个 $\text{diagm}()$ 函数,构造块对角矩阵。该函数的清单为

```
function A=diagm(varargin), A=[];
for i=1:length(varargin), A1=varargin{i};
[n,m]=size(A); [n1,m1]=size(A1); A(n+1:n+n1,m+1:m+m1)=A1;
end
```

该函数的调用格式为 $A = \text{diagm}(A_1, A_2, \dots, A_n)$,其中,子矩阵个数是任意多的。该函数可以得出块对角矩阵

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_n \end{bmatrix} \quad (4-1-2)$$

4. Hankel 矩阵

Hankel 矩阵的一般形式如下

$$\mathbf{H} = \begin{bmatrix} c_1 & c_2 & \cdots & c_m \\ c_2 & c_3 & \cdots & c_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+m-1} \end{bmatrix} \quad (4-1-3)$$

如果 $n \rightarrow \infty$, 则可以构造无穷型 Hankel 矩阵。Hankel 矩阵是对称矩阵, 其特点是每条反对角线上所有的元素都相同。

在 MATLAB 语言中, 给定两个向量 \mathbf{c} 和 \mathbf{r} , 如果用 $\mathbf{H} = \text{hankel}(\mathbf{c}, \mathbf{r})$ 来生成 \mathbf{H} , 则首先将 \mathbf{H} 矩阵的第一列的各个元素定义为 \mathbf{c} 向量, 将最后一行各个元素定义为 \mathbf{r} , 这样就可以依照 Hankel 矩阵反对角线上元素相等这一特性来写出相应的 Hankel 矩阵。根据 Hankel 矩阵的性质, 其最后一行的第一个元素应该等于第 1 列的最后一个元素, 如果冲突将给出元素冲突的警告信息, 该函数会舍弃 \mathbf{r} 向量的第一个元素构造 Hankel 矩阵。

如果已知一个向量 \mathbf{c} , 则也可以由 $\text{hankel}(\mathbf{c})$ 函数来构造出一个 Hankel 矩阵。将 \mathbf{H} 矩阵的第一列的各个元素定义为 \mathbf{c} 向量, 这样就可以依照 Hankel 矩阵反对角线上元素相等这一特性来写出相应的 Hankel 矩阵, 使得下三角矩阵均为 0。

例 4-3 试用 MATLAB 语句输入下面两个给出的 Hankel 矩阵 \mathbf{H} 。

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 0 \\ 3 & 0 & 0 \end{bmatrix}$$

解 分析给出的矩阵, 可以用向量分别表示该矩阵的首列和最后一行, $\mathbf{C} = [1, 2, 3]$, $\mathbf{R} = [3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9]$, 则可以由下面语句可以生成所需的 Hankel 矩阵。

```
>> c=[1 2 3]; r=[3 4 5 6 7 8 9]; H1=hankel(c,r), H2=hankel(c)
```

5. Hilbert 矩阵及 Hilbert 逆矩阵

Hilbert 矩阵是一类特殊矩阵, 它的第 (i, j) 元素的值满足 $h_{i,j} = 1/(i + j - 1)$, 这时一个 $n \times n$ 阶的 Hilbert 矩阵可以写成

$$\mathbf{H} = \begin{bmatrix} 1 & 1/2 & 1/3 & \cdots & 1/n \\ 1/2 & 1/3 & 1/4 & \cdots & 1/(n+1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/(n+1) & 1/(n+2) & \cdots & 1/(2n-1) \end{bmatrix} \quad (4-1-4)$$

产生 Hilbert 矩阵的 MATLAB 函数为 $\mathbf{A} = \text{hilb}(n)$, 其中, n 为要产生的矩阵阶次。

高阶 Hilbert 矩阵一般为坏条件的矩阵, 所以直接对之求逆一般往往会引出浮点溢出现象。MATLAB 提供了直接求取 Hilbert 逆矩阵的算法及函数 $\mathbf{B} = \text{invhilb}(n)$ 。

由于 Hilbert 矩阵本身接近奇异的性质, 所以在处理该矩阵时建议尽量采用符号运算工具箱, 而采用数值解时应该检验结果的正确性。

6. Vandermonde 矩阵

假设有一个序列 \mathbf{c} , 其各个元素满足 $\{c_1, c_2, \dots, c_n\}$, 则可以写出一个矩阵, 其第 (i, j) 元素满足 $v_{i,j} = c_i^{n-j}$, $i, j = 1, 2, \dots, n$ 。这样可以构成一个矩阵

$$\mathbf{V} = \begin{bmatrix} c_1^{n-1} & c_1^{n-2} & \cdots & c_1 & 1 \\ c_2^{n-1} & c_2^{n-2} & \cdots & c_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_n^{n-1} & c_n^{n-2} & \cdots & c_n & 1 \end{bmatrix} \quad (4-1-5)$$

该矩阵称作 Vandermonde 矩阵。如果已知向量 $\mathbf{c} = [c_1, c_2, \dots, c_n]$, 则可以由 MATLAB 提供的 `V = vander(c)` 函数来构造一个 Vandermonde 矩阵。

例 4-4 试建立 Vandermonde 矩阵 $\mathbf{V} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 9 & 16 & 25 \\ 1 & 8 & 27 & 64 & 125 \\ 1 & 16 & 81 & 256 & 625 \end{bmatrix}$ 。

解 依给出的矩阵类型, 先生成向量 $\mathbf{c} = [1, 2, 3, 4, 5]$, 得出其 Vandermonde 标准型后再将其逆时针旋转 90° , 则可以直接得出所需 \mathbf{V} 矩阵。

```
>> c=[1, 2, 3, 4, 5]; V=vander(c); V=rot90(V)
```

7. 相伴矩阵

假设有一个首一化的多项式

$$p(s) = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_{n-1} s + a_n \quad (4-1-6)$$

则可以写出一个相伴矩阵 (或称友矩阵、companion 矩阵)

$$\mathbf{A}_c = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \quad (4-1-7)$$

生成相伴矩阵的 MATLAB 函数调用格式为 `Ac = compan(a)`, 其中, \mathbf{a} 为降幂排列的多项式系数向量, 该函数将自动对多项式进行首一化处理。

例 4-5 考虑一个多项式 $P(s) = 2s^4 + 4s^2 + 5s + 6$, 试写出该多项式的相伴矩阵。

解 先输入特征多项式, 则相伴矩阵可以通过下面的语句建立起来, 赋给 \mathbf{A} 矩阵

```
>> P=[2 0 4 5 6]; A=compan(P)
```

这些语句可以得出相伴矩阵

$$\mathbf{A} = \begin{bmatrix} -0 & -2 & -2.5 & -3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

4.1.2 符号矩阵的输入

如果已经建立起了数值矩阵 A , 则可以由 $B = \text{sym}(A)$ 语句将其转换成符号矩阵。这样, 所有数值矩阵均可以通过这样的形式转换成符号矩阵, 可以利用符号运算工具箱获得更高精度的解。相反地, 一个全数值的符号矩阵 B 可以通过 $A_1 = \text{double}(B)$ 转换成双精度矩阵 A_1 。

对于一些特殊矩阵形式, 如 Vandermonde 矩阵、Hankel 矩阵及相伴矩阵, 符号运算工具箱不直接支持它们, 所以需要编写下面的一些函数。其实, 处理这类问题最好的方法是编写同名的重载函数, 以前的版本确实也支持这样的方法, 不过新版本不支持这样的处理, 所以我们需要编写一些普通的函数, 我们给这些函数的文件名后面加 `sym` 字符以示区别。

参考 MATLAB 语言对数值矩阵生成的相应函数, 可以改写出适合符号运算的新函数。例如, 可以编写出生成相伴矩阵的 MATLAB 函数 `companysym()`

```
function A=companysym(c), n=length(c);
if min(size(c))>1, error('Input argument must be a vector. '), end
if n<=1, A=[]; elseif n==2, A=-c(2)/c(1);
else, c=c(:)'; A=sym(diag(ones(1,n-2),-1)); A(1,:)= -c(2:n)./c(1); end
```

例 4-6 试用解析方法建立起下面多项式的相伴矩阵

$$P(\lambda) = a_1\lambda^9 + a_2\lambda^8 + a_3\lambda^7 + \cdots + a_8\lambda^2 + a_9\lambda + a_{10}$$

解 由上面编写的函数, 可以先申明符号变量, 并以向量形式输入多项式, 最后可以通过下面的语句直接建立所需的相伴矩阵。

```
>> syms a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
A=companysym([a1 a2 a3 a4 a5 a6 a7 a8 a9 a10])
```

这样建立的矩阵为

$$\begin{bmatrix} -a_2/a_1 & -a_3/a_1 & -a_4/a_1 & -a_5/a_1 & -a_6/a_1 & -a_7/a_1 & -a_8/a_1 & -a_9/a_1 & -a_{10}/a_1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

类似地, Hankel 矩阵的 MATLAB 函数 `hankelsym()` 为

```
function H=hankelsym(c,r)
c=c(:); nc=length(c); if nargin==1, r=zeros(size(c)); end
r=r(:); nr = length(r); x=[c; r((2:nr)')]; cidx=(1:nc)';
ridx=0:(nr-1); H1=cidx(:,ones(nr,1))+ridx(ones(nc,1),:); H=x(H1);
```

还可以建立起生成 Vandermonde 矩阵的 MATLAB 函数 `vandersym()`, 该函数的格式与 MATLAB 语言中的数值函数完全一致

```
function A=vandersym(v)
n=length(v); v=v(:); A=sym(ones(n)); for j=n-1:-1:1, A(:,j)=v.*A(:,j+1); end
```

下面给出生成元素为 a_{ij} 的任意矩阵的函数, 该函数可以生成指定阶次的任意矩阵

```
function A=any_matrix(n,varargin)
[m,a_str]=default_vals({n,'a'},varargin{:});
for i=1:n, for j=1:m,
    str=[a_str int2str(i),int2str(j)]; eval(['syms ' str]);
    eval(['A(i,j)= ' str ';']);
end, end
```

例 4-7 调用下面的语句可以直接生成三个任意矩阵

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, \quad B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}, \quad C = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{bmatrix}$$

```
>> A=any_matrix(4), B=A=any_matrix(4,2), C=any_matrix(4,4,'f')
```

4.1.3 稀疏矩阵的输入

在很多应用中经常需要描述一些特殊的大型矩阵, 而这类矩阵的大部分元素都是零, 仅有少部分非零元素, 这样的矩阵称为稀疏矩阵。如果选择合适的求解算法, 稀疏矩阵的计算比常规矩阵效率更高。MATLAB 支持稀疏矩阵的输入, 且很多矩阵分析函数也支持稀疏矩阵的特别处理。

稀疏矩阵可以由 `sparse()` 函数读入 MATLAB, 其调用格式为 **`A = sparse(p,q,w)`**, 其中 **`p,q`** 为非零元素的行号和列号构成的向量, **`w`** 为相应位置的矩阵元素构成的向量。这三个向量的长度是一致的, 否则将给出错误信息。

`B = full(A)` 函数可将稀疏矩阵 **`A`** 转换成常规矩阵 **`B`**, 也可以由 **`A = sparse(B)`** 将常规矩阵转回稀疏矩阵。如果一个矩阵 2/3 以上的元素为零, 则利用稀疏矩阵的方式存储矩阵比较经济, 且矩阵的稀疏度越高存储越经济。6.7 节将通过例子演示稀疏矩阵的应用。

4.2 矩阵基本分析

4.2.1 矩阵基本概念与性质

1. 行列式

矩阵 $A = \{a_{ij}\}$ 的行列式定义为

$$D = |A| = \det(A) = \sum (-1)^k a_{1k_1} a_{2k_2} \cdots a_{nk_n} \quad (4-2-1)$$

式中, k_1, k_2, \cdots, k_n 是将序列 $1, 2, \cdots, n$ 的元素交换 k 次所得出的一个序列, 每个这样的序列称为一个置换 (permutation); 而 Σ 表示对 k_1, k_2, \cdots, k_n 取遍 $1, 2, \cdots, n$ 的所有排列的求和。

例 4-8 试求出矩阵 A 的行列式

$$\mathbf{A} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; det(A)
```

```
>> tic, H=sym(hilb(20)); det(H), toc
```

$$\det(\boldsymbol{H}) = \frac{1}{\underbrace{2377454716 \cdots 368000000000000000000000000000000}_{225 \text{ 位, 因排版限制省略了中间的数字}}} \approx 4.206179 \times 10^{-224}$$

例 4-10 试给出一般 4×4 矩阵的行列式计算公式。

```
>> A=any_matrix(4); d=det(A)
```

$$\begin{aligned} d = & a_{11}a_{22}a_{33}a_{44} - a_{11}a_{22}a_{34}a_{43} - a_{11}a_{23}a_{32}a_{44} + a_{11}a_{23}a_{34}a_{42} + a_{11}a_{24}a_{32}a_{43} - a_{11}a_{24}a_{33}a_{42} \\ & - a_{12}a_{21}a_{33}a_{44} + a_{12}a_{21}a_{34}a_{43} + a_{12}a_{23}a_{31}a_{44} - a_{12}a_{23}a_{34}a_{41} - a_{12}a_{24}a_{31}a_{43} + a_{12}a_{24}a_{33}a_{41} \\ & + a_{13}a_{21}a_{32}a_{44} - a_{13}a_{21}a_{34}a_{42} - a_{13}a_{22}a_{31}a_{44} + a_{13}a_{22}a_{34}a_{41} + a_{13}a_{24}a_{31}a_{42} - a_{13}a_{24}a_{32}a_{41} \\ & - a_{14}a_{21}a_{32}a_{43} + a_{14}a_{21}a_{33}a_{42} + a_{14}a_{22}a_{31}a_{43} - a_{14}a_{22}a_{33}a_{41} - a_{14}a_{23}a_{31}a_{42} + a_{14}a_{23}a_{32}a_{41} \end{aligned}$$

如果想求出 A_{23} 的值,有两种方法可以采用,其一是由定义直接求出,下面语句可以得出其结果为 $A_{23} = -a_{11}a_{32}a_{44} + a_{11}a_{34}a_{42} + a_{12}a_{31}a_{44} - a_{12}a_{34}a_{41} - a_{14}a_{31}a_{42} + a_{14}a_{32}a_{41}$ 。

```
>> i=2; j=3; B=A; B(i,:)=[]; B(:,j)=[]; A23=(-1)^(i+j)*det(B)
```

另一种方法是从前面得出的 d 值中把不含 a_{23} 的项剔除掉,然后再除以 a_{23} ,具体的剔除方法是用 d 减去将 a_{23} 置零后剩下的项。这样就可以得出其代数余子式的值,与前面得出的完全一致。

```
>> syms a23; A23_1=simple((d-subs(d,a23,0))/a23)
```

2. 矩阵的迹

假设一个方阵为 $\mathbf{A} = \{a_{ij}\}$, $i, j = 1, 2, \dots, n$, 则矩阵 \mathbf{A} 的迹定义为该矩阵对角线上各个元素之和,即

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^n a_{ii} \quad (4-2-2)$$

由代数理论可知,矩阵的迹和该矩阵的特征值之和是相同的,矩阵 \mathbf{A} 的迹可以由 MATLAB 函数 `trace()` 求出,该函数的调用和数学表示相似,即 $t = \text{trace}(\mathbf{A})$ 。

例 4-8 中矩阵的迹可以由 MATLAB 语句直接求出为 $\text{trace}(\mathbf{A}) = 34$ 。

3. 矩阵的秩

若矩阵所有的列向量中共有 r_c 个线性无关,则称矩阵的列秩为 r_c 。如果 $r_c = m$, 则称 \mathbf{A} 为列满秩矩阵。相应地,若矩阵 \mathbf{A} 的行向量中有 r_r 个是线性无关的,则称矩阵 \mathbf{A} 的行秩为 r_r 。如果 $r_r = n$, 则称 \mathbf{A} 为行满秩矩阵。可以证明,矩阵的行秩和列秩是相等的,故称之为矩阵的秩,记作

$$\text{rank}(\mathbf{A}) = r_c = r_r \quad (4-2-3)$$

这时,矩阵的秩为 $\text{rank}(\mathbf{A})$ 。矩阵的秩也表示该矩阵中行列式不等于 0 的子式的最大阶次。所谓子式,即为从原矩阵中任取 k 行及 k 列所构成的子矩阵。

矩阵求秩的算法也是多种多样的,其区别是有的算法是稳定的,而有的算法可能因矩阵的条件数过大不是很稳定。MATLAB 中采用的算法是基于矩阵的奇异值分解的算法^[1]。首先对矩阵进行奇异值分解,得出矩阵 \mathbf{A} 的 n 个奇异值 σ_i , $i = 1, 2, \dots, n$, 在这 n 个奇异值中找出大于给定误差限 ε 的个数 r , 这时 r 就可以认为是 \mathbf{A} 矩阵的秩。

MATLAB 提供的内核函数 `rank()` 可以求取给定矩阵的秩。该函数的调用格式为

```
 $r = \text{rank}(\mathbf{A})$       % 用默认的精度求数值秩
 $r = \text{rank}(\mathbf{A}, \varepsilon)$  % 给定精度  $\varepsilon$  下求数值秩
```

其中, \mathbf{A} 为给定矩阵, ε 为机器精度。符号运算工具箱中也提供了 `rank()` 函数,可以求出数值矩阵秩的解析解,其调用格式与前面的方法完全一致。

例 4-11 试求出例 4-8 中给出的 \mathbf{A} 矩阵的秩。

解 用 `rank(A)` 函数可以得出该矩阵的秩。该矩阵的秩为 3, 小于矩阵的阶次,故可以得出结论: 矩阵 \mathbf{A} 是非满秩矩阵或奇异矩阵。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; rank(A)
```

例 4-12 现在考虑例 4-9 中给出的 20×20 的 Hilbert 矩阵,考虑用数值方法和解析方法分别求该矩阵的秩,并比较其正确性。

解 先考虑数值方法,应该给出下面的命令,然而得出的数值秩为 12。

```
>> H=hilb(20); rank(H)
```

故而可以得出结论。因为该矩阵的秩和矩阵阶次相差太多,所以 H 矩阵为非满秩矩阵。其实该函数对一些接近奇异的矩阵可能出现错误结论,用数值解的方法应该注意。如果有可能应该采用解析解的方法求解该问题,下面语句可以得出矩阵的秩为 20

```
>> H=sym(hilb(20)); rank(H) % 可见原矩阵为非奇异矩阵
```

4. 矩阵的范数

矩阵的范数是对矩阵的一种测度。在介绍矩阵的范数之前,首先要介绍向量范数的基本概念。如果对线性空间中的一个向量 \mathbf{x} 存在一个函数 $\rho(\mathbf{x})$ 满足下面 3 个条件:

- (1) $\rho(\mathbf{x}) \geq 0$, 且 $\rho(\mathbf{x}) = 0$ 的充要条件是 $\mathbf{x} = \mathbf{0}$;
- (2) $\rho(a\mathbf{x}) = |a|\rho(\mathbf{x})$, a 为任意标量;
- (3) 对向量 \mathbf{x} 和 \mathbf{y} 有 $\rho(\mathbf{x} + \mathbf{y}) \leq \rho(\mathbf{x}) + \rho(\mathbf{y})$ 。

则称 $\rho(\mathbf{x})$ 为 \mathbf{x} 向量的范数。范数的形式是多种多样的。可以证明,下面给出的一族式子都满足上述的 3 个条件

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}, \quad p = 1, 2, \dots, \text{ 且 } \|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i| \quad (4-2-4)$$

这里用到了向量范数的记号 $\|\mathbf{x}\|_p$ 。

矩阵的范数定义比向量的稍复杂一些,其数学定义为:对于任意的非零向量 \mathbf{x} , 矩阵 \mathbf{A} 的范数为

$$\|\mathbf{A}\| = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} \quad (4-2-5)$$

和向量的范数一样,对矩阵来说也有常用的范数定义方法

$$\|\mathbf{A}\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|, \quad \|\mathbf{A}\|_2 = \sqrt{s_{\max}(\mathbf{A}^T \mathbf{A})}, \quad \|\mathbf{A}\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \quad (4-2-6)$$

其中, $s(\mathbf{X})$ 为 \mathbf{X} 矩阵的特征值,而 $s_{\max}(\mathbf{A}^T \mathbf{A})$ 为 $\mathbf{A}^T \mathbf{A}$ 矩阵的最大特征值。事实上, $\|\mathbf{A}\|_2$ 还等于 \mathbf{A} 矩阵的最大奇异值。

MATLAB 提供了求取矩阵范数的函数 `norm()`, 允许求各种意义下的矩阵的范数。该函数的调用格式为 `N = norm(A, 选项)`, 其中“选项”可以为 1, 2 等, 具体见表 4-1。如果不给出任何选项, 则将计算出 $\|\mathbf{A}\|_2$ 。

例 4-13 求例 4-8 中矩阵 \mathbf{A} 的各种范数。可以由下面的 MATLAB 函数直接求出

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; n1=norm(A)
      n2=norm(A,2), n3=norm(A,1), n4=norm(A,Inf), n5=norm(A,'fro')
```

得出的各个范数为 $\|\mathbf{A}\|_1 = \|\mathbf{A}\|_2 = \|\mathbf{A}\|_\infty = 34$, $\|\mathbf{A}\|_F = 38.6782$ 。

这里有两点值得注意,首先 `norm(A)` 和 `norm(A,2)` 应该给出同样的结果,因为它们都表示 $\|\mathbf{A}\|_2$, 其次因为巧合,在这个例子中, $\|\mathbf{A}\|_1 = \|\mathbf{A}\|_\infty$ 。但一般情况下, $\|\mathbf{A}\|_1 = \|\mathbf{A}\|_\infty$ 不一定能满足。

符号运算工具箱的 `norm()` 函数只能用于数值矩阵求取范数,并不能用于一般含有变量的矩阵。早期版本中即使数值型的符号矩阵也不能直接使用 `norm()` 函数,而先将矩阵用 `double()` 函数转换成双精度数值矩阵,然后再调用双精度矩阵的 `norm()` 函数。

表 4-1 矩阵范数函数的选项表

选 项	意义及算法
无	矩阵的最大奇异值, 即 $\ \mathbf{A}\ _2$
2	与默认调用方式相同, 即 $\ \mathbf{A}\ _2$
1	矩阵的 1-范数, 即 $\ \mathbf{A}\ _1$
Inf 或 'inf'	矩阵的无穷范数, 即 $\ \mathbf{A}\ _\infty$
'fro'	矩阵的 Frobinus 范数, 即 $\ \mathbf{A}\ _F = \sqrt{\sum (\mathbf{A}^T \mathbf{A})_{ii}}$
数值 P	对向量可取任何整数, 而对矩阵只可取 1, 2, inf 或 'fro'
-inf	只可用于向量, $\ \mathbf{A}\ _{-\infty} = \min(\sum a_i)$

5. 特征多项式

引入算子 s , 并构造一个矩阵 $s\mathbf{I} - \mathbf{A}$, 再求出该矩阵的行列式, 则可以得出一个关于算子 s 的多项式

$$C(s) = \det(s\mathbf{I} - \mathbf{A}) = s^n + c_1 s^{n-1} + \cdots + c_{n-1} s + c_n \quad (4-2-7)$$

这样的多项式 $C(s)$ 称为矩阵 \mathbf{A} 的特征多项式。其中, 系数 $c_i, i = 1, 2, \cdots, n$ 称为矩阵的特征多项式系数。

MATLAB 提供了求取矩阵特征多项式系数的函数 `c = poly(A)`, 返回的 \mathbf{c} 为行向量, 其各个分量为矩阵 \mathbf{A} 的降幂排列的特征多项式系数。该函数的另外一种调用格式是, 如果给定的 \mathbf{A} 为向量, 则假定该向量是一个矩阵的特征值, 由此求出该矩阵的特征多项式系数, 如果向量 \mathbf{A} 中有 Inf 或 NaN 值, 则首先剔除它再计算特征多项式系数。

值得指出的是, 如果 \mathbf{A} 为符号矩阵, 该函数仍然适用, 但得出的不是系数向量, 而是多项式的数学表达式本身。

例 4-14 试求出例 4-8 中给出的 \mathbf{A} 矩阵的特征多项式。

解 可以通过下面的 `poly()` 函数直接求出该矩阵的特征多项式, $p \approx [1, -34, -80, 2720, 0]$, 经检验误差为 5.6248×10^{-12} 。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
p=poly(A), norm(p-[1,-34,-80,2720,0])
```

用符号运算工具箱中的 `poly(sym(A))` 函数同样可以求出矩阵 \mathbf{A} 的特征多项式的解析形式。

在实际应用中还有其他简单的数值方法可以精确地求出矩阵的特征多项式系数。例如, 下面给出的 Leverrier-Faddeev 递推算法也可以求出矩阵的特征多项式

$$c_{k+1} = -\frac{1}{k} \text{tr}(\mathbf{A}\mathbf{R}_k), \quad \mathbf{R}_{k+1} = \mathbf{A}\mathbf{R}_k + c_{k+1}\mathbf{I}, \quad k = 1, \cdots, n \quad (4-2-8)$$

其中 $\mathbf{R}_1 = \mathbf{I}, c_1 = 1$ 。该算法首先给出一个单位阵 \mathbf{I} , 并将之赋给 \mathbf{R}_1 , 然后对每个 k 的值分别求出特征多项式参数, 并更新 \mathbf{R}_k 矩阵, 最终得出矩阵的特征多项式系数 c_k 。该算法可以直接由下面的 MATLAB 语句编写一个 `poly1()` 函数实现

```
function c=poly1(A)
[nr,nc]=size(A);
if nc==nr, I=eye(nc); R=I; c=[1 zeros(1,nc)];
    for k=1:nc, c(k+1)=-1/k*trace(A*R); R=A*R+c(k+1)*I; end
elseif (nr==1 | nc==1), A=A(isfinite(A)); n=length(A);
    c = [1 zeros(1,n)];
    for j=1:n, c(2:(j+1))=c(2:(j+1))-A(j).*c(1:j); end
else, error('Argument must be a vector or a square matrix.');
```

调用新的 `poly1(A)` 函数, 则可以得出特征多项式的精确结果。

例 4-15 试推导出向量 $B = [a_1, a_2, a_3, a_4, a_5]$ 对应的 Hankel 矩阵的特征多项式。

解 可以首先构造 Hankel 矩阵 A , 这样就能用 `poly(A)` 函数获得该矩阵的特征多项式

```
>> syms a1 a2 a3 a4 a5 x; A=hankelsym([a1 a2 a3 a4 a5]);
    collect(poly(A),x) % 按 x 合并同类项, 化简多项式
```

该矩阵的特征多项式数学表示为

$$\begin{aligned} \det(xI - A) = & x^5 + (-a_3 - a_5 - a_1)x^4 + (a_5a_1 + a_3a_1 + a_5a_3 - 2a_4^2 - 2a_5^2 - a_2^2 - a_3^2)x^3 \\ & + (-a_1a_3a_5 + 2a_5^3 - 2a_2a_4a_3 + a_2^2a_5 + a_1a_4^2 + a_3^3 + a_1a_5^2 + a_3a_5^2 + a_5a_4^2 + a_4^2a_3 - 2a_2a_5a_4)x^2 \\ & + (2a_2a_5^2a_4 + a_4^4 + a_5^4 + a_3^2a_5^2 + a_5^2a_4^2 - 3a_3a_5a_4^2 - a_1a_5^3 - a_3a_5^3)x - a_5^5 \end{aligned}$$

MATLAB 的 `charpoly()` 函数可以用来提取特征多项式的系数 $p = \text{charpoly}(A)$ 。早期版本则可以用 `polycoef()` 函数^[2]提取多项式系数

```
function pc=polycoef(p,x,n)
pc(n+1)=subs(p,x,0); p1=p;
for i=1:n, p1=diff(p1,x)/i; pc(n-i+1)=subs(p1,x,0); end
```

例 4-16 试提取例 4-15 中 A 矩阵特征多项式的系数。

解 仿照前面的例子, 可以求出并提取矩阵的特征多项式系数

```
>> syms a1 a2 a3 a4 a5 x; A=hankelsym([a1 a2 a3 a4 a5]);
    P=charpoly(A) % 或早期版本使用 p=poly(A); P=polycoef(p,x,5)
```

结果为

$$\begin{aligned} p_1 &= 1, \quad p_2 = -a_3 - a_5 - a_1, \quad p_3 = a_5a_1 + a_3a_1 + a_5a_3 - 2a_4^2 - 2a_5^2 - a_2^2 - a_3^2 \\ p_4 &= -a_1a_3a_5 + 2a_5^3 - 2a_2a_4a_3 + a_2^2a_5 + a_1a_4^2 + a_3^3 + a_1a_5^2 + a_3a_5^2 + a_5a_4^2 + a_4^2a_3 - 2a_2a_5a_4 \\ p_5 &= 2a_2a_5^2a_4 + a_4^4 + a_5^4 + a_3^2a_5^2 + a_5^2a_4^2 - 3a_3a_5a_4^2 - a_1a_5^3 - a_3a_5^3, \quad p_6 = -a_5^5 \end{aligned}$$

6. 矩阵多项式的求解

矩阵多项式的数学形式为

$$B = a_1A^n + a_2A^{n-1} + \cdots + a_nA + a_{n+1}I \quad (4-2-9)$$

其中, A 为一个给定矩阵, I 为和 A 同阶次的单位矩阵, 这时返回的矩阵 B 为矩阵多项式的值。矩阵多项式的值在 MATLAB 语言环境中可以由 `polyvalm()` 函数求出, 该函数的

调用格式为 $\mathbf{B} = \text{polyvalm}(\mathbf{a}, \mathbf{A})$, 其中, \mathbf{a} 为多项式系数降幂排列构成的向量, 即 $\mathbf{a} = [a_1, a_2, \dots, a_n, a_{n+1}]$ 。

相应地, 还可以按点运算的方式定义一种多项式运算为

$$\mathbf{C} = a_1 \mathbf{x}.^n + a_2 \mathbf{x}^{(n-1)} + \dots + a_{n+1} \quad (4-2-10)$$

这时, 矩阵 \mathbf{C} 可以由下面的语句直接计算出来: $\mathbf{C} = \text{polyval}(\mathbf{a}, \mathbf{x})$ 。

若由 MATLAB 的符号运算工具箱给出多项式 p , 则可以调用 `subs()` 函数求出点运算意义下多项式的值。该函数在此问题上的具体调用格式为 $\mathbf{C} = \text{subs}(p, s, \mathbf{x})$ 。

MATLAB 给出的 `polyvalm()` 函数只能用于数值矩阵的多项式矩阵求值, 对该函数拓展, 即可以写出用于符号矩阵的多项式矩阵, 拓展的函数如下

```
function B=polyvalmsym(p,A)
E=eye(size(A)); B=zeros(size(A)); n=length(A);
for i=n+1:-1:1, B=B+p(i)*E; E=E*A; end
```

例 4-17 Cayley-Hamilton 定理是矩阵理论中一个重要的定理: 若矩阵 \mathbf{A} 的特征多项式为

$$f(s) = \det(s\mathbf{I} - \mathbf{A}) = a_1 s^n + a_2 s^{n-1} + \dots + a_n s + a_{n+1} \quad (4-2-11)$$

则有 $f(\mathbf{A}) = \mathbf{0}$, 亦即

$$a_1 \mathbf{A}^n + a_2 \mathbf{A}^{n-1} + \dots + a_n \mathbf{A} + a_{n+1} \mathbf{I} = \mathbf{0} \quad (4-2-12)$$

假设矩阵 \mathbf{A} 为 Vandermonde 矩阵, 验证其满足 Cayley-Hamilton 定理

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 729 & 243 & 81 & 27 & 9 & 3 & 1 \\ 4096 & 1024 & 256 & 64 & 16 & 4 & 1 \\ 15625 & 3125 & 625 & 125 & 25 & 5 & 1 \\ 46656 & 7776 & 1296 & 216 & 36 & 6 & 1 \\ 117649 & 16807 & 2401 & 343 & 49 & 7 & 1 \end{bmatrix}$$

解 可以试图由下面的 MATLAB 语句来验证 Cayley-Hamilton 定理

```
>> A=vander([1 2 3 4 5 6 7]), aa=poly(A); B=polyvalm(aa,A); e=norm(B)
```

由于使用的 `poly()` 函数会产生一定的误差, 而该误差在矩阵多项式求解中导致了巨大的误差 $e = 3.5654 \times 10^5$, 从而得出错误结论。由此看来, `poly()` 函数的误差有时是不可忽略的。如果把上面语句中的 `poly()` 函数用前面编写的 `poly1()` 函数代替, 得出的 \mathbf{B} 矩阵就会完全等于 $\mathbf{0}$, 故该矩阵满足 Cayley-Hamilton 定理。

```
>> aa1=poly1(A); B1=polyvalm(aa1,A); norm(B1)
```

例 4-18 试证明一般 5×5 矩阵满足 Cayley-Hamilton 定理。

解 要求解此问题需要首先申明一批符号变量, 构造出任意的 5×5 矩阵 \mathbf{A} , 然后用 `polychar()` 函数求出其特征多项式系数向量。因为 `polyvalm()` 不支持符号变量的处理, 这里需要由底层命令求取多项式矩阵。得出的多项式矩阵化简后可以得出其范数为 0 的结论, 故此证明这样任意的 5×5 矩阵满足 Cayley-Hamilton 定理。下面的语句耗时大约一分钟

```
>> A=any_matrix(5); p=charpoly(A); E=polyvalmsym(p,A); norm(simple(E))
```

7. 符号多项式与数值多项式的转换

若已知数值多项式系数构成的向量 $\mathbf{p} = [a_1, a_2, \dots, a_{n+1}]$, 则可以通过符号运算工具箱提供的 `poly2sym()` 函数转换成多项式表示。若已知多项式的符号表达式, 则可以由 `sym2poly()` 函数转换成系数向量形式。这两个函数的调用格式都是很简单的

$f = \text{poly2sym}(\mathbf{p})$ 或 $f = \text{poly2sym}(\mathbf{p}, x)$, $\mathbf{p} = \text{sym2poly}(f)$

例 4-19 已知多项式 $f = s^5 + 2s^4 + 3s^3 + 4s^2 + 5s + 6$, 试用不同形式表示该多项式。

解 该多项式可以用两种形式先定义出来。例如, 可以用数值形式先定义之, 则可以用相应的方式将其转换成符号型的多项式, 也可以转换回向量。

```
>> P=[1 2 3 4 5 6]; f=poly2sym(P,'v'), P1=sym2poly(f)
```

4.2.2 逆矩阵与广义逆矩阵

1. 矩阵的逆矩阵

对一个已知的 $n \times n$ 非奇异方阵 \mathbf{A} 来说, 如果有一个同样大小的 \mathbf{C} 矩阵满足

$$\mathbf{AC} = \mathbf{CA} = \mathbf{I} \quad (4-2-13)$$

式中 \mathbf{I} 为单位阵, 则称 \mathbf{C} 矩阵为 \mathbf{A} 矩阵的逆矩阵, 并记作 $\mathbf{C} = \mathbf{A}^{-1}$ 。

MATLAB 语言中提供了 `$\mathbf{C} = \text{inv}(\mathbf{A})$` 函数, 可以直接用来求取矩阵的逆矩阵 \mathbf{C} 。该函数同样适用于符号变量构成的矩阵的求逆。

例 4-20 试求取 Hilbert 矩阵的逆矩阵。

解 先考虑 4×4 Hilbert 矩阵, 调用 MATLAB 的 `inv()` 函数可以立即得出该矩阵的逆矩阵来。

```
>> format long; H=hilb(4); H1=inv(H), norm(H*H1-eye(4))
```

这样得出的逆矩阵如下, 且误差矩阵的范数为 1.3931×10^{-13}

$$\mathbf{H}^{-1} = \begin{bmatrix} 15.999999999999 & -119.99999999999 & 239.99999999998 & -139.99999999999 \\ -119.99999999999 & 1199.9999999999 & -2699.9999999997 & 1679.9999999998 \\ 239.99999999998 & -2699.9999999997 & 6479.9999999994 & -4199.9999999996 \\ -139.99999999999 & 1679.9999999998 & -4199.9999999996 & 2799.9999999997 \end{bmatrix}$$

如果误差矩阵的范数是一个微小的数, 则可以接受得出的逆矩阵, 否则应该认为其不正确。从本例的结果看, 此误差虽然未小于 MATLAB 矩阵运算的一般误差 ($10^{-15} \sim 10^{-16}$ 级), 但还是比较小的, 因此可以接受得出的逆矩阵。

考虑到高阶 Hilbert 矩阵接近于奇异矩阵, 一般不建议用 `inv()` 函数直接求解, 可以采用 `invhilb()` 函数直接产生逆矩阵, 得出的误差为 $n = 5.684 \times 10^{-14}$

```
>> H2=invhilb(4); n=norm(H*H2-eye(size(H)))
```

可见, 对于低阶矩阵, 用 `invhilb()` 计算出来的逆矩阵的精度也显著改善了。现在考虑 10×10 的 Hilbert 矩阵, 则两个误差分别为 $n_1 = 1.4718 \times 10^{-4}$, $n_2 = 1.6129 \times 10^{-5}$

```
>> H=hilb(10); H1=inv(H); n1=norm(H*H1-eye(size(H)))
H2=invhilb(10); n2=norm(H*H2-eye(size(H)))
```

这样虽然后者得出的逆矩阵精度远高于直接求逆的精度,但还是难以达到较高的要求。进一步扩大矩阵的阶次,例如需要研究 13×13 的 Hilbert 矩阵,则两个逆矩阵的误差分别为 $n_1 = 2.1315$, $n_2 = 11.3549$,可见得出的误差过大,说明原矩阵接近奇异矩阵。

```
>> H=hilb(13); H1=inv(H); n1=norm(H*H1-eye(size(H)))
      H2=invhilb(13); n2=norm(H*H2-eye(size(H)))
```

符号运算工具箱中也对符号矩阵提供了 `inv()` 重载函数,即使对更高阶的非奇异矩阵也可以精确求解出矩阵的逆矩阵来。下面的语句可以求出 7×7 的 Hilbert 逆矩阵

```
>> H=sym(hilb(7)); H1=inv(H)
```

得出的逆矩阵为

$$H_1 = \begin{bmatrix} 49 & -1176 & 8820 & -29400 & 48510 & -38808 & 12012 \\ -1176 & 37632 & -317520 & 1128960 & -1940400 & 1596672 & -504504 \\ 8820 & -317520 & 2857680 & -10584000 & 18711000 & -15717240 & 5045040 \\ -29400 & 1128960 & -10584000 & 40320000 & -72765000 & 62092800 & -20180160 \\ 48510 & -1940400 & 18711000 & -72765000 & 133402500 & -115259760 & 37837800 \\ -38808 & 1596672 & -15717240 & 62092800 & -115259760 & 100590336 & -33297264 \\ 12012 & -504504 & 5045040 & -20180160 & 37837800 & -33297264 & 11099088 \end{bmatrix}$$

其实,用符号运算工具箱可以求解出更高阶 Hilbert 矩阵的逆矩阵。例如,求解 30 阶矩阵,可以使用下面的命令,得出精确的结果——误差为零。

```
>> H=sym(hilb(30)); norm(H*inv(H)-eye(size(H)))
```

例 4-21 试对例 4-8 中的奇异矩阵 A 求逆,并观察用数值方法对该矩阵求逆会发生什么现象。

解 首先输入该矩阵,则可以用 `inv()` 函数对其求逆

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; B=inv(A), A*B
```

矩阵求逆将得出下面的警告信息“Warning: Matrix is close to singular or badly scaled”,说明矩阵趋于奇异(事实上, A 矩阵就是一个奇异矩阵,但通过数值算法后得出的是接近奇异的结论),并提示得出的逆矩阵可能是不正确的。

由上面的语句求出的“逆矩阵” B 及 AB 分别为

$$B = \begin{bmatrix} 0.9382 & 2.8147 & -2.8147 & -0.9382 \\ 2.8147 & 8.4442 & -8.4442 & -2.8147 \\ -2.8147 & -8.4442 & 8.4442 & 2.8147 \\ -0.9382 & -2.8147 & 2.8147 & 0.9382 \end{bmatrix} \times 10^{14}, AB = \begin{bmatrix} 1 & 0 & -1 & -0.25 \\ -0.25 & 0 & 0 & 0.875 \\ 0.25 & 0.5 & 0 & 0.25 \\ 0.1563 & 0.125 & 0 & 1.7344 \end{bmatrix}$$

如果对上述的结果进行验算,会发现误差很大,所以逆矩阵是错误的。

事实上,奇异矩阵根本不存在一个相应的逆矩阵,能满足式(4-2-13)中的条件。对这里给出的问题还可以试用下面的语句求解,但由于矩阵奇异,故 `inv()` 函数也无能为力。下面语句将给出确切的错误信息“Error, (in inverse) singular matrix”,明确指出原矩阵奇异,不存在逆矩阵。

```
>> A=sym(A); inv(A)
```

例 4-22 MATLAB 的矩阵求逆函数同样适用于含有变量的矩阵。例如,对于下面的 Hankel 矩阵,可以直接用 `inv()` 函数得出其逆矩阵

```
>> syms a1 a2 a3 a4; H=hankelsym([a1 a2 a3 a4]); inv(H)
```

可以直接得出下面的逆矩阵表示

$$\mathbf{H}^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1/a_4 \\ 0 & 0 & 1/a_4 & -1/a_4^2 a_3 \\ 0 & 1/a_4 & -1/a_4^2 a_3 & -1/a_4^3 (a_2 a_4 - a_3^2) \\ 1/a_4 & -1/a_4^2 a_3 & -1/a_4^3 (a_2 a_4 - a_3^2) & -(a_1 a_4^2 - 2a_2 a_3 a_4 + a_3^3)/a_4^4 \end{bmatrix}$$

在经典线性代数教材中,通常采用基本行变换的方式求解矩阵的逆,例如在 \mathbf{H} 矩阵的右侧补一个单位矩阵,然后通过基本行变换将新矩阵左侧变换成单位矩阵,这样新矩阵的右侧自然就是逆矩阵了。在 MATLAB 下提供了 $\mathbf{H}_1 = \text{rref}(\mathbf{H})$ 函数直接求取 \mathbf{H} 矩阵的基本行变换矩阵 \mathbf{H}_1 ,其中 \mathbf{H} 既可以为数值矩阵也可以为符号矩阵。

例 4-23 下面的语句可以通过基本行变换的方法重新求前例矩阵的逆矩阵

```
>> syms a1 a2 a3 a4; H=hankelsym([a1 a2 a3 a4]); inv(H)
H1=[H eye(4)]; H2=rref(H1), H3=H2(:,5:8)
```

得出的 \mathbf{H}_3 与前面的完全一致,中间变量 \mathbf{H}_2 的左侧为单位矩阵。右侧为得出的逆矩阵 \mathbf{H}_3

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1/a_4 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1/a_4 & -1/a_4^2 a_3 \\ 0 & 0 & 1 & 0 & 0 & 1/a_4 & -1/a_4^2 a_3 & -1/a_4^3 (a_2 a_4 - a_3^2) \\ 0 & 0 & 0 & 1 & 1/a_4 & -1/a_4^2 a_3 & -1/a_4^3 (a_2 a_4 - a_3^2) & -(a_1 a_4^2 - 2a_2 a_3 a_4 + a_3^3)/a_4^4 \end{bmatrix}$$

2. 矩阵的广义逆

前面已经介绍过,即使用解析解求解的符号运算工具箱对奇异矩阵的求逆也是无能为力的,因为其逆矩阵根本不存在。另外,长方形的矩阵有时也会涉及求逆的问题,这样就需要定义一种新的“逆矩阵”。对于要研究的矩阵 \mathbf{A} ,如果存在一个矩阵 \mathbf{N} ,它满足

$$\mathbf{A}\mathbf{N}\mathbf{A} = \mathbf{A} \quad (4-2-14)$$

则 \mathbf{N} 矩阵称为 \mathbf{A} 的广义逆矩阵,记作 $\mathbf{N} = \mathbf{A}^-$ 。如果 \mathbf{A} 矩阵是一个 $n \times m$ 的长方形矩阵,则 \mathbf{N} 矩阵为 $m \times n$ 阶矩阵。满足这一条件的广义逆矩阵有无穷多个。

定义下面的范数最小化指标为

$$\min_M \|\mathbf{A}\mathbf{M} - \mathbf{I}\| \quad (4-2-15)$$

则可以证明,对于给定的矩阵 \mathbf{A} ,存在一个唯一的矩阵 \mathbf{M} 使得下面的 3 个条件同时成立:

- (1) $\mathbf{A}\mathbf{M}\mathbf{A} = \mathbf{A}$;
- (2) $\mathbf{M}\mathbf{A}\mathbf{M} = \mathbf{M}$;
- (3) $\mathbf{A}\mathbf{M}$ 与 $\mathbf{M}\mathbf{A}$ 均为 Hermite 对称矩阵。

这样的矩阵 \mathbf{M} 称为矩阵 \mathbf{A} 的 Moore-Penrose 广义逆矩阵,或伪逆,记作 $\mathbf{M} = \mathbf{A}^+$ 。从上面的 3 个条件中可以看出,第一个条件和一般广义逆的定义也是一样的,所不同的是它还要求满足第二个和第三个条件,这样就会得出唯一的广义逆矩阵 \mathbf{M} 了。

MATLAB 提供了求取矩阵 Moore-Penrose 广义逆的函数 `pinv()`,其格式为

$\mathbf{M} = \text{pinv}(\mathbf{A}, \epsilon)$, % 按指定精度 ϵ 求解 Moore-Penrose 广义逆矩阵

其中, ϵ 为判 0 用误差限,如果省略此参数,则判 0 用误差限选用机器的精度 `eps`,这时将返

回 A 的 Moore–Penrose 广义逆矩阵 M 。如果 A 矩阵为非奇异方阵, 则该函数得出的结果就是矩阵的逆阵, 但这样求解的速度将明显慢于 `inv()` 函数。

例 4-24 考虑例 4-8 中给出的奇异矩阵 A , 例 4-21 中用符号运算工具箱中 `inv()` 函数仍不能获得问题的解析解, 因为解析解不存在。所以这里将考虑 Moore–Penrose 广义逆矩阵的求解。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; B=pinv(A), A*B
```

得出的 B 矩阵和 AB 矩阵分别为

$$B = \begin{bmatrix} 0.1011 & -0.0739 & -0.0614 & 0.0636 \\ -0.0364 & 0.0386 & 0.0261 & 0.0011 \\ 0.0136 & -0.0114 & -0.0239 & 0.0511 \\ -0.0489 & 0.0761 & 0.0886 & -0.0864 \end{bmatrix}, AB = \begin{bmatrix} 0.95 & -0.15 & 0.15 & 0.05 \\ -0.15 & 0.55 & 0.45 & 0.15 \\ 0.15 & 0.45 & 0.55 & -0.15 \\ 0.05 & 0.15 & -0.15 & 0.95 \end{bmatrix}$$

这时 AB 矩阵不再是单位阵了, 因为不存在一个 A^+ 能使它成为单位阵。这样得出的 A^+ 应该能使式 (4-2-15) 中的范数取最小值。现在检验 Moore–Penrose 广义逆的 3 个条件的误差范数均为 10^{-14} 级, 由此验证得出的矩阵确实是 A 的 Moore–Penrose 广义逆矩阵。

```
>> norm(A*B*A-A), norm(B*A*B-B), norm(A*B-(A*B)'), norm(B*A-(B*A'))
```

现在对得出的 B 再求一次 Moore–Penrose 广义逆, 则可看出, $(A^+)^+ = A$ 。

```
>> pinv(B), norm(ans-A)
```

例 4-25 考虑一个给定的长方形矩阵 A , 请对该矩阵进行基本分析, 例如获得矩阵的秩、Moore–Penrose 广义逆等, 并分析得出的广义逆矩阵性质。

$$A = \begin{bmatrix} 6 & 1 & 4 & 2 & 1 \\ 3 & 0 & 1 & 4 & 2 \\ -3 & -2 & -5 & 8 & 4 \end{bmatrix}$$

解 可以给出下面的语句对该矩阵进行分析, 得出矩阵为非满秩矩阵的结论

```
>> A=[6,1,4,2,1; 3,0,1,4,2; -3,-2,-5,8,4]; rank(A)
```

由于 A 矩阵为奇异矩阵, 所以应使用 `pinv()` 函数求取矩阵的 Moore–Penrose 广义逆, 并可以通过下面的检验语句对 Moore–Penrose 广义逆的条件逐一验证, 证实该广义逆矩阵确实满足条件。

```
>> iA=pinv(A) % 非满秩矩阵的广义逆
```

```
norm(A*iA*A-A), norm(iA*A-A'*iA'), norm(iA*A-A'*iA'), norm(A*iA-iA'*A')
```

可以得出矩阵的广义逆为

$$A^+ = \begin{bmatrix} 0.073 & 0.0413 & -0.0221 \\ 0.0108 & 0.002 & -0.0156 \\ 0.0459 & 0.0178 & -0.0385 \\ 0.0327 & 0.0431 & 0.0638 \\ 0.0164 & 0.0215 & 0.0319 \end{bmatrix}, \text{ 且 } \begin{cases} \|A^+AA^+ - A^+\| = 1.0263 \times 10^{-16} \\ \|AA^+A - A\| = 8.1145 \times 10^{-15} \\ \|A^+A - (A)^*(A^+)^*\| = 3.9098 \times 10^{-16} \\ \|AA^+ - (A^+)^*(A)^*\| = 1.6653 \times 10^{-16} \end{cases}$$

4.2.3 矩阵的特征值问题

1. 一般矩阵的特征值与特征向量

对一个矩阵 A 来说, 如果存在一个非零的向量 x , 且有一个标量 λ 满足

$$Ax = \lambda x \quad (4-2-16)$$

则称 λ 为 \mathbf{A} 矩阵的一个特征值, 而 \mathbf{x} 称为对应于特征值 λ 的特征向量。严格说来, \mathbf{x} 应该称为 \mathbf{A} 的右特征向量。如果矩阵 \mathbf{A} 的特征值不包含重复的值, 则对应的各个特征向量为线性无关的, 这样由各个特征向量可以构成一个非奇异的矩阵。如果用它对原始矩阵作相似变换, 则可以得出一个对角矩阵。矩阵的特征值与特征向量由 MATLAB 提供的函数 `eig()` 可以容易地求出。该函数的调用格式为

```
d = eig(A)           % 只求解特征值
[V, D] = eig(A)    % 求解特征值和特征向量
```

其中, \mathbf{d} 为特征值构成的向量, \mathbf{D} 为一个对角矩阵, 其对角线上的元素为矩阵 \mathbf{A} 的特征值, 而每个特征值对应的 \mathbf{V} 矩阵的列为该特征值的特征向量, 该矩阵是一个满秩矩阵。MATLAB 的矩阵特征值矩阵满足 $\mathbf{AV} = \mathbf{VD}$, 且每个特征向量各元素的平方和 (即 2 范数) 均为 1。如果调用该函数时只给出一个返回变量, 则将只返回矩阵 \mathbf{A} 的特征值。即使 \mathbf{A} 为复数矩阵, 也照样可以由 `eig()` 函数得出其特征值与特征向量矩阵。

前面介绍的矩阵特征多项式的根和特征值是同样的概念, 所以若精确已知矩阵的特征多项式系数, 则可以调用 `roots()` 函数来计算矩阵的特征值。

矩阵特征值的求解算法是多种多样的, 最常用的有求解实对称矩阵特征值与特征向量的 Jacobi 算法, 有原点平移 QR 分解法与两步 QR 算法。矩阵的特征值与特征向量的求解有许多标准子程序或程序库可以直接调用, 如著名的 EISPACK 软件包^[3,4]等。MATLAB 中的 `eig()` 函数是基于两步 QR 算法实现的, 该函数也同样可以求解复数矩阵的特征值与特征向量矩阵。当矩阵含有重特征值时, 特征向量矩阵可能趋于奇异, 所以在使用此函数时应该注意。

例 4-26 求出例 4-8 中给出的矩阵 \mathbf{A} 的特征值与特征向量矩阵。

解 可以调用 `eig()` 函数直接获得矩阵 \mathbf{A} 的特征值为 34, ± 8.9443 , -2.2348×10^{-15} 。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; eig(A), [v,d]=eig(A)
```

可以得出特征向量矩阵和特征值矩阵为

$$\mathbf{v} = \begin{bmatrix} -0.5 & -0.8236 & 0.3764 & -0.2236 \\ -0.5 & 0.4236 & 0.0236 & -0.6708 \\ -0.5 & 0.0236 & 0.4236 & 0.6708 \\ -0.5 & 0.3764 & -0.8236 & 0.2236 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 34 & 0 & 0 & 0 \\ 0 & 8.9443 & 0 & 0 \\ 0 & 0 & -8.9443 & 0 \\ 0 & 0 & 0 & -2.2348 \times 10^{-15} \end{bmatrix}$$

符号运算工具箱中也提供了 `eig()` 函数, 理论上可以求解任意高阶矩阵的精确特征值, 对于给定的 \mathbf{A} 矩阵, 可以由下面的命令求出特征值的精确解为 0, 34, $\pm 4\sqrt{5}$ 。

```
>> eig(sym(A)), vpa(ans,70), [v,d]=eig(sym(A))
```

得出的相应矩阵如下

$$\mathbf{v} = \begin{bmatrix} -1 & 1 & 12\sqrt{5}/31 - 41/31 & -12\sqrt{5}/31 - 41/31 \\ -3 & 1 & 17/31 - 8\sqrt{5}/31 & 8\sqrt{5}/31 + 17/31 \\ 3 & 1 & -4\sqrt{5}/31 - 7/31 & 4\sqrt{5}/31 - 7/31 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 34 & 0 & 0 \\ 0 & 0 & -4\sqrt{5} & 0 \\ 0 & 0 & 0 & 4\sqrt{5} \end{bmatrix}$$

可见, 在前面的例子中两次调用了 `eig()` 函数, 但由于返回参数个数不一致, 所以后面的调用返回矩阵 \mathbf{A} 的特征值与特征向量, 而前面的调用只返回了矩阵 \mathbf{A} 的特征值而不返

回特征向量矩阵。另外,返回特征值的格式也因返回变量个数不同而不同。

如果一个矩阵包含重特征值,则理论上矩阵 \mathbf{V} 将为奇异矩阵。但因为 MATLAB 数值运算出现的误差,不一定能精确计算出矩阵的重根,这样将得出接近奇异的 \mathbf{V} 矩阵。

若想由 C 或 FORTRAN 语句从最底层编程,则需要编写相当长的程序才可以完成实矩阵特征值和特征向量的计算,例如,EISPACK^[4] 中提供的子程序源程序有 500 多条。

2. 矩阵的广义特征向量问题

若某矩阵 \mathbf{A} 含有重特征值,则必定会使得特征向量矩阵为奇异矩阵,这会约束特征向量矩阵的应用。为了保证特征向量矩阵非奇异,需要引入广义特征向量的问题。假设存在一个标量 λ 和一个非零向量 \mathbf{x} ,使得

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x} \quad (4-2-17)$$

成立,其中 \mathbf{B} 矩阵为对称正定矩阵,则 λ 称为广义特征值,而 \mathbf{x} 向量称为广义特征向量。MATLAB 还提供了求取广义特征值的方法。事实上,普通的矩阵特征值问题可以看成是广义特征值问题的一个特例,因为若假定 $\mathbf{B} = \mathbf{I}$ 为单位阵,则式(4-2-17)中的形式可以直接转化成普通矩阵特征值问题。

如果 \mathbf{B} 矩阵为一个非奇异方阵,则上面的方程可以容易地转换成一般矩阵 $\mathbf{B}^{-1}\mathbf{A}$ 的特征值问题

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \quad (4-2-18)$$

即 λ 和 \mathbf{x} 分别为 $\mathbf{B}^{-1}\mathbf{A}$ 矩阵的特征值和特征向量。但一般情况下不能随便假设 \mathbf{B} 阵为非奇异的方阵,所以文献[5]中给出了广义特征值问题的 QZ 算法。在 MATLAB 中给出的 eig() 函数可以直接用来求取矩阵的广义特征值和特征向量,这时的调用格式为

```
d = eig(A,B)           % 求解广义特征值
[V,D] = eig(A,B)       % 求解广义特征值和特征向量
```

这一函数可以直接得出矩阵的广义特征值向量 \mathbf{d} ,也可以返回一个特征向量矩阵 \mathbf{V} 及一个对角型特征值矩阵 \mathbf{D} ,满足 $\mathbf{A}\mathbf{V} = \mathbf{B}\mathbf{V}\mathbf{D}$ 。值得指出的是,该函数可以求解 \mathbf{B} 矩阵为奇异矩阵时的广义特征值问题。

例 4-27 假设给出如下的矩阵,试求出 \mathbf{A}, \mathbf{B} 矩阵的广义特征值与特征向量矩阵

$$\mathbf{A} = \begin{bmatrix} -4 & -6 & -4 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 6 & -1 & -2 \\ 5 & -1 & 2 & 3 \\ -3 & -4 & 1 & 10 \\ 5 & -2 & -3 & 8 \end{bmatrix}$$

解 原矩阵 \mathbf{A} 理论上有重特征值 -1 ,但数值求解一般得不出精确的特征值。使用下列命令可以求出矩阵 (\mathbf{A}, \mathbf{B}) 的广义特征值和特征向量

```
>> B=[2,6,-1,-2; 5,-1,2,3; -3,-4,1,10; 5,-2,-3,8];
```

```
A=[-4,-6,-4,-1; 1,0,0,0; 0,1,0,0; 0,0,1,0]; [V,D]=eig(A,B), norm(A*V-B*V*D)
```

得出的特征值、特征向量矩阵如下,误差矩阵的范数为 6.3931×10^{-15} 。

$$V = \begin{bmatrix} 0.0268 & -1 & -0.2413 & 0.0269 \\ -1 & 0.7697 & -0.6931 & 0.0997 \\ -0.1252 & -0.3666 & 1 & 0.0796 \\ -0.3015 & 0.4428 & -0.1635 & -1 \end{bmatrix}, \quad D = \begin{bmatrix} -1.2830 & & & \\ & 0.1933 & & \\ & & -0.2422 & \\ & & & -0.0096 \end{bmatrix}$$

4.3 矩阵的基本变换与分解

4.3.1 矩阵的相似变换与正交矩阵

对某方阵 A 来说,如果存在一个非奇异的 B 矩阵,则可以通过下面的方式对原 A 矩阵进行变换

$$X = B^{-1}AB \quad (4-3-1)$$

这样的变换称为相似变换,而 B 称为相似变换矩阵。相似变换后, X 矩阵的秩、迹、行列式和特征值等均不发生变化,其值和 A 矩阵完全一致。通过适当选择变换矩阵 B ,就能有目的地将任意给定的 A 矩阵相似变换成特殊的矩阵表示形式,而不改变原来 A 的重要性质。

对于一类特殊的相似变换矩阵 T ,如果它本身满足 $T^{-1} = T^*$,其中 T^* 为 T 的 Hermite 共轭转置矩阵,则称 T 为正交矩阵,并将之记为 $Q = T$ 。可见,正交矩阵 Q 满足条件

$$Q^*Q = I, \text{ 且 } QQ^* = I \quad (4-3-2)$$

其中, I 为 $n \times n$ 的单位阵。

MATLAB 中提供了求取正交矩阵的函数 `orth()`,其调用格式为 $Q = \text{orth}(A)$,该函数能求出 A 矩阵的正交基矩阵 Q 。若 A 为非奇异矩阵,则得出的正交基矩阵 Q 满足式(4-3-2)的条件。若 A 为奇异矩阵,则得出的矩阵 Q 的列数即为 A 矩阵的秩,且满足 $Q^*Q = I$,而不满足 $QQ^* = I$ 。

例 4-28 求出 $A = \begin{bmatrix} 5 & 9 & 8 & 3 \\ 0 & 3 & 2 & 4 \\ 2 & 3 & 5 & 9 \\ 3 & 4 & 5 & 8 \end{bmatrix}$ 矩阵的正交矩阵。

解 矩阵的正交矩阵可以用 `orth()` 函数直接得出,并可以验证满足正交矩阵的性质

```
>> A=[5,9,8,3; 0,3,2,4; 2,3,5,9; 3,4,5,8];
Q=orth(A), norm(Q'*Q-eye(4)), norm(Q*Q'-eye(4))
```

得出的正交矩阵如下,误差矩阵的范数分别为 $\|Q^*Q - I\| = 4.6395 \times 10^{-16}$, $\|QQ^* - I\| = 4.9270 \times 10^{-16}$ 。

$$Q = \begin{bmatrix} -0.6197 & 0.7738 & -0.0262 & -0.1286 \\ -0.2548 & -0.1551 & 0.949 & 0.1017 \\ -0.5198 & -0.5298 & -0.1563 & -0.6517 \\ -0.53 & -0.3106 & -0.2725 & 0.7406 \end{bmatrix}$$

例 4-29 重新考虑例 4-8 中给出的奇异矩阵 A ,试求出其正交基矩阵,并验证其正交性质。

解 可以通过下面的 MATLAB 语句求取并检验其正交基矩阵。注意,因为 A 为奇异矩阵,故得出的 Q 为 4×3 长方形矩阵。


```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1];
```

```
Q=orth(A), norm(Q'*Q-eye(3))
```

奇异矩阵可以如下得出, 并可以检验出误差矩阵的范数为 $\|Q^*Q - I\| = 1.0140 \times 10^{-15}$ 。

$$Q = \begin{bmatrix} -0.5 & 0.6708 & 0.5 \\ -0.5 & -0.2236 & -0.5 \\ -0.5 & 0.2236 & -0.5 \\ -0.5 & -0.6708 & 0.5 \end{bmatrix}$$

4.3.2 矩阵的三角分解和 Cholesky 分解

1. 一般矩阵的三角分解

矩阵的三角分解又称为 LU 分解, 它的目的是将一个矩阵分解成一个下三角矩阵 L 和一个上三角矩阵 U 的乘积, 亦即 $A = LU$, 其中 L 和 U 矩阵可以分别写成

$$L = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix} \quad (4-3-3)$$

这样产生的矩阵与原来的 A 矩阵的关系可以写成

$$\begin{aligned} a_{11} &= u_{11}, & a_{12} &= u_{12}, & \cdots & a_{1n} &= u_{1n} \\ a_{21} &= l_{21}u_{11}, & a_{22} &= l_{21}u_{12} + u_{22}, & \cdots & a_{2n} &= l_{21}u_{1n} + u_{2n} \\ \vdots & & \vdots & & \ddots & \vdots & \\ a_{n1} &= l_{n1}u_{11}, & a_{n2} &= l_{n1}u_{12} + l_{n2}u_{22}, & \cdots & a_{nn} &= \sum_{k=1}^{n-1} l_{nk}u_{kn} + u_{nn} \end{aligned} \quad (4-3-4)$$

由式 (4-3-4) 可以立即得出求取 l_{ij} 和 u_{ij} 的递推计算公式

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}}{u_{jj}}, \quad (j < i), \quad u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}, \quad (j \geq i) \quad (4-3-5)$$

该公式的递推初值为

$$u_{1i} = a_{1i}, \quad i = 1, 2, \cdots, n \quad (4-3-6)$$

注意, 在上述算法中并未对主元素进行任何选取, 因此该算法并不一定数值稳定, 因为在运算过程中 0 或很小的数值可能被用作除数。在 MATLAB 中也给出了基于主元素的矩阵 LU 分解函数 `lu()`, 该函数的调用格式为

```
[L,U] = lu(A)      % LU 分解, A = LU
[L,U,P] = lu(A)    % P 为置换矩阵, A = P-1LU
```

其中, L 、 U 分别为变换后的下三角和上三角矩阵。在 MATLAB 的 `lu()` 函数中考虑了主元素选取的问题, 所以该函数一般会给出可靠的结果。由该函数得出的下三角矩阵 L 并不一定是一个真正的下三角矩阵, 因为选取它可能进行了一些元素行的交换, 这样主对角线的元素可能不是 1, 而在矩阵 L 内存在一个唯一的如式 (4-2-1) 中定义的置换, 其各个元素的值均是 1。如果想获得有关换行信息, 则可以由后一种格式调用 `lu()` 函数, 这时 P 为单位阵变换出的置换矩阵, A 矩阵可以分解成 $A = P^{-1}LU$ 。

由上述算法还可以编写出 LU 分解的解析解程序, 该程序不涉及主元素处理, 因此得出的是真正的上、下三角矩阵

```
function [L,U]=lusym(A)
n=length(A); U=sym(zeros(size(A))); L=sym(eye(size(A)));
U(1,:)=A(1,:); L(:,1)=A(:,1)/U(1,1);
for i=2:n,
    for j=2:i-1, L(i,j)=(A(i,j)-L(i,1:j-1)*U(1:j-1,j))/U(j,j); end
    for j=i:n, U(i,j)=A(i,j)-L(i,1:i-1)*U(1:i-1,j); end
end
```

例 4-30 再考虑例 4-8 中矩阵的 LU 分解问题。分别用两种方法调用 MATLAB 中的 `lu()` 函数, 则可以得出不同的结果。

解 先输入 A 矩阵, 并求出三角分解矩阵

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; [L1,U1]=lu(A)
```

得出的分解矩阵分别为

$$L_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.3125 & 0.7685 & 1 & 0 \\ 0.5625 & 0.4352 & 1 & 1 \\ 0.25 & 1 & 0 & 0 \end{bmatrix}, \quad U_1 = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 0 & 13.5 & 14.25 & -2.25 \\ 0 & 0 & -1.8889 & 5.6667 \\ 0 & 0 & 0 & 3.55 \times 10^{-15} \end{bmatrix}$$

可见, 这样得出的 L_1 矩阵并非下三角矩阵, 这是因为在分解过程中采用了主元素交换的方法。现在考虑 `lu()` 函数的另一种调用方法

```
>> [L,U,P]=lu(A)
```

这样可以得出新的分解矩阵为

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.25 & 1 & 0 & 0 \\ 0.3125 & 0.7685 & 1 & 0 \\ 0.5625 & 0.4352 & 1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 0 & 13.5 & 14.25 & -2.25 \\ 0 & 0 & -1.8889 & 5.6667 \\ 0 & 0 & 0 & 3.55 \times 10^{-15} \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

注意, 这里得出的 P 矩阵不是一个单位矩阵, 而是单位矩阵的置换矩阵。结合得出的 L_1 矩阵可以看出, P 矩阵的 $p_{2,4} = 1$, 表明需要将 L_1 矩阵的第 4 行换到第 2 行, $p_{3,2} = p_{4,3} = 1$ 表明需要将 L_1 的第 2 行换至第 3 行, 将原第 3 行换至第 4 行, 这样就可以得出一个真正的下三角矩阵 L 了。将 P 、 L 、 U 代入并检验, 即由 `inv(P)*L*U` 命令可以精确地还原 A 矩阵。

采用解析解函数, 则可以对原矩阵重新进行三角分解

```
>> [L2,U2]=lusym(A)
```

这样,可以分别得出如下的三角分解矩阵为

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 5/16 & 1 & 0 & 0 \\ 9/16 & 47/83 & 1 & 0 \\ 1/4 & 108/83 & -3 & 1 \end{bmatrix}, U_2 = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 0 & 83/8 & 145/16 & 63/16 \\ 0 & 0 & -68/83 & 204/83 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

2. 对称矩阵的三角分解 —— Cholesky 分解

如果 A 为对称矩阵,利用对称矩阵的特点则可以用 LU 分解的方法对之进行分解,这样可以将原来矩阵 A 分解成

$$A = LL^T = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix} \quad (4-3-7)$$

如果利用对称矩阵的性质,则 LU 分解可以简化成

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \quad l_{ji} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right), \quad j < i \quad (4-3-8)$$

且初始条件为 $l_{11} = \sqrt{a_{11}}, l_{j1} = a_{j1}/l_{11}$ 。该算法又称为对称矩阵的 Cholesky 分解算法。

MATLAB 提供了 `chol()` 函数来求取矩阵的 Cholesky 分解矩阵 D , 其结果为一个下三角矩阵。该函数的调用格式为 $D = \text{chol}(A)$, 其中 $D = L^T$ 。由上述算法还可以编写出 Cholesky 分解的解析解求解函数 `cholsym()`, 该函数返回 D 矩阵, 即使原矩阵不是正定的, 由此函数仍可求解。

```
function D=cholsym(A)
n=length(A); D(1,1)=sqrt(A(1,1)); D(1,2:n)=A(2:n,1)/D(1,1);
for i=2:n, k=1:i-1; D(i,i)=sqrt(A(i,i)-sum(D(k,i).^2));
    for j=i+1:n, D(i,j)=(A(j,i)-sum(D(k,j).*D(k,i)))/D(i,i); end
end
```

例 4-31 试求出对称的 4 阶 $A = \begin{bmatrix} 9 & 3 & 4 & 2 \\ 3 & 6 & 0 & 7 \\ 4 & 0 & 6 & 0 \\ 2 & 7 & 0 & 9 \end{bmatrix}$ 矩阵的 Cholesky 分解。

解 用下面的语句可以对 A 进行 Cholesky 分解, 得出 D 矩阵

```
>> A=[9,3,4,2; 3,6,0,7; 4,0,6,0; 2,7,0,9]; D=chol(A), D1=cholsym(sym(A))
```

可以由解析法和数值法分别得出分解矩阵为

$$D = \begin{bmatrix} 3 & 1 & 1.3333 & 0.6667 \\ 0 & 2.2361 & -0.5963 & 2.8324 \\ 0 & 0 & 1.9664 & 0.4068 \\ 0 & 0 & 0 & 0.6065 \end{bmatrix}, D_1 = \begin{bmatrix} 3 & 1 & 4/3 & 2/3 \\ 0 & \sqrt{5} & -4\sqrt{5}/15 & 19\sqrt{5}/15 \\ 0 & 0 & \sqrt{870}/15 & 2\sqrt{870}/145 \\ 0 & 0 & 0 & 4\sqrt{174}/87 \end{bmatrix}$$

3. 正定、正规矩阵的定义与判定

正定矩阵是在对称矩阵基础上建立起来的概念。在介绍该概念之前,先给出主子行列式定义。假设对称矩阵 A 可以写成

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{12} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{13} & a_{23} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & a_{3n} & \cdots & a_{nn} \end{bmatrix} \quad (4-3-9)$$

则左上角的各个子矩阵的行列式称为主子行列式。如果一个对称矩阵所有的主子行列式符号相同,则称该矩阵为正定矩阵。

相应地,可以引入对称矩阵的半正定矩阵的概念,如果主子行列式均为非负的数值,则称为半正定矩阵。MATLAB 的函数 `chol()` 还可以用来判定矩阵的正定性。该函数的另一种调用格式为 `[D,p]=chol(A)`,式中对正定的 A 矩阵来说,返回的 $p=0$ 。所以可以利用这个性质来判定一个对称矩阵是否为正定矩阵。对非正定矩阵,则返回一个正的 p 值, $p-1$ 为 A 矩阵中正定的子矩阵的阶次,亦即 D 将为 $(p-1)$ 阶方阵。

如果复数方阵满足

$$A^*A = AA^* \quad (4-3-10)$$

则其中 A^* 为 A 的 Hermite 转置,亦即共轭转置,该矩阵称为正规矩阵。判定正规矩阵由定义直接判定 `norm(A'*A-A*A')<epsilon`,如果得出的结果为 1,则 A 为正规矩阵。

例 4-32 试判定对称矩阵 $A = \begin{bmatrix} 7 & 5 & 5 & 8 \\ 5 & 6 & 9 & 7 \\ 5 & 9 & 9 & 0 \\ 8 & 7 & 0 & 1 \end{bmatrix}$ 是否为正定矩阵,并对其进行 Cholesky 分解。

解 用下面的语句可以对 A 矩阵进行分解,得出 D 矩阵,并求出正定的阶次为 2,从而说明原矩阵并非正定矩阵,因为 $p \neq 0$ 。

```
>> A=[7,5,5,8; 5,6,9,7; 5,9,9,0; 8,7,0,1]; [D,p]=chol(A), D1=cholsym(sym(A))
```

这样,矩阵的正定子矩阵 D 如下,其中 $p=3 \neq 0$,解析三角矩阵 D_1 也可得出,其第 3、4 行含有虚部,说明正定子矩阵为 2×2 矩阵,与前面得出的结果一致。

$$D = \begin{bmatrix} 2.6458 & 1.8898 \\ 0 & 1.5584 \end{bmatrix}, D_1 = \begin{bmatrix} \sqrt{7} & 5\sqrt{7}/7 & 5\sqrt{7}/7 & 8\sqrt{7}/7 \\ 0 & \sqrt{119}/7 & 38\sqrt{119}/119 & 9\sqrt{119}/119 \\ 0 & 0 & \sqrt{1938}j/17 & 73\sqrt{1938}j/969 \\ 0 & 0 & 0 & 2\sqrt{1767}/57 \end{bmatrix}$$

非对称矩阵也可以调用 `chol()` 函数,但结果是错误的,它将首先将给定的矩阵强制按上三角子矩阵转换成对称矩阵。在严格的数学意义下,非对称矩阵是没有 Cholesky 分解的。

4.3.3 矩阵的相伴变换、对角变换和 Jordan 变换

1. 一般矩阵变换成相伴矩阵

对给定矩阵 A , 如果存在一个列向量 x , 使得矩阵 $T = [x, Ax, \dots, A^{n-1}x]$ 为非奇异, 则矩阵 A 可以通过线性相似变换的方式变换成类似相伴矩阵的形式。由此看来, 能够进行这样变换的矩阵有无穷多个。若想得出式 (4-1-7) 中定义的相伴矩阵, 则还需要一个左右翻转的单位矩阵, 下面通过例子演示这样的变换方法。

例 4-33 试将例 4-27 中的矩阵变换成相伴矩阵。

解 可以随机生成一个列向量 x , 判定生成的 T 矩阵是否为非奇异, 如果奇异则重新生成随机列向量。得到非奇异 T 矩阵后, 通过线性相似变换可以对原矩阵进行处理

```
>> A=[5,7,6,5; 7,10,8,7; 6,8,10,9; 5,7,9,10];
while(1), x=floor(2*rand(4,1)); T=sym([x A*x A^2*x A^3*x]);
if rank(T)==4, break; end, end, T, A1=inv(T)*A*T
```

上述语句可以得出

$$T = \begin{bmatrix} 1 & 11 & 326 & 9853 \\ 0 & 15 & 453 & 13696 \\ 1 & 16 & 472 & 14296 \\ 0 & 14 & 444 & 13489 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 100 \\ 0 & 1 & 0 & -146 \\ 0 & 0 & 1 & 35 \end{bmatrix}$$

可见, 这样得出的 T 矩阵不唯一。得出的矩阵 A_1 类似于式 (4-1-7) 中定义的相伴矩阵。下面的语句可以确实将原矩阵变换成相伴矩阵

```
>> T1=inv(T*fliplr(eye(4)))', A2=inv(T1)*A*T1
```

这样, 变换矩阵和得出的相伴矩阵分别为

$$T = \frac{1}{14053} \begin{bmatrix} -318 & 10591 & -29493 & 19064 \\ -176 & 5243 & 3298 & -11368 \\ 318 & -10591 & 29493 & -5011 \\ 75 & -1835 & -13063 & 2928 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 35 & -146 & 100 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2. 矩阵的对角化

如果矩阵 A 的特征值互异, 则特征向量矩阵 T 为非奇异矩阵, 这样, 选择该矩阵即可将原矩阵变换成对角矩阵。由于 MATLAB 可以同等处理复数矩阵, 所以含有复数特征值的矩阵能得出复数的对角矩阵和复数相似变换矩阵。

例 4-34 试求出矩阵 $A = \begin{bmatrix} 3 & 2 & 2 & 2 \\ 1 & 2 & -2 & -2 \\ -1 & -2 & 0 & -2 \\ 0 & 1 & 3 & 5 \end{bmatrix}$ 的对角矩阵及变换矩阵。

解 可以由下面的语句得出矩阵的特征值为 1, 2, 3, 4, 因为它们互不相同, 变换矩阵即其特征向量矩阵, 所以问题可以由下面的语句直接求解

```
>> A=[3,2,2,2; 1,2,-2,-2; -1,-2,0,-2; 0,1,3,5];
[v,d]=eig(sym(A)); A1=inv(v)*A*v
```

变换矩阵和对角矩阵分别为

$$v = \begin{bmatrix} 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & -1 \\ -1 & -1 & 1 & 0 \\ 1 & 1 & -2 & 1 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

例 4-35 试求出下面含有复数特征值的矩阵 A 的对角矩阵变换

$$A = \begin{bmatrix} 1 & 0 & 4 & 0 \\ 0 & -3 & 0 & 0 \\ -2 & 2 & -3 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$

解 复数特征值矩阵的 Jordan 标准型及广义特征向量矩阵问题也可以由 `jordan()` 函数求取。可以用下面的语句得出相应的特征向量矩阵与 Jordan 矩阵为

```
>> A=[1,0,4,0; 0,-3,0,0; -2,2,-3,0; 0,0,0,-2]; [V,D]=eig(sym(A))
```

得出的分解矩阵分别为

$$V = \begin{bmatrix} -1 & 0 & -1+j & -1-j \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad J = \begin{bmatrix} -3 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -1-2j & 0 \\ 0 & 0 & 0 & -1+2j \end{bmatrix}$$

如果用户需要得到实数矩阵而不是复数矩阵,则可以编写出下面的转换函数

```
function V1=realjordan(V)
n=length(V); i=0; vr=real(V); vi=imag(V); n1=n;
while(i<n1), i=i+1; V1(:,i)=vr(:,i), vv=vi(:,i);
    if any(vv~=0), for j=i+1:n, if all(vi(:,j)+vv==0), V1(:,j)=vv; n1=n1-1;
    end, end, end, end
```

这样可以将变换矩阵变成实数矩阵,从而使得 Jordan 矩阵变换成实数矩阵

```
>> V=realjordan(V), J=inv(V)*A*V
```

可以得出改进的 Jordan 矩阵及其变换矩阵为

$$J_1 = \begin{bmatrix} -3 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 2 & -1 \end{bmatrix}, \quad V = \begin{bmatrix} -1 & 0 & -1 & 1 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

3. 矩阵的 Jordan 变换

对含有重特征值的矩阵 A 通常不能直接分解成对角矩阵,而用纯特征值求解方法必定能使矩阵的特征向量矩阵 V 含有若干相同的列,使得该矩阵为奇异矩阵。

例 4-36 分别用数值、解析方法求 $A = \begin{bmatrix} -71 & -65 & -81 & -46 \\ 75 & 89 & 117 & 50 \\ 0 & 4 & 8 & 4 \\ -67 & -121 & -173 & -58 \end{bmatrix}$ 的特征值、特征向量矩阵。

解 用 MATLAB 语言中的数值算法和解析方法可以求出该矩阵的特征值

```
>> A=[-71,-65,-81,-46; 75,89,117,50; 0,4,8,4; -67,-121,-173,-58];
D=eig(A), [v,d]=eig(sym(A))
```

得出的数值解和解析解分别为

$$D = \begin{bmatrix} -8.0061 \\ -8+j0.0061 \\ -8-j0.0061 \\ -7.9939 \end{bmatrix}, \quad v = \begin{bmatrix} 17/8 \\ -13/8 \\ 1 \\ -19/8 \end{bmatrix}, \quad d = \begin{bmatrix} -8 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -8 \end{bmatrix}$$

该矩阵的特征值为位于 -8 的 4 重根, 所以用数值解方法得出的特征值有很大的误差, 故在这样的的问题上不适合采用数值算法。而解析解方法可以得出精确的解, 故而得出的特征向量矩阵实际上是奇异矩阵, 因为 4 列均相同, 所以只保留了一列。

由于 MATLAB 语言、其他数值运算语言、软件包本身及数值算法的缺陷, 均在计算内核中使用 `double` 型数据结构, 导致对某些有重根矩阵的计算中间过程出现误差, 所以在纯数值运算上将有很大的误差, 即使使用现在最好的数值运算方法, 如果不突破双精度的数据结构也是难以避免的。

为解决这样的问题, 需要使用符号运算工具箱中的 `jordan()` 函数来分解出 Jordan 标准型, 并求出非奇异的广义特征向量矩阵。该函数的调用格式为

```
J = jordan(A)           % 只返回 Jordan 矩阵 J
[V, J] = jordan(A)      % 返回 Jordan 矩阵 J 和广义特征向量矩阵 V
```

有了广义特征向量矩阵 V , 则 Jordan 标准型可以由 $J = V^{-1}AV$ 变换出来。注意, Jordan 矩阵主对角线为矩阵的特征值, 次主对角线为 1。

例 4-37 试对例 4-36 中给出的矩阵进行 Jordan 分解。

解 符号矩阵的 Jordan 分解可以用 `jordan()` 函数直接进行分解, 得出所需的矩阵

```
>> A = [-71, -65, -81, -46; 75, 89, 117, 50; 0, 4, 8, 4; -67, -121, -173, -58];
[V, J] = jordan(sym(A))
```

这样得出的分解矩阵为

$$V = \begin{bmatrix} -18496 & 2176 & -63 & 1 \\ 14144 & -800 & 75 & 0 \\ -8704 & 32 & 0 & 0 \\ 20672 & -1504 & -67 & 0 \end{bmatrix}, \quad J = \begin{bmatrix} -8 & 1 & 0 & 0 \\ 0 & -8 & 1 & 0 \\ 0 & 0 & -8 & 1 \\ 0 & 0 & 0 & -8 \end{bmatrix}.$$

得出的 V 矩阵就是满秩的矩阵, 对它求逆, 就可以实现用普通数值运算难以实现的功能。该问题将在后面矩阵函数的例子中演示。

例 4-38 试得出下面矩阵的 Jordan 标准型和变换矩阵。

$$A = \begin{bmatrix} 0 & -1 & 0 & 0 & -1 & 1 \\ 0.5 & 0 & -0.5 & 0 & -1 & 0.5 \\ -0.5 & 0 & -0.5 & 0 & 0 & 0.5 \\ 468.5 & 452 & 304.5 & 577 & 225 & 360.5 \\ -468 & -450 & -303 & -576 & -223 & -361 \\ -467.5 & -451 & -303.5 & -576 & -223 & -361.5 \end{bmatrix}$$

解 下面语句可以输入 A 矩阵, 并求出矩阵的特征值

```
>> A = [0, -1, 0, 0, -1, 1; 0.5, 0, -0.5, 0, -1, 0.5; -0.5, 0, -0.5, 0, 0, 0.5;
468.5, 452, 304.5, 577, 225, 360.5; -468, -450, -303, -576, -223, -361;
-467.5, -451, -303.5, -576, -223, -361.5];
A = sym(A); eig(A), [v, J] = jordan(A)
```

得出的特征值分别为 $-2, -2, -1 \pm j2, -1 \pm j2$, 即包含 2 重实特征值 -2 , 2 重复特征值 $-1 \pm j2$ 。用下面的语句可以得出 Jordan 矩阵为

$$J = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1+j2 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1+j2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1-j2 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1-j2 \end{bmatrix}$$

而变换矩阵是复数矩阵,显示从略。如下修改变换矩阵

```
>> V=realjordan(v), J=inv(V)*A*V
```

则可以得出新的实变换矩阵与变换后的实 Jordan 块变形矩阵分别为

$$V = \begin{bmatrix} 423/25 & -543/125 & 851/100 & 334/125 & 757/100 & -9321/1000 \\ -423/25 & 7431/250 & 2459/100 & -7431/500 & 663/100 & -509/1000 \\ 423/5 & -471/10 & -757/40 & 471/20 & 851/40 & -1887/80 \\ 4371/25 & -70677/250 & -47327/400 & 70677/500 & -9191/100 & 247587/4000 \\ -4653/25 & 31353/125 & 16263/200 & -31353/250 & 15991/200 & -96843/2000 \\ -5922/25 & 76539/250 & 22507/200 & -76539/500 & 12399/200 & -74767/2000 \end{bmatrix}$$

$$J = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -2 & 0 \\ 0 & 0 & 0 & -1 & 0 & -2 \\ 0 & 0 & 2 & 0 & -1 & 1 \\ 0 & 0 & 0 & 2 & 0 & -1 \end{bmatrix}$$

4.3.4 矩阵的奇异值分解

矩阵的奇异值也可以看成是矩阵的一种测度。对任意的 $n \times m$ 矩阵 A 来说,总有

$$A^T A \geq 0, AA^T \geq 0 \quad (4-3-11)$$

且在理论上

$$\text{rank}(A^T A) = \text{rank}(AA^T) = \text{rank}(A) \quad (4-3-12)$$

进一步可以证明, $A^T A$ 与 AA^T 有相同的非负特征值 λ_i , 在数学上把这些非负的特征值的平方根称作矩阵 A 的奇异值, 记 $\sigma_i(A) = \sqrt{\lambda_i(A^T A)}$ 。

例 4-39 假设矩阵 $A = \begin{bmatrix} 1 & 1 \\ \mu & 0 \\ 0 & \mu \end{bmatrix}$, 其中 $\mu = 5\text{eps}$, 试根据式 (4-3-12) 求取 A 矩阵的秩^[6]。

解 显然, A 矩阵的秩为 2。用 MATLAB 运算也将得出同样的结论

```
>> A=[1 1; 5*eps,0; 0,5*eps]; rank(A)
```

现在考虑用式 (4-3-12) 中给出的方法计算矩阵 A 的秩。利用普通的乘法运算得到 $A^T A$

$$A^T A = \begin{bmatrix} 1+\mu^2 & 1 \\ 1 & 1+\mu^2 \end{bmatrix}$$

在双精度数值运算中, 由于 μ^2 为 10^{-30} 级数值, 所以加到 1 上事实上就已经被忽略了, 这样 $A^T A$ 矩阵将退化成么矩阵, 再求其秩显然为 1, 从而可以断定原矩阵 A 的秩为 1, 这与实际矛盾, 故对这样的问题应该引入一个新的量作为矩阵秩的测度, 即需要引入奇异值的概念。

假设 \mathbf{A} 矩阵为 $n \times m$ 矩阵, 则 \mathbf{A} 矩阵可以分解为

$$\mathbf{A} = \mathbf{L}\mathbf{A}_1\mathbf{M} \quad (4-3-13)$$

其中, \mathbf{L} 和 \mathbf{M} 为正交矩阵, $\mathbf{A}_1 = \text{diag}(\sigma_1, \dots, \sigma_n)$ 为对角矩阵, 其对角元素满足不等式 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ 。如果存在 $\sigma_n = 0$, 则原矩阵 \mathbf{A} 为奇异矩阵, 这时矩阵 \mathbf{A} 的秩应该等于矩阵 \mathbf{A}_1 中非 0 对角元素的个数。

MATLAB 提供了直接求取矩阵奇异值分解的函数 `svd()`, 其调用方式为

```
S = svd(A)           % 只计算矩阵的奇异值
[L, A1, M] = svd(A)  % 计算矩阵奇异值与变换矩阵
```

其中, \mathbf{A} 为原始矩阵, 返回的 \mathbf{A}_1 为对角矩阵, 而 \mathbf{L} 和 \mathbf{M} 均为正交矩阵, 且 $\mathbf{A} = \mathbf{L}\mathbf{A}_1\mathbf{M}^T$ 。

矩阵的奇异值大小通常决定矩阵的性态, 如果矩阵的奇异值变化特别大, 则矩阵中某个元素有一个微小的变化将严重影响到原矩阵的参数, 这样的矩阵又称为病态矩阵或坏条件矩阵, 而在矩阵存在趋于 0 的奇异值时称为奇异矩阵。矩阵最大奇异值 σ_{\max} 和最小奇异值 σ_{\min} 的比值又称为该矩阵的条件数, 记作 $\text{cond}(\mathbf{A})$, 即 $\text{cond}(\mathbf{A}) = \sigma_{\max}/\sigma_{\min}$, 矩阵的条件数越大, 则对元素变化越敏感。矩阵的最大奇异值和最小奇异值还分别记作 $\bar{\sigma}(\mathbf{A})$ 和 $\underline{\sigma}(\mathbf{A})$ 。在 MATLAB 中也提供了函数 `cond(A)` 来求取矩阵 \mathbf{A} 的条件数。

例 4-40 试对例 4-8 中给出的 \mathbf{A} 矩阵进行奇异值分解。

解 如果调用 MATLAB 中给出的矩阵奇异值分解函数 `svd()`, 则可以容易地求出 \mathbf{L} , \mathbf{A}_1 和 \mathbf{M} 矩阵, 并可以容易地求出该矩阵的条件数

```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1]; [L,A1,M]=svd(A)
```

得出的分解矩阵为

$$\mathbf{L} = \begin{bmatrix} -0.5 & 0.6708 & 0.5 & -0.2236 \\ -0.5 & -0.2236 & -0.5 & -0.6708 \\ -0.5 & 0.2236 & -0.5 & 0.6708 \\ -0.5 & -0.6708 & 0.5 & 0.2236 \end{bmatrix}, \quad \mathbf{A}_1 = \begin{bmatrix} 34 & 0 & 0 & 0 \\ 0 & 17.8885 & 0 & 0 \\ 0 & 0 & 4.4721 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} -0.5 & 0.5 & 0.6708 & 0.2236 \\ -0.5 & -0.5 & -0.2236 & 0.6708 \\ -0.5 & -0.5 & 0.2236 & -0.6708 \\ -0.5 & 0.5 & -0.6708 & -0.2236 \end{bmatrix}$$

可见该矩阵含有 0 奇异值, 故原矩阵为奇异矩阵。该矩阵的条件数可以由语句 `cond(A)` 得出, 接近于 ∞ , 但在双精度数值运算上有一定的误差。如果先将 \mathbf{A} 矩阵转换成符号矩阵, 则调用 `svd()` 将得出更精确的奇异值分解矩阵。

例 4-41 对于 $n \neq m$ 的矩阵 $\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$ 来说, 也可以对之作奇异值分解。例如, 可以对上面的长方形矩阵进行奇异值分解, 并检验分解的结果。

解 使用如下命令进行求解

```
>> A=[1,3,5,7; 2,4,6,8]; [L,A1,M]=svd(A), A2=L*A1*M', norm(A-A2)
```

可以得出如下的分解矩阵, 并可得出 $\|\mathbf{L}\mathbf{A}_1\mathbf{V}^T - \mathbf{A}\| = 9.7277 \times 10^{-15}$, $\mathbf{L}\mathbf{A}_1\mathbf{V}^T$ 能恢复 \mathbf{A} 矩阵。

$$L = \begin{bmatrix} -0.6414 & -0.7672 \\ -0.7672 & 0.6414 \end{bmatrix}, A_1 = \begin{bmatrix} 14.2691 & 0 & 0 & 0 \\ 0 & 0.6268 & 0 & 0 \end{bmatrix}$$

$$M = \begin{bmatrix} -0.1525 & 0.8226 & -0.3945 & -0.38 \\ -0.3499 & 0.4214 & 0.2428 & 0.8007 \\ -0.5474 & 0.0201 & 0.6979 & -0.4614 \\ -0.7448 & -0.3812 & -0.5462 & 0.0407 \end{bmatrix}$$

4.4 矩阵方程的计算机求解

4.4.1 线性方程组的计算机求解

考虑下面给出的线性代数方程

$$Ax = B \quad (4-4-1)$$

其中, A 和 B 均为给定矩阵

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix} \quad (4-4-2)$$

可以由给定的 A 和 B 矩阵构造出解的判定矩阵 C

$$C = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_{11} & b_{12} & \cdots & b_{1p} \\ a_{21} & a_{22} & \cdots & a_{2n} & b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix} \quad (4-4-3)$$

这样可以不加证明地给出线性方程组有解的判定定理^[7]:

(1) 当 $m = n$, 且 $\text{rank}(A) = n$ 时, 方程组式 (4-4-1) 有唯一解

$$x = A^{-1}B \quad (4-4-4)$$

用 MATLAB 语言可以立即得出该方程的解为 $x = \text{inv}(A)*B$ 。但是 $\text{inv}()$ 函数的调用也有值得注意之处。例如, 若 A 矩阵为奇异的或接近奇异的, 则利用此函数有可能产生错误的结果。

若采用符号运算工具箱, 则可以直接使用 $\text{inv}()$ 函数, 如果能得出方程的解, 则解是唯一的, 如果出现错误信息, 则再考虑其他的情形。

例 4-42 求解线性代数方程组

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 1 & 3 & 2 & 4 \\ 4 & 1 & 3 & 2 \end{bmatrix} X = \begin{bmatrix} 5 & 1 \\ 4 & 2 \\ 3 & 3 \\ 2 & 4 \end{bmatrix}$$

解 上述方程可以用下面的语句直接求出, 并验证其精度

```
>> A=[1 2 3 4; 4 3 2 1; 1 3 2 4; 4 1 3 2]; B=[5 1; 4 2; 3 3; 2 4];
x=inv(A)*B, e1=norm(A*x-B), x1=inv(sym(A))*B, e2=norm(A*x1-B)
```

这样,数值解、解析解与产生的误差如下给出,可见用解析解方法可以得出没有误差的解。

$$\mathbf{x} = \begin{bmatrix} -1.8 & 2.4 \\ 1.8667 & -1.2667 \\ 3.8667 & -3.2667 \\ -2.1333 & 2.7333 \end{bmatrix}, e_1 = 5.2283 \times 10^{-15}, \mathbf{x}_1 = \begin{bmatrix} -9/5 & 12/5 \\ 28/15 & -19/15 \\ 58/15 & -49/15 \\ -32/15 & 41/15 \end{bmatrix}, e_2 = 0$$

(2) 当 $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{C}) = r < n$ 时,方程组(4-4-1)有无穷多解,可以构造出线性方程组的 $n-r$ 个化零向量 $\mathbf{x}_i, i = 1, 2, \dots, n-r$,原方程组对应的齐次方程组的解 $\hat{\mathbf{x}}$ 可以由 \mathbf{x}_i 的线性组合来表示,即

$$\hat{\mathbf{x}} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \dots + \alpha_{n-r} \mathbf{x}_{n-r} \quad (4-4-5)$$

其中, $\alpha_i, i = 1, 2, \dots, n-r$ 为任意常数。在 MATLAB 语言中可以由 `null()` 直接求出,其调用格式为 `Z = null(sym(A))`,该函数也可以用于数值解问题,其中 \mathbf{Z} 的列数为 $n-r$,而各列构成的向量又称为矩阵 \mathbf{A} 的基础解系。

求解式(4-4-1)中给出的非齐次方程组也是较简单的,只要能求出该方程的任意一个特解 \mathbf{x}_0 ,则原非齐次方程组的解为 $\mathbf{x} = \hat{\mathbf{x}} + \mathbf{x}_0$ 。其实,在 MATLAB 语言中求解该方程的一个特解并非难事,用 `$\mathbf{x}_0 = \text{pinv}(\mathbf{A}) * \mathbf{B}$` 即可求出。

例 4-43 求解线性代数方程组^[8]

$$\begin{bmatrix} 1 & 4 & 0 & -1 & 0 & 7 & -9 \\ 2 & 8 & -1 & 3 & 9 & -13 & 7 \\ 0 & 0 & 2 & -3 & -4 & 12 & -8 \\ -1 & -4 & 2 & 4 & 8 & -31 & 37 \end{bmatrix} \mathbf{X} = \begin{bmatrix} 3 \\ 9 \\ 1 \\ 4 \end{bmatrix}$$

解 用下面的语句可以输入 \mathbf{A} 和 \mathbf{B} 矩阵,并构造出 \mathbf{C} 矩阵,从而判定矩阵方程的可解性

```
>> A=[1,4,0,-1,0,7,-9;2,8,-1,3,9,-13,7;0,0,2,-3,-4,12,-8;-1,-4,2,4,8,-31,37];
B=[3; 9; 1; 4]; C=[A B]; rank(A), rank(C)
```

通过检验秩的方法得出矩阵 \mathbf{A} 和 \mathbf{C} 的秩相同,都等于 3,小于矩阵 \mathbf{A} 的列数 7,由此可以得出结论,原线性代数方程组有无穷多组解。如需求解原代数方程组,可以先求出化零空间 \mathbf{Z} ,并得出满足方程的一个特解 \mathbf{x}_0

```
>> Z=null(sym(A)), x0=sym(pinv(A)*B)
syms a1 a2 a3 a4; x=Z*[a1;a2;a3;a4]+x0, E=A*x-B
```

这样可以先得出基础解系 \mathbf{Z} 及一个特解 \mathbf{x}_0 。对任选的 a_1 和 a_2 ,可以构造出原线性代数方程全部的解析解如下,得出的误差矩阵为零矩阵

$$\mathbf{Z} = \begin{bmatrix} -4 & -2 & -1 & 3 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -5 \\ 0 & -2 & 6 & -6 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix} 92/395 \\ 368/395 \\ 459/790 \\ -24/79 \\ 347/790 \\ 247/790 \\ 303/790 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} -4a_1 - 2a_2 - a_3 + 3a_4 + 92/395 \\ a_1 + 368/395 \\ -a_2 + 3a_3 - 5a_4 + 459/790 \\ -2a_2 + 6a_3 - 6a_4 - 24/79 \\ a_2 + 347/790 \\ a_3 + 247/790 \\ a_4 + 303/790 \end{bmatrix}$$

采用基本行变换方法也能求解该方程

```
>> C=[A B]; D=rref(C)
```

得出

$$D = \begin{bmatrix} 1 & 4 & 0 & 0 & 2 & 1 & -3 & 4 \\ 0 & 0 & 1 & 0 & 1 & -3 & 5 & 2 \\ 0 & 0 & 0 & 1 & 2 & -6 & 6 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

可见,这时的自由变量为 x_2, x_5, x_6, x_7 , 它们可以选择任意数值。令 $x_2 = b_1, x_5 = b_2, x_6 = b_3, x_7 = b_4$, 由得出的 D 可以手工写出方程的解为 $x_1 = -4b_1 - 2b_2 - b_3 + 3b_4 + 4, x_3 = -b_2 + 3b_3 - 5b_4 + 2, x_4 = -2b_2 + 6b_3 - 6b_4 + 1$ 。

例 4-44 试求解线性代数方程组 $\begin{bmatrix} 4 & 7 & 1 & 4 \\ 3 & 7 & 4 & 6 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ 。

解 显然, A 和 C 矩阵的秩相同, 都为 2, 所以, 原方程有无穷多组解。方程的解析解可以用下面两种方法直接求解, 得出下面两组解

$$\mathbf{x}_1 = \begin{bmatrix} a_1 \\ a_2 + 8/21 \\ 6a_1/5 + 7a_2/5 + 1/3 \\ -13a_1/10 - 21a_2/10 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 3b_1 + 2b_2 - 1 \\ -13b_1/7 - 12b_2/7 + 1 \\ b_1 \\ b_2 \end{bmatrix}$$

```
>> A=[4,7,1,4; 3,7,4,6]; B=[3; 4]; C=[A B]; rank(A), rank(C)
syms a1 a2 b1 b2; x1=null(sym(A))*[a1; a2]+sym(A\B), A*x1-B
a=rref(sym([A B])); x2=[a(:,3:5)*[-b1; -b2; 1]; b1; b2], A*x2-B
```

经验证, 这两组解均满足原方程。

(3) 若 $\text{rank}(A) < \text{rank}(C)$, 则方程组(4-4-1)为矛盾方程, 这时只能利用 Moore-Penrose 广义逆求解出方程的最小二乘解为 $\mathbf{x} = \text{pinv}(A)*B$, 该解不满足原方程, 只能使误差的范数测度 $\|A\mathbf{x} - B\|$ 取最小值。

例 4-45 考虑线性代数方程

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 2 & 1 & 1 \\ 2 & 4 & 6 & 8 \\ 4 & 4 & 2 & 2 \end{bmatrix} \mathbf{X} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

解 先输入两个矩阵, 并构建出解的判定矩阵 C , 再求解它们的秩

```
>> A=[1 2 3 4; 2 2 1 1; 2 4 6 8; 4 4 2 2]; B=[1:4]';
C=[A B]; rank(A), rank(C)
```

可见, $\text{rank}(A) = 2 \neq \text{rank}(C) = 3$, 故原始方程是矛盾方程, 不存在任何解。可以使用 `pinv()` 函数求取 Moore-Penrose 广义逆, 从而求出原始方程的最小二乘解为

```
>> x=pinv(A)*B, norm(A*x-B)
```

得出的解为 $\mathbf{x} = [0.5466, 0.4550, 0.0443, -0.0473]^T$, 该解不满足原始代数方程组, 但它能使得最小二乘误差最小。这时得出的误差矩阵的范数为 0.4472。

如果线性方程为

$$\mathbf{x}\mathbf{A} = \mathbf{B} \quad (4-4-6)$$

则可以对上式两端进行转置处理, 得出

$$\mathbf{A}^T \mathbf{z} = \mathbf{B}^T \quad (4-4-7)$$

式中, $\mathbf{z} = \mathbf{x}^T$, 亦即可以得出形为式(4-4-1)的新线性代数方程, 套用上述的几种情况则可以求解原始线性方程组。

4.4.2 Lyapunov 方程的计算机求解

1. 连续 Lyapunov 方程

连续 Lyapunov 方程可以表示成

$$\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^T = -\mathbf{C} \quad (4-4-8)$$

Lyapunov 方程来源于微分方程稳定性理论, 其中要求 $-\mathbf{C}$ 为对称正定的 $n \times n$ 矩阵, 从而可以证明解 \mathbf{X} 亦为 $n \times n$ 对称矩阵。这类方程直接求解是很困难的, 不过有了 MATLAB 这样的计算机数学语言, 求解这样的问题就轻而易举了。可以由控制系统工具箱中提供的 `lyap()` 函数立即得出方程的解, 该函数的调用格式为 $\mathbf{X} = \text{lyap}(\mathbf{A}, \mathbf{C})$ 。所以若给出 Lyapunov 方程中的 \mathbf{A} 和 \mathbf{C} , 则可以立即获得相应 Lyapunov 方程的数值解。下面将通过例子演示一般 Lyapunov 方程的求解。

例 4-46 假设式(4-4-8)中 \mathbf{A} 、 \mathbf{C} 矩阵如下, 试求解相应的 Lyapunov 方程, 并验证解的精度。

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}, \quad \mathbf{C} = -\begin{bmatrix} 10 & 5 & 4 \\ 5 & 6 & 7 \\ 4 & 7 & 9 \end{bmatrix}$$

解 输入给定的矩阵, 可以由下面的 MATLAB 语句求出该方程的解

```
>> A=[1 2 3;4 5 6; 7 8 0]; C=-[10,5,4; 5,6,7; 4,7,9];
X=lyap(A,C), norm(A*X+X*A'+C)
```

可以得出方程的数值解如下。从最后一个语句得出解的误差为 $\|\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^T + \mathbf{C}\| = 2.64742 \times 10^{-14}$, 可见得出的方程解 \mathbf{X} 基本满足原方程, 且有较高精度。

$$\mathbf{X} = \begin{bmatrix} -3.944444444444 & 3.888888888889 & 0.388888888889 \\ 3.888888888889 & -2.777777777778 & 0.222222222222 \\ 0.388888888889 & 0.222222222222 & -0.111111111111 \end{bmatrix}$$

2. Lyapunov 方程的解析解

为方便叙述, 可以将 Lyapunov 方程的各个矩阵参数表示为

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ x_{m+1} & x_{m+2} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n-1)m+1} & x_{(n-1)m+2} & \cdots & x_{nm} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} c_1 & c_2 & \cdots & c_m \\ c_{m+1} & c_{m+2} & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(n-1)m+1} & c_{(n-1)m+2} & \cdots & c_{nm} \end{bmatrix} \quad (4-4-9)$$

利用 Kronecker 乘积的表示方法, 可以将 Lyapunov 方程写成

$$(A \otimes I + I \otimes A)x = -c \quad (4-4-10)$$

其中, $A \otimes B$ 表示矩阵 A 和 B 的 Kronecker 乘积, 其定义为

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix} \quad (4-4-11)$$

其 MATLAB 函数表示为 $C = \text{kron}(A, B)$ 。

可见, 这样的方程有唯一解的条件并不局限于 $-C$ 为对称正定矩阵, 形如式 (4-4-8) 的方程只要满足 $(A \otimes I + I \otimes A)$ 为非奇异的方阵即可保证唯一解。

例 4-47 仍考虑例 4-46 中给出的 Lyapunov 方程, 试求出其解析解。

解 由下面的语句可以求出其解析解, 将其解代入原方程可以验证这一点

```
>> A0=sym(kron(A,eye(3))+kron(eye(3),A)); c=reshape(C',9,1);
x0=-inv(A0)*c; x=reshape(x0,3,3)', norm(A*x+x*A'+C)
```

方程的解析解为 $x = \begin{bmatrix} -71/18 & 35/9 & 7/18 \\ 35/9 & -25/9 & 2/9 \\ 7/18 & 2/9 & -1/9 \end{bmatrix}$, 经检验该解没有误差。

例 4-48 传统 Lyapunov 方程的条件 (C 为实对称正定矩阵) 能否突破?

解 受微分方程稳定性影响, 以前的传统观念似乎 Lyapunov 类方程有唯一解的充分必要条件是 $-C$ 矩阵为实对称正定矩阵。事实上, 式 (4-4-8) 中给出的线性矩阵方程在不满足该条件的情况下仍有唯一解。例如, 例 4-46 中给出的 A 矩阵不变, 将 C 矩阵改为复数非对称矩阵

$$C = - \begin{bmatrix} 1+1j & 3+3j & 12+10j \\ 2+5j & 6 & 11+6j \\ 5+2j & 11+j & 2+12j \end{bmatrix}$$

用上述方法输入 A 和 C 矩阵, 可以立即解出满足该方程的复数解

```
>> A=[1 2 3;4 5 6; 7 8 0];
C=-[1+1i, 3+3i, 12+10i; 2+5i, 6, 11+6i; 5+2i, 11+1i, 2+12i];
A0=sym(kron(A,eye(3))+kron(eye(3),A)); c=reshape(C',9,1);
x0=-inv(A0)*c; x=reshape(x0,3,3)', norm(A*x+x*A'+C)
```

可以得出方程的解析解如下, 经验证该解没有误差

$$x = \begin{bmatrix} -5/102 + j1457/918 & 15/17 - j371/459 & -61/306 + j166/459 \\ 4/17 - j626/459 & -10/51 + j160/459 & 115/153 + j607/459 \\ -55/306 + j166/459 & -26/153 - j209/459 & 203/153 + j719/918 \end{bmatrix}$$

得出的解经验证确实满足原始 Lyapunov 方程。故可以得出结论, 如果不考虑 Lyapunov 方程稳定性的物理意义和 Lyapunov 函数为能量的物理原型, 完全可以将 Lyapunov 方程进一步扩展成能处理任意 C 矩阵的情形。

3. Stein 方程的求解

Stein 方程的一般形式为

$$\mathbf{A}\mathbf{X}\mathbf{B} - \mathbf{X} + \mathbf{Q} = \mathbf{0} \quad (4-4-12)$$

这里,所有的矩阵均应该是 $n \times n$ 方阵。类似于前面的介绍,令 \mathbf{X} 矩阵的向量展开为 \mathbf{x} , \mathbf{Q} 矩阵的向量展开为 \mathbf{q} , 则 Stein 方程可以由下面的线性方程直接解出

$$\left(\mathbf{I}_{n^2 \times n^2} - \mathbf{B}^T \otimes \mathbf{A} \right) \mathbf{x} = \mathbf{q} \quad (4-4-13)$$

例 4-49 试求解 Stein 方程 $\begin{bmatrix} -2 & 2 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 2 \end{bmatrix} \mathbf{X} \begin{bmatrix} -2 & -1 & 2 \\ 1 & 3 & 0 \\ 3 & -2 & 2 \end{bmatrix} - \mathbf{X} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \end{bmatrix} = \mathbf{0}$ 。

解 由下面的语句可以直接求解该方程

```
>> A=[-2,2,1; -1,0,-1; 1,-1,2]; B=[-2,-1,2; 1,3,0; 3,-2,2];
    Q=[0,-1,0; -1,1,0; 1,-1,-1]; x=inv(sym(eye(9))-kron(B',A))*Q(:);
    X=reshape(x,3,3), norm(double(A*X*B-X+Q))
```

可以得出方程的解析解为

$$\mathbf{X} = \begin{bmatrix} 4147/47149 & 3861/471490 & -40071/235745 \\ -2613/94298 & 2237/235745 & -43319/235745 \\ 20691/94298 & 66191/235745 & -10732/235745 \end{bmatrix}$$

4. 离散 Lyapunov 方程

离散 Lyapunov 方程的一般表示形式为

$$\mathbf{A}\mathbf{X}\mathbf{A}^T - \mathbf{X} + \mathbf{Q} = \mathbf{0} \quad (4-4-14)$$

该方程是 Stein 方程的一个特例。该方程还可以由 MATLAB 控制系统工具箱的 `dlyap()` 函数直接求解。该函数的调用格式为 $\mathbf{X} = \text{dlyap}(\mathbf{A}, \mathbf{Q})$ 。其实,如果 \mathbf{A} 矩阵是非奇异矩阵,则等式两端同时右乘 $(\mathbf{A}^T)^{-1}$,就可以将其变换成连续的 Sylvester 方程,可以用 4.4.3 节给出的算法求解其解析解。

例 4-50 求解离散 Lyapunov 方程

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \mathbf{X} \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}^T - \mathbf{X} + \begin{bmatrix} 16 & 4 & 1 \\ 9 & 3 & 1 \\ 4 & 2 & 1 \end{bmatrix} = \mathbf{0}$$

解 该方程可以直接用 `dlyap()` 方程求解出来

```
>> A=[8,1,6; 3,5,7; 4,9,2]; Q=[16,4,1; 9,3,1; 4,2,1];
    X=dlyap(A,Q), norm(A*X*A'-X+Q) % 精度验证
```

可以得出方程的数值解如下,其误差为 1.3462×10^{-14}

$$\mathbf{X} = \begin{bmatrix} -0.1647 & 0.0691 & -0.0168 \\ 0.0528 & -0.0298 & -0.0062 \\ -0.102 & 0.045 & -0.0305 \end{bmatrix}$$

4.4.3 Sylvester 方程的计算机求解

Sylvester 方程的一般形式为

$$AX + XB = -C \quad (4-4-15)$$

其中, A 为 $n \times n$ 矩阵, B 为 $m \times m$ 矩阵, C 和 X 均为 $n \times m$ 矩阵。该方程又称为广义 Lyapunov 方程。仍可以利用 MATLAB 控制系统工具箱中的 `lyap()` 函数直接求解该方程, `X = lyap(A, B, C)`, 该函数采用的是 Schur 分解的数值解法求解方程。如果想得到解析解, 类似于前述一般 Lyapunov 方程, 可以采用 Kronecker 乘积的形式将原始方程进行变换, 得出下面的线性代数方程

$$(A \otimes I_m + I_n \otimes B^T)x = c \quad (4-4-16)$$

如果 $(A \otimes I_m + I_n \otimes B^T)$ 矩阵为非奇异矩阵, 则 Sylvester 方程有唯一解。

综合上述算法, 可以编写出 Sylvester 型方程的解析解求解程序 `lyapsym.m`, 改写的函数清单为

```
function X=lyapsym(A,B,C)
if nargin==2, C=B; B=A'; end
[nr,nc]=size(C); A0=kron(A,eye(nc))+kron(eye(nr),B');
try
    C1=C'; x0=-inv(A0)*C1(:); X=reshape(x0,nc,nr)';
catch, error('singular matrix found. '), end
```

考虑式 (4-4-14) 中给出的离散 Lyapunov 方程, 两端同时右乘 $(A^T)^{-1}$, 则原来的离散 Lyapunov 方程可以变换成

$$AX + X[-(A^T)^{-1}] = -Q(A^T)^{-1} \quad (4-4-17)$$

故令 $B = -(A^T)^{-1}$, $C = Q(A^T)^{-1}$, 则可以将其变换成式 (4-4-15) 所示的 Sylvester 方程, 故也可以通过新的 `lyap()` 函数求解该方程。该函数的具体调用格式为

```
X = lyapsym(sym(A),C)           % 连续 Lyapunov 方程
X = lyapsym(sym(A),-inv(B),Q*inv(B)) % Stein 方程
X = lyapsym(sym(A),-inv(A'),Q*inv(A')) % 离散 Lyapunov 方程
X = lyapsym(sym(A),B,C)         % Sylvester 方程
```

例 4-51 求解下面的 Sylvester 方程

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} X + X \begin{bmatrix} 16 & 4 & 1 \\ 9 & 3 & 1 \\ 4 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

解 调用 `lyap()` 函数可以立即得出原方程的数值解

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[16,4,1; 9,3,1; 4,2,1];
    C=-[1,2,3; 4,5,6; 7,8,0]; X=lyap(A,B,C), norm(A*X+X*B+C)
```


可以得出该方程的数值解如下,经检验该解的误差为 9.5337×10^{-15} ,精度较高。

$$\mathbf{X} = \begin{bmatrix} 0.0749 & 0.0899 & -0.4329 \\ 0.0081 & 0.4814 & -0.216 \\ 0.0196 & 0.1826 & 1.1579 \end{bmatrix}$$

如果想获得原方程的解析解,则可以使用下面的语句直接求解

```
>> x=lyapsym(sym(A),B,C), norm(A*x+x*B+C)
```

得出方程的解如下,经检验该解是原方程的解析解

$$\mathbf{x} = \begin{bmatrix} 1349214/18020305 & 648107/7208122 & -15602701/36040610 \\ 290907/36040610 & 3470291/7208122 & -3892997/18020305 \\ 70557/3604061 & 1316519/7208122 & 8346439/7208122 \end{bmatrix}$$

当然,lyapsym() 函数仍然可以用于数值求解 Sylvester 方程,如果 \mathbf{A} 矩阵是数值矩阵,不变换成符号矩阵,则可以得出原方程的数值解,得出的误差为 4.2837×10^{-15} ,此例子的运算得出的误差略小于前面介绍的 lyap() 函数。

```
>> x=lyapsym(A,B,C), norm(A*x+x*B+C)
```

例 4-52 重新考虑例 4-50 中给出的离散 Lyapunov 方程,试求取其解析解。

解 该方程可以通过下面的语句求解出解析解

```
>> A=[8,1,6; 3,5,7; 4,9,2]; Q=[16,4,1; 9,3,1; 4,2,1];
```

```
x=lyapsym(sym(A),-inv(A'),Q*inv(A')), norm(A*x*A'-x+Q)
```

得出方程的解如下,经检验该解是原方程的解析解。

$$\mathbf{x} = \begin{bmatrix} -22912341/139078240 & 48086039/695391200 & -11672009/695391200 \\ 36746487/695391200 & -20712201/695391200 & -4279561/695391200 \\ -70914857/695391200 & 31264087/695391200 & -4247541/139078240 \end{bmatrix}$$

例 4-53 求解下面的 Sylvester 方程

$$\mathbf{A} = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

解 Sylvester 方程能解决的问题中并未要求 \mathbf{C} 矩阵为方阵,利用上面的语句仍然能求出此方程的解析解,这里还可以尝试上面编写的 Lyapunov 方程解析解求解的新函数 lyapsym(),可以直接求解上述的方程

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[2,3; 4,5]; C=-[1,2; 3,4; 5,6];
```

```
X=lyapsym(sym(A),B,C), norm(A*X+X*B+C) % 经检验没有误差
```

得出的解为 $\mathbf{X} = \begin{bmatrix} -2853/14186 & -11441/56744 \\ -557/14186 & -8817/56744 \\ 9119/14186 & 50879/56744 \end{bmatrix}$,经检验该解是原方程的解析解。

4.4.4 Riccati 方程的计算机求解

Riccati 方程是一类很著名的二次型矩阵方程式,其一般形式为

$$\mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} - \mathbf{X} \mathbf{B} \mathbf{X} + \mathbf{C} = \mathbf{0} \quad (4-4-18)$$

由于含有未知矩阵 \mathbf{X} 的二次项,所以 Riccati 方程的求解数学上要比 Lyapunov 方程更难。MATLAB 的控制系统工具箱中提供了现成函数 `are()`,可以直接求解式(4-4-18)给出的方程,该函数的具体调用格式为 $\mathbf{X} = \text{are}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ 。

例 4-54 考虑式(4-4-18)中给出的 Riccati 方程,其中

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & -3 \\ -1 & 0 & -2 \\ 0 & -1 & -2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 2 & 2 & -2 \\ -1 & 5 & -2 \\ -1 & 1 & 2 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 5 & -4 & 4 \\ 1 & 0 & 4 \\ 1 & -1 & 5 \end{bmatrix}$$

试求出该方程的数值解,并验证解的正确性。

解 可以用下面的语句直接求解该方程。

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
C=[5 -4 4; 1 0 4; 1 -1 5]; X=are(A,B,C), norm(A'*X+X*A-X*B*X+C)
```

得到的解如下,代入原方程可以得出误差为 7.9301×10^{-15} ,故得出的解满足原方程。

$$\mathbf{X} = \begin{bmatrix} 0.9874 & -0.7983 & 0.4189 \\ 0.5774 & -0.1308 & 0.5775 \\ -0.284 & -0.073 & 0.6924 \end{bmatrix}$$

当然,求解函数 `are()` 的局限性是极大的,如果方程的形式稍有变化,如

$$\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{D} - \mathbf{X}\mathbf{B}\mathbf{X} + \mathbf{C} = \mathbf{0}, \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{D} - \mathbf{X}\mathbf{B}\mathbf{X}^T + \mathbf{C} = \mathbf{0} \quad (4-4-19)$$

这类函数都无能为力。即便对标准的 Riccati 方程来说, `are()` 也只能求出其一个根,不能得出其他的根。第 6 章将探讨一般矩阵方程的数值解方法,并试图得到矩阵方程所有的根。

4.4.5 一类线性不等式的求解

假设已知一类线性不等式的数学形式为

$$\begin{cases} |a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n| \leq b_1 \\ |a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n| \leq b_2 \\ \vdots \\ |a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n| \leq b_n \end{cases} \quad (4-4-20)$$

假设系数矩阵 $\mathbf{A} = (a_{ij})$ 为非奇异方阵,且 $b_i \geq 0, i = 1, 2, \cdots, n$,则该不等式的解为^[9]

$$|x_i| \leq \frac{1}{|\det(\mathbf{A})|} \sum_{j=1}^n b_j |A_{ji}|, \quad i = 1, 2, \cdots, n \quad (4-4-21)$$

其中 A_{ji} 为 a_{ji} 的代数余子式。根据上述方法可以编写出下面的 MATLAB 函数,可以直接求解线性矩阵不等式

```
function b0=linineq(A,b)
for i=1:length(b), s=0;
```

```

for j=1:length(b)
    B=A; B(j,:)=[]; B(:,j)=[]; s=s+b(j)*abs(det(B));
end, b0(i,1)=s/abs(det(A));
end

```

例 4-55 试求解线性矩阵不等式 $|Ax| \leq b$, 其中 $A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ 。

解 由下面语句可以直接求解该不等式, 得出边界为 $b_0 = [113/180, 19/45, 127/180]^T$ 。

```
>> A=[8,1,6; 3,5,7; 4,9,2]; b=[1; 2; 3]; b0=linineq(sym(A),b)
```

4.5 非线性运算与矩阵函数求值

4.5.1 面向矩阵元素的非线性运算

MATLAB 提供了大量函数, 允许用户对矩阵进行处理, 前面介绍的主要是矩阵的线性变换, 本节将介绍如何对矩阵进行非线性运算。

事实上, MATLAB 提供了两类函数, 其中一类是对矩阵的各个元素进行单独运算的, 而另一类是对整个矩阵进行运算的。前面曾经用到了 `sin()` 函数, 该函数属于第一类, 是对矩阵的各个元素单独运算的, 而不是对整个矩阵进行运算的。这类常用的 MATLAB 函数在表 4-2 中列出来, 它们的调用方法是很显然的, 其标准调用格式为

$B = \text{函数名}(A)$; 例如 $B = \sin(A)$;

表 4-2 面向矩阵元素的非线性函数表

函数名	意义	函数名	意义
<code>abs()</code>	求模(绝对值)函数	<code>asin()</code> , <code>acos()</code> , <code>atan()</code>	反正弦、余弦、正切函数
<code>sqrt()</code>	求平方根函数	<code>log()</code> , <code>log10()</code>	自然和常用对数
<code>exp()</code>	指数函数	<code>real()</code> , <code>imag()</code> , <code>conj()</code>	求实虚部及共轭复数
<code>sin()</code> , <code>cos()</code> , <code>tan()</code>	正弦、余弦、正切函数	<code>round()</code> , <code>floor()</code> , <code>ceil()</code>	取整数函数

例 4-56 考虑例 4-8 中给出的 A 矩阵, 调用其中的一些函数, 其结果在下面给出

```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1]; exp(A), sin(A)
```

可以得出

$$\exp(A) = \begin{bmatrix} 8.8861 \times 10^6 & 7.3891 & 20.0855 & 0.4424 \times 10^6 \\ 148.4132 & 5.9874 \times 10^4 & 2.2026 \times 10^4 & 2980.958 \\ 8103.0839 & 1096.6332 & 403.4288 & 1.6275 \times 10^5 \\ 54.5982 & 1.2026 \times 10^6 & 3.2690 \times 10^6 & 2.7183 \end{bmatrix}$$

$$\sin(A) = \begin{bmatrix} -0.2879 & 0.9093 & 0.1411 & 0.4202 \\ -0.9589 & -1 & -0.544 & 0.9894 \\ 0.4121 & 0.657 & -0.2794 & -0.5366 \\ -0.7568 & 0.9906 & 0.6503 & 0.8415 \end{bmatrix}$$

4.5.2 矩阵函数求值

1. 矩阵指数与对数的函数运算

除了对矩阵的单个元素进行单独计算以外,一般还常常要求对整个矩阵做这样的非线性运算。例如,想求出一个矩阵的 e 指数,就需要特殊的算法来完成了。文献 [10] 中叙述了求解矩阵指数的 19 种不同方法,每一种方法都有自己的特点及适用范围。MATLAB 中提供了求取矩阵指数的函数 `expm()`,其调用格式为 $E = \text{expm}(A)$ 。该函数还支持 A 为符号变量问题的求解,如矩阵函数 e^{At} 。MATLAB 提供了 $C = \text{logm}(A)$ 求数值矩阵的对数函数。

例 4-57 试求出下面矩阵的指数 e^A 和对数 $\ln A$ 。

$$A = \begin{bmatrix} -2 & 1 & 0 & & \\ 0 & -2 & 1 & & \\ 0 & 0 & -2 & & \\ & & & -5 & 1 \\ & & & 0 & -5 \end{bmatrix}$$

解 如果对此矩阵进行指数运算和对数运算,则可以给出下列命令

```
>> A1=[-2 1 0; 0 -2 1; 0 0 -2]; A2=[-5 1; 0 -5]; A=diagm(A1,A2)
B=expm(A), C=logm(B), norm(C-A)
```

得出的指数矩阵如下,变换矩阵 C 和 A 矩阵几乎一致,误差矩阵的范数为 $\|C - A\| = 1.6760 \times 10^{-15}$

$$B = \begin{bmatrix} 0.1353 & 0.1353 & 0.0677 & 0 & 0 \\ 0 & 0.1353 & 0.1353 & 0 & 0 \\ 0 & 0 & 0.1353 & 0 & 0 \\ 0 & 0 & 0 & 0.0067 & 0.0067 \\ 0 & 0 & 0 & 0 & 0.0067 \end{bmatrix}$$

对得出的指数结果进行对数运算可以按相当高的精度还原原始矩阵,从而表明,矩阵对数运算还是很精确的。

原始问题还可以调用解析解函数 `expm()`,直接求解 e^{At} 。注意,这里包含了变量 t ,所以这是数值算法无法解出的。

```
>> syms t; expm(A*t)
```

该语句可以立即写出指数矩阵如下。由于该矩阵已经由 Jordan 矩阵形式给出,所以指数矩阵的结果可以直接写出

$$e^{At} = \begin{bmatrix} e^{-2t} & te^{-2t} & t^2e^{-2t}/2 & 0 & 0 \\ 0 & e^{-2t} & te^{-2t} & 0 & 0 \\ 0 & 0 & e^{-2t} & 0 & 0 \\ 0 & 0 & 0 & e^{-5t} & te^{-5t} \\ 0 & 0 & 0 & 0 & e^{-5t} \end{bmatrix}$$

例 4-58 已知一般矩阵 A ,试求出 e^{At}

$$A = \begin{bmatrix} -3 & -1 & -1 \\ 0 & -3 & -1 \\ 1 & 2 & 0 \end{bmatrix}$$

解 如果 Jordan 标准型不那么明显,则不能采用直接写出的方法求解 e^{At} ,而应该采用广义特征向量矩阵的方式进行变换。由下面的语句

```
>> syms t; A=[-3,-1,-1; 0,-3,-1; 1,2,0]; simple(expm(A*t))
```

可以得出

$$e^{At} = \begin{bmatrix} -e^{-2t}(-1+t) & -te^{-2t} & -te^{-2t} \\ -t^2e^{-2t}/2 & -e^{-2t}(-1+t+t^2/2) & -te^{-2t}(2+t/2) \\ te^{-2t}/2 & te^{-2t}(2+t/2) & e^{-2t}(1+2t+t^2/2) \end{bmatrix}$$

下面演示基于 Jordan 矩阵变换的 e^{At} 矩阵处理方法。

```
>> [V,J]=jordan(A) % Jordan 矩阵变换
```

可以得出 Jordan 矩阵 J 和广义特征向量矩阵 V , 并由 Jordan 矩阵写出 e^{Jt}

$$V = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \quad J = \begin{bmatrix} -2 & 1 & 0 \\ 0 & -2 & 1 \\ 0 & 0 & -2 \end{bmatrix}, \quad e^{Jt} = e^{-t} \begin{bmatrix} 1 & t & t^2/2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix}$$

这样利用 Jordan 矩阵的性质即可以求出原矩阵的指数函数, 与前面得出的完全一致。

```
>> J1=exp(-2*t)*[1 t t^2/2; 0 1 t; 0 0 1]; A1=simple(V*J1*inv(V))
```

其实, 用这样的方法求解矩阵指数不是此例的目的, 因为用符号运算工具箱中的 `expm()` 函数可以立即得出所需的结果。后面将通过例子演示其他函数, 如正弦等函数如何用 Jordan 矩阵的方法求解。

2. 矩阵的三角函数运算

MATLAB 下没有对矩阵进行三角函数运算的现成函数, 其数值解可以通过 `funm()` 函数得出。该函数的目的是求出矩阵的任意函数, 调用方法为 $A_1 = \text{funm}(A, \text{'函数名'})$, 其中, 函数名应该由单引号括起来。例如, 可以通过命令 $B = \text{funm}(A, \text{'sin'})$ 求出矩阵 A 的正弦矩阵。

例 4-59 重新考虑例 4-58 中给出的矩阵, 试求出其正弦 $\sin A$ 。

解 如果想对其中的 A 矩阵进行正弦运算, 则可以得出下面的语句

```
>> A=[-3,-1,-1; 0,-3,-1; 1,2,0]; B=funm(A,'sin')
```

这样得出的矩阵正弦函数为

$$B = \begin{bmatrix} -0.4931 & 0.4161 & 0.4161 \\ -0.4546 & -0.9478 & -0.0385 \\ 0.0385 & -0.3776 & -1.2869 \end{bmatrix}$$

事实上, 矩阵的非线性函数运算可以通过幂级数的方法简单地求出。例如, 正弦函数可以由下面的幂级数展开式求出

$$\sin A = \sum_{i=0}^{\infty} (-1)^i \frac{A^{2i+1}}{(2i+1)!} = A - \frac{1}{3!}A^3 + \frac{1}{5!}A^5 + \cdots \quad (4-5-1)$$

可以用 MATLAB 实现正弦函数幂级数的展开, 如果误差足够小则停止叠加程序

```
function E=sinm1(A)
E=zeros(size(A)); F=A; k=1;
while norm(E+F-E,1)>0, E=E+F; F=-A^2*F/((k+2)*(k+1)); k=k+2; end
```

例 4-60 由上面的程序可以看出, 看起来比较复杂的矩阵正弦函数的幂级数展开运算可以由几

条 MATLAB 语句容易地编写出来。利用函数 $\text{sinm1}(\mathbf{A})$ 可以容易地求出原矩阵的正弦矩阵,与前面的完全一致。

可以测出,该函数一共进行了 39 次叠加运算。对上面的结果矩阵再进行反正弦运算,不难得出这样的结论,这种运算可以还原出原来的矩阵 \mathbf{A} ,而这样的结果光靠 MATLAB 提供的 $\text{funm}()$ 函数是不可能得出的。所以在使用 $\text{funm}()$ 函数时应该格外注意,如果确实不能得出正确的结果,则建议采用幂级数的方法编写程序来直接求出。

3. 矩阵三角函数的解析求解

再考虑矩阵三角函数的解析求解方法。先考虑标量三角函数的运算公式,根据著名的 Euler 公式 $e^{ja} = \cos a + j \sin a$ 与 $e^{-ja} = \cos a - j \sin a$ 可以立即推导出

$$\sin a = \frac{1}{j2}(e^{ja} - e^{-ja}), \quad \cos a = \frac{1}{2}(e^{ja} + e^{-ja}) \quad (4-5-2)$$

此公式可以直接用于 a 为矩阵的形式。由于前面已经给出了可靠的指数矩阵求解函数 $\text{expm}()$,利用该函数可以直接得出一般矩阵的正弦和余弦函数的解析解运算结果。

例 4-61 仍考虑例 4-57 中给出的矩阵,试求解 $\sin \mathbf{A}$ 。

解 可以利用现成的 $\text{expm}()$ 函数求出矩阵的正弦函数

```
>> A1=[-2 1 0; 0 -2 1; 0 0 -2]; A2=[-5 1; 0 -5]; A=diagm(A1,A2);
j=sqrt(-1); A1=(expm(A*j)-expm(-A*j))/(2*j)
```

可见,这样得出的解与例 4-60 完全一致,证明该解是正确的。

例 4-62 假设已知如下矩阵有重特征值,试求出该矩阵的正弦函数 $\sin \mathbf{A}t$ 和余弦函数 $\cos \mathbf{A}t$ 。

$$\mathbf{A} = \begin{bmatrix} -7 & 2 & 0 & -1 \\ 1 & -4 & 2 & 1 \\ 2 & -1 & -6 & -1 \\ -1 & -1 & 0 & -4 \end{bmatrix}$$

解 根据式(4-5-2)可以由下面的语句求解矩阵的正弦和余弦函数

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4]; syms t
A1=(expm(A*j*t)-expm(-A*j*t))/(2*j); A1=simple(A1)
A2=(expm(A*j*t)+expm(-A*j*t))/2; A2=simple(A2)
```

上述的语句可以直接求出如下的解

$$\sin \mathbf{A}t = \begin{bmatrix} -2/9 \sin 3t + (t^2 - 7/9) \sin 6t - 5/3t \cos 6t & -1/3 \sin 3t + 1/3 \sin 6t + t \cos 6t \\ -2/9 \sin 3t + (t^2 + 2/9) \sin 6t + 1/3t \cos 6t & -1/3 \sin 3t - 2/3 \sin 6t + t \cos 6t \\ -2/9 \sin 3t + (-2t^2 + 2/9) \sin 6t + 4/3t \cos 6t & -1/3 \sin 3t + 1/3 \sin 6t - 2t \cos 6t \\ 4/9 \sin 3t + (t^2 - 4/9) \sin 6t + 1/3t \cos 6t & 2/3 \sin 3t - 2/3 \sin 6t + t \cos 6t \\ -2/9 \sin 3t + (2/9 + t^2) \sin 6t - 2/3t \cos 6t & 1/9 \sin 3t + (-1/9 + t^2) \sin 6t - 2/3t \cos 6t \\ -2/9 \sin 3t + (2/9 + t^2) \sin 6t + 4/3t \cos 6t & 1/9 \sin 3t + (-1/9 + t^2) \sin 6t + 4/3t \cos 6t \\ -2/9 \sin 3t - (7/9 + 2t^2) \sin 6t - 2/3t \cos 6t & 1/9 \sin 3t - (1/9 + 2t^2) \sin 6t - 2/3t \cos 6t \\ 4/9 \sin 3t + (-4/9 + t^2) \sin 6t + 4/3t \cos 6t & -2/9 \sin 3t + (-7/9 + t^2) \sin 6t + 4/3t \cos 6t \end{bmatrix}$$

$$\cos \mathbf{A}t = \begin{bmatrix} 2/9 \cos 3t + (-t^2 + 7/9) \cos 6t - 5/3t \sin 6t & 1/3 \cos 3t - 1/3 \cos 6t + t \sin 6t \\ 2/9 \cos 3t - (t^2 + 2/9) \cos 6t + 1/3t \sin 6t & 1/3 \cos 3t + 2/3 \cos 6t + t \sin 6t \\ 2/9 \cos 3t + (2t^2 - 2/9) \cos 6t + 4/3t \sin 6t & 1/3 \cos 3t - 1/3 \cos 6t - 2t \sin 6t \\ -4/9 \cos 3t + (-t^2 + 4/9) \cos 6t + 1/3t \sin 6t & -2/3 \cos 3t + 2/3 \cos 6t + t \sin 6t \end{bmatrix}$$

$$\begin{bmatrix} 2/9 \cos 3t - (2/9 + t^2) \cos 6t - 2/3t \sin 6t & -1/9 \cos 3t + (1/9 - t^2) \cos 6t - 2/3t \sin 6t \\ 2/9 \cos 3t - (2/9 + t^2) \cos 6t + 4/3t \sin 6t & -1/9 \cos 3t + (1/9 - t^2) \cos 6t + 4/3t \sin 6t \\ 2/9 \cos 3t + (7/9 + 2t^2) \cos 6t - 2/3t \sin 6t & -1/9 \cos 3t + (1/9 + 2t^2) \cos 6t - 2/3t \sin 6t \\ -4/9 \cos 3t + (4/9 - t^2) \cos 6t + 4/3t \sin 6t & 2/9 \cos 3t + (7/9 - t^2) \cos 6t + 4/3t \sin 6t \end{bmatrix}$$

4.5.3 一般矩阵函数的运算

除了对整个矩阵求取矩阵指数、对数函数之外, MATLAB 还允许求取矩阵的其他非线性函数, 其中常用的函数还有 `sqrtm()` (矩阵求平方根) 和 `funm()` (矩阵求任意函数) 等。可以看出, 这里的函数名很有特点, 每个函数名在标准函数名的后面加了一个后缀 `m`, 表示对矩阵而不是对矩阵元素进行运算。遗憾的是, 现有的 `funm()` 不能求取矩阵函数的解析解。这里将介绍基于 Jordan 矩阵的矩阵函数求解方法^[11]并给出其 MATLAB 实现。

首先可以将 m_i 阶 Jordan 块 \mathbf{J}_i 写成 $\mathbf{J}_i = \lambda_i \mathbf{I} + \mathbf{H}_{m_i}$, 其中, λ_i 为 Jordan 矩阵的重特征值, \mathbf{H}_{m_i} 为幂零矩阵——次对角线元素为 1, 其他均为 0, 且有 $k \geq m_i$ 时 $\mathbf{H}_{m_i}^k \equiv \mathbf{0}$ 。这样可以证明, Jordan 矩阵块 \mathbf{J}_i 的矩阵函数 $\psi(\mathbf{J}_i)$ 可以由下式求出

$$\psi(\mathbf{J}_i) = \psi(\lambda_i) \mathbf{I}_{m_i} + \psi'(\lambda_i) \mathbf{H}_{m_i} + \cdots + \frac{\psi^{(m_i-1)}(\lambda_i)}{(m_i-1)!} \mathbf{H}_{m_i}^{m_i-1} \quad (4-5-3)$$

如果通过 Jordan 矩阵分解的方法可以将任意矩阵 \mathbf{A} 分解成

$$\mathbf{A} = \mathbf{V} \begin{bmatrix} \mathbf{J}_1 & & & \\ & \mathbf{J}_2 & & \\ & & \ddots & \\ & & & \mathbf{J}_m \end{bmatrix} \mathbf{V}^{-1} \quad (4-5-4)$$

这样, 该矩阵的任意函数 $\psi(\mathbf{A})$ 可以最终如下求出。如果通过 Jordan 矩阵分解的方法可以将任意矩阵 \mathbf{A} 分解成

$$\psi(\mathbf{A}) = \mathbf{V} \begin{bmatrix} \psi(\mathbf{J}_1) & & & \\ & \psi(\mathbf{J}_2) & & \\ & & \ddots & \\ & & & \psi(\mathbf{J}_m) \end{bmatrix} \mathbf{V}^{-1} \quad (4-5-5)$$

根据上面的算法可以立即编写出新的函数 `funmsym()`, 可以推导任意矩阵函数的解析解。该函数的清单为

```
function F=funmsym(A,fun,x)
[V,T]=jordan(A); vec=diag(T); v1=[0,diag(T,1)',0]; v2=find(v1==0);
v_n=v2(2:end)-v2(1:end-1); lam=vec(v2(1:end-1)); vec(v2(1:end-1));
m=length(lam); F=sym([]);
for i=1:m,
    k=v2(i):v2(i)+v_n(i)-1; J1=T(k,k); fJ=funJ(J1,fun,x); F(k,k)=fJ;
end
```

```

F=V*F*inv(V);
function fJ=funJ(J,fun,x)
lam=J(1,1); f1=fun; fJ=subs(fun,x,lam)*eye(size(J));
H=diag(diag(J,1),1); H1=H;
for i=2:length(J)
    f1=diff(f1,x); a1=subs(f1,x,lam); fJ=fJ+a1*H1; H1=H1*H/i;
end

```

该函数的调用格式为 $\mathbf{A}_1 = \text{funmsym}(\mathbf{A}, \text{funx}, x)$, 其中, x 为符号型自变量, funx 为 x 的函数表示。例如, 若想求出 $e^{\mathbf{A}}$, 则可以将 funx 填写成 $\exp(x)$ 。其实, funx 参数可以描述任意复杂的函数, 如 $\exp(x*t)$ 表示求取 $e^{\mathbf{A}t}$, 其中 t 也应该事先设置成符号变量。另外, 该函数还可以表示成 $\exp(x*\cos(x*t))$ 型的复合函数, 表示需要求取 $\psi(\mathbf{A}) = e^{\mathbf{A} \cos \mathbf{A} t}$ 。

例 4-63 已知给定矩阵 \mathbf{A} , 试求出矩阵函数 $\psi(\mathbf{A}) = e^{\mathbf{A} \cos(\mathbf{A}t)}$

$$\mathbf{A} = \begin{bmatrix} -7 & 2 & 0 & -1 \\ 1 & -4 & 2 & 1 \\ 2 & -1 & -6 & -1 \\ -1 & -1 & 0 & -4 \end{bmatrix}$$

解 可以用下面的语句将其输入到 MATLAB 环境中

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4];
```

如果想求出 $\psi(\mathbf{A}) = e^{\mathbf{A} \cos \mathbf{A} t}$, 分析题意, 可以用下面的语句

```
>> syms x t; A1=funmsym(A,exp(x*cos(x*t)),x)
```

得出的结果是很冗长的, 根本无法显示全部内容, 这里只给出其中一项为

$$\psi_{1,1}(\mathbf{A}) = 2/9e^{-3 \cos 3t} + (2t \sin 6t + 6t^2 \cos 6t)e^{-6 \cos 6t} + (\cos 6t - 6t \sin 6t)^2 e^{-6 \cos 6t} - 5/3(\cos 6t - 6t \sin 6t)e^{-6 \cos 6t} + 7/9e^{-6 \cos 6t}$$

可见, 这样得出的 $\psi_{1,1}(t)$ 有很多项均是 $e^{-6 \cos 6t}$ 的系数项, 故可以通过合并同类项的化简方法手动给出下面的命令

```
>> collect(A1(1,1),exp(-6*cos(6*t)))
```

则可以得出如下的化简结果

$$\psi_{1,1}(\mathbf{A}) = \left[12t \sin 6t + 6t^2 \cos 6t + (\cos 6t - 6t \sin 6t)^2 - \frac{5}{3} \cos 6t + \frac{7}{9} \right] e^{-6 \cos 6t} + \frac{2}{9} e^{-3 \cos 3t}$$

进一步地, 若令 $t = 1$, 则可以求出 $e^{\mathbf{A} \cos \mathbf{A}}$ 的精确数值解

```
>> subs(A1,t,1) % 该结果与语句 expm(A*funm(A,'cos')) 得出的完全一致
```

得出

$$e^{\mathbf{A} \cos \mathbf{A}} = \begin{bmatrix} 4.3583 & 6.5044 & 4.3635 & -2.1326 \\ 4.3718 & 6.5076 & 4.3801 & -2.116 \\ 4.2653 & 6.4795 & 4.2518 & -2.2474 \\ -8.6205 & -12.984 & -8.6122 & 4.3832 \end{bmatrix}$$

4.6 习 题

1. Jordan 矩阵是矩阵分析中一类很实用的矩阵,其一般形式为

$$\mathbf{J} = \begin{bmatrix} -\alpha & 1 & 0 & \cdots & 0 \\ 0 & -\alpha & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\alpha \end{bmatrix}, \quad \text{例如 } \mathbf{J}_1 = \begin{bmatrix} -5 & 1 & 0 & 0 & 0 \\ 0 & -5 & 1 & 0 & 0 \\ 0 & 0 & -5 & 1 & 0 \\ 0 & 0 & 0 & -5 & 1 \\ 0 & 0 & 0 & 0 & -5 \end{bmatrix}$$

试利用 `diag()` 函数给出构造 \mathbf{J}_1 的语句。

2. 试不用循环方式输入下面两个 20×20 矩阵,并求出它们的行列式、迹和特征多项式系数。

$$\mathbf{A} = \begin{bmatrix} a & & & & b \\ & \ddots & & & \\ & & a & b & \ddots \\ & & b & a & \\ b & & & & a \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} x & a & a & \cdots & a \\ a & x & a & \cdots & a \\ a & a & x & \cdots & a \\ \vdots & \vdots & \vdots & \ddots & a \\ a & a & a & \cdots & x \end{bmatrix}$$

3. 幂零矩阵是一类特殊的矩阵,其基本形式如下,即矩阵的次主对角线元素为 1,其余均为 0,试验证对指定阶次的幂零矩阵,有 $\mathbf{H}_n^i = \mathbf{0}$ 对所有的 $i \geq n$ 成立。

$$\mathbf{H}_n = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

4. 试从矩阵的显示格式区分符号矩阵和数值矩阵,明确它们的含义和应用场合。若 \mathbf{A} 矩阵为数值矩阵, \mathbf{B} 为符号矩阵, $\mathbf{C} = \mathbf{A} * \mathbf{B}$ 运算得出的 \mathbf{C} 矩阵是符号矩阵还是数值矩阵?

5. 请将下面给出的矩阵 \mathbf{A} 和 \mathbf{B} 输入到 MATLAB 环境中,并将它们转换成符号矩阵。

$$\mathbf{A} = \begin{bmatrix} 5 & 7 & 6 & 5 & 1 & 6 & 5 \\ 2 & 3 & 1 & 0 & 0 & 1 & 4 \\ 6 & 4 & 2 & 0 & 6 & 4 & 4 \\ 3 & 9 & 6 & 3 & 6 & 6 & 2 \\ 10 & 7 & 6 & 0 & 0 & 7 & 7 \\ 7 & 2 & 4 & 4 & 0 & 7 & 7 \\ 4 & 8 & 6 & 7 & 2 & 1 & 7 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & 5 & 5 & 0 & 1 & 2 & 3 \\ 3 & 2 & 5 & 4 & 6 & 2 & 5 \\ 1 & 2 & 1 & 1 & 3 & 4 & 6 \\ 3 & 5 & 1 & 5 & 2 & 1 & 2 \\ 4 & 1 & 0 & 1 & 2 & 0 & 1 \\ -3 & -4 & -7 & 3 & 7 & 8 & 12 \\ 1 & -10 & 7 & -6 & 8 & 1 & 5 \end{bmatrix}$$

6. 试求出 Vandermonde 矩阵 \mathbf{A} 的行列式,并以最简的形式显示结果。

$$\mathbf{A} = \begin{bmatrix} a^4 & a^3 & a^2 & a & 1 \\ b^4 & b^3 & b^2 & b & 1 \\ c^4 & c^3 & c^2 & c & 1 \\ d^4 & d^3 & d^2 & d & 1 \\ e^4 & e^3 & e^2 & e & 1 \end{bmatrix}$$

7. 利用 MATLAB 语言提供的现成函数对习题 5 中给出的两个矩阵进行分析,判定它们是否为奇异矩阵,得出矩阵的秩、行列式、迹和逆矩阵,检验得出的逆矩阵是否正确。

8. 试求出习题5中给出的 \mathbf{A} 和 \mathbf{B} 矩阵的特征多项式、特征值与特征向量,并验证 Cayley-Hamilton 定理,解释并验证如何运算能消除误差。

9. 对任意矩阵 $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$, 试验证 Cayley-Hamilton 定理。

$$\mathbf{A}_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, \quad \mathbf{A}_3 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

10. 试对习题5中给出的 \mathbf{A} 和 \mathbf{B} 矩阵进行奇异值分解、LU 分解及正交分解。

11. 试求出下面矩阵的特征值、特征向量、奇异值。

$$\mathbf{A} = \begin{bmatrix} 2 & 7 & 5 & 7 & 7 \\ 7 & 4 & 9 & 3 & 3 \\ 3 & 9 & 8 & 3 & 8 \\ 5 & 9 & 6 & 3 & 6 \\ 2 & 6 & 8 & 5 & 4 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 703 & 795 & 980 & 137 & 661 \\ 547 & 957 & 271 & 12 & 284 \\ 445 & 523 & 252 & 894 & 469 \\ 695 & 880 & 876 & 199 & 65 \\ 621 & 173 & 737 & 299 & 988 \end{bmatrix}$$

12. 试判定下面矩阵是否为正定矩阵,如果是,则得出其 Cholesky 分解矩阵。

$$\mathbf{A} = \begin{bmatrix} 9 & 2 & 1 & 2 & 2 \\ 2 & 4 & 3 & 3 & 3 \\ 1 & 3 & 7 & 3 & 4 \\ 2 & 3 & 3 & 5 & 4 \\ 2 & 3 & 4 & 4 & 5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 16 & 17 & 9 & 12 & 12 \\ 17 & 12 & 12 & 2 & 18 \\ 9 & 12 & 18 & 7 & 13 \\ 12 & 2 & 7 & 18 & 12 \\ 12 & 18 & 13 & 12 & 10 \end{bmatrix}$$

13. 试对矩阵 \mathbf{A} 进行 Jordan 变换,并得出变换矩阵。

$$\mathbf{A} = \begin{bmatrix} -2 & 0.5 & -0.5 & 0.5 \\ 0 & -1.5 & 0.5 & -0.5 \\ 2 & 0.5 & -4.5 & 0.5 \\ 2 & 1 & -2 & -2 \end{bmatrix}$$

14. 试求下面齐次方程的基础解系。

$$(1) \begin{cases} 6x_1 + x_2 + 4x_3 - 7x_4 - 3x_5 = 0 \\ -2x_1 - 7x_2 - 8x_3 + 6x_4 = 0 \\ -4x_1 + 5x_2 + x_3 - 6x_4 + 8x_5 = 0 \\ -34x_1 + 36x_2 + 9x_3 - 21x_4 + 49x_5 = 0 \\ -26x_1 - 12x_2 - 27x_3 + 27x_4 + 17x_5 = 0 \end{cases} \quad (2) \mathbf{A} = \begin{bmatrix} -1 & 2 & -2 & 1 & 0 \\ 0 & 3 & 2 & 2 & 1 \\ 3 & 1 & 3 & 2 & -1 \end{bmatrix}$$

15. 试求下面线性代数方程的解析解与数值解,并检验解的正确性。

$$\begin{bmatrix} 2 & -9 & 3 & -2 & -1 \\ 10 & -1 & 10 & 5 & 0 \\ 8 & -2 & -4 & -6 & 3 \\ -5 & -6 & -6 & -8 & -4 \end{bmatrix} \mathbf{X} = \begin{bmatrix} -1 & -4 & 0 \\ -3 & -8 & -4 \\ 0 & 3 & 3 \\ 9 & -5 & 3 \end{bmatrix}$$

16. 试求解线性代数方程组,并检验解的正确性。

$$\mathbf{X} \begin{bmatrix} 7 & 6 & 9 & 7 \\ 7 & 1 & 3 & 2 \\ 2 & 1 & 5 & 5 \\ 6 & 4 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 3 & 1 & 2 \end{bmatrix}$$

17. 试判定下面的线性代数方程是否有解。

$$\begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix} \mathbf{X} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 7 \end{bmatrix}$$

18. 试求出线性代数方程的解析解,并验证解的正确性。

$$\begin{bmatrix} 2 & 9 & 4 & 12 & 5 & 8 & 6 \\ 12 & 2 & 8 & 7 & 3 & 3 & 7 \\ 3 & 0 & 3 & 5 & 7 & 5 & 10 \\ 3 & 11 & 6 & 6 & 9 & 9 & 1 \\ 11 & 2 & 1 & 4 & 6 & 8 & 7 \\ 5 & -18 & 1 & -9 & 11 & -1 & 18 \\ 26 & -27 & -1 & 0 & -15 & -13 & 18 \end{bmatrix} \mathbf{X} = \begin{bmatrix} 1 & 9 \\ 5 & 12 \\ 4 & 12 \\ 10 & 9 \\ 0 & 5 \\ 10 & 18 \\ -20 & 2 \end{bmatrix}$$

19. 对下面的矩阵 \mathbf{A} 和 \mathbf{B} , 试计算 $\mathbf{A} \otimes \mathbf{B}$ 和 $\mathbf{B} \otimes \mathbf{A}$, 并判定二者是否相等。

$$\mathbf{A} = \begin{bmatrix} -1 & 2 & 2 & 1 \\ -1 & 2 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & 0 & 3 \\ 3 & 2 & 2 \\ 3 & 1 & 1 \end{bmatrix}$$

20. 试用数值方法和解析方法求解下面的 Sylvester 方程, 并验证得出的结果。

$$\begin{bmatrix} 3 & -6 & -4 & 0 & 5 \\ 1 & 4 & 2 & -2 & 4 \\ -6 & 3 & -6 & 7 & 3 \\ -13 & 10 & 0 & -11 & 0 \\ 0 & 4 & 0 & 3 & 4 \end{bmatrix} \mathbf{X} + \mathbf{X} \begin{bmatrix} 3 & -2 & 1 \\ -2 & -9 & 2 \\ -2 & -1 & 9 \end{bmatrix} = \begin{bmatrix} -2 & 1 & -1 \\ 4 & 1 & 2 \\ 5 & -6 & 1 \\ 6 & -4 & -4 \\ -6 & 6 & -3 \end{bmatrix}$$

21. 试求出下面矩阵方程的解析解并验证得出的结果。

$$\begin{bmatrix} -2 & 2 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 2 \end{bmatrix} \mathbf{X} \begin{bmatrix} -2 & -1 & 2 \\ 1 & 3 & 0 \\ 3 & -2 & 2 \end{bmatrix} - \mathbf{X} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \end{bmatrix} = 0$$

22. 某些函数可以用多项式函数, 如 Taylor 幂级数来表示, 对这些函数来说, 如果用矩阵 \mathbf{A} 去取代自变量 x , 则可以求出矩阵非线性函数的值。对下面的非线性矩阵函数试编写出相应的求解 M-函数, 并和 `funm()` 或 `funmsym()` 函数的结果进行比较。

$$(1) \cos \mathbf{A} = \mathbf{I} - \frac{1}{2!} \mathbf{A}^2 + \frac{1}{4!} \mathbf{A}^4 - \frac{1}{6!} \mathbf{A}^6 + \cdots + \frac{(-1)^n}{(2n)!} \mathbf{A}^{2n} + \cdots$$

$$(2) \arcsin \mathbf{A} = \mathbf{A} + \frac{1}{2 \cdot 3} \mathbf{A}^3 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5} \mathbf{A}^5 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 7} \mathbf{A}^7 + \frac{1 \cdot 3 \cdot 5 \cdot 7}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \mathbf{A}^9 + \cdots + \frac{(2n)!}{2^{2n}(n!)^2(2n+1)} \mathbf{A}^{2n+1} + \cdots$$

$$(3) \ln \mathbf{A} = \mathbf{A} - \mathbf{I} - \frac{1}{2}(\mathbf{A} - \mathbf{I})^2 + \frac{1}{3}(\mathbf{A} - \mathbf{I})^3 - \frac{1}{4}(\mathbf{A} - \mathbf{I})^4 + \cdots + \frac{(-1)^{n+1}}{n}(\mathbf{A} - \mathbf{I})^n + \cdots$$

23. 已知自治型线性微分方程 $\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t)$ 的解析解可以写成 $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$, 试求出下面自治微分方程的解析解

$$\mathbf{x}'(t) = \begin{bmatrix} -3 & 0 & 0 & 1 \\ -1 & -1 & 1 & -1 \\ 1 & 0 & -2 & 1 \\ 0 & 0 & 0 & -4 \end{bmatrix} \mathbf{x}(t), \quad \mathbf{x}(0) = \begin{bmatrix} -1 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

24. 假设某 Riccati 方程的数学表达式为 $\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0}$, 且已知

$$\mathbf{A} = \begin{bmatrix} -27 & 6 & -3 & 9 \\ 2 & -6 & -2 & -6 \\ -5 & 0 & -5 & -2 \\ 10 & 3 & 4 & -11 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 3 \\ 16 & 4 \\ -7 & 4 \\ 9 & 6 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 6 & 5 & 3 & 4 \\ 5 & 6 & 3 & 4 \\ 3 & 3 & 6 & 2 \\ 4 & 4 & 2 & 6 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 4 & 1 \\ 1 & 5 \end{bmatrix}$$

试求解该方程, 得出 \mathbf{P} 矩阵, 并检验得出解的精度。

25. 假设已知某 Jordan 块矩阵 \mathbf{A} 及其组成部分为

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \mathbf{A}_2 & \\ & & \mathbf{A}_3 \end{bmatrix}, \quad \mathbf{A}_1 = \begin{bmatrix} -3 & 1 & 0 \\ 0 & -3 & 1 \\ 0 & 0 & -3 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} -5 & 1 \\ 0 & -5 \end{bmatrix}, \quad \mathbf{A}_3 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

试用解析解运算的方式得出 $\mathbf{e}^{\mathbf{A}t}, \sin\left(2\mathbf{A}t + \frac{\pi}{3}\right), \mathbf{e}^{\mathbf{A}^2t}\mathbf{A}^2 + \sin(\mathbf{A}^3t)\mathbf{A}t + \mathbf{e}^{\sin \mathbf{A}t}$ 。

26. 假设已知矩阵 $\mathbf{A} = \begin{bmatrix} -4.5 & 0 & 0.5 & -1.5 \\ -0.5 & -4 & 0.5 & -0.5 \\ 1.5 & 1 & -2.5 & 1.5 \\ 0 & -1 & -1 & -3 \end{bmatrix}$, 试求出 $\mathbf{e}^{\mathbf{A}t}, \sin \mathbf{A}t, \mathbf{e}^{\mathbf{A}t} \sin (\mathbf{A}^2\mathbf{e}^{\mathbf{A}t}t)$ 。

参考文献

- [1] Dongarra J J, Bunsh J R, Molor C B. LINPACK user's guide. Philadelphia: Society of Industrial and Applied Mathematics, 1979
- [2] 薛定宇, 陈阳泉. 高等应用数学问题的 MATLAB 求解 (第 2 版). 北京: 清华大学出版社, 2008
- [3] Garbow B S, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide extension, Lecture notes in computer sciences, volume 51. New York: Springer-Verlag, 1977
- [4] Smith B T, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide. Lecture notes in computer sciences. New York: Springer-Verlag, second edition, 1976
- [5] Moler C B, Stewar G W. An algorithm for generalized matrix eigenvalue problems. SIAM Journal of Numerical Analysis, 1973, 10:241–256
- [6] Klema V, Laub A. The singular value decomposition: its computation and some applications. IEEE Transactions on Automatic Control, 1980, AC-25(2):164–176
- [7] 《数学手册》编写组. 数学手册. 北京: 人民教育出版社, 1979
- [8] Beezer R A. A first course in linear algebra, version 2.99. Washington: Department of Mathematics and Computer Science University of Puget Sound, 1500 North Warner, Tacoma, Washington, 98416-1043, <http://linear.ups.edu/>, 2012
- [9] The MathWorks Inc. MATLAB mathematics
- [10] Moler C B, Van Loan C F. Nineteen dubious ways to compute the exponential of a matrix. SIAM Review, 1979, 20:801–836
- [11] 黄琳. 系统与控制理论中的线性代数. 北京: 科学出版社, 1984

第5章 积分变换与复变函数问题的计算机求解

积分变换技术可以将某些难以分析的问题通过映射的方式映射到其他域内的表达式后再进行分析。例如, Laplace 变换可以将时域函数映射成复域函数, 从而可以将某时域函数的微分方程映射成复域的多项式代数方程, 使得原微分方程在诸多方面, 如稳定性、解析解等方面更便于分析, 这样的变换方法构成了经典自动控制理论的基础。在实际应用中, Fourier 变换、Mellin 变换及 Hankel 变换都是有其应用领域的。如何利用计算机求解积分变换的解析解是本章主要介绍的问题之一。5.1 节将首先介绍 Laplace 变换与反变换的定义及基本性质, 然后介绍用 MATLAB 语言中的符号运算工具箱函数求取 Laplace 变换及反变换问题的解析解方法, 还给出了复杂函数 Laplace 反变换的数值求解方法与应用实例。5.2 节将介绍 Fourier 变换及反变换的定义、性质和变换问题的 MATLAB 解法, 并介绍 Fourier 余弦变换、正弦变换、离散 Fourier 正余弦变换等问题的计算机求解方法, 并介绍快速 Fourier 变换的求解与应用。5.3 节将介绍 Mellin 变换、Hankel 变换等问题的 MATLAB 语言的求解算法, 可以得出函数的相应变换及反变换。 z 变换是另一类实用的变换方法, 该变换方法也是离散控制理论的数学基础。5.4 节将介绍 z 变换及其反变换的定义和性质, 并介绍基于 MATLAB 语言符号运算工具箱的 z 变换问题的计算机辅助求解方法。本章的另一个主要问题是复变函数问题及其 MATLAB 语言求解, 可以用 5.5 节中介绍的方法计算复变函数的奇点与留数, 进行部分分式展开等运算, 讨论了有理函数 Laplace 反变换的求解方法和化简方法, 基于留数定理还探讨了封闭曲线积分的求解方法。5.6 节还将介绍各种差分方程的求解方法。

5.1 Laplace 变换及其反变换

法国数学家 Pierre-Simon Laplace (1749 – 1827) 引入的积分变换可以巧妙地将一般常系数微分方程映射成代数方程, 奠定了很多领域, 如电路分析、自动控制原理等的数学模型基础。本节将首先介绍 Laplace 变换及其反变换的定义与性质, 然后介绍利用计算机数学语言 MATLAB 求解 Laplace 变换及其反变换的方法与应用, 最后给出复杂函数 Laplace 反变换的数值求解方法与实用函数。

5.1.1 Laplace 变换及反变换的定义与性质

一个时域函数 $f(t)$ 的 Laplace 变换可以定义为

$$\mathcal{L}[f(t)] = \int_0^{\infty} f(t)e^{-st} dt = F(s) \quad (5-1-1)$$

式中, $\mathcal{L}[f(t)]$ 为 Laplace 变换的简单记号。

下面将不加证明地列出一些常用的 Laplace 变换性质。

(1) **线性性质**。若 a 与 b 均为标量, 则 $\mathcal{L}[af(t) \pm bg(t)] = a\mathcal{L}[f(t)] \pm b\mathcal{L}[g(t)]$ 。

(2) **时域平移性质**。 $\mathcal{L}[f(t-a)] = e^{-as}F(s)$ 。

(3) **s -域平移性质**。 $\mathcal{L}[e^{-at}f(t)] = F(s+a)$ 。

(4) **微分性质**。 $\mathcal{L}\left[\frac{df(t)}{dt}\right] = sF(s) - f(0^+)$, 一般地, n 阶微分可以由下式求出

$$\mathcal{L}\left[\frac{d^n f(t)}{dt^n}\right] = s^n F(s) - s^{n-1}f(0^+) - s^{n-2}\frac{df(0^+)}{dt} - \dots - \frac{d^{n-1}f(0^+)}{dt^{n-1}} \quad (5-1-2)$$

若假设函数 $f(t)$ 及其各阶导数的初值均为 0, 则式 (5-1-2) 可以简化成

$$\mathcal{L}\left[\frac{d^n f(t)}{dt^n}\right] = s^n F(s) \quad (5-1-3)$$

此性质事实上是微分方程映射成代数方程的关键式子。

(5) **积分性质**。若假设零初始条件, $\mathcal{L}\left[\int_0^t f(\tau) d\tau\right] = \frac{F(s)}{s}$, 一般地, 函数 $f(t)$ 的 n 重积分的 Laplace 变换可以由下式求出

$$\mathcal{L}\left[\int_0^t \dots \int_0^t f(\tau) d\tau^n\right] = \frac{F(s)}{s^n} \quad (5-1-4)$$

(6) **初值性质**。 $\lim_{t \rightarrow 0} f(t) = \lim_{s \rightarrow \infty} sF(s)$ 。

(7) **终值性质**。如果 $F(s)$ 没有 $s \geq 0$ 的极点, 则 $\lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s)$ 。

(8) **卷积性质**。 $\mathcal{L}[f(t) * g(t)] = \mathcal{L}[f(t)]\mathcal{L}[g(t)]$, 式中, 卷积算子 $*$ 的定义为

$$f(t) * g(t) = \int_0^t f(\tau)g(t-\tau)d\tau = \int_0^t f(t-\tau)g(\tau)d\tau \quad (5-1-5)$$

(9) **其他性质**。

$$\mathcal{L}[t^n f(t)] = (-1)^n \frac{d^n F(s)}{ds^n}, \quad \mathcal{L}\left[\frac{f(t)}{t^n}\right] = \int_s^\infty \dots \int_s^\infty F(s) ds^n \quad (5-1-6)$$

如果已知函数的 Laplace 变换表达式 $F(s)$, 则可以通过下面的反变换公式反演求出其 Laplace 反变换

$$f(t) = \mathcal{L}^{-1}[F(s)] = \frac{1}{2j\pi} \int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st} ds \quad (5-1-7)$$

其中 σ 大于 $F(s)$ 的奇点实部, 奇点的概念将在后面给出。

5.1.2 Laplace 变换的计算机求解

求解已知函数的 Laplace 变换用数值编程的方式是无法实现的, 如果需要采用计算机来求解这样的问题, 必须借助于计算机数学语言, 如本书介绍的 MATLAB 语言。MATLAB

语言的符号运算工具箱可以轻松地求解 Laplace 变换问题。具体的变换及反变换问题的求解步骤为：

- (1) 用 `syms` 命令申明符号变量 t , 这样就能描述时域表达式 f 了。
- (2) 直接调用 `laplace()` 函数, 就可以得出所需的时域函数 Laplace 变换式子

```
F = laplace(f)           % 采用默认的  $t$  为时域变量
F = laplace(f, v, u)     % 用户指定时域变量  $v$  和复域变量名  $u$ 
```

还可以考虑采用 `simple()` 等函数对其进行化简。

(3) 对复杂的问题来说, 得出的结果形式通常需要调用 `simple()` 函数化简, 此外, 有时变换的结果难以阅读, 所以需要调用 `pretty()` 函数或 `latex()` 函数对结果进行进一步处理。可以在屏幕上或利用 L^AT_EX 的强大功能将结果用可读性更强的形式显示出来。

如果已知 Laplace 变换式子, 则应该首先给出 Laplace 变换式子 f , 然后采用符号运算工具箱中的 `ilaplace()` 函数对其进行反变换。该函数的调用格式为

```
f = ilaplace(F)          % 采用默认的  $s$  为时域变量
f = ilaplace(F, u, v)    % 用户指定时域变量  $v$  和复域变量名  $u$ 
```

获得变化式 f 后也可以对之进一步化简和改变显示格式。

例 5-1 已知函数 $f(t) = t^2 e^{-2t} \sin(t + \pi)$, 试求取该函数的 Laplace 变换。

解 分析原题, 可以先申明 t 为符号变量, 再用 MATLAB 语句表示给定的 $f(t)$ 函数, 然后就可以用下面的语句对该函数进行 Laplace 变换

```
>> syms t; f=t^2*exp(-2*t)*sin(t+pi); F=simple(laplace(f))
```

直接得出原函数的 Laplace 变换为 $F(s) = -\frac{2(3s^2 + 12s + 11)}{(s^2 + 4s + 5)^3}$ 。

例 5-2 假设给出的函数为 $f(x) = x^2 e^{-2x} \sin(x + \pi)$, 试求其 Laplace 变换, 并对结果进行 Laplace 反变换, 看是否能变换回原函数。

解 同样可以采用 `laplace()` 函数求解该问题

```
>> syms x w; f=x^2*exp(-2*x)*sin(x+pi); F=laplace(f,x,w)
f1=simple(ilaplace(F))
```

可见, 这样的结果和上面的例子是完全一致的, 但需要按要求做一下变量替换。

使用 Laplace 反变换的函数 `ilaplace(F)` 得出原函数 $-t^2 e^{-2t} \sin t$, 因为 $\sin(t + \pi) = -\sin t$ 。

例 5-3 试求出下面函数的 Laplace 反变换。

$$G(x) = \frac{-17x^5 - 7x^4 + 2x^3 + x^2 - x + 1}{x^6 + 11x^5 + 48x^4 + 106x^3 + 125x^2 + 75x + 17}$$

解 从原来给出的问题, 似乎用下面的语句就能直接求解出所需结果

```
>> syms x t;
G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)/(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17);
f=ilaplace(G,x,t)
```

得出的结果可读性很差。事实上, 这样的问题因为方程 $x^6 + 11x^5 + 48x^4 + 106x^3 + 125x^2 + 75x + 17 = 0$

不存在解析解,所以导致原问题不存在解析解。若利用 MATLAB 的变精度算法 $\text{vpa}(f)$ 则可以得出高精度的数值解,这里因篇幅所限将其解表示为

$$y(t) = -556.2565e^{-3.2617t} + 1.7589e^{-1.0778t} \cos 0.6021t + 10.9942e^{-1.0778t} \sin 0.6021t \\ + 0.2126e^{-0.5209t} + 537.2850e^{-2.5309t} \cos 0.3998t - 698.2462e^{-2.5309t} \sin 0.3998t$$

例 5-4 对例 5-1 给出的原函数 $f(t)$, 试得出 $\mathcal{L}[d^5 f(t)/dt^5]$ 和 $s^5 \mathcal{L}[f(t)]$ 之间的关系。

解 如果想求解这样的问题,可以利用符号运算工具箱中的 $\text{diff}()$ 函数对函数 $f(t)$ 求五阶导数,再进行 Laplace 变换,则

```
>> syms t s; f=t^2*exp(-2*t)*sin(t+pi); F=simple(laplace(diff(f,t,5)))
```

对 $f(t)$ 进行 Laplace 变换,并将变换结果乘以 s^5 ,将得出的结果与前面直接得出的结果相减,可以得到差为 $6s - 48$ 。

```
>> F0=laplace(f); simple(F-s^5*F0)
```

由于二者之差不为 0,所以看起来和式 (5-1-3) 是不同的。这是因为 $f(t)$ 函数有 $f(0) = f'(0) = 0$,但其高阶导数在 $t = 0$ 处的值不为 0,故不满足式 (5-1-3),而满足式 (5-1-2)。由式 (5-1-2) 直接可见,考虑初始条件后,得出的差和上述的差完全一致。

```
>> ss=0; f1=f; for i=4:-1:0, ss=ss-s^i*subs(f1,t,0); f1=diff(f1,t); end, ss
```

例 5-5 试推导出 $\mathcal{L}[d^2 f(t)/dt^2]$ 的微分公式。

解 MATLAB 的符号运算工具箱还可以进行一些简单的 Laplace 变换公式推导。假想导出 $f(t)$ 的二阶导数的 Laplace 变换,首先应该先定义一下 $f(t)$ 函数,这可以通过如下语句实现,并推导出二阶导数的 Laplace 变换公式

```
>> syms t; y=sym('f(t)'); laplace(diff(y,t,2))
```

得出的结果为 $s*(s*\text{laplace}(f(t),t,s)-f(0))-D(f)(0)$,其数学表示为 $s[sF(s) - f(0)] - f'(0)$,展开即可发现,该结果与式 (5-1-2) 中的式子完全一致。当然,该功能可以进一步引申,求出函数 8 阶导数的 Laplace 变换

```
>> Y=collect(laplace(diff(y,t,8)),s)
```

例 5-6 已知 $f(t) = e^{-5t} \cos(2t + 1) + 5$, 试求出 $\mathcal{L}[d^5 f(t)/dt^5]$ 。

解 这个例子是上个例子的引申。若已知某具体函数 $f(t)$,则可以将 $\text{diff}()$ 函数与 $\text{laplace}()$ 函数结合起来使用,这样用下面的 MATLAB 命令则可以得出所需的结果

```
>> syms t; f=exp(-5*t)*cos(2*t+1)+5;
F=laplace(diff(f,t,5)); F=simple(F)
```

其结果为 $\frac{1475 \cos 1 s - 1189 \cos 1 - 24360 \sin 1 - 4282 \sin 1 s}{s^2 + 10s + 29}$ 。对化简后的结果其实还可以采用其他化简方法微调。例如,想将分子多项式合并同类项,则可以给出如下语句

```
>> syms s; F1=collect(F)
```

则将得出的显示结果为

$$F_1(s) = \frac{(1475 \cos 1 - 4282 \sin 1) s - 1189 \cos 1 - 24360 \sin 1}{s^2 + 10s + 29}$$

5.1.3 Laplace 变换问题的数值求解

前面给出了 Laplace 变换的求解函数 `laplace()`, 该函数可以推导出很多时域函数的 Laplace 变换的解析解, 同时也有很多函数不适合用解析解方法求解, 所以应该考虑数值方法求解 Laplace 变换问题。

Juraj Valsa 开发了基于数值方法的 Laplace 反变换的 MATLAB 函数 `INVLAP()` [1,2], 该函数的调用格式为 `[t,y]=INVLAP(f,t0,tf,N,其他参数)`, 其中原函数由含有字符 s 的字符串表示, (t_0, t_f) 为感兴趣的区间, N 为用户选择的计算点数, 用户可以选择不同的 N 值来检验运算的结果。“其他参数”的选取可以参考原函数的联机帮助, 不过这里建议除非特别需要没有必要去人为修改这些默认参数。

例 5-7 试用数值方法重新求解例 5-3 中给出函数的 Laplace 反变换的问题。

解 由前面给出的例题可知, 虽然原问题的解析解是未知的, 可以通过符号运算工具箱相关函数求出高精度的数值解。对同样的问题, 可以利用 MATLAB 语句将其变换成关于 s 的字符串变量 G , 对其进行数值求解, 则可以得出该函数 Laplace 反变换的数值解。和精确的数值解相比, 这个例子的相对误差为 5.826×10^{-5} , 应该满足一般科学运算的要求。

```
>> syms x t;
G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)/(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17);
f=ilaplace(G,x,t); fun=char(subs(G,x,'s'));
[t1,y1]=INVLAP(fun,0.01,5,100); y0=subs(f,t,t1); norm(vpa((y1-y0)./y0))
```

从计算量看, 如果将计算点数从 100 增加到 5000, 采用 `INVLAP()` 函数所需时间为 2.25s, 而采用 `ilaplace()` 与 `subs()` 函数则需时 10.1s, 可见对某些问题来说, 采用数值方法更高效。

在一般应用中, 如果某复杂系统 $G(s)$ 输入信号的 Laplace 变换可求且已知为 $R(s)$, 可以得出输出信号的 Laplace 变换表达式为 $Y(s) = G(s)U(s)$, 则可以通过前面给出的方法得出输出信号的数值解。

例 5-8 考虑已知的 Laplace 变换表达式 $G(s) = \frac{(s^{0.4} + 0.4s^{0.2} + 0.5)}{\sqrt{s}(s^{0.2} + 0.02s^{0.1} + 0.6)^{0.4}(s^{0.3} + 0.5)^{0.6}}$ 。试用数值方法绘制出 $t \in (0.01, 1)$ 区间内的 Laplace 反变换时域函数曲线。

解 该函数不能用 `ilaplace()` 函数求取 Laplace 反变换解析解, 只能采用数值方法求解此问题。选择计算点数 $N = 1000$, 则可以由下面的语句对 $G(s)$ 进行 Laplace 反变换, 得出的时域函数曲线如图 5-1 所示。增加计算点数到 $N = 5000$ 仍然能得出一致的结果, 说明这样得出的结果是正确的

```
>> f='(s^0.4+ 0.4*s^0.2+0.5)/(s^0.2+0.02*s^0.1+0.6)^0.4/(s^0.3+0.5)^0.6';
[t,y]=INVLAP(f,0.01,1,1000); plot(t,y)
```

值得指出的是, 分数阶多项式 $p^\gamma(x)$ 本身的精确解应该对应于无穷级数, 其 Laplace 反变换的解析解是不可能求出的, 必须借助于数值方法来求解原始问题。另外, 该数值算法速度足够快, 实际求解时间只需 0.3s。

如果给出函数 $u(t)$ 的 Laplace 变换的解析解是不存在的, 则可以考虑对原始的问题用数值方法求解输入信号的 Laplace 变换。分析 `INVLAP()` 源程序可见, 该函数主要采用循环结构, 每一个循环步骤内生成一个 s 向量, 根据该向量可以采用数值积分的方法求出输入信

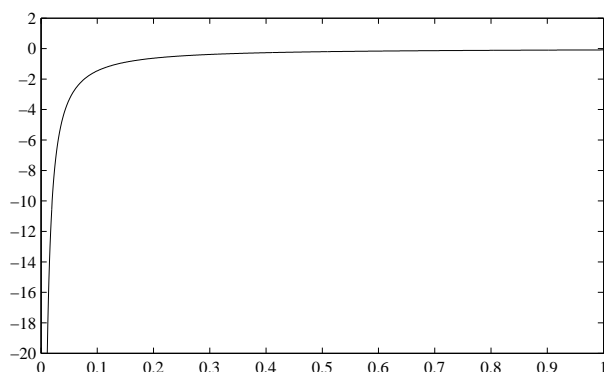


图 5-1 数值 Laplace 反变换时域函数曲线

号的 Laplace 变换

$$\mathcal{L}[u(t)] = \int_0^{\infty} u(t)e^{-st} dt = U(s) \quad (5-1-8)$$

其中 s 为向量。由于积分中含有 e^{-st} 项,在实际应用中只要求解的时间区间 $(0, t_f)$ 足够大,即可以用该有限时间内的积分代替无穷积分,得出近似的数值 Laplace 变换。如果输入信号只是已知一些样本点向量 x_0, u_0 ,则实际 $u(t)$ 信号在 t 时刻的值可以通过插值求出,插值方法在第 8 章中将详细叙述。

该变换与传递函数 $G(s)$ 点乘则可以得出输出信号的数值 Laplace 变换,对其进行数值 Laplace 反变换则可以得出输出信号的数值解。在函数中利用了 MATLAB 8.0 版对向量 s 的数值积分函数 `integral()`,早期版本可以考虑由 `quadv()` 函数替代。

```
function [t,y]=num_laplace(G,t0,tf,nnt,x0,u0)
FF=strrep(strrep(strrep(G,'*','.*'),'/', './'),'^','.^');
a=6; ns=20; nd=19; t=linspace(t0,tf,nnt);
if t0==0, t=t(2:end); nnt=nnt-1; end
n=1:ns+1+nd; alfa=a+(n-1)*pi*1j; beta=-exp(a)*(-1).^n; n=1:nd
bdif=fliplr(cumsum(gamma(nd+1)./gamma(nd+2-n)./gamma(n)))./2^nd;
beta(ns+2:ns+1+nd)=beta(ns+2:ns+1+nd).*bdif; beta(1)=beta(1)/2;
for kt=1:nnt
    tt=t(kt); s=alfa/tt; bt=beta/tt;
    if isnumeric(x0), f=@(x)interp1(x0,u0,x,'spline').*exp(-s.*x);
    else, f=@(x)x0(x).*exp(-s.*x); end
    U=integral(f,t0,tf,'ArrayValued',true);
    btF=bt.*eval(FF).*U; y(kt)=sum(real(btF));
end
```

该函数的调用格式如下,其中 G 为系统的传递函数模型

```
[t,y] = num_laplace(G,t0,tf,N,f),      % 已知输入信号的匿名函数 f
[t,y] = num_laplace(G,t0,tf,N,x0,u0),  % 已知输入的一组采样点 x0、u0
```

例 5-9 假设前面给出的 $G(s)$ 为某系统的分数阶传递函数模型, 试求出该系统在 $u(t) = e^{-0.3t} \sin t^2$ 激励下的响应曲线。

解 分数阶传递函数的输入方法与前面给出的完全一致。将输入信号用匿名函数描述出来, 则可由下面的语句计算出输出信号的曲线, 如图 5-2 所示。该函数内部采用了数值积分运算, 所以耗时比前面介绍的 Laplace 反变换数值算法长得多, 大约需要 40s。如果想解决耗时问题需要在算法上有所突破。

```
>> f=@(t)exp(-0.3*t).*sin(t.^2);
G='(s^0.4+ 0.4*s^0.2+0.5)/(s^0.2+0.02*s^0.1+0.6)^0.4/(s^0.3+0.5)^0.6';
tic, [t,y]=num_laplace(G,0,15,400,f); toc, plot(t,y)
```

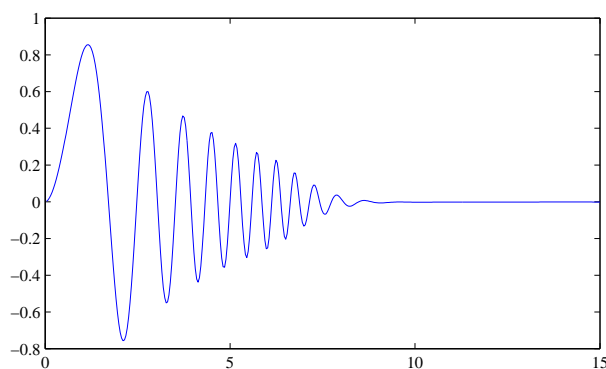


图 5-2 复杂分数阶系统的输出曲线

现在重新考虑输入信号, 假设已知在 $t \in (0, 15)$ 周期内已知的采样点可以直接如下求出, 基于这些样本点可以求出输出信号的数值解, 该解与前面得出的完全一致, 但由于该函数内部采用了插值运算, 所以耗时极长, 计算点数缩减一半以后本例仍需要大约 5 分钟的时间。

```
>> x0=0:0.2:15; u0=exp(-0.3*x0).*sin(x0.^2);
tic, [t,y]=num_laplace(G,0,15,400,x0,u0); toc, plot(t,y)
```

5.2 Fourier 变换及其反变换

5.2.1 Fourier 变换及反变换定义与性质

Fourier 变换的一般定义为

$$\mathcal{F}[f(t)] = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt = F(\omega) \quad (5-2-1)$$

如果已知 Fourier 变换式 $F(\omega)$, 则可以由 Fourier 反变换公式反演出 $f(t)$ 函数为

$$f(t) = \mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{j\omega t} d\omega \quad (5-2-2)$$

这里仍将不加证明地列出一些 Fourier 变换的性质。

(1) 线性性质。若 a 与 b 均为标量, 则 $\mathcal{F}[af(t) \pm bg(t)] = a\mathcal{F}[f(t)] \pm b\mathcal{F}[g(t)]$ 。

(2) 平移性质。 $\mathcal{F}[f(t \pm a)] = e^{\pm j\omega a} F(\omega)$ 。

(3) 复域平移性质。 $\mathcal{F}[e^{\pm jat} f(t)] = F(\omega \mp a)$ 。

(4) 微分性质。 $\mathcal{F}\left[\frac{df(t)}{dt}\right] = j\omega F(\omega)$, 一般地, n 阶微分可以由下式求出

$$\mathcal{F}\left[\frac{d^n f(t)}{dt^n}\right] = (j\omega)^n \mathcal{F}[f(t)] \quad (5-2-3)$$

(5) 积分性质。 $\mathcal{F}\left[\int_{-\infty}^t f(\tau) d\tau\right] = \frac{F(\omega)}{j\omega}$, 一般地, 函数 $f(t)$ 的 n 重积分的 Fourier 变换可以由下式求出

$$\mathcal{F}\left[\int_{-\infty}^t \cdots \int_{-\infty}^t f(\tau) d\tau^n\right] = \frac{\mathcal{F}[f(t)]}{(j\omega)^n} \quad (5-2-4)$$

(6) 尺度变换。 $\mathcal{F}[f(at)] = \frac{1}{a} F\left(\frac{\omega}{a}\right)$ 。

(7) 卷积性质。 $\mathcal{F}[f(t) * g(t)] = \mathcal{F}[f(t)]\mathcal{F}[g(t)]$, 其中, 卷积定义仍为式 (5-1-5)。

5.2.2 Fourier 变换的计算机求解

和 Laplace 变换一样, 应该先申明符号变量, 并定义出原函数为 f , 这样就可以按如下的格式调用 Fourier 变换求解函数 `fourier()`, 得出该函数的 Fourier 变换式

```
F = fourier(f)           % 按默认变量进行 Fourier 变换
F = fourier(f,v,u)       % 将 v 的函数变换成 u 的函数
```

给出了 Fourier 变换表达式, 则可以通过 `ifourier()` 函数求解该函数的 Fourier 反变换问题。该函数的具体调用格式为

```
f = ifourier(F)          % 按默认变量进行 Fourier 反变换
f = ifourier(F,u,v)      % 将 u 的函数变换成 v 的函数
```

从上面的语句可以看出, 如果定义了已知函数, 则可以用一个语句求解出其 Fourier 变换或反变换式, 变换函数的调用和 Laplace 变换一样简单。如果已知的是 MATLAB 得出的 Fourier 变换式, 则可以直接使用 `ifourier()` 函数进行变换。

例 5-10 考虑 $f(t) = 1/(t^2 + a^2)$, $a > 0$, 试写出该函数的 Fourier 变换式。

解 可以用下面的语句得出原函数的 Fourier 变换

```
>> syms t w; syms a positive, f=1/(t^2+a^2); F=fourier(f,t,w)
```

该语句将返回结果 $\pi(e^{-a\omega} \text{Heaviside}(\omega) + e^{a\omega} \text{Heaviside}(-\omega))/a$, 其中得出的 $\text{Heaviside}(\omega)$ 函数为 ω 的阶跃函数, 又称为 Heaviside 函数, 当 $\omega > 0$ 时, 该函数的值为 1, $\omega = 0$ 取 0.5, 否则为 0, 而当 $\omega < 0$ 时, $\text{Heaviside}(-\omega)$ 的值为 1, $\omega = 0$ 时为 0.5, 否则为 0。假设 $\omega > 0$, 则 F 可以简化成 $\pi e^{-a\omega}/a$, 若 $\omega < 0$, 则 F 可以简化成 $\pi e^{a\omega}/a$, 故可以将上述结果写成 $\mathcal{F}[f(t)] = \frac{\pi e^{-a|\omega|}}{a}$ 。

然而,这样的最简表达式是通过 MATLAB 语言的自动化简功能无法得出的。对得出的结果进行 Fourier 反变换,则将还原出原函数 $1/(a^2 + x^2)$ 。

```
>> syms w; syms a positive; f=pi*exp(-a*abs(w))/a;
F=fourier(f), f1=ifourier(F)
```

即使不手工化简,仍可以直接对前面得出的 F 函数进行反变换,并将得出原函数。

例 5-11 假设时域函数为 $f(t) = \sin^2(at)/t, a > 0$, 试求出其 Fourier 变换。

解 由给定的式子,可以用下面的语句获得原函数的 Fourier 变换

```
>> syms t w; syms a positive; f=sin(a*t)^2/t; fourier(f,t,w)
```

这样得出的结果为 $j\pi(\text{Heaviside}(\omega - 2a) + \text{Heaviside}(\omega + 2a) - 2\text{Heaviside}(\omega))/2$ 。可见,该结果仍然依赖于 $\text{Heaviside}()$ 函数,所以理解 $\text{Heaviside}()$ 函数将使得用户能得到更简单的形式。当 $\omega > 2a$, 则 3 个 $\text{Heaviside}()$ 函数的值均为 1,故 $F(\omega) = 0$ 。若 $\omega \leq -2a$, 则 3 个函数的值均为 0,故 $F(\omega) = 0$ 。若 $0 < \omega < 2a$, 则第 2 和第 3 个 $\text{Heaviside}()$ 的值为 1,故 $F(\omega) = -j\pi/2$ 。当 $0 > \omega > -2a$, 则 $F(\omega) = j\pi/2$ 。综上所述,原函数的 Fourier 变换可以化简成

$$\mathcal{F}[f(t)] = \begin{cases} 0, & |\omega| > 2a \\ -j\pi \text{sign}(\omega)/2, & |\omega| < 2a \end{cases}$$

例 5-12 再考虑一个稍微复杂的例子。假设函数为 $f(t) = e^{-a|t|}/\sqrt{|t|}$, 试用 MATLAB 提供的现成函数和直接积分的方法分别求解 Fourier 变换问题。

解 先考虑用现成的函数 $\text{fourier}()$ 对给出的原函数进行变换,可以试图采用下面的语句求取其 Fourier 变换

```
>> syms w t; syms a positive; f=exp(-a*abs(t))/sqrt(abs(t)); F=fourier(f,t,w)
```

很遗憾,该函数无法求取原函数的 Fourier 变换式。还可以考虑从底层进行积分计算,将式 (5-2-1) 给出的积分式分成两段 $(-\infty, 0), [0, \infty)$ 分别求积分,这样可以给出如下的 MATLAB 命令,但这样的方法仍无法得出该函数的 Fourier 变换。

```
>> syms t real; f1=exp(a*t)/sqrt(-t); f2=exp(-a*t)/sqrt(t); j=sym(sqrt(-1));
F=int(f1*exp(-j*w*t),-inf,0)+int(f2*exp(-j*w*t),0,inf)
```

事实上,上述原函数是存在有限 Fourier 变换的,但由于 MATLAB 与底层的 MuPAD、Maple 计算引擎本身存在的问题,目前无法求取该函数的 Fourier 变换。

5.2.3 Fourier 正弦和余弦变换

Fourier 正弦变换的一般定义为

$$\mathcal{F}_s[f(t)] = \int_0^{\infty} f(t) \sin(\omega t) dt = F_s(\omega) \quad (5-2-5)$$

Fourier 余弦变换的一般定义为

$$\mathcal{F}_c[f(t)] = \int_0^{\infty} f(t) \cos(\omega t) dt = F_c(\omega) \quad (5-2-6)$$

MATLAB 语言的符号运算工具箱中并未直接提供余弦 Fourier 变换的函数,所以可以考虑采用符号积分的方法直接求取余弦 Fourier 变换。另外,早期版本的 MATLAB 中,可以调用底层 Maple 函数直接计算给定函数的 Fourier 正弦、余弦变换,具体格式为

```
F = maple('fouriersin',f,t,w)      % 求解 Fourier 正弦变换
F = maple('fouriercos',f,t,w)      % 求解 Fourier 余弦变换
f = maple('invfouriersin',F,w,t)   % 求解 Fourier 反正弦变换
f = maple('invfouriercos',F,w,t)   % 求解 Fourier 反余弦变换
```

其中, `maple()` 函数是 MATLAB 符号运算工具箱中的函数,允许用户直接调用 Maple 中的有关函数, 'fouriersin' 等是 Maple 中的函数名,其余参数是 Maple 中参数的格式。

下面将提供具体例子演示正弦和余弦 Fourier 变换的推导方法。

例 5-13 试求出 $f(t) = t^n e^{-at}$, $a > 0, n = 1, 2, \dots, 8$ 的余弦 Fourier 变换。

解 解决这样的问题可以采用循环结构,对不同的 i 值,可以用直接积分的算法求取 Fourier 余弦变换,并将得出的结果进行化简,如表 5-1 所示。

```
>> syms t w; syms a positive
for i=1:8
    f=t^i*exp(-a*t); F=int(f*cos(w*t),t,0,inf); Fs(i)=simple(F);
end, Fs
```

其实,按照数学手册^[3]中给出的公式,对整数 n ,可以得出

$$\mathcal{F}_c[t^n e^{-at}] = n! \Re \left(\frac{1}{a + j\omega} \right)^{n+1} = n! \left(\frac{a}{a^2 + \omega^2} \right)^{n+1} \sum_{m=0}^{[n/2]} (-1)^m C_{n+1}^{2m+1} \left(\frac{\omega}{a} \right)^{2m+1} \quad (5-2-7)$$

表 5-1 n 取不同值时 Fourier 余弦变换结果

n 值	$\mathcal{F}_c[f(t)]$
1~4	$\frac{a^2 - \omega^2}{(a^2 + \omega^2)^2}, -2\frac{(-a^2 + 3\omega^2)a}{(a^2 + \omega^2)^3}, 6\frac{(-a^2 + 2a\omega + \omega^2)(-a^2 - 2a\omega + \omega^2)}{(a^2 + \omega^2)^4}, 24\frac{a(a^4 - 10a^2\omega^2 + 5\omega^4)}{(a^2 + \omega^2)^5}$
5,6	$-120\frac{(-a + \omega)(a + \omega)(a^2 - 4\omega a + \omega^2)(a^2 + 4\omega a + \omega^2)}{(a^2 + \omega^2)^6}, -720\frac{(-a^6 + 21a^4\omega^2 - 35\omega^4 a^2 + 7\omega^6)a}{(a^2 + \omega^2)^7}$
7	$5040\frac{(a^4 + 4a^3\omega - 6a^2\omega^2 - 4a\omega^3 + \omega^4)(a^4 - 4a^3\omega - 6a^2\omega^2 + 4a\omega^3 + \omega^4)}{(a^2 + \omega^2)^8}$
8	$40320\frac{a(-a^2 + 3\omega^2)(-a^6 + 33a^4\omega^2 - 27a^2\omega^4 + 3\omega^6)}{(a^2 + \omega^2)^9}$

例 5-14 试求取分段函数 $f(t) = \begin{cases} \cos t, & 0 < x < a \\ 0, & \text{其他} \end{cases}$ 的 Fourier 余弦变换。

解 研究 Fourier 余弦变换定义可见,式 (5-2-6) 的被积函数在 $t \in (a, \infty)$ 区间的值为 0,这样,其积分亦为 0,故整个积分问题就变成 $t \in (0, a)$ 区间的积分问题了,故可以用下面的语句求出该函数的 Fourier 余弦变换

```
>> syms t w; syms a positive; f=cos(t); F=simple(int(f*cos(w*t),t,0,a))
```

其结果是分段函数

$$F_c(\omega) = \begin{cases} \frac{a}{2} + \frac{\sin 2a}{4}, & \omega \in (-1, 1) \\ \frac{\sin(a\omega - 1)}{2\omega - 2} + \frac{\sin a(\omega + 1)}{2\omega + 2}, & \text{其他} \end{cases}$$

如果将 f 用分段函数直接表示,也可以得到一致的结果。

```
>> f=piecewise('t<a and t>=0','cos(t)','t>=a','0'); F=int(f*cos(w*t),t,0,inf)
```

5.2.4 离散 Fourier 正弦、余弦变换

离散 Fourier 正弦、余弦变换又称为有限 Fourier 正弦、余弦变换,和前面介绍的 Fourier 正弦、余弦变换相比,其积分区间从 $t \in (0, \infty)$ 变成了 $t \in (0, a)$, 故其定义为

$$F_s(k) = \int_0^a f(t) \sin \frac{k\pi t}{a} dt, \quad F_c(k) = \int_0^a f(t) \cos \frac{k\pi t}{a} dt \quad (5-2-8)$$

相应地,可以定义出有限 Fourier 正弦、余弦反变换为

$$f(t) = \frac{2}{a} \sum_{k=1}^{\infty} F_s(k) \sin \frac{k\pi t}{a} \quad (5-2-9)$$

$$f(t) = \frac{1}{a} F_c(0) + \frac{2}{a} \sum_{k=1}^{\infty} F_c(k) \cos \frac{k\pi t}{a} \quad (5-2-10)$$

和前面定义的不同,反变换不再是积分式子,而是无穷级数的求和。下面通过例子介绍如何用 MATLAB 及其符号运算工具箱求取给定函数的有限变换。

例 5-15 考虑分段函数 $f(t) = \begin{cases} t, & t \leq a/2 \\ a-t, & t > a/2 \end{cases}$, 其中, $a > 0$, 试求其离散 Fourier 正弦变换。

解 函数的离散 Fourier 正弦变换可以由下面的语句直接求出

```
>> syms t k; syms a positive, f1=t; f2=a-t;
Fs=int(f1*sin(k*pi*t/a),t,0,a/2)+int(f2*sin(k*pi*t/a),t,a/2,a);
simple(Fs)
```

因为符号运算工具箱不直接支持整数变量,考虑到 k 为整数,故有 $\sin k\pi \equiv 0$, 所以该方程只能最终手工化简成

$$F_s[f(t)] = \frac{a^2}{k^2\pi^2} \left(2 \sin \frac{k\pi}{2} - \sin k\pi \right) = \frac{2a^2}{k^2\pi^2} \sin \frac{k\pi}{2}$$

如果采用分段函数的方式描述 f 函数,也可以得出完全一致的结果。

```
>> f=piecewise('t<=a/2','t','t>a/2','a-t');
Fs=simple(int(f*sin(k*pi*t/a),t,0,a))
```

5.2.5 快速 Fourier 变换

从前面的叙述看只有一部分简单的函数可以通过各种 Fourier 变换的公式得出其相应的变换表达式,限制了这样解析变换方式在实际中的应用。在实际应用中,更多地采用数值形式直接对信号的采样值进行离散 Fourier 变换。离散数据 $x_i, i = 1, 2, \dots, N$ 的 Fourier 变换是数字信号处理的基础。离散 Fourier 变换的数学表示为

$$X(k) = \sum_{i=1}^N x_i e^{-2\pi j(k-1)(i-1)/N}, \text{ 其中 } 1 \leq k \leq N \quad (5-2-11)$$

其逆变换定义为

$$x(k) = \frac{1}{N} \sum_{i=1}^N X(i) e^{2\pi j(k-1)(i-1)/N}, \text{ 其中 } 1 \leq k \leq N \quad (5-2-12)$$

快速 Fourier 变换(fast Fourier transform, FFT)技术是求解离散 Fourier 变换的最实用、也是最通用的方法。MATLAB 提供了内核函数 `fft()`, 该函数的调用格式很简单, 为 `f = fft(x)` 可以进行 FFT, 而 `x = ifft(f)` 可进行反变换。

MATLAB 提供的 `fft()` 函数可以高效地求解 FFT 问题。该函数的另一个显著特点是它可以对任意长度的向量进行变换, 而不要求所变换的向量长度满足 2^n 约束, 尽管满足这样长度的变换计算速度快些。

例 5-16 假设给定数学函数 $x(t) = 12 \sin(2\pi \times t + \pi/4) + 5 \cos(2\pi \times 4t)$, 选择步长为 h , 对其进行 FFT 变换, 试绘制变换结果的幅值曲线。对得出的结果试用 FFT 反变换的方法, 观察是否能通过反变换还原出所需的信号。

解 对采用的采样周期 h , 时间区间为 $t \in (0, t_f)$, 可以产生 L 个时间值 t_i , 并求出这些点上的函数值为 x_i , 其相应的频率点可以由 $f_0 = 1/(ht_f), 2f_0, 3f_0, \dots$ 构成, 然后可以由语句

```
>> h=0.01; t=0:h:10;
    x=12*sin(2*pi*t+pi/4)+5*cos(2*pi*4*t); X=fft(x); f=t/h/10;
    stem(f(1:floor(length(f)/2)),abs(X(1:floor(length(f)/2)))), xlim([0,10])
```

得出 FFT 幅值与频率的关系, 如图 5-3 (a) 所示。这里仅取一半数据绘制图形的原因是为了避免 FFT 分析的假频(aliasing)现象。分析结果可以看出, 在幅值曲线上有两个峰值点, 对应的频率值为 1 Hz 和 4 Hz, 正是给定函数中的两个频率值。

快速 Fourier 逆变换可以由 `ifft()` 函数直接求解, 误差向量的范数为 $e = 1.029 \times 10^{-13}$

```
>> ix=real(ifft(X)); plot(t,x,t,ix,':'); xlim([0,1]); e=norm(x-ix)
```

这样得出的逆 FFT 变换结果与原函数在图 5-3 (b) 中给出。可以看出二者完全一致。由于采样点较稀疏, 故曲线看起来不是很光滑。

此外, MATLAB 还提供了二维或更高维的 FFT 与逆 FFT 函数。二维问题可以调用 `fft2()` 和 `ifft2()` 函数, 而高维问题可以使用 `fftn()` 和 `ifftn()` 函数。

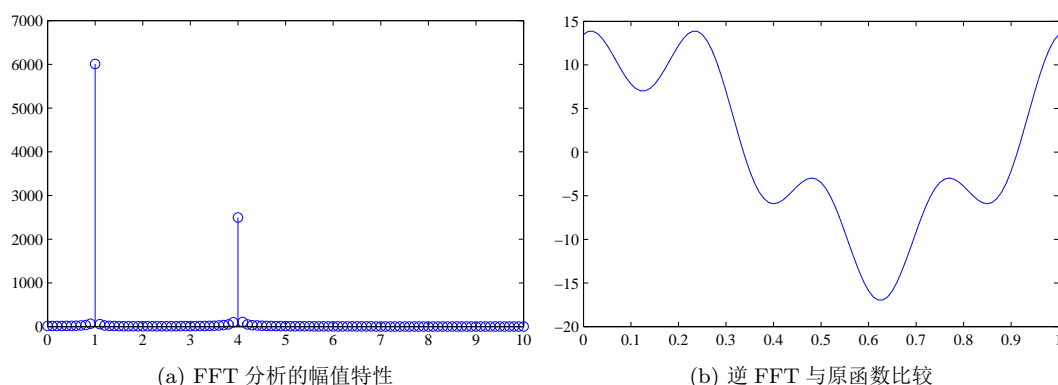


图 5-3 数据的 FFT 分析

5.3 其他积分变换问题及求解

除了 Laplace 变换和各种 Fourier 变换外,在不同的领域还需要各种各样其他的变换,如 Mellin 变换、Hankel 变换等。实践表明,新版本的符号运算工具在求解这两种变换中的表现不很理想,所以建议本节采用 2008a 或以前版本直接求解相应问题。标准的 MATLAB 符号运算工具箱中未直接提供求解这些变换的现成函数,所以解决这个问题仍然有两种方法,其一是采用直接积分的方法,另一种是采用 Maple 语言中的相应函数直接求解。

5.3.1 Mellin 变换

Mellin 变换可以定义为

$$\mathcal{M}[f(x)] = \int_0^{\infty} f(x)x^{z-1}dx = M(z) \quad (5-3-1)$$

相应地,Mellin 反变换可以定义为

$$f(x) = \mathcal{M}^{-1}[M(z)] = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} M(z)x^{-z}dz \quad (5-3-2)$$

MATLAB 符号运算工具箱并没有直接可用的 Mellin 正反变换的函数,但仍可以通过积分的形式计算 Mellin 变换。下面将通过例子解决这类问题。

例 5-17 考虑时域函数 $f(t) = \ln t/(t+a)$, 其中 $a > 0$, 试求其 Mellin 变换。

解 根据定义可以用下面语句求取该函数的 Mellin 变换(注意:新版本无法得出所需的解)。

```
>> syms t z; syms a positive;
f=log(t)/(t+a); M=simple(int(f*t^(z-1),t,0,inf))
```

经过化简,可以立即得出结果 $\mathcal{M}[f(t)] = a^{z-1}\pi(\ln a \sin \pi z - \pi \cos \pi z) \csc^2 \pi z$ 。

例 5-18 假设给定函数 $f(t) = 1/(t+a)^n$, ($a > 0$), 对若干个 n 值求取 Mellin 变换,并总结出对一般 n 值的规律。

解 下面的语句将给出 $n = 1, 2, \dots, 8$ 时 Mellin 变换的结果

```
>> syms t z; syms a positive
    for i=1:8, f=1/(t+a)^i; disp(int(f*t^(z-1),t,0,inf)), end
```

上面的循环结构可以得出如下结果

```
a^{z-1} \pi \csc \pi z
-a^{-2+z} \pi (z-1) \csc \pi z
1/2 a^{-3+z} \pi (-2+z) (z-1) \csc \pi z
-1/6 a^{z-4} \pi (z-1) (-2+z) (-3+z) \csc \pi z
1/24 a^{-5+z} \pi (z-4) (-3+z) (-2+z) (z-1) \csc \pi z
-1/120 a^{-6+z} \pi (-5+z) (z-4) (-3+z) (-2+z) (z-1) \csc \pi z
1/720 a^{-7+z} \pi (-6+z) (-5+z) (z-4) (-3+z) (-2+z) (z-1) \csc \pi z
-1/5040 a^{-8+z} \pi (-7+z) (-6+z) (-5+z) (z-4) (-3+z) (-2+z) (z-1) \csc \pi z
```

所以,可以总结出一般的 Mellin 变换规律为

$$\mathcal{M} \left[\frac{1}{(t+a)^n} \right] = \frac{(-1)^{k-1} \pi}{(n-1)!} a^{z-n} \prod_{i=1}^{n-1} (z-i) \csc \pi z$$

早期版本的 MATLAB 可以调用 Maple 下的底层函数直接求取 Mellin 正反变换问题,相应的函数为 `mellin()` 和 `inv mellin()`,这些函数在符号运算工具箱中未给出 MATLAB 实现,但可以通过其支持的语句直接调用 Maple 语言中现成的函数得出所需的结果。这两个函数的具体调用格式为

```
F = maple('mellin',f,t,z)      % 求解 Mellin 变换
f = maple('inv mellin',F,z,t)  % 求解 Mellin 反变换
```

例 5-19 仍考虑例 5-18 中给出的函数,若 $n=8$,试利用 Maple 语言中的相应函数求取该函数的 Mellin 变换,并对结果进行反变换。

解 利用 Maple 语言中提供的函数,可以用下面的语句直接得出该函数的 Mellin 变换

```
>> syms t z; syms a positive
F=maple('mellin',1/(t+a)^8,t,z), f1=maple('inv mellin',F,z,t)
```

这样可以得出 Mellin 变换结果 $-\frac{\pi(z-1)!}{5040(-8+z)!} a^{-8+z} \csc \pi z$ 。

这里的反变换命令得出了一个很长的结果,但仍无法变换回原来给定的函数,也无法得出原问题的有意义的解。

5.3.2 Hankel 变换及求解

Hankel 变换是另一类常用的数学变换, ν 阶 Hankel 变换的数学表达式为

$$\mathcal{H}[f(t)] = \int_0^\infty t f(t) J_\nu(\omega t) dt = H_\nu(\omega) \quad (5-3-3)$$

其中, $J_\nu(\cdot)$ 为 Bessel 函数。还可以定义 ν 阶 Hankel 反变换公式为

$$\mathcal{H}^{-1}[H(\omega)] = \int_0^\infty \omega H_\nu(\omega) J_\nu(\omega t) d\omega \quad (5-3-4)$$

早期版本的 MATLAB 可以借助于 Maple 语言中的 `hankel()` 和 `invhankel()` 函数求取给定函数的 Hankel 正反变换,其调用格式为

```
F = maple('hankel',f,t,w,v)    % 求解 Hankel 变换
f = maple('invhankel',F,w,t,v) % 求解 Hankel 反变换
```

例 5-20 求取 $f(t) = t^a$ 函数的 0 阶 Hankel 变换。

解 利用 Maple 中的函数可以用下面语句在 MATLAB 环境中立即写出 Hankel 变换的结果

```
>> syms t w; syms a positive, f=t^a; F=maple('hankel',f,t,w,0)
```

即可以得出如下的变换结果

$$\mathcal{H}[f(t)] = \frac{\omega^{-1-a} \sin(\pi(2a+3)/4) 2^{a+1/2} \Gamma^2(a/2+3/4)}{\pi}$$

利用 Maple 语言中现有的 Hankel 变换功能,目前只适合于求解 t^a 型函数的正 Hankel 变换,其他函数的 Hankel 变换用现有的函数不一定能计算出最简的解析结果。例如 $f(t) = a/(a^2+t^2)^{3/2}$ 这样的函数用现有的函数求解 0 阶 Hankel 变换,则有

```
>> syms t x w; syms a positive; F=maple('hankel',a/(a^2+t^2)^(3/2),t,w,0)
```

将得出如下的结果

$$F = \frac{\sqrt{w} \operatorname{Hyp}([3/4], [1/4, 1], wa^2/4) \Gamma^2(3/4)}{\sqrt{a\pi}} - \frac{4aw^2\pi \operatorname{Hyp}([3/2], [7/4, 7/4], wa^2/4)}{9\Gamma^2(3/4)}$$

其中, $\operatorname{Hyp}(\cdot)$ 为广义超几何函数,其内容超出本书范围,在此不予详细介绍。而事实上, $f(t)$ 函数的 0 阶 Hankel 变换即为 $e^{-a\omega}$,故在得出的结果上有很大的差距。所以无论是利用 MATLAB 语言,还是利用现有的 Maple 等计算机数学语言,目前尚不具备进行 Hankel 正反变换的能力,应该用查表的方法得出所需的结果。

5.4 z 变换及其反变换

严格说来, z 变换并不属于积分变换,但由于其定义、性质和求解方法也类似于 Laplace 变换,并且该方法可以在描述序列信号中起重要作用,所以本节将介绍 z 变换及其求解方法,并将给出通过长除法求取 z 反变换的数值方法的 MATLAB 实现。

5.4.1 z 变换及反变换定义与性质

离散序列信号 $f(k), k = 1, 2, \dots$ 的 z 变换可以定义为

$$\mathcal{Z}[f(k)] = \sum_{k=0}^{\infty} f(k)z^{-k} = F(z) \quad (5-4-1)$$

类似于前面介绍的 Laplace 变换和 Fourier 变换, z 变换也有很多类似的性质,这里仍不加以证明地列出一些性质如下:

- (1) **线性性质**。若 a 与 b 均为标量,则 $\mathcal{Z}[af(k) \pm bg(k)] = a\mathcal{Z}[f(k)] \pm b\mathcal{Z}[g(k)]$ 。
- (2) **后向平移性质**。 $\mathcal{Z}[f(k-n)] = z^{-n}F(z)$ 。

(3) 前向平移性质。对非零初值的问题,前向平移的 z 变换可以由下式计算

$$\mathcal{Z}[f(k+n)] = z^n F(z) - \sum_{i=0}^{n-1} z^{n-i} f(i) \quad (5-4-2)$$

特别地,对零初值问题有 $\mathcal{Z}[f(k+n)] = z^n F(z)$ 。

(4) s -域比例性质。 $\mathcal{Z}[r^{-k} f(k)] = F(rz)$ 。

(5) 频域微分性质。 $\mathcal{Z}[kf(k)] = -z \frac{dF(z)}{dz}$ 。

(6) 频域积分性质。 $\mathcal{Z}[f(k)/k] = \int_z^\infty \frac{F(\omega)}{\omega} d\omega$ 。

(7) 初值性质。 $\lim_{k \rightarrow 0} f(k) = \lim_{z \rightarrow \infty} F(z)$ 。

(8) 终值性质。如果 $F(z)$ 无单位圆外的极点,则 $\lim_{k \rightarrow \infty} f(k) = \lim_{z \rightarrow 1} (z-1)F(z)$ 。

(9) 卷积性质。 $\mathcal{Z}[f(k) * g(k)] = \mathcal{Z}[f(k)] \mathcal{Z}[g(k)]$, 式中离散信号的卷积算子 $*$ 定义为

$$f(k) * g(k) = \sum_{l=0}^{\infty} f(k)g(k-l) \quad (5-4-3)$$

给定 z 变换式子 $F(z)$, 则其 z 反变换的数学表示为

$$f(k) = \mathcal{Z}^{-1}[f(k)] = \frac{1}{2\pi j} \oint F(z) z^{k-1} dz \quad (5-4-4)$$

5.4.2 z 变换的计算机求解

利用 MATLAB 的符号运算工具箱,则 z 变换及其反变换可以很容易地求取出来,掌握这样的工具可以免除复杂问题的手工推导,既节省时间也能避免底层的低级错误。利用符号运算工具箱中提供的 `ztrans()` 和 `iztrans()` 函数可以得出给定函数的正反 z 变换。这两个函数的调用格式为

$F = \text{ztrans}(f, k, z)$ % z 变换,将 k 的函数变换成 z 的函数

$F = \text{iztrans}(f, z, k)$ % z 反变换,将 z 的函数变换成 k 的函数

若原函数只有一个变量,则调用时无需给出 k 和 z 。

例 5-21 求解 $f(kT) = akT - 2 + (akT + 2)e^{-akT}$ 函数的 z 变换问题。

解 原函数的 z 变换可以用下面的语句来完成

```
>> syms a T k; f=a*k*T-2+(a*k*T+2)*exp(-a*k*T); F=ztrans(f)
```

该结果可以表示为

$$\mathcal{Z}[f(kT)] = \frac{aTz}{(z-1)^2} - \frac{2z}{z-1} + \frac{aTze^{-aT}}{(z-e^{-aT})^2} + 2ze^{aT} \left(\frac{z}{e^{-aT}} - 1 \right)^{-1}$$

例 5-22 考虑 $F(z) = q/(z^{-1} - p)^m$ 函数的 z 反变换问题,这里可以对不同的 m 值进行反变换,并总结出一般规律。

解 根据要求,可以用符号运算工具箱求出 $m = 1, 2, \dots, 8$ 的 z 反变换

```
>> syms p q z; for i=1:8, disp(simple(iztrans(q/(1/z-p)^i))), end
```

对不同的 i 值循环可以得出如下结果

$$\begin{aligned} & -q/p(1/p)^n \\ & q/p^2(1+n)(1/p)^n \\ & -1/2q(1/p)^n(1+n)(2+n)/p^3 \\ & 1/6q(1/p)^n(3+n)(2+n)(1+n)/p^4 \\ & -1/24q(1/p)^n(4+n)(3+n)(2+n)(1+n)/p^5 \\ & 1/120q(1/p)^n(5+n)(4+n)(3+n)(2+n)(1+n)/p^6 \\ & -1/720q(1/p)^n(6+n)(5+n)(4+n)(3+n)(2+n)(1+n)/p^7 \\ & 1/5040q(1/p)^n(7+n)(6+n)(5+n)(4+n)(3+n)(2+n)(1+n)/p^8 \end{aligned}$$

总结上述结果的规律,可以写出一般的 z 反变换结果为

$$\mathcal{Z}^{-1} \left[\frac{q}{(z^{-1} - p)^m} \right] = \frac{(-1)^m q}{(m-1)! p^{n+m}} \prod_{i=1}^{m-1} (n+i)$$

5.4.3 双边 z 变换

前面叙述的 z 变换由于描述的是 $n \geq 0$ 时序列信号的性质,所以又称为单边 z 变换,如果将 n 的范围扩展到整个整数集合,则可以定义出双边 z 变换

$$\mathcal{Z}[f(k)] = \sum_{k=-\infty}^{\infty} f(k)z^{-k} = F(z) \quad (5-4-5)$$

MATLAB 中并未提供双边 z 变换的求解函数,但可以从定义直接求出给定函数 f 的双边 z 变换表达式 `F = symsum(f*z^(-k),k,0,inf) + symsum(f*z^(-k),k,-inf,-1)`。MATLAB 的 `symsum()` 函数有时不支持 $(-\infty, \infty)$ 区间的求和。

例 5-23 试求出分段函数 $f(n)$ 的双边 z 变换^[4], 其中 $f(n) = \begin{cases} 2^n, & n \geq 0 \\ -3^n, & n < 0 \end{cases}$ 。

解 采用前面介绍的方法,整个求和可以分成两个区间单独计算,可以由下面的语句直接得出函数的双边 z 变换。下面语句将得出 $F = \frac{z}{z-2} + \frac{z}{z-3}$ 。

```
>> syms z n; F=symsum(2^n*z^(-n),n,0,inf)+symsum(-3^n*z^(-n),n,-inf,-1)
```

5.4.4 有理函数 z 反变换的数值求解

很多函数 z 反变换的解析解是不能由 `iztrans()` 求出的,即使对某些有理函数也不能解析地求出其 z 反变换。假设一般有理函数 z 变换表达式可以写成

$$F(z^{-1}) = z^{-d} \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{m-1} z^{-(m-1)} + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n-1} z^{-(n-1)} + a_n z^{-n}} \quad (5-4-6)$$

将该函数作关于 z^{-k} 的幂级数展开则可以写出

$$F(z^{-1}) = f_0 + f_1 z^{-1} + f_2 z^{-2} + \cdots = \sum_{k=0}^{\infty} f_k z^{-k} \quad (5-4-7)$$

上式恰巧是 z 变换的定义式。可以通过长除法展开 $F(z^{-1})$, 从而得出 $F(z^{-1})$ 函数 z 反变换的数值解。我们编写了长除法函数, 其调用格式为 **$y = \text{inv_z}(\text{num}, \text{den}, d, N)$** , 其中, d 为纯延迟的步数, 系数向量 $\text{num} = [b_0, b_1, b_2, \cdots, b_m]$ 、 $\text{den} = [a_0, a_1, a_2, \cdots, a_n]$, 默认的计算点数为 $N = 10$ 。应用数值方法求出的序列信号由 y 向量返回。

```
function y=inv_z(num,den,d,N)
if nargin==2, d=0; end, if nargin<=3, N=10; end, num(N)=0;
for i=1:N-d, y(d+i)=num(1)/den(1);
    if length(num)>1, ii=2:length(den);
        if length(den)>length(num); num(length(den))=0; end
        num(ii)=num(ii)-y(end)*den(ii); num(1)=[];
    end, end
```

例 5-24 试用数值方法求 $G(z) = \frac{z^2 + 0.4}{z^5 - 4.1z^4 + 6.71z^3 - 5.481z^2 + 2.2356z - 0.3645}$ 的 z 反变换。

解 原函数分子和分母同时乘以 z^{-5} 则可以得出下面所需的表达式

$$F(z^{-1}) = z^{-3} \frac{1 + 0.4z^{-2}}{1 - 4.1z^{-1} + 6.71z^{-2} - 5.481z^{-3} + 2.2356z^{-4} - 0.3645z^{-5}}$$

这样由下面的语句可以直接得出原函数的 z 反变换, 得出的序列如图 5-4 所示。

```
>> num=[1 0 0.4]; den=[1 -4.1 6.71 -5.481 2.2356 -0.3645];
N=50; y=inv_z(num,den,3,N); t=0:(N-1); stem(t,y)
```

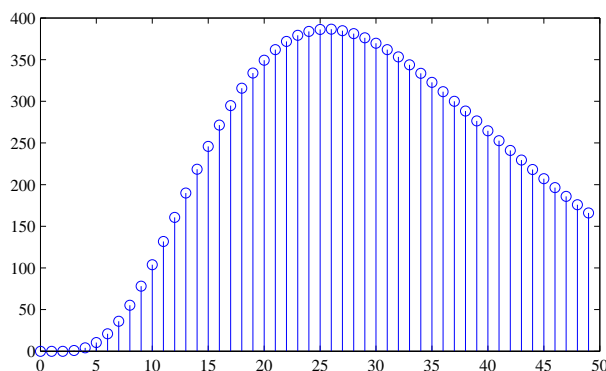


图 5-4 z 反变换的数值解

5.5 复变函数问题的计算机求解

5.5.1 复数矩阵及其变换

前面介绍过, MATLAB 可以用来直接表示复数矩阵。假设已知一个复数矩阵 Z , 则可以使用简单函数对该矩阵进行如下变换:

- (1) 共轭复数矩阵 $Z_1 = \text{conj}(Z)$ 。
- (2) 实部、虚部提取 $R = \text{real}(Z)$, $I = \text{imag}(Z)$ 。
- (3) 幅值、相位表示 $A = \text{abs}(Z)$, $P = \text{angle}(Z)$, 其中相位的单位为弧度。

例 5-25 重新考虑例 4-35 中的 Jordan 标准型问题。由于原矩阵存在复数特征值, 所以需要用手工的方法修改变换矩阵。这里采用下面语句修改变换矩阵, 也能起到同样的作用。

```
>> A=[1,0,4,0; 0,-3,0,0; -2,2,-3,0; 0,0,0,-2]; [V,D]=eig(sym(A));
      V=real(V)+imag(V), D1=inv(V)*A*V
```

5.5.2 复变函数的映射

若某函数 $f(z)$ 的自变量 z 为复数, 则该函数称为复变函数。由于复数是 MATLAB 的最基本的数据结构, 在大多数算法中均未特别区分实数和复数, 所以前面介绍的绝大多数内容均可以直接用于复变函数的分析。例如, 前面的微积分运算的解析解和数值解均可以直接用于复变函数的微积分运算。

例 5-26 已知某复变函数为 $f(z) = \frac{z^2 + 3z + 4}{(z-1)^5}$, 其中 z 为复数变量, 试求出 $f^{(3)}(-j\sqrt{5})$ 的值。

解 由下面语句可以立即得出所需的结果为 $d_3 = 0.8150 - j0.6646$ 。

```
>> syms z; f=(z^2+3*z+4)/(z-1)^5; f3=diff(f,z,3); d3=subs(f3,z,-sqrt(-5))
```

在复变函数中一种很重要的数学变换形式是映射, 即函数可以从一个变量 z 变换成另一个变量 w 的函数, 其中 $z = g(w)$ 为给定的函数。经常使用的映射是平移映射 $z = w + \gamma$, 反演映射 $z = 1/w$ 和双线性映射 $z = (aw + b)/(cw + d)$, 这里, γ 为给定复数, a, b, c, d 为给定实数。其中平移映射将函数的原点平移到 γ 点; 反演映射可以将单位圆内外的点相互映射, 而双线性变换实现直线与圆的相互映射。求解函数映射的最直接的 MATLAB 函数是 `subs()`, 下面通过例子演示函数的映射。

例 5-27 考虑例 5-26 中的复变函数 $f(z) = \frac{z^2 + 3z + 4}{(z-1)^5}$, 试得出 $z = \frac{s-1}{s+1}$ 下的映射函数 $F(s)$ 。

解 映射问题由 `subs()` 函数直接得出, 得出的结果需要化简

```
>> syms z s; f=(z^2+3*z+4)/(z-1)^5; F=simple(subs(f,z,(s-1)/(s+1)))
```

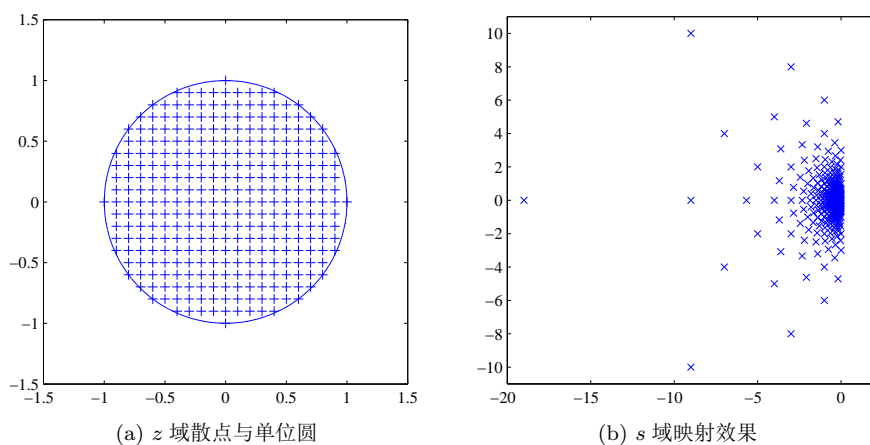
该语句可以直接得出映射函数为 $F(s) = -\frac{1}{16}(s+1)^3(4s^2 + 3s + 1)$ 。

下面的语句还可以绘制出 $s = (z-1)/(z+1)$ 的映射效果, 单位圆内如图 5-5(a) 所示的点可以通过映射, 映射成左半平面的点, 如图 5-5(b) 所示。

```
>> [x,y]=meshgrid(-1:0.1:1); ii=find(x.^2+y.^2<=1); x=x(ii); y=y(ii);
      z=x+sqrt(-1)*y; plot(z,'+'); hold on; ezplot('x^2+y^2=1')
      figure; s=(z-1)./(z+1); plot(s,'x')
```

5.5.3 Riemann 面绘制

复变函数映射的三维图形表示和实函数是不一致的, 复变函数映射应该首先采用

图 5-5 z 域到 s 域的映射示意图

`cplxgrid()` 函数生成极坐标网格, 用户可以根据给定的单值复变函数公式计算出变量 f , 然后由 `cplxmap()` 函数绘制该复变函数的映射曲面, 这类曲面又称为 Riemann 面。这些函数具体调用格式为

`z = cplxgrid(n);` 给出语句计算 f ; `cplxmap(z, f)`

例 5-28 试绘制出复变函数 $f(z) = z^3 \sin z^2$ 的映射曲面。

解 由下面的语句可以立即绘制出相应的复变函数的映射曲面, 如图 5-6 所示。

```
>> z=cplxgrid(50); f=z.^3.*sin(z.^2); cplxmap(z,f)
```

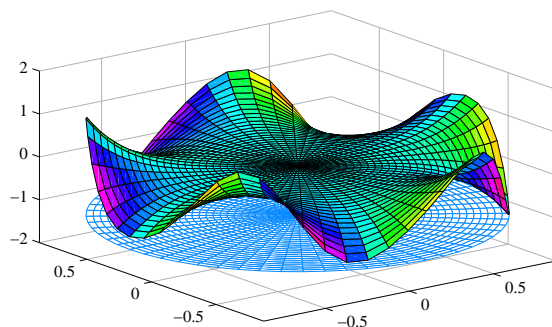


图 5-6 复变函数的 Riemann 面

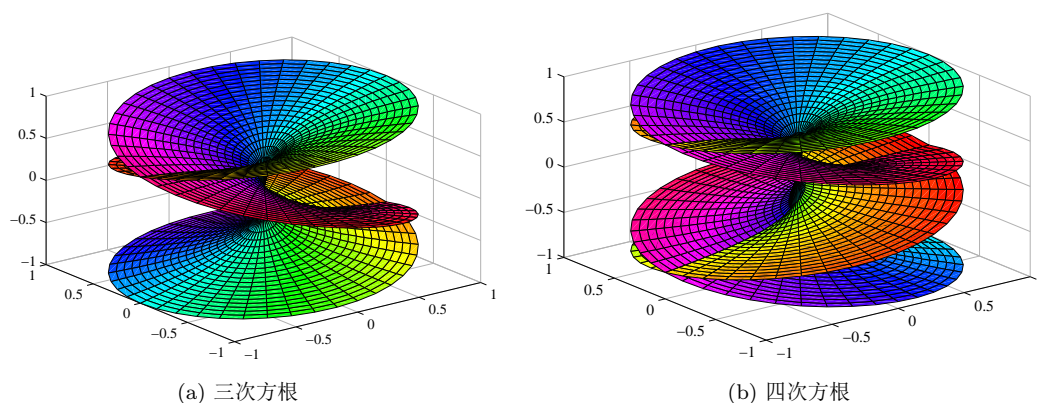
对于复数变量 z , 多值复变函数的 Riemann 面可能有多个分支, 比如 $f(z) = \sqrt[n]{z}$ 就有 n 个分支。MATLAB 提供了绘制其各次方根 Riemann 曲面的函数 `cplxroot(n)`, 可以直接绘制出 $\sqrt[n]{z}$ 的 Riemann 曲面。

例 5-29 试绘制 $\sqrt[3]{z}$ 和 $\sqrt[4]{z}$ 的 Riemann 曲面。

解 由 `cplxroot()` 函数可以直接绘制出 $\sqrt[3]{z}$ 和 $\sqrt[4]{z}$ 的 Riemann 面, 如图 5-7(a)、(b) 所示。

```
>> cplxroot(3), figure, cplxroot(4)
```

从上面给出的 `cplxroot()` 函数可见, 其局限性在于只能绘制出方根函数的 Riemann

图 5-7 $\sqrt[n]{z}$ 方根函数的 Riemann 面

面,对其他类型的多值复变函数无能为力,所以我们可以考虑扩展 `cplxmap()` 函数:将该函数另存为 `cplxmap1()` 函数,再删除掉其中的 `mesh()` 函数和 `hold` 语句,这样就可以考虑多值复变函数 Riemann 面的绘制了。

例 5-30 利用上述思路重新绘制 $\sqrt[3]{z}$ 的 Riemann 面。

解 首先考虑重新绘制 $\sqrt[3]{z}$ 函数的 Riemann 面。我们知道,如果一个函数 $f_1(z)$ 是 $f(z) = \sqrt[3]{z}$ 的一个分支,则另两个分支可以由 $f_1(z)e^{-2j\pi/3}$ 和 $f_1(z)e^{-4j\pi/3}$ 求出。这样可以由下面语句直接绘制 $\sqrt[3]{z}$ 的 Riemann 面,与图 5-7(a) 给出的完全一致。

```
>> z=cplxgrid(30); f1=z.^(1/3); a=exp(-2i*pi/3); cplxmap1(z,f1)
    hold on; cplxmap1(z,a*f1); cplxmap1(z,a^2*f1); zlim([-1 1])
```

5.5.4 留数的概念与计算

在介绍留数概念之前,应该先介绍一下复变函数解析的概念。若函数 $f(z)$ 在复平面的区域内各点处均为单值,且其导数为有限值,则称 $f(z)$ 在复平面内为解析的,这样就将单值函数上不解析的点称为奇点。假设 $z = a$ 为 $f(z)$ 函数上的奇点,若存在一个最小整数 m 使得乘积 $(z - a)^m f(z)$ 在 $z = a$ 点处解析,则称 $z = a$ 为 m 重奇点。

若 $z = a$ 为 $f(z)$ 函数的单奇点,则函数在该奇点处的留数(residue)可以定义为

$$\text{Res}[f(z), a] = \lim_{z \rightarrow a} (z - a) f(z) \quad (5-5-1)$$

若 $z = a$ 为函数 $f(z)$ 的 m 重奇点,则该点的留数定义为

$$\text{Res}[f(z), a] = \lim_{z \rightarrow a} \frac{1}{(m-1)!} \frac{d^{m-1}}{dz^{m-1}} \left[f(z)(z-a)^m \right] \quad (5-5-2)$$

利用 MATLAB 的符号运算工具箱求留数的方法很简单。假设已知奇点 a 和重数 m ,则用下面的 MATLAB 语句自然可以求出相应的留数。

```
c = limit(F*(z-a),z,a) % 单奇点
c = limit(diff(F*(z-a)^m,z,m-1)/prod(1:m-1),z,a) % m 重奇点
```

例 5-31 试求出函数 $f(z) = \frac{1}{z^3(z-1)} \sin\left(z + \frac{\pi}{3}\right) e^{-2z}$ 的留数。

解 对原函数的分析可见, $z=0$ 是三重奇点, $z=1$ 是单奇点, 故可以直接使用下面的 MATLAB 语句将这两个奇点处的留数分别求出。可以得出 $z=0$ 处的留数为 $-\frac{\sqrt{3}}{4} + \frac{1}{2}$, $z=1$ 处的留数为 $\frac{1}{2}e^{-2}\sin 1 + \frac{\sqrt{3}}{2}e^{-2}\cos 1$ 。

```
>> syms z; f=sin(z+pi/3)*exp(-2*z)/(z^3*(z-1))
F1=limit(diff(f*z^3,z,2)/prod(1:2),z,0), F2=limit(f*(z-1),z,1)
```

例 5-32 试求函数 $f(z) = \frac{\sin z - z}{z^6}$ 的留数。

解 乍看该函数很容易认定 $z=0$ 为 6 重奇点, 所以用下面的语句很容易就可以求出该点处的留数值, 其值为 $1/120$ 。

```
>> syms z; f=(sin(z)-z)/z^6; limit(diff(f*z^6,z,5)/prod(1:5),z,0)
```

其实这里说 $z=0$ 为 6 重奇点有些太保守, 不严格地说, 从 $k=1$ 开始尝试, 能够使得

$$\lim_{z \rightarrow a} \frac{d^{k-1}}{dz^{k-1}} [(z-a)^k f(z)] < \infty$$

成立的最小的 k 就是奇点的重数, 记作 m , 可以考虑 $k=2$, 可见导数 F_1 为无穷大, 所以再试验更大的 k 值。对此例子来说, k 的最小值为 $m=3$, 留数的值 F_2 仍然为 $1/120$ 。试凑更大的 k 值, 如 $k=20$, 也不会改变求出的留数值, $F_3 = 1/120$ 。

```
>> syms z; f=(sin(z)-z)/z^6;
F1=limit(diff(f*z^2,z,1)/prod(1:1),z,0)
F2=limit(diff(f*z^3,z,2)/prod(1:2),z,0) % 再增加阶次
F3=limit(diff(f*z^20,z,19)/prod(1:19),z,0) % 再进一步增加阶次
```

可见, 若选择的 n 值大于或等于奇点的实际重数, 则可以正确得到该函数的留数。在一般应用时可选择一个较大的 n 值来求取留数。

例 5-33 试求出函数 $f(z) = \frac{1}{z \sin z}$ 的留数。

解 分析该函数, 因为 $\sin z$ 在 $z=0$ 点的收敛速度和 z 是一样的, 显然, $z=0$ 点为 $f(z)$ 的二重奇点, 这时, 相应的留数可以用下面语句求出 $c_0 = 0$ 。

```
>> syms z; f=1/(z*sin(z)); c0=limit(diff(f*z^2,z,1),z,0)
```

进一步分析给定函数 $f(z)$, 可以发现该函数在 $z = \pm k\pi$ 处均不解析, 其中 k 为正整数, 且这些点是原函数的单奇点, 由于 MATLAB 的符号运算工具箱并未给出整数的定义, 所以这里只能对一些 k 值进行试探, 求出它们的留数, 最后将结果归纳成所需的公式。

```
>> k=[-4 4 -3 3 -2 2 -1 1]; c=[];
for kk=k; c=[c,limit(f*(z-kk*pi),z,kk*pi)]; end; c
```

对向量 $k = [-4, 4, -3, 3, -2, 2, -1, 1]$, 可以得出留数为 $c = [-1/(4\pi), 1/(4\pi), 1/(3\pi), -1/(3\pi), -1/(2\pi), 1/(2\pi), 1/\pi, -1/\pi]$ 。综上, 可以归纳出 $\text{Res}[f(z), \pm k\pi] = \pm(-1)^k/(k\pi)$ 。

5.5.5 有理函数的部分分式展开

考虑有理函数

$$G(x) = \frac{B(x)}{A(x)} = \frac{b_1x^m + b_2x^{m-1} + \cdots + b_mx + b_{m+1}}{x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n} \quad (5-5-3)$$

其中, a_i 和 b_i 均为常数。有理函数的互质概念是一个非常重要的概念。所谓互质, 就是指多项式 $A(x)$ 和 $B(x)$ 没有公约数。对一般给定的两个多项式来说, 用手工方式判定多项式互质还是比较困难的, 但利用 MATLAB 符号运算工具箱中的 `gcd()` 函数可以直接求出两个多项式的最大公约数。该函数的调用方法为 `C = gcd(A, B)`, 其中, A 和 B 分别表示两个多项式, 该函数将得出这两个多项式的最大公约数 C , 若得出的 C 为多项式, 则两个多项式为非互质的多项式, 这时两个多项式可以约简为 A/C 和 B/C 。

例 5-34 给出两个多项式 $A(x) = x^4 + 7x^3 + 13x^2 + 19x + 20$, $B(x) = x^7 + 16x^6 + 103x^5 + 346x^4 + 655x^3 + 700x^2 + 393x + 90$, 试判定它们是否互质。

解 求解这样的问题可以采用 MATLAB 语言提供的 `gcd()` 函数完成

```
>> syms x; A=x^4+7*x^3+13*x^2+19*x+20;
```

```
B=x^7+16*x^6+103*x^5+346*x^4+655*x^3+700*x^2+393*x+90; d=gcd(A,B)
```

可见, 两个多项式具有最大公约数 $d = x + 5$, 故两个多项式不是互质的, 这两个多项式可以进一步简化为 $(x^3 + 2x^2 + 3x + 4)/[(x+2)(x+3)^2(x+1)^3]$ 。

```
>> simple(A/d), simple(B/d)
```

若互质多项式 $A(x) = 0$ 的根均为相异的值 $-p_i$, $i = 1, 2, \cdots, n$, 则可以将 $G(x)$ 函数写成下面的部分分式展开形式

$$G(x) = \frac{r_1}{x + p_1} + \frac{r_2}{x + p_2} + \cdots + \frac{r_n}{x + p_n} \quad (5-5-4)$$

其中 r_i 为留数, $r_i = \text{Res}[G(x), -p_i]$, 其值可以由下面的极限式求出

$$r_i = \text{Res}[G(x), -p_i] = \lim_{x \rightarrow -p_i} (x + p_i) G(x) \quad (5-5-5)$$

如果分母多项式中含有 $(x + p_i)^k$ 项, 亦即 $-p_i$ 为 k 重根, 则相对这部分特征值的部分分式展开项可以写成

$$\frac{r_i}{x + p_i} + \frac{r_{i+1}}{(x + p_i)^2} + \cdots + \frac{r_{i+k-1}}{(x + p_i)^k} \quad (5-5-6)$$

这时, r_{i+j-1} 可以用下面的公式直接求出, 其中一次项系数为留数, 其他为普通系数。

$$r_{i+j-1} = \frac{1}{(j-1)!} \lim_{x \rightarrow -p_i} \frac{d^{j-1}}{dx^{j-1}} \left[(x + p_i)^k G(x) \right], \quad j = 1, 2, \cdots, k \quad (5-5-7)$$

MATLAB 语言中给出了现成的数值函数 `residue()` 求取有理函数 $G(x)$ 的部分分式展开表示, 该函数的调用格式为 `[r, p, k] = residue(b, a)`, 其中, $a = [1, a_1, a_2, \cdots, a_n]$,

$\mathbf{b}=[b_1, b_2, \dots, b_m]$, 返回的 \mathbf{r} 和 \mathbf{p} 向量为式(5-5-4)的 r_i, p_i 系数, 若有重根则应该相应地由式(5-5-6)中给出的系数取代。 k 为余项, 对 $m < n$ 的函数来说该项为空矩阵。该函数并未给出 $-p_i$ 是否为重根的自动判定功能, 所以部分分式展开的结果需要手动写出。值得指出的是, 数值运算对含有重奇点的问题经常会导致不精确的结果。

例 5-35 试求下面有理函数的部分分式展开。

$$G(s) = \frac{s^3 + 2s^2 + 3s + 4}{s^6 + 11s^5 + 48s^4 + 106s^3 + 125s^2 + 75s + 18}$$

解 用下面的语句可以求出该函数的部分分式展开

```
>> n=[1,2,3,4]; d=[1,11,48,106,125,75,18]; format long
[r,p,k]=residue(n,d); [n,d1]=rat(r); [n,d1,p]
```

其中, \mathbf{p} 为奇点向量, \mathbf{n}, \mathbf{d}_1 为每个 \mathbf{p} 值对应系数的分子和分母数值。由数值方法直接求出的分母多项式的根为小数, 有一些误差。事实上, 该分母多项式的特征值为: -3 为二重奇点, -2 为单奇点, -1 为三重奇点。分析奇点的情况, 可以写出部分分式展开为

$$G(s) = -\frac{17}{8(s+3)} - \frac{7}{4(s+3)^2} + \frac{2}{s+2} + \frac{1}{8(s+1)} - \frac{1}{2(s+1)^2} + \frac{1}{2(s+1)^3}$$

例 5-36 写出下面式子的部分分式展开。

$$G(s) = \frac{2s^7 + 2s^3 + 8}{s^8 + 30s^7 + 386s^6 + 2772s^5 + 12093s^4 + 32598s^3 + 52520s^2 + 45600s + 16000}$$

解 采用 MATLAB 自带的 `residue()` 函数, 只能求解得出数值解, 对本例给出的问题, 可以用下面的语句直接求出有理函数的部分分式展开式

```
>> n=[2,0,0,0,2,0,0,8];
d=[1,30,386,2772,12093,32598,52520,45600,16000]; [r,p]=residue(n,d)
```

从得出的结果较难判定重根情况, 故难以准确写出部分分式展开式。由得出的数据, 假定 $p_1 = -5$ 为三重实根, $p_2 = -4$ 为三重实根, $p_3 = -2, p_4 = -1$ 为单个实根, 可以写出部分分式展开的表达式为

$$G_1(s) = \frac{49995.9030930686}{(s+5)} + \frac{28488.5832580441}{(s+5)^2} + \frac{13040.9999762507}{(s+5)^3} - \frac{50473.1527861460}{(s+4)} \\ + \frac{21449.5555022347}{(s+4)^2} - \frac{5481.3333201362}{(s+4)^3} + \frac{1.2222222224}{(s+2)} + \frac{0.0023148148}{(s+1)}$$

应该指出, 上面给出的展开方式在分母上作了近似, 实际数值方法得出的分母不精确。另外, MATLAB 数值运算在处理重根问题中经常会导致不精确的结果。所以对这样的问题来说, 可以考虑编写更好的解析算法。

新版的 MATLAB 符号运算工具箱配备的 MuPAD 提供了部分分式展开的底层函数 `partfrac()`, 其调用格式为 `F = feval(symengine, 'partfrac', f)`。我们也为其编写了简单的同名接口函数, 其调用格式为 `F = partfrac(f)` 或 `F = partfrac(f, s)`, 其中, 接口函数 `partfrac()` 的内容为

```
function Y=partfrac(varargin)
Y=feval(symengine, 'partfrac', varargin{:});
```

例 5-37 考虑例 5-35 中给出的函数 $f(s)$, 试用解析方式求出其部分分式展开。

解 用下面的语句可立即得出该函数的部分分式展开式, 该结果与原例中数值结果完全一致。

```
>> syms s; f=(s^3+2*s^2+3*s+4)/(s^6+11*s^5+48*s^4+106*s^3+125*s^2+75*s+18);
    G1=partfrac(f), % 或 G2=feval(symengine,'partfrac',f)
```

得出的结果为

$$G_1(s) = -\frac{17}{8(s+3)} - \frac{7}{4(s+3)^2} + \frac{2}{s+2} + \frac{1}{8(s+1)} - \frac{1}{2(s+1)^2} + \frac{1}{2(s+1)^3}$$

例 5-38 仍考虑例 5-36 中给出的有理函数 $G(s)$, 试用解析方法写出其部分分式展开。

解 原函数可以由 `residue()` 函数进行部分分式展开。若将得出的部分分式展开减去原函数并化简, 则结果为 0, 表示得出的结果是正确的。

```
>> syms s, G=(2*s^7+2*s^3+8)/... % 其中 ... 表示续行
    (s^8+30*s^7+386*s^6+2772*s^5+12093*s^4+32598*s^3+52520*s^2+45600*s+16000);
    f=partfrac(G); simple(f-G)
```

这样, 可以得出原分式的部分分式展开如下, 经检验该展开式与原式完全一致。

$$\frac{13041}{(s+5)^3} + \frac{341863}{12(s+5)^2} + \frac{7198933}{144(s+5)} - \frac{16444}{3(s+4)^3} + \frac{193046}{9(s+4)^2} - \frac{1349779}{27(s+4)} + \frac{11}{9(s+2)} + \frac{1}{432(s+1)}$$

例 5-39 考虑例 5-34 中的非互质多项式构成的有理函数 $G(x) = A(x)/B(x)$, 试用数值方法和解析方法写出其部分分式展开。

解 用部分分式展开函数可以直接得出所需的展开, 因为得出的最大公约数对应的展开项系数为 0, 会被直接忽略, 所以无需事先求出公分母。

```
>> syms x; B=x^7+16*x^6+103*x^5+346*x^4+655*x^3+700*x^2+393*x+90;
    A=x^4+7*x^3+13*x^2+19*x+20; F=partfrac(A/B)
```

这样得出的解为

$$F(x) = \frac{A(x)}{B(x)} = -\frac{7}{4(x+3)^2} - \frac{17}{8(x+3)} + \frac{2}{(x+2)} + \frac{1}{2(x+1)^3} - \frac{1}{2(x+1)^2} + \frac{1}{8(x+1)}$$

用前面介绍的解析解方法可以得出和 $x = -5$ 奇点完全无关的解析解, 亦即 `partfrac()` 函数同样适合于非互质有理函数的部分分式展开。

例 5-40 求有理函数的部分分式展开。

$$G(x) = \frac{-17x^5 - 7x^4 + 2x^3 + x^2 - x + 1}{x^6 + 11x^5 + 48x^4 + 106x^3 + 125x^2 + 75x + 17}$$

解 可以试图由 `partfrac()` 函数直接对原函数进行部分分式展开, 然而因为原有理函数的分母不能进行因式分解, 所以不能得出精确的部分分式展开表达式, 该函数的调用也没有任何结果。后面将介绍近似的部分分数展开方法。

```
>> syms x; G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)...
    /(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17); G1=partfrac(G)
```

如果原有理函数分母 $D(x)$ 的某无理数根为 x_0 , 通过任何的求根算法根本不能在有限

位内得出 x_0 的精确值, 只能得到其近似值 \hat{x}_0 , 所以代入式 (5-5-2) 的留数公式则有

$$\text{Res}[f(x), \hat{x}_0] = \lim_{x \rightarrow \hat{x}_0} \frac{1}{(m-1)!} \frac{d^{m-1}}{dx^{m-1}} [(x - \hat{x}_0)^m f(\hat{x}_0)] \quad (5-5-8)$$

假设用 `vpa()` 函数能求出多项式方程所有的根 $x_i, i = 1, 2, \dots, n$, 可以由这些根重组多项式的分母而取代原来的分母, 则能得出新的函数 $f_1(x)$ 来取代式 (5-5-8) 中的 $f(x)$, 这样就能确保 $f_1(x)$ 和 $(x - \hat{x}_0)^m$ 在 \hat{x}_0 处能真正相消, 得出原函数的留数。在新版本下实际求取近似的部分分数展开系数时采用变量替换的方法而不能采用极限方法。该函数的清单为

```
function f=partfrac1(F,s)
f=sym(0); if nargin==1, syms s; end, [num,den]=numden(F); x0=solve(den);
[x,ii]=sort(double(x0)); x0=x0(ii); x=[x0; rand(1)];
kvec=find(diff(double(x))~=0); ee=x(kvec); kvec=[kvec(1); diff(kvec(:,1))];
a0=limit(den/s^length(x0),s,inf); F1=num/(a0*prod(s-x0));
for i=1:length(kvec), for j=1:kvec(i),
    m=kvec(i); s0=ee(i); k=subs(diff(F1*(s-s0)^m,s,j-1),s,s0);
    f=f+k/(s-s0)^(m-j+1)/factorial(j-1);
end, end
```

该函数的调用格式为 $f = \text{partfrac1}(F, s)$, 其中, F 为有理函数的解析表达式, s 为自变量。返回的结果 f 是部分分式展开的表达式。

例 5-41 重新考虑例 5-40 中有理函数的部分分式展开问题。

解 由于原有理函数不能进行严格的因式分解, 所以前面介绍的部分分数展开函数 `partfrac()` 对此问题无能为力, 这里可以采用 `partfrac1()` 函数进行处理

```
>> syms x; G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)...
/(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17); F=partfrac1(G,x)
```

得出的部分分式展开为 (为排版需要这里只保留了有限几位有效数字)

$$F(s) = \frac{0.2125568}{x + 0.5208596} + \frac{0.879464926 + 5.497076258j}{x + 1.07775887 + 0.602106591j} + \frac{268.642522 - 349.1231095j}{x + 2.53094582 + 0.399763j} \\ + \frac{556.25653069}{x + 3.261731} + \frac{0.879464926 - 5.497076258j}{x + 1.07775887 - 0.602106591j} + \frac{268.642522 + 349.1231095j}{x + 2.53094582 - 0.399763j}$$

5.5.6 基于部分分式展开的 Laplace 反变换

符号运算工具箱中提供的 Laplace 反变换函数 `ilaplace()` 可以较好地解决一般函数的 Laplace 反变换问题。但带有复特征值的有理函数的 Laplace 反变换问题不适合由该函数直接求解, 这已经在例 5-3 中说明了, 直接用该函数求解出的解可读性很差。

观察例 5-40 的结果可以立即发现, 若某项部分分式展开式为 $(a + jb)/(s + c + jd)$, 则一定会含有其共轭项 $(a - jb)/(s + c - jd)$, 这样就需要对下面的式子进行化简

$$(a + jb)e^{(c+jd)t} + (a - jb)e^{(c-jd)t} = ae^{ct} \sin(dt + \phi) \quad (5-5-9)$$

其中, $\alpha = -2\sqrt{a^2 + b^2}$, 且 $\phi = -\arctan(b/a)$ 。

有了这样的算法, 则可以用 MATLAB 语言编写出实现上述算法的数值函数 `pfrac()`。该函数基于 MATLAB 的原始 `residue()` 函数, 其内容为

```
function [R,P,K]=pfrac(num,den)
[R,P,K]=residue(num,den);
for i=1:length(R),
    if imag(P(i))>eps, a=real(R(i)); b=imag(R(i));
        R(i)=-2*sqrt(a^2+b^2); R(i+1)=-atan2(a,b);
    elseif abs(imag(P(i)))<eps, R(i)=real(R(i));
end, end
```

该函数的调用格式为 `[r,p,K]=pfrac(num,den)`, 其中, p 和 K 的定义和 `residue()` 函数一致, r 稍有不同, 若相应的 p_i 项为实数, 则 r_i 和 `residue()` 一致, 若某个 p_i 的值为复数, 则 r_i, r_{i+1} 项分别为相应的 α 和 ϕ 值。

例 5-42 重新考虑例 5-40 中的问题, 可以用 MATLAB 数值算法得出如下结果

```
>> num=[-17,-7,2,1,-1,1]; den=[1,11,48,106,125,75,17];
[r,p,k]=pfrac(num,den); format long e; [r,p]
```

这样, 由得出的结果可以写出有理函数的可读性更好的 Laplace 反变换为

$$\mathcal{L}^{-1}[F(s)] = -556.2565e^{-3.2617t} - 881.0352e^{-2.5309t} \sin(0.3998t - 0.6559) \\ + 0.2126e^{-0.5209t} - 11.1340e^{-1.0778t} \sin(0.6021t - 2.9829)$$

比较此结果与例 5-3 中的结果, 显而易见这个结果更简洁。

5.5.7 封闭曲线积分问题计算

考虑如下定义的曲线积分

$$\oint_{\Gamma} f(z)dz \quad (5-5-10)$$

其中, Γ 为二维平面内的走行方向为逆时针的封闭曲线, 如果积分线为顺时针的, 则应该将被积函数乘以 -1 。这时假设该封闭曲线逆时针包围 m 个奇点 p_i ($i = 1, 2, \dots, m$), 则可以分别用前面的算法求出这些奇点上的留数为 $\text{Res}[f(z), p_i]$, 这时封闭曲线积分的值等于 $2\pi j$ 乘以这些留数的和, 即

$$\oint_{\Gamma} f(z)dz = 2\pi j \sum_{i=1}^m \text{Res}[f(z), p_i] \quad (5-5-11)$$

例 5-43 试求函数 $f(z)$ 在 $|z| = 6$ 逆时针曲线上的曲线积分。其中, 函数 $f(z)$ 为

$$f(z) = \frac{2z^7 + 2z^3 + 8}{z^8 + 30z^7 + 386z^6 + 2772z^5 + 12093z^4 + 32598z^3 + 52520z^2 + 45600z + 16000}$$

解 可以用例 5-38 中给出的方法求出其部分分式展开为

$$\frac{13041}{(s+5)^3} + \frac{341863}{12(s+5)^2} + \frac{7198933}{144(s+5)} - \frac{16444}{3(s+4)^3} + \frac{193046}{9(s+4)^2} - \frac{1349779}{27(s+4)} + \frac{11}{9(s+2)} + \frac{1}{432(s+1)}$$

可见,该函数的奇点为: $p_1 = -1$ 为单奇点, $p_2 = -2$ 亦为单奇点, $p_3 = -4$, $p_4 = -5$ 均为 3 重奇点。由上面的部分分式展开式可知,各个奇点的留数为部分分式展开的一次项的系数,这样可以得出所需的封闭曲线积分值为

$$\oint_{|z|=6} f(z)dz = 2\pi j \left[\frac{1798933}{144} - \frac{1349779}{27} + \frac{11}{9} + \frac{1}{432} \right] = 4\pi j$$

由第3章介绍的曲线积分方法,可以直接求出该积分。积分路径 Γ 由圆 $|z| = 6$ 给出,可以表示为 $z = 6\cos t + j6\sin t, t \in [0, 2\pi]$, 用下面语句直接积分也可以同样得出 $I = j4\pi$

```
>> syms z t; G=(2*z^7+2*z^3+8)/(z^8+30*z^7+386*z^6+2772*z^5+12093*z^4+...
32598*z^3+52520*z^2+45600*z+16000);
F=subs(G,z,6*cos(t)+6*sin(t)*sqrt(-1));
I=int(F*diff(6*cos(t)+6*sin(t)*sqrt(-1),t),t,0,2*pi)
```

如果要求解的问题是 $|z| = 3$ 的逆时针曲线积分,则在前面的求和式子中去除 p_3, p_4 两个奇点的留数(因为这两个奇点在封闭曲线外),可以最终得出曲线积分的值为 $I = 529j\pi/216$ 。用下面的语句直接求曲线积分也可以得出同样的结果

```
>> F=subs(G,z,3*cos(t)+3*sin(t)*sqrt(-1));
I=int(F*diff(3*cos(t)+3*sin(t)*sqrt(-1),t),t,0,2*pi)
```

例 5-44 试求出下面的曲线积分^[5] $\oint_{|z|=2} \frac{1}{(z+j)^{10}(z-1)(z-3)} dz$ 。

解 经过简单观察原函数,可以发现该函数在 $z = 1, z = 3$ 处有单个奇点,在 $z = -j$ 处有一个 10 重奇点,又因为给定的封闭曲线 Γ 为 $|z| = 2$ 正向圆周,所以 $z = 1, z = -j$ 在该圆周包围的范围内, $z = 3$ 在该圆外,不必计算该留数,这样,原曲线积分的值可以用下面的语句直接求出,其值为 $(237/312500000 + j779/78125000)\pi$ 。

```
>> i=sym(sqrt(-1)); syms z; f=1/((z+i)^10*(z-1)*(z-3)); % 定义被积函数
r1=limit(diff(f*(z+i)^10,z,9)/prod(1:9),z,-i);
r2=limit(f*(z-1),z,1); a=2*pi*i*(r1+r2)
```

根据文献 [5] 中给出的方法,手工求解时建议先计算 $z = 3$ 的留数,故得出整个环路积分值为 $-\pi j/(3+j)^{10}$ 。二者之差为 0,说明两个结果是完全一致的。

```
>> a+pi*i/(3+i)^10
```

由直接曲线积分方法也可以得出同样的结果

```
>> F=subs(f,z,2*cos(t)+2*sin(t)*sqrt(-1))
I=int(F*diff(2*cos(t)+2*sin(t)*sqrt(-1),t),t,0,2*pi)
```

若曲线 Γ 的方程为 $|z| = 4$,则该曲线将 $z = 1, 3, -j$ 三个奇点均包围在内,这时曲线积分应该和这三个留数的和有关,故可以用下面的语句求出曲线积分的值为 0。

```
>> r3=limit(f*(z-3),z,3); b=2*pi*i*(r1+r2+r3)
```

既然用符号运算的方法计算留数特别简单,所以没有必要再用间接的方法计算了,可以通过式(5-5-11)中给出的算法直接求出。该结果与直接曲线积分方法完全一致。


```
>> F=subs(f,z,4*cos(t)+4*sin(t)*sqrt(-1))
I=int(F*diff(4*cos(t)+4*sin(t)*sqrt(-1),t),t,0,2*pi)
```

利用变量替换的方法可以将开区间积分变换成封闭曲线积分。例如,有的书上用变换的方法求解 $f(x) = \sin x/x$ 在 $(0, \infty)$ 区域的无穷积分。而事实上,这样的积分直接用 MATLAB 中的 `int()` 函数更容易求解,所以本书不介绍这类方法。

5.6 差分方程的求解

常系数线性差分方程的一般形式为

$$\begin{aligned} y[(k+n)T] + a_1y[(k+n-1)T] + a_2y[(k+n-2)T] + \cdots + a_ny(kT) \\ = b_1u[(k-d)T] + b_2u[(k-d-1)T] + \cdots + b_mu[(k-d-m+1)T] \end{aligned} \quad (5-6-1)$$

其中 T 为采样周期。和微分方程描述的连续系统类似,这里的系数 a_i 和 b_i 也是常数,所以这类系统称为线性时不变离散系统。另外,对应系统的输入信号和输出信号也可以由 $u(kT)$ 和 $y(kT)$ 表示。 $u(kT)$ 为第 k 个采样周期的输入信号, $y(kT)$ 为该时刻的输出信号。为方便起见,简记 $y(t) = y(kT)$,且记 $y[(k+i)T]$ 为 $y(t+i)$,则前面的差分方程可以简记为

$$\begin{aligned} y(t+n) + a_1y(t+n-1) + a_2y(t+n-2) + \cdots + a_ny(t) \\ = b_1u(t+m-d) + b_2u(t+m-d-1) + \cdots + b_{m+1}u(t-d) \end{aligned} \quad (5-6-2)$$

5.6.1 一般差分方程的解析求解方法

前面给出了线性常系数差分方程的一般形式,若信号的初值 $y(0), y(1), \cdots, y(n-1)$ 含有非零元素,则对式 (5-6-2) 两边进行 z 变换可以得出

$$\begin{aligned} z^n Y(z) - \sum_{i=0}^{n-1} z^{n-i} y(i) + a_1 z^{n-1} Y(z) - a_1 \sum_{i=0}^{n-2} z^{n-i} y(i) + \cdots + a_n Y(z) \\ = z^{-d} \left[b_1 z^m U(z) - b_1 \sum_{i=0}^{m-1} z^{n-i} u(i) + \cdots + b_{m+1} U(z) \right] \end{aligned} \quad (5-6-3)$$

由此得出

$$Y(z) = \frac{(b_1 z^m + b_2 z^{m-1} + \cdots + b_{m+1}) z^{-d} U(z) + E(z)}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n} \quad (5-6-4)$$

其中, $E(z)$ 为输入、输出信号初值按式 (5-4-2) 中的变换计算出来的表达式

$$E(z) = \sum_{i=0}^{n-1} z^{n-i} y(i) - a_1 \sum_{i=0}^{n-2} z^{n-i} y(i) - a_2 \sum_{i=0}^{n-3} z^{n-i} y(i) - \cdots - a_n z y(0) + \hat{u}(n) \quad (5-6-5)$$

其中

$$\hat{u}(n) = -b_1 \sum_{i=0}^{m-1} z^{n-i} u(i) - \cdots - b_m z u(0) \quad (5-6-6)$$

对 $Y(z)$ 进行 z 反变换则可以得出差分方程的解析解 $y(t)$ 。根据前面的算法,可以编写出一般差分方程的通用求解函数

```
function y=diff_eq(A,B,y0,U,d)
E=0; n=length(A)-1; syms z; if nargin==4, d=0; end
m=length(B)-1; u=iztrans(U); u0=subs(u,0:m-1);
for i=1:n, E=E+A(i)*y0(1:n+1-i)*[z.^(n+1-i:-1:1)].'; end
for i=1:m, E=E-B(i)*u0(1:m+1-i)*[z.^(m+1-i:-1:1)].'; end
Y=(poly2sym(B,z)*U*z^(-d)+E)/poly2sym(A,z); y=iztrans(Y);
```

其调用语句为 $y = \text{diff_eq}(A, B, y_0, U, d)$, 其中, A 、 B 向量分别表示差分方程左侧和右侧的系数向量, U 为输入信号的 z 变换表达式, y_0 给出输出信号的初值向量, d 为延迟步数, 其默认值为 0。调用该函数可以直接获得差分方程的解析解。该函数也可用于非首一化的差分方程。下面将给出具体实例来演示一般差分方程的求解方法。

例 5-45 试求解差分方程

$$48y(n+4) - 76y(n+3) + 44y(n+2) - 11y(n+1) + y(n) = 2u(n+2) + 3u(n+1) + u(n)$$

其中, $y(0) = 1, y(1) = 2, y(2) = 0, y(3) = -1$, 且输入 $u(n) = (1/5)^n$, 试求出差分方程的解析解。

解 由给出的问题可以直接提取出 A 、 B 向量, 将初始输出向量和输入信号送给计算机, 再调用 $\text{diff_eq}()$ 直接求解给出的差分方程

```
>> syms z n; u=(1/5)^n; U=ztrans(u);
y=diff_eq([48 -76 44 -11 1],[2 3 1],[1 2 0 -1],U)
n0=0:20; y0=subs(y,n,n0); stem(n0,y0)
```

可以得出差分方程的解为 $y(n) = \frac{432}{5} \left(\frac{1}{3}\right)^n - \frac{26}{5} \left(\frac{1}{2}\right)^n - \frac{752}{5} \left(\frac{1}{4}\right)^n + \frac{175}{3} \left(\frac{1}{5}\right)^n - \frac{42}{5} \left(\frac{1}{2}\right)^n (n-1)$,

其图形显示如图 5-8 所示, 给出的几个初始点均在得出的解中。

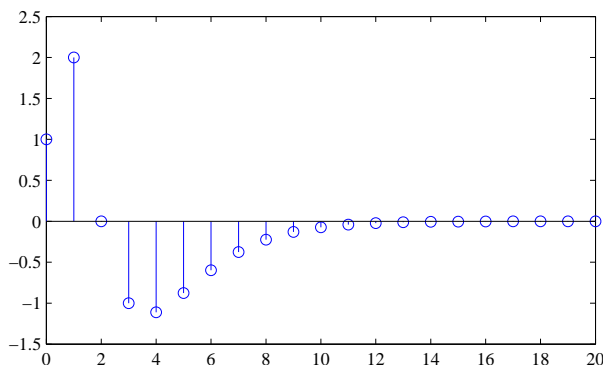


图 5-8 差分方程解的曲线表示

5.6.2 线性时变差分方程的数值解法

线性时变差分状态方程一般可以写成

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) \\ \mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{D}(k)\mathbf{u}(k) \end{cases}, \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (5-6-7)$$

可见,采用递推方法,则

$$\mathbf{x}(1) = \mathbf{F}(0)\mathbf{x}_0 + \mathbf{G}(0)\mathbf{u}(0)$$

$$\mathbf{x}(2) = \mathbf{F}(1)\mathbf{x}(1) + \mathbf{G}(1)\mathbf{u}(1) = \mathbf{F}(1)\mathbf{F}(0)\mathbf{x}_0 + \mathbf{F}(1)\mathbf{G}(0)\mathbf{u}(0) + \mathbf{G}(1)\mathbf{u}(1)$$

⋮

最终可以直接得出

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{F}(k-1)\mathbf{F}(k-2)\cdots\mathbf{F}(0)\mathbf{x}_0 + \mathbf{G}(k-1)\mathbf{u}(k-1) \\ &\quad + \mathbf{F}(k-1)\mathbf{G}(k-2)\mathbf{u}(k-2) + \cdots + \mathbf{F}(k-1)\cdots\mathbf{F}(0)\mathbf{G}(0)\mathbf{u}(0) \\ &= \prod_{j=0}^{k-1} \mathbf{F}(j)\mathbf{x}_0 + \sum_{i=0}^{k-1} \left[\prod_{j=i+1}^{k-1} \mathbf{F}(j) \right] \mathbf{G}(i)\mathbf{u}(i) \end{aligned} \quad (5-6-8)$$

若已知 $\mathbf{F}(i)$, $\mathbf{G}(i)$, 则可以通过上面的递推算法直接求出离散状态方程的解。从数值求解的角度看,还可以用迭代方法求解本方程,即从已知的 $\mathbf{x}(0)$ 根据方程式 (5-6-7) 推出 $\mathbf{x}(1)$, 再由 $\mathbf{x}(1)$ 计算 $\mathbf{x}(2)$, \cdots , 这样就可以递推地得出系统在各个时刻的状态。可见,迭代法更适合计算机实现。

例 5-46 试求解离散线性时变差分方程^[6]

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & \cos(k\pi) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \sin(k\pi/2) \\ 1 \end{bmatrix} u(k)$$

其中 $\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, 且 $u(k) = \begin{cases} 1, & k = 0, 2, 4, \cdots \\ -1, & k = 1, 3, 5, \cdots \end{cases}$ 。

解 采用迭代方法,可以用下面的循环结构立即得出状态变量在各个时刻的值,如图 5-9 所示。

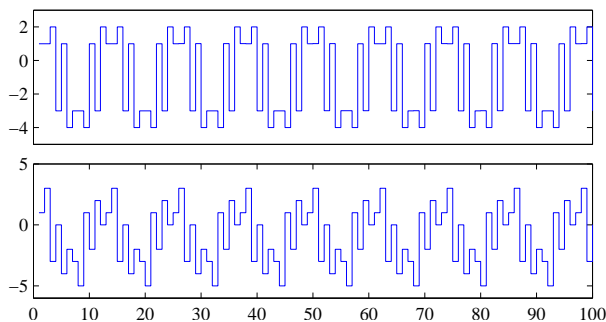


图 5-9 离散时变系统的响应曲线

```
>> x0=[1; 1]; x=x0;
    for k=0:100, if rem(k,2)==0, u=1; else, u=-1; end
```

```
F=[0 1; 1 cos(k*pi)]; G=[sin(k*pi/2); 1]; x1=F*x0+G*u; x0=x1; x=[x x1];
end
subplot(211), stairs(x(1,:)), subplot(212), stairs(x(2,:))
```

5.6.3 线性时不变差分方程的解法

线性时不变差分方程有 $\mathbf{F}(k) = \cdots = \mathbf{F}(0) = \mathbf{F}$, $\mathbf{G}(k) = \cdots = \mathbf{G}(0) = \mathbf{G}$, 由式(5-6-8)可以立即得出

$$\mathbf{x}(k) = \mathbf{F}^k \mathbf{x}_0 + \sum_{i=0}^{k-1} \mathbf{F}^{k-i-1} \mathbf{G} \mathbf{u}(i) \quad (5-6-9)$$

由于计算机数学语言并不能直接求出 k 是变量形式时 \mathbf{F}^k 的解析表达式, 所以用上述表达式无法求出状态变量的解析解, 必须考虑其他方法。

再重新考虑式(5-6-7), 其时不变形式可以写成

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k) \\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \end{cases}, \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (5-6-10)$$

两端同时求 z 变换, 由式(5-4-2)的性质可以得出

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{F})^{-1}[z\mathbf{x}_0 + \mathbf{G}\mathbf{U}(z) - \mathbf{G}z\mathbf{u}_0] \quad (5-6-11)$$

这样可以推导出离散状态方程的解析解为

$$\mathbf{x}(k) = \mathcal{Z}^{-1}[(z\mathbf{I} - \mathbf{F})^{-1}z] \mathbf{x}_0 + \mathcal{Z}^{-1}\{(z\mathbf{I} - \mathbf{F})^{-1}[\mathbf{G}\mathbf{U}(z) - \mathbf{G}z\mathbf{u}_0]\} \quad (5-6-12)$$

例 5-47 已知某离散系统的状态方程如下, 试求出各个状态阶跃响应的解析解

$$\mathbf{x}(k+1) = \begin{bmatrix} 11/6 & -5/4 & 3/4 & -1/3 \\ 1 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/4 & 0 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(k), \quad \mathbf{x}_0 = 0$$

解 直接套用下面的公式, 则可以求解出状态方程的解析解为

```
>> F=sym([11/6 -5/4 3/4 -1/3; 1 0 0 0; 0 1/2 0 0; 0 0 1/4 0]);
G=sym([4; 0; 0; 0]); syms z k; U=ztrans(sym(1));
x=iztrans(inv(z*eye(4)-F)*G*U,z,k)
```

从而得出各个状态的解析解为

$$\mathbf{x}(k) = \begin{bmatrix} -12(8+k+k^2)(1/2)^k + 48(1/3)^k + 48 \\ 24(-8+k+2k^2)(1/2)^k + 144(1/3)^k + 48 \\ 24(-10+3k-k^2)(1/2)^k + 216(1/3)^k \\ 12(-14+5k-k^2)(1/2)^k + 162(1/3)^k + 6 \end{bmatrix}$$

5.6.4 一般非线性差分方程的数值求解方法

假设已知差分方程的显式形式,即

$$y(t) = f(t, y(t-1), \dots, y(t-n), u(t), \dots, u(t-m)) \quad (5-6-13)$$

则可以通过递推的方法直接求解该方程,得出方程的数值解。

例 5-48 假设非线性差分方程可以表示为

$$y(t) = \frac{y(t-1)^2 + 1.1y(t-2)}{1 + y(t-1)^2 + 0.2y(t-2) + 0.4y(t-3)} + 0.1u(t)$$

并假设输入信号为正弦函数 $u(t) = \sin t$, 采样周期为 $T = 0.05$ s, 试求解该方程的数值解。

解 引入一个存储向量 y_0 , 其三个分量 $y_{0,1}$, $y_{0,2}$ 和 $y_{0,3}$ 分别表示 $y(t-3)$, $y(t-2)$ 和 $y(t-1)$, 在每一步递推后更新一次 y_0 向量。这样, 用下面的循环结构就可以求解该方程, 并绘制出输入信号和输出信号的曲线, 如图 5-10 所示。可见, 在正弦信号激励下, 非线性系统的输出会产生畸变, 这与线性系统响应是不同的。

```
>> y0=zeros(1,3); h=0.05; t=0: h: 8*pi; u=sin(t);
for i=1:length(t)
    y(i)=(y0(3)^2+1.1*y0(2))/(1+y0(3)^2+0.2*y0(2)+0.4*y0(1))+...
        0.1*u(i); y0=[y0(2:3), y(i)];
end
plot(t,y,t,u)
```

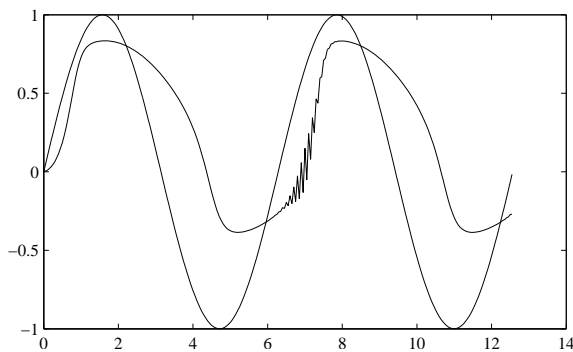


图 5-10 非线性离散差分方程数值解曲线

5.7 习 题

1. 对下列的函数 $f(t)$ 进行 Laplace 变换:

$$(1) f_a(t) = \frac{\sin \alpha t}{t}, \quad (2) f_b(t) = t^5 \sin \alpha t, \quad (3) f_c(t) = t^8 \cos \alpha t, \quad (4) f_d(t) = t^6 e^{\alpha t},$$

$$(5) f_e(t) = 5e^{-at} + t^4 e^{-at} + 8e^{-2t}, \quad (6) f_f(t) = e^{\beta t} \sin(\alpha t + \theta), \quad (7) f_g(t) = e^{-12t} + 6e^{9t}.$$

2. 对下面的 $F(s)$ 式进行 Laplace 反变换:

$$\begin{aligned} (1) F_a(s) &= \frac{1}{\sqrt{s}(s^2 - a^2)(s + b)}, & (2) F_b(s) &= \sqrt{s - a} - \sqrt{s - b}, & (3) F_c(s) &= \ln \frac{s - a}{s - b}, \\ (4) F_d(s) &= \frac{1}{\sqrt{s}(s + a)}, & (5) F_e(s) &= \frac{3a^2}{s^3 + a^3}, & (6) F_f(s) &= \frac{(s - 1)^8}{s^7}, \\ (7) F_g(s) &= \ln \frac{s^2 + a^2}{s^2 + b^2}, & (8) F_h(s) &= \frac{s^2 + 3s + 8}{\prod_{i=1}^8 (s + i)}, & (9) F_i(s) &= \frac{1}{2} \frac{s + \alpha}{s - \alpha}. \end{aligned}$$

3. Laplace 变换的一个重要应用是求解常系数线性微分方程, 可以利用当函数和各阶导数的零初始值下 $\mathcal{L}[d^n f(t)/dt^n] = s^n \mathcal{L}[f(t)]$ 这一性质对微分方程进行 Laplace 变换的方法去求解微分方程, 非零初值问题也可以利用相应方法求解。试使用这样的方法求解下面的微分方程。

$$(1) y''(t) + 3y'(t) + 2y(t) = e^{-t}, \quad y(0) = y'(0) = 0;$$

$$(2) y'' - y = 4 \sin t + 5 \cos 2t, \quad y(0) = -1, \quad y'(0) = -2;$$

$$(3) \begin{cases} x'' - x + y + z = 0, \\ x + y'' - y + z = 0, \\ x + y + z'' - z = 0, \end{cases} \quad x(0) = 1, \quad x'(0) = y(0) = y'(0) = z(0) = z'(0) = 0.$$

4. 假设某分数阶系统是由两个子模型 $G_1(s)$ 和 $G_2(s)$ 并联而成, 则系统的总模型可以由 $G(s) = G_1(s) + G_2(s)$ 计算出来。试对下面的两个子模型并联的总系统绘制出阶跃响应曲线。

$$G_1(s) = \frac{(s^{0.4} + 2)^{0.8}}{\sqrt{s}(s^2 + 3s^{0.9} + 4)^{0.3}}, \quad G_2(s) = \frac{s^{0.4} + 0.6s + 3}{(s^{0.5} + 3s^{0.4} + 5)^{0.7}}$$

5. 系统模型 $G_1(s)$ 、 $G_2(s)$ 串联连接构造的总系统可以由 $G(s) = G_2(s)G_1(s)$ 表示, 试求出上例两个子传递函数串联后的阶跃响应曲线。

6. 试求出下面函数的 Fourier 变换, 对得出的结果再进行 Fourier 反变换, 观察是否能得出原函数。

$$(1) f(x) = x^2(3\pi - 2|x|), \quad 0 \leq x \leq 2\pi, \quad (2) f(t) = t^2(t - 2\pi)^2, \quad 0 \leq t \leq 2\pi,$$

$$(3) f(t) = e^{-t^2}, \quad -l \leq t \leq l, \quad (4) f(t) = te^{-|t|}, \quad -\pi \leq t \leq \pi.$$

7. 试求出下面函数的 Fourier 正弦和余弦变换, 并用 Fourier 正弦、余弦反变换对得出的结果进行处理, 观察是否能还原原函数。

$$(1) f(t) = e^{-t} \ln t, \quad (2) f(x) = \frac{\cos x^2}{x}, \quad (3) f(x) = \ln \frac{1}{\sqrt{1 + x^2}}.$$

8. 试求下面函数的离散 Fourier 正弦、余弦变换。

$$(1) f(x) = e^{kx}, \quad (2) f(x) = x^3.$$

9. 试对分段函数 $f(x) = \begin{cases} \sin(a \ln x), & x \leq 1 \\ 0, & \text{其他 } x \end{cases}$ 进行 Mellin 变换。

10. 请将下述时域序列函数 $f(kT)$ 进行 z 变换, 并对结果进行反变换检验。

$$(1) f_a(kT) = \cos(kaT), \quad (2) f_b(kT) = (kT)^2 e^{-akT}, \quad (3) f_c(kT) = \frac{1}{a}(akT - 1 + e^{-akT}),$$

$$(4) f_d(kT) = e^{-akT} - e^{-bkT}, \quad (5) f_e(kT) = \sin(akT), \quad (6) f_f(kT) = 1 - e^{-akT}(1 + akT).$$

11. 已知下述各个 z 变换表达式 $F(z)$, 试对它们分别进行 z 反变换。

$$(1) F_a(z) = \frac{10z}{(z-1)(z-2)}, \quad (2) F_b(z) = \frac{z^2}{(z-0.8)(z-0.1)}, \quad (3) F_c(z) = \frac{z}{(z-a)(z-1)^2},$$

$$(4) F_d(z) = \frac{z^{-1}(1-e^{-aT})}{(1-z^{-1})(1-z^{-1}e^{-aT})}, \quad (5) F_e(z) = \frac{Az[z\cos\beta - \cos(\alpha T - \beta)]}{z^2 - 2z\cos(\alpha T) + 1}.$$

12. 对下面的 Laplace 变换式求出相应的 z 变换, 并对结果进行检验。

$$(1) G(s) = \frac{b}{s^2(s+a)}, \quad (2) G(s) = \frac{b}{s^2(s+a)^2} \frac{1-e^{-2s}}{s}.$$

13. 已知函数 $G(s) = \frac{1}{(s+1)^3}$, 若用 $s = \frac{2(z-1)}{T(z+1)}$ 替换 $G(s)$, 则可以得出函数 $H(z)$ 。这样的变换又称为双线性变换。若 $T = 1/2$, 试求出 $H(z)$ 。若对结果进行新变换 $z = \frac{1+Ts/2}{1-Ts/2}$, 则得出双线性反变换的结果。试观察这样的反变换是否能恢复原函数。

14. 试用计算机证明

$$\mathcal{Z} \left\{ 1 - e^{-akT} \left[\cos(bkT) + \frac{a}{b} \sin(bkT) \right] \right\} = \frac{z(Az+B)}{(z-1)(z^2 - 2e^{-aT} \cos(bT)z + e^{-2aT})}$$

$$\text{式中, } A = 1 - e^{-aT} \cos(bT) - \frac{a}{b} e^{-aT} \sin(bT), B = e^{-2aT} + \frac{a}{b} e^{-aT} \sin(bT) - e^{-aT} \cos(bT).$$

15. 试判定下面的多项式组是否互质。如果非互质, 试化简 $B(s)/A(s)$ 。

$$(1) B(x) = -3x^4 + x^5 - 11x^3 + 51x^2 - 62x + 24,$$

$$A(x) = x^7 - 12x^6 + 26x^5 + 140x^4 - 471x^3 - 248x^2 + 1284x - 720;$$

$$(2) B(x) = 3x^6 - 36x^5 + 120x^4 + 90x^3 - 1203x^2 + 2106x - 1080,$$

$$A(x) = x^9 + 15x^8 + 79x^7 + 127x^6 - 359x^5 - 1955x^4 - 3699x^3 - 3587x^2 - 1782x - 360.$$

16. 试绘制如下复变函数的 Riemann 曲面。

$$(1) f(z) = z \cos z^2, \quad (2) f(z) = ze^{-z^2}(\cos z - \sin z).$$

17. 试求出下面有理函数的部分分式展开。

$$(1) f(x) = \frac{3x^4 - 21x^3 + 45x^2 - 39x + 12}{x^7 + 15x^6 + 96x^5 + 340x^4 + 720x^3 + 912x^2 + 640x + 192},$$

$$(2) f(s) = \frac{s+5}{s^8 + 21s^7 + 181s^6 + 839s^5 + 2330s^4 + 4108s^3 + 4620s^2 + 3100s + 1000},$$

$$(3) f(x) = \frac{3x^6 - 36x^5 + 120x^4 + 90x^3 - 1203x^2 + 2106x - 1080}{x^7 + 13x^6 + 52x^5 + 10x^4 - 431x^3 - 1103x^2 - 1062x - 360},$$

$$(4) f(x) = \frac{(x^2 + 4x + 3)e^{-5x}}{x^5 + 7x^4 - 2x^3 - 100x^2 - 232x - 160}.$$

18. 试求出下列函数奇点处的留数。

$$(1) f(z) = \frac{1 - \sin ze^{-2z}}{z^7 \sin(z - \pi/3)} (z^4 + 10z^3 + 35z^2 + 50z + 24),$$

$$(2) f(z) = \frac{(z-3)^4}{z^4 + 5z^3 + 9z^2 + 7z + 2} (\sin z - e^{-3z}),$$

$$(3) f(z) = \frac{(1 - \cos 2z)(1 - e^{-z^2})}{z^3 \sin z}.$$

19. 试求解下面给出的差分方程模型。

$$(1) 72y(t) + 102y(t-1) + 53y(t-2) + 12y(t-3) + y(t-4) = 12u(t) + 7u(t-1), u(t) \text{ 为阶跃信号, 且 } y(-3) = 1, y(-2) = -1, y(-1) = y(0) = 0;$$

$$(2) y(t) - 0.6y(t-1) + 0.12y(t-2) + 0.008y(t-3) = u(t), u(t) = e^{-0.1t}, \text{ 且 } y(t) \text{ 初值为 } 0.$$

20. 试求解下面的非线性差分方程。

$$y(t) = u(t) + y(t-2) + 3y^2(t-1) + \frac{y(t-2) + 4y(t-1) + 2u(t)}{1 + y^2(t-2) + y^2(t-1)}, \text{ 且 } t \leq 0 \text{ 时 } y(t) = 0, u(t) = e^{-0.2t}$$

21. 求下面的封闭环路积分。

$$(1) \oint_{\Gamma} \frac{z^{15}}{(z^2 + 1)^2(z^4 + 2)^3} dz, \text{ 其中, } \Gamma \text{ 为 } |z| = 3 \text{ 正向圆周};$$

$$(2) \oint_{\Gamma} \frac{z^3}{1+z} e^{1/z} dz, \text{ 其中, } \Gamma \text{ 为 } |z| = 2 \text{ 正向圆周}.$$

22. 已知某离散系统的状态方程模型如下, 且 $\mathbf{x}^T(0) = [1, -1]$, 试求该系统阶跃响应的解析解, 并比较数值解。

$$(1) \mathbf{x}(t+1) = \begin{bmatrix} 0 & 1 \\ -0.16 & -1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t),$$

$$(2) \mathbf{x}(t+1) = \begin{bmatrix} 11/6 & -1/4 & 25/24 & -2 \\ 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & -3/4 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 2 \\ 1/2 \\ -3/8 \\ 1/4 \end{bmatrix} u(t).$$

参考文献

- [1] Valsa J, Brančik L. Approximate formulae for numerical inversion of Laplace transforms. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 1998, 11(3):153-166
- [2] Valsa J. Numerical inversion of Laplace transforms in MATLAB, MATLAB Central File ID: #32824, 2011
- [3] 《数学手册》编写组. 数学手册. 北京:人民教育出版社, 1979
- [4] 宋叔尼, 孙涛, 张国伟. 复变函数与积分变换. 北京:科学出版社, 2006
- [5] 西安交通大学高等数学教研室编. 复变函数(第二版). 北京:人民教育出版社, 1981
- [6] 郑大钟. 线性系统理论(第2版). 北京:清华大学出版社, 2002

第6章 代数方程与最优化问题的计算机求解

方程求解问题是科学与工程研究中经常遇到的问题。线性代数方程可以用第4章中介绍的方法直接求解,非线性方程或一般多项式方程的求解将在本章6.1节中介绍,在该节中将先介绍一元或二元方程的图解法,然后将介绍基于MATLAB符号运算工具箱的多项式方程或一类可以化成多项式方程的代数方程的准解析解方法,再介绍一般非线性方程组的数值解法,还将介绍非线性矩阵方程的数值解方法,试图得出感兴趣区域内的全部数值解。

最优化技术是当前科学研究中一类重要的手段。所谓最优化就是找出使得目标函数值达到最小或最大的自变量值的方法,可以毫不夸张地说,学会了最优化问题的求解思想,可以将科研的水平提高一个档次,因为原来解决问题得到一个解就满足了,学会了最优化的思想后,很自然地将追求问题最好的解。最优化问题从其分类看有无约束最优化问题和有约束最优化问题。6.2节将详细介绍无约束最优化问题以及MATLAB求解方法,介绍图解法和一般的数值算法,引入全局最优解与局部最优解的概念,并介绍梯度信息在最优化问题求解中的应用。还将介绍决策变量区间受限条件下的无约束最优化问题的求解方法。6.3节将介绍有约束最优化的概念,引入约束可行区域的概念,并就线性规划问题、二次型规划问题和一般非线性规划问题的MATLAB语言求解进行详细的介绍。6.4节将进一步引申最优化问题,引入整数规划和混合整数规划的概念,介绍如何用MATLAB语言求解小规模问题的穷举方法,并介绍基于“分枝定界法”的一般混合整数规划问题与0-1规划问题的MATLAB实现。6.5节将引入一类特殊的线性规划问题——线性矩阵不等式的概念、分类与求解问题。6.6节介绍多目标规划和极大极小问题的求解方法,6.7节将以最优路径规划为例介绍动态规划问题的计算机求解方法。通过本章内容的学习,读者应该能掌握一般非线性方程及非线性最优化问题的实际求解方法。

6.1 代数方程的求解

6.1.1 代数方程的图解法

MATLAB提供了很强的一元、二元隐函数绘制功能,充分利用这些功能就可以将一元、二元的方程用曲线表示,并由曲线的交点读出方程的实数根来。然而,方程的图解法是有局限性的,仅适用于一元、二元方程,多元方程是不能用图解法直接求解的。本节将通过例子演示一元、二元方程的求根问题。

1. 一元方程的图解法

第2章介绍过,用`ezplot()`函数可以绘制出给定的隐函数 $f(x) = 0$ 曲线,所以可以用图解法从给出的曲线和 $y = 0$ 线的交点上读出所有的实数解。

例6-1 用图解法求解方程 $e^{-3t} \sin(4t + 2) + 4e^{-0.5t} \cos(2t) = 0.5$ 。

解 用 `ezplot()` 函数可以绘制出如图 6-1(a) 所示的曲线, 该曲线与横轴的所有交点均是给出的一元方程的解。

```
>> ezplot('exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5',[0 5])
      line([0,5],[0,0])    % 同时绘制横轴
```

从得出的曲线可以看出, 该方程可能有多个实根, 如果想用图解法得出某个根, 可以对该点附近局部放大, 直到曲线穿越横轴, 且横轴给出的各个标点的数值完全一致时, 可以断定原方程的解即为 t 轴标度, 如图 6-1(b) 所示, 即该方程的一个解为 $t = 0.6738$ 。将其代入方程

```
>> t=0.6738; vpa(exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5)
```

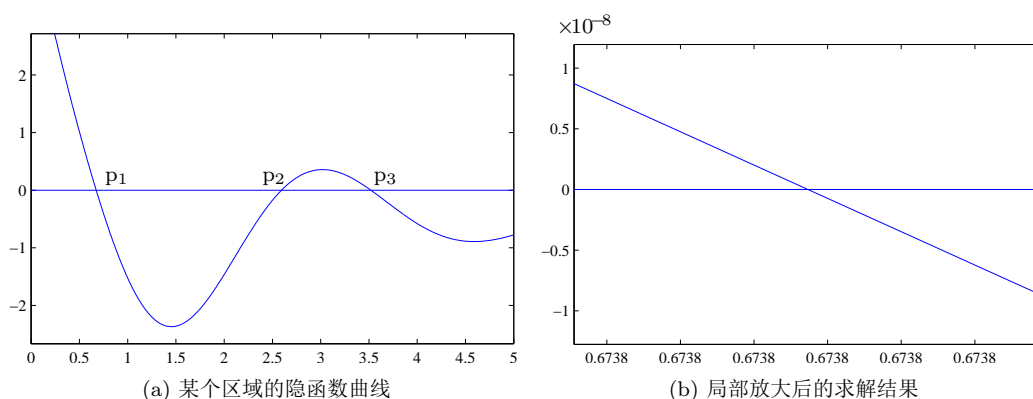


图 6-1 一元方程的图解法

可见, 得出的 p_1 点的函数值为 -0.2985×10^{-3} , 故可见得出的根是原方程的根, 但精度不是很高。用类似的方法还可以得出并验证其他的解。

2. 二元方程的图解法

二元方程也是可以通过图解法求解的, 可以通过 `ezplot()` 函数将第一个方程对应的曲线绘制出来, 再使用 `hold on` 命令保持图形不被刷新, 最后调用 `ezplot()` 函数将第二条曲线在同一坐标系下叠印出来。得出曲线后就可以通过读取交点坐标的方式得出联立方程的根。即使某个联立方程有多个根, 图解法也只能单独求取某一个感兴趣的根。

例 6-2 用图解法求解联立方程
$$\begin{cases} x^2 e^{-xy^2/2} + e^{-x/2} \sin(xy) = 0 \\ y^2 \cos(x+y^2) + x^2 e^{x+y} = 0 \end{cases}.$$

解 利用隐函数图形绘制的方法, 可以用图解法直接求解二元方程组, 可以通过下面的语句绘制出第一个方程的曲线, 如图 6-2(a) 所示。

```
>> ezplot('x^2*exp(-x*y^2/2)+exp(-x/2)*sin(x*y)') % 第一个方程曲线
```

该曲线上所有的点均满足第一个方程。可以用 `hold on` 语句保护当前的坐标系, 再用 `ezplot()` 函数绘制第二个方程的曲线, 这样在同一坐标系下绘制出两组曲线, 如图 6-2(b) 所示。

```
>> hold on; ezplot('y^2*cos(y+x^2)+x^2*exp(x+y)')
```

这两个方程对应曲线的交点都是联立方程的解, 可以通过图解法来求取二元联立方程的实根。

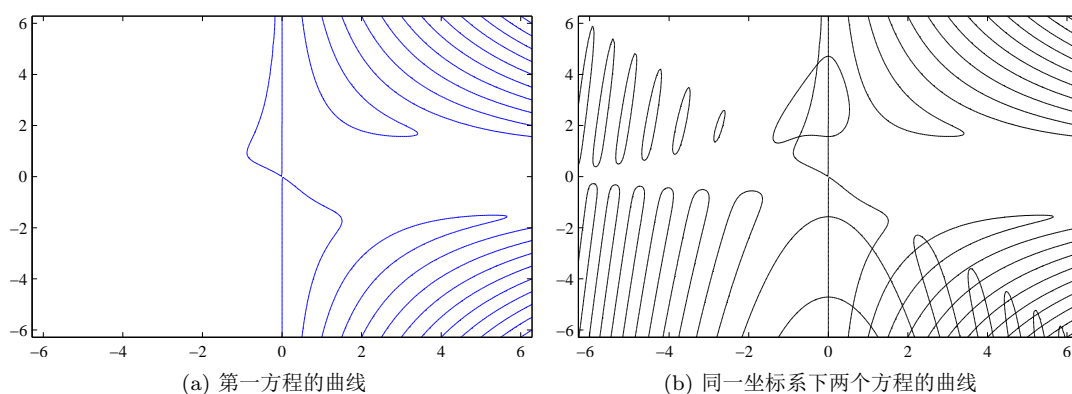


图 6-2 联立方程图解法示意图

6.1.2 多项式型方程的准解析解法

在介绍多项式方程的一般解法之前,先考虑下面给出的一个例子。

例 6-3 试用图解方法求解二元方程

$$\begin{cases} x^2 + y^2 - 1 = 0 \\ 0.75x^3 - y + 0.9 = 0 \end{cases}$$

解 用图解方法可以由下面的语句直接绘制出两条曲线,如图 6-3 所示,其交点就是原方程的解。

```
>> ezplot('x^2+y^2-1'); hold on; ezplot('0.75*x^3-y+0.9')
```

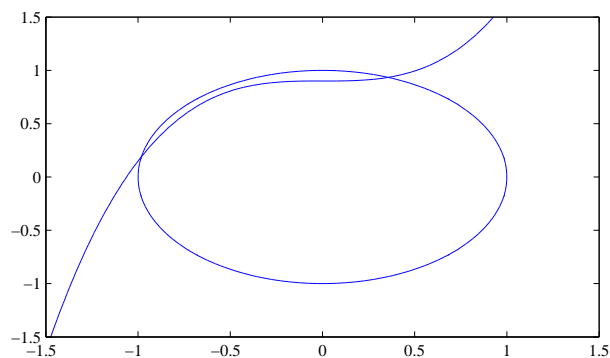


图 6-3 联立方程图解法示意图

从图 6-3 可见,这两条曲线共有两个交点,故可能轻易地得出结论,原联立方程有两个根。事实上,这样的结论是错误的,由第 2 个方程可以显式地将 y 写成 x^3 的形式,代入第 1 方程则得出关于 x 的 6 次多项式方程,该方程应该有 6 对根。因为用二维图形只能求解出方程的实根,而不能求解出方程的复数根,所以利用图解法有时也能得出错误的结论。

一般多项式方程的根可以为实数,也可以为复数。MATLAB 符号运算工具箱中给出的 `solve()` 函数对多项式类方程是十分有效的,可以用该函数求解出多项式方程所有的根。该函数的定义格式为

```
S = solve(eqn1,eqn2,...,eqn_n) % 最简调用方式
```

```
[x,y,...] = solve(eqn1,eqn2,...,eqn_n)           % 直接得出根
[x,y,...] = solve(eqn1,eqn2,...,eqn_n,'x,y,...') % 同上,并指定变量
```

其中, eqn_i 为第 i 个方程的符号表达式或字符串。该函数可以同时求解若干个联立方程,还可以按照第 3 种调用格式显式地指出需要求解方程的变量名,第 2、3 种调用格式将求出方程的根,并按各个变量名返回到 MATLAB 的工作空间,第 1 种调用格式将返回一个结构体变量 S ,其各个成员变量,如 $S.x$ 、 $S.y$ 等表示方程的根。

例 6-4 试用 `solve()` 函数求取例 6-3 中给出的联立方程。

解 考虑上述方法,利用 `solve()` 函数在 MATLAB 中可以给出如下的命令

```
>> [x,y]=solve('x^2+y^2-1=0','75*x^3/100-y+9/10=0')
```

可以得出方程的高精度数值解为

$$x = \begin{bmatrix} .35696997189122287798839037801365 \\ .86631809883611811016789809418650 + j1.2153712664671427801318378544391 \\ -.55395176056834560077984413882735 + j.35471976465080793456863789934944 \\ -.98170264842676789676449828873194 \\ -.55395176056834560077984413882735 - j.35471976465080793456863789934944 \\ .86631809883611811016789809418650 - j1.2153712664671427801318378544391 \end{bmatrix}$$

$$y = \begin{bmatrix} .93411585960628007548796029415446 \\ -1.4916064075658223174787216959259 + j.70588200721402267753918827138837 \\ .92933830226674362852985276677202 + j.21143822185895923615623381762210 \\ .19042035099187730240977756415289 \\ .92933830226674362852985276677202 - j.21143822185895923615623381762210 \\ -1.4916064075658223174787216959259 - j.70588200721402267753918827138837 \end{bmatrix}$$

显然,这样得出的方程阶次太高,不存在解析解。然而,利用 MATLAB 的符号运算工具箱可以得出原始问题的高精度数值解,故这里称之为准解析解。可以看出,除了前面得出的两组实数根外,还得出另外 4 组复数根,这是用普通数值解法所得不出来的。下面验证一下这样得出的根是不是原方程的根。直接给出下面的语句则可以发现,得出的根基本满足原方程。

```
>> [eval('x.^2+y.^2-1') eval('75*x.^3/100-y+9/10')]'
```

误差转置矩阵为 $\begin{bmatrix} -0.1 \times 10^{-31} & 0.5 \times 10^{-30} + j0.1 \times 10^{-30} & 0 & 0 & 0 & 0.5 \times 10^{-30} - j0.1 \times 10^{-30} \\ 0 & 0 & 0 & 0 & -j0 & -j0 \end{bmatrix}$ 。

例 6-5 多元多项式方程也可以用 `solve()` 函数直接求解。试求解下面的联立方程

$$\begin{cases} x + 3y^3 + 2z^2 = 1/2 \\ x^2 + 3y + z^3 = 2 \\ x^3 + 2z + 2y^2 = 2/4 \end{cases}$$

解 给出的联立方程是关于 x, y, z 的三元联立方程,可见它只含有多项式项,所以从理论上说可以将之转换成一元多项式方程,故方程可以由下面的 MATLAB 语句求出高精度数值解

```
>> [x,y,z]=solve('x+3*y^3+2*z^2=1/2','x^2+3*y+z^3=2','x^3+2*z+2*y^2=2/4')
```

事实上,该方程最终被变换成 27 次的多项式方程,所以得出的解向量 x, y, z 均为有 27 个分量的向量。由于篇幅所限,在这里不列出全部的解,只列出其中一个根。

$$\begin{cases} x_1 = -1.0869654762986136074917644096117 \\ y_1 = 0.03726466845064437552775081129721 \\ z_1 = 0.89073290972562790151300874796949 \end{cases}$$

可以用下面的语句验证解的误差为 6.9146×10^{-26} , 故而得出的结果是很精确的。

```
>> err=[x+3*y.^3+2*z.^2-1/2, x.^2+3*y+z.^3-2, x.^3+2*z+2*y.^2-2/4];
      norm(eval(err))
```

其实, 如果方程中存在多项式乘积的形式也可以直接求解。例如, 联立方程最后一个式子如果改写成 $x^3 + 2zy^2 = 2/4$, 其中含有非线性项 zy^2 , 该方程仍能通过 `solve()` 函数直接求解, 这时只需将求解语句变为

```
>> [x,y,z]=solve('x+3*y.^3+2*z.^2=1/2','x.^2+3*y+z.^3=2','x.^3+2*z*y.^2=2/4');
      err=[x+3*y.^3+2*z.^2-1/2, x.^2+3*y+z.^3-2, x.^3+2*z.*y.^2-2/4];
      norm(eval(err)) % 将解代入求误差
```

这时误差达到 4.3077×10^{-26} , 说明得出的解仍然是精确的。

例 6-6 试求解下面的方程, 其中含有自变量的倒数等形式

$$\begin{cases} \frac{1}{2}x^2 + x + \frac{3}{2} + 2\frac{1}{y} + \frac{5}{2y^2} + 3\frac{1}{x^3} = 0 \\ \frac{y}{2} + \frac{3}{2x} + \frac{1}{x^4} + 5y^4 = 0 \end{cases}$$

解 这样的方程指望求取解析解是基本上不可能的, 但用下面的语句可以直接得出原方程的精确数值解, 共有 26 对根

```
>> [x,y]=solve('x.^2/2+x+3/2+2/y+5/(2*y.^2)+3/x.^3=0',...
               'y/2+3/(2*x)+1/x.^4+5*y.^4','x,y'); size(x)
```

其中, 第 1 对根为

$$\begin{cases} x_1 = 0.93020851860976459141084889836970 + j0.44242013491916995256639798767139 \\ y_1 = -0.39388334385234983573088749775055 + j0.62771855170677843775952969855352 \end{cases}$$

将得出的全部方程根代入原始方程, 则能得出很小的计算误差, 达到 10^{-30} 级, 说明该方程的各个解都是非常精确的。

```
>> e=[x.^2/2+x+3/2+2./y+5./(2*y.^2)+3./x.^3,y/2+3./(2*x)+1./x.^4+5*y.^4];
      norm(eval(e))
```

例 6-7 试求解带有参数的方程 $\begin{cases} x^2 + ax^2 + 6b + 3y^2 = 0 \\ y = a + x + 3 \end{cases}$

解 MATLAB 符号运算工具箱中提供的 `solve()` 函数还可以直接实现带有变量的方程的解, 这样的求解用普通的数值解方法是不能实现的。求解上述方程只需给出下面语句即可

```
>> syms a b; [x,y]=solve('x.^2+a*x.^2+6*b+3*y.^2=0','y=a+(x+3)','x,y')
```

该方程的解析解为

$$x = \frac{-6a - 18 \pm 2\sqrt{-21a^2 - 45a - 27 - 24b - 6ab - 3a^3}}{2(4+a)}, \quad y = a + x + 3$$

其实, 该方法同样适用于更高阶方程的解, 但得出的解是很冗长的, 不适合显示出来。

然而,解析求解的方法并不是万能的,因为这里的例子最终可以转换为一元多项式方程,所以能用它求解,但更一般的方程是不能解出来的。

6.1.3 一般非线性方程数值解

MATLAB 语言环境提供了 `fsolve()` 函数,能够求出已知多元方程的一个实数根。该函数的调用格式为

```
x = fsolve(Fun,x0) % 最简求解语句
[x,f,flag,out] = fsolve(Fun,x0,opt,p1,p2,...) % 一般求解格式
```

其中, `Fun` 为所需求解方程的 M-函数或匿名函数描述, `x0` 为搜索点的初值,方程求根程序将从该值开始以逐步减小误差的算法搜索出满足方程的实根 `x`。若返回的 `flag` 大于 0,则表示求解成功,否则求解出现问题,应该参考给出的警告信息。

对于更复杂的问题,用户可以定义方程求解控制参数模板 `opt` 来控制求解方法或其他要求,更好地得出方程的根。该变量是一个结构体数据,其常用的成员变量在表 6-1 中给出,用户可以用下面的语句修改控制变量

```
opt = optimset; % 获得默认的常用变量
opt.TolX = 1e-10; 或 set(opt,'TolX',1e-10) % 用这两种方法修改参数
```

其中的某些值,如 `MaxFunEvals` 和问题类型有关,如方程求解和有约束最优化问题一般选择其值为 100 倍的自变量个数,而无约束最优化问题一般支持 200 倍的决策变量个数。在该求解函数中,方程求解是采用迭代方式进行的,如果两步间的搜索步距小于成员变量 `TolX`,或方程的误差小于成员变量 `FunTol`,迭代搜索的过程也将停止下来。

表 6-1 方程求解与最优化的控制参数表

参数名	参 数 说 明
Display	中间结果显示方式,其值可以取 <code>off</code> 表示不显示中间值, <code>iter</code> 表示逐步显示, <code>notify</code> 表示在求解不收敛时给出提示, <code>final</code> 只显示最终值
GradObj	求解最优化问题时使用,表示目标函数的梯度是否已知,可以选择为 <code>'off'</code> 或 <code>'on'</code>
LargeScale	表示是否使用大规模问题算法,取值为 <code>on</code> 或 <code>off</code> ,一般几个变量的问题不必采用该算法
MaxIter	方程求解和优化过程最大允许的迭代次数,若方程未求出解,可以适当增加该值
MaxFunEvals	方程函数或目标函数的最大调用次数
TolFun	误差函数误差限控制量,当函数的绝对值小于此值即终止求解
TolX	解的误差限控制量,当解的绝对值小于此值即终止求解

例 6-8 试用数值方法求解例 6-3 中给出的二元方程。

解 令 $p_1 = x, p_2 = y$, 可以编写出一个描述此二元方程的函数如下

```
>> f=@(p)[p(1)*p(1)+p(2)*p(2)-1; 0.75*p(1)^3-p(2)+0.9];
```

这样,就可以在给定的初值 $x_0 = 1, y_0 = 2$ 下调用 `fsolve()` 函数,直接求出方程的根

```
>> OPT=optimset; OPT.LargeScale='off'; [x,Y,c,d]=fsolve(f,[1; 2],OPT)
```

可以得出方程的数值解为 $\mathbf{x} = [0.35696997, 0.93411586]^T$, 将其带入方程后得出的残差为 $\mathbf{Y} = [0.1215 \times 10^{-9}, 0.0964 \times 10^{-9}]$ 。在求解此二元方程时仅调用了方程函数 21 次就得出了方程的解。可见, 引入匿名函数无需为要求解的每个数学问题都编写一个单独的 MATLAB 模型文件, 这样使得问题的求解与文件管理变得更容易、方便。

若改变初始猜测值, 令 $\mathbf{x}_0 = [-1, 0]^T$, 则

```
>> [x,Y,c,d]=fsolve(f,[-1,0]',0PT); x, norm(Y), kk=d.funcCount
```

这时搜索出的解为 $\mathbf{x} = [-0.981703, 0.1904204]^T$, 方程的残差为 0.5618×10^{-10} , f 函数的调用次数为 15 次。可见, 初值改变之后, 还能得出另外一组解。所以初值的选择有时对整个问题的求解有很大的影响, 在某些初值下甚至无法搜索到方程的解。

例 6-9 重新考虑例 6-1 中给出的方程 $e^{-3t} \sin(4t+2) + 4e^{-0.5t} \cos 2t = 0.5$, 试用数值方法求其更精确的数值解。

解 显然该非线性方程没有解析解。例 6-1 中用图解法得出了一个数值解为 $t = 0.6738$, 以此解为初值, 则可以用 `fsolve()` 函数得出更精确的数值解。

先使用符号运算工具箱中的 `solve()` 函数求解此问题

```
>> syms t x; solve(exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5)
```

得出的高精度数值解为 0.673745705001347567。然而, 这样的方法不允许手工选择初始搜索点, 所以只能获得这一个解。用图解法选择初值, 再用数值解法可以得出该初值附近的精确的数值解。

```
>> y=@(t)exp(-3*t).*sin(4*t+2)+4*exp(-0.5*t).*cos(2*t)-0.5;
```

```
[t,f]=fsolve(y,0.6738)
```

这时 $t = 0.673745703880081$, 代入原方程, $f = 6.1650 \times 10^{-9}$ 。该解的精度明显高于图解法的精度, 且可以任意选择初值。从算法本身看, 默认的求解精度偏低, 所以可以试着重新设置相关的控制变量, 就可以得出下面更精确的结果: $t = 0.673745705001348$, $f = 0$ 。

```
>> ff=optimset; ff.TolX=1e-16; ff.TolFun=1e-30; [t,f]=fsolve(y,0.6738,ff)
```

从求解效果看, 给出一个很精确的初值对整个求解速度和精度没有太大的帮助, 真正使得本例获得高精度解的还是 `TolFun` 属性。

6.1.4 求解多解方程的全部解

第 4 章 4.4.4 节曾讨论过一类特殊非线性矩阵方程 —— 代数 Riccati 方程的求解方法, 然而, 这样的方法依赖于巧妙的 Schur 分解才能求解。这样的方法局限性很大, 方程类型稍有变化则无能为力。例如, 考虑下面的扩展方程

$$\mathbf{AX} + \mathbf{XD} - \mathbf{XBX} + \mathbf{C} = 0 \quad (6-1-1)$$

或考虑一个更不易求解的新形式, 这里称作类 Riccati 方程

$$\mathbf{AX} + \mathbf{XD} - \mathbf{XBX}^T + \mathbf{C} = 0 \quad (6-1-2)$$

则前面介绍的 `are()` 函数不能再直接使用。这里我们将探索基于非线性方程数值解的一般矩阵方程求解方法。

假设某矩阵方程 $F(X) = 0$, 其中 X 为 $n \times m$ 阶矩阵, 且 $F(\cdot)$ 也为 $n \times m$ 阶矩阵, 则仍然可以用匿名函数或 M-函数直接描述方程, 然后就可以用一般非线性方程求解函数 `fsolve()` 直接求解了。

为了更好地求解矩阵方程或多解方程, 我们编写了函数 `more_sols()`, 该函数在感兴趣的范围内随机选择初值, 并搜索方程的解。如果找到新的解则将该解存储起来, 再重新选择随机初值搜索新的解。函数的整体结构采用了 `while` 循环结构, 用户可以随时按下 `Ctrl-C` 键中断程序的运行, 也可以等待一段指定的时间 (如 30s), 找不到新的解停止运行。

```
function more_sols(f,X0,varargin)
[A,tol,tlim]=default_vals(1000,eps,30,varargin{:}); [n,m,i]=size(X0);
if length(A)==2, a=A(1); b=A(2); else, a=-0.5*A; b=0.5*A; end
ff=optimset; ff.Display='off'; ff1=ff; ff.TolX=tol; ff.TolFun=tol; X=X0;
try, err=evalin('base','err'); catch, err=0; end, if i<=1; err=0; end, tic
while (1), x0=a+(b-a)*rand(n,m); [x,aa,key]=fsolve(f,x0,ff1);
    t=toc; if t>tlim, break; end
    if key>0, N=size(X,3);
        for j=1:N, if norm(X(:,j)-x)<1e-5; key=0; break; end, end
        if key>0, [x1,aa,key]=fsolve(f,x,ff);
            if norm(x-x1)<1e-5 & key>0; X(:,i+1)=x1; assignin('base','X',X);
                err=max([norm(aa),err]); assignin('base','err',err); i=i+1, tic
            end, end, end, end
```

该函数的调用格式为 `more_sols(f,X0,A,ε,tlim)`, 底层函数 `default_vals()` 在第3章给出, 可以用于读取几个默认的控制变量, 其中 A 是感兴趣区域的表示, 如果该值为标量则表示该感兴趣的求解区间为 $(-A/2, A/2)$, 默认值为 $A = 1000$, 表示可以大范围求解代数方程, A 为向量则表示解区间。 ϵ 的默认值为 `eps`, 大约为 10^{-16} 级别。 t_{lim} 的默认值为 30, 表示如果 30s 内没有找到新的解则程序停止。用户可以随时用 `Ctrl-C` 键中断程序运行。

和其他函数相比, 这个函数是很特殊的函数, 该函数使用了死循环结构, 除了 30s 没有发现新解正常停止程序外, 经常需要用 `Ctrl-C` 中断程序, 所以不适合安排返回变量。该函数采用了 `assignin()` 函数将得出的结果写入 MATLAB 的工作空间, 工作空间中的变量名 X 存储方程的解, $X(:, :, i)$ 存储方程的第 i 个解, 另一个 MATLAB 工作空间变量 `err` 存储所得出的最大误差矩阵的范数。

如果该函数停止或中断, 而用户想继续寻找新的解, 可以给出命令 `more_sols(f,X)`。

例 6-10 重新考虑求解例 4-54 中的 Riccati 方程, 其矩阵如下

$$A = \begin{bmatrix} -2 & 1 & -3 \\ -1 & 0 & -2 \\ 0 & -1 & -2 \end{bmatrix}, B = \begin{bmatrix} 2 & 2 & -2 \\ -1 & 5 & -2 \\ -1 & 1 & 2 \end{bmatrix}, C = \begin{bmatrix} 5 & -4 & 4 \\ 1 & 0 & 4 \\ 1 & -1 & 5 \end{bmatrix}$$

解 在这里研究的 Riccati 方程中, 未知变量 X 为矩阵。可以用下面的语句描述原方程, 再调用 `more_sols()` 函数得出方程所有的根

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
    C=[5 -4 4; 1 0 4; 1 -1 5]; f=@(X)A'*X+X*A-X*B*X+C;
```



```
more_sols(f,zeros(3,3,0)); X, err
```

上述语句得出的最大误差为 1.4904×10^{-12} , 得出的 8 组根为

$$\begin{aligned} \mathbf{X}_1 &= \begin{bmatrix} 0.9874 & -0.7983 & 0.4189 \\ 0.5774 & -0.1308 & 0.5775 \\ -0.2840 & -0.0730 & 0.6924 \end{bmatrix}, \mathbf{X}_2 = \begin{bmatrix} 1.2213 & -0.4165 & 1.9775 \\ 0.3578 & -0.4894 & -0.8863 \\ -0.7414 & -0.8197 & -2.3560 \end{bmatrix} \\ \mathbf{X}_3 &= \begin{bmatrix} 0.6665 & -1.3223 & -1.7200 \\ 0.3120 & -0.5640 & -1.1910 \\ -1.2273 & -1.6129 & -5.5939 \end{bmatrix}, \mathbf{X}_4 = \begin{bmatrix} -2.1032 & 1.2978 & -1.9697 \\ -0.2467 & -0.3563 & -1.4899 \\ -2.1494 & 0.7190 & -4.5465 \end{bmatrix} \\ \mathbf{X}_5 &= \begin{bmatrix} -0.1538 & 0.1087 & 0.4623 \\ 2.0277 & -1.7437 & 1.3475 \\ 1.9003 & -1.7513 & 0.5057 \end{bmatrix}, \mathbf{X}_6 = \begin{bmatrix} 0.8878 & -0.9609 & -0.2446 \\ 0.1072 & -0.8984 & -2.5563 \\ -0.0185 & 0.3604 & 2.4620 \end{bmatrix} \\ \mathbf{X}_7 &= \begin{bmatrix} 23.9467 & -20.6673 & 2.4529 \\ 30.1460 & -25.9830 & 3.6699 \\ 51.9666 & -44.9108 & 4.6410 \end{bmatrix}, \mathbf{X}_8 = \begin{bmatrix} -0.7619 & 1.3312 & -0.8400 \\ 1.3183 & -0.3173 & -0.1719 \\ 0.6371 & 0.7885 & -2.1996 \end{bmatrix} \end{aligned}$$

例 6-11 试求出并检验式 (6-1-2) 中描述的一类 Riccati 方程的全部根, 其中

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 9 \\ 9 & 7 & 9 \\ 6 & 5 & 3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 3 & 6 \\ 8 & 2 & 0 \\ 8 & 2 & 8 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 7 & 0 & 3 \\ 5 & 6 & 4 \\ 1 & 4 & 4 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 3 & 9 & 5 \\ 1 & 2 & 9 \\ 3 & 3 & 0 \end{bmatrix}$$

解 迄今为止这类方程没有其他的求解方法, 只能采用这里给出的搜索方法求解。采用默认的 $A = 1000$, 则该函数可以大范围搜索方程的根。该函数可以正常结束, 得出全部的 16 个根, 最大的误差为 1.1×10^{-10} 。

```
>> A=[2 1 9; 9 7 9; 6 5 3]; B=[0 3 6; 8 2 0; 8 2 8];
C=[7 0 3; 5 6 4; 1 4 4]; D=[3 9 5; 1 2 9; 3 3 0];
f=@(X)A*X+X*D-X*B*X.'+C; more_sols(f,zeros(3,3,0)); X, err
```

例 6-12 考虑例 6-2 给出的联立方程, 试求出 $-2\pi \leq x, y \leq 2\pi$ 范围内的全部数值解。

$$\begin{cases} x^2 e^{-xy^2/2} + e^{-x/2} \sin(xy) = 0 \\ y^2 \cos(x+y^2) + x^2 e^{x+y} = 0 \end{cases}$$

解 例 6-2 中用图解法得出了方程解的图示, 如果想比较精确地得出某个根, 可以用图解法得出该点附近的一个近似的根, 以其为初始位置再调用 `fsolve()` 将其根的精确值搜索出来。如果想同时得出感兴趣区域内所有的根, 则可以采用前面介绍的 `more_sols()` 函数直接求解。因为感兴趣的区域是 $(-2\pi, 2\pi)$, 所以可以选择 $A = 13$ 或 $A = [-2\pi, 2\pi]$ 。另外已知 $[0, 0]$ 点是方程的一个根, 所以可以以其为出发点直接搜索。给出下面的命令即可先定义方程的匿名函数, 然后求出方程在感兴趣区域内的所有根, 并得出解的最大误差为 2.7711×10^{-13} 。

```
>> f=@(x)[x(1)^2*exp(-x(1)*x(2)^2/2)+exp(-x(1)/2)*sin(x(1)*x(2));
x(2)^2*cos(x(2)+x(1)^2)+x(1)^2*exp(x(1)+x(2))];
more_sols(f,[0; 0],[-2*pi,2*pi]); err
```

得出在指定区域内方程所有的解则可以由下面的语句将它们显示出来, 如图 6-4 所示, 可见, 通过该函数的调用, 感兴趣内的所有实根确实均求出来了。

```
>> ezplot('x^2*exp(-x*y^2/2)+exp(-x/2)*sin(x*y)=0')
```

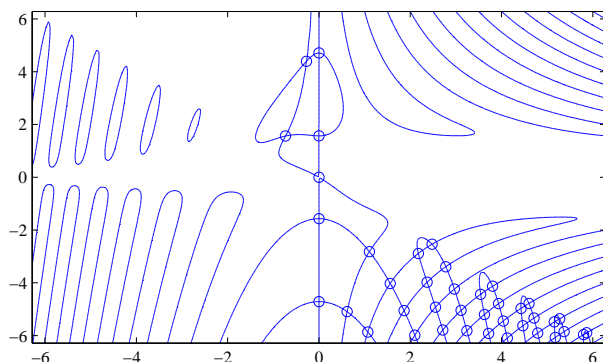


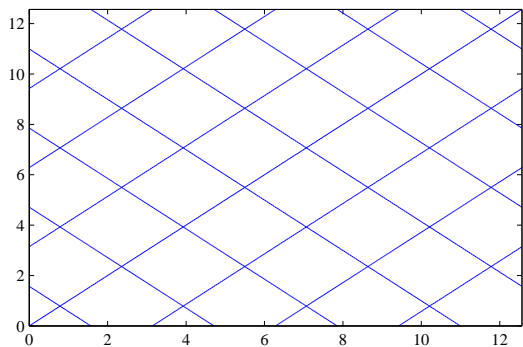
图 6-4 非线性方程的全部解

```
hold on; ezplot('y^2*cos(y+x^2)+x^2*exp(x+y)=0')
x=X(1,1,:); x=x(:); y=X(2,1,:); y=y(:); plot(x,y,'o')
```

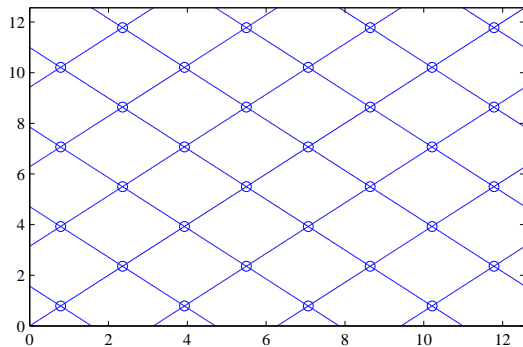
例 6-13 考虑联立方程 $\sin(x-y)=0, \cos(x+y)=0$ 。试求出 $0 \leq x, y \leq 4\pi$ 范围内的全部根。

解 可以先由图解法画出方程感兴趣区域的解,其分布图如图 6-5 (a)。

```
>> ezplot('sin(x-y)',[0,4*pi]), hold on, ezplot('cos(x+y)',[0,4*pi])
```



(a) 图解法



(b) 感兴趣区域内发现的全部解

图 6-5 非线性方程的全部解

将原方程用匿名函数表示出来,并选择 $A = [0, 4\pi]$,则可以通过下面的语句得出方程在感兴趣区域的所有的根,如图 6-5(b) 所示,用该函数确实能得出方程的全部实根。

```
>> f=@(x)[sin(x(1)-x(2)); cos(x(1)+x(2))]; more_sols(f,zeros(2,1,0),[0,4*pi]);
x=X(1,1,:); y=X(2,1,:); plot(x(:),y(:),'o')
```

6.2 无约束最优化问题求解

无约束最优化问题是最简单的一类最优化问题,其一般数学描述为

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (6-2-1)$$

其中, $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ 称为优化变量, $f(\cdot)$ 函数称为目标函数, 该数学表示的含义亦即求取一组 \mathbf{x} 向量, 使得最优化目标函数 $f(\mathbf{x})$ 为最小, 故这样的问题又称为最小化问题。其实, 最小化是最优化问题的通用描述, 它不失普遍性。如果要想求解最大化问题, 那么只需给目标函数 $f(\mathbf{x})$ 乘以一个负号就能立即将最大化问题转换成最小化问题。所以本书中描述的全部问题都是最小化问题。

6.2.1 解析解法和图解法

无约束最优化问题的最优解 \mathbf{x}^* 处, 目标函数 $f(\mathbf{x})$ 对 \mathbf{x} 各个分量的一阶导数为 0, 从而可以列出下面的方程

$$\left. \frac{\partial f}{\partial x_1} \right|_{\mathbf{x}=\mathbf{x}^*} = 0, \left. \frac{\partial f}{\partial x_2} \right|_{\mathbf{x}=\mathbf{x}^*} = 0, \dots, \left. \frac{\partial f}{\partial x_n} \right|_{\mathbf{x}=\mathbf{x}^*} = 0 \quad (6-2-2)$$

求解这些方程构成的联立方程可以得出极值点。其实, 解出的一阶导数均为 0 的极值点不一定是极小值的点, 其中有的还可能是极大值点。极小值问题还应该要有正的二阶导数。对于单变量的最优化问题, 可以考虑采用解析解的方法进行求解。然而多变量最优化问题因为需要将其转换成求解多元非线性方程, 其难度也不低于直接求取最优化问题, 所以没有必要采用解析解方法求解。

一元函数最优化问题的图解法也是很直观的, 应绘制出该函数的曲线, 在曲线上就能看出其最优值点。二元函数的最优化也可以通过图解法求出。但三元或多元函数, 由于用图形没有办法表示, 所以不适合用图解法求解。

例 6-14 对例 6-1 中给出的方程 $f(t) = e^{-3t} \sin(4t+2) + 4e^{-0.5t} \cos(2t) - 0.5$, 试用解析求解和图形求解的方法研究该函数的最优性。

解 可以先表示该函数, 并解析地求解该函数的一阶导数, 用 `ezplot()` 函数可以绘制出 $t \in [0, 4]$ 区间内一阶导函数的曲线, 如图 6-6(a) 所示。

```
>> syms t; y=exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5;
    y1=diff(y,t); % 求取一阶导函数
    ezplot(y1,[0,4]) % 绘制出选定区间内一阶导函数曲线
```

其实, 求解导函数等于 0 的方程不比直接求解其最优值简单。用图解法可以看出, 在这个区间内有两个点, A_1 和 A_2 , 使得它们的一阶导函数为 0, 但从其一阶导数走向看, A_2 点对应负的二阶导数值, 所以该点对应于极大值点, 而 A_1 点对应于正的二阶导数值, 故为极小值点。 A_1 点的值可以由下面的语句直接解出, $t_0 = 1.4528424981725411893375778048840$, 该点的二阶导数为 $b = 7.8553 > 0$, 说明该点对应于最小值。

```
>> t0=solve(y1), ezplot(y,[0,4]) % 求出一阶导数等于零的点
    y2=diff(y1); b=subs(y2,t,t0) % 并验证二阶导数为正
```

还可以用图解方法进一步验证得出的结果, 如图 6-6(b) 所示, 可见, A_1 为最小值, A_2 为最大值。

然而因为给定的函数是非线性函数, 所以用解析法或类似的方法求解最小值问题一点都不比直接求解最优化问题简单。因此, 除演示之外, 不建议用这样的方法求解该问题, 而直接采用最优化问题求解程序得出问题的解。

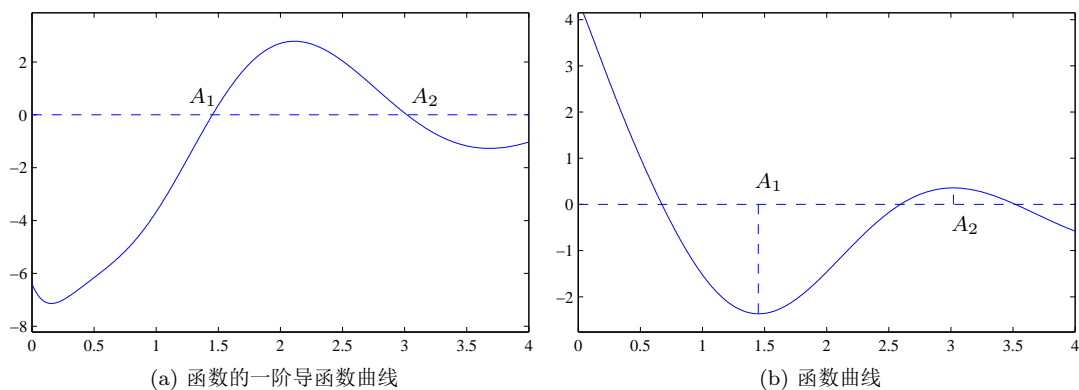


图 6-6 一元函数的导数和方程图解法

6.2.2 基于 MATLAB 的数值解法

MATLAB 语言中提供了求解无约束最优化的函数 `fminsearch()`, 其最优化工具箱中还提供了函数 `fminunc()`, 二者的调用格式完全一致, 为

```
x = fminunc(Fun,x0) % 最简求解语句
[x,f,flag,out] = fminunc(Fun,x0,opt,p1,p2,...) % 一般求解格式
```

其输入与返回参数的定义与 `fsolve()` 函数中的控制变量完全一致。该函数主要采用了文献 [1] 中提出的单纯形算法。下面将通过例子来演示无约束最优化问题的数值解法。

例 6-15 已知二元函数 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$, 试用 MATLAB 提供的求解函数求出其最小值, 并用图形方法表示其求解过程。

解 因为函数中给出的自变量是 x, y , 而最优化函数需要求取的是自变量向量 x , 故在求解前应该先进行变量替换, 如令 $x_1 = x, x_2 = y$, 这样就可以用下面的语句由匿名函数形式定义出目标函数 f , 用下面的语句求解出最优解为 $x = [0.6111, -0.3056]$

```
>> f=@(x)(x(1)^2-2*x(1))*exp(-x(1)^2-x(2)^2-x(1)*x(2)); x0=[2; 1];
[x,b,c,d]=fminsearch(f,x0)
```

同样的问题用 `fminunc()` 函数求解, 则可以得出同样的结果。

```
>> [x,b,c,d]=fminunc(f,[2; 1])
```

比较两种方法, 显然可以看出, 用 `fminunc()` 函数的效率明显高于 `fminsearch()`, 因为对目标函数调用的次数明显少于后者。所以在无约束最优化问题求解时, 如果安装了最优化工具箱则建议使用 `fminunc()` 函数。

在求解过程中, 如果手工修改 `fminunc()` 下级的 `fminsub()` 函数文件, 就可以追踪出各个搜索中间点的坐标。下面在图形上显示出搜索中间过程。假设选择 $x = [2, 1]^T$, 则由下面的语句可以得出所需的解, 同时还能由修改的函数得出中间点坐标为

x_i	2	0.2401	-0.1398	0.2168	0.3355	0.5514	0.6129	0.6111
y_i	1	1.0502	0.5752	1.0210	-0.5508	-0.1775	-0.3053	-0.3058

用下面的语句可以绘制出搜索过程中间点的轨线, 如图 6-7 所示

```
>> xx=[2 0.2401 -0.1398 0.2168 0.3355 0.5514 0.6129 0.6111
```

```

1  1.0502  0.5752  1.0210 -0.5508 -0.1775 -0.3053 -0.3058];
[x,y]=meshgrid(-3:.1:2, -2:.1:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
contour(x,y,z,30); line(xx(1,:),xx(2,:))
h=line(xx(1,:),xx(2,:)); set(h,'Marker','o')

```

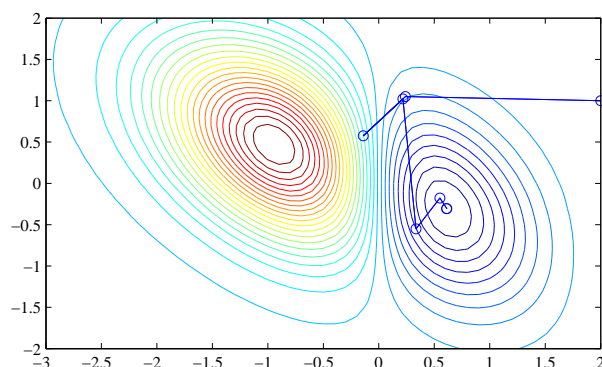


图 6-7 求解过程示意图

MATLAB 最优化工具箱还支持用结构体变量来描述最优化问题,这样可以使最优化问题的描述更规范。可以建立一个结构体变量 `problem`,用 `problem.objective` 成员变量描述目标函数,其成员变量 `x0` 描述初始搜索点,用 `problem.options = optimset, problem.solver = 'fminunc'`,则可以用 `[x,fm,flag] = fminunc(problem)` 直接求解无约束最优化问题。

例 6-16 试用结构体的方式重新描述并求解例 6-15 中的无约束最优化问题。

解 可以用下面命令建立起最优化问题的结构体变量 `problem`,然后调用 `fminunc()` 函数即可以直接求解原始问题,得出的结果与例 6-15 中的结果完全一致。

```

>> problem.solver='fminunc'; problem.options=optimset;
    problem.objective=@(x)(x(1)^2-2*x(1))*exp(-x(1)^2-x(2)^2-x(1)*x(2));
    problem.x0=[2; 1]; [x,b,c,d]=fminunc(problem)

```

6.2.3 全局最优解与局部最优解

以单变量 x 为例,无约束最优化问题函数有解的必要条件是 $df(x)/dx = 0$,但满足该条件的 x 值可能不唯一,可能存在多个解。从最优化搜索的角度来说,可能找到其中一个这样的点。下面将通过例子引入全局最优解和局部最优解的概念。

例 6-17 假设目标函数为 $y(t) = e^{-2t} \cos 10t + e^{-3t-6} \sin 2t$, $t \geq 0$,试观察不同的初值能得出的最小值,并讨论局部最小值与全局最小值的概念。

解 由给定的目标函数,可以立即写出可以用于无约束最优化搜索的 MATLAB 表示,可以采用下面的匿名函数或编写相应的 MATLAB 文件

```

>> f=@(t)exp(-2*t).*cos(10*t)+exp(-3*(t+2)).*sin(2*t); % 目标函数

```

若选定初始搜索点为 $t_0 = 1$, 则可以给出如下语句获得目标函数的最优值为 $f_1 = -0.1547$, 最优解为 $t_1 = 0.9228$ 。

```
>> t0=1; [t1,f1]=fminsearch(f,t0)
```

若选择另一个初始点, 则可以得出最优解为 $t_2 = 0.2945$, 目标函数为 $f_2 = -0.5436$ 。

```
>> t0=0.1; [t2,f2]=fminsearch(f,t0)
```

可见, 给出不同的初值, 此函数能得出不同的“最优解”, 但从最优解处的函数值看, t_1 处的函数值显然大于 t_2 处的函数值。故可以得出结论, t_1 得出的并非真正的全局最优解, 而是某种局部最优解。试凑其他的初值, 如 $t_0 = 1.5, 2, 2.5, \dots$ 还可能得出其他的局部最优值, 这里不一一列出。用 MATLAB 的 `ezplot()` 函数可以绘制给定目标函数 $y(t)$ 在 $t \in (0, 2.5)$ 定义域内的曲线, 如图 6-8(a) 所示, 区间内的全局和局部最优值均由虚线表示出来。

```
>> syms t; y=exp(-2*t)*cos(10*t)+exp(-3*(t+2))*sin(2*t);
ezplot(y,[0,2.5]); ylim([-0.6,1])
```

从图 6-8(a) 可以看出, 在 $t \geq 0$ 定义域内, t_1 点是目标函数真正的全局最优值, 在最优化理论中又称为全局最优解, 由于初值选择不同的值可能得出不同的最优值, 其中有些是局部最优值, 所以这里给出的 `fminsearch()` 函数并不能保证求出全局最优值。事实上, 目前所有的最优化算法没有哪一种能保证能求出最优化问题的全局最优解, 只能说“更可能”获得全局最优解。

现在再考虑更大些的定义域, 即 $t \geq -0.5$, 则用下面的语句能绘制出该函数在新定义域内的曲线, 如图 6-8(b) 所示, 其中 $t_3 = -0.3340$, $f_3 = -1.9163$ 。

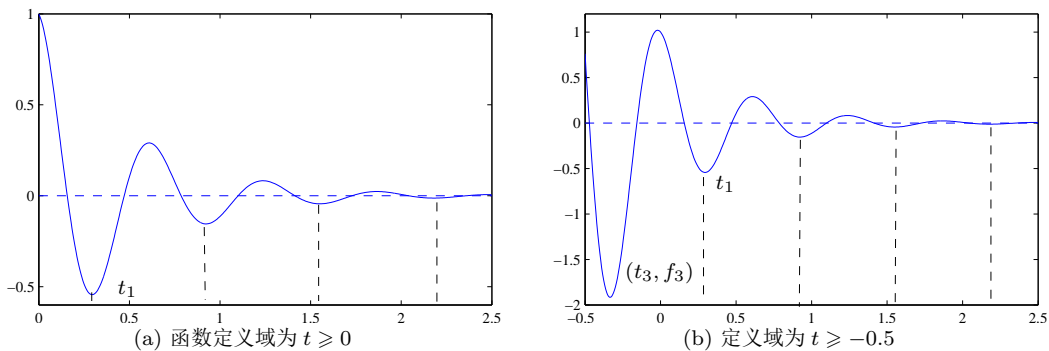


图 6-8 全局最优解与局部最优解

```
>> ezplot(y,[-0.5,2.5]); ylim([-2,1.2])
f=@(t)exp(-2*t).*cos(10*t)+exp(-3*(t+2)).*sin(2*t);
t0=-0.2; [t2,f2]=fminsearch(f,t0)
```

从图 6-8(b) 可见, 前面得出的全局最优解在新的定义域内就不再是全局最优解了, 比起新最优解, 它只是局部最优解, 若进一步扩大定义域, 则得出的 t_3 将也不再是全局最优解了, 而变成局部最优解。若将定义域扩展到 $t \in (-\infty, \infty)$ 区间, 则原问题没有真正意义的全局最优解。

通过上面的例子可以看出, 利用 MATLAB 的求解函数或其他现成的最优化问题求解函数可能得出局部最优解, 而不是全局最优解, 这就需要读者自己去试不同的初值, 看看得出的最优解是否相同, 如果不同, 则比较一下哪个是局部最优解。遗传算法提供了一种同时

试测不同初值的算法,在求解全局最优解上有一定的改进,但也不能保证得出全局最优解。有关遗传算法及其在最优化问题中的应用请具体参见第 10 章 10.4 节。

6.2.4 利用梯度求解最优化问题

有时最优化问题求解速度较慢,有时甚至无法搜索到较精确的最优点,尤其是变量较多的最优化问题,所以需要引入目标函数梯度,以加快计算速度,改进搜索精度。然而,有时计算梯度也是需要时间的,也会影响整个运算速度,所以实际求解时应该考虑是不是值得引入梯度的概念。

在利用 MATLAB 最优化工具箱求解最优化问题时,也应该和目标函数在同一函数中描述梯度函数,亦即这时 MATLAB 的目标函数应该返回两个变量,第一个变量仍然表示目标函数,第二个变量可以返回梯度函数。同时,还应该将求解控制变量的 `GradObj` 属性设置成 'on',这样就可以利用梯度来求解最优化问题了。

例 6-18 试求解 Rosenbrock 函数 $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ 的无约束最优化问题。

解 从目标函数可以看出,由于它为两个平方数的和,所以当 $x_2 = x_1 = 1$ 时,整个目标函数有最小值 0。用下面语句可以绘制出目标函数的三维等高线图,如图 6-9 所示。

```
>> [x,y]=meshgrid(0.5:0.01:1.5); z=100*(y.^2-x).^2+(1-x).^2;
    contour3(x,y,z,100), zlim([0,310])
```

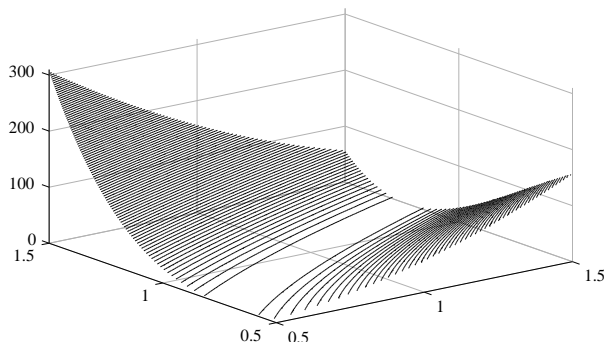


图 6-9 Rosenbrock 目标函数的三维等高线图

由得出的曲线看,其最小值点在图中的一个很窄的白色带状区域内,故 Rosenbrock 目标函数又称为香蕉函数,而在这个区域内的函数值变化较平缓,这就给最优化求值带来很多麻烦。该函数经常用来测试最优化算法的优劣。现在观察用下面语句求解最优化问题

```
>> f=@(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
    ff=optimset; ff.TolX=1e-20; ff.TolFun=1e-20; x=fminunc(f,[0;0],ff)
```

这时得出的最优解为 $x = [0.99999558847268, 0.99999116718532]^T$ 。可见,即使设置了苛刻的终止条件,该算法也无法精确搜索到真值 (1, 1),用传统的最速下降法更无法搜索到真值,所以这时需要引入梯度的概念。

对给定的 Rosenbrock 函数,利用符号运算工具箱即可以求出其梯度向量

```
>> syms x1 x2; f=100*(x2-x1^2)^2+(1-x1)^2; J=jacobian(f,[x1,x2])
```

可以求出梯度向量为 $\mathbf{J} = [-400(x_2 - x_1^2)x_1 - 2 + 2x_1, 200x_2 - 200x_1^2]$ 。这时,可以在目标函数中描述其梯度,故需要重新编写目标函数为

```
function [y,Gy]=c6fun3(x)
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
Gy=[-400*(x(2)-x(1)^2)*x(1)-2+2*x(1); 200*x(2)-200*x(1)^2];
```

这样,就应该给出如下命令得出 $\mathbf{x} = [1.000000000000012, 1.000000000000023]^T$

```
>> ff.GradObj='on'; x=fminunc(@c6fun3,[0;0],ff)
```

可见,引入了梯度则可以明显加快搜索的进度,且最优解也基本上逼近真值,这是不使用梯度不可能得到的,所以从本例可以看出梯度在搜索中的作用。然而,在有些例子中引入梯度也不是很必要,因为梯度本身的计算和编程需要更多的时间。

如果用结构体的方式描述最优化问题,则可以得出与前面一致的解。

```
>> problem.solver='fminunc'; ff=optimset; problem.objective=@c6fun3;
ff.GradObj='on'; ff.TolX=1e-20; ff.TolFun=1e-20; problem.options=ff;
problem.x0=[2; 1]; [x,b,c,d]=fminunc(problem)
```

值得指出的是,Rosenbrock 函数是为检测寻优算法而建立起来的人造函数,解决该问题的有效方法需要引入目标函数的梯度。实际应用中,很多寻优算法都是无需梯度信息的,利用目标函数本身的信息即可成功地解决数值寻优的问题。

6.2.5 带有变量边界约束的最优化问题求解

前面介绍的最优化问题是纯粹的无约束最优化问题。在实际应用中,更常见的无约束最优化问题并不完全是绝对的无约束问题,通常优化变量需要在指定的范围内选择,所以这样的问题一般可以表示成

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \end{aligned} \quad (6-2-3)$$

其中,记号 s.t. 是英文 subject to 的缩写,表示满足后面的关系。所以式(6-2-3)所描述的问题是, \mathbf{x} 在指定的范围内取多少时能使得目标函数取最优值。这样的问题由 `fminsearch()` 函数是不能直接求解的。John D'Errico 开发的 `fminsearchbnd()` 函数扩展了现有函数的功能,能直接求解这样的问题^[2],该函数的调用格式为

```
 $\mathbf{x} = \text{fminsearchbnd}(\text{Fun}, \mathbf{x}_0, \mathbf{x}_m, \mathbf{x}_M)$ 
 $[\mathbf{x}, f, \text{flag}, \text{out}] = \text{fminsearchbnd}(\text{Fun}, \mathbf{x}_0, \mathbf{x}_m, \mathbf{x}_M, \text{opt}, p_1, p_2, \dots)$ 
```

如果上界或下界约束没有给出,则可以将其设置为空矩阵 `[]`。

例 6-19 重新考虑例 6-18 中研究的 Rosenbrock 函数的最优化问题。显然, $x_1 = x_2 = 1$ 是该问题的最优解。如果 x_1 和 x_2 的允许范围为 $x_1 \in (2, 4), x_2 \in (3, 6)$, 试得出满足要求的最优解。

解 根据 x_1, x_2 的范围,可以直接得出 $\mathbf{x}_m = [2, 3]^T, \mathbf{x}_M = [4, 6]^T$, 这样调用 `fminsearchbnd()` 函数则可以得出在容许范围内的最优解为 $\mathbf{x} = [2, 3.9999996]^T$ 。


```
>> f=@(x)[100*(x(2)-x(1)^2)^2+(1-x(1))^2]; xm=[2,3]; xM=[4,6];
x=fminsearchbnd(f,[0,0],xm,xM)
```

6.3 有约束最优化问题的计算机求解

有约束最优化问题的一般描述为

$$\min_{\mathbf{x} \text{ s.t. } \mathbf{G}(\mathbf{x}) \leq \mathbf{0}} f(\mathbf{x}) \quad (6-3-1)$$

其中, $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ 。该数学表示的含义为求取一组 \mathbf{x} 向量,使得在满足约束条件 $\mathbf{G}(\mathbf{x}) \leq \mathbf{0}$ 的前提下能够使目标函数 $f(\mathbf{x})$ 最小化。在实际遇到的最优化问题中,有时约束条件可能是很复杂的,它既可以是等式约束,也可以是不等式约束;既可以是线性的,也可能是非线性的,有时甚至不能用纯数学函数来描述。

6.3.1 约束条件与可行解区域

满足约束条件 $\mathbf{G}(\mathbf{x}) \leq \mathbf{0}$ 的 \mathbf{x} 范围称为可行解区域(feasible region)。下面通过例子演示二元问题的可行解范围与图解结果。

例 6-20 考虑下面二元最优化问题的求解,试用图解方法对该问题进行研究。

$$\begin{aligned} \max \quad & -x_1^2 - x_2 \\ \mathbf{x} \text{ s.t. } \quad & \begin{cases} 9 \geq x_1^2 + x_2^2 \\ x_1 + x_2 \leq 1 \end{cases} \end{aligned}$$

解 若在 $[-3, 3]$ 区间生成网格,则可以得出无约束时目标函数的三维图形数据。

```
>> [x1,x2]=meshgrid(-3:.1:3); % 生成网格型矩阵
z=-x1.^2-x2; % 计算出矩阵上各点的高度
```

引入了约束条件,则在图形上需要将约束条件以外的点剔除掉。2.5 节中介绍了如何剔除三维图形中点的方法,具体的方法是找出这些点的横纵坐标值,将其函数值设置成不定式 NaN 即可剔除这些坐标点。这样可以使用如下的语句进行求解

```
>> i=find(x1.^2+x2.^2>9); z(i)=NaN; % 找出  $x_1^2 + x_2^2 > 9$  的坐标,并置函数值为 NaN
i=find(x1+x2>1); z(i)=NaN; % 找出  $x_1 + x_2 > 1$  的坐标,并置函数值为 NaN
surf(x1,x2,z); shading interp;
```

该语句可以直接绘制出如图 6-10 (a) 所示的三维图形,若想从上向下观察该图形,则可以使用 `view(0,90)` 命令,这样可以得出如图 6-10 (b) 所示的二维投影图。图形上的区域为相应最优化问题的可行区域,即满足约束条件的区域。该区域内对应目标函数的最大值就是原问题的解,故从图形可以直接得出结论,问题的解为 $x_1 = 0, x_2 = -3$, 用 `max(z(:))` 可以得出最大值为 3。

对于一般的一元问题和二元问题,可以用图解法直接得出问题的最优解。但对于一般的多元问题和较复杂的问题,则不适合用图解法求解,而只能用数值解的方法进行求解,也没有检验全局最优性的方法。

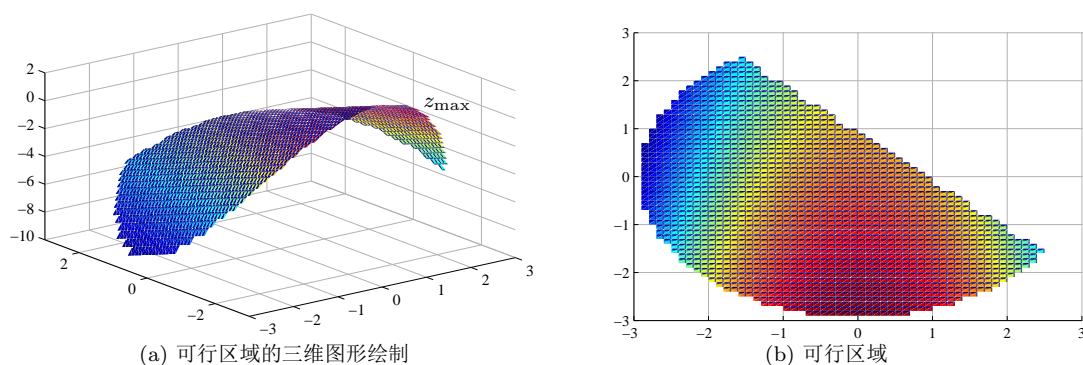


图 6-10 二维最优化问题的图解法

6.3.2 线性规划问题的计算机求解

线性规划问题是一类特殊的问题,也是最简单的有约束最优化问题。在线性规划中,目标函数和约束函数都是线性的,其整个问题的数学描述为

$$\begin{aligned} \min \quad & \mathbf{f}^T \mathbf{x} \\ \mathbf{x} \text{ s.t.} \quad & \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{B} \\ \mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{B}_{\text{eq}} \\ \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \end{cases} \end{aligned} \quad (6-3-2)$$

为描述原问题的方便及求解的高效性起见,这里的约束条件已经进一步细化为线性等式约束 $\mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{B}_{\text{eq}}$,线性不等式约束 $\mathbf{A}\mathbf{x} \leq \mathbf{B}$, \mathbf{x} 变量的上界向量 \mathbf{x}_M 和下界向量 \mathbf{x}_m ,使得 $\mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M$ 。

对不等式约束来说,MATLAB 定义的标准型是“ \leq ”关系式。如果约束条件中某个式子是“ \geq ”关系式,则在不等号两边同时乘以 -1 就可以转换成“ \leq ”关系式了。

线性规划是一类最简单的有约束最优化问题,求解线性规划问题有多种算法。其中,单纯形法是最有效的一种方法,MATLAB 的最优化工具箱中实现了该算法,提供了求解线性规划问题的 `linprog()` 函数。该函数的调用格式为

$$[\mathbf{x}, f_{\text{opt}}, \text{flag}, \mathbf{c}] = \text{linprog}(\mathbf{f}, \mathbf{A}, \mathbf{B}, \mathbf{A}_{\text{eq}}, \mathbf{B}_{\text{eq}}, \mathbf{x}_m, \mathbf{x}_M, \mathbf{x}_0, \text{OPT})$$

其中, \mathbf{f} , \mathbf{A} , \mathbf{B} , \mathbf{A}_{eq} , \mathbf{B}_{eq} , \mathbf{x}_m , \mathbf{x}_M 与前面约束与目标函数公式中的记号是完全一致的, \mathbf{x}_0 为初始搜索点。各个矩阵约束如果不存在,则应该用空矩阵来占位。OPT 为控制选项,该函数还允许使用附加参数 p_1, p_2, \dots 。最优化运算完成后,结果将在变量 \mathbf{x} 中返回,最优化的目标函数将在 f_{opt} 变量中返回。我们将通过下面的例子来演示线性规划的求解问题。

例 6-21 试求解下面的线性规划问题

$$\begin{aligned} \min \quad & -2x_1 - x_2 - 4x_3 - 3x_4 - x_5 \\ \mathbf{x} \text{ s.t.} \quad & \begin{cases} 2x_2 + x_3 + 4x_4 + 2x_5 \leq 54 \\ 3x_1 + 4x_2 + 5x_3 - x_4 - x_5 \leq 62 \\ x_1, x_2 \geq 0, x_3 \geq 3.32, x_4 \geq 0.678, x_5 \geq 2.57 \end{cases} \end{aligned}$$

解 从给出的数学式子可以看出,其目标函数可以用其系数向量 $\mathbf{f} = [-2, -1, -4, -3, -1]^T$ 表示,不等式约束有两个,即

$$A = \begin{bmatrix} 0 & 2 & 1 & 4 & 2 \\ 3 & 4 & 5 & -1 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 54 \\ 62 \end{bmatrix}$$

另外, 由于没有等式约束, 故可以定义 A_{eq} 和 B_{eq} 为空矩阵。由给出的数学问题还可以看出, x 的下界可以定义为 $x_m = [0, 0, 3.32, 0.678, 2.57]^T$, 且对上界没有限制, 故可以将其写成空矩阵。由前面的分析, 可以给出如下的 MATLAB 命令来求解线性规划问题, 并立即得出结果为 $x = [19.785, 0, 3.32, 11.385, 2.57]^T, f_{opt} = -89.5750$ 。

```
>> f=-[2 1 4 3 1]'; A=[0 2 1 4 2; 3 4 5 -1 -1]; B=[54; 62]; Ae=[];
Be=[]; xm=[0,0,3.32,0.678,2.57]; ff=optimset; ff.LargeScale='off';
ff.TolX=1e-15; ff.TolFun=1e-20; ff.TolCon=1e-20;
[x,f_opt,key,c]=linprog(f,A,B,Ae,Be,xm,[],[],ff)
```

从列出的结果看, 由于 key 值为 1, 故求解是成功的。以上只用了 5 步就得出了线性规划问题的解, 可见求解程序功能是很强大的, 可以很容易得出线性规划问题的解。

线性规划问题也可以由结构体变量 **problem** 直接描述, 需要用其 lb 和 ub 成员变量描述决策变量的下限 x_m 和上限 x_M , 用 Aineq、Bineq、Aeq、Beq 成员变量描述 A 、 B 、 A_{eq} 、 B_{eq} 矩阵, 而将其 solver 成员变量设置成 'linprog', 目标函数成员变量 f 设置成常数向量 f , 这样就可以用 **linprog(problem)** 直接求解。

例 6-22 可以用下面的结构体方式构造出线性规划问题的变量 P, 某些值为默认值的成员变量可以不指定, 如 Aeq 成员变量的默认值为空矩阵, 既然本问题没有涉及 A_{eq} 矩阵, 可以不给出该成员变量, 可以同样解决原始问题, 得出的解与前面方法完全一致。这里由于初值和决策变量上限采用了默认值, 所以无需对结构体的相应成员变量赋值。

```
>> P.f=-[2 1 4 3 1]'; P.Aineq=[0 2 1 4 2; 3 4 5 -1 -1]; P.Bineq=[54; 62];
P.lb=[0,0,3.32,0.678,2.57]; P.solver='linprog'; ff=optimset;
ff.TolX=1e-15; ff.TolFun=1e-20; ff.TolCon=1e-20; P.options=ff;
[x,f_opt,key,c]=linprog(P)
```

例 6-23 考虑下面的 4 元线性规划问题, 试用 MATLAB 的最优化工具箱求解此问题。

$$\begin{aligned} & \max \quad \frac{3}{4}x_1 - 150x_2 + \frac{1}{50}x_3 - 6x_4 \\ & x \text{ s.t. } \begin{cases} x_1/4 - 60x_2 - x_3/50 + 9x_4 \leq 0 \\ -x_1/2 + 90x_2 + x_3/50 - 3x_4 \geq 0 \\ x_3 \leq 1, x_1 \geq -5, x_2 \geq -5, x_3 \geq -5, x_4 \geq -5 \end{cases} \end{aligned}$$

解 原问题中应该求解的是最大值问题, 所以需要首先将之转换成最小化问题, 即将原目标函数乘以 -1 , 则目标函数将改写成 $-3x_1/4 + 150x_2 - x_3/50 + 6x_4$ 。套用线性规划的格式可以得出 f^T 向量为 $[-3/4, 150, -1/50, 6]$ 。

再分析约束条件, 可见, 由最后一条可以写成 $x_i \geq -5$, 所以可确定自变量的最小值向量为 $x_m = [-5; -5; -5; -5]$ 。类似地, 还能写出自变量的最大值向量为 $x_M = [\text{Inf}; \text{Inf}; 1; \text{Inf}]$, 其中可以使用 Inf 表示 $+\infty$ 。约束条件的前两条均为不等式约束, 其中第 2 条为 \geq 表示, 需要将两端均乘以 -1 , 转换成 \leq 不等式, 这样可以写出不等式约束为

$$A = \begin{bmatrix} 1/4 & -60 & -1/50 & 9 \\ 1/2 & -90 & -1/50 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

由于原问题中没有等式约束,故应该令 $A_{eq} = []$, $B_{eq} = []$ 。最终可以输入如下的命令来求解此最优化问题,得出原问题的最优解。

```
>> f=[-3/4,150,-1/50,6]; Aeq=[]; Beq=[];
A=[1/4,-60,-1/50,9; 1/2,-90,-1/50,3]; B=[0;0];
xm=[-5;-5;-5;-5]; xM=[Inf;Inf;1;Inf]; ff=optimset;
ff.TolX=1e-15; ff.TolFun=1e-20; TolCon=1e-20; ff.LargeScale='off';
[x,f_opt,key,c]=linprog(f,A,B,Aeq,Beq,xm,xM,[0;0;0;0],ff)
```

可见,经过 10 步迭代,就能以很高精度得出原问题的最优解为 $x = [-5, -0.1947, 1, -5]^T$ 。最后一个语句可以用下面语句取代,得出完全一致的结果。注意,求解之前应该采用 clear 命令清除 P 变量,否则以前使用的 P 变量的一些成员变量可能遗留下来,影响本次求解

```
>> clear P; P.f=f; P.Aineq=A; P.Bineq=B; P.solver='linprog';
P.lb=xm; P.ub=xM; P.options=ff; linprog(P)
```

例 6-24 在一些研究领域中经常遇到下面类型的线性规划问题,自变量不是前面介绍的单下标的,而是双下标或多重下标的,试求解下面的线性规划问题

$$\begin{aligned} \min \quad & 2800(x_{11} + x_{21} + x_{31} + x_{41}) + 4500(x_{12} + x_{22} + x_{32}) + 6000(x_{13} + x_{23}) + 7300x_{14} \\ \text{s.t.} \quad & \begin{cases} x_{11} + x_{12} + x_{13} + x_{14} \geq 15 \\ x_{12} + x_{13} + x_{14} + x_{21} + x_{22} + x_{23} \geq 10 \\ x_{13} + x_{14} + x_{22} + x_{23} + x_{31} + x_{32} \geq 20 \\ x_{14} + x_{23} + x_{32} + x_{41} \geq 12 \\ x_{ij} \geq 0, (i=1,2,3,4, j=1,2,3,4) \end{cases} \end{aligned}$$

解 这样的问题显然不能直接用前面介绍的方法直接求解,而应该首先将原问题转换成单下标自变量的最优化问题。为做这样的转换,应该重新选定变量,例如令 $x_1 = x_{11}, x_2 = x_{12}, x_3 = x_{13}, x_4 = x_{14}, x_5 = x_{21}, x_6 = x_{22}, x_7 = x_{23}, x_8 = x_{31}, x_9 = x_{32}, x_{10} = x_{41}$, 这样将原问题手工改写成

$$\begin{aligned} \min \quad & 2800(x_1 + x_5 + x_8 + x_{10}) + 4500(x_2 + x_6 + x_9) + 6000(x_3 + x_7) + 7300x_4 \\ \text{s.t.} \quad & \begin{cases} -(x_1 + x_2 + x_3 + x_4) \leq -15 \\ -(x_2 + x_3 + x_4 + x_5 + x_6 + x_7) \leq -10 \\ -(x_3 + x_4 + x_6 + x_7 + x_8 + x_9) \leq -20 \\ -(x_4 + x_7 + x_9 + x_{10}) \leq -12 \\ x_i \geq 0, i=1,2,\dots,10 \end{cases} \end{aligned}$$

这样就可以用下面的语句解出问题的最优解

```
>> f=2800*[1 0 0 0 1 0 0 1 0 1]+4500*[0 1 0 0 0 1 0 0 1 0]+...
6000*[0 0 1 0 0 0 1 0 0 0]+7300*[0 0 0 1 0 0 0 0 0 0];
A=-[1 1 1 1 0 0 0 0 0 0; 0 1 1 1 1 1 1 1 0 0;
0 0 1 1 0 1 1 1 1 0; 0 0 0 1 0 0 1 0 1 1];
B=[15; 10; 20; 12]; xm=[0 0 0 0 0 0 0 0 0 0]; Aeq=[]; Beq=[];
x=linprog(f,A,B,Aeq,Beq,xm)
```

得出 $x = [4.2069, 0, 0, 10.7931, 0, 0, 0, 8, 1.2069, 0.0000]$ 。将得出的结果再反代回双下标自变量,则可得 $x_{11} = 4.2069, x_{14} = 10.7931, x_{31} = 8, x_{32} = 1.2069$, 其余的自变量均为 0。

6.3.3 二次型规划的求解

二次型规划问题是另一种简单的有约束最优化问题,其目标函数为 \mathbf{x} 的二次型形式,约束条件仍然为线性不等式约束。一般二次型规划问题的数学表示为

$$\begin{aligned} \min \quad & \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x} \\ \mathbf{x} \text{ s.t. } \quad & \begin{cases} \mathbf{A} \mathbf{x} \leq \mathbf{B} \\ \mathbf{A}_{\text{eq}} \mathbf{x} = \mathbf{B}_{\text{eq}} \\ \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \end{cases} \end{aligned} \quad (6-3-3)$$

和线性规划问题相比,二次型规划目标函数中多了一个二次项 $\mathbf{x}^T \mathbf{H} \mathbf{x}$ 来描述 x_i^2 和 $x_i x_j$ 项。MATLAB 的最优化工具箱提供了求解二次型规划问题的 `quadprog()` 函数,其调用格式为 `[x,f_opt,flag,c]=quadprog(H,f,A,B,Aeq,Beq,xm,xM,x0,OPT)`。

例 6-25 试求解下面的四元二次型规划问题

$$\begin{aligned} \min \quad & (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (x_4 - 4)^2 \\ \mathbf{x} \text{ s.t. } \quad & \begin{cases} x_1 + x_2 + x_3 + x_4 \leq 5 \\ 3x_1 + 3x_2 + 2x_3 + x_4 \leq 10 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases} \end{aligned}$$

解 首先应该将原始问题写成二次型规划的模式。展开目标函数得

$$\begin{aligned} f(\mathbf{x}) &= x_1^2 - 2x_1 + 1 + x_2^2 - 4x_2 + 4 + x_3^2 - 6x_3 + 9 + x_4^2 - 8x_4 + 16 \\ &= x_1^2 + x_2^2 + x_3^2 + x_4^2 - 2x_1 - 4x_2 - 6x_3 - 8x_4 + 30 \end{aligned}$$

因为目标函数中的常数对最优化结果没有影响,所以可以放心地略去。这样就可以将二次型规划标准型中的 \mathbf{H} 矩阵和 \mathbf{f}^T 向量写为 $\mathbf{H} = \text{diag}([2, 2, 2, 2])$, $\mathbf{f}^T = [-2, -4, -6, -8]$, 从而可以给出下列 MATLAB 命令来求解二次型最优化问题

```
>> f=[-2,-4,-6,-8]; H=diag([2,2,2,2]);
OPT=optimset; OPT.LargeScale='off'; % 不使用大规模问题算法
A=[1,1,1,1; 3,3,2,1]; B=[5;10]; Aeq=[]; Beq=[]; LB=zeros(4,1);
[x,f_opt]=quadprog(H,f,A,B,Aeq,Beq,LB,[],[],OPT)
```

这样得出的最优解为 $\mathbf{x} = [0, 0.6667, 1.6667, 2.6667]^T$, 目标函数的值为 -23.6667 。

套用二次型规划标准型时,一定要注意 \mathbf{H} 矩阵的生成,因为在式 (6-3-3) 中有一个 $1/2$ 项,所以在本例中, \mathbf{H} 矩阵对角元素是 2, 而不是 1。另外,这里得出的目标函数实际上不是原始问题中的最优函数,因为人为地除去了常数项。将得出的结果再补上已经除去了的常数项,则可以求出原问题目标函数的值为 6.3333。

二次型规划问题也可以由结构体描述,这时其 `H` 成员变量描述 \mathbf{H} 矩阵, `solver` 成员变量设置为 'quadprog' 即可。

6.3.4 一般非线性规划问题的求解

有约束非线性最优化问题的一般描述为

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \mathbf{x} \text{ s.t. } \quad & \mathbf{G}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \quad (6-3-4)$$

其中, $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ 。为求解方便, 约束条件还可以进一步细化为线性等式约束、线性不等式约束、 \mathbf{x} 变量的上下界向量, 还允许一般非线性函数的等式和不等式约束, 这时原规划问题可以改写成

$$\begin{aligned} & \min \quad f(\mathbf{x}) \\ & \mathbf{x} \text{ s.t. } \begin{cases} \mathbf{Ax} \leq \mathbf{B} \\ \mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{B}_{\text{eq}} \\ \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \\ \mathbf{C}(\mathbf{x}) \leq \mathbf{0} \\ \mathbf{C}_{\text{eq}}(\mathbf{x}) = \mathbf{0} \end{cases} \end{aligned} \quad (6-3-5)$$

MATLAB 最优化工具箱中提供了一个 `fmincon()` 函数, 专门用于求解各种约束下的最优化问题。该函数的调用格式为

`[x, f_opt, flag, c] = fmincon(F, x0, A, B, Aeq, Beq, xm, xM, CF, OPT, p1, p2, ...)`

其中, `F` 为给目标函数写的 M-函数或匿名函数, `x0` 为初始搜索点。各个矩阵约束如果不存在, 则应该用空矩阵来占位。`CF` 为给非线性约束函数写的 M-函数, `OPT` 为控制选项。最优化运算完成后, 结果将在变量 `x` 中返回, 最优化的目标函数将在 `f_opt` 变量中返回。和其他优化函数一样, 选项 `OPT` 有时是很重要的。

例 6-26 试求解下面的有约束最优化问题

$$\begin{aligned} & \min \quad 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\ & \mathbf{x} \text{ s.t. } \begin{cases} x_1^2 + x_2^2 + x_3^2 - 25 = 0 \\ 8x_1 + 14x_2 + 7x_3 - 56 = 0 \\ x_1, x_2, x_3 \geq 0 \end{cases} \end{aligned}$$

解 分析给出的最优化问题可以发现, 约束条件中含有非线性不等式, 故而不能使用二次型规划的方式求解, 必须用非线性规划的方式来求解。根据给出的问题可以直接写出目标函数为

```
>> f=@(x)1000-x(1)*x(1)-2*x(2)*x(2)-x(3)*x(3)-x(1)*x(2)-x(1)*x(3);
```

同时, 给出的两个约束条件均为等式约束, 所以应该写出非线性约束函数为

```
function [c,ceq]=opt_con1(x)
```

```
ceq=[x(1)*x(1)+x(2)*x(2)+x(3)*x(3)-25; 8*x(1)+14*x(2)+7*x(3)-56]; c=[];
```

非线性约束函数返回变量分为 `c` 和 `ceq` 两个量, 其中, 前者为不等式约束的数学描述, 后者为非线性等式约束, 如果某个约束不存在, 则应该将其值赋为空矩阵。

描述了给出的非线性等式约束后, 则 `A`, `B`, `Aeq`, `Beq` 都将为空矩阵了。另外, 应该给出搜索初值向量 $\mathbf{x}_m = [0, 0, 0]^T$, 因此, 可以调用 `fmincon()` 函数求解此约束最优化问题。

```
>> ff=optimset; ff.LargeScale='off';
ff.TolFun=1e-30; ff.TolX=1e-15; ff.TolCon=1e-20;
x0=[1;1;1]; xm=[0;0;0]; xM=[]; A=[]; B=[]; Aeq=[]; Beq=[];
[x,f_opt,c,d]=fmincon(f,x0,A,B,Aeq,Beq,xm,xM,@opt_con1,ff)
```

上述语句可以得出最优结果为 $\mathbf{x} = [3.5121, 0.2170, 3.5522]^T$, 目标函数最优值为 $f_{\text{opt}} = 961.7151$ 。由 `d` 的分量还可以看出, 求解过程中共调用目标函数 113 次。

非线性规划问题还可以通过下面的语句描述并求解, 得出的结果与前面得出的完全一致。可

见用结构体的方法描述原始问题将更简洁,求解也更直观。

```
>> clear P; P.objective=f; P.nonlcon=@opt_con1; P.x0=x0;
    P.lb=xm; P.options=ff; P.solver='fmincon'; [x,f_opt,c,d]=fmincon(P)
第2个约束条件是线性等式约束,可以将其从非线性约束函数中除去,则该约束函数简化为
function [c,ceq]=opt_con2(x)
    ceq=x(1)*x(1)+x(2)*x(2)+x(3)*x(3)-25; c=[];
```

线性等式约束可以由相应的矩阵定义出来,这时可以用下面的命令求解原始的最优化问题,且可以得出和前面完全一致的结果

```
>> x0=[1;1;1]; Aeq=[8,14,7]; Beq=56;
    [x,f_opt,c,d]=fmincon(f,x0,A,B,Aeq,Beq,xm,xM,@opt_con2,ff)
```

同样可以用结构体描述一般非线性规划问题,solver 成员变量应该设置成 'fmincon',非线性约束函数可以通过 nonlcon 成员变量表示出来,这样就可以直接调用 fmincon() 函数直接求解原始问题。

例 6-27 试求出下面有约束非线性规划问题的解。

$$\begin{aligned} \min \quad & e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1) \\ \text{s.t.} \quad & \begin{cases} x_1 + x_2 \leq 0 \\ -x_1x_2 + x_1 + x_2 \geq 1.5 \\ x_1x_2 \geq -10 \\ -10 \leq x_1, x_2 \leq 10 \end{cases} \end{aligned}$$

解 由下面的语句可以先描述出原始问题的约束函数

```
function [c,ce]=c6exmcon(x)
    ce=[]; c=[x(1)+x(2); x(1)*x(2)-x(1)-x(2)+1.5; -10-x(1)*x(2)];
```

这样,可以给出下面的命令直接求解原始的最优化问题

```
>> clear P; P.nonlcon=@c6exmcon; P.solver='fmincon'; P.options=optimset;
    P.objective=@(x)exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
    ff=optimset; ff.TolX=1e-20; ff.TolFun=1e-20; P.options=ff;
    P.lb=[-10; -10]; P.ub=-P.lb; P.x0=[0;0]; [x,f1,flag]=fmincon(P)
```

该函数运行结束后将显示得出的“最优解”为 $\mathbf{x} = [0.4195, 0.4195]^T$, 目标函数值为 $f_1 = 5.4737$ 。仔细观察得出的解,特别是 flag 变量会发现,该值为 0,并不是我们期望的正数,所以得出的结果并非原问题的最优解,得到的警告信息提示为“fmincon stopped because it exceeded the function evaluation limit”,说明目标函数调用次数超过最高允许次数,函数调用异常终止。这也提示我们在使用 MATLAB 求解科学运算问题时,不但应该关注得出的解,同样也应关注得出的其他信息。如果得出的解伴随警告或错误信息,则应该想办法重新求解原始问题。可以把前面得到的 \mathbf{x} 向量作为初值重新搜索最优解,求解后仍需要检验是不是有警告信息,或 flag 的值是不是正数,如果不是,则应该再以得出的结果为初值继续搜索。这样的搜索过程适合用循环结构实现,如果得出的 flag 为正数则结束循环。经过下面的求解语句可以得出原问题的最优解为 $\mathbf{x} = [1.1825, -1.7398]^T$, 最优目标函数为 3.0608,迭代次数为 $i = 4$ 。不过这样得出的结果仍可能是局部最优解,可以选择另一个初值

$x_0 = [-10, -10]$, 再重新求解, 看看不能不得到更好的解。

```
>> i=1; while 1, P.x0=x; [x,a,b]=fmincon(P); if b>0, break; end, i=i+1; end
```

例 6-28 试求解下面的最优化问题 [3]

$$\begin{aligned} \min \quad & k \\ \text{s.t.} \quad & \begin{cases} q_3 + 9.625q_1w + 16q_2w + 16w^2 + 12 - 4q_1 - q_2 - 78w = 0 \\ 16q_1w + 44 - 19q_1 - 8q_2 - q_3 - 24w = 0 \\ 2.25 - 0.25k \leq q_1 \leq 2.25 + 0.25k \\ 1.5 - 0.5k \leq q_2 \leq 1.5 + 0.5k \\ 1.5 - 1.5k \leq q_3 \leq 1.5 + 1.5k \end{cases} \end{aligned}$$

解 从给出的最优化问题看, 这里要求解的决策变量为 q, w, k , 而标准最优化方法只能求解向量型决策变量, 所以应该作变量替换, 把要求解的决策变量由决策变量向量表示出来。对本例来说, 可以引入 $x_1 = q_1, x_2 = q_2, x_3 = q_3, x_4 = w, x_5 = k$, 另外, 需要将一些不等式进一步处理一下。这样, 可以将原始问题手工改写成

$$\begin{aligned} \min \quad & x_5 \\ \text{s.t.} \quad & \begin{cases} x_3 + 9.625x_1x_4 + 16x_2x_4 + 16x_4^2 + 12 - 4x_1 - x_2 - 78x_4 = 0 \\ 16x_1x_4 + 44 - 19x_1 - 8x_2 - x_3 - 24x_4 = 0 \\ -0.25x_5 - x_1 \leq -2.25 \\ x_1 - 0.25x_5 \leq 2.25 \\ -0.5x_5 - x_2 \leq -1.5 \\ x_2 - 0.5x_5 \leq 1.5 \\ -1.5x_5 - x_3 \leq -1.5 \\ x_3 - 1.5x_5 \leq 1.5 \end{cases} \end{aligned}$$

这样可以由下面语句描述原问题的非线性约束条件

```
function [c,ceq]=c6exnls(x), c=[];
ceq=[x(3)+9.625*x(1)*x(4)+16*x(2)*x(4)+16*x(4)^2+12-4*x(1)-x(2)-78*x(4);
16*x(1)*x(4)+44-19*x(1)-8*x(2)-x(3)-24*x(4)];
```

为方便起见这里采用结构体形式描述原始问题。可以随机选择初值求解原问题, 从而得出原问题的解为 $x = [1.9638, 0.9276, -0.2172, 0.0695, 1.1448]$, 且标志 flag 为 1, 说明求解成功。

```
>> clear P; P.objective=@(x)x(5); P.nonlcon=@c6exnls; P.solver='fmincon';
P.Aineq=[-1,0,0,0,-0.25; 1,0,0,0,-0.25; 0,-1,0,0,-0.5;
0,1,0,0,-0.5; 0,0,-1,0,-1.5; 0,0,1,0,-1.5];
P.Bineq=[-2.25; 2.25; -1.5; 1.5; -1.5; 1.5]; P.options=optimset;
P.x0=rand(5,1); [x,fm,flag]=fmincon(P)
```

值得指出的是, 用随机选择初值的方法有可能得到局部最优值, 所以考虑采用循环结构执行 100 次寻优, 每次采用一个新的随机数初值, 这样由下面语句求解原始问题很可能得到其全局最优解。经过实际运算可以得出 $x = [2.4544, 1.9088, 2.7263, 1.3510, 0.8175]^T$, 目标函数的值为 x_5 。

```
>> f0=fm; xx=x;
for i=1:100, P.x0=10*rand(5,1); [x,fm,flag]=fmincon(P);
if (flag>0 & fm<f0), f0=fm; xx=x; i, f0,xx, end
```


end

例 6-29 考虑例 6-26 中的最优化问题, 试利用梯度求解最优化问题, 并比较和原方法的优劣。

解 由给出的目标函数 $f(x)$ 可以立即求出下面的梯度函数 (或 Jacobi 矩阵)

```
>> syms x1 x2 x3; f=1000-x1*x1-2*x2*x2-x3*x3-x1*x2-x1*x3;
J=jacobian(f,[x1,x2,x3])
```

其数学形式可以写成

$$\mathbf{J} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \right]^T = \begin{bmatrix} -2x_1 - x_2 - x_3 \\ -4x_2 - x_1 \\ -2x_3 - x_1 \end{bmatrix}$$

有了梯度, 则可以重新改写目标函数为

```
function [y,Gy]=opt_fun2(x)
y=1000-x(1)*x(1)-2*x(2)*x(2)-x(3)*x(3)-x(1)*x(2)-x(1)*x(3);
Gy=[-2*x(1)-x(2)-x(3); -4*x(2)-x(1); -2*x(3)-x(1)];
```

其中, Gy 表示目标函数的梯度向量。再调用最优化求解函数将得出下面的结果

```
>> x0=[1;1;1]; xm=[0;0;0]; xM=[]; A=[]; B=[]; Aeq=[]; Beq=[];
ff=optimset; ff.GradObj='on'; ff.LargeScale='off';
ff.TolFun=1e-30; ff.TolX=1e-15; ff.TolCon=1e-20;
[x,f_opt,c,d]=fmincon(@opt_fun2,x0,A,B,Aeq,Beq,xm,xM,@opt_con1,ff)
```

采用结构体方法描述原始问题, 则可以由下面语句直接求解, 得出相同结果

```
>> clear P; P.x0=x0; P.lb=xm; P.options=ff; P.objective=@opt_fun2;
P.nonlcon=@opt_con1; P.solver='fmincon'; x=fmincon(P)
```

可见, 若已知目标函数的偏导数, 则仅需 86 步目标函数的调用就能求出原问题的解, 比前面需要的步数 (113 步) 明显减少。但考虑求取和编写梯度函数所需的时间, 实际需要的时间可能更多。注意, 若已知梯度函数, 则应该将 GradObj 选项设置成 'on', 否则不能识别该梯度。

6.4 混合整数规划问题的计算机求解

在很多应用领域中, 最优化问题的要求除了前面的满足约束条件的规则外, 还需要使得全部和部分决策变量取整数, 这类问题又称为整数规划。部分决策变量为整数的最优化问题又称为混合整数规划问题。若决策变量只能是 0 或 1, 这类规划问题又称为 0-1 规划问题。

6.4.1 整数线性规划问题的求解

混合整数线性规划的一般数学描述为

$$\begin{aligned} \min \quad & \mathbf{f}^T \mathbf{x} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{B} \\ \mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{B}_{\text{eq}} \\ \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \\ \hat{\mathbf{x}} \text{ 为整数} \end{cases} \end{aligned} \quad (6-4-1)$$

其中 \hat{x} 为变量 x 的子集。如果 \hat{x} 为全部的 x , 则原始问题为整数规划问题。

MATLAB 自身没有提供整数线性规划的函数, 但可以使用荷兰 Eindhoven 科技大学 Michel Berkelaar 等人开发的 LP_Solve 包中的 MATLAB 支持的 mex 文件, 不过该软件不支持当前的 MATLAB 版本, 后面将再介绍小规模非线性整数规划的穷举方法和一般非线性混合整数规划问题的求解函数。

6.4.2 整数规划问题的穷举方法

所谓穷举方法, 就是将决策变量所有可能的取值都衡量一番, 从满足约束条件的决策变量可能中找出目标函数值最下的组合, 这样的解即为原始问题的全局最优解。

如果已知自变量所在的区间, 则理论上可以考虑用穷举方法列举出区间内所有的变量组合, 逐个判定约束条件是否满足, 从满足的组合中逐个求取函数的值并排序, 由其最小值的对应关系可以简单地求解所需的自变量值。这个方法看似简单、直观, 但对稍微多些自变量的情形是不可行的, 因为这时计算量为天文数字。相应的数学问题又称为 NP 难 (non-polynomial hard) 问题, 故穷举方法只适合于极有限的小规模问题。

例 6-30 考虑例 6-21 中给出的线性规划问题, 为方便起见重新给出

$$\begin{aligned} \min \quad & -2x_1 - x_2 - 4x_3 - 3x_4 - x_5 \\ \text{s.t.} \quad & \begin{cases} 2x_2 + x_3 + 4x_4 + 2x_5 \leq 54 \\ 3x_1 + 4x_2 + 5x_3 - x_4 - x_5 \leq 62 \\ x_1, x_2 \geq 0, x_3 \geq 3.32, x_4 \geq 0.678, x_5 \geq 2.57 \end{cases} \end{aligned}$$

如果要求自变量 x_i 均为整数, 则原来的问题就变成整数线性规划问题, 试求解该整数规划问题。

解 对于小规模问题, 可以考虑采用穷举算法。人为假定 x_M 的各个元素均为 25, 当然采用下面语句就可以逐个求取函数值, 得出的全局最优解为 $x = [19, 0, 4, 10, 5]^T$ 。

```
>> N=25; [x1,x2,x3,x4,x5]=ndgrid(1:N,0:N,4:N,1:N,3:N);
i=find((2*x2+x3+4*x4+2*x5<=54) & (3*x1+4*x2+5*x3-x4-x5<=62));
x1=x1(i); x2=x2(i); x3=x3(i); x4=x4(i); x5=x5(i);
f=-2*x1-x2-4*x3-3*x4-x5; [fmin,ii]=sort(f);
index=ii(1); x=[x1(index),x2(index),x3(index),x4(index),x5(index)]
```

然而这里有两个问题值得注意。其一, 本算法得出的结果是 $x_1 \in [0, 25]$ 区间的最小值, 但这个概念不能随意拓展到此区间之外, 如果想将 25 变为 30, 在一般的计算机配置下都实现不了, 因为所需内存过大, 5 个变量的存储量为 $31^5 \times 5 \times 8/2^{20} = 1092.1\text{MB}$ 空间。所以在求解整数规划时不适合采用穷举算法。其二, 除了得出的最优解之外, 事实上还可以得出若干组合, 使得该规划问题有次最优解。可以显示排序后的函数值为 $x_1 = [-89, -88, -88, -88, -88, -88, -88, -88, -88, -87, -87]$ 。

```
>> fx=fmin(1:10)
in=ii(1:15); x=[x1(in),x2(in),x3(in),x4(in),x5(in),fmin(1:15)]
```

可见, 函数的最小值为 -89。此外, 还有若干个点的值为 -88, 求出最优解的同时, 还可以列出各个变量的次最优解, 如表 6-2 所示。

例 6-31 试求解下面的整数规划问题

表 6-2 最优解及部分次最优解

x_1	x_2	x_3	x_4	x_5	f	说明	x_1	x_2	x_3	x_4	x_5	f	说明	x_1	x_2	x_3	x_4	x_5	f	说明
19	0	4	10	5	-89	最优	19	0	4	9	7	-88	次优	11	0	8	10	3	-87	次优
18	0	4	11	3	-88	次优	16	0	6	8	8	-88	次优	10	0	9	9	4	-87	次优
17	0	5	10	4	-88	次优	20	0	4	7	11	-88	次优	8	0	10	9	4	-87	次优
15	0	6	10	4	-88	次优	15	0	6	10	3	-87	次优	5	0	12	8	5	-87	次优
12	0	8	9	5	-88	次优	13	0	7	10	3	-87	次优	18	0	4	10	5	-87	次优

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 \\ \text{s.t.} \quad & \begin{cases} -x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4 + 8 \geq 0 \\ -x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 + 10 \geq 0 \\ -2x_1^2 - x_2^2 - x_3^2 - 2x_4^2 + x_2 + x_4 + 5 \geq 0 \end{cases} \end{aligned}$$

解 选择感兴趣的决策变量整数范围为 $-N \sim N$, 并选择 $N = 30$, 则可以通过穷举方法得出问题的全局最优解为 $\mathbf{x} = [0, 1, 2, 0]^T$, 相应的目标函数为 -38 。除了最优解外, 还可以搜索出一批次最优解, 如 $[0, 0, 2, 0]$, $[0, 1, 2, 1]$, $[0, 1, 1, -1]$ 和 $[1, 2, 1, 0]$ 。

```
>> N=30; [x1 x2 x3 x4]=ndgrid(-N:N);
ii=find(-x1.^2-x2.^2-x3.^2-x4.^2-x1+x2-x3+x4+8>=0 & ...
        -x1.^2-2*x2.^2-x3.^2-2*x4.^2+x1+x4+10>=0 & ...
        -2*x1.^2-x2.^2-x3.^2-2*x4.^2+x2+x4+5>=0);
x1=x1(ii); x2=x2(ii); x3=x3(ii); x4=x4(ii);
ff=x1.^2+x2.^2+2*x3.^2+x4.^2-5*x1-5*x2-21*x3+7*x4;
[fm,ii]=sort(ff); k=ii(1:5); X=[x1(k),x2(k),x3(k),x4(k)],fm(1:5)
```

值得指出的是, 穷举方法只能求解所有决策变量的可能都可以列出的最优化问题, 如整数规划问题, 不能求解混合整数规划问题和一般最优化问题, 因为决策变量是连续可变的, 不可能将全部的可能都列出来, 所以这些问题只能通过搜索方法进行求解。

6.4.3 一般非线性整数规划问题与求解

前面介绍的穷举方法只适合于小规模整数规划问题, 在实际应用中经常要求解非线性整数规划或混合规划问题, 该领域中一种常用的算法是分枝定界 (branch and bound) 算法, 具体算法在这里不详细介绍, 只介绍一个基于该算法编写的现成函数 BNB20(), 可以用来求解一般非线性整数规划的问题。该函数是荷兰 Groningen 大学的 Koert Kuipers 编写的, 可以从 MathWorks 网站上直接下载^[4]。

由于该函数十余年没有更新, 对 MATLAB 的后续版本支持不理想, 包括个别语句对新版本不支持、不能使用匿名函数等描述目标函数、也不支持用结构体描述最优化问题, 另外输入变量和返回变量对混合整数规划的支持不理想, 我们对该函数做了适当的修改, 利于更好解决一般混合整数规划问题, 新版本改名为 BNB20_new(), 其调用格式为

```
[err,f,x]=BNB20_new(fun,x0,intlist,xm,xM,A,B,Aeq,Beq,CFun)
```

其中, 调用过程中的大部分输入变量与最优化工具箱函数几乎完全一致, 该函数直接调用

了最优化工具箱中的 `fmincon()` 函数,该函数还可以根据需要带附加参数,返回的变量 `err` 为函数的错误信息字符串, \mathbf{x} 和 f 分别为最优解和其函数值。标志向量 `intlist` 是与决策变量 \mathbf{x} 相同长度的向量,其第 i 个元素为 1 表示 $x(i)$ 为整数,为 0 则为非整数。作为一个特例,如果 `intlist(i)` 的值取 2,则表示相应的 $x(i)$ 为固定值,不参与优化。如果正确返回最优解,则 `err` 字符串为空字符串,这时,返回的 \mathbf{x} 为最优决策向量, f 为目标函数的最优值。否则,该字符串返回错误信息。

如果原始最优化问题用结构体 `P` 描述,应该将其成员变量 `intlist` 设置成前面介绍的 `intlist` 向量,这样就可以用 `[err,f,x] = BNB20_new(P)` 函数直接求解了。

例 6-32 试用 `BNB20_new()` 函数求解例 6-30 中给出的线性整数规划问题。

解 在修改后的 `BNB20_new()` 中允许使用匿名函数来描述目标函数。和前面介绍的线性规划问题求解不同,上限变量不能再选择为无穷大,而应该选择为较大的数值,例如均选择为 20000。同样,整数的下界如果给定为小数,新函数中允许自动向上取整转换。这样用下面的语句求解出的线性整数规划问题与例 6-30 得出的完全一致。

```
>> f=@(x)-[2 1 4 3 1]*x; xm=[0,0,3.32,0.678,2.57]'; x0=ceil(xm);
    A=[0 2 1 4 2; 3 4 5 -1 -1]; intlist=ones(5,1); Aeq=[]; Beq=[];
    B=[54; 62]; xM=20000*ones(5,1);
    [errmsg,fm,X]=BNB20_new(f,x0,intlist,xm,xM,A,B,Aeq,Beq)
```

若采用结构体形式描述原始问题,则可以给出下面的语句,结果也完全一致

```
>> clear P; P.objective=f; P.lb=xm; P.x0=x0; P.ub=xM;
    P.Aineq=A; P.Bineq=B; P.intlist=intlist; [errmsg,fm,X]=BNB20_new(P)
```

如果仍要求 x_1, x_4, x_5 为整数,其他两个变量为任意值,则应该修改一下 `intlist` 变量,将其设置为 `intlist = [1,0,0,1,1]`,则可以用下面的语句求出原问题的解为 $\mathbf{X} = [19, 0, 3.8, 11, 3]^T$ 。

```
>> intlist=[1,0,0,1,1]';
    [errmsg,fm,X]=BNB20_new(f,x0,intlist,xm,xM,A,B,Aeq,Beq)
```

如果采用下面的结构体形式求解原问题也将得出完全一致的结果。

```
>> P.intlist=[1,0,0,1,1]; [errmsg,fm,X]=BNB20_new(P)
```

例 6-33 对著名的 Rosenbrock 函数稍加修改,可以写出 $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 - (4.5543 - x_1)^2$, 试求解整数 x_1 和 x_2 ,使得

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \mathbf{x} \text{ s.t. } \quad & \begin{cases} -100 \leq x_1 \leq 100 \\ -100 \leq x_2 \leq 100 \end{cases} \end{aligned}$$

解 在一般最优化问题中, $(4.5543, 4.5543^2)$ 显然为最优点,这里考虑整数规划问题的求解方法。调用 `BNB20_new()` 函数,选择合适的上下界约束,可以直接得出原来问题的解为 $\mathbf{x} = [5, 25]^T$ 。

```
>> f=@(x) 100*(x(2)-x(1)^2)^2+(4.5543-x(1))^2;
    x0=[1;1]; xm=-1000*[1;1]; xM=1000*[1;1];
    A=[]; B=[]; Aeq=[]; Beq=[]; intlist=[1,1]';
    [errmsg,fm,x]=BNB20_new(f,x0,intlist,xm,xM,A,B,Aeq,Beq)
```

该搜索语句将搜索范围设置成 $-1000 \leq x_1, x_2 \leq 1000$, 可以联机得出原问题的解, 即使选择更大的搜索范围如 $-100000 \leq x_1, x_2 \leq 100000$, 调用 `BNB20_new()` 求解也不会增加太多的计算量。相反地, 如果用户为节省时间选择一个很小的搜索区间, 如 $x_{1,2} \in [-20, 20]$, 则将得出结果为 $x = [5, 20]$, 可见该值不是原问题的最优解

```
>> xm=-20*[1;1]; xM=20*[1;1];
[errmsg,f,x]=BNB20_new(f,x0,intlist,xm,xM,A,B,Aeq,Beq)
```

其实, 对这样小规模的问题, 选择较大的搜索范围, 如 $x_{1,2} \in (-1000, 1000)$, 用穷举搜索算法能立即得出问题的解, 和上述结果一致。更大的搜索范围将产生“out of memory”现象, 所以说穷举法对解决一般整数规划问题来说局限性较大, 有时不宜采用。

```
>> N=1000; [x1,x2]=meshgrid(-N:N); f=100*(x2-x1.^2).^2+(4.5543-x1).^2;
[fmin,i]=sort(f(:)); x=[x1(i(1)),x2(i(1))]
```

例 6-34 试求解下面的整数规划问题^[5]

$$\begin{aligned} \min \quad & x_1^3 + x_2^2 - 4x_1 + 4 + x_3^4 \\ \text{s.t.} \quad & \begin{cases} x_1 - 2x_2 + 12 + x_3 \geq 0 \\ -x_1^2 + 3x_2 - 8 - x_3 \geq 0 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases} \end{aligned}$$

解 由于原问题含有非线性约束, 可以编写下面的函数将其描述出来

```
function [c,ce]=c6exin1(x)
ce=[]; c=[-x(1)+2*x(2)-12-x(3); x(1)^2-3*x(2)+8+x(3)];
```

这样就可以调用下面的语句求解整数规划问题, 得出的解为 $x = [1, 3, 0]^T$ 。

```
>> clear P; P.objective=@(x)x(1)^3+x(2)^2-4*x(1)+4+x(3)^4;
P.intlist=[1;1;1]; P.nonlcon=@c6exin1; P.lb=[0;0;0];
P.ub=100*[1;1;1]; P.x0=P.ub; [err,fm x]=BNB20_new(P)
```

由于原始问题是小规模问题, 所以可以考虑采用穷举方法求解, 该方法得出的全局最优解和前面得出的完全一致。除此之外, 采用穷举方法还能求出一些次优解, 这是搜索方法做不到的。

```
>> N=200; [x1 x2 x3]=meshgrid(0:N);
ii=find(x1-2*x2+12+x3>=0 & -x1.^2+3*x2-8-x3>=0);
x1=x1(ii); x2=x2(ii); x3=x3(ii);
ff=x1.^3+x2.^2-4*x1+4+x3.^4; [fm,ij]=sort(ff);
k=ij(1:5); [x1(k) x2(k) x3(k)], fm(1:5)
```

例 6-35 试求解离散最优化问题^[6], 其中 x_1 是 0.25 的整数倍, x_2 是 0.1 的整数倍, 且 $x_2 \geq 3$ 。

$$\begin{aligned} \min \quad & 2x_1^2 + x_2^2 - 16x_1 - 10x_2 \\ \text{s.t.} \quad & \begin{cases} x_1^2 - 6x_1 + x_2 - 11 \leq 0 \\ -x_1x_2 + 3x_2 + e^{x_1-3} - 1 \leq 0 \end{cases} \end{aligned}$$

解 MATLAB 不能直接求解离散最优化问题, 不过既然这里给出了步距, 可以引入两个新的变量 $y_1 = 4x_1, y_2 = 10x_2$, 即采用变量替换 $x_1 = y_1/4, x_2 = y_2/10$, 这时原来的问题可以改写成下面的关于 y_i 的整数规划问题

$$\begin{aligned} \min \quad & 2y_1^2/16 + y_2^2/100 - 4y_1 - y_2 \\ \text{s.t.} \quad & \begin{cases} y_1^2/16 - 6y_1/4 + y_2/10 - 11 \leq 0 \\ -y_1y_2/40 + 3y_2/10 + e^{y_1/4-3} - 1 \leq 0 \\ y_2 \geq 30 \end{cases} \end{aligned}$$

可以用 MATLAB 直接写出非线性约束函数

```
function [c,ceq]=c6mdisp(y), ceq=[];
c=[y(1)^2/10-6*y(1)/4+y(2)/10-11; -y(1)*y(2)/40+3*y(2)/10+exp(y(1)/4-3)-1];
```

调用 BNB20_new() 则可以采用下面语句直接求解最优化问题,假设 y_1 搜索的上下限是 ± 200 (即 x_1 的上下限 ± 50), y_2 的上限为 200, 下限为 30 (即 $3 \leq x_2 \leq 20$), 这样调用下面的语句可以搜索出问题的最优解为 $x = [4, 5]^T$, 该值与文献 [5] 给出的 (4, 4.75) 略有不同, 这里给出的解在满足约束条件的前提下目标函数略小于该解, 说明此解更合适

```
>> clear P; P.objective=@(y)2*y(1)^2/16+y(2)^2/100-4*y(1)-y(2);
P.nonlcon=@c6mdisp; P.lb=[-200;30]; P.ub=[200;200]; P.intlist=[1;1];
P.x0=[12;30]; [errmsg,ym,y]=BNB20_new(P); x=[y(1)/4,y(2)/10]
```

穷举方法并不只可用于整数规划问题, 它也同样适用于离散最优化问题。假设决策变量的搜索范围为 $(-20, 20)$, 则可以给出如下的语句, 同样可以得出全局最优解为 (4, 5), 同时也可以得出一些次最优解, 如 (4, 5.1), (4, 4.9), (4, 4.8), (4, 5.2) 等, 这些点的目标函数均略大于 (4, 5)。

```
>> [x1 x2]=meshgrid(-20:0.25:20,3:0.1:20);
ii=find(x1.^2-6*x1+x2-11<=0 & -x1.*x2+3*x2+exp(x1-3)-1<=0);
x1=x1(ii); x2=x2(ii); ff=2*x1.^2+x2.^2-16*x1-10*x2; [fm,ij]=sort(ff);
k=ij(1:5); X=[x1(k) x2(k)], fm(1:5)
```

6.4.4 0-1 规划问题求解

所谓 0-1 规划, 即指决策变量 x_i 的值或者为 0, 或者为 1。所以求解 0-1 规划看起来很简单, 让每个自变量 x_i 遍取 0、1, 在得出的组合中选择既满足约束条件又使目标函数取最小值的项。而事实上, 随着问题规模的增大, 这样的计算量将按指数增长。例如, 自变量的个数为 n , 则可能的排列数为 2^n , 在 n 较大时其值可能是个天文数字, 故仍然需要考虑其他算法进行求解。

MATLAB 函数 **$x = \text{bintprog}(f, A, B, A_{eq}, B_{eq})$** 可以用来求解 0-1 线性规划问题, 但该函数不能直接求解非线性 0-1 规划问题。bintprog() 函数也可以求解由结构体描述的最优控制问题, 其 solver 成员变量应该设置为 'bintprog'。后面还将演示非线性 0-1 规划问题的求解。

例 6-36 试求解下面给出的 0-1 线性规划问题

$$\begin{aligned} \min \quad & -3x_1 + 2x_2 + 5x_3 \\ \text{s.t.} \quad & \begin{cases} x_1 + 2x_2 - x_3 \leq 2 \\ x_1 + 4x_2 + x_3 \leq 4 \\ x_1 + x_2 \leq 3 \\ 4x_2 + x_3 \leq 6 \end{cases} \end{aligned}$$

解 套用所需的最优化模型,可以立即求出 f 、 A 和 B 矩阵,这样可以给出如下的语句求解 0-1 线性规划问题,得出 $x = [1, 0, 1]^T$

```
>> f=[-3,2,-5]; A=[1 2 -1; 1 4 1; 1 1 0; 0 4 1]; B=[2;4;5;6];
x=bintprog(f,A,B,[],[])
```

如果采用结构体描述最优化问题,则可以给出下面语句,得出的结果也完全一致

```
>> clear P; P.f=f; P.Aineq=A; P.Bineq=B; P.solver='bintprog';
P.options=optimset; x=bintprog(P)
```

对于小规模问题,当然可以采用下面语句,逐个判定约束条件并寻找出目标函数的值,通过排序即能得出所需的结果为 $x_1 = [1, 0, 1]^T$, 目标函数为 $f(x_1) = -8$, 且可以保证此结果为全局最优解。除了全局最优解外,还可以得出其他的可行解为 $x_2 = [0, 0, 1]^T, f(x_2) = -5, x_3 = [1, 0, 0]^T, f(x_3) = -3, x_4 = [0, 0, 0]^T, f(x_4) = 0, x_5 = [0, 1, 0]^T, f(x_5) = 2$ 。

```
>> [x1,x2,x3]=meshgrid([0,1]);
i=find((x1+2*x2-x3<=2) & (x1+4*x2+x3<=4) & (x1+x2<=3) & (4*x1+x3<=6));
x1=x1(i); x2=x2(i); x3=x3(i); f=-3*x1+2*x2-5*x3; [fmin,ii]=sort(f);
index=ii(1); x=[x1(index),x2(index),x3(index)]
x1=[x1(ii),x2(ii),x3(ii)]; [x1 fmin] % 还可以列出所有的可行解
```

非线性 0-1 规划可以调用前面的 BNB20_new() 函数直接求解。用户需要首先设定上下限 x_m 和 x_M 分别为零向量和幺向量,然后再求整数规划就能得出原问题的解。

例 6-37 试用 BNB20_new() 函数求解例 6-36 给出的 0-1 线性规划问题。

解 由给出的问题,可以用匿名函数描述目标函数,然后分别设定 x_m 和 x_M 为零向量和幺向量,这样可以给出如下的语句求解 0-1 整数规划问题,得出 $x = [1, 0, 1]^T$, 结果和前面得出的完全一致。

```
>> f=@(x)[-3,2,-5]*x; x0=[1; 1; 1]; xm=[0;0;0]; xM=[1;1;1]; intlist=[1;1;1];
A=[1 2 -1; 1 4 1; 1 1 0; 0 4 1]; B=[2;4;5;6]; Aeq=[]; Beq=[];
[errmsg,fm,x]=BNB20_new(f,x0,intlist,xm,xM,A,B,Aeq,Beq)
```

事实上,分析给定的约束条件,可以发现后两个约束条件是冗余的,可以取消。

例 6-38 试求解下面的 0-1 混合规划问题^[5]。

$$\begin{aligned} \min \quad & 5y_1 + 6y_2 + 8y_3 + 10x_1 - 7x_3 - 18\ln(x_2 + 1) - 19.2\ln(x_1 - x_2 + 1) + 10 \\ \text{s.t.} \quad & \begin{cases} 0.8\ln(x_2+1)+0.96\ln(x_1-x_2+1)-0.8x_3 \leq 0 \\ \ln(x_2+1)+1.2\ln(x_1-x_2+1)-x_3-2y_3 \geq -2 \\ x_2-x_1 \leq 0 \\ x_2-2y_1 \leq 0 \\ x_1-x_2-2y_2 \leq 0 \\ y_1+y_2 \leq 1 \\ 0 \leq x \leq [2,2,1]^T, y \in \{0,1\} \end{cases} \end{aligned}$$

解 由于本问题含有非线性约束和非线性目标函数,所以用 bintprog() 函数无能为力,需要调用一般非线性混合整数规划求解函数求解。和以前介绍的内容类似,这里给出的是 x, y 两个决策向量的问题求解,而 MATLAB 现有的函数只能求解单个决策变量向量的问题,所以需要引入一组新的决策变量向量 x , 其前三个是原来的 x , 后 3 个为 $x_4 = y_1, x_5 = y_2, x_6 = y_3$, 这样,原最优化问题

可以手工改写成

$$\begin{aligned} \min \quad & 5x_4 + 6x_5 + 8x_6 + 10x_1 - 7x_3 - 18\ln(x_2 + 1) - 19.2\ln(x_1 - x_2 + 1) + 10 \\ \text{s.t.} \quad & \begin{cases} 0.8\ln(x_2+1)+0.96\ln(x_1-x_2+1)-0.8x_3 \geq 0 \\ \ln(x_2+1)+1.2\ln(x_1-x_2+1)-x_3-2x_6 \geq -2 \\ x_2-x_1 \leq 0 \\ x_2-2x_4 \leq 0 \\ x_1-x_2-2x_5 \leq 0 \\ x_4+x_5 \leq 1 \\ 0 \leq \mathbf{x} \leq [2, 2, 1, 1, 1, 1]^T \end{cases} \end{aligned}$$

可以将非线性约束用下面的 MATLAB 函数表示出来

```
function [c,ceq]=c6mmibp(x), ceq=[];
c=[-0.8*log(x(2)+1)-0.96*log(x(1)-x(2)+1)+0.8*x(3);
    -log(x(2)+1)-1.2*log(x(1)-x(2)+1)+x(3)+2*x(6)-2];
```

这样可以由结构体描述本例给出的混合 0-1 规划问题,然后调用求解程序,可以直接得出原始问题的最优解为 $\mathbf{x} = [1.301, 0, 1, 0, 1, 0]^T$, 相应的最优目标函数为 6.098。文献 [5] 给出的推荐解有误,为 $\mathbf{x} = [1.301, 0, 1, 1, 0, 1]^T$, 该解对应的目标函数为 13.0098, 显然有误。

```
>> clear P; P.intlist=[0 0 0 1 1 1]; P.x0=[0 0 0 0 0 0]';
P.objective=@(x)5*x(4)+6*x(5)+8*x(6)+10*x(1)-7*x(3) ...
    -18*log(x(2)+1)-19.2*log(x(1)-x(2)+1)+10;
P.ub=[2 2 1 1 1 1]'; P.lb=[0 0 0 0 0 0]'; P.Bineq=[0;0;0;1];
P.Aineq=[-1 1 0 0 0 0; 0 1 0 -2 0 0; 1 -1 0 0 -2 0; 0 0 0 1 1 0];
P.nonlcon=@c6mmibp; [errmsg, fm, x]=BNB20_new(P)
```

6.5 线性矩阵不等式问题求解

线性矩阵不等式 (linear matrix inequalities, LMI) 的理论与应用是近 20 年来在控制界受到较广泛关注的问题 [7]。线性矩阵不等式的概念及其在控制系统研究中的应用是由 Willems 提出的 [8], 该方法的提出可以将很多控制中的问题变换成线性规划问题的求解, 而线性规划问题的求解是很成熟的, 所以由线性矩阵不等式来求解控制问题是很有意义的。

本节将首先给出线性矩阵不等式的基本概念和常见形式, 介绍必要的变换方法, 然后介绍基于 MATLAB 中鲁棒控制工具箱和免费工具箱 YALMIP 的线性矩阵不等式求解方法。

6.5.1 线性矩阵不等式的一般描述

线性矩阵不等式的一般描述为

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + x_1\mathbf{F}_1 + \cdots + x_m\mathbf{F}_m < 0 \quad (6-5-1)$$

式中, $\mathbf{x} = [x_1, \cdots, x_m]^T$ 为多项式系数向量, 又称为决策向量。 \mathbf{F}_i 为实对称矩阵或复 Hermite 矩阵。整个矩阵不等式小于零表示 $\mathbf{F}(\mathbf{x})$ 为负定矩阵, 该不等式的解 \mathbf{x} 是凸集, 亦即

$$\mathbf{F}[\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2] = \alpha\mathbf{F}(\mathbf{x}_1) + (1-\alpha)\mathbf{F}(\mathbf{x}_2) < 0 \quad (6-5-2)$$

其中 $\alpha > 0, 1 - \alpha > 0$ 。该解又称为可行解。这样的线性矩阵不等式可以作为最优化问题的约束条件。假设有两个线性矩阵不等式 $\mathbf{F}_1(\mathbf{x}) < 0$ 和 $\mathbf{F}_2(\mathbf{x}) < 0$ ，则可以如下构造出一个单一的线性矩阵不等式

$$\begin{bmatrix} \mathbf{F}_1(\mathbf{x}) & 0 \\ 0 & \mathbf{F}_2(\mathbf{x}) \end{bmatrix} < 0 \quad (6-5-3)$$

这样两个线性矩阵不等式可以写成一个单一的线性矩阵不等式。类似地，多个线性矩阵不等式 $\mathbf{F}_i(\mathbf{x}) < 0, i = 1, 2, \dots, k$ 也可以合并成单一的线性矩阵不等式 $\mathbf{F}(\mathbf{x}) < 0$ ，其中

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} \mathbf{F}_1(\mathbf{x}) & & & \\ & \mathbf{F}_2(\mathbf{x}) & & \\ & & \ddots & \\ & & & \mathbf{F}_k(\mathbf{x}) \end{bmatrix} < 0 \quad (6-5-4)$$

6.5.2 Lyapunov 不等式

为演示一般控制问题和线性矩阵不等式之间的关系，首先考虑 Lyapunov 稳定性判定问题。对线性系统来说，若对给定的正定矩阵 \mathbf{Q} ，Lyapunov 方程

$$\mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} = -\mathbf{Q} \quad (6-5-5)$$

存在正定的解 \mathbf{X} ，则该系统是稳定的。上述问题很自然地可以表示成对下面的 Lyapunov 不等式的求解问题

$$\mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} < 0 \quad (6-5-6)$$

由于 \mathbf{X} 是对称矩阵，所以用 $n(n+1)/2$ 个元素构成的向量 \mathbf{x} 即可以描述该矩阵

$$x_i = X_{i,1}, i = 1, \dots, n, x_{n+i} = X_{i,2}, i = 2, \dots, n, \dots \quad (6-5-7)$$

该规律可以写成

$$x_{(2n-j+2)(j-1)/2+i} = X_{i,j}, j = 1, 2, \dots, n, i = j, j+1, \dots, n \quad (6-5-8)$$

则给出 \mathbf{x} 的下标即可以求出 i, j 的值。根据这样的思路可以编写出如下的 MATLAB 函数，该函数可以将 Lyapunov 方程转换为线性矩阵不等式

```
function F=lyap2lmi(A0)
if prod(size(A0))==1, n=A0;
for i=1:n, for j=1:n,
    i1=int2str(i);j1=int2str(j); eval(['syms a' i1 j1])
    eval(['A(' i1 ', ' j1 ')'=a' i1 j1, ',''])
end, end
else, n=size(A0,1); A=A0; end
vec=0; for i=1:n, vec(i+1)=vec(i)+n-i+1; end
```

```

for k=1:n*(n+1)/2,
    X=zeros(n); i=find(vec>=k); i=i(1)-1; j=i+k-vec(i)-1;
    X(i,j)=1; X(j,i)=1; F(:, :, k)=A.'*X+X*A;
end

```

该函数允许两种调用格式。若已知 A 矩阵, 由 $F = \text{lyap2lmi}(A)$, 则返回的 F 是三维数组, 其第 i 层, 即 $F(:, :, i)$ 为所需的 F_i 矩阵。若只想得出 $n \times n$ 的 A 矩阵转换出的线性矩阵不等式, 则 $F = \text{lyap2lmi}(n)$, 这时得出的 F 仍为上述定义的三维数组。在程序中, 若使 $x_i = 1$, 而其他的 x_i 的值都为 0, 则可以求出 F_i 矩阵。

例 6-39 若 $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$, 试求出其 Lyapunov 线性矩阵不等式表示。若 A 为一般 3×3 实

矩阵, 试得出相应的线性矩阵不等式。

解 输入 A 矩阵, 再给出求解语句

```
>> A=[1,2,3; 4,5,6; 7,8,0]; F=lyap2lmi(A)
```

则可以得出 F_i 矩阵分别为

$$x_1 \begin{bmatrix} 2 & 2 & 3 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 8 & 6 & 6 \\ 6 & 4 & 3 \\ 6 & 3 & 0 \end{bmatrix} + x_3 \begin{bmatrix} 14 & 8 & 1 \\ 8 & 0 & 2 \\ 1 & 2 & 6 \end{bmatrix} + x_4 \begin{bmatrix} 0 & 4 & 0 \\ 4 & 10 & 6 \\ 0 & 6 & 0 \end{bmatrix} + x_5 \begin{bmatrix} 0 & 7 & 4 \\ 7 & 16 & 5 \\ 4 & 5 & 12 \end{bmatrix} + x_6 \begin{bmatrix} 0 & 0 & 7 \\ 0 & 0 & 8 \\ 7 & 8 & 0 \end{bmatrix} < 0$$

若研究一般 3×3 矩阵, 则可以给出如下命令

```
>> F=lyap2lmi(3)
```

这时得出的线性矩阵不等式为

$$x_1 \begin{bmatrix} 2a_{11} & a_{12} & a_{13} \\ a_{12} & 0 & 0 \\ a_{13} & 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 2a_{21} & a_{22}+a_{11} & a_{23} \\ a_{22}+a_{11} & 2a_{12} & a_{13} \\ a_{23} & a_{13} & 0 \end{bmatrix} + x_3 \begin{bmatrix} 2a_{31} & a_{32} & a_{33}+a_{11} \\ a_{32} & 0 & a_{12} \\ a_{33}+a_{11} & a_{12} & 2a_{13} \end{bmatrix} \\ + x_4 \begin{bmatrix} 0 & a_{21} & 0 \\ a_{21} & 2a_{22} & a_{23} \\ 0 & a_{23} & 0 \end{bmatrix} + x_5 \begin{bmatrix} 0 & a_{31} & a_{21} \\ a_{31} & 2a_{32} & a_{33}+a_{22} \\ a_{21} & a_{33}+a_{22} & 2a_{23} \end{bmatrix} + x_6 \begin{bmatrix} 0 & 0 & a_{31} \\ 0 & 0 & a_{32} \\ a_{31} & a_{32} & 2a_{33} \end{bmatrix} < 0$$

某些非线性的不等式也可以通过变换转换成线性矩阵不等式。其中, 分块矩阵不等式的 Schur 补性质^[9]是进行这样变换的常用方法。该性质的内容是: 若某个仿射函数矩阵 $F(x)$ 可以分块表示成

$$F(x) = \begin{bmatrix} F_{11}(x) & F_{12}(x) \\ F_{21}(x) & F_{22}(x) \end{bmatrix} \quad (6-5-9)$$

其中 $F_{11}(x)$ 是方阵, 则下面三个矩阵不等式是等价的

$$F(x) < 0 \quad (6-5-10)$$

$$F_{11}(x) < 0, \quad F_{22}(x) - F_{21}(x)F_{11}^{-1}(x)F_{12}(x) < 0 \quad (6-5-11)$$

$$F_{22}(x) < 0, \quad F_{11}(x) - F_{12}(x)F_{22}^{-1}(x)F_{21}(x) < 0 \quad (6-5-12)$$

例如,对一般代数 Riccati 方程稍加变换,则可以得出 Riccati 不等式

$$\mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} + (\mathbf{X} \mathbf{B} - \mathbf{C}) \mathbf{R}^{-1} (\mathbf{X} \mathbf{B} - \mathbf{C}^T)^T < 0 \quad (6-5-13)$$

式中 $\mathbf{R} = \mathbf{R}^T > 0$ 。显然,该不等式因为含有二次项,所以它本身不是线性矩阵不等式。由 Schur 补性质可以看出,原非线性不等式可以等价地变换成

$$\mathbf{X} > 0, \begin{bmatrix} \mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} & \mathbf{X} \mathbf{B} - \mathbf{C}^T \\ \mathbf{B}^T \mathbf{X} - \mathbf{C} & -\mathbf{R} \end{bmatrix} < 0 \quad (6-5-14)$$

6.5.3 线性矩阵不等式问题分类

线性矩阵不等式问题通常可以分为三类问题:可行解问题、线性目标函数最优化问题与广义特征值最优化问题:

(1) **可行解问题**。所谓可行解问题就是最优化问题中的约束条件求解问题

$$\mathbf{F}(\mathbf{x}) < 0 \quad (6-5-15)$$

得出满足该不等式一个解的问题。求解线性矩阵不等式可行解等价于求解 $\mathbf{F}(\mathbf{x}) < \sigma \mathbf{I}$, 其中 σ 是能够用数值方法找到的最小值。如果找到的 $\sigma < 0$, 则得出的解是原问题的可行解, 否则会提示无法找到可行解。

(2) **线性目标函数最优化问题**。考虑下面的最优化问题

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{F}(\mathbf{x}) < 0 \end{aligned} \quad (6-5-16)$$

由于约束条件是由线性矩阵不等式表示的, 而目标函数也可以由决策变量 \mathbf{x} 构造的线性矩阵表示, 所以这样的问题就是普通的线性规划求解问题。

(3) **广义特征值最优化问题**。广义特征值问题是线性矩阵不等式理论的一类最一般的问题。回顾第 3 章介绍的广义特征值问题, $\mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$, 由该式演化可以得到更一般的不等式 $\mathbf{A}(\mathbf{x}) < \lambda \mathbf{B}(\mathbf{x})$, 可将 λ 看作矩阵的广义特征值, 从而归纳出下面的最优化问题

$$\begin{aligned} \min \quad & \lambda \\ \lambda, \mathbf{x} \text{ s.t.} \quad & \begin{cases} \mathbf{A}(\mathbf{x}) < \lambda \mathbf{B}(\mathbf{x}) \\ \mathbf{B}(\mathbf{x}) > 0 \\ \mathbf{C}(\mathbf{x}) < 0 \end{cases} \end{aligned} \quad (6-5-17)$$

另外还可以有其他约束, 归类成 $\mathbf{C}(\mathbf{x}) < 0$ 。在这样约束条件求取最小的广义特征值的问题可以由一类特殊的线性矩阵不等式来表示。事实上, 若将这几个约束归并成单一的线性矩阵不等式, 则这样的最优化问题和线性目标函数最优化问题是同样的问题。

6.5.4 线性矩阵不等式问题的 MATLAB 求解

早期的 MATLAB 中提供了线性矩阵不等式工具箱,可以直接求解相应的问题。新版本的 MATLAB 中将该工具箱并入了鲁棒控制工具箱,调用该工具箱中的函数可以求解线性矩阵不等式的各种问题。

描述线性矩阵不等式的方法是较烦琐的,用鲁棒控制工具箱中相应的函数描述这样的问题也是比较烦琐的。这里将介绍相关 MATLAB 语句的调用方法,并将给出例子演示相关函数的使用方法。

描述线性矩阵不等式应该有几个步骤:

(1) **创建 LMI 模型**。若想描述一个含有若干的 LMI 的整体线性矩阵不等式问题,需要首先调用 `setlmis([])` 函数来建立 LMI 框架,这样将建立一个 LMI 模型框架。

(2) **定义需要求解的变量**。未知矩阵变量可以由 `lmivar()` 函数申明,该函数的调用格式为 `P=lmivar(key,[n1,n2])`,其中 `key` 是未知矩阵类型的标记,若 `key` 的值为 2,则变量 `P` 表示为 $n_1 \times n_2$ 的一般矩阵。若 `key` 为 1,则 `P` 矩阵为 $n_1 \times n_1$ 的对称矩阵。若 `key` 为 1,且 n_1 和 n_2 为向量,则 `P` 为块对角对称矩阵。若 `key` 取值 3,则表示 `P` 为特殊类型的矩阵。

(3) **描述分块形式给出线性矩阵不等式**。申明了需求解的变量名后,可以由 `lmiterm()` 函数来描述各个 LMI 式子,该函数的调用格式为 `lmiterm([k,i,j,P],A,B,flag)`,其中 `k` 为 LMI 编号,一个线性矩阵不等式问题可以由若干个 LMI 构成,用这样的方法可以分别描述各个 LMI。`k` 取负值时表示不等号 $<$ 右侧的项。一个 LMI 子项可以由多个 `lmiterm()` 函数来描述。若第 `k` 个 LMI 是以分块形式给出的,则 `i,j` 表示该分块所在的行号和列号。`P` 为已经由 `lmivar()` 函数申明过的变量名。`A,B` 矩阵表示该项中变量 `P` 左乘和右乘的矩阵,即该项含有 APB 。`A` 和 `B` 设置成 1 和 -1 则分别表示单位矩阵 I 或负单位阵 $-I$ 。若 `flag` 选择为 's',则该项表示对称项 $APB + (APB)^T$ 。如果该项为常数矩阵,则可以将相应的 `P` 设置为 0,同时略去 `B` 矩阵。

(4) **完成 LMI 模型描述**。由 `lmiterm()` 函数定义所有的 LMI 后,就可以用 `getlmis()` 函数来确定 LMI 问题的描述,该函数的调用格式为 `G=getlmis`。

(5) **求解 LMI 问题**。定义 `G` 模型后,就可以根据问题的类型调用相应函数直接求解

```
[tmin,x] = feasp(G,options,target)           % 可行解问题
[copt,x] = mincx(G,c,options,x0,target)      % 线性目标函数问题
[λ,x] = gevp(G,nlfc,options,λ0,x0,target) % 广义特征值问题
```

这样获得的解 x 是一个向量,可以调用 `dec2mat()` 函数将所需的解矩阵提取出来。控制选项 `options` 是由 5 个值构成的向量,其第一个量表示要求的求解精度,通常可以取为 10^{-5} 或其他数值。

例 6-40 考虑 Riccati 不等式 $A^T X + X A + X B R^{-1} B^T X + Q < 0$, 其中

$$A = \begin{bmatrix} -2 & -2 & -1 \\ -3 & -1 & -1 \\ 1 & 0 & -4 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}, \quad Q = \begin{bmatrix} -2 & 1 & -2 \\ 1 & -2 & -4 \\ -2 & -4 & -2 \end{bmatrix}, \quad R = I_2$$

试求出该不等式的一个正定可行解 X 。

解 该不等式显然不是线性矩阵不等式,类似前面介绍的 Riccati 不等式,可以引用 Schur 补性质对其进行变换,得出分块的 LMI 表示为 $\begin{bmatrix} \mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} + \mathbf{Q} & \mathbf{X} \mathbf{B} \\ \mathbf{B}^T \mathbf{X} & -\mathbf{R} \end{bmatrix} < 0$ 。考虑到需要求出原不等式的正定解 \mathbf{X} ,故除了上面变换后的 Riccati 不等式外还需要满足 $\mathbf{X} > 0$ 。可以将 Riccati 不等式设置成不等式 1,正定不等式设置成不等式 2,这样使用 `lmiterm()` 函数时,只需将 k 设置成 1 和 2 即可。另外,根据 \mathbf{A} 和 \mathbf{B} 矩阵的维数,可以假定 \mathbf{X} 为 3×3 对称矩阵。这样就可以用下面几个语句建立并求解可行解问题。因为第 2 不等式为 $\mathbf{X} > 0$,所以序号采用 -2。

```
>> A=[-2,-2,-1; -3,-1,-1; 1,0,-4]; B=[-1,0; 0,-1; -1,-1];
    Q=[-2,1,-2; 1,-2,-4; -2,-4,-2]; R=eye(2);
    setlmis([]); % 建立空白的 LMI 框架
    X=lmivar(1,[3 1]); % 申明需要求解的矩阵 X 为 3×3 对称矩阵
    lmiterm([1 1 1 X],A',1,'s') % (1,1) 分块,对称表示为  $\mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A}$ 
    lmiterm([1 1 1 0],Q) % (1,1) 分块后面补一个 Q 常数矩阵
    lmiterm([1 1 2 X],1,B) % (1,2) 分块,填写 XB
    lmiterm([1 2 2 0],-1) % (2,2) 分块,填写 -R
    lmiterm([-2,1,1,X],1,1) % 设置第 2 不等式,即不等式  $\mathbf{X} > 0$ 
    G=getlmis; % 完成 LMI 框架的设置
    [tmin b]=feasp(G); X=dec2mat(G,b,X) % 求解可行解并提取解矩阵
```

这样可以得出 $t_{\min} = -0.2427$,原问题的可行解为

$$\mathbf{X} = \begin{bmatrix} 1.0329 & 0.4647 & -0.23583 \\ 0.4647 & 0.77896 & -0.050684 \\ -0.23583 & -0.050684 & 1.4336 \end{bmatrix}$$

值得指出的是,可能是由于该工具箱本身的问题,如果在描述 LMI 时给出了对称项,如 `lmiterm([1 2 1 X],B',1)`,则该函数将得出错误的结果。所以在求解线性矩阵不等式问题时一定不能给出对称项。

6.5.5 基于 YALMIP 工具箱的最优化求解方法

Johan Löfberg 博士开发了一个基于符号运算工具箱编写的模型优化工具箱 YALMIP (yet another LMI package) [10],该工具箱提供的线性矩阵不等式求解方法和鲁棒控制工具箱中的 LMI 函数相比要直观得多。该工具箱的演示程序中还介绍了其他相关的最优化问题求解方法 [11]。

YALMIP 工具箱提供了简单的决策变量表示方法,可以调用 `sdpvar()` 函数来表示,该函数的调用方法为

```
 $\mathbf{X} = \text{sdpvar}(n)$  % 对称方阵的表示方法
 $\mathbf{X} = \text{sdpvar}(n,m)$  % 长方型一般矩阵的表示方法
 $\mathbf{X} = \text{sdpvar}(n,n,'full')$  % 一般方阵的表示方法
```

这样定义的矩阵还可以进一步利用,例如,这样定义的向量还可以和 `hankel()` 联合使用,

构造出 Hankel 矩阵。类似地,由 `intvar()` 和 `binvar()` 函数还可以定义整型变量和二进制变量,从而求解整数规划和 0-1 规划问题。

由该工具箱针对 `sdpvar` 型变量定义的 `set()` 函数还可以描述矩阵不等式。如果有若干个这样的矩阵不等式,可以用 `+` 号将联立的若干个不等式“加”起来。

当然使用类似的方法还可以定义目标函数,描述了矩阵不等式约束后就可以分别如下调用 `solvesdp()` 函数直接求解各类问题

```
s = solvesdp(F)           % 求解可行解问题
s = solvesdp(F,f)         % 求解一般最优化问题,其中 f 为目标函数
s = solvesdp(F,f,options) % 允许设定选项,如算法选择
```

其中, F 为 LMI 表示。求解结束后,可以由 `X = double(X)` 语句提取得出的解矩阵。

例 6-41 利用 YALMIP 工具箱,例 6-40 中的问题可以由下面语句更简洁地求解相应的矩阵不等式问题,这里的结果和前面得出的完全一致。

```
>> A=[-2,-2,-1; -3,-1,-1; 1,0,-4]; B=[-1,0; 0,-1; -1,-1];
    Q=[-2,1,-2; 1,-2,-4; -2,-4,-2]; R=eye(2); X=sdpvar(3);
    F=set([A'*X+X*A+Q, X*B; B'*X, -R]<0)+set(X>0);
    sol=solvesdp(F); X=double(X)
```

例 6-42 试用 YALMIP 工具箱求解例 6-21 中给出的线性规划问题。

解 为方便起见,将该问题重新表述如下

$$\begin{aligned} \min \quad & -2x_1 - x_2 - 4x_3 - 3x_4 - x_5 \\ \text{s.t.} \quad & \begin{cases} 2x_2 + x_3 + 4x_4 + 2x_5 \leq 54 \\ 3x_1 + 4x_2 + 5x_3 - x_4 - x_5 \leq 62 \\ x_1, x_2 \geq 0, x_3 \geq 3.32, x_4 \geq 0.678, x_5 \geq 2.57 \end{cases} \end{aligned}$$

显然, x 是一个 5×1 列向量,这样可以由下面语句求解原问题

```
>> x=sdpvar(5,1); A=[0 2 1 4 2; 3 4 5 -1 -1]; B=[54; 62];
    xm=[0,0,3.32,0.678,2.57]'; F=set(A*x<=B)+set(x>=xm);
    sol=solvesdp(F,-[2 1 4 3 1]*x); x=double(x)
```

由该函数可以立即得出问题的解 $x = [19.785, 0, 3.32, 11.385, 2.57]^T$, 与前面得出的完全一致。如果将决策变量设置为整数,可以用 `intvar()` 定义,并用下面语句求解整数规划

```
>> x=intvar(5,1); F=set(A*x<=B)+set(x>=xm);
    options=sdpsettings('solver','bnb'); % 选择分枝定界法
    sol=solvesdp(F,-[2 1 4 3 1]*x,options); double(x)
```

其解为 $x = [19, 0, 4, 10, 5]^T$, 与例 6-30 中得出的完全一致。

例 6-43 对线性系统 (A, B, C, D) 来说,其 H_∞ 范数可以由控制系统工具箱中的 `norm()` 函数直接求解。采用 LMI 方法也可以求解系统的 H_∞ 范数,其数学描述为

$$\min_{\gamma, P} \gamma \quad (6-5-18)$$

$$\text{s.t.} \begin{cases} \begin{bmatrix} A^T P + P A & P B & C^T \\ B^T P & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} < 0 \\ P > 0 \end{cases}$$

试求解下面给出的线性系统模型的 \mathcal{H}_∞ 范数。

$$A = \begin{bmatrix} -4 & -3 & 0 & -1 \\ -3 & -7 & 0 & -3 \\ 0 & 0 & -13 & -1 \\ -1 & -3 & -1 & -10 \end{bmatrix}, B = \begin{bmatrix} 0 \\ -4 \\ 2 \\ 5 \end{bmatrix}, C = [0, 0, 4, 0], D = 0$$

解 通过下面的语句,可以利用 YALMIP 工具箱描述 \mathcal{H}_∞ 范数问题,从而得出其值为 0.4640,该结果和 norm() 函数的结果完全一致。

```
>> A=[-4,-3,0,-1; -3,-7,0,-3; 0,0,-13,-1; -1,-3,-1,-10];
    B=[0; -4; 2; 5]; C=[0,0,4,0]; D=0; gam=sdpvar(1); P=sdpvar(4);
    F=set([A*P+P*A',P*B,C'; B'*P,-gam,D'; C,D,-gam]<0)+set(P>0);
    sol=solvesdp(F,gam); double(gam), norm(ss(A,B,C,D),'inf')
```

6.6 多目标优化问题求解

前面所有的最优化问题都假设目标函数 $f(x)$ 为标量函数,所以前面介绍的最优化问题称为单目标规划问题。对这一假设进行突破,可以将目标函数扩展为向量函数,这时,最优化问题将称为多目标优化问题。本节将简要介绍多目标最优化问题的建模与求解方法。

6.6.1 多目标优化模型

多目标最优化问题的一般表示为

$$J = \min_{x \text{ s.t. } G(x) \leq 0} F(x) \quad (6-6-1)$$

其中 $F(x) = [f_1(x), f_2(x), \dots, f_p(x)]^T$ 。下面将通过例子演示多目标规划问题的建模问题,揭示多目标优化的物理意义。

例 6-44 设某商店有 A_1, A_2, A_3 三种糖果,单价分别为 4, 2.8 和 2.4 元/kg,现在要筹办一次茶话会,要求买糖果的钱不超过 20 元,糖果总量不得少于 6kg, A_1 和 A_2 两种糖果总量不得少于 3kg,应该如何确定最好的买糖方案?(问题来源:文献 [12])

解 首先应该决定目标函数如何选择的问题。在本例中,好的方案意味着少花钱多办事,这应该对应于两个目标函数,一个是花钱最少,另一个是买糖果的总量最重。其余的条件可以认为是约束条件。当然,这两个目标函数多少有些矛盾。下面考虑如何将这样的问题用数学表示。

假设 A_1, A_2, A_3 三种糖果的购买量分别为 x_1, x_2 和 x_3 kg,这时两个目标函数分别为

$$\text{花钱: } f_1(x) = 4x_1 + 2.8x_2 + 2.4x_3 \rightarrow \min, \text{ 糖果总量: } f_2(x) = x_1 + x_2 + x_3 \rightarrow \max$$

如果统一用最小值的方式表示,则有约束的多目标优化问题可以表示成

$$\min_{\mathbf{x} \text{ s.t. } \begin{cases} 4x_1 + 2.8x_2 + 2.4x_3 \leq 20 \\ x_1 + x_2 + x_3 \geq 6 \\ x_1 + x_2 \geq 3 \\ x_1, x_2, x_3 \geq 0 \end{cases}} \begin{bmatrix} 4x_1 + 2.8x_2 + 2.4x_3 \\ -(x_1 + x_2 + x_3) \end{bmatrix}$$

模型建立起来以后,可以考虑采用后面介绍的方法求解。

6.6.2 无约束多目标函数的最小二乘求解

假设多目标规划问题目标函数 $\mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})]^T$, 则可以按照下面的方式将其转换成单目标问题

$$\min_{\mathbf{x} \text{ s.t. } \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M} f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) + \dots + f_k^2(\mathbf{x}) \quad (6-6-2)$$

这样,就可以用以前介绍的方法直接求解该问题了。MATLAB 还提供了 `lsqnonlin()` 函数直接求解这类问题,该函数的调用格式为

$$[\mathbf{x}, n_f, \mathbf{f}_{\text{opt}}, \text{flag}, \mathbf{c}] = \text{lsqnonlin}(\mathbf{F}, \mathbf{x}_0, \mathbf{x}_m, \mathbf{x}_M)$$

其中, \mathbf{F} 为给目标函数写的 M-函数或匿名函数,该函数为向量函数。 \mathbf{x}_0 为初始搜索点。最优化运算完成后,结果将在变量 \mathbf{x} 中返回,最优化的目标函数向量将在 \mathbf{f}_{opt} 变量中返回,其范数由 n_f 返回。和其他优化函数一样,选项 `OPT` 有时是很重要的。

例 6-45 试求解下面无约束非线性多目标规划问题的最小二乘解

$$\min_{\mathbf{x} \text{ s.t. } \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leq \mathbf{x} \leq \begin{bmatrix} 3 \\ \pi \\ 5 \end{bmatrix}} \begin{bmatrix} (x_1 + 2x_2 + 3x_3) \sin(x_1 + x_2) e^{-x_1^2 - x_3^2} + 5x_3 \\ e^{-x_2^2 - 4x_2^3} \cos(4x_1 + x_2) \end{bmatrix}$$

解 写出向量型的目标函数,则可以调用下面语句直接求解原问题,得出 $\mathbf{x} = [2.9998, 3.1415, 0]$ 。

```
>> f=@(x)[(x(1)+2*x(2)+3*x(3))*sin(x(1)+x(2))*exp(-x(1)^2-x(3)^2)+5*x(3);
exp(-x(2)^2-4*x(2)^3)*cos(4*x(1)+x(2))];
xm=[0; 0; 0]; xM=[3; pi; 5]; x0=xM; x=lsqnonlin(f,x0,xm,xM)
```

事实上,如果采用前面介绍的 `fmincon()` 函数,则可以重新定义目标函数,调用该函数求解可以直接得出所需结果 $\mathbf{x} = [3, 3.1416, 0]$ 。值得指出的是,用后者可以求取含有约束的多目标规划最小二乘问题。当然,有约束问题也可以利用单目标问题直接求解。

```
>> G=@(x)f(x)'*f(x); x=fmincon(G,x0,[],[],[],[],xm,xM)
```

6.6.3 多目标问题转换为单目标问题求解

前面各节已经很全面地介绍了单目标问题的数值求解方法,事实上,多目标问题可以按照某种方法转换成特定的单目标问题,例如对多目标函数进行加权或最小二乘处理等。本小节侧重介绍这类方法。

1. 线性加权变换及求解

显然,前面介绍的单目标优化算法不能直接用于求解上述多目标优化问题,在求解之前,需要引入某种方法将其变换成单目标优化问题。最简单的变换方法是根据对两个指标的侧重情况引入加权,使得目标函数改写成标量形式

$$f(\boldsymbol{x}) = w_1f_1(\boldsymbol{x}) + w_2f_2(\boldsymbol{x}) + \cdots + w_pf_p(\boldsymbol{x}) \tag{6-6-3}$$

其中 $w_1 + w_2 + \cdots + w_p = 1$, 且 $0 \leq w_1, w_2, \cdots, w_p \leq 1$ 。

例 6-46 试在不同的加权系数下, 求出例 6-44 中问题的解。

解 原问题可以重新修改成下面的线性规划系数

$$\begin{aligned} \min \quad & (w_1[4, 2.8, 2.4] - w_2[1, 1, 1])\boldsymbol{x} \\ \boldsymbol{x} \text{ s.t.} \quad & \begin{cases} 4x_1 + 2.8x_2 + 2.4x_3 \leq 20 \\ -x_1 - x_2 - x_3 \leq -6 \\ -x_1 - x_2 \leq -3 \\ x_1, x_2, x_3 \geq 0 \end{cases} \end{aligned}$$

这样,不同加权系数下的最优购买方案可以由下面循环结构得出,如表 6-3 所示。可见,由于加权系数选择不同,得出的最优解也是不同的。另外, $x_1 \equiv 0$, 这是因为,对 x_1 本身没有约束,所以其值当然是越小越好了。

```
>> f1=[4,2.8,2.4]; f2=[-1,-1,-1]; Aeq=[]; Beq=[]; xm=[0;0;0]; C=[];
A=[4 2.8 2.4; -1 -1 -1; -1 -1 0]; B=[20;-6;-3]; ww1=[0:0.1:1];
for w1=ww1, w2=1-w1;
    x=linprog(w1*f1+w2*f2,A,B,Aeq,Beq,xm); C=[C; w1 w2 x' f1*x -f2*x]
end
```

表 6-3 不同加权系数下的最优方案

w_1	w_2	x_1	x_2	x_3	总花费	糖果总量	w_1	w_2	x_1	x_2	x_3	总花费	糖果总量
0	1	0	3	4.8333	20	7.8333	0.6	0.4	0	3	3	15.6	6
0.1	0.9	0	3	4.8333	20	7.8333	0.7	0.3	0	3	3	15.6	6
0.2	0.8	0	3	4.8333	20	7.8333	0.8	0.2	0	3	3	15.6	6
0.3	0.7	0	3	3	15.6	6	0.9	0.1	0	3	3	15.6	6
0.4	0.6	0	3	3	15.6	6	1	0	0	3	3	15.6	6
0.5	0.5	0	3	3	15.6	6							

2. 线性规划问题的最佳妥协解

考虑一类特殊的线性规划问题

$$\begin{aligned} J = \max \quad & C\boldsymbol{x} \\ \boldsymbol{x} \text{ s.t.} \quad & \begin{cases} A\boldsymbol{x} \leq B \\ A_{eq}\boldsymbol{x} = B_{eq} \\ \boldsymbol{x}_m \leq \boldsymbol{x} \leq \boldsymbol{x}_M \end{cases} \end{aligned} \tag{6-6-4}$$

和传统线性规划问题不同,这里的目标函数不是一个向量,而是一个矩阵。每一个目标函数 $f_i(\mathbf{x}) = \mathbf{c}_i \mathbf{x}, i = 1, 2, \dots, p$, 可以理解成第 i 方的利益分配,所以这样的最优化问题可以认为是各方利益的最大分配。当然,在约束条件的限制和相互制约下,不可能每一方的利益均能真正地最大化,这就需要各方作出适当的妥协,得出唯一的最佳妥协解。最佳妥协解的求解步骤如下:

- (1) 单独求解每个单目标函数的最优化问题,得出最优解 $f_k, k = 1, 2, \dots, p$ 。
- (2) 通过归范化构造单独的目标函数

$$f(\mathbf{x}) = -\frac{1}{f_1} \mathbf{c}_1 \mathbf{x} - \frac{1}{f_2} \mathbf{c}_2 \mathbf{x} - \dots - \frac{1}{f_p} \mathbf{c}_p \mathbf{x} \quad (6-6-5)$$

- (3) 最佳妥协解可以变换成下面的单目标线性规划问题直接求解

$$J = \min \quad f(\mathbf{x}) \quad (6-6-6)$$

$$\mathbf{x} \text{ s.t. } \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{B} \\ \mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{B}_{\text{eq}} \\ \mathbf{x}_{\text{m}} \leq \mathbf{x} \leq \mathbf{x}_{\text{M}} \end{cases}$$

根据上述算法,可以编写出一个最佳妥协解的求解程序。注意,该函数求解的是最大值问题的妥协解。

```
function [x,f,flag,cc]=linprog_c(C,A,B,Aeq,Beq,xm,xM)
[p,m]=size(C); c=0;
for i=1:p, [x,f]=linprog(C(i,:),A,B,Aeq,Beq,xm,xM); c=c-C(i,:)/f; end
[x,f,flag,cc]=linprog(c,A,B,Aeq,Beq,xm,xM);
```

例 6-47 试求出例 6-44 中问题的最佳妥协解。

解 由下面的语句可以立即得出最佳妥协解为 $\mathbf{x} = [0, 3, 4.8333]^T$, 总花费 20 元, 购买糖果的总量为 7.8333 kg。

```
>> C=[-4 -2.8 -2.4; 1 1 1]; A=[4 2.8 2.4; -1 -1 -1; -1 -1 0];
    B=[20; -6; -3]; Aeq=[]; Beq=[]; xm=[0;0;0]; xM=[];
    x=linprog_c(C,A,B,Aeq,Beq,xm,xM), C*x
```

例 6-48 试求出下面多目标线性规划问题的最佳妥协解

$$\min \quad \begin{bmatrix} 3x_1 + x_2 + 6x_4 \\ 10x_2 + 7x_4 \\ 2x_1 + x_2 + 8x_3 \\ x_1 + x_2 + 3x_3 + 2x_4 \end{bmatrix}$$

$$\mathbf{x} \text{ s.t. } \begin{cases} 2x_1 + 4x_2 + x_4 \leq 110 \\ 5x_3 + 3x_4 \geq 180 \\ x_1 + 2x_2 + 6x_3 + 5x_4 \leq 250 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

解 由给出的多目标线性规划问题,可以容易地写出 \mathbf{C} 矩阵,并写出其他的约束条件,这样调用前面编写的 `linprog_c()` 函数可以直接求得最佳妥协解为 $\mathbf{x} = [0, 26.087, 32.6087, 5.6522]^T$, 各方妥协的目标函数为 $[60, 300.4348, 286.9565, 135.2174]^T$ 。

```
>> C=-[3,1,0,6; 0,10,0,7; 2,1,8,0; 1,1,3,2];
A=[2,4,0,1; 0,0,-5,-3; 1,1,6,5]; B=[110; -180; 250];
Aeq=[]; Beq=[]; xm=[0;0;0;0]; xM=[];
x=linprog_c(C,A,B,Aeq,Beq,xm,xM), -C*x
```

3. 线性规划问题的最小二乘解

考虑下面多目标线性规划问题的最小二乘表示

$$\begin{aligned} \min \quad & \frac{1}{2} \|Cx - d\|^2 \\ \text{s.t.} \quad & \begin{cases} Ax \leq B \\ A_{eq}x = B_{eq} \\ x_m \leq x \leq x_M \end{cases} \end{aligned} \quad (6-6-7)$$

则最小二乘解可以由 $x = \text{lsqlin}(C, d, A, B, A_{eq}, B_{eq}, x_m, x_M, x_0, options)$ 函数直接得出。因为原问题为凸优化问题,初值的 x_0 的选择不是很重要,可以忽略。

例 6-49 考虑例 6-48 中给出的多目标线性规划问题,试求其最小二乘解。

解 由给出的多目标线性规划问题,可以容易地写出 C 矩阵,并写出其他的约束条件,这样调用 `lsqlin()` 函数可以直接求得最小二乘解为 $x = [0, 0, 28.4456, 12.5907]^T$,各个目标函数为 $[75.544, 88.1347, 227.5648, 110.5181]^T$ 。注意,同样的问题,由于求解方法或目标函数选择不同,得出的最优解可能也不同。对本例来说,最优妥协解显然不同于最小二乘解。

```
>> C=[3,1,0,6; 0,10,0,7; 2,1,8,0; 1,1,3,2]; d=zeros(4,1);
A=[2,4,0,1; 0,0,-5,-3; 1,1,6,5]; B=[110; -180; 250]; Aeq=[]; Beq=[];
xm=[0;0;0;0]; xM=[]; x=lsqlin(C,d,A,B,Aeq,Beq,xm,xM), C*x
```

6.6.4 多目标优化问题的 Pareto 解集

从前面的分析看,一般情况下多目标优化问题的解是不唯一的,其解可能随着决策者的偏好而不同。现在重新考虑原始多目标优化问题,假设某一个目标函数分量取一系列离散点,则原来问题的目标函数的个数将减少 1,这样可能给原问题的研究带来新的结果。下面将通过例子演示这样的分析方法。

例 6-50 采用上述的离散点分析方法重新研究例 6-44 中的多目标优化问题。

解 对原始问题中花费的钱数取一系列离散点 $m_i \in (15, 20)$,则原始问题可以改写成单目标的线性规划问题

$$\begin{aligned} \min \quad & -[1, 1, 1]x \\ \text{s.t.} \quad & \begin{cases} 4x_1 + 2.8x_2 + 2.4x_3 = m_i \\ -x_1 - x_2 - x_3 \leq -6 \\ -x_1 - x_2 \leq -3 \\ x_1, x_2, x_3 \geq 0 \end{cases} \end{aligned}$$

则该问题的含义是,若花费为 m_i ,在满足约束条件的前提下最多可以买多少糖果。显然,用这样的方法可以得出最多糖果量 n_i 的值。对不同的 m_i 可以得出不同的 n_i ,它们之间的关系曲线如图 6-11 所示,然而这样得出的曲线上的点并非全是原问题的解,因为没有考虑 m_i 的优化问题。

```
>> f2=[-1,-1,-1]; Aeq=[4 2.8 2.4]; xm=[0;0;0];
A=[-1 -1 -1; -1 -1 0]; B=[-6;-3]; mi=15:0.1:20; ni=[];
for m=mi, Beq=m; x=linprog(f2,A,B,Aeq,Beq,xm); ni=[ni,-f2*x]; end
plot(mi,ni)
```

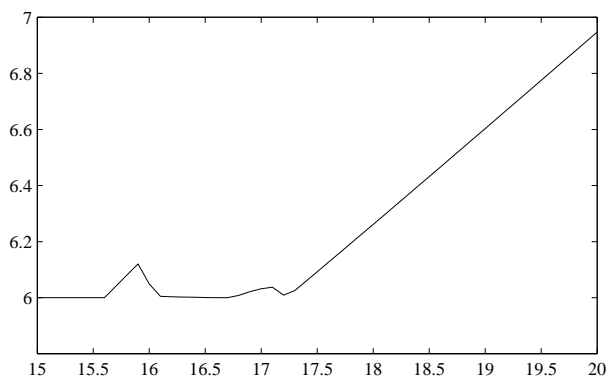


图 6-11 多目标优化问题两个目标函数的关系曲线

考虑一个双目标函数的问题,可以首先得出可行解的离散点,将这些点先在二维平面上显示出来,如图 6-12 所示。因为原始问题是求取两个坐标系 f_1 和 f_2 的最小值,所以可以从得出的可行解离散点提取出区域左下角的一条曲线,这个曲线上的点都是原问题的解,称为 Pareto 解集 (Pareto set 或 Pareto front)。Yi Cao 在 Gianluca Dorini 贡献的 Pareto 解集提取程序的基础上,开发了改进的快速提取程序^[13],主函数 `paretofront()` 的调用格式为 $K = \text{paretofront}([f_1, f_2, \dots, f_p])$,其中 f_1, f_2, \dots, f_p 为可行解离散点构成的列向量, K 向量为标志向量,指示可行解离散点是否为 Pareto 解集中的点。

例 6-51 试提取例 6-44 中的 Pareto 解集。

解 类似于前面介绍的穷举方法,首先生成一组 x_1, x_2, x_3 网格数据,将不满足约束条件的点剔除掉,留下可行解,然后调用 `paretofront()` 函数提取并绘制 Pareto 解集,如图 6-13 所示。

```
>> [x1,x2,x3]=meshgrid(0:0.1:4);
ii=find(4*x1+2.8*x2+2.4*x3<=20&x1+x2+x3>=6&x1+x2>=3);
xx1=x1(ii); xx2=x2(ii); xx3=x3(ii); f1=4*xx1+2.8*xx2+2.4*xx3;
f2=-(xx1+xx2+xx3); k=paretofront([f1 f2]);
plot(f1,f2,'x'), hold on; plot(f1(k),f2(k),'o')
```

6.6.5 极小极大问题求解

多目标优化的一类很重要的问题是极小极大问题。假设有某一组 p 个目标函数 $f_i(\mathbf{x})$, $i = 1, 2, \dots, p$, 它们中的每一个均可以提取出一个最大值 $\max_{\mathbf{x} \text{ s.t. } \mathbf{G}(\mathbf{x}) \leq 0} f_i(\mathbf{x})$, 而这样得出的

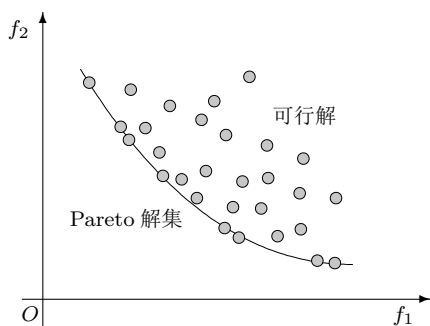


图 6-12 Pareto 解集示意图

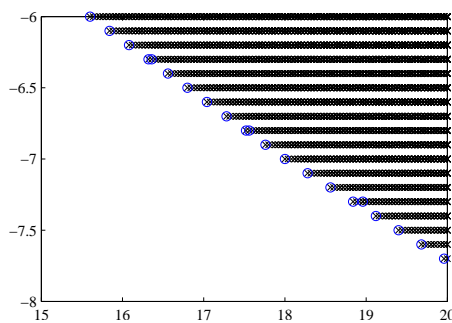


图 6-13 例 6-44 的 Pareto 解集

一组最大值仍然是 \mathbf{x} 的函数。现在想对这些最大值进行最小化搜索,即

$$J = \min \left[\max_{\mathbf{x} \text{ s.t. } \mathbf{G}(\mathbf{x}) \leq 0} f_i(\mathbf{x}) \right] \quad (6-6-8)$$

则这类问题称为极小极大问题。换句话说,极小极大问题是在最不利的条件下寻找最有决策方案的一种方法。

考虑各类约束条件,极小极大问题可以更一般地改写成

$$J = \min \max_{\mathbf{x} \text{ s.t. } \begin{cases} \mathbf{Ax} \leq \mathbf{B} \\ \mathbf{A}_{\text{eq}} \mathbf{x} = \mathbf{B}_{\text{eq}} \\ \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \\ \mathbf{C}(\mathbf{x}) \leq 0 \\ \mathbf{C}_{\text{eq}}(\mathbf{x}) = 0 \end{cases}} f_i(\mathbf{x}) \quad (6-6-9)$$

MATLAB 最优化工具箱提供了 `fminimax()` 函数,可以直接求解极小极大问题。该函数的调用格式为

$$[\mathbf{x}, f_{\text{opt}}, \text{flag}, \mathbf{c}] = \text{fminimax}(\mathbf{F}, \mathbf{x}_0, \mathbf{A}, \mathbf{B}, \mathbf{A}_{\text{eq}}, \mathbf{B}_{\text{eq}}, \mathbf{x}_m, \mathbf{x}_M, \mathbf{CF}, \text{OPT}, p_1, p_2, \dots)$$

该函数的调用格式接近于前面介绍的 `fmincon()` 函数,不同的是,目标函数为向量形式描述,当然,用匿名函数和 M-函数均可以表示新目标函数。

例 6-52 试求解下面的极小极大问题

$$\min \max_{\mathbf{x} \text{ s.t. } \begin{cases} 4.3x_1 + 3.8x_2 \leq 4.9 \\ x_1 + x_2 \leq 3 \end{cases}} \begin{bmatrix} x_1^2 \sin x_2 + x_2 - 3x_1 x_2 \cos x_1 \\ -x_1^2 e^{-x_2} - x_2^2 e^{-x_1} + x_1 x_2 \cos x_1 x_2 \\ x_1^2 + x_2^2 - 2x_1 x_2 + x_1 - x_2 \\ -x_1^2 - x_2^2 \cos x_1 x_2 \end{bmatrix}$$

解 上述最优化问题可以通过下面的语句直接求解,并选择随机数为初值,则可以得出原问题的解为 $\mathbf{x} = [0.5319, 0.6876]$ 。

```
>> f=@(x) [x(1)^2*sin(x(2))+x(2)-3*x(1)*x(2)*cos(x(1));
            -x(1)^2*exp(-x(2))-x(2)^2*exp(-x(1))+x(1)*x(2)*cos(x(1)*x(2));
```

```

x(1)^2+x(2)^2-2*x(1)*x(2)+x(1)-x(2);
-x(1)^2-x(2)^2*cos(x(1)*x(2))];
A=[4.3 3.8; 1 1]; B=[4.9; 3]; x=fminimax(f,rand(2,1),A,B)

```

其实,有了 `fminimax()` 函数,还可以求解相关的变形问题,如极小极大问题

$$J = \min \left[\min_{\mathbf{x} \text{ s.t. } \mathbf{G}(\mathbf{x}) \leq 0} f_i(\mathbf{x}) \right] \quad (6-6-10)$$

该问题可以直接转换成下面的极小极大问题

$$J = \min \left[\max_{\mathbf{x} \text{ s.t. } \mathbf{G}(\mathbf{x}) \leq 0} -f_i(\mathbf{x}) \right] \quad (6-6-11)$$

6.6.6 目标规划问题求解

在实际最优化问题的求解过程中,有的时候会发现找不到可行解,这就需要放松约束条件,比如,不等式约束条件 $\mathbf{Ax} \leq \mathbf{B}$ 可以改写成 $\mathbf{Ax} \leq \mathbf{B} + \mathbf{d}^- - \mathbf{d}^+$,而实际求解过程中,将偏差对 $(\mathbf{d}^-, \mathbf{d}^+)$ 引入目标函数,使得偏差最小。相应地,目标函数也应该给出某一满意指标。例如,若目标函数为运费,则规划者应该有一个能承受的范围,这样,目标规划问题事实上转换为在运费能接受的前提下尽量减小偏差,使得严格的不等式约束尽可能小地被突破。这时,相应的最优化问题称为目标规划问题。

MATLAB 的最优化工具箱提供了目标规划问题的求解函数 `fgoalattain()`,可以直接求解下述目标规划的标准型问题

$$\min \quad \gamma \quad (6-6-12)$$

$$\mathbf{x}, \gamma \text{ s.t. } \begin{cases} \mathbf{F}(\mathbf{x}) - \mathbf{w}\gamma \leq \mathbf{g} \\ \mathbf{Ax} \leq \mathbf{B} \\ \mathbf{A}_{eq}\mathbf{x} = \mathbf{B}_{eq} \\ \mathbf{c}(\mathbf{x}) \leq 0 \\ \mathbf{c}_{eq}(\mathbf{x}) = 0 \\ \mathbf{x}_m \leq \mathbf{x} \leq \mathbf{x}_M \end{cases}$$

其中, $\mathbf{F}(\mathbf{x})$ 为原始的多目标向量, \mathbf{w} 为各个目标函数的加权系数, \mathbf{g} 为用户人为引入的目标。该函数的调用格式为

```
x = fgoalattain(F,x0,g,w,A,B,Aeq,Beq,xm,xM,CF,OPT,p1,p2,...)
```

例 6-53 试用目标规划方法求解例 6-44 中给出的问题。

解 显然,两个目标函数可以接受的目标分别是 20 和 6 (更确切地说是 -6,因为需要将最大化问题转换成最小化问题)。除此之外,还需要人为地选择权重,例如若更看重“少花钱”这一指标,则可以将其权重设置为 80%,而将另一个指标的权重设置成 20%,这样就可以调用下面的语句直接求解原问题,得出 $\mathbf{x} = [0, 3, 3.6875]^T$, 且 $\mathbf{f}(\mathbf{x}) = [17.25, -6.6875]^T$

```

>> f=@(x)[4,2.8,2.4]*x; [-1 -1 -1]*x]; Aeq=[]; Beq=[]; xm=[0;0;0];
x0=xm; w=[0.8,0.2]; goal=[20; -6]; A=[-1 -1 0]; B=[-3];
x=fgoalattain(f,x0,goal,w,A,B,[],[],xm), f(x)

```

6.7 动态规划及其在路径规划中的应用

前面介绍的最优化问题均属于静态最优化问题,因为目标函数和约束条件都是事先固定好的。在实际的科学研究中,有时会遇到另外一类问题,其目标函数和其他要求呈明显的阶段性和序列性,例如在生产计划制订时,每一年度的计划均取决于前一年的实际情况,这样,最优化问题不再是静态的了,而需要引入动态的最优化问题。

动态规划是 Richard Bellman^[14]在 1959 年引入的一个新的最优化领域,该成就是所谓现代控制理论的三个基础之一。该理论在多段决策过程和网络路径优化等领域有重要的作用。本节主要介绍动态规划在有向图和一般路径规划问题的最短路径求解中的应用。

6.7.1 图的矩阵表示方法

在介绍图表示之前,先给出一些关于图的基本概念并介绍图的 MATLAB 描述方法。在图论中,图是由节点和边构成的,所谓边,就是连接两个节点的直接路径。如果边是有向的,则图称为有向图,否则称为无向图。

图可以有多种表示方法,然而,最适合计算机表示和处理的是矩阵表示方法。假设一个图有 n 个节点,则可以用一个 $n \times n$ 矩阵 \mathbf{R} 来表示它。假设由节点 i 到节点 j 的边权值为 k ,则相应的矩阵元素可以表示为 $\mathbf{R}(i, j) = k$ 。这样的矩阵称为关联矩阵。若第 i 和第 j 节点间不存在边,则可令 $\mathbf{R}(i, j) = 0$,当然,也有的算法要求 $\mathbf{R}(i, j) = \infty$,后面将作相应的介绍。

MATLAB 语言还支持关联矩阵的稀疏矩阵表示方法。假设已知某图由 n 个节点构成,图中含有 m 条边,由 a_i 节点出发到 b_i 节点为止的边权值为 $w_i, i = 1, 2, \dots, m$ 。这样,可以建立三个向量,并由它们构造出关联矩阵

```
a = [a1, a2, ..., am, n];    % 起始节点向量
b = [b1, b2, ..., bm, n];    % 终止节点向量
w = [w1, w2, ..., wm, 0];    % 边权值向量
R = sparse(a, b, w);          % 关联矩阵的稀疏矩阵表示
```

注意,各个向量最后的一个值使得关联矩阵成为方阵,这是很多搜索方法所要求的。一个稀疏矩阵可以由 `full()` 函数变换成常规矩阵,而常规矩阵可以由 `sparse()` 函数转换成稀疏矩阵。

6.7.2 有向图的路径寻优

有向图与最优路径搜索是很多领域都能遇到的常见问题,应用动态规划理论,通常需要从终点反推回起点,搜索最优路径。本小节先给出一个例子,演示手工反推的寻优方法,然后将介绍基于 MATLAB 生物信息学工具箱^[15]的最优路径求解方法,最后将介绍一个实用的 Dijkstra 算法及其 MATLAB 实现。

1. 有向图最短路径问题的手工求解

这里将通过一个有向图研究的实例来介绍动态规划问题的手工求解方法。

例 6-54 考虑如图 6-14 所示的有向图^[16], 路径上的数字为从该路径起始节点到终止节点所花费的时间, 试求出从节点①到节点⑨的最优路径。

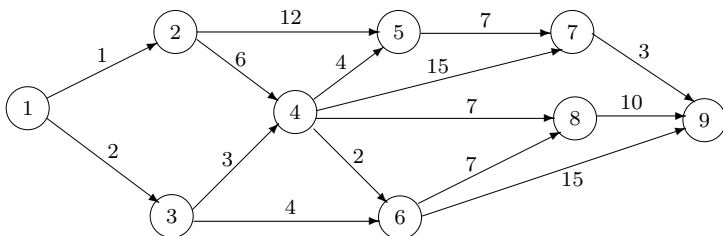


图 6-14 有向图的最短路径问题

解 先考虑终点, 即节点⑨, 将其时刻设置为 0, 表示为 (0)。下一个步骤是求出和它相连的上一级节点⑥⑦⑧的最短路径, 由于这些节点到节点⑨只有一个边, 故它们的时刻值分别标注为 (15), (3) 和 (10), 即相应边的时间。由节点⑤到节点⑦的边只有一条, 故节点⑤的标注应该为节点⑦的标注加上这条边的时间, 即 (10)。现在分析节点④的标注, 由节点④出发的路径分别到达节点⑤⑥⑦⑧, 将这些节点的标注值和边的权值相加, 可以发现, 节点④到节点⑤的路径与其标注的和最小, 为 14, 而到节点⑥⑦⑧的值依次为 17, 18, 17, 故节点④应该标注 (14)。节点②③的标注应该为由它们出发到下一级节点的路径值与标注和的最小值, 故发现由节点④返回节点②③的值最小, 可以分别标注为 (20) 和 (17), 这样返回节点①的最短路径应该是 19, 即由节点③返回路径最短。综上, 最短路径为 ①→③→④→⑤→⑦→⑨, 如图 6-15 所示。

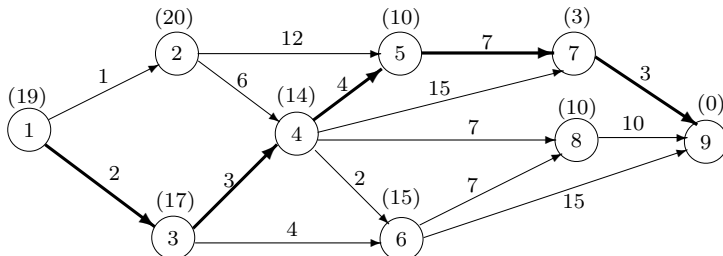


图 6-15 有向图最短路径问题的手工求解

由上面叙述可见, 求解的方法较直观, 且简单易行, 然而, 对大规模问题来说, 这样的过程可能很烦琐, 容易出错, 应该引入好的算法和程序求解这类问题。

2. 有向图搜索及图示

生物信息学工具箱中提供了有向图及最短路径搜索的现成函数, 如 `biograph()` 可以建立有向图对象, `view()` 函数可以显示有向图, 而 `graphshortestpath()` 函数可以直接求解最短路径问题。这些函数的具体调用格式为

```
P = biograph(R) % 建立有向图对象 P
[d, p] = graphshortestpath(P, n1, n2) % 求解最短路径问题
```

其中, R 为连接关系矩阵, 它可以为普通的矩阵形式, 也可以是稀疏矩阵的形式, 其具体表示方法在后面例子中给出演示。`biograph()` 函数还将允许其他的参数。对图 6-14 中描述的

有向图来说, $R(i, j)$ 的值表示由节点 i 出发, 到节点 j 为止的路径的权值。建立了有向图对象 P 后, 则由 `graphshortestpath()` 函数可以直接求解最短路径问题, 输入变量 n_1 和 n_2 为起始和终止节点序号, d 为最短距离, 而 p 为最短路径上节点序号构成的序列。在图示结果中, 还需要调用其他的函数来进一步修饰, 这些函数后面将通过实例演示。

例 6-55 试利用生物信息学工具箱中函数重新求解例 6-54 中的问题。

解 由图 6-14 中的节点与路径关系可以手工整理出表 6-4, 列出了每条路径的起始与终止节点即权值。由下面的语句可以按照稀疏矩阵的格式输入关联矩阵, 并建立起有向图的描述, 并用图形表示出该有向图, 如图 6-16 (a) 所示。注意, 在构造关联矩阵 R 时, 应使得它为方阵。

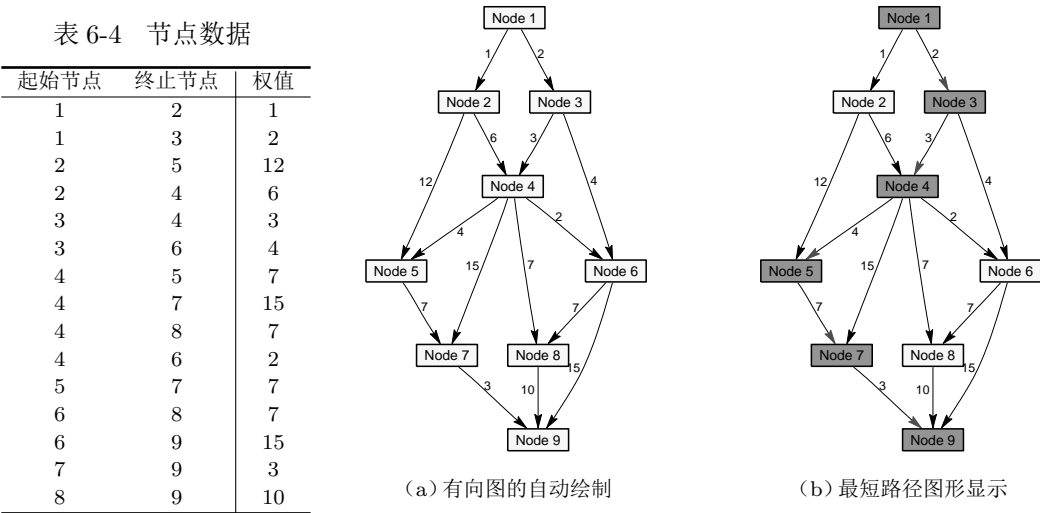


图 6-16 有向图的最短路径问题的解

```
>> ab=[1 1 2 2 3 3 4 4 4 4 5 6 6 7 8]; bb=[2 3 5 4 4 6 5 7 8 6 7 8 9 9 9];  
w=[1 2 12 6 3 4 4 15 7 2 7 7 15 3 10]; R=sparse(ab,bb,w); R(9,9)=0;  
h=view(biograph(R,[],'ShowWeights','on')) % 显示各个路径权值,并赋给句柄 h
```

建立了有向图对象 R , 则可以由 `graphshortestpath()` 函数求解最短路径, 并将其显示出来, 如图 6-16 (b) 所示。可见, 这样得出的结果与前面手工推导出的结果完全一致。

```
>> [d,p]=graphshortestpath(R,1,9) % 求节点①到节点⑨的最短路径  
set(h.Nodes(p),'Color',[1 0.4 0.4])  
edges=getedgesbynodeid(h,get(h.Nodes(p),'ID'));  
set(edges,'LineColor',[1 0 0]) % 上面语句用红色修饰最短路径
```

3. Dijkstra 最短路径算法及实现

两个节点间的最短路径可以通过 Dijkstra 最短路径算法^[17]直接求出。事实上, 如果指定了起始节点, 则该点到其他所有节点的最短路径可以一次性地求出, 而不会影响该算法的搜索速度。在最优路径搜索中, Dijkstra 是最有效的方法之一。假设节点个数为 n , 起始节点为 s , 则该算法的具体步骤为:

(1) **初始化**。建立三个向量存储各节点的状态,其中 **visited** 表示各个节点是否更新,初始值为 0; **dist** 存储起始节点到本节点的最短距离,初始值为 ∞ ; **parent** 向量存储到本节点的上一个节点,默认值为 0。另外设起始节点处 $\text{dist}(s) = 0$ 。

(2) **循环求解**。让 i 作 $n - 1$ 次循环,更新能由本节点经过一个边到达的节点距离与上级节点信息,并更新由本节点可以到达的未访问节点的最短路径信息。循环直到所有未访问节点完全处理完成。

(3) **提取到终止节点 t 的最短路径**。利用 **parent** 向量逐步提取最优路径。

根据 Dijkstra 搜索算法,可以编写出如下 MATLAB 程序

```
function [d,path]=dijkstra(W,s,t)
[n,m]=size(W); ix=(W==0); W(ix)=Inf;
if n~=m, error('Square W required'); end
visited(1:n)=0; dist(1:n)=Inf; parent(1:n)=0; dist(s)=0; d=Inf;
for i=1:(n-1), % 求出每个节点与起始节点关系
    ix=(visited==0); vec(1:n)=Inf; vec(ix)=dist(ix);
    [a,u]=min(vec); visited(u)=1;
    for v=1:n, if (W(u,v)+dist(u)<dist(v)),
        dist(v)=dist(u)+W(u,v); parent(v)=u;
    end; end; end
if parent(t)~=0, path=t; d=dist(t); % 回溯最短路径
    while t~=s, p=parent(t); path=[p path]; t=p; end
end
```

该函数的调用格式为 $[d, p] = \text{dijkstra}(W, s, t)$, 其中, W 为关联矩阵, s 和 t 分别为起始节点和终止节点的序号。返回的 d 为最短加权路径长度, p 为最优路径节点的序号向量。注意, 在该程序中, W 矩阵为 0 的权值将自动设置为 ∞ , 使得 Dijkstra 算法能正常运行。

例 6-56 试用 Dijkstra 算法重新求解例 6-54 中的问题。

解 下面语句可以直接用于求解, 结果和前面的也完全一致。

```
>> ab=[1 1 2 2 3 3 4 4 4 4 5 6 6 7 8]; bb=[2 3 5 4 4 6 5 7 8 6 7 8 9 9 9];
w=[1 2 12 6 3 4 4 15 7 2 7 7 15 3 10]; R=sparse(ab,bb,w); R(9,9)=0;
W=ones(9); [d,p]=dijkstra(R.*W,1,9)
```

6.7.3 无向图的路径最优搜索

在实际应用中, 比如在城市道路寻优问题中, 所涉及的图通常是无向图, 因为两个节点 A、B 间, 既可以由节点 A 走向 B, 也可以由节点 B 走向 A。无向图的具体处理方法其实也很简单。在无向图中若不存在环路, 即某条边的起点和终点为同一节点, 则可以先按照有向图的方式构造关联矩阵 R , 这时, 无向图的关联矩阵 R_1 可以由 $R_1 = R + R^T$ 直接计算出来。如果无向图中, 某些边是有向的, 例如城市中的单行路, 则可以在得出 R_1 之后, 手工修改该矩阵。例如从节点 i 到节点 j 的边是有向的, 从 i 到 j , 这样应该手工设置成 $R_1(j, i) = 0$ 。

对一般无向图来说, 由节点 i 到 j 与由节点 j 到 i 的边权值是不同的, 比如在城市交通

中,涉及上坡和下坡的问题,则需要对 R_1 矩阵的某些值使用手工方法重新定义和修改。

6.7.4 绝对坐标节点的最优路径规划算法与应用

如果各个节点以绝对坐标 (x_i, y_i) 的方式给出,且给出节点间的连接关系,则边权值可以由两点间的 Euclid 距离计算处理,这样就可以直接进行路径规划问题的求解了。下面将通过例子演示相应问题的求解方法。

例 6-57 假设有 11 个城市,其分布的坐标分别为 (4, 49), (9, 30), (21, 56), (26, 26), (47, 19), (57, 38), (62, 11), (70, 30), (76, 59), (76, 4), (96, 4), 其间的公路如图 6-17 所示。试求出由城市 A 到城市 B 的最短路径。如果城市 6 与 8 之间修路,试重新搜索最优路径。

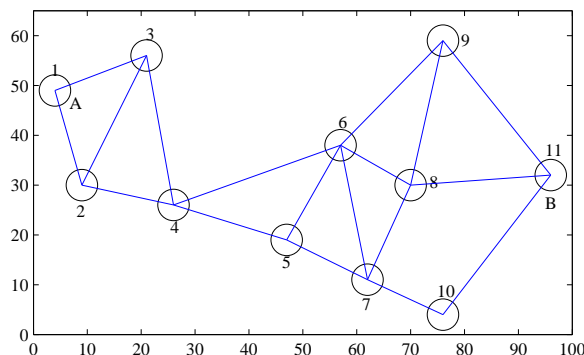


图 6-17 城市布局 and 交通图

解 首先输入关联矩阵,可以先按有向图的方式输入该稀疏矩阵,并将连接的权值设置成 1。然后再按无向图的方式转换出所需的关联矩阵。矩阵的实际权值可以由节点间的 Euclid 距离,即 $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ 直接计算出来。这样,有效的权值矩阵可以由这两个矩阵的点乘得出。由下面的语句可以得出最优的路径为 $1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 11$,且最短距离为 111.6938。

```
>> x=[4,9,21,26,47,57,62,70,76,76,96]; y=[49,30,56,26,19,38,11,30,59,4,32];
    for i=1:11, for j=1:11, % 计算 Euclid 距离矩阵
        D(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
    end, end
    n1=[1 1 2 2 3 4 4 5 5 6 6 6 7 8 7 10 8 9];
    n2=[2 3 3 4 4 5 6 6 7 7 8 9 8 9 10 11 11 11];
    R=sparse(n1,n2,1); R(11,11)=0; R=R+R'; [d,p]=dijkstra(R.*D,1,11)
```

如果节点 6 和节点 8 之间的路不通,则可以设置 $R(8,6) = R(6,8) = \infty$ 。这样,由下面的语句可以重新求解最优路径问题,得出 $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 11$,最短距离为 122.9394。

```
>> R(6,8)=Inf; R(8,6)=Inf; [d,p]=dijkstra(R.*D,1,11)
```

6.8 习 题

1. 求解能转换成多项式方程的联立方程,并检验得出的高精度数值解的精度。

$$(1) \begin{cases} x_1^2 - x_2 - 1 = 0 \\ (x_1 - 2)^2 + (x_2 - 0.5)^2 - 1 = 0 \end{cases}, \quad (2) \begin{cases} x^2 y^2 - zxy - 4x^2 yz^2 = xz^2 \\ xy^3 - 2yz^2 = 3x^3 z^2 + 4xyz^2 \\ y^2 x - 7xy^2 + 3xz^2 = x^4 zy \end{cases}.$$

2. 试用图解法求解下面的一元和二元方程, 并验证得出的结果。

$$(1) f(x) = e^{-(x+1)^2 + \pi/2} \sin(5x + 2), \quad (2) f(x, y) = (x^2 + y^2 + xy)e^{-x^2 - y^2 - xy}.$$

3. 用数值求解函数求解习题2中方程的根, 并对得出的结果进行检验。

4. 试找出下面 Riccati 变形方程全部的解矩阵, 并验证得出的结果。

$$\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{D} - \mathbf{X}\mathbf{B}\mathbf{X} + \mathbf{C} = 0$$

其中

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 9 \\ 9 & 7 & 9 \\ 6 & 5 & 3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 3 & 6 \\ 8 & 2 & 0 \\ 8 & 2 & 8 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 7 & 0 & 3 \\ 5 & 6 & 4 \\ 1 & 4 & 4 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 3 & 9 & 5 \\ 1 & 2 & 9 \\ 3 & 3 & 0 \end{bmatrix}$$

5. 试求出使 $\int_0^1 (e^x - cx)^2 dx$ 取极小值的 c 值。

6. 试求解下面的无约束最优化问题。

$$\min_{\mathbf{x}} \quad 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2) + (1 - x_3^2)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$$

7. 一组富有挑战性的最优化基准测试问题也可以用 MATLAB 语言直接求解, 试求解之。

(1) De Jong 问题^[18]

$$J = \min_{\mathbf{x}} \mathbf{x}^T \mathbf{x} = \min_{\mathbf{x}} (x_1^2 + x_2^2 + \cdots + x_p^2), \quad \text{其中 } x_i \in [-512, 512]$$

其中 $i = 1, \dots, p$ 。本问题的理论解为 $x_1 = \cdots = x_p = 0$ 。

(2) Griewangk 基准测试问题

$$J = \min_{\mathbf{x}} \left(1 + \sum_{i=1}^p \frac{x_i^2}{4000} - \prod_{i=1}^p \cos \frac{x_i}{\sqrt{i}} \right), \quad \text{where } x_i \in [-600, 600]$$

(3) Ackley 基准测试问题^[19]

$$J = \min_{\mathbf{x}} \left[20 + 10^{-20} \exp \left(-0.2 \sqrt{\frac{1}{p} \sum_{i=1}^p x_i^2} \right) - \exp \left(\frac{1}{p} \sum_{i=1}^p \cos 2\pi x_i \right) \right]$$

8. 考虑 Rastrigin 函数^[20] $f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos \pi x_1 + \cos \pi x_2)$, 试用三维曲面绘制该函数的函数值, 选择初值求取该函数的最小值, 并理解全局最优解和局部最优解的概念以及最优解对初值的依赖关系。

9. 试用图解法求解下面的非线性规划问题, 并用数值求解算法验证结果。

$$\begin{aligned} \min \quad & x_1^3 + x_2^2 - 4x_1 + 4 \\ \text{s.t.} \quad & \begin{cases} x_1 - x_2 + 2 \geq 0 \\ -x_1^2 + x_2 - 1 \geq 0 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \end{aligned}$$

10. 试求解下面的线性规划问题。

$$(1) \quad \min \quad -3x_1 + 4x_2 - 2x_3 + 5x_4, \quad (2) \quad \min \quad x_6 + x_7.$$

$$\mathbf{x} \text{ s.t. } \begin{cases} 4x_1 - x_2 + 2x_3 - x_4 = -2 \\ x_1 + x_2 - x_3 + 2x_4 \leq 14 \\ 2x_1 - 3x_2 - x_3 - x_4 \geq -2 \\ x_{1,2,3} \geq -1, x_4 \text{ 无约束} \end{cases} \quad \mathbf{x} \text{ s.t. } \begin{cases} x_1 + x_2 + x_3 + x_4 = 4 \\ -2x_1 + x_2 - x_3 - x_6 + x_7 = 1 \\ 3x_2 + x_3 + x_5 + x_7 = 9 \\ x_{1,2,\dots,7} \geq 0 \end{cases}$$

11. 试求解下面的二次型规划问题,并用图示的形式解释结果。

$$(1) \quad \min \quad 2x_1^2 - 4x_1x_2 + 4x_2^2 - 6x_1 - 3x_2, \quad (2) \quad \min \quad (x_1 - 1)^2 + (x_2 - 2)^2.$$

$$\mathbf{x} \text{ s.t. } \begin{cases} x_1 + x_2 \leq 3 \\ 4x_1 + x_2 \leq 9 \\ x_{1,2} \geq 0 \end{cases} \quad \mathbf{x} \text{ s.t. } \begin{cases} -x_1 + x_2 = 1 \\ x_1 + x_2 \leq 2 \\ x_{1,2} \geq 0 \end{cases}$$

12. 试求解下面的非线性规划问题。

$$\min \quad \frac{1}{2 \cos x_6} \left[x_1 x_2 (1 + x_5) + x_3 x_4 \left(1 + \frac{31.5}{x_5} \right) \right]$$

$$\mathbf{x} \text{ s.t. } \begin{cases} 0.003079x_1^3x_2^3x_5 - \cos^3 x_6 \geq 0 \\ 0.1017x_3^3x_4^3 - x_5^2 \cos^3 x_6 \geq 0 \\ 0.09939(1+x_5)x_1^3x_2^2 - \cos^2 x_6 \geq 0 \\ 0.1076(31.5+x_5)x_3^3x_4^2 - x_5^2 \cos^2 x_6 \geq 0 \\ x_3x_4(x_5+31.5) - x_5[2(x_1+5) \cos x_6 + x_1x_2x_5] \geq 0 \\ 0.2 \leq x_1 \leq 0.5, 14 \leq x_2 \leq 22, 0.35 \leq x_3 \leq 0.6 \\ 16 \leq x_4 \leq 22, 5.8 \leq x_5 \leq 6.5, 0.14 \leq x_6 \leq 0.2618 \end{cases}$$

13. 试求解下面的最优化问题。

$$\min \quad 0.6224x_1x_2x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$\mathbf{x} \text{ s.t. } \begin{cases} 0.0193x_3 - x_1 \leq 0 \\ 0.00954x_3 - x_2 \leq 0 \\ 750 \times 1728 - \pi x_3^2x_4 - 4\pi x_3^3/3 \leq 0 \\ x_4 - 240 \leq 0 \\ 0.0625 \leq x_{1,2} \leq 6.1875, 10 \leq x_3, x_4 \leq 200 \end{cases}$$

14. 试求解下面的最优化问题^[3]。

$$\min \quad k$$

$$\mathbf{q}, k \text{ s.t. } \begin{cases} g(\mathbf{q}) \leq 0 \\ 800 - 800k \leq q_1 \leq 800 + 800k \\ 4 - 2k \leq q_2 \leq 4 + 2k \\ 6 - 3k \leq q_3 \leq 6 + 3k \end{cases}$$

其中 $g(\mathbf{q}) = 10q_2^2q_3^3 + 10q_2^3q_3^2 + 200q_2^2q_3^2 + 100q_2^3q_3 + q_1q_2q_3^2 + q_1q_2^2q_3 + 1000q_2q_3^3 + 8q_1q_3^2 + 1000q_2^2q_3 + 8q_1q_2^2 + 6q_1q_2q_3 - q_1^2 + 60q_1q_3 + 60q_1q_2 - 200q_1$ 。

15. 求解下面的整数线性规划问题。

$$(1) \quad \max \quad 592x_1 + 381x_2 + 273x_3 + 55x_4 + 48x_5 + 37x_6 + 23x_7,$$

$$\mathbf{x} \text{ s.t. } \begin{cases} \mathbf{x} \geq \mathbf{0} \\ 3534x_1 + 2356x_2 + 1767x_3 + 589x_4 + 528x_5 + 451x_6 + 304x_7 \leq 119567 \end{cases}$$

$$(2) \quad \max \quad 120x_1 + 66x_2 + 72x_3 + 58x_4 + 132x_5 + 104x_6.$$

$$\mathbf{x} \text{ s.t. } \begin{cases} x_1 + x_2 + x_3 = 30 \\ x_4 + x_5 + x_6 = 18 \\ x_1 + x_4 = 10 \\ x_2 + x_5 \leq 18 \\ x_3 + x_6 \geq 30 \\ x_1, \dots, x_6 \geq 0 \end{cases}$$

16. 试求解下面的非线性整数规划问题^[5], 并用穷举方法检验结果。

$$(1) \quad \min \quad \left(\frac{1}{6.931} - \frac{x_2 x_3}{x_1 x_4} \right)^2,$$

$$\mathbf{x} \text{ s.t. } 12 \leq x_i \leq 32$$

$$(2) \quad \min \quad (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 10x_6 - 8x_7.$$

$$\mathbf{x} \text{ s.t. } \begin{cases} -2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 + 127 \geq 0 \\ 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 + 282 \geq 0 \\ 23x_1 - x_2^2 - 6x_6^2 + 8x_7 + 196 \geq 0 \\ -4x_1^2 - x_2^2 + 3x_1 x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0 \end{cases}$$

17. 试求解下面的 0-1 线性规划问题, 并用穷举方法检验得出的结果。

$$(1) \quad \min \quad 5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5,$$

$$\mathbf{x} \text{ s.t. } \begin{cases} x_1 - x_2 + 5x_3 + x_4 - 4x_5 \geq 2 \\ -2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0 \\ -2x_2 + 2x_3 - x_4 - x_5 \leq 1 \end{cases}$$

$$(2) \quad \min \quad -3x_1 - 4x_2 - 5x_3 + 4x_4 + 4x_5 + 2x_6.$$

$$\mathbf{x} \text{ s.t. } \begin{cases} x_1 - x_6 \leq 0 \\ x_1 - x_5 \leq 0 \\ x_2 - x_4 \leq 0 \\ x_2 - x_5 \leq 0 \\ x_3 - x_4 \leq 0 \\ x_1 + x_2 + x_3 \leq 2 \end{cases}$$

18. 试求解下面的线性 0-1 规划问题, 比较 `bintprog()` 和 `BNB20_new()` 得出的结果。

$$\max \quad -\mathbf{f}\mathbf{x}$$

$$\mathbf{x} \text{ s.t. } \mathbf{A}\mathbf{x} \leq \begin{bmatrix} 600 \\ 600 \end{bmatrix}$$

$$\text{其中 } \mathbf{A} = \begin{bmatrix} 45 & 0 & 85 & 150 & 65 & 95 & 30 & 0 & 170 & 0 & 40 & 25 & 20 & 0 \\ 30 & 20 & 125 & 5 & 80 & 25 & 35 & 73 & 12 & 15 & 15 & 40 & 5 & 10 \\ 0 & 25 & 0 & 0 & 25 & 0 & 165 & 0 & 85 & 0 & 0 & 0 & 0 & 100 \\ 10 & 12 & 10 & 9 & 0 & 20 & 60 & 40 & 50 & 36 & 49 & 40 & 19 & 150 \end{bmatrix}$$

$$\mathbf{f} = [1898, 440, 22507, 270, 14148, 3100, 4650, 30800, 615, 4975, 1160, 4225, 510, 11880, 479, 440, 490, 330, 110, 560, 24355, 2885, 11748, 4550, 750, 3720, 1950, 10500]$$

19. 试用鲁棒控制工具箱和 YALMIP 工具箱求解下面的最优化问题。

$$\min \quad \text{tr}(\mathbf{X})$$

$$\mathbf{X} \text{ s.t. } \begin{cases} \begin{bmatrix} \mathbf{A}^T \mathbf{X} + \mathbf{X} \mathbf{A} + \mathbf{Q} & \mathbf{X} \mathbf{B} \\ \mathbf{B}^T \mathbf{X} & -\mathbf{I} \end{bmatrix} < 0 \\ \mathbf{X} < 0 \end{cases}$$

其中 $A = \begin{bmatrix} -1 & -2 & 1 \\ 3 & 2 & 1 \\ 1 & -2 & -1 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, $Q = \begin{bmatrix} 1 & -1 & 0 \\ -1 & -3 & -12 \\ 0 & -12 & -36 \end{bmatrix}$ 。

20. 求解下面的线性矩阵不等式问题。

$$\begin{cases} P^{-1} > 0, \text{ 或等效地 } P > 0 \\ A_1 P + P A_1^T + B_1 Y + Y^T B_1^T < 0 \\ A_2 P + P A_2^T + B_2 Y + Y^T B_2^T < 0 \end{cases}$$

其中 $A_1 = \begin{bmatrix} -1 & 2 & -2 \\ -1 & -2 & 1 \\ -1 & -1 & 0 \end{bmatrix}$, $B_1 = \begin{bmatrix} -2 \\ 1 \\ -1 \end{bmatrix}$, $A_2 = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 0 & 1 \end{bmatrix}$, $B_2 = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix}$ 。

21. 试求解下面多目标线性规划问题的最佳妥协解。

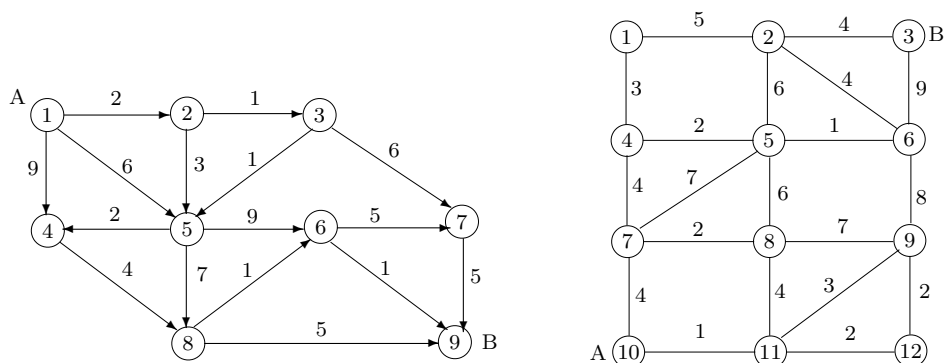
$$\begin{aligned} \max \quad & z_1 = 100x_1 + 90x_2 + 80x_3 + 70x_4 \\ \min \quad & z_2 = 3x_2 + 2x_4, \end{aligned}$$

(1) $\begin{cases} x_1 + x_2 \geq 30 \\ x_3 + x_4 \geq 30 \\ 3x_1 + 2x_2 \leq 120 \\ 3x_2 + 2x_4 \leq 48 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$

(2) $\max \begin{bmatrix} 50x_1 + 20x_2 + 100x_3 + 60x_4 \\ 20x_1 + 70x_2 + 5x_3 \\ 3x_2 + 5x_4 \\ 2x_1 + 20x_3 + 2x_4 \end{bmatrix}$

$\begin{cases} 2x_1 + 5x_2 + 10x_3 \leq 100 \\ x_1 + 6x_2 + 8x_4 \leq 250 \\ 5x_1 + 8x_2 + 7x_3 + 10x_4 \leq 350 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$

22. 试求出图 6-18(a)、(b) 中由节点 A 到节点 B 的最短路径。



(a) 有向图最优路径问题

(b) 无向图最优路径问题

图 6-18 有向图最短路径问题

23. 假设某人常驻城市为 C_1 , 他想不定期到其他城市 C_2, \dots, C_8 办事, 下面矩阵的 $R_{i,j}$ 表示从 C_i 到 C_j 的交通费用, 试设计出他从城市 C_1 到各个其他城市的最便宜交通路线图。

$$R = \begin{bmatrix} 0 & 364 & 314 & 334 & 330 & \infty & 253 & 287 \\ 364 & 0 & 396 & 366 & 351 & 267 & 454 & 581 \\ 314 & 396 & 0 & 232 & 332 & 247 & 159 & 250 \\ 334 & 300 & 232 & 0 & 470 & 50 & 57 & \infty \\ 330 & 351 & 332 & 470 & 0 & 252 & 273 & 156 \\ \infty & 267 & 247 & 50 & 252 & 0 & \infty & 198 \\ 253 & 454 & 159 & 57 & 273 & \infty & 0 & 48 \\ 260 & 581 & 220 & \infty & 156 & 198 & 48 & 0 \end{bmatrix}$$

参考文献

- [1] Nelder J A, Mead R. A simplex method for function minimization. *Computer Journal*, 1965, 7:308–313
- [2] D’Errico J. Fminsearchbnd. MATLAB Central File ID: #8277, 2005
- [3] Henrion D. A review of the global optimization toolbox for Maple, 2006
- [4] Kuipers K. BNB20 solves mixed integer nonlinear optimization problems, MATLAB Central File ID: #95, 2000
- [5] Leyffer S. Deterministic methods for mixed integer nonlinear programming. Ph.D. thesis, Department of Mathematics & Computer Science, University of Dundee, U.K., 1993
- [6] Cha J Z, Mayne R W. Optimization with discrete variables via recursive quadratic programming: Part 2 – algorithms and results. *Transactions of the ASME, Journal of Mechanisms, Transmissions, and Automation in Design*, 1994, 111:130–136
- [7] Boyd S, Ghaoui L El, Feron E, et al. *Linear matrix inequalities in systems and control theory*. Philadelphia: SIAM books, 1994
- [8] Willems J C. Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Transactions on Automatic Control*, 1971, 16(6):621–634
- [9] Scherer C, Weiland S. *Linear matrix inequalities in control*. Delft University of Technology, 2005
- [10] Löfberg J. YALMIP: a toolbox for modeling and optimization in MATLAB. *Proceedings of IEEE International Symposium on Computer Aided Control Systems Design*. Taipei, 2004, 284–289
- [11] Löfberg J. YALMIP 下载地址 <http://control.ee.ethz.ch/~joloef/yalmip.php>
- [12] 高雷阜. 最优化理论与方法. 沈阳: 东北大学出版社, 2005
- [13] Cao Y. Pareto set. MATLAB Central File ID: # 15181, 2007
- [14] Bellman R. *Dynamic programming*. Princeton, NJ: Princeton University Press, 1957
- [15] The MathWorks Inc. *Bioinformatics users manual*
- [16] 林诒勋. 动态规划与序贯最优化. 郑州: 河南大学出版社, 1997
- [17] Dijkstra E W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959, 1:269–271
- [18] Chipperfield A, Fleming P. *Genetic algorithm toolbox user’s guide*. Department of Automatic Control and Systems Engineering, University of Sheffield, 1994
- [19] Ackley D H. *A connectionist machine for genetic hillclimbing*. Boston, USA: Kluwer Academic Publishers, 1987
- [20] Goldberg D E. *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley, 1989

第7章 微分方程问题的计算机求解

微分方程是描述动态系统最常用的数学工具,也是很多科学与工程领域数学建模的基础。线性微分方程和低阶特殊微分方程往往可以通过解析解的方法求解,但一般的非线性微分方程是没有解析解的,故需要用数值解的方式求解。本章 7.1 节将研究微分方程的解析解算法,介绍在 MATLAB 环境中如何用微分方程求解函数直接得出线性微分方程组的解析解,并对一阶简单的非线性微分方程的解析解求解进行探讨,从而得出结论,一般非线性微分方程是没有解析解的。7.2 节引入数值解的概念,并以最简单的一阶微分方程的 Euler 算法为例,介绍一般数值解法的思路并介绍了变步长求解的概念,还介绍 MATLAB 下微分方程的实用数值求解函数,通过例子演示该函数在一般一阶显式常微分方程组的数值求解方法及 MATLAB 的应用,并介绍数值解正确性的验证方法。由于一般微分方程初值函数能直接求解的方程是一阶显式微分方程组,若给出的方程不是这类函数,则需要首先将其变换成一阶显式微分方程组,然后使用常规方法对其求解。7.3 节介绍状态变量的选择方法以及各种不同微分方程转换成一阶显式微分方程组的一般性方法,以便使用给定的求解函数直接求解。7.4 节将介绍其他各类常微分方程数值求解的方法及 MATLAB 实现,包括刚性微分方程、微分代数方程组、隐式微分方程组、切换微分方程以及随机微分方程的数值求解方法。7.5 节介绍各类延迟微分方程的数值求解方法,包括一般延迟微分方程、变延迟微分方程和中立型微分方程的数值解方法。7.6 节和 7.7 节将分别介绍常微分方程组边值问题的数值解法和一般偏微分方程的数值求解方法,并将介绍如何用 MATLAB 语言的偏微分方程工具箱提供的程序界面求解偏微分方程的方法及步骤。7.8 节还将简介 Simulink 仿真环境,并将介绍如何在 Simulink 环境下建立微分方程的数学模型,还将介绍通过仿真求解微分方程的一般步骤及方法,用这样的方法理论上可以求解任意复杂的常微分方程组初值问题数值解。

7.1 常系数线性微分方程的解析解方法

7.1.1 线性常系数微分方程解析解的数学描述

假设已知常系数线性微分方程的一般描述为

$$\begin{aligned} \frac{d^n y(t)}{dt^n} + a_1 \frac{d^{n-1} y(t)}{dt^{n-1}} + a_2 \frac{d^{n-2} y(t)}{dt^{n-2}} + \cdots + a_{n-1} \frac{dy(t)}{dt} + a_n y(t) = \\ b_1 \frac{d^m u(t)}{dt^m} + b_2 \frac{d^{m-1} u(t)}{dt^{m-1}} + \cdots + b_m \frac{du(t)}{dt} + b_{m+1} u(t) \end{aligned} \quad (7-1-1)$$

其中, a_i, b_i 均为常数,利用 5.1 节介绍的性质,对零初值问题有 $\mathcal{L}[d^m y(t)/dt^m] = s^m \mathcal{L}[y(t)]$, 可以对应得出下面的多项式代数方程

$$s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_{n-1} s + a_n = 0 \quad (7-1-2)$$

假设代数方程的特征值 s_i 均可以求出,且假设它们均相异,则可以得出原微分方程的解析解一般形式为

$$y(t) = C_1 e^{r_1 t} + C_2 e^{r_2 t} + \cdots + C_n e^{r_n t} + \gamma(t) \quad (7-1-3)$$

其中, C_i 为待定系数,而 $\gamma(t)$ 是满足 $u(t)$ 输入的一个特解。 s_i 有重根的情况也有相应的解析解形式。

从得出的代数方程式 (7-1-2) 看,由著名的 Abel-Ruffini 定理可知,4 次及以下的多项式代数方程是能求出根的解析解的,故可以得出结论,低阶常系数线性微分方程有一般意义下的解析解,结合多项式方程的准解析解法可以得出一般高次多项式代数方程的高精度数值解,构造出高阶常系数线性微分方程的准解析解方法。本节将介绍用 MATLAB 语言及其符号运算工具箱求解线性常系数微分方程解析解的方法。

7.1.2 微分方程的解析解方法

MATLAB 语言的符号运算工具箱提供了一个线性常系数微分方程求解的实用函数 `dsolve()`,该函数允许用字符串的形式描述微分方程及初值、边值条件,最终将得出微分方程的解析解。该函数的调用格式为

```
y = dsolve(f1,f2,...,fm)      % 默认自变量为 t
y = dsolve(f1,f2,...,fm,'x') % 指明自变量
```

其中,字符串型变量 f_i 既可以描述微分方程,又可以描述初始条件或边界条件。在描述微分方程时,可以用 `D4y` 这样的记号表示 $y^{(4)}(t)$,还可以用 `'D2y(2) = 3'` 这类记号表示 $y''(2) = 3$ 这样的已知条件,该函数可以容易地得出原微分方程的解析解。特别注意:如果描述微分方程的自变量不是 t 而是 x ,则可以由后一个 MATLAB 语句格式指明自变量,否则将得出错误的结果。

例 7-1 假设输入信号为 $u(t) = e^{-5t} \cos(2t + 1) + 5$,试求出下面微分方程的通解。

$$y^{(4)}(t) + 10y'''(t) + 35y''(t) + 50y'(t) + 24y(t) = 5u''(t) + 4u'(t) + 2u(t)$$

解 若想求解本微分方程,首先应该定义 t 为符号变量,这样就可以推导出给定微分方程等式右侧的表达式,用 `char()` 函数将其转换为字符串,再和方程左侧的表达式串联起来,则可以构造出整个方程的字符串表达式,对其求解则可以得出其解析解

```
>> syms t y; u=exp(-5*t)*cos(2*t+1)+5; uu=5*diff(u,t,2)+4*diff(u,t)+2*u;
y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',char(uu)]); y=simple(y)
```

通过上述语句的求解,可以得出原微分方程的通解为

$$y(t) = \frac{5}{12} - \frac{343}{520} e^{-5t} \cos(2t + 1) - \frac{547}{520} e^{-5t} \sin(2t + 1) + C_1 e^{-4t} + C_2 e^{-3t} + C_3 e^{-2t} + C_4 e^{-t}$$

其中, C_i 为任意常数(新版符号运算工具箱中 C_i 的序号杂乱无章,本书统一改成从 C_1 开始的序号)。若给出初始条件或边界条件,则可以通过这些条件建立方程,求出 C_i 的值。这样的结果和高等数学中微分方程求解是一致的。

在新版本的 MATLAB 下,这些任意常数的变量名是自动生成的,在本例中自动生成的变量名

是 $C_2 \sim C_5$, 所以要检验上述微分方程的通解, 可以将其代入原方程 (有时需要调用 `simple()` 函数化简得出的误差), 可见误差为零, 说明这样得出的通解满足原始方程。

```
>> diff(y,4)+10*diff(y,3)+35*diff(y,2)+50*diff(y)+24*y-uu
```

仍考虑上面的微分方程, 假设已知 $y(0) = 3, y'(0) = 2, y''(0) = y'''(0) = 0$, 则可以通过下面的命令求取满足该微分方程的特解

```
>> y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',char(uu)],...
            'y(0)=3','Dy(0)=2','D2y(0)=0','D3y(0)=0'])
```

可以得出满足该条件的方程解析解为

$$y(t) = \frac{5}{12} - \frac{343}{520}e^{-5t}\cos(2t+1) - \frac{547}{520}e^{-5t}\sin(2t+1) \\ + \left(-\frac{445}{26}\cos 1 - \frac{51}{13}\sin 1 - \frac{69}{2}\right)e^{-2t} + \left(-\frac{271}{30}\cos 1 + \frac{41}{15}\sin 1 - \frac{25}{4}\right)e^{-4t} \\ + \left(\frac{179}{8}\cos 1 + \frac{5}{8}\sin 1 + \frac{73}{3}\right)e^{-3t} + \left(\frac{133}{30}\cos 1 + \frac{97}{60}\sin 1 + 19\right)e^{-t}$$

利用强大的 MATLAB 符号运算工具箱, 还可以求解出以往看似不可能的问题的解析解。例如, 设置 $y(0) = 1/2, y'(\pi) = 1, y''(2\pi) = 0, y'(2\pi) = 1/5$, 则可以由下面的命令直接求解方程

```
>> y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',char(uu)],...
            'y(0)=1/2','Dy(pi)=1','D2y(2*pi)=0','Dy(2*pi)=1/5'])
```

如果用推导的方法求 C_i 的值, 则每个系数的解析解至少要写出 10 数行, 所以应该采用近似的方式从方程的解析解由 `vpa(y,10)` 解出 C_i 系数

$$y(t) = \frac{5}{12} - \frac{343}{520}e^{-5t}\cos(2t+1) - \frac{547}{520}e^{-5t}\sin(2t+1) - 219.1291604e^{-t} \\ + 442590.9052e^{-4t} + 31319.63786e^{-2t} - 473690.0889e^{-3t}$$

例 7-2 前面介绍的方程只含有实数极点, 其实符号运算工具箱提供的 `dsolve()` 函数同样适用于有复数极点的微分方程解析解。假设微分方程如下

$$y^{(5)}(t) + 5y^{(4)}(t) + 12y'''(t) + 16y''(t) + 12y'(t) + 4y(t) = u'(t) + 3u(t)$$

且假设输入信号为正弦信号 $u(t) = \sin t$, 并假设 $y(0) = y'(0) = y''(0) = y'''(0) = y^{(4)}(0) = 0$, 试用解析方法求解该方程。

解 用下面的方法可以求出原微分方程的解析解

```
>> syms t y; u=sin(t); uu=3*diff(u)+3*u;
y=dsolve(['D5y+5*D4y+12*D3y+16*D2y+12*Dy+4*y=',char(uu)],...
        'y(0)=0','Dy(0)=0','D2y(0)=0','D3y(0)=0','D4y(0)=0'])
```

其解析解的数学描述为

$$y(t) = -\frac{12}{25}\cos t - \frac{9}{25}\sin t + \frac{57}{50}e^{-t}\sin t + \frac{12}{25}e^{-t}\cos t + \frac{3}{5}te^{-t}\sin t - \frac{3}{10}te^{-t}\cos t$$

或更简单地手工修改为 $y(t) = -\frac{12}{25}\cos t - \frac{9}{25}\sin t + \left(\frac{57}{50} + \frac{3}{5}t\right)e^{-t}\sin t + \left(\frac{12}{25} - \frac{3}{10}t\right)e^{-t}\cos t$ 。

例 7-3 试求解线性微分方程组
$$\begin{cases} x''(t) + 2x'(t) = x(t) + 2y(t) - e^{-t} \\ y'(t) = 4x(t) + 3y(t) + 4e^{-t} \end{cases}$$

解 线性微分方程组也可以用 `dsolve()` 函数直接求解。上述线性微分方程组可以由下面的 MATLAB 语句直接求解

```
>> [x,y]=dsolve('D2x+2*Dx=x+2*y-exp(-t)', 'Dy=4*x+3*y+4*exp(-t)')
```

这样得出的结果为

$$\begin{cases} x(t) = -6te^{-t} + C_1e^{-t} + C_2e^{(1+\sqrt{6})t} + C_3e^{-(1+\sqrt{6})t} \\ y(t) = 6te^{-t} - C_1e^{-t} + 2(2+\sqrt{6})C_2e^{(1+\sqrt{6})t} + 2(2-\sqrt{6})C_3e^{-(1+\sqrt{6})t} + \frac{1}{2}e^{-t} \end{cases}$$

例 7-4 试求出时变线性微分方程 $(2x+3)^3y''' + 3(2x+3)y'' - 6y = 0$ 的解析解。

解 对某些时变线性微分方程仍然可以试用 `dsolve()` 函数直接求解。如果用下面语句求解

```
>> y=dsolve('(2*x+3)^3*D3y+3*(2*x+3)*Dy-6*y=0'); y=simple(y)
```

则得出的是错误的结果。由于此例的自变量是 x , 不是默认的 t , 所以要在调用语句中给出 'x' 选项

```
>> y=dsolve('(2*x+3)^3*D3y+3*(2*x+3)*Dy-6*y=0', 'x'); y=simple(y)
```

```
syms x; simple((2*x+3)^3*diff(y,x,3)+3*(2*x+3)*diff(y,x)-6*y)
```

上面的语句可以得出的微分方程解析解如下, 将解析解代入原方程可知误差为 0。

$$y(x) = \frac{\sqrt{2x+3}}{2} \left(6\sqrt{2}C_3 - 2\sqrt{2}C_2 + C_1\sqrt{2x+3} + 2\sqrt{2}C_3x \right)$$

例 7-5 试求解下面的高阶常系数线性微分方程组

$$\begin{cases} x'' - x + y + z = 0, \\ x + y'' - y + z = 0, \quad x(0) = 1, y(0) = z(0) = x'(0) = y'(0) = z'(0) = 0 \\ x + y + z'' - z = 0, \end{cases}$$

解 用下面的语句可以求解上面给出的微分方程组

```
>> [x,y,z]=dsolve('D2x-x+y+z=0', 'x+D2y-y+z=0', 'x+y+D2z-z=0', ...
'x(0)=1, y(0)=0, z(0)=0', 'Dx(0)=0, Dy(0)=0, Dz(0)=0')
```

上面的语句得出方程的解为

$$x(t) = \frac{e^{\sqrt{2}t}}{3} + \frac{e^{-\sqrt{2}t}}{3} + \frac{\cos t}{3}, \quad y(t) = \frac{\cos t}{3} - \frac{e^{-\sqrt{2}t}}{6} - \frac{e^{\sqrt{2}t}}{6}, \quad z(t) = \frac{\cos t}{3} - \frac{e^{-\sqrt{2}t}}{6} - \frac{e^{\sqrt{2}t}}{6}$$

例 7-6 试求解下面给出的时变微分方程

$$x^2(2x-1)y''' - (4x-3)xy'' - 2xy' + 2y = 0$$

解 这里给出的方程是关于 y 和 x 之间的方程, 不是默认意义下的 t , 所以在求解语句中应该给出 'x' 标识, 否则得出的解是错误的。即使此方程是时变微分方程, 仍然可以用常规的方法将微分方程用字符串描述出来, 这样可以直接得出该方程的解析解如下, 代入原方程误差为 0

$$y(x) = -\frac{C_1}{2} - x \left(2C_2 + \frac{C_1 \ln x}{2} \right) - \frac{1}{x} \left(\frac{C_3}{8} - \frac{C_1}{16} \right)$$

```
>> y=dsolve('x^2*(2*x-1)*D3y+(4*x-3)*x*D2y-2*x*Dy+2*y=0', 'x'); simple(y)
```

```
syms x; simple(x^2*(2*x-1)*diff(y,3)+(4*x-3)*x*diff(y,2)-2*x*diff(y)+2*y)
```

7.1.3 线性状态空间方程的解析解

假设线性状态空间模型的一般表示为

$$\begin{cases} \mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{cases} \quad (7-1-4)$$

其中 $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ 为常数矩阵, 且已知状态向量初值 \mathbf{x}_0 。该方程的解析解可以写成

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}(t_0) + \int_{t_0}^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau) d\tau \quad (7-1-5)$$

对给定的输入信号 $\mathbf{u}(t)$ 来说, 通过矩阵积分运算可以直接得出该方程的解析解。从解析解表达式可见, 涉及矩阵的 e 指数求解, 也涉及函数的积分运算, 这些分别调用 MATLAB 的符号运算函数可以直接求解。

例 7-7 假设输入信号为 $u(t) = 2 + 2e^{-3t} \sin 2t$, 求出下面矩阵描述的状态空间方程的解析解

$$\mathbf{A} = \begin{bmatrix} -19 & -16 & -16 & -19 \\ 21 & 16 & 17 & 19 \\ 20 & 17 & 16 & 20 \\ -20 & -16 & -16 & -19 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 2 \end{bmatrix}, \quad \mathbf{C}^T = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{D} = 0, \quad \mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

解 由式 (7-1-5) 可以直接得出方程的解析解

```
>> syms t tau; u=2+2*exp(-3*t)*sin(2*t); u1=subs(u,t,tau);
A=[-19,-16,-16,-19; 21,16,17,19; 20,17,16,20; -20,-16,-16,-19];
B=[1; 0; 1; 2]; C=[2 1 0 0]; x0=[0; 1; 1; 2];
y=C*(expm(A*t)*x0+int(expm(A*(t-tau))*B*u1,tau,0,t)); simple(y)
```

这样得出的解析解数学表示为

$$y(t) = \frac{119}{8}e^{-t} + 57e^{-3t} + \frac{127t}{4}e^{-t} + 4t^2e^{-t} - \frac{135}{8}e^{-3t} \cos 2t + \frac{77}{4}e^{-3t} \sin 2t - 54$$

7.1.4 特殊非线性微分方程的解析解

部分非线性微分方程也是可以用 `dsolve()` 函数求解析解的, 这样的方程描述方式和前面介绍的线性微分方程是一致的, 描述了这样的微分方程, 则可以直接求解出微分方程的解析解。下面将通过例子演示非线性方程的解析解求解, 同时还将演示不能求解的例子。

例 7-8 试求出一阶非线性微分方程 $x'(t) = x(t)(1 - x^2(t))$ 的解析解。

解 这样简单的一阶非线性方程可以考虑用 `dsolve()` 函数直接解出

```
>> syms t x; x=dsolve('Dx=x*(1-x^2)')
```

即该微分方程的解析解为 $x(t) = \sqrt{-\frac{1}{e^{C-2t}-1}}$, 此外常数 ± 1 与 0 均为方程的解。

其实, 稍微改变原微分方程, 例如将等号右侧加上 1 , 则可以用下面的语句试解该方程。读者会发现原始的微分方程是没有解析解的

```
>> syms t x; x=dsolve('Dx=x*(1-x^2)+1')
```

上述语句将得出警告信息显示“Warning: Explicit solution could not be found; implicit solution returned”,说明该方程的解析解是不存在的。

例 7-9 考虑著名的 Van der Pol 方程

$$\frac{d^2 y(t)}{dt^2} + \mu(y^2(t) - 1) \frac{dy(t)}{dt} + y(t) = 0 \quad (7-1-6)$$

试用 `dsolve()` 函数求解它,看看能得出什么结论。

解 由前面的讨论可见,似乎所有的微分方程都可以直接用 MATLAB 语言提供的强大的 `dsolve()` 函数求解,这样很自然地想到一般非线性微分方程的解析解问题。

对前面给出的 Van der Pol 方程,用户尝试如下的 MATLAB 命令

```
>> syms y mu; y=dsolve('D2y+mu*(y^2-1)*Dy+y')
```

该函数将得出原微分方程无解析解的提示“Warning: Explicit solution could not be found”。

可见,微分方程解析解求解函数 `dsolve()` 并不能直接应用于一般非线性方程解析解的求解。所以非线性微分方程只能用数值解法求解,即使看起来很简单的非线性微分方程也是没有解析解的,只有极特殊的非线性微分方程解析可解。下面的内容将集中介绍各类非线性微分方程的数值解方法。

7.2 微分方程问题的数值解法

前面介绍了微分方程的解析解方法,同时也指出很多非线性微分方程是不存在解析解的,需要使用数值解法对之进行研究。从本节开始着重讨论基于 MATLAB/Simulink 语言的各类微分方程的数值解方法。

7.2.1 微分方程问题算法概述

一般微分方程的数值解法很大一类是关于微分方程初值问题的数值解法,这类问题需要用一阶显式的微分方程组描述为

$$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t)) \quad (7-2-1)$$

其中, $\mathbf{x}^T(t) = [x_1(t), x_2(t), \dots, x_n(t)]$ 称为状态向量, $\mathbf{f}^T(\cdot) = [f_1(\cdot), f_2(\cdot), \dots, f_n(\cdot)]$ 可以是任意非线性函数。所谓初值问题是指,若已知初始状态 $\mathbf{x}_0 = [x_1(t_0), x_2(t_0), \dots, x_n(t_0)]^T$, 用数值求解方法求出在某个时间区间 $t \in [t_0, t_f]$ 内各个时刻状态变量 $\mathbf{x}(t)$ 的数值,这里 t_f 又称为终止时间。

对多元非线性常微分方程初值问题来说, Euler 算法是最直观的一类求解算法。虽然该算法比较简单,但理解该算法对理解其他复杂的微分方程算法是很有帮助的,故这里将以 Euler 算法为例介绍微分方程初值问题的数值算法。

假设已知在 t_0 时刻系统状态向量的初值为 $\mathbf{x}(t_0)$, 若选择一个很小的计算步长 h , 则可以将微分方程左侧的导数近似为 $(\mathbf{x}(t_0 + h) - \mathbf{x}(t_0))/(t_0 + h - t_0)$, 代入微分方程则可以解

出在 $t_0 + h$ 时刻方程的近似解为

$$\hat{\mathbf{x}}(t_0 + h) = \mathbf{x}(t_0) + h\mathbf{f}(t_0, \mathbf{x}(t_0)) \quad (7-2-2)$$

更严格地, 因为这样的近似解存在误差, 所以可以写出在 $t_0 + h$ 时刻系统状态向量的值为

$$\mathbf{x}(t_0 + h) = \hat{\mathbf{x}}(t_0 + h) + \mathbf{R}_0 = \mathbf{x}(t_0) + h\mathbf{f}(t_0, \mathbf{x}(t_0)) + \mathbf{R}_0 \quad (7-2-3)$$

简记 $\mathbf{x}_1 = \mathbf{x}(t_0 + h)$, 则 $\hat{\mathbf{x}}_1 = \hat{\mathbf{x}}(t_0 + h)$ 为系统状态向量在 $t_0 + h$ 时刻的近似值, 亦即数值解。可见, \mathbf{R}_0 为数值解的舍入误差。在实际解法中为简单起见, 经常可以舍弃 $\hat{}$ 记号, 而将数值解直接记为 \mathbf{x}_1 。

一般地, 假设已知在 t_k 时刻系统的状态向量为 \mathbf{x}_k , 则在 $t_k + h$ 时刻 Euler 算法的数值解可以写成

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\mathbf{f}(t_k, \mathbf{x}_k) \quad (7-2-4)$$

这样, 用迭代的方法可以由给定的初值问题逐步求出在所选择的时间段 $t \in [0, T]$ 内各个时刻 $t_0 + h, t_0 + 2h, \dots$ 处的原问题数值解。

提高数值解精度的一种方法是减小步长 h 的值。然而, 并不能无限制地减小 h 的值, 这主要有两个原因:

(1) **减慢计算速度**。因为对选定的求解时间而言, 减小步长就意味着增加在这个时间段内的计算点数目, 故计算速度减慢。

(2) **增加累积误差**。因为不论选择多小的步长, 所得出的数值解都将有一个舍入误差, 减小计算步长则将增加计算的次数, 从而使得整个计算过程的舍入误差的叠加和传递次数增多, 产生较大的累积误差。舍入误差、累积误差和总误差关系的示意图如图 7-1(a) 所示。

所以在对微分方程求解过程中, 应采取下列措施:

(1) **选择适当的步长**。采用像 Euler 法这样简单的算法时, 应适当地选择步长, 既不能太大, 又不能太小。

(2) **改进近似算法精度**。由于 Euler 算法只是将原积分问题进行梯形近似, 其近似精度很低, 因而不能很有效地逼近原始问题。可以用各种更精确的插值方法来取代 Euler 算法, 从而改进运算精度。比较成功的是 Runge-Kutta 法、Adams 法等。

(3) **采用变步长方法**。前面提及“适当”地选择步长, 这本身就是个模糊的概念, 如何适当地选择步长取决于经验。事实上, 很多种方法都允许变步长的求解, 如果误差较小时, 可自动地增大步长, 而误差较大时再自动减小步长, 从而精确、有效地求解给出的常微分方程初值问题。

一般变步长算法的原理如图 7-1(b) 所示。已知 t_k 时刻的状态变量 \mathbf{x}_k , 则先在某步长 h 下计算出 $t_k + h$ 时刻的状态变量 $\tilde{\mathbf{x}}_{k+1}$ 。另外, 将步长变成原来步长的一半, 分两步从 \mathbf{x}_k 计算出 $t_k + h$ 时刻的状态变量 $\hat{\mathbf{x}}_{k+1}$ 。如果两种运算步长下的误差 $\epsilon = \|\hat{\mathbf{x}}_{k+1} - \tilde{\mathbf{x}}_{k+1}\|$ 小于给定的误差限, 则可以采用该步长或适当增大步长, 如果误差大, 则进一步减小步长。自适应变步长算法可以较好地解决计算速度问题, 另外能保证计算的精度。

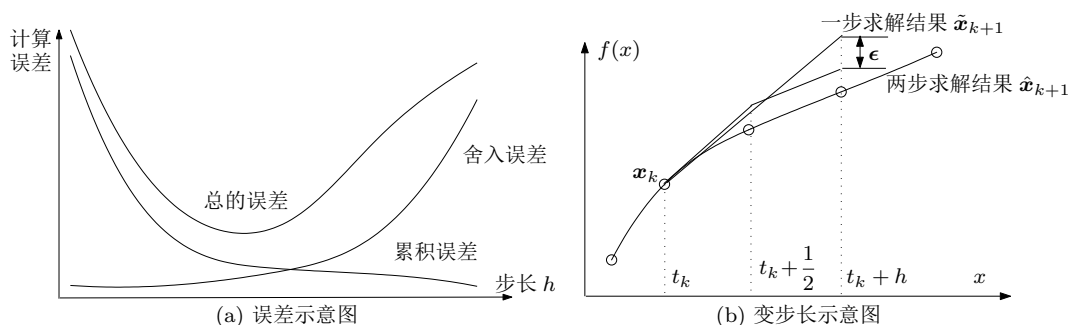


图 7-1 误差及步长

7.2.2 四阶定步长 Runge-Kutta 算法及 MATLAB 实现

四阶定步长的 Runge-Kutta 算法是传统数值分析课程和系统仿真课程中经常介绍的算法,被认为是求解微分方程的一种有效的方法,该算法结构很简单,可以先定义如下 4 个附加向量

$$\begin{cases} k_1 = hf(t_k, x_k) \\ k_2 = hf\left(t_k + \frac{h}{2}, x_k + \frac{k_1}{2}\right) \\ k_3 = hf\left(t_k + \frac{h}{2}, x_k + \frac{k_2}{2}\right) \\ k_4 = hf(t_k + h, x_k + k_3) \end{cases} \quad (7-2-5)$$

其中, h 为计算步长,在实际应用中该步长是一个常数,这样由四阶 Runge-Kutta 算法可以求解出下一个步长的状态变量值为

$$x_{k+1} = x_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (7-2-6)$$

这样,用迭代的方法由给定的初值问题逐步求出在所选择的时间段 $t \in [t_0, t_f]$ 内各个时刻 $t_0 + h, t_0 + 2h, \dots$ 处的原问题数值解。

有了上面的数学算法,则可以用 MATLAB 语言容易地编写出该算法的函数如下

```
function [tout,yout]=rk_4(odefile,tspan,y0)
ts=tspan; t0=ts(1); tf=ts(2); yout=[]; tout=[]; y0=y0(:);
if length(ts)==3, h=ts(3); else, h=(ts(2)-ts(1))/100; tf=ts(2); end
for t=[t0:h:tf]
    k1=h*feval(odefile,t,y0); k2=h*feval(odefile,t+h/2,y0+0.5*k1);
    k3=h*feval(odefile,t+h/2,y0+0.5*k2); k4=h*feval(odefile,t+h,y0+k3);
    y0=y0+(k1+2*k2+2*k3+k4)/6; yout=[yout; y0']; tout=[tout; t];
end
```

其中, $tspan$ 可以有两种构成方法,第一种方法可以是一个等间距的时间向量;第二种方法是 $tspan = [t_0, t_f, h]$, 其中, t_0 和 t_f 为计算的初始和终止值, h 为计算步长, $odefile$ 是一

个字符串变量,为表示微分方程的文件名, \mathbf{y}_0 是初值列向量。函数调用完成后,时间向量与各个时刻状态变量构成的矩阵分别由 `tout` 和 `yout` 返回。

从求解数值问题的效果看,该算法不是一个较好的方法,故一般不使用该方法,后面将通过例子演示微分方程求解方法,并和现成的 MATLAB 函数进行对比分析。

7.2.3 一阶微分方程组的数值解

1. 四阶五级 Runge–Kutta–Fehlberg 算法

德国学者 Fehlberg 对传统的 Runge–Kutta 方法进行了改进^[1],在每一个计算步长内对 $f_i(\cdot)$ 函数进行 6 次求值,以保证更高的精度和数值稳定性,该算法又称为四阶五级 RKF 算法。假设当前的步长为 h_k ,则可以定义下面的 6 个 \mathbf{k}_i 变量

$$\mathbf{k}_i = h_k \mathbf{f} \left(t_k + \alpha_i h_k, \mathbf{x}_k + \sum_{j=1}^{i-1} \beta_{ij} \mathbf{k}_j \right), \quad i = 1, 2, \dots, 6 \quad (7-2-7)$$

式中, t_k 为当前计算时刻,中间参数 α_i , β_{ij} 及其他参数由表 7-1 给出,其中, α_i , β_{ij} 参数对又称为 Dormand–Prince 对。这时下一步的状态向量可以由下式求出

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=1}^6 \gamma_i \mathbf{k}_i \quad (7-2-8)$$

表 7-1 四阶五级 RKF 算法系数表

α_i	β_{ij}				γ_i	γ_i^*
0					16/135	25/216
1/4	1/4				0	0
3/8	3/32	9/32			6656/12825	1408/2565
12/13	1932/2197	-7200/2197	7296/2197		28561/56430	2197/4104
1	439/216	-8	3680/513	-845/4104	-9/50	-1/5
1/2	-8/27	2	-3544/2565	1859/4104	-11/40	2/55

当然,直接采用这一方法为定步长的方法。在实际问题中往往希望在一些情况下(如解变化很快时)采用较小的步长,而在另一些情况下(如解的变化很缓慢时)采用较大的步长,这样做既可以保证较高的精度,又可以保证较高的运算速度。在此算法中,定义误差向量

$$\boldsymbol{\epsilon}_k = \sum_{i=1}^6 (\gamma_i - \gamma_i^*) \mathbf{k}_i \quad (7-2-9)$$

可以由其大小改变计算步长,所以这种能自动变换步长的方法又称为自适应变步长算法。

定步长算法相当于用开环控制的思想求解微分方程,选定步长后将不考虑计算是不是有误差,也不考虑误差是否能被接受,一味采用单一步长计算全程。而变步长算法则是采用由误差作为监测指标的闭环控制思想,在计算过程中修正步长,使得预期的计算精度得以保

证,同时由于采用变步长思想,所以大部分问题的求解时间将明显缩短,因为在计算过程中在保证计算精度的前提下也允许大步长。

2. 基于 MATLAB 的微分方程求解函数

MATLAB 下求解一阶微分方程组初值问题数值解最常用的方法是 `ode45()` 函数,该函数实现了前面介绍的变步长四阶五级 Runge–Kutta–Fehlberg 算法,可以采用变步长的算法求解微分方程。该函数的调用格式为

```
[t,x] = ode45(Fun,[t0,tf],x0) % 直接求解
[t,x] = ode45(Fun,[t0,tf],x0,options) % 带有控制选项
[t,x] = ode45(Fun,[t0,tf],x0,options,p1,p2,...,pm) % 带有附加参数
```

其中,微分方程应该用 MATLAB 函数 `Fun` 或匿名函数按指定的格式描述,有关该函数的编写方法后面将通过例子专门介绍, $[t_0,t_f]$ 描述微分方程求解的区间,如果只给出一个值 t_f 则表示初始时刻为 $t_0 = 0$,终止时刻为 t_f 的问题求解。为使得微分方程能够求解,还应该已知初值问题的初始状态变量 \mathbf{x}_0 。另外,该函数还允许 $t_f < t_0$,即可以认为 t_0 为终值时刻, t_f 为初始时刻, \mathbf{x}_0 表示状态变量的终值,故该函数可以直接求解终值问题。

求解一阶显式微分方程组的关键是用 MATLAB 语言编写一个函数,描述需要求解的微分方程组。该函数的入口应该为

```
function xd = funname(t,x) % 不需附加参数的格式
function xd = funname(t,x,p1,p2,...,pm) % 可以使用附加参数
```

其中, t 是时间变量或自变量,即使需要求解的微分方程不是时变的,也需要给出 t 占位,否则 MATLAB 在变量传递过程中将出现问题。变量 \mathbf{x} 为状态向量,返回的 \mathbf{x}_d 为状态向量的导数。该函数允许求解带有附加变量 p_1,p_2,\cdots,p_m 的微分方程,这里的附加变量必须与微分方程求解程序中的完全对应。

在微分方程求解中有时需要对求解算法及控制条件进行进一步设置,这可以通过求解过程中的 `options` 变量进行修改。初始 `options` 变量可以通过 `odeset()` 函数获取,该变量是一个结构体变量,其中有众多成员变量,表 7-2 中列出了常用的一些成员变量。修改该变量有两种方式,其一是用 `odeset()` 函数设置,其二是直接修改 `options` 的成员变量。例如,若想将相对误差设置成较小的 10^{-7} ,则需要给出

表 7-2 常微分方程求解函数的控制参数表

参数名	参 数 说 明
RelTol	为相对误差容许上限,默认值为 0.001 (即 0.1% 的相对误差),在一些特殊的微分方程求解中,为了保证较高的精度,还应该再适当减小该值
AbsTol	为一个向量,其分量表示每个状态变量允许的绝对误差,其默认值为 10^{-6} 。当然可以自由设置其值,以改变求解精度
MaxStep	为求解方程最大允许的步长
Mass	微分代数方程中的质量矩阵,可以用于描述微分代数方程
Jacobian	为描述 Jacobi 矩阵函数 $\partial \mathbf{f} / \partial \mathbf{x}$ 的函数名,如果已知该 Jacobi 矩阵,则能加速仿真过程

```
options = odeset('RelTol',1e-7);           % 或更直观地
options = odeset; options.RelTol = 1e-7;
```

在实际求解过程中经常需要定义一些附加参数,这些参数由 p_1, p_2, \dots, p_m 表示,在编写方程函数时也应该一一对应地写出。后面将通过例子详细介绍相关的调用格式。

例 7-10 假设著名的 Lorenz 模型的状态方程表示为

$$\begin{cases} x_1'(t) = -\beta x_1(t) + x_2(t)x_3(t) \\ x_2'(t) = -\rho x_2(t) + \rho x_3(t) \\ x_3'(t) = -x_1(t)x_2(t) + \sigma x_2(t) - x_3(t) \end{cases}$$

其中,设 $\beta = 8/3, \rho = 10, \sigma = 28$ 。若其初值为 $x_1(0) = x_2(0) = 0, x_3(0) = \epsilon$, 而 ϵ 为机器上可以识别的小常数,如取一个很小的正数 $\epsilon = 10^{-10}$, 试求解该微分方程组。

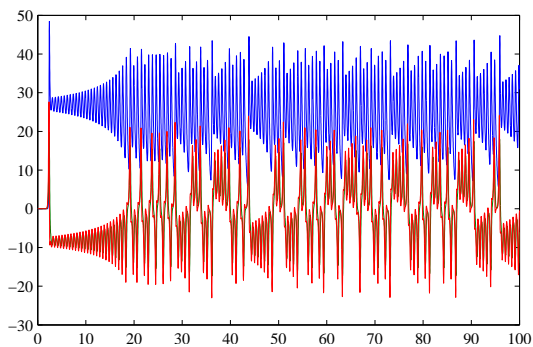
解 该方程是非线性微分方程,所以不存在解析解,只能用数值解法求解。若想用数值算法求解该微分方程,可以按下面的格式编写出一个匿名函数来描述系统的动态模型。其内容为

```
>> f=@(t,x)[-8/3*x(1)+x(2)*x(3); -10*x(2)+10*x(3); -x(1)*x(2)+28*x(2)-x(3)];
```

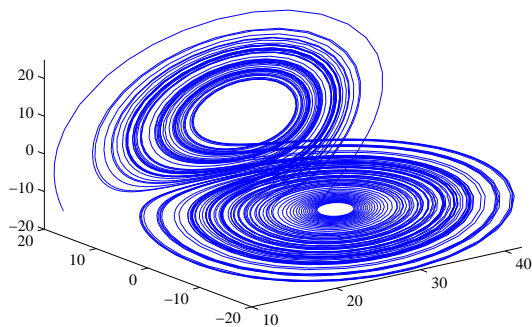
这时,可以调用微分方程数值解 `ode45()` 函数对匿名函数 f 描述的系统进行数值求解,并将结果进行图形显示

```
>> t_final=100; x0=[0;0;1e-10]; [t,x]=ode45(f,[0,t_final],x0); plot(t,x)
figure; plot3(x(:,1),x(:,2),x(:,3));
axis([10 42 -20 20 -20 25]); % 根据实际数值手动设置坐标系
```

其中, `t_final` 为设定的仿真终止时间, x_0 为初始状态。第一个绘图命令绘制出系统的各个状态和时间关系的二维曲线图,如图 7-2(a) 所示;第二个绘图命令可以绘制出三个状态的相空间曲线,如图 7-2(b) 所示。可以看出,看似很复杂的三元一阶非线性常微分方程组的数值解问题由几条简单直观的 MATLAB 语句就求解出来了。此外,用 MATLAB 语言还可以轻易、直观地将结果用可视化方式直接显示出来,这就是选择 MATLAB 语言作为本书主要语言的原因。



(a) 状态变量的时间响应图



(b) 相空间三维图

图 7-2 Lorenz 方程的仿真结果图示

其实,观察相空间轨迹走行的最好方法是采用 `comet3()` 函数绘制动画式的轨迹,故可将最后一个语句改成 `comet3(x(:,1),x(:,2),x(:,3))`。

从前面的微分方程求解实例看,如果能建立一个描述微分方程组的 M-函数或匿名函数,则调用 `ode45()` 可以立即解出方程的数值解。可以看出,编写一个 MATLAB 函数来描

述微分方程是常微分方程初值问题数值求解的关键。

3. MATLAB 下带有附加参数的微分方程求解

在基于 MATLAB 语言的微分方程求解中,引入附加参数的目的是,若微分方程的某些参数可以选择不同的值,对不同值求解时考虑附加参数可以避免每次修改模型文件。例如,例 7-10 中给出的 Lorenz 方程中, β, ρ, σ 可以看成附加参数,这样在表示它们的变化时,无需修改描述原始微分方程的 MATLAB 函数,而只需在调用微分方程求解时从外部修改它们的参数即可。

例 7-11 试编写带有附加参数的 MATLAB 函数来描述例 7-10 中的 Lorenz 方程,并利用该函数研究分别求解在该例中给定参数下和一组新参数 $\beta = 2, \rho = 5, \sigma = 20$ 下 Lorenz 方程的数值解。

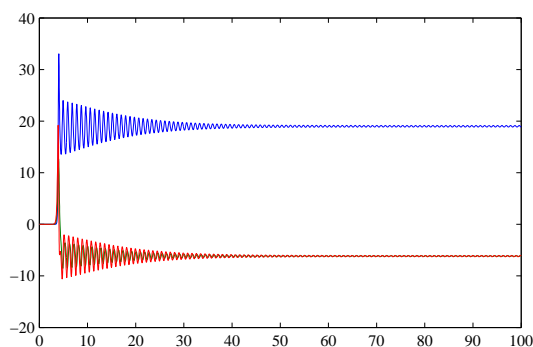
解 选定附加参数为 β, ρ, σ , 根据上述的微分方程描述格式,可以编写出如下的 MATLAB 函数来描述给出的常微分方程组

```
function dx=lorenz1(t,x,b,r,s)
dx=[-b*x(1)+x(2)*x(3); -r*x(2)+r*x(3); -x(1)*x(2)+s*x(2)-x(3)];
```

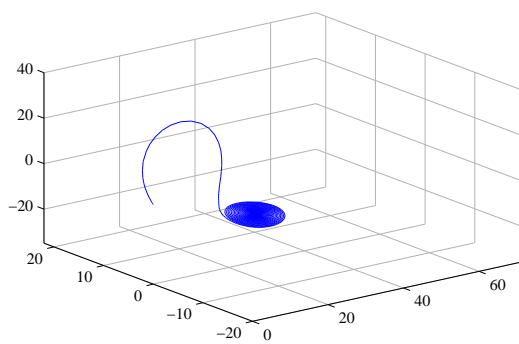
然后求取方程的数值解。从下面的调用格式可以看出,调用函数时无需使用和函数本身完全一致的变量名,只要对应关系正确就可以了。在 ode45() 语句中,空矩阵表示使用默认的控制模板。

```
>> b1=8/3; r1=10; s1=28; t_final=100; x0=[0;0;1e-10];
[t,x]=ode45(@lorenz1,[0,t_final],x0,[],b1,r1,s1); plot(t,x)
figure; plot3(x(:,1),x(:,2),x(:,3)); axis([10 42 -20 20 -20 25]);
```

有了带有附加参数的微分方程 M-函数后,就可以在其他参数值 β, ρ, σ 下求解微分方程,而无需改变 lorenz1.m 文件。例如,若选择 $\beta = 2, \rho = 5, \sigma = 20$,则可以用下面的语句直接求出数值解,这样将分别得出如图 7-3(a)、(b) 所示的二维和三维图形。



(a) 状态变量的时间响应图



(b) 相空间三维图

图 7-3 新参数下 Lorenz 方程的仿真结果图示

```
>> t_final=100; x0=[0;0;1e-10]; b2=2; r2=5; s2=20;
[t2,x2]=ode45(@lorenz1,[0,t_final],x0,[],b2,r2,s2); plot(t2,x2)
figure; plot3(x2(:,1),x2(:,2),x2(:,3)); axis([0 72 -20 22 -35 40]);
```

如果微分方程比较简单,则可以采用匿名函数的形式描述该方程,这时无需使用附加变量的形式,因为匿名函数可以直接使用 MATLAB 工作空间中的变量,这时,求解语句变为

```
>> b=8/3; r=10; s=28;
f=@(t,x)[-b*x(1)+x(2)*x(3); -r*x(2)+r*x(3); -x(1)*x(2)+s*x(2)-x(3)];
[t,x]=ode45(f,[0,t_final],x0);
```

7.2.4 微分方程数值解的验证

前面通过例子曾演示过,若仿真算法和控制参数选择不当,如相对误差限,则可能得出不可信的结果,甚至是错误的结果。所以在方程求解结束之后,应该对仿真结果进行检验。然而所求解的问题又不存在解析解,怎么检验所得结果的正确性呢?一种可行的方法是修改仿真控制参数,如可以接受的误差限,例如将 `RelTol` 或 `AbsTol` 选项设置成一个更小的值,观察所得的结果,看看是不是和上次得出的结果完全一致,如果存在不能接受的差异,则应该考虑再进一步减小误差限。另外,同样的问题选择不同的微分方程求解算法也可以检验所得结果的正确性。

7.3 微分方程转换

由前面介绍的微分方程求解函数和微分方程标准型可见,如果常微分方程由一个或多个高阶常微分方程给出,要得出该方程的数值解,则应该先将该方程变换成一阶常微分方程组。这里将分以下几种情况加以考虑。

7.3.1 单个高阶常微分方程处理方法

假设一个高阶常微分方程的一般形式为

$$y^{(n)} = f(t, y, y', \dots, y^{(n-1)}) \quad (7-3-1)$$

且已知输出变量 $y(t)$ 的各阶导数初始值为 $y(0), y'(0), \dots, y^{(n-1)}(0)$, 则可以选择一组状态变量 $x_1 = y, x_2 = y', \dots, x_n = y^{(n-1)}$, 这样, 就可以将原高阶常微分方程模型变换成下面的一阶方程组形式

$$\begin{cases} x'_1 = x_2 \\ x'_2 = x_3 \\ \vdots \\ x'_n = f(t, x_1, x_2, \dots, x_n) \end{cases} \quad (7-3-2)$$

且初值 $x_1(0) = y(0), x_2(0) = y'(0), \dots, x_n(0) = y^{(n-1)}(0)$ 。这样, 变换以后就可以直接求取原方程的数值解了。

例 7-12 已知 $y(0) = -0.2, y'(0) = -0.7$, 试求 Van der Pol 方程 $y'' + \mu(y^2 - 1)y' + y = 0$ 的数值解, 并绘制出不同 μ 参数下相平面曲线。

解 例 7-9 中已经演示过, 该方程没有解析解, 所以不能用解析方法求解该方程, 只有依靠数值解法。因为该方程不是由 MATLAB 直接可解的一阶显式微分方程组给出的, 所以需要对该方程进行变换。选择状态变量 $x_1 = y, x_2 = y'$, 则原方程可以变换成

$$\begin{cases} x_1' = x_2 \\ x_2' = -\mu(x_1^2 - 1)x_2 - x_1 \end{cases}$$

这里的 μ 是一个可变参数,如果对每一个要研究的 μ 值都编写一个函数则显得不方便,所以应该采用附加参数的概念将 μ 的值传给该函数,这样可以写出描述此模型的 M-函数为

```
>> f=@(t,x,mu)[x(2); -mu*(x(1)^2-1)*x(2)-x(1)];
```

可见,在函数定义时多了一个 μ 项,该项应该在 `ode45()` 函数调用时传给匿名函数。假定初值为 $x = [-0.2, -0.7]^T$, 则最终的求解函数格式为

```
>> x0=[-0.2; -0.7]; t_final=20;
mu=1; [t1,y1]=ode45(f,[0,t_final],x0,[],mu);
mu=2; [t2,y2]=ode45(f,[0,t_final],x0,[],mu); plot(t1,y1,t2,y2,'--')
figure; plot(y1(:,1),y1(:,2),y2(:,1),y2(:,2),'--')
```

这样,在 $\mu = 1, 2$ 时的时间响应曲线和相平面曲线分别如图 7-4(a)、(b) 所示。

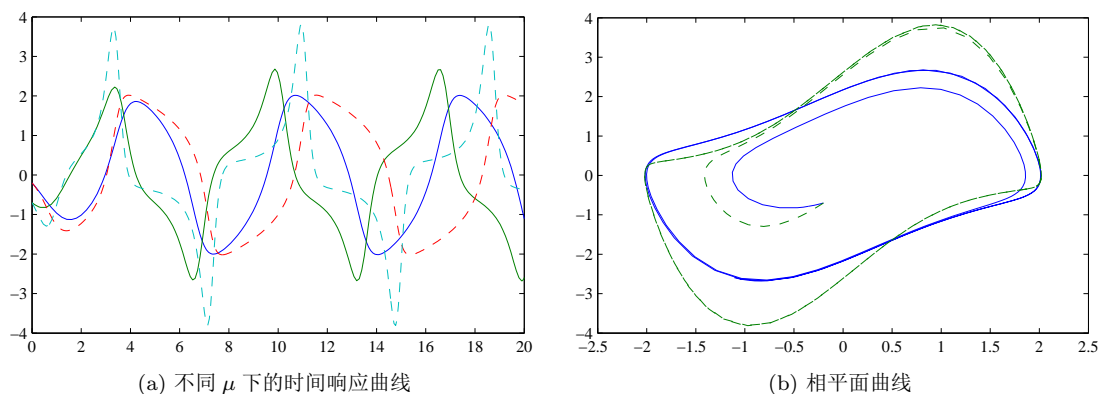


图 7-4 不同 μ 值下 Van der Pol 方程解

在 `ode45()` 调用命令中的附加参数个数应该和方程 M-函数中的附加参数个数完全对应,否则将出现错误结果。改变 μ 的值,令 $\mu = 1000$,并设仿真终止时间为 3000,则可以采用下面的命令求解相应的 Van der Pol 方程

```
>> tic, x0=[2;0]; t_final=3000;
mu=1000; [t,y]=ode45(f,[0,t_final],x0,[],mu); toc
```

经过长时间的等待,将得出错误信息“Out of memory. Type HELP MEMORY for your options”。事实上,由于变步长所采用的步长过小,而要求的仿真终止时间比较大,导致输出的 y 矩阵过大,超出了计算机存储空间的容限。所以,这个问题不适合采用 `ode45()` 来求解,后面将采用刚性方程求解的算法来解决这个问题。

7.3.2 高阶常微分方程组的变换方法

这里以两个高阶微分方程构成的微分方程组为例介绍如何将之变换成一个一阶显式

常微分方程组。如果可以显式地将两个方程写成

$$\begin{cases} x^{(m)} = f(t, x, x', \dots, x^{(m-1)}, y, y', \dots, y^{(n-1)}) \\ y^{(n)} = g(t, x, x', \dots, x^{(m-1)}, y, y', \dots, y^{(n-1)}) \end{cases} \quad (7-3-3)$$

则仍旧可以选择状态变量 $x_1 = x, x_2 = x', \dots, x_m = x^{(m-1)}, x_{m+1} = y, x_{m+2} = y', \dots, x_{m+n} = y^{(n-1)}$, 这样就可以将原方程变换成

$$\begin{cases} x'_1 = x_2 \\ \vdots \\ x'_m = f(t, x_1, x_2, \dots, x_{m+n}) \\ x'_{m+1} = x_{m+2} \\ \vdots \\ x'_{m+n} = g(t, x_1, x_2, \dots, x_{m+n}) \end{cases} \quad (7-3-4)$$

再对初值进行相应的变换, 就可以得出所期望的一阶微分方程组了。下面将通过一个例子演示常微分方程组的转换与求解。

例 7-13 已知 Apollo 卫星的运动轨迹 (x, y) 满足下面的方程

$$x'' = 2y' + x - \frac{\mu^*(x + \mu)}{r_1^3} - \frac{\mu(x - \mu^*)}{r_2^3}, \quad y'' = -2x' + y - \frac{\mu^*y}{r_1^3} - \frac{\mu y}{r_2^3}$$

其中, $\mu = 1/82.45$, $\mu^* = 1 - \mu$, $r_1 = \sqrt{(x + \mu)^2 + y^2}$, $r_2 = \sqrt{(x - \mu^*)^2 + y^2}$, 试在初值 $x(0) = 1.2$, $x'(0) = 0$, $y(0) = 0$, $y'(0) = -1.04935751$ 下进行求解, 并绘制出 Apollo 位置的 (x, y) 轨迹。

解 选择一组状态变量 $x_1 = x, x_2 = x', x_3 = y, x_4 = y'$, 这样就可以得出一阶常微分方程组为

$$\begin{cases} x'_1 = x_2 \\ x'_2 = 2x_4 + x_1 - \mu^*(x_1 + \mu)/r_1^3 - \mu(x_1 - \mu^*)/r_2^3 \\ x'_3 = x_4 \\ x'_4 = -2x_2 + x_3 - \mu^*x_3/r_1^3 - \mu x_3/r_2^3 \end{cases}$$

式中, $r_1 = \sqrt{(x_1 + \mu)^2 + x_3^2}$, $r_2 = \sqrt{(x_1 - \mu^*)^2 + x_3^2}$, 且 $\mu = 1/82.45, \mu^* = 1 - \mu$ 。

由于该模型需要首先计算中间变量 r_1, r_2 , 所以不适合使用匿名函数的形式描述, 只能用 M-函数的方式描述原方程

```
function dx=apolloeq(t,x)
mu=1/82.45; mu1=1-mu; r1=sqrt((x(1)+mu)^2+x(3)^2);
r2=sqrt((x(1)-mu1)^2+x(3)^2);
dx=[x(2); 2*x(4)+x(1)-mu1*(x(1)+mu)/r1^3-mu*(x(1)-mu1)/r2^3;
    x(4); -2*x(2)+x(3)-mu1*x(3)/r1^3-mu*x(3)/r2^3];
```

调用 ode45() 函数可以求出该方程的数值解

```
>> x0=[1.2; 0; 0; -1.04935751]; tic, [t,y]=ode45(@apolloeq,[0,20],x0); toc
length(t), plot(y(:,1),y(:,3))
```

得出的轨迹如图 7-5(a) 所示,通过计算共得出 689 个数据点。

其实,这样直接得出的 Apollo 轨道是不正确的,因为这时 `ode45()` 函数选择的默认精度控制 `RelTol` 设置得太大,从而导致较高的误差传递。可以减小该值,直至减小到 10^{-6} ,使用下面的语句进行仿真研究

```
>> options=odeset; options.RelTol=1e-6;
tic, [t1,y1]=ode45(@apolloeq,[0,20],x0,options); toc
length(t1), plot(y1(:,1),y1(:,3))
```

得出的轨迹如图 7-5(b) 所示,得出数据点 1873 个。可见,在新的默认精度下结果是完全不同的。这时,再进一步减小精度控制误差限也不会有太大的改进了。所以在仿真结束后有时有必要减小精度误差限 `RelTol`,看看得出的结果是否还相同,用这样的方法检验数值解的正确性。

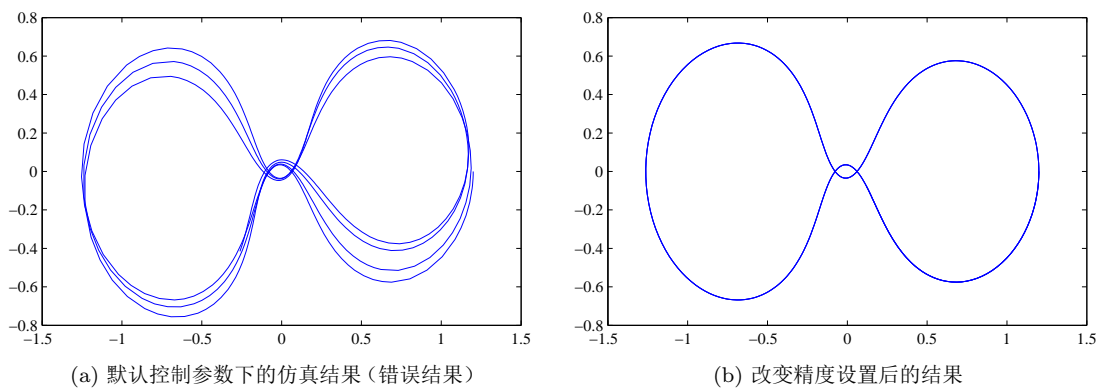


图 7-5 不同精度要求下绘制的 Apollo 轨迹图

用下面的 MATLAB 命令还求出求解全程所采用的最小步长的值为 1.8927×10^{-4} ,并可以绘制出计算步长的曲线,如图 7-6 所示。

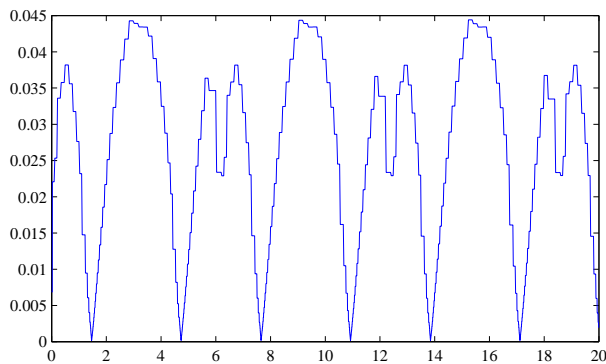


图 7-6 仿真过程中的计算步长

```
>> min(diff(t1)); plot(t1(1:end-1),diff(t1))
```

从得出的图形可以看出变步长算法的意义,即在需要小步长时取小步长,而变换缓慢时取大步长计算,这样就可以保证以高效率求解方程。

由给出的计算步长图可以看出,部分时间段用了大于 0.04 的步长,这是一般定步长计算问题求

解中不敢用的大步长。为了保证某些具体时间点上的计算精度,还会自动采用 2×10^{-4} 的小步长。换言之,在这些点上如果采用比该值大的步长,则计算误差就不能保证在 10^{-6} 之下。考虑定步长计算的方式,如果想保证 10^{-6} 之下的误差,全程选择的步长就应该是这样的值,这样计算的点就要达到 10^5 个,是变步长算法的 56 倍。

例 7-14 试用定步长的四阶 Runge-Kutta 算法求解 Apollo 微分方程。

解 用定步长的方法求解微分方程将面临两个问题:(1) 如何选择步长;(2) 如何确保求解的精度。前一个问题可以通过试凑的方法,从步长选择的角度看,选择很小的步长对保证求解精度有利,但计算量会明显增加。对前面介绍的 Apollo 轨迹方程,可以试着选择步长为 0.01,则用下面语句可以求解微分方程,并绘制出 Apollo 轨迹曲线,如图 7-7(a) 所示。所需求解时间为 2.7 s。

```
>> x0=[1.2; 0; 0; -1.04935751];
    tic, [t1,y1]=rk_4(@apolloeq,[0,20,0.01],x0); toc
    plot(y1(:,1),y1(:,3)) % 绘制出轨迹曲线
```

显而易见,这样求解的结果是错误的,应该采用更小的步长求解,直至选择步长为 0.001,则可以求解微分方程,并绘制出更精确的轨迹曲线,如图 7-7(b) 所示,但求解时间达到 97 s,是变步长算法的 138 倍。

```
>> tic, [t2,y2]=rk_4(@apolloeq,[0,20,0.001],x0); toc
    plot(y2(:,1),y2(:,3)) % 绘制出轨迹曲线
```

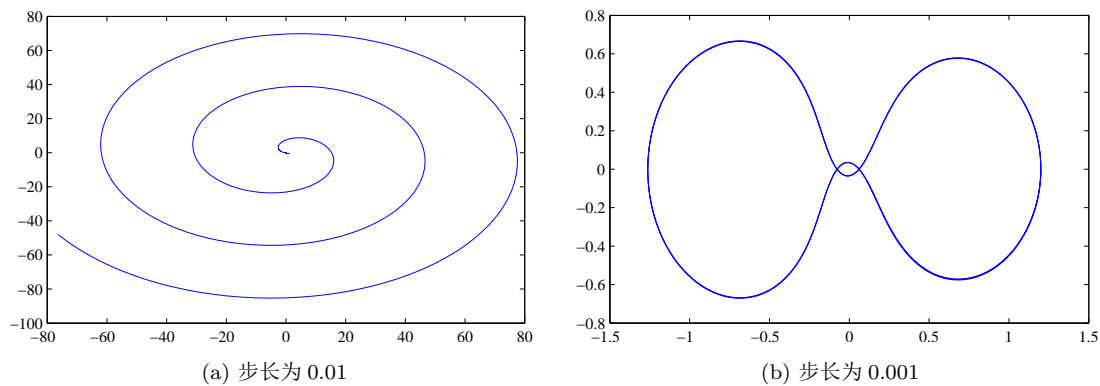


图 7-7 不同精度要求下绘制的 Apollo 轨迹图

其实,上面的结果在某些点上严格说来仍不能满足 10^{-6} 的误差限,只是形似变步长得出的结果,所以在求解常微分方程组时建议采用变步长算法,而没有必要自己按照数值分析类课程中介绍的定步长算法去编写程序求解。

如果两个高阶微分方程都同时隐式地含有 $x^{(m)}$ 和 $y^{(n)}$ 项,则首先需要对之进行相应的处理,然后再用上述方法进行最终变换。下面将通过一个例子加以说明。

例 7-15 假设系统模型以二元方程组形式如下给出,试将其转换成一阶微分方程组。

$$\begin{cases} x'' + 2y'x = 2y'' \\ x''y' + 3x'y'' + xy' - y = 5 \end{cases}$$

解 可见,这两个方程均同时含有 x'' 和 y'' ,所以仍可以选择一组状态变量 $x_1 = x, x_2 = x', x_3 =$

$y, x_4 = y'$, 我们的目的是先消去其中一个高阶导数, 求解第一个式子, 得出 y'' 的解析表达式为

$$y'' = y'x + \frac{x''}{2}$$

然后将它代入第二个式子中, 可以解出 x'' 为 $x'' = \frac{2y + 10 - 2xy' - 6xx'y'}{2y' + 3x'}$, 这样可以写出其状态

方程表示为 $x'_2 = \frac{2x_3 + 10 - 2x_1x_4 - 6x_1x_2x_4}{2x_4 + 3x_2}$, 将上面的结果再代入 y'' 的方程中, 得

$$x'_4 = \frac{x_3 + 5 - x_1x_4 + 2x_1x_4^2}{2x_4 + 3x_2}$$

综上所述, 可以列写出方程的一阶微分方程组表示为

$$\begin{cases} x'_1 = x_2 \\ x'_2 = \frac{2x_3 + 10 - 2x_1x_4 - 6x_1x_2x_4}{2x_4 + 3x_2} \\ x'_3 = x_4 \\ x'_4 = \frac{x_3 + 5 - x_1x_4 + 2x_1x_4^2}{2x_4 + 3x_2} \end{cases}$$

事实上, 这样的方程还是不太容易手工求解的, 但可以依赖 MATLAB 的符号运算工具箱来求解该问题。为了方便求解起见, 记 $dx=x''$, $dy=y''$, 这样, dx 和 dy 实际上还是 x'_2 和 x'_4 , 故可以用下面的语句得出方程的解, 这样可以直接求解出和前面完全一致的结果。

```
>> syms x1 x2 x3 x4
[dx,dy]=solve('dx+2*x4*x1=2*dy','dx*x4+3*x2*dy+x1*x4-x3=5','dx,dy')
```

对于更复杂的问题来说, 手工变换的难度将很大, 所以如果可能, 可以采用计算机去求解有关方程, 获得解析解。如果不能获得方程的解析解, 也需要在描写一阶常微分方程组时列写出式子, 得出问题的数值解。

7.3.3 矩阵微分方程的变换与求解方法

在实际应用中经常遇到矩阵形式的微分方程模型, 如在机器人等学科中的 Lagrange 方程, 对应的微分方程为下面的矩阵微分方程

$$MX'' + CX' + KX = Fu(t) \quad (7-3-5)$$

其中 M, C, K 为 $n \times n$ 矩阵, X, F 均为 $n \times 1$ 列向量。引入 $x_1 = X, x_2 = X'$, 则 $x'_1 = x_2, x'_2 = X''$ 。由式 (7-3-5) 可见, $X'' = M^{-1}[Fu(t) - CX' - KX]$ 。这样选择状态变量 $x = [x_1^T, x_2^T]^T$, 则可以写出系统的状态方程模型

$$x'(t) = \begin{bmatrix} x_2(t) \\ M^{-1}[Fu(t) - Cx_2(t) - Kx_1(t)] \end{bmatrix} \quad (7-3-6)$$

可见, 该微分方程是关于 $x(t)$ 列向量的显式一阶微分方程组, 所以可以通过 MATLAB 提供的函数直接求解。下面将通过例子给出问题的求解方法。

例 7-16 已知二级倒立摆系统的数学模型由下式给出^[2]

$$\mathbf{M}(\boldsymbol{\theta})\boldsymbol{\theta}'' + \mathbf{C}(\boldsymbol{\theta}, \boldsymbol{\theta}')\boldsymbol{\theta}' = \mathbf{F}(\boldsymbol{\theta})$$

其中 $\boldsymbol{\theta} = [a, \theta_1, \theta_2]^T$, 且 a 为小车位置, θ_1 、 θ_2 分别为下摆杆、上摆杆与垂直方向的夹角, 倒立摆系统的各个矩阵为

$$\mathbf{M}(\boldsymbol{\theta}) = \begin{bmatrix} m_c + m_1 + m_2 & (0.5m_1 + m_2)L_1 \cos \theta_1 & 0.5m_2 L_2 \cos \theta_2 \\ (0.5m_1 + m_2)L_1 \cos \theta_1 & (m_1/3 + m_2)L_1^2 & 0.5m_2 L_1 L_2 \cos \theta_1 \\ 0.5m_2 L_2 \cos \theta_2 & 0.5m_2 L_1 L_2 \cos \theta_1 & m_2 L_2^2/3 \end{bmatrix}$$

$$\mathbf{C}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \begin{bmatrix} 0 & -(0.5m_1 + m_2)L_1\theta'_1 \sin \theta_1 & -0.5m_2 L_2\theta'_2 \sin \theta_2 \\ 0 & 0 & 0.5m_2 L_1 L_2\theta'_2 \sin(\theta_1 - \theta_2) \\ 0 & -0.5m_2 L_1 L_2\theta'_1 \sin(\theta_1 - \theta_2) & 0 \end{bmatrix}$$

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{bmatrix} u(t) \\ (0.5m_1 + m_2)L_1 g \sin \theta_1 \\ 0.5m_2 L_2 g \sin \theta_2 \end{bmatrix}$$

已知二级倒立摆的参数为 $m_c = 0.85\text{kg}$, $m_1 = 0.04\text{kg}$, $m_2 = 0.14\text{kg}$, $L_1 = 0.1524\text{m}$, $L_2 = 0.4318\text{m}$, 试用数值方法求解系统的阶跃响应。

解 可见, 因为系数矩阵 $\mathbf{M}(\theta_1, \theta_2)$ 、 $\mathbf{C}(\theta_1, \theta_2)$ 和 $\mathbf{F}(\theta_1, \theta_2)$ 都含有状态变量 \mathbf{x} 的非线性项, 如 θ_1 的正弦余弦项等, 所以原来的系统是非线性微分方程。引入附加参数 $\mathbf{x}_1 = \boldsymbol{\theta}$, $\mathbf{x}_2 = \boldsymbol{\theta}'$, 并构造状态变量 $\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T]^T$, 则可以用下面的语句描述上述的一阶显式微分方程模型

```
function dx=inv_pendulum(t,x,u,mc,m1,m2,L1,L2,g)
M=[mc+m1+m2, (0.5*m1+m2)*L1*cos(x(2)), 0.5*m2*L2*cos(x(3))
(0.5*m1+m2)*L1*cos(x(2)), (m1/3+m2)*L1^2, 0.5*m2*L1*L2*cos(x(2))
0.5*m2*L2*cos(x(3)), 0.5*m2*L1*L2*cos(x(2)), m2*L2^2/3];
C=[0, -(0.5*m1+m2)*L1*cos(x(5))*sin(x(2)), -0.5*m2*L2*x(6)*sin(x(3))
0, 0, 0.5*m2*L1*L2*x(6)*sin(x(2)-x(3))
0, -0.5*m2*L1*L2*x(5)*sin(x(2)-x(3)), 0];
F=[u; (0.5*m1+m2)*L1*g*sin(x(2)); 0.5*m2*L2*g*sin(x(3))];
dx=[x(4:6); inv(M)*(F-C*x(4:6))];
```

若用阶跃信号激励该系统, 则可以由下面的语句直接求出方程的数值解, 如图 7-8 所示

```
>> opt=odeset; opt.RelTol=1e-8; u=1; mc=0.85;
m1=0.04; m2=0.14; L1=0.1524; L2=0.4318; g=9.81;
[t,x]=ode45(@inv_pendulum,[0,0.5],zeros(6,1),opt,u,mc,m1,m2,L1,L2,g);
plot(t,x(:,1:3)), figure; plot(t,x(:,4:6))
```

值得指出的是, 由于二级倒立摆系统是自然不稳定系统, 所以若给系统施加阶跃输入是没有任何意义的, 应该给其施加某些控制才能使其到达稳定状态。

另外, 如果各个矩阵 \mathbf{M} 、 \mathbf{C} 、 \mathbf{K} 与 \mathbf{F} 均和 \mathbf{X} 无关, 则该微分方程为线性微分方程, 还可以通过简单的变换将其改写为相应的线性状态方程模型为

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{F} \end{bmatrix} u(t) \quad (7-3-7)$$

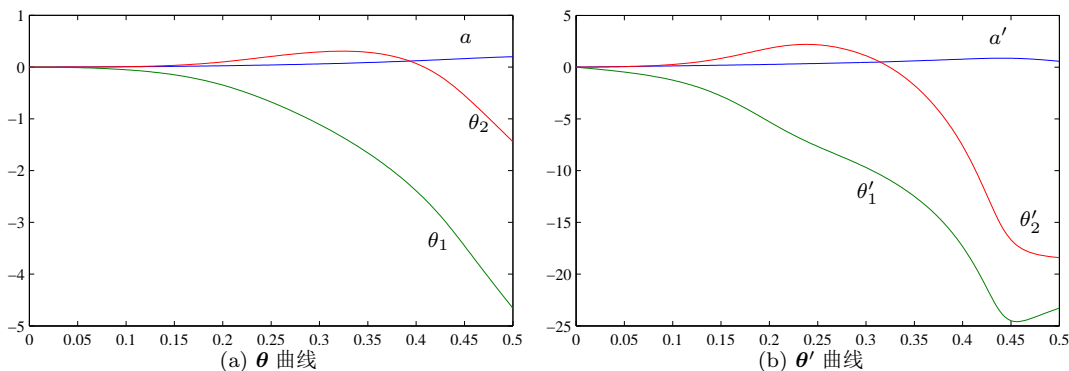


图 7-8 二级倒立摆的阶跃响应曲线

Riccati 微分方程是另一类常见的矩阵微分方程,其一般形式为

$$\mathbf{P}'(t) = \mathbf{A}^T \mathbf{P}(t) + \mathbf{P}(t) \mathbf{A} + \mathbf{P}(t) \mathbf{B} \mathbf{P}(t) + \mathbf{C} \quad (7-3-8)$$

其中 \mathbf{B} 、 \mathbf{C} 为对称矩阵。已知该微分方程在某一个时刻 t_f 的值 $\mathbf{P}(t_f)$, 要求出在时间段 (t_0, t_f) 内的数值解。求解这样的方程同样需要转换成向量型一阶显式微分方程组, 然后进行求解。这就需要向量和矩阵的相互转换, 可以由 `reshape()` 函数将向量转换成矩阵, 或由 `A(:)` 将矩阵展成列向量。可以编写一个通用的描述 Riccati 微分方程的 MATLAB 函数

```
function dy=ric_de(t,x,A,B,C)
P=reshape(x,size(A)); Y=A'*P+P*A+P*B*P+C; dy=Y(:);
```

这样用下面的语句就可以调用 `ode45()` 函数来求解 Riccati 微分方程。注意, 在微分方程求解函数 (如 `ode45()`) 调用语句中, 允许终止时间小于起始时间。

```
[t,p] = ode45(@ric_de,[t1,0],P1(:),options,A,B,C)
```

例 7-17 若已知某 Riccati 微分方程中矩阵及初值如下, 试求解该方程

$$\mathbf{A} = \begin{bmatrix} 6 & 6 & 17 \\ 1 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 2 & 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 8 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \mathbf{P}_1(0.5) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

解 先输入这些矩阵, 然后即可求解该微分方程, 得出的结果如图 7-9 所示

```
>> A=[6,6,17; 1,0,-1; -1,0,0]; B=[0,0,0; 0,4,2; 0,2,1];
C=[1,2,0; 2,8,0; 0,0,4]; P1=[1,0,0; 0,3,0; 0,0,5];
[t,p]=ode45(@ric_de,[0.5,0],P1(:),[],A,B,C); plot(t,p)
```

以得出的 $t = 0$ 时刻的解为初值重新求解该微分方程, 则可以得出和前面完全一致的结果

```
>> P=p(end,:); P0=reshape(P,size(A));
[t1,p1]=ode45(@ric_de,[0,0.5],P0(:),[],A,B,C); line(t1,p1)
```

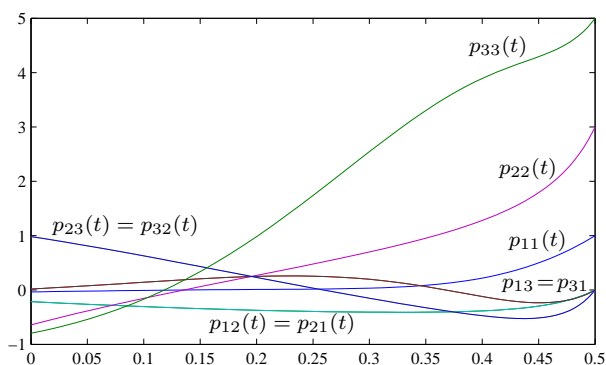


图 7-9 Riccati 微分方程的数值解

7.4 特殊微分方程的数值解

从 7.2 节的介绍和例子看,一般常微分方程组均可以转换成一阶显式常微分方程组,然后通过给定的算法及 MATLAB 求解函数,如 `ode45()` 直接求出方程的数值解。从前面的例子还可以看出, `ode45()` 函数有时失效,所以应该引入一类方程,即刚性微分方程的专门求解函数来解决这样的问题。另外,微分代数方程、隐式微分方程等也是需要引入的微分方程类型,它们的求解将弥补 `ode45()` 函数本身的不足,本节将着重介绍这些方程的求解问题。

7.4.1 刚性微分方程的求解

在许多领域中,经常遇到一类特殊的常微分方程,其中一些解变化缓慢,另一些变化快,且相差较悬殊,这类方程常常称为刚性方程,又称为 Stiff 方程。刚性问题一般不适合由 `ode45()` 这类函数求解,而应该采用 MATLAB 求解函数 `ode15s()`,该函数的调用格式和 `ode45()` 完全一致。

例 7-18 试求解 $\mu = 1000$ 时 Van der Pol 方程的数值解。

解 仿照前面的例子可以给出如下的 MATLAB 命令,在 1.87s 内可以得出方程的数值解

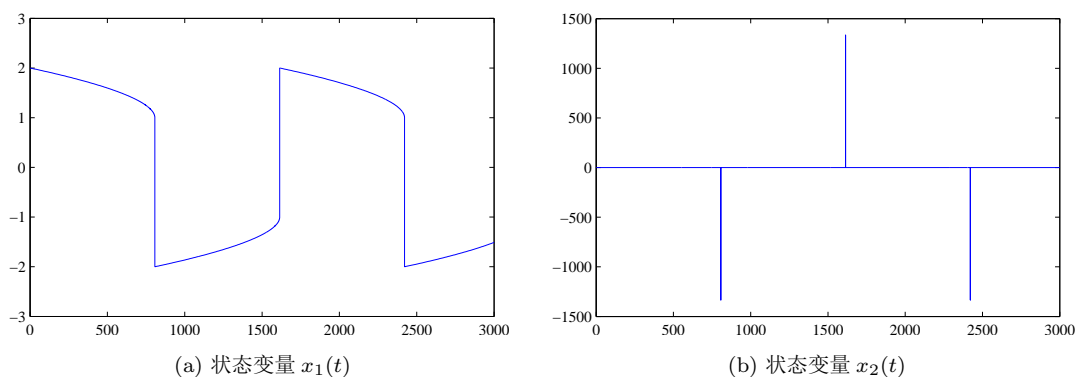
```
>> h_opt=odeset; h_opt.RelTol=1e-10; h_opt.AbsTol=1e-10; x0=[2;0];
    t_final=3000; f=@(t,x,mu)[x(2); -mu*(x(1)^2-1)*x(2)-x(1)];
    tic, mu=1000; [t,y]=ode15s(f,[0,t_final],x0,h_opt,mu); toc
    plot(t,y(:,1)); figure; plot(t,y(:,2))
```

可见,用刚性方程求解函数可以快速求出该方程的数值解,并将两个状态变量的时间曲线分别绘制出来,如图 7-10 所示。从得出的图形可以看出, $x_1(t)$ 曲线变化较平滑,而 $x_2(t)$ 变化在某些点上较快,所以当 $\mu = 1000$ 时, Van der Pol 方程属于典型的刚性方程,应该采用刚性方程的函数求解。

例 7-19 在传统的有关常微分方程数值解的教科书^[3]中,都认为下面的微分方程是刚性的

$$\mathbf{y}' = \begin{bmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{bmatrix} \mathbf{y}, \quad \mathbf{y}_0 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

试用 MATLAB 语言求解该微分方程。

图 7-10 $\mu = 1000$ 时 Van der Pol 方程的解

解 该方程的解析解可以通过 MATLAB 符号运算工具箱中的语句直接求出

```
>> syms t; A=sym([-21,19,-20; 19,-21,20; 40,-40,-40]);
    y0=[1; 0; -1]; y=expm(A*t)*y0
```

原方程的解析解为

$$\mathbf{y}(t) = \begin{bmatrix} 0.5e^{-2t} + 0.5e^{-40t}(\cos 40t + \sin 40t) \\ 0.5e^{-2t} - 0.5e^{-40t}(\cos 40t + \sin 40t) \\ e^{-40t}(\sin 40t - \cos 40t) \end{bmatrix}$$

现在考虑该问题的数值求解方法。根据原始问题,可以立即写出该模型的匿名函数,再利用下面的 MATLAB 语句,可以得出方程的数值解

```
>> f=@(t,x)[-21,19,-20; 19,-21,20; 40,-40,-40]*x;
    opt=odeset; opt.RelTol=1e-6; tic,[t,y]=ode45(f,[0,1],[1;0;-1],opt); toc
    x1=exp(-2*t); x2=exp(-40*t).*cos(40*t); x3=exp(-40*t).*sin(40*t);
    y1=[0.5*x1+0.5*x2+0.5*x3, 0.5*x1-0.5*x2-0.5*x3, -x2+x3]; plot(t,y,t,y1,'--')
```

原方程的解析解和数值解如图 7-11 (a) 所示。可以看出,问题的数值解的精度比较高,计算速度相对也较快,但从这里似乎看不出原问题的刚性所在。究其原因,因为在 MATLAB 下采用了变步长的算法,它可以依照要求的精度自动地修正步长,所以感受不到它是个刚性问题。

如果采用定步长方法,利用前面编写的四阶 Runge-Kutta 定步长算法程序 rk_4(),再用下面的语句就能求解原方程了

```
>> tic, [t2,y2]=rk_4(f,[0,1,0.01],[1;0;-1]); toc; plot(t,y1,t2,y2,'--')
```

这样得出的曲线与解析解的对比见图 7-11 (b)。从计算的结果看,显然采用定步长的算法在取较大的步长时,得出的解是不正确的。减小步长时,直至减小到 0.0001 时,仍能从得出的结果看出与解析解的差距,而这时的计算时间为 26 s,是前面 ode45() 变步长算法所需时间的 162 倍。所以在实际应用中最好采用变步长算法。

从得出的曲线可以看出,这 3 条曲线的变化速度差别不是特别悬殊,在以前计算能力受限时被误认为是刚性问题,而在 MATLAB 下则可以由普通的函数求解。

由此可以得出结论,许多传统的刚性问题采用 MATLAB 的普通求解函数就可以直接

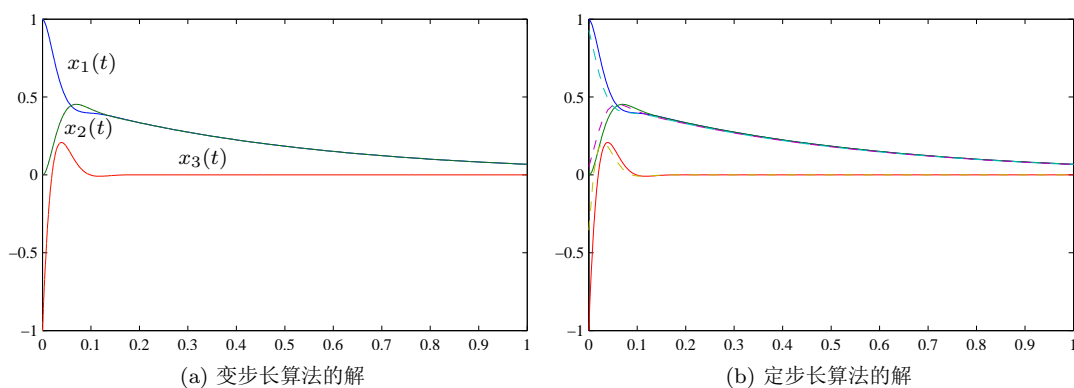


图 7-11 给定的传统刚性问题解法比较

解出,而不必刻意地去选择刚性问题的解法。当然,在有些问题的求解中确实需要采用刚性问题的解法,这将在例 7-20 中加以说明。

例 7-20 考虑下面的常微分方程

$$\begin{cases} y_1' = 0.04(1 - y_1) - (1 - y_2)y_1 + 0.0001(1 - y_2)^2 \\ y_2' = -10^4 y_1 + 3000(1 - y_2)^2 \end{cases}$$

其中,该方程的初值为 $y_1(0) = 0, y_2(0) = 1$ 。取计算区间为 $t \in (0, 100)$,试选择合适的算法得出该方程的数值解。

解 根据给出的微分方程,可以写出其匿名函数,再给出下面的语句

```
>> f=@(t,y)[0.04*(1-y(1))-(1-y(2))*y(1)+0.0001*(1-y(2))^2; ...
-10^4*y(1)+3000*(1-y(2))^2];
tic,[t2,y2]=ode45(f,[0,100],[0;1]); toc; length(t2), plot(t2,y2)
```

经过长时间的等待,可以得出原问题的数值解,如图 7-12(a) 所示。可以看出,调用普通的解法函数 `ode45()` 计算所需的时间过长,计算的点高达 356941 个,对这个例子来说,计算的点高达 35 万。再分析变步长解法所使用的步长,语句如下

```
>> [min(diff(t2)), max(diff(t2))], plot(t2(1:end-1),diff(t2))
```

则可以看出,由于设定的精度要求较高,不得不采用小步长来解决问题。实际的步长如图 7-12(b) 所示。可见在大部分时间内,所采用的步长小于 0.0002,这使得解题时间大大增加。

考虑用 `ode15s()` 替代 `ode45()`,则耗时 0.281 s,计算点个数为 606,并可以得出方程的数值解。可见,求解时间大大减小,效率提高了 1100 多倍,得出的曲线则几乎完全一致。

```
>> opt=odeset; opt.RelTol=1e-10; opt.AbsTol=1e-10;
tic,[t1,y1]=ode15s(f,[0,100],[0;1],opt); toc, length(t1), plot(t1,y1)
```

7.4.2 隐式微分方程求解

所谓隐式微分方程就是那些不能转换成式 (7-2-1) 中一阶显式常微分方程组的微分方程。MATLAB 语言早期版本中未提供能直接求解隐式微分方程的算法及函数,所以仍可以借用显式微分方程求解的函数来求解这类问题。本节将通过两个例子来演示隐式微分方程

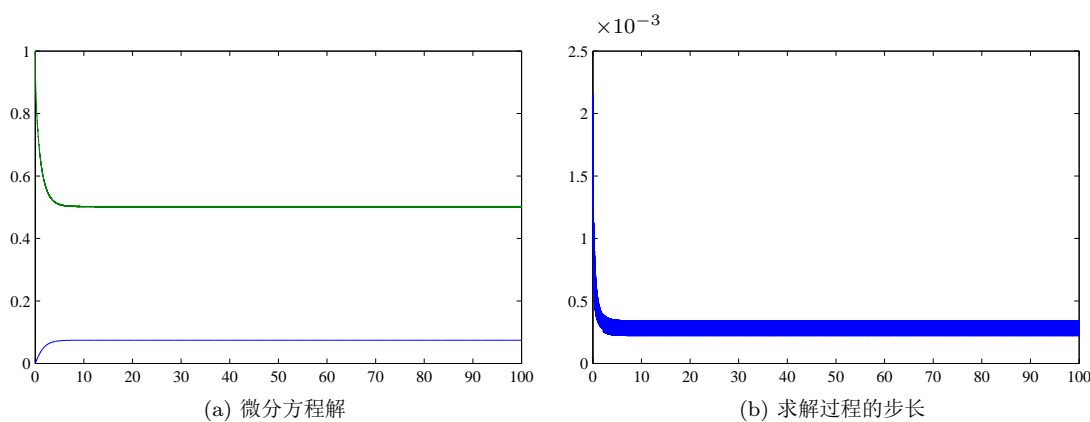


图 7-12 四阶五级 Runge-Kutta-Fehlberg 法结果

的求解方法与应用。

例 7-21 给定如下隐式微分方程, 且已知 $x_1(0) = x_2(0) = 0$, 试求出该方程的数值解。

$$\begin{cases} x_1' \sin x_1 + x_2' \cos x_2 + x_1 = 1 \\ -x_1' \cos x_2 + x_2' \sin x_1 + x_2 = 0 \end{cases}$$

解 令 $\mathbf{x} = [x_1, x_2]^T$, 则可以将原方程改写成矩阵形式 $\mathbf{A}(\mathbf{x})\mathbf{x}' = \mathbf{B}(\mathbf{x})$, 其中

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} \sin x_1 & \cos x_2 \\ -\cos x_2 & \sin x_1 \end{bmatrix}, \quad \mathbf{B}(\mathbf{x}) = \begin{bmatrix} 1 - x_1 \\ -x_2 \end{bmatrix}$$

如果能证明 $\mathbf{A}(\mathbf{x})$ 为非奇异的矩阵, 则直接能将该方程变换成标准的显式一阶微分方程组的形式, 即 $\mathbf{x}' = \mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})$, 套用 MATLAB 函数即可求解对应的方程。事实上, 由于无法严格证明 $\mathbf{A}(\mathbf{x})$ 矩阵是非奇异矩阵, 故可以大胆地假设该矩阵为非奇异矩阵, 利用 MATLAB 语言试解该方程, 如果在求解过程中不出现 $\mathbf{A}(\mathbf{x})$ 为奇异矩阵的警告信息, 则说明对求出的解不存在 $\mathbf{A}(\mathbf{x})$ 为奇异矩阵的现象, 得出的解是有效的。若求解过程中确实出现奇异矩阵警告, 则得出的解没有实际意义。

对于这里要研究的隐式微分方程模型, 可以用匿名函数描述原微分方程, 这样, 可以给出下面的命令求解方程

```
>> f=@(t,x)inv([sin(x(1)) cos(x(2)); -cos(x(2)) sin(x(1))])*[1-x(1); -x(2)];
    opt=odeset; opt.RelTol=1e-6; [t,x]=ode45(f,[0,10],[0; 0],opt); plot(t,x)
```

同时将绘制出状态变量的时间曲线, 如图 7-13 所示。在求解的过程中也没有得出有关矩阵奇异的错误信息, 故得出的结果是可信的。

例 7-22 前面的例子很简单, 可以将其立即变换成显式微分方程。现在考虑一个更复杂的隐式微分方程组, 如下所示。假设该方程初始状态为 $\mathbf{x} = [1, 0, 0, 1]^T$, 试求取该微分方程的数值解。

$$\begin{cases} x'' \sin y' + y''^2 = -2xye^{-x'} + xx''y' \\ xx''y'' + \cos y'' = 3yx'e^{-x} \end{cases}$$

解 可以仍然选定状态变量 $x_1 = x, x_2 = x', x_3 = y, x_4 = y'$, 这样仍然有 $x_1' = x_2, x_3' = x_4$ 。显然, 并不可能像例 7-15 那样解析地求解方程, 得出 x_2' 和 x_4' 的显式表达式, 但可以讨论该方程组数值解的方法, 对每组状态变量 \mathbf{x} 得出它们的值。

由原来给出的方程, 假设 $p_1 = x'', p_2 = y''$, 则可以将原方程改写成

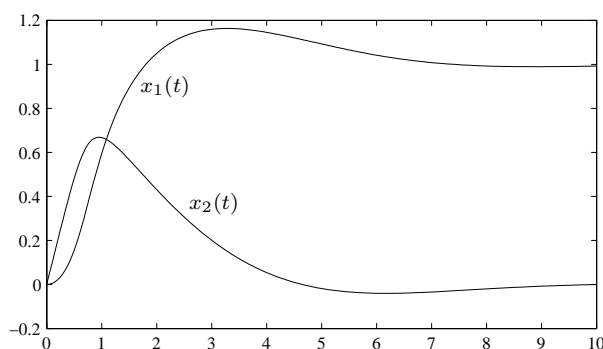


图 7-13 隐式方程的时间响应曲线

$$\begin{cases} p_1 \sin x_4 + p_2^2 + 2x_1 x_3 e^{-x_2} - x_1 p_1 x_4 = 0 \\ x_1 p_1 p_2 + \cos p_2 - 3x_3 x_2 e^{-x_1} = 0 \end{cases}$$

依据该方程可以得出如下的 MATLAB 语句,从而写出相应的函数来描述微分方程。

```
function dy=c7impode(t,x)
dx=@(p,x)[p(1)*sin(x(4))+p(2)^2+2*x(1)*x(3)*exp(-x(2))-x(1)*...
    p(1)*x(4); x(1)*p(1)*p(2)+cos(p(2))-3*x(3)*x(2)*exp(-x(1))];
ff=optimset; ff.Display='off'; dx1=fsolve(dx,x([1,3]),ff,x);
dy=[x(2); dx1(1); x(4); dx1(2)];
```

进入该函数时,由状态变量 x 和新定义的 p_1, p_2 可以写出匿名函数,描述未知量 p_i 代入方程后两个方程的误差,而这里 x 是作为已知的附加参数给出的。微分方程求解程序每次调用这个原型函数时均求解一次关于 p_i 的代数方程,得出的结果 p_1, p_2 实际上就是 x'_2, x'_4 ,这样就可以构造出微分方程对应的状态变量的导数。在求解代数方程中使用了一个小技巧,即代数方程的初值选择 $p_1(0) = x_1, p_2(0) = x_3$,这样会使得代数方程收敛速度和精度都加快。

建立起微分方程模型后,就可以通过下面的语句直接求解微分方程,并绘制出状态变量的时间曲线,如图 7-14 所示。

```
>> [t,x]=ode15s(@c7impode,[0,2],[1,0,0,1]); plot(t,x)
```

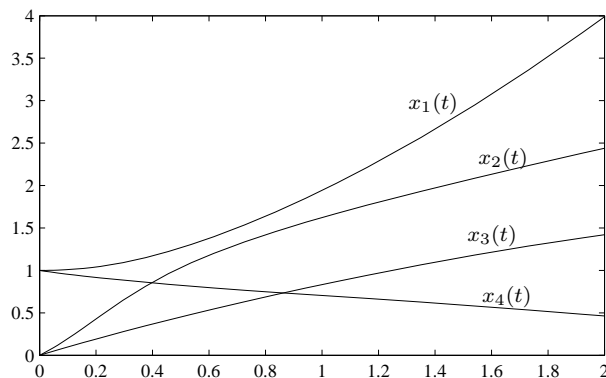


图 7-14 隐式方程的时间响应曲线

MATLAB 函数 `ode15i()` 可以直接用于隐式微分方程的求解。若隐式微分方程为

$$F[t, \mathbf{x}(t), \mathbf{x}'(t)] = \mathbf{0}, \text{ 且 } \mathbf{x}(0) = \mathbf{x}_0, \mathbf{x}'(0) = \mathbf{x}'_0 \quad (7-4-1)$$

则可以编写一个函数 `fun()` 描述该隐式微分方程, 然后用 `decic()` 函数求出未完全定义的初值条件, 再调用 `ode15i()` 函数即可以求解该隐式微分方程如下

```
[x0*, x'0*] = decic(fun, t0, x0, x0F, x'0, x'0F) % 获得相容初始条件
[t, x] = ode15i(fun, tspan, x0*, x'0*) % 求解隐式方程
```

隐式微分方程不同于一般显示微分方程, 求解之前需要给出 $(\mathbf{x}_0, \mathbf{x}'_0)$, 它们不能任意赋值, 只能有 n 个是独立的, 其余的需要用隐式方程求解, 否则将可能出现矛盾的初始条件。所以在实际求解过程中, 如果不能确定 \mathbf{x}'_0 值, 则应该先调用 `decic()` 函数得出相容的初值。在函数调用中 $(\mathbf{x}_0, \mathbf{x}'_0)$ 为任意给定的初值, \mathbf{x}_0^F 和 \mathbf{x}'_0^F 均为 n 维列向量, 其值为 1 表示需要保留的初值, 为 0 表示需要求解的初值项, 通过方程求解将得出相容的初值 \mathbf{x}_0^* , \mathbf{x}'_0^* , 该初值可以直接用于隐式微分方程求解函数 `ode15i()`。下面将通过具体例子演示隐式微分方程的求解方法。

例 7-23 试用隐式微分方程求解的方法解出例 7-22 中给出的隐式微分方程。

解 选择状态变量 $x_1 = x, x_2 = x', x_3 = y, x_4 = y'$, 则原方程可以变换成

$$\begin{cases} x'_1 - x_2 = 0 \\ x'_2 \sin x_4 + x_4'^2 + 2e^{-x_2} x_1 x_3 - x_1 x'_2 x_4 = 0 \\ x'_3 - x_4 = 0 \\ x_1 x'_2 x'_4 + \cos x'_4 - 3e^{-x_1} x_3 x_2 = 0 \end{cases}$$

这样, 可以编写出如下所示的描述隐式微分方程的 MATLAB 函数并求解微分方程

```
>> f=@(t,x,xd)[xd(1)-x(2);
    xd(2)*sin(x(4))+xd(4)^2+2*exp(-x(2))*x(1)*x(3)-x(1)*xd(2)*x(4);
    xd(3)-x(4);
    x(1)*xd(2)*xd(4)+cos(xd(4))-3*exp(-x(1))*x(3)*x(2)];
x0=[1,0,0,1]; xd0=[0;1;1;-1]; x0F=[1 1 1 1]; % 保留 x0
xd0F=[]; [x0,xd0]=decic(f,0,x0,x0F,xd0,xd0F) % 由 x0 确定 x0'
[t,x]=ode15i(f,[0,2],x0,xd0); plot(t,x) % 绘制时间响应曲线
```

其中, 初值 \mathbf{x}_0 与前面例子中一致。为得到相容的一阶微分向量初值, 可以设置 \mathbf{x}_0^F 为幺向量, 表示 \mathbf{x}_0 初值需要保留, 而 \mathbf{x}'_0^F 应该设置成零向量, 表示一阶微分初始向量应该根据需要重新计算。通过上述运算可以得出 $\mathbf{x}'_0 = [0, 1.6833, 1, -0.5166]^T$, 这样调用上面的语句即可以求解隐式微分方程, 且可绘制出该方程解的时间曲线, 和图 7-14 中给出的曲线完全一致。

7.4.3 微分代数方程的求解

在前面的介绍中, 所介绍的常微分方程数值解法主要是针对能够转换成一阶常微分方程组的类型, 假设其中的一些微分方程退化为代数方程, 则用前面介绍的算法无法求解, 必须借助微分代数方程的特殊解法。

所谓微分代数方程 (differential algebraic equation, DAE), 是指在微分方程中, 某些变量间满足某些代数方程的约束, 所以这样的方程不能用前面介绍的常微分方程解法直接进行求解。假设微分方程的更一般形式可以写成

$$\mathbf{M}(t, \mathbf{x})\mathbf{x}' = \mathbf{f}(t, \mathbf{x}) \quad (7-4-2)$$

描述 $\mathbf{f}(t, \mathbf{x})$ 的方法和普通常微分方程完全一致, 而对真正的微分代数方程来说, $\mathbf{M}(t, \mathbf{x})$ 矩阵为奇异矩阵, 在微分代数方程求解程序中应该由求解选项中的成员变量 **Mass** 来表示该矩阵, 考虑了这些因素则可以立即求解方程的解了。

例 7-24 考虑下面给出的微分代数方程

$$\begin{cases} x_1' = -0.2x_1 + x_2x_3 + 0.3x_1x_2 \\ x_2' = 2x_1x_2 - 5x_2x_3 - 2x_2^2 \\ 0 = x_1 + x_2 + x_3 - 1 \end{cases}$$

并已知初始条件为 $x_1(0) = 0.8$, $x_2(0) = x_3(0) = 0.1$, 试求取该方程的数值解。

解 可以看出, 最后的一个方程为代数方程, 可以视之为 3 个状态变量间的约束关系。用矩阵的形式可以表示该微分代数方程为

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} -0.2x_1 + x_2x_3 + 0.3x_1x_2 \\ 2x_1x_2 - 5x_2x_3 - 2x_2^2 \\ x_1 + x_2 + x_3 - 1 \end{bmatrix}$$

这样就可以写出相应的 MATLAB 函数如下

```
>> f=@(t,x)[-0.2*x(1)+x(2)*x(3)+0.3*x(1)*x(2);  
            2*x(1)*x(2)-5*x(2)*x(3)-2*x(2)*x(2); x(1)+x(2)+x(3)-1];
```

可以将 \mathbf{M} 矩阵输入给 MATLAB 工作空间, 并在命令窗口中给出如下命令, 由上面的语句可以得出此微分代数方程的解, 如图 7-15 所示。

```
>> M=[1,0,0; 0,1,0; 0,0,0]; options=odeset; options.Mass=M;  
x0=[0.8; 0.1; 0.1]; [t,x]=ode15s(f,[0,20],x0,options); plot(t,x)
```

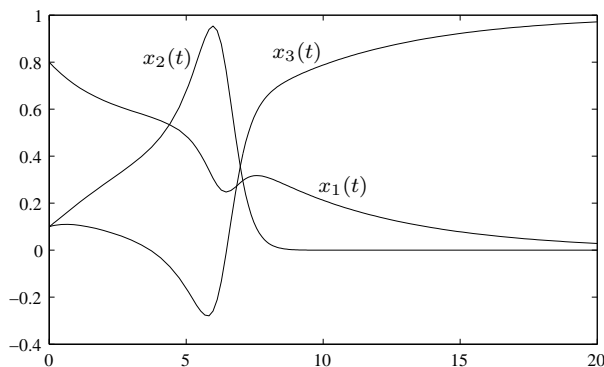


图 7-15 微分代数方程的数值解

事实上, 有些微分代数方程可以转换成常微分方程求解。例如, 在本例中, 可以从约束式子中求出 $x_3(t) = 1 - x_1(t) - x_2(t)$, 将其代入其他两个微分方程式子, 则有

$$\begin{cases} x_1' = -0.2x_1 + x_2[1 - x_1(t) - x_2(t)] + 0.3x_1x_2 \\ x_2' = 2x_1x_2 - 5x_2[1 - x_1(t) - x_2(t)] - 2x_2^2 \end{cases}$$

根据该方程可以写出匿名函数描述微分方程

```
>> f=@(t,x)[-0.2*x(1)+x(2)*(1-x(1)-x(2))+0.3*x(1)*x(2);
          2*x(1)*x(2)-5*x(2)*(1-x(1)-x(2))-2*x(2)*x(2)];
```

这样则可以用下面的命令求解变换后的微分方程组,从而最终得出原微分代数方程的解,所得出的解与前面的直接解法得出的完全一致。

```
>> fDae=@(t,x)[-0.2*x(1)+x(2)*(1-x(1)-x(2))+0.3*x(1)*x(2);...
          2*x(1)*x(2)-5*x(2)*(1-x(1)-x(2))-2*x(2)*x(2)];
x0=[0.8; 0.1]; [t1,x1]=ode45(fDae,[0,20],x0); plot(t1,x1,t1,1-sum(x1'))
```

注意,这里如果使用 ode45() 函数也不会出现求解错误。

如果利用 MATLAB 的隐式微分方程求解函数 ode15i(), 则可以用下面的语句描述该方程

```
>> f=@(t,x,xd)[xd(1)+0.2*x(1)-x(2)*x(3)-0.3*x(1)*x(2);
          xd(2)-2*x(1)*x(2)+5*x(2)*x(3)+2*x(2)^2; x(1)+x(2)+x(3)-1];
```

令 $x_0 = [1, 1, *]^T$, 其中 * 表示自由值, 则可以用下面的语句解出相容的初始条件, 并直接求解该微分代数方程, 得出的结果将和前面完全一致, 但求解更直观。

```
>> x0=[0.8;0.1;2]; x0F=[1;1;0]; xd0=[1;1;1]; xd0F=[];
[x0,xd0]=decic(f,0,x0,x0F,xd0,xd0F); [x0,xd0] % 相容初始条件
res=ode15i(f,[0,20],x0,xd0); plot(res.x,res.y)
```

得出的相容初始条件为 $x_0 = [0.8, 0.1, 0.1]^T$, $x'_0 = [-0.126, 0.09, 1]^T$ 。

例 7-25 试用微分代数方程求解的方式求解例 7-21 中定义的隐式微分方程。

解 在例 7-21 中采用对 $A(x)$ 矩阵直接求逆的形式将原隐式方程转换成显式一阶微分方程组, 这样就可以用一般微分方程组数值解法直接得出方程的解。其实, 在这样的求解过程中作了一个假设, 即 $A(x)$ 矩阵为非奇异矩阵, 虽然对这个例子碰巧是正确的, 但这种解法毕竟不严密, 所以需要采用微分代数方程的方法来求解该问题。

对原方程进行分析发现, 可以编写一个匿名函数来描述微分方程与质量矩阵

```
>> f=(t,x)[1-x(1);-x(2)]; M=@(t,x)[sin(x(1)),cos(x(2));-cos(x(2)),sin(x(1))];
```

这样就可以用下面的语句调用微分代数方程求解微分代数方程

```
>> options=odeset; options.Mass=M; options.RelTol=1e-6;
[t,x]=ode45(f,[0,10],[0;0],options); plot(t,x)
```

得出的图形仍将与图 7-13 中的曲线完全一致。

7.4.4 切换微分方程的求解

切换系统是控制理论中的一个重要的研究领域^[4], 所谓切换系统就是在某种规律下其

模型在多个模型间切换的系统。切换系统的微分方程模型可以表示为

$$\mathbf{x}'(t) = \mathbf{f}_i(t, \mathbf{x}, \mathbf{u}), \quad i = 1, \dots, m \quad (7-4-3)$$

该系统允许在某个控制规律下, 整个系统在各个模型之间切换。利用切换系统的理论, 可以设计控制器, 使得不稳定的各个模型 $\mathbf{f}_i(\cdot)$ 通过合理的切换达到整个系统的稳定。

例 7-26 假设已知系统模型 $\mathbf{x}' = \mathbf{A}_i \mathbf{x}$, 其中 $\mathbf{A}_1 = \begin{bmatrix} 0.1 & -1 \\ 2 & 0.1 \end{bmatrix}$, $\mathbf{A}_2 = \begin{bmatrix} 0.1 & -2 \\ 1 & 0.1 \end{bmatrix}$ 。可见, 两个系统都不稳定。若 $x_1 x_2 < 0$, 即状态处于第 II、IV 象限时, 切换到系统 \mathbf{A}_1 , 而 $x_1 x_2 \geq 0$, 即状态处于 I、III 象限时切换到 \mathbf{A}_2 。令初始状态为 $x_1(0) = x_2(0) = 5$, 试求解该系统的微分方程。

解 按照系统模型及切换律, 可以容易地写出切换系统的 MATLAB 表示为

```
function dx=switch_sys(t,x)
```

```
if x(1)*x(2)<0, A=[0.1 -1; 2 0.1]; else, A=[0.1 -2; 1 0.1]; end, dx=A*x;
```

这样就能用下面的语句直接求解切换系统的方程, 得出如图 7-16 所示的时间响应曲线和相平面曲线。可见, 不稳定的状态方程模型在某种指定的切换律下, 可以得出稳定的整体系统状态。

```
>> [t,x]=ode45(@switch_sys,[0,30],[5;5]); plot(t,x)
```

```
figure, plot(x(:,1),x(:,2))
```

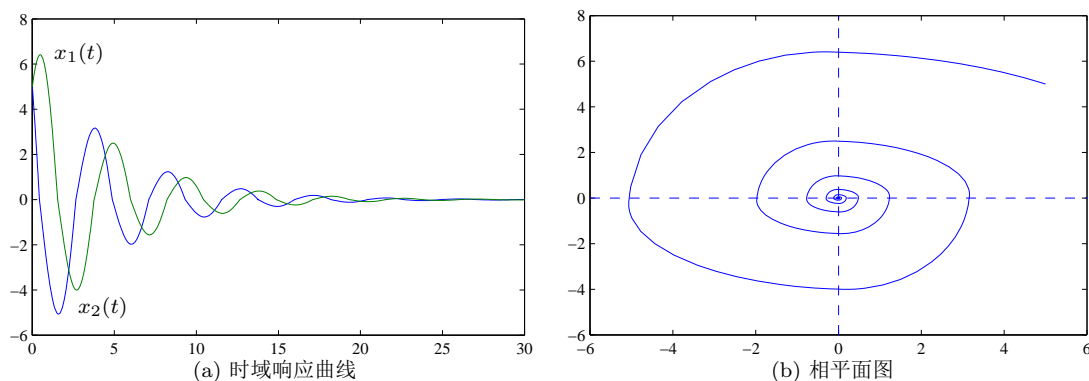


图 7-16 切换系统的响应与切换效果

其实, 为方便起见, 该切换系统还可以由匿名函数表示成

```
>> f=@(t,x)(x(1)*x(2)<0)*[0.1 -1; 2 0.1]*x+(x(1)*x(2)>=0)*[0.1 -2; 1 0.1]*x;
```

7.4.5 随机线性微分方程的求解

假设随机线性连续状态方程模型为

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}[\mathbf{d}(t) + \boldsymbol{\gamma}(t)], \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \quad (7-4-4)$$

式中 \mathbf{A} , \mathbf{B} , \mathbf{C} 为兼容矩阵, $\mathbf{d}(t)$ 为确定性输入向量, $\boldsymbol{\gamma}(t)$ 为 Gauss 白噪声向量, 满足

$$\mathbb{E}[\boldsymbol{\gamma}(t)] = 0, \quad \mathbb{E}[\boldsymbol{\gamma}(t)\boldsymbol{\gamma}^T(\tau)] = \mathbf{V}_\sigma \delta(t - \tau) \quad (7-4-5)$$

定义一个变量 $\gamma_c(t) = \mathbf{B}\gamma(t)$, 则可以证明 $\gamma_c(t)$ 亦为 Gauss 白噪声, 满足

$$\mathbb{E}[\gamma_c(t)] = 0, \quad \mathbb{E}[\gamma_c(t)\gamma_c^T(\tau)] = \mathbf{V}_c\delta(t - \tau) \quad (7-4-6)$$

其中 $\mathbf{V}_c = \mathbf{B}\mathbf{V}_\sigma\mathbf{B}^T$ 为一个协方差矩阵, 这时式 (7-4-4) 可以改写成

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{d}(t) + \gamma_c(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \quad (7-4-7)$$

状态变量的解析解可以写成

$$\mathbf{x}(t) = e^{-\mathbf{A}t}\mathbf{x}(t_0) + \int_{t_0}^t e^{\mathbf{A}(t-\tau)}\mathbf{d}(\tau)\mathbf{B}d\tau + \int_{t_0}^t \gamma_c(\tau)d\tau \quad (7-4-8)$$

假设 $t_0 = k\Delta t$, $t = (k+1)\Delta t$, 其中 Δt 为计算步长, 并假定在一个计算步长内确定性输入信号 $\mathbf{d}(t)$ 为一个常数, 亦即, 如 $\Delta t \leq t \leq (k+1)\Delta t$ 时有 $\mathbf{d}(t) = \mathbf{d}(k\Delta t)$, 则式 (7-4-8) 的离散形式可以写成

$$\mathbf{x}[(k+1)\Delta t] = \mathbf{F}\mathbf{x}(k\Delta t) + \mathbf{G}\mathbf{d}(k\Delta t) + \gamma_d(k\Delta t), \quad \mathbf{y}(k\Delta t) = \mathbf{C}\mathbf{x}(k\Delta t) \quad (7-4-9)$$

式中 $\mathbf{F} = e^{\mathbf{A}\Delta t}$, $\mathbf{G} = \int_0^{\Delta t} e^{\mathbf{A}(\Delta t-\tau)}\mathbf{B}d\tau$, 且

$$\gamma_d(k\Delta t) = \int_{k\Delta t}^{(k+1)\Delta t} e^{\mathbf{A}[(k+1)\Delta t-\tau]}\gamma_c(\tau)d\tau = \int_0^{\Delta t} e^{\mathbf{A}t}\gamma_c[(k+1)\Delta t - \tau]d\tau \quad (7-4-10)$$

可见矩阵 \mathbf{F} , \mathbf{G} 和确定性系统的离散化形式是一样的, 所以会很容易求得, 但可以看出, 若系统含有随机输入时, 系统的离散化形式与传统形式是不同的。可以证明 $\gamma_d(t)$ 亦为 Gauss 白噪声向量, 且满足

$$\mathbb{E}[\gamma_d(k\Delta t)] = 0, \quad \mathbb{E}[\gamma_d(k\Delta t)\gamma_d^T(j\Delta t)] = \mathbf{V}\delta_{kj} \quad (7-4-11)$$

式中 $\mathbf{V} = \int_0^{\Delta t} e^{\mathbf{A}t}\mathbf{V}_c e^{\mathbf{A}^T t}dt$ 。利用 Taylor 幂级数展开技术可得

$$\mathbf{V} = \int_0^{\Delta t} \sum_{k=0}^{\infty} \frac{\mathbf{R}_k(0)}{k!} t^k dt = \sum_{k=0}^{\infty} \mathbf{V}_k \quad (7-4-12)$$

其中 $\mathbf{R}_k(0)$ 与 \mathbf{V}_k 可以由下式递推求出 [5]

$$\begin{cases} \mathbf{R}_k(0) = \mathbf{A}\mathbf{R}_{k-1}(0) + \mathbf{R}_{k-1}(0)\mathbf{A}^T \\ \mathbf{V}_k = \frac{\Delta t}{k+1}(\mathbf{A}\mathbf{V}_{k-1} + \mathbf{V}_{k-1}\mathbf{A}^T) \end{cases} \quad (7-4-13)$$

递推初值为 $\mathbf{R}_0(0) = \mathbf{R}(0) = \mathbf{V}_c$, $\mathbf{V}_0 = \mathbf{V}_c\Delta t$ 。由奇异值分解理论, 可以将矩阵 \mathbf{V} 写成 $\mathbf{V} = \mathbf{U}\mathbf{\Gamma}\mathbf{U}^T$, 其中 \mathbf{U} 为正交矩阵, $\mathbf{\Gamma}$ 为含有非零对角元素的对角矩阵, 这样可以

得出 Cholesky 分解 $V = DD^T$ 。且 $\gamma_d(k\Delta t) = De(k\Delta t)$ ，式中 $e(k\Delta t)$ 为 $n \times 1$ 向量，且 $e(k\Delta t) = [e_k, e_{k+1}, \dots, e_{k+n-1}]^T$ ，使得各个分量 e_k 满足标准正态分布，即 $e_k \sim N(0, 1)$ 。系统的离散形式递推解可以写成

$$\begin{cases} \mathbf{x}[(k+1)\Delta t] = \mathbf{F}\mathbf{x}(k\Delta t) + \mathbf{G}d(k\Delta t) + \mathbf{D}e(k\Delta t) \\ \mathbf{y}(k\Delta t) = \mathbf{C}\mathbf{x}(k\Delta t) \end{cases} \quad (7-4-14)$$

根据上面的算法，可以编写出随机输入下连续线性系统离散化的 `sc2d()` 如下

```
function [F,G,D,C]=sc2d(G,sig,T)
G=ss(G); G=balreal(G); Gd=c2d(G,T); A=G.a; B=G.b; C=G.c; i=1;
F=Gd.a; G=Gd.b; V0=B*sig*B'*T; Vd=V0; V1=Vd;
while (norm(V1)<eps)
    V1=T/(i+1)*(A*V0+V0*A'); Vd=Vd+V1; V0=V1; i=i+1;
end
[U,S,V0]=svd(Vd); V0=sqrt(diag(S)); Vd=diag(V0); D=U*Vd;
```

该函数的调用格式为 `[F,G,D,C]=sc2d(G,σ,Δt)`，其中， G 为系统模型， σ 为输入信号协方差矩阵， Δt 为采样周期， (F, G, D, C) 为离散化状态方程的相应矩阵。

在仿真时，可以产生一组伪随机数，从而产生向量 $e(k\Delta t)$ ，然后求出状态变量 $\mathbf{x}[(k+1)\Delta t]$ 并求出输出变量 $\mathbf{y}[(k+1)\Delta t]$ 。

例 7-27 考虑受控对象的传递函数模型为 $G(s) = \frac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24}$ ，如果用白噪声信号激励该系统，试对其进行仿真分析并得出输出信号的统计规律。

解 假设系统的采样周期为 $T = 0.02\text{s}$ ，下面的语句

```
>> G=tf([1,7,24,24],[1,10,35,50,24]); [F,G0,D,C]=sc2d(G,1,0.02)
```

可以得出离散化的状态方程模型为

$$\mathbf{F} = \begin{bmatrix} 0.9838 & -0.00673 & 0.0132 & 0.00129 \\ 0.00673 & 0.9883 & 0.07022 & 0.00364 \\ 0.0132 & -0.07022 & 0.8653 & -0.0257 \\ 0.00129 & -0.0036401 & -0.0257 & 0.9684 \end{bmatrix}, \quad \mathbf{G}_0 = \begin{bmatrix} 0.01823 \\ -0.00355 \\ -0.00757 \\ -0.000718 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} -0.12893 & 0.00088028 & -4.6919 \times 10^{-6} & 4.6917 \times 10^{-10} \\ 0.0251 & -0.0012 & -1.3573 \times 10^{-5} & 2.3791 \times 10^{-9} \\ 0.05356 & 0.002635 & -5.2322 \times 10^{-6} & -8.8812 \times 10^{-10} \\ 0.00508 & 0.0005 & 3.1358 \times 10^{-6} & 9.5176 \times 10^{-9} \end{bmatrix}$$

由离散化的状态方程模型出发，可以用下列 MATLAB 语句对之进行仿真，其中仿真点数设为 30000 个，如果太少则统计结论不一定正确。

```
>> n=30000; e=randn(n+4,1); e=e-mean(e); y=zeros(n,1); x=zeros(4,1); d0=0;
for i=1:n, x=F*x+G0*d0+D*e(i:i+3); y(i)=C*x; end
T=0.02; t=0:T:(n-1)*T; plot(t,y), v=norm(G)
```

得出的响应曲线如图 7-17(a) 所示，同时还可以得出系统的 H_2 范数的值为 $v = 0.6655$ 。不过从得出的曲线看，这样的响应似乎杂乱无章，所以对随机输入来说，分析其统计规律应该更有用。可以考虑

将输出范围 $(-2.5, 2.5)$ 划分成宽度为 $w = 0.2$ 的小区间, 累加出落入每个小区间的输出点个数, 由这些值除以 nw 则可以得出基于仿真结果的概率密度值, 如图 7-17(b) 所示。另外, 可以从理论上证明 [6], 输出信号的概率密度为 $p(y) = \frac{1}{\sqrt{2\pi}v} e^{-y^2/(2v^2)}$, 这样, 在得出的直方图上还可以叠印上系统的理论概率密度, 可见和由仿真得出的结果较吻合。

```
>> w=0.2; x=-2.5:w:2.5; y1=hist(y,x); bar(x,y1/n/w);
    x1=-2.5:0.05:2.5; y2=1/sqrt(2*pi)/v*exp(-x1.^2/2/v^2);
    line(x1,y2) % 叠印理论概率密度函数曲线
```

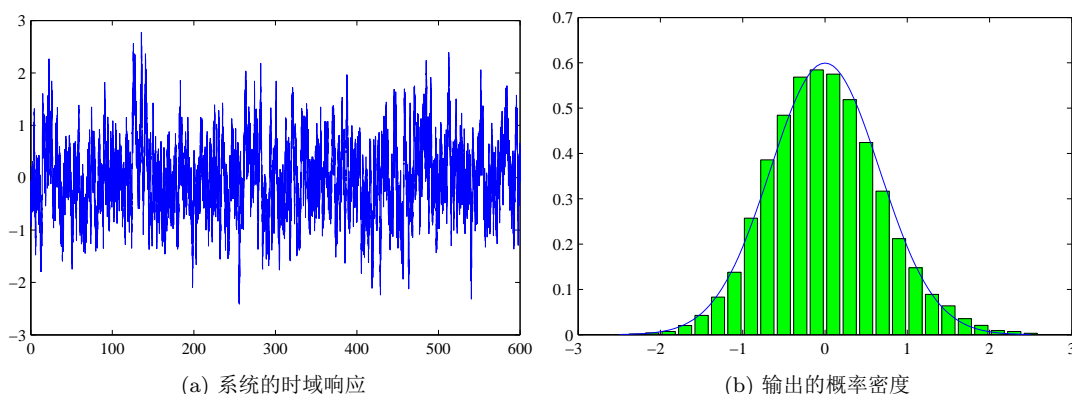


图 7-17 随机输入系统的响应

7.5 延迟微分方程求解

前面介绍的所有微分方程描述的都是 $\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t))$ 形式的微分方程, 其中方程中所有的信号都是同时发生在当前时刻 t 的信号。如果方程中不但含有某些信号当前时刻的值, 还含有某些信号以前的值, 这些方程又称为延迟微分方程。本节将介绍各类延迟微分方程, 包括一般延迟微分方程以及中立型延迟微分方程、变时间延迟方程的数值求解方法等。

7.5.1 典型延迟微分方程的数值求解

延迟微分方程组的一般形式为

$$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{x}(t - \tau_1), \mathbf{x}(t - \tau_2), \dots, \mathbf{x}(t - \tau_m)) \quad (7-5-1)$$

其中, $\tau_i \geq 0$ 为状态变量 $\mathbf{x}(t)$ 的延迟常数。和以前介绍的微分方程不同, 这里涉及的信号除了当前 t 时刻的信号之外, 还有以前时刻的信号, 这些以前的信号即由延迟信号来表示。

MATLAB 提供了求解这类方程的隐式 Runge-Kutta 算法 `dde23()`, 可以直接求解延迟微分方程。该函数的调用格式为 `sol = dde23(f1, τ , f2, [t0, tf], options)`, 其中, 和前面介绍的一样, `options` 是微分方程求解器的控制模板, $\tau = [\tau_1, \tau_2, \dots, \tau_m]$ 为描述延迟微分方程的 MATLAB 语言函数, f_2 为描述 $t \leq t_0$ 时的状态变量值的函数。如果是函数则可以为 MATLAB 语言函数, 如果为常量则可以由向量直接给出。该函数返回的变量 `sol` 为结

构体数据,其 `sol.x` 成员变量为时间向量 t ,成员变量 `sol.y` 为各个时刻的状态向量构成的矩阵 x ,和 `ode45()` 等返回的 x 矩阵是不一样的,它是按照行排列的,正好是该函数结果的转置矩阵。可见,该函数调用格式很不规范,期望能在后面的版本中有所改变。描述延迟微分方程时除了常规的标量 t 和状态向量 x 之外,还需要矩阵 Z ,其第 k 列的向量 $Z(:,k)$ 为状态变量的 τ_k 时间延迟向量。

例 7-28 已知延迟微分方程组
$$\begin{cases} x'(t) = 1 - 3x(t) - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5), \\ y''(t) + 3y'(t) + 2y(t) = 4x(t) \end{cases}$$

其中,在 $t \leq 0$ 时, $x(t) = y(t) = y'(t) = 0$,试求出该方程的数值解。

解 可见,该方程中含有 $x(t)$ 、 $y(t)$ 信号在 t 、 $t-1$ 、 $t-0.5$ 时刻的值,所以需要专门的延迟微分方程求解算法和程序来求解。若想得出该方程的数值解,需要将其变换成一阶显式微分方程组。实现转换的最直观方法是引入一组状态变量 $x_1(t) = x(t)$, $x_2(t) = y(t)$, $x_3(t) = y'(t)$,这样可以得出下面给出的一阶微分方程组

$$\begin{cases} x_1'(t) = 1 - 3x_1(t) - x_2(t-1) - 0.2x_1^3(t-0.5) - x_1(t-0.5) \\ x_2'(t) = x_3(t) \\ x_3'(t) = 4x_1(t) - 2x_2(t) - 3x_3(t) \end{cases}$$

本方程可以定义两个时间常数 $\tau_1 = 1$, $\tau_2 = 0.5$ 。这样,由第一个方程可见,在其中需要 τ_1 延迟时间常数的是状态变量 x_2 ,即中间变量 $Z(2,1)$,而需要 τ_2 的状态变量是 x_1 ,即 $Z(1,2)$,所以应编写如下的匿名函数描述延迟微分方程

```
>> f=@(t,x,Z)[1-3*x(1)-Z(2,1)-0.2*Z(1,2)^3-Z(1,2); x(3); 4*x(1)-2*x(2)-3*x(3)];
```

由于该方程为常数初始值,所以可以直接在调用语句中使用零向量。这样,用下面的 MATLAB 语句可以立即得出该延迟微分方程的数值解,如图 7-18(a) 所示。

```
>> tau=[1 0.5]; tx=dde23(f,tau,zeros(3,1),[0,10]); plot(tx.x,tx.y(2,:))
```

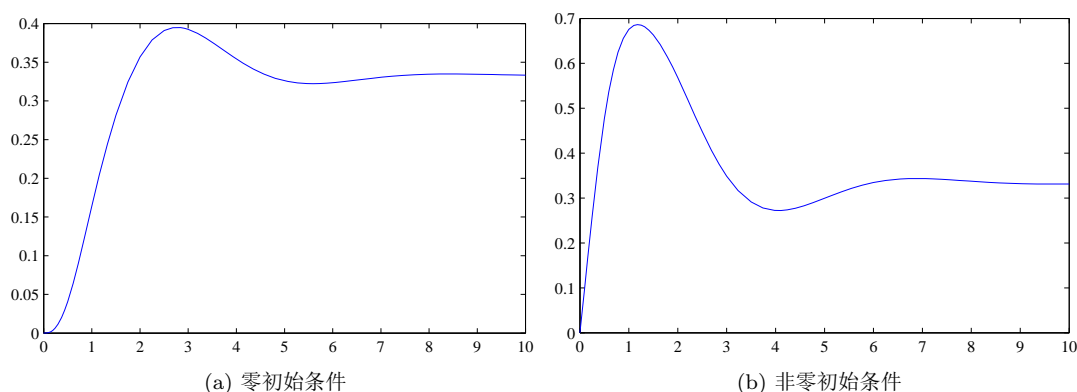


图 7-18 延迟微分方程的数值解

值得指出的是,如果用常数向量 x_0 表示历史函数 f_2 ,则意味着在 $t \leq t_0$ 时,状态变量的历史值始终都是 x_0 ,这在求解微分方程时应该格外注意。下面的例子将演示已知时变初始条件下延迟微分方程的解。

例 7-29 重新考虑例 7-28 中给出的延迟微分方程,若已知该微分方程三个状态变量的历史函数分别为 $x_1(t) = e^{2.1t}$, $x_2(t) = \sin t$, $x_3(t) = \cos t$, 其中 $t \leq 0$, 试重新求解该延迟微分方程。

解 用匿名函数的形式描述 $t \leq 0$ 时的初值方程,则可以由下面的语句直接求解原延迟微分方程,得出的结果如图 7-18(b) 所示。

```
>> f=@(t,x,Z)[1-3*x(1)-Z(2,1)-0.2*Z(1,2)^3-Z(1,2); x(3); 4*x(1)-2*x(2)-3*x(3)];
    f2=@(t,x)[exp(2.1*t); sin(t); cos(t)];
    lags=[1 0.5]; tx=dde23(f,lags,f2,[0,10]); plot(tx.x,tx.y(2,:))
```

7.5.2 变时间延迟微分方程的求解

MATLAB 提供的 `ddesd()` 函数可以用于时变延迟的微分方程,该函数的时间延迟向量允许使用变时间延迟的函数句柄,这样该函数完全可以处理带有时变延迟的延迟微分方程。当然这样的求解方法还可以扩展到由 `ddensd()` 函数求解中立型微分方程。`ddesd()` 函数的调用格式为 `sol = ddesd(f, fτ, f2, [t0, tf], options)`, 其中 f_{τ} 为时间延迟的函数句柄, f_2 为历史函数句柄,可以由 M-函数和匿名函数描述。

例 7-30 如果各个状态变量初始条件为零,试求解下面的变时间延迟微分方程。

$$\begin{cases} x_1'(t) = -2x_2(t) - 3x_1(t - 0.2|\sin t|) \\ x_2'(t) = -0.05x_1(t)x_3(t) - 2x_2(t - 0.8) + 2 \\ x_3'(t) = 0.3x_1(t)x_2(t)x_3(t) + \cos(x_1(t)x_2(t)) + 2\sin 0.1t^2 \end{cases}$$

解 显然,由于延迟微分方程中存在变时间延迟,即存在 $t - 0.2|\sin t|$ 时刻的 x_1 信号,所以不适合用 `dde23()` 函数求解。假设状态变量第一延迟为 $0.2|\sin t|$, 第二延迟为常数 0.8,并假设虚拟的状态变量导数延迟为空矩阵 [], 在 $t \leq 0$ 时状态变量的初值为零向量,这样可以用下面的语句直接求解变时间延迟的微分方程,得出方程的解如图 7-19 所示。

```
>> tau=@(t,x)[t-0.2*abs(sin(t)); t-0.8];
    f=@(t,x,Z)[-2*x(2)-3*Z(1,1); -0.05*x(1)*x(3)-2*Z(2,2)+2;
               0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*t^2)];
    sol=ddesd(f,tau,zeros(3,1),[0,10]); plot(sol.x,sol.y)
```

对该系统进行仿真,将得出如图 7-19 所示的数值解结果。可以测试不同的仿真控制参数,如相对误差限或仿真算法,以验证结果的正确性。

```
>> ff=odeset; ff.RelTol=1e-12; sol=ddesd(f,tau,zeros(3,1),[0,10],ff);
    hold on; plot(sol.x,sol.y)
```

值得指出的是,既然使用了匿名函数来描述变时间延迟,就应该注意,即使第 2 时间延迟为常数,也不能将该延迟简单地写成 0.8,必须写成 $t - 0.8$,否则将得出错误的结果——如果写成 0.8,求解函数会将该项认定为 $x_2(0.8)$,而不是 $x_2(t - 0.8)$ 。

例 7-31 如果前面给出的微分方程的历史值为 $x_1(t) = \sin(t + 1)$, $x_2(t) = \cos t$, $x_3(t) = e^{3t}$, $t \leq 0$, 试求解该微分方程。如果该微分方程的历史值在 $t < 0$ 时均为 0,只是在 $t = 0$ 时刻微分方程的初值满足前面的初值公式,试求解微分方程。

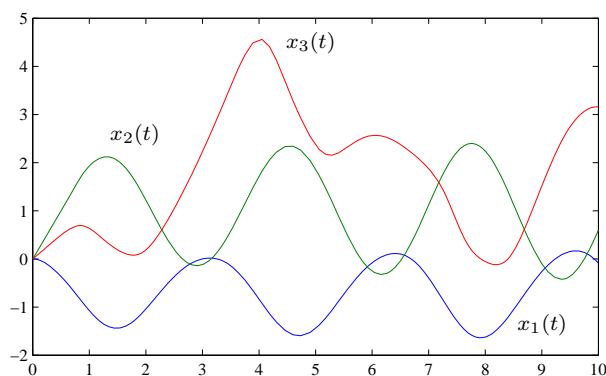
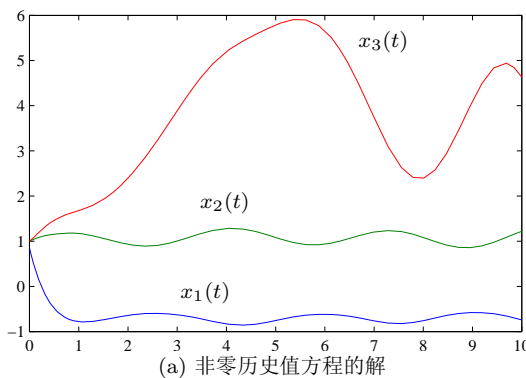


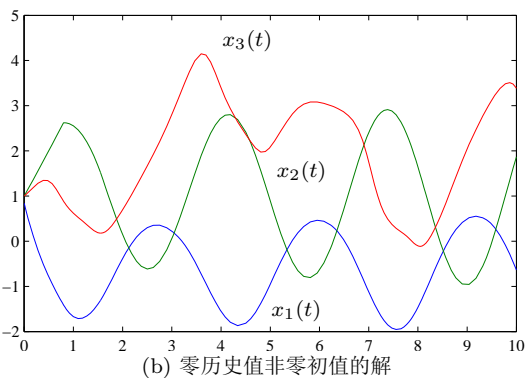
图 7-19 变时间延迟微分方程数值解

解 前面给出的历史值方程可以由匿名函数表示,这样原变延迟微分方程的数值解可以由下面的语句直接求出,如图 7-20(a) 所示。

```
>> tau=@(t,x)[t-0.2*abs(sin(t)); t-0.8];
f=@(t,x,Z)[-2*x(2)-3*Z(1,1); -0.05*x(1)*x(3)-2*Z(2,2)+2;
0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*t^2)];
f2=@(t,x)[sin(t+1); cos(t); exp(3*t)];
sol=ddesd(f,tau,f2,[0,10]); plot(sol.x,sol.y)
```



(a) 非零历史值方程的解



(b) 零历史值非零初值的解

图 7-20 非零初值延迟微分方程的数值解

如果状态变量的历史值为 0,仅在初始时刻 $t=0$ 时初始状态向量非零,则可以用下面的匿名函数描述历史值函数,这样可以得出微分方程的解,如图 7-20(b) 所示。可见,由于与前面介绍的方程仅仅是历史值不同,方程的解可能会有很大的差异。

```
>> f2=@(t,x)[sin(t+1); cos(t); exp(3*t)]*(t==0);
sol=ddesd(f,tau,f2,[0,10]); plot(sol.x,sol.y)
```

例 7-32 对前面给出的微分方程稍做改动,试重新求解下面的变时间延迟微分方程。

$$\begin{cases} x_1'(t) = -2x_2(t) - 3x_1(t - 0.2|\sin t|) \\ x_2'(t) = -0.05x_1(t)x_3(t) - 2x_2(\alpha t) + 2, \text{ 其中 } \alpha = 0.77 \\ x_3'(t) = 0.3x_1(t)x_2(t)x_3(t) + \cos(x_1(t)x_2(t)) + 2\sin 0.1t^2 \end{cases}$$

解 可见第2个方程中含有 $x_2(0.77t)$ 项,说明该方程含有 x_2 信号在 $0.77t$ 处的值,则应该采用下面的语句求解此微分方程,得出的结果如图 7-21 所示。

```
>> tau=@(t,x)[t-0.2*abs(sin(t)); 0.77*t];
f=@(t,x,Z)[-2*x(2)-3*Z(1,1); -0.05*x(1)*x(3)-2*Z(2,2)+2;
0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*t^2)];
sol=ddesd(f,tau,zeros(3,1),[0,10]); plot(sol.x,sol.y)
```

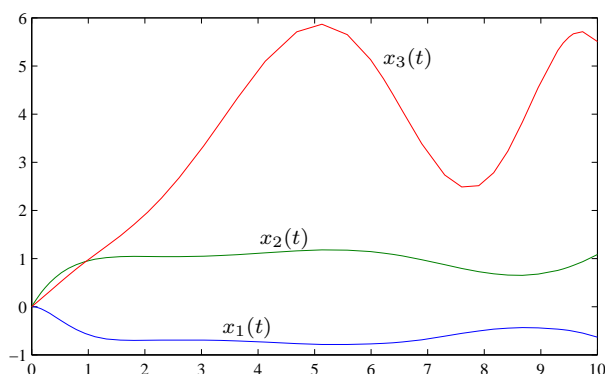


图 7-21 变延迟微分方程数值解

如果延迟微分方程的延迟时刻描述向量中含有当前时刻 t 之后的值,如例 7-32 中的 $\alpha = 1.1$,说明该微分方程含有未来时刻的值,目前没有任何算法可以直接求解这样的方程。虽然用户可将描述延迟函数的句柄写成 $\text{tau} = @(t,x)[t-0.2*\text{abs}(\sin(t)); 1.1*t]$, $\text{ddesd}()$ 函数也无法正常求解。该求解函数会使用 $x(t)$ 去取代 $x(1.1t)$ 的值。

7.5.3 中立型延迟微分方程的求解

中立型(neutral-type)延迟微分方程的一般表示形式为

$$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{x}(t-\tau_{p_1}), \dots, \mathbf{x}(t-\tau_{p_m}), \mathbf{x}'(t-\tau_{q_1}), \dots, \mathbf{x}'(t), \dots, \mathbf{x}'(t-\tau_{q_k})) \quad (7-5-2)$$

其中既包括了状态变量的延迟信号,又包括了状态变量导数的延迟信号,这可以由两个常数向量 $\boldsymbol{\tau}_1 = [\tau_{p_1}, \tau_{p_2}, \dots, \tau_{p_m}]$, $\boldsymbol{\tau}_2 = [\tau_{q_1}, \tau_{q_2}, \dots, \tau_{q_k}]$ 来表示。中立型延迟微分方程不能用 $\text{dde23}()$ 求解,但可以使用新版本(MATLAB 8.0)中的 $\text{ddensd}()$ 函数求解,该函数的调用格式为

```
sol = ddensd(f, tau1, tau2, f2, [t0, tf], options)
```

如果该微分方程的延迟不是固定的常数,则可以仿照 $\text{ddesd}()$ 函数,将 $\boldsymbol{\tau}_1$ 和 $\boldsymbol{\tau}_2$ 表示成函数句柄,既可以用匿名函数来描述,也可以用 M-函数来描述。

例 7-33 试求解下面给出的中立型延迟微分方程

$$\mathbf{x}'(t) = \mathbf{A}_1 \mathbf{x}(t - 0.15) + \mathbf{A}_2 \mathbf{x}'(t - 0.5) + \mathbf{B}u(t)$$

其中,输入信号 $u(t) \equiv 1$,且已知矩阵为

$$A_1 = \begin{bmatrix} -13 & 3 & -3 \\ 106 & -116 & 62 \\ 207 & -207 & 113 \end{bmatrix}, A_2 = \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0.03 & 0 \\ 0 & 0 & 0.04 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

解 因为方程中同时包含 $x'(t)$ 和 $x'(t-0.5)$ 项,所以单纯采用 `dde23()` 是无能为力的,而需要引入 `ddensd()` 函数直接求解。这里,状态信号的延迟为 $\tau_1 = 0.15$,状态变量导数的延迟为 $\tau_2 = 0.5$,这样可以由下面匿名函数描述中立型延迟微分方程,然后可以由下面语句直接求解该微分方程,得出状态变量的时间响应曲线,如图 7-22 所示。

```
>> A1=[-13,3,-3; 106,-116,62; 207,-207,113]; u=1;
A2=diag([0.02,0.03,0.04]); B=[0; 1; 2];
f=@(t,x,z1,z2)A1*z1+A2*z2+B*u; x0=zeros(3,1);
sol=ddensd(f,0.15,0.5,x0,[0,15]); plot(sol.x,sol.y)
```

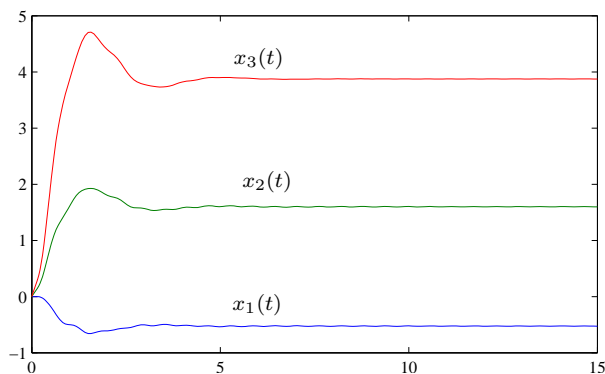


图 7-22 延迟微分方程的数值解

例 7-34 重新考虑例 7-32 中的变延迟非中立型微分方程,试重新求解该方程。

解 可以考虑采用 `ddensd()` 函数求解此微分方程,这时需假设状态变量导数的延迟为空矩阵 `[]`,这样可以改用下面的语句求解原始的微分方程,得出的结果与图 7-19 完全一致。

```
>> f=@(t,x,Z,z)[-2*x(2)-3*Z(1,1); -0.05*x(1)*x(3)-2*Z(2,2)+2;
0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*t^2)];
sol=ddensd(f,tau,[],zeros(3,1),[0,10]); plot(sol.x,sol.y)
```

7.6 边值问题的计算机求解

前面的微分方程数值解中侧重研究初值问题,即已知 x_0 对其他时刻状态变量值进行求解的方法。在实际应用中,经常会遇到这样的问题,已知部分状态在 $t = 0$ 时刻的值,还知道部分状态在 $t = t_f$ 时刻的值,这类问题即所谓的边值问题。边值问题也是 `ode45()` 类函数无法直接求解的一类问题。本节将讨论边值问题的计算机求解方法。

二阶微分方程边值问题的数学描述为

$$y''(x) = F[x, y(x), y'(x)] \quad (7-6-1)$$

假设想在区间 $[a, b]$ 上研究该方程的解, 且已知在这两个边界点上满足边界条件

$$\alpha_a y(a) + \beta_a y'(a) = \gamma_a, \quad \alpha_b y(b) + \beta_b y'(b) = \gamma_b \quad (7-6-2)$$

显然, 前面的初值问题算法是不能直接使用的, 因为并不能直接获得在初始时刻的各个变量的值。下面将讨论各种边值问题的数值解法。

7.6.1 线性方程边值问题的打靶算法

首先考虑一种简单的情况, 即线性微分方程的边值问题求解的数值算法。考虑下面给出的微分方程

$$y''(t) + p(t)y'(t) + q(t)y(t) = f(t) \quad (7-6-3)$$

其中, $p(t)$, $q(t)$ 和 $f(t)$ 均为给定函数, 则式 (7-6-2) 可以进一步简化成

$$y(a) = \gamma_a, \quad y(b) = \gamma_b \quad (7-6-4)$$

打靶算法 (shooting method) 的基本想法是找出能够满足式 (7-6-2) 边值的相应初值 $y(0)$ 和 $y'(0)$, 然后再利用前面介绍的初值算法去求解这一初值问题, 这里不加证明地给出线性系统边值问题的打靶算法的主要计算步骤为

(1) 求出下面方程初值问题的数值解 $y_1(b)$

$$y_1''(t) + p(t)y_1'(t) + q(t)y_1(t) = 0, \quad y_1(a) = 1, \quad y_1'(a) = 0 \quad (7-6-5)$$

(2) 求出下面方程初值问题的数值解 $y_2(b)$

$$y_2''(t) + p(t)y_2'(t) + q(t)y_2(t) = 0, \quad y_2(a) = 0, \quad y_2'(a) = 1 \quad (7-6-6)$$

(3) 求出下面方程初值问题的数值解 $y_p(b)$

$$y_p''(t) + p(t)y_p'(t) + q(t)y_p(t) = f(t), \quad y_p(a) = 0, \quad y_p'(a) = 1 \quad (7-6-7)$$

(4) 若 $y_2(b) \neq 0$, 则计算 $m = \frac{\gamma_b - \gamma_a y_1(b) - y_p(b)}{y_2(b)}$ (7-6-8)

(5) 计算下面初值问题的数值解, 则 $y(x)$ 即为原边值问题的数值解

$$y''(t) + p(t)y'(t) + q(t)y(t) = f(t), \quad y(a) = \gamma_a, \quad y'(a) = m \quad (7-6-9)$$

式 (7-6-5)、式 (7-6-6) 对应的是系统的微分方程组

$$\begin{cases} x_1' = x_2 \\ x_2' = -q(t)x_1 - p(t)x_2 \end{cases} \quad (7-6-10)$$

初始状态向量分别为 $[1, 0]^T$ 、 $[0, 1]^T$ 。式 (7-6-7)、式 (7-6-9) 对应的是下面的微分方程组

$$\begin{cases} x_1' = x_2 \\ x_2' = -q(t)x_1 - p(t)x_2 + f(t) \end{cases} \quad (7-6-11)$$

其初始状态向量分别为 $[0, 0]^T$ 、 $[\gamma_a, m]^T$ 。由上面的算法可以编写出如下 MATLAB 函数

```
function [t,y]=ln_shooting(p,q,f,tspan,x0f,varargin)
if isnumeric(p), p=@(t)p; end, if isnumeric(q), q=@(t)q; end
if isnumeric(f), f=@(t)f; end, f1=@(t,x)[x(2); -q(t)*x(1)-p(t)*x(2)];
f2=@(t,x)f1(t,x)+[0;f(t)]; t0=tspan(1); tfinal=tspan(2); ga=x0f(1);gb=x0f(2);
[t,y1]=ode45(f1,tspan,[1; 0],varargin{:});
[t,y2]=ode45(f1,tspan,[0; 1],varargin{:});
[t,yp]=ode45(f2,tspan,[0; 0],varargin{:});
m=(gb-ga*y1(end,1)-yp(end,1))/y2(end,1);
[t,y]=ode45(f2,tspan,[ga;m],varargin{:});
```

该函数的调用格式为 $[t,y] = \text{ln_shooting}(p,q,f,tspan,x_{0f})$, 其中 p, q, f 对应于公式中相应函数的句柄, 如果某函数为常数则可以直接将其表示成常数。 $tspan = [a,b]$ 为计算区间, 而 $x_{0f} = [\gamma_a, \gamma_b]$ 为边值向量。该函数还可以将 `options` 和附加常数作为输入变量, 其格式与 `ode45()` 等函数一致。下面将通过简单的例子介绍该函数的使用。

例 7-35 试求线性微分方程 $y'' - 3y' + 2y = t$ 在 $[0, 1]$ 区间的数值解, 其中 $y(0) = 1, y(1) = 2$ 。

解 原来微分方程的解析解可以通过下面语句直接求解

```
>> syms t; y1=dsolve('D2y-3*Dy+2*y=t','y(0)=1','y(1)=2')
```

这样得出的解析解为

$$\frac{t}{2} + \frac{e^{2t}(e-3)}{4(e-e^2)} - \frac{e^t(e^2-3)}{4(e-e^2)} + \frac{3}{4}$$

采用下面的语句就可以求出原方程的解, 并将其绘制出来, 如图 7-23 所示。可见, 得出的 $x_1(t)$ 函数满足给定的边界条件

```
>> p=-3; q=2; f=@(t)t; [t0,y]=ln_shooting(p,q,f,[0,1],[1;2]); plot(t0,y)
```

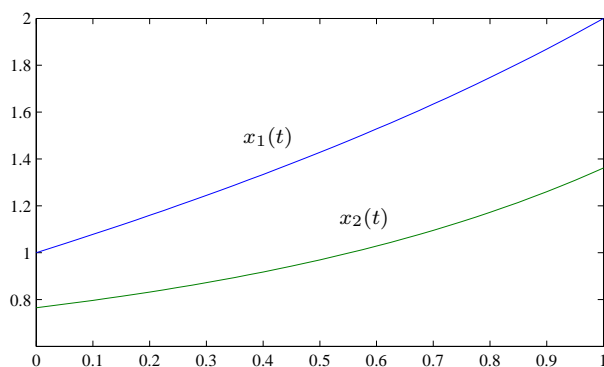


图 7-23 线性两点边值问题的解

可以由下面的语句求出上面得出结果的精度。由检验结果可见, $e_1 = 4.48 \times 10^{-8}, e_2 = 2.26 \times 10^{-8}$, 这样求出的解精度还是较高的。

```
>> y0=subs(y1,t,t0); e1=norm(y(:,1)-y0), e2=norm(y(end,1)-2)
```

如果想进一步提高精度, 则需要调入微分方程求解的控制模板 `odeset`, 提高求解精度要求, 重新求解边值问题, 这时的 $e_1 = 3.52 \times 10^{-14}, e_2 = 2.89 \times 10^{-15}$ 。

```
>> opt=odeset; opt.RelTol=1e-20; opt.AbsTol=1e-20;
[t0,y]=ln_shooting(p,q,f,[0,1],[1;2],opt);
y0=subs(y1,t,t0); e1=norm(y(:,1)-y0), e2=norm(y(end,1)-2)
```

例 7-36 试求出微分方程 $y''(x) - \left(2 - \frac{1}{x}\right)y'(x) + \left(1 - \frac{1}{x}\right)y(x) = x^2e^{-5x}$ 的数值解, 已知边界条件为 $y(1) = \pi$, $y(\pi) = 1$ 。

解 给出的方程解析解是不可求的, 只能采用数值解方法。对照给出的微分方程和本例的微分方程, 可以立即得出这三个函数 p, q, f , 用匿名函数描述这三个函数, 可以得出微分方程的数值解, 其时域响应曲线如图 7-24 所示。

```
>> p=@(x)(2-1./x); q=@(x)1-1./x; f=@(x)x.^2.*exp(-5*x);
[t,y]=ln_shooting(p,q,f,[1,pi],[pi; 1]); plot(t,y)
```

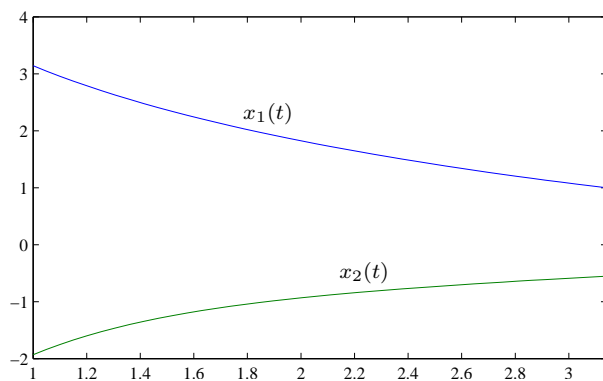


图 7-24 线性两点边值问题的解

7.6.2 非线性方程边值问题的打靶算法

前面对一类特殊的微分方程(即线性微分方程)边值问题进行了探讨, 但这种方法并不能求解式(7-6-1)和式(7-6-2)所描述的一般问题。这里给出对此一般问题的数值解法。

假定原始问题可以转换成下面的初值问题

$$y'' = F(x, y, y'), \quad y(a) = \gamma_a, \quad y'(a) = m \quad (7-6-12)$$

则问题转换成求解 $y(m; b) = \gamma_b$, 可以采用下面的 Newton-Raphson 迭代法来求取 m

$$m_{i+1} = m_i - \frac{y(m_i; b) - \gamma_b}{(\partial y / \partial m)(m_i; b)} = m_i - \frac{v_1(b) - \gamma_b}{v_3(b)} \quad (7-6-13)$$

式中, $v_1 = y(m_i; x)$, $v_2 = y'(m_i; x)$, $v_3 = (\partial y / \partial m)(m_i; x)$, $v_4 = (\partial y' / \partial m)(m_i; x)$, 且可以由

下面的微分方程初值问题来求解

$$\begin{cases} v_1' = v_2, & v_1(a) = \gamma_a \\ v_2' = F(x, v_1, v_2), & v_2(a) = m \\ v_3' = v_4, & v_3(a) = 0 \\ v_4' = \frac{\partial F}{\partial y}(x, v_1, v_2)v_3 + \frac{\partial F}{\partial y'}(x, v_1, v_2)v_4, & v_4(a) = 1 \end{cases} \quad (7-6-14)$$

其中,要求能显式地求出 $\partial F/\partial y$, $\partial F/\partial y'$ 。在具体计算中可以指定一个 m 值,然后求解式(7-6-14)中的初值问题,将结果代入式(7-6-13)迭代一步,并将结果代入式(7-6-14)重新计算,直至两次计算出来的 m 值的误差在允许的范围内,则可以采用此 m 值,最后代入式(7-6-12)求解原始问题。上面算法的 MATLAB 实现为

```
function [t,y]=nlbound(funcn,funcv,tspan,x0f,varargin)
t0=tspan(1); tfinal=tspan(2); ga=x0f(1); gb=x0f(2); m=1; m0=0; tol=eps;
if nargin>4, kk=varargin{1}.RelTol; if ~isempty(kk), tol=kk; end, end
while (norm(m-m0)>tol), m0=m;
    [t,v]=ode45(funcv,tspan,[ga;m;0;1],varargin{:});
    m=m0-(v(end,1)-gb)/(v(end,3));
end
[t,y]=ode45(funcn,tspan,[ga; m],varargin{:});
```

其中,用户必须自己编写一个 `funcv()` 函数来描述式(7-6-14)中的初值问题。

例 7-37 试求解非线性微分方程边值问题 $y'' = F(x, y, y') = 2yy'$, $y(0) = -1$, $y(\pi/2) = 1$ 。

解 可以容易地求出偏导数 $\partial F/\partial y = 2y'$, $\partial F/\partial y' = 2y$, 代入式(7-6-14)中的第 4 个式子可以立即得出 $v_4' = 2v_2v_3 + 2v_1v_4$, 这样,原方程和本方程的 MATLAB 匿名函数表示分别为

```
>> f2=@(t,v)[v(2); 2*v(1)*v(2)];
f1=@(t,v)[f2(t,v(1:2)); v(4); 2*v(2)*v(3)+2*v(1)*v(4)];
```

这时,可以通过下面的 MATLAB 语句来求解原始问题,得出的结果如图 7-25 所示。可见,解出的 $x_1(t)$ 函数满足给定的边界要求。

```
>> opt=odeset; opt.RelTol=1e-10; opt.AbsTol=1e-10;
[t,y]=nlbound(f2,f1,[0,pi/2],[-1,1],opt); plot(t,y);
```

该问题的解析解为 $y(x) = \tan(x - \pi/4)$, 得出的误差向量范数为 2.26×10^{-10} 。可见,这样得出的数值解和解析解极其接近,且满足原始边值条件。

```
>> y0=tan(t-pi/4); e1=norm(y(:,1)-y0)
```

7.6.3 一般边值微分方程的求解方法

由前面的叙述可见,两点边值求解方法的局限性较大,对原来的方程有许多约束,如阶次约束、边值约束等。如果要求解更复杂的边值问题,必须利用其他的方法。假设要研究的微分方程为

$$y' = f(t, y, \theta) \quad (7-6-15)$$

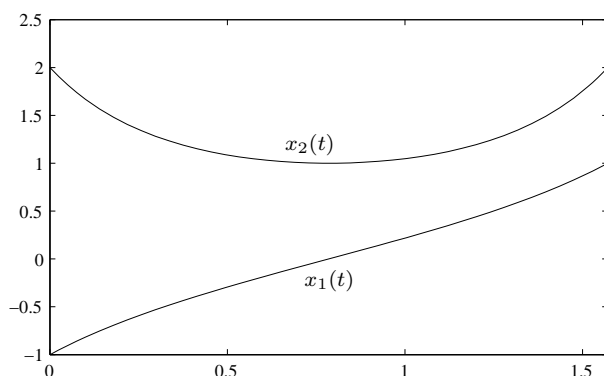


图 7-25 线性两点边值问题的解

其中 \mathbf{y} 为状态变量向量, $\boldsymbol{\theta}$ 为方程中其他未知参数向量。该方程已知的边界值为

$$\phi[\mathbf{y}(a), \mathbf{y}(b), \boldsymbol{\theta}] = \mathbf{0} \quad (7-6-16)$$

如果想将原始问题变换成初值问题,则需要求解若干个代数方程。若需要求解的变量个数和这样建立起来的方程个数相同时,则可以通过求解方程的方法先将它们求解出来,然后求解微分方程。和典型的边值问题相比,这里研究的方程求解更具一般性,因为除了传统的边值问题之外,还可以求解其他的未知参数问题。

MATLAB 提供的 `bvp5c()` 函数^[7]可以很好地求解微分方程的边值问题。正确求解一个常微分方程的边值问题,一般应该经过以下几个步骤:

(1) **参数初始化。**调用 `bvpinit()` 函数即可输入信息。当然这样的描述决不仅仅局限于边值,其他待定变量也可以在这里一起描述,其调用格式为 `sinit = bvpinit(v, x0, theta0)`, 其中, \mathbf{v} 应该包含测试的时间向量,可以用 `v = linspace(a, b, M)` 生成,注意为保障足够快的计算速度, M 不能取得过大,一般取 $M = 5$ 即可。除了 \mathbf{v} 向量,当然还应该给出状态变量初值 \mathbf{x}_0 和待定参数 $\boldsymbol{\theta}_0$ 的初始搜索点。

(2) **微分方程和边值问题的 MATLAB 函数描述。**微分方程本身的描述和初值问题完全一致,边值问题描述出式(7-6-16)中的各个式子即可,具体格式将由下面的例子演示。

(3) **边值问题的求解。**调用 `bvp5c()` 函数就可以直接求解边值问题了

`sol = bvp5c(@fun1, @fun2, sinit, options, 附加参数)`

其中, `fun1.m` 和 `fun2.m` 分别为描述微分方程和边值条件的 MATLAB 函数,当然它们也可以通过匿名函数直接表示。返回的 `sol` 为结构体型变量,其 `sol.x` 为 t 向量, `sol.y` 的每一行对应一个状态变量。`sol.parameters` 将返回待定参数 $\boldsymbol{\theta}$ 。

后面将通过例子介绍该函数的编写方法。另外,还需要编写一个函数来描述一阶微分方程组,这和以前微分方程组的描述是完全一致的。

例 7-38 试用 `bvp5c()` 函数重新求解例 7-37 中的边值问题。

解 令 $x_1 = y, x_2 = y'$, 则可以得出一阶显式微分方程为 $x'_1 = x_2, x'_2 = 2x_1x_2$, 这里采用了匿名函数的形式描述微分方程和边值条件,其效果和编写 MATLAB 函数是完全一致的。可以由下面的

语句直接求解本边值问题,结果曲线如图 7-26 (a) 所示。得出的结果和前面例子是一致的。

```
>> f1=@(t,x)[x(2); 2*x(1)*x(2)]; f2=@(xa,xb)[xa(1)+1; xb(1)-1];
sinit=bvpinit(linspace(0,pi/2,5),rand(2,1));
sol=bvp5c(f1,f2,sinit); plot(sol.x,sol.y)
```

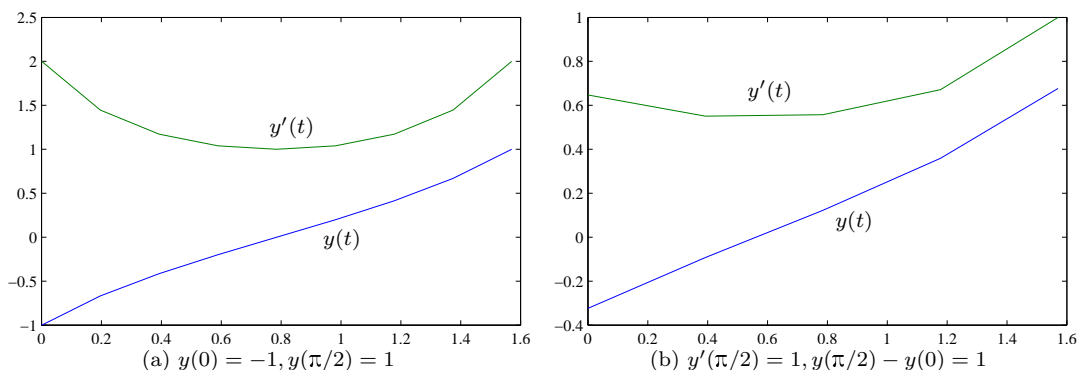


图 7-26 边值问题的解

利用 `bvp5c()` 函数还可以解决更复杂的边值问题,例如若边值问题修改成 $y'(\pi/2) = 1$, $y(\pi/2) - y(0) = 1$,则可以如下修改 `f2`,并求解该方程,得出如图 7-26 (b) 所示的结果。

```
>> f2=@(xa,xb)[xb(2)-1; xb(1)-xa(1)-1];
sol=bvp5c(f1,f2,sinit); plot(sol.x,sol.y)
```

例 7-39 已知某常微分方程模型如下,试求出 α, β 并求解本微分方程。

$$\begin{cases} x' = 4x - \alpha xy \\ y' = -2y + \beta xy, \end{cases} \quad \text{且已知 } x(0) = 2, y(0) = 1, x(3) = 4, y(3) = 2$$

解 先引入状态变量 $x_1 = x, x_2 = y$,则可以将原问题转换成关于 x 的微分方程。另外,令 $v_1 = \alpha, v_2 = \beta$ 。和边值问题一样,需要描述系统的动态模型和边值问题如下

```
>> f=@(t,x,v)[4*x(1)+v(1)*x(1)*x(2); -2*x(2)+v(2)*x(1)*x(2)];
g=@(ya,yb,v)[ya(1)-2; ya(2)-1; yb(1)-4; yb(2)-2];
```

从表示的函数可以看出,边值的描述还是很直观的。这时,可以先调用 `bvpinit()` 初始化函数来定义求解时间段及网格划分方法,并令状态初始值和参数 α, β 的初始值,因为有两个初始状态,两个未定参数,所以它们的初值均可以设置为随机数向量 `rand(2,1)`。定义了这些参数,则可以调用 `bvp5c()` 函数来求解边值问题的 α, β 参数,并求解在此参数下的系统方程

```
>> x1=[1;1]; x2=[-1;1]; sinit=bvpinit(linspace(0,3,20),x1,x2);
sol=bvp5c(f,g,sinit); sol.parameters % 显示待定参数
plot(sol.x,sol.y); figure; plot(sol.y(1,:),sol.y(2,:));
```

得出的结果如图 7-27 所示,同时可以求出 $\alpha = -2.3721, \beta = 0.8934$ 。由得出的仿真曲线可以看出,方程状态的边值条件可以满足,所以求出的解是正确的。这里的初值向量 x_1 和 x_2 选择应该注意,如果选择不当可能使得求解过程中的 Jacobi 矩阵奇异,所以实际求解时若出现此现象,则应该选择其他的初值。

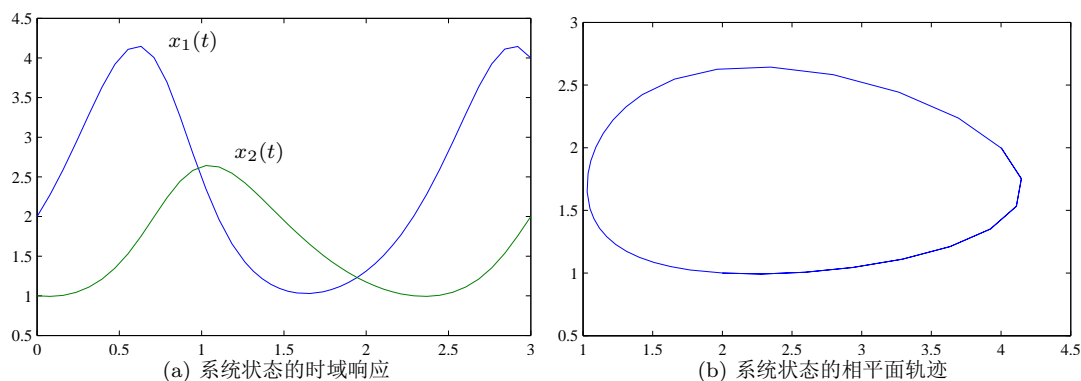


图 7-27 微分方程的数值解表示

7.7 偏微分方程求解入门

MATLAB 可以求解一般的偏微分方程,也可以用偏微分方程工具箱中给出的相应函数求解一些偏微分方程。本节将首先介绍一般偏微分方程的数值解法,然后介绍利用偏微分方程工具箱求解几类典型偏微分方程的方法。

7.7.1 偏微分方程组求解

MATLAB 语言提供了 `pdepe()` 函数,可以直接求解偏微分方程

$$c\left(x, t, u, \frac{\partial u}{\partial x}\right) \frac{\partial u}{\partial t} = x^{-m} \frac{\partial}{\partial x} \left[x^m f\left(x, t, u, \frac{\partial u}{\partial x}\right) \right] + s\left(x, t, u, \frac{\partial u}{\partial x}\right) \quad (7-7-1)$$

这样,偏微分方程可以编写下面的函数描述,其入口为 `[c, f, s] = pdefun(x, t, u, u_x)`,其中, `pdefun` 为函数名。这样,由给定的输入变量即可计算出 `c, f, s` 这 3 个函数。由于需要返回的变量个数多于 1 个,所以不能采用匿名函数的格式,只能编写 M-函数。

边界条件可以用下面的函数描述为

$$p(x, t, u) + q(x, t, u) \cdot f\left(x, t, u, \frac{\partial u}{\partial x}\right) = 0 \quad (7-7-2)$$

并由此派生出 (a, b) 区间端点上的边界条件为

$$p(a, t, u) + q(a, t, u) \cdot f\left(a, t, u, \frac{\partial u}{\partial x}\right) = 0 \quad (7-7-3)$$

$$p(b, t, u) + q(b, t, u) \cdot f\left(b, t, u, \frac{\partial u}{\partial x}\right) = 0 \quad (7-7-4)$$

这样的边值函数可以由 MATLAB 函数描述 `[p_a, q_a, p_b, q_b] = pdebc(x, t, u, u_x)`。

除了这两种函数外,还应该写出初始条件函数。偏微分方程初始条件的数学描述为 $u(x, t_0) = u_0$ 。这样,需要一个简单的函数来描述,编写简单函数 `u_0 = pdeic(x)` 即可。

还可以选择 x 和 t 的向量,再加上描述的这些函数,就可以用 `pdepe()` 函数求解次偏微分方程,这需要用下面的格式求解该偏微分方程。

```
sol = pdepe(m,@pdefun,@pdeic,@pdebc,x,t)
```

例 7-40 试求解下面的偏微分方程^[8]

$$\begin{cases} \frac{\partial u_1}{\partial t} = 0.024 \frac{\partial^2 u_1}{\partial x^2} - F(u_1 - u_2) \\ \frac{\partial u_2}{\partial t} = 0.17 \frac{\partial^2 u_2}{\partial x^2} + F(u_1 - u_2) \end{cases}$$

其中, $F(x) = e^{5.73x} - e^{-11.46x}$, 且满足初始条件 $u_1(x, 0) = 1$, $u_2(x, 1) = 0$ 及边界条件

$$\frac{\partial u_1}{\partial x}(0, t) = 0, u_2(0, t) = 0, u_1(1, t) = 1, \frac{\partial u_2}{\partial x}(1, t) = 0$$

解 对照给出的偏微分方程和式(7-7-1),则可以将原方程改写为

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \frac{\partial}{\partial t} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \frac{\partial}{\partial x} \begin{bmatrix} 0.024 \partial u_1 / \partial x \\ 0.17 \partial u_2 / \partial x \end{bmatrix} + \begin{bmatrix} -F(u_1 - u_2) \\ F(u_1 - u_2) \end{bmatrix}$$

可见, $m = 0$, 且

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, f = \begin{bmatrix} 0.024 \partial u_1 / \partial x \\ 0.17 \partial u_2 / \partial x \end{bmatrix}, s = \begin{bmatrix} -F(u_1 - u_2) \\ F(u_1 - u_2) \end{bmatrix}$$

这样,可以编写出下面的描述偏微分方程的 MATLAB 函数为

```
function [c,f,s]=c7mpde(x,t,u,du)
c=[1; 1]; y=u(1)-u(2); F=exp(5.73*y)-exp(-11.46*y); s=[-F; F];
f=[0.024*du(1); 0.17*du(2)];
```

套用式(7-7-2)中的边界条件,可以写出如下的边值方程

$$\text{左边界} \begin{bmatrix} 0 \\ u_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \text{右边界} \begin{bmatrix} u_1 - 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

对照已知的标准型可以得出, $p_a = [0, u_2(a)]^T$, $q_a = [1, 0]^T$, $p_b = [u_1(b) - 1, 0]^T$, $q_b = [0, 1]^T$, 从而编写出下面描述边界条件的 MATLAB 函数

```
function [pa,qa,pb,qb]=c7mpbc(xa,ua,xb,ub,t)
pa=[0; ua(2)]; qa=[1;0]; pb=[ub(1)-1; 0]; qb=[0;1];
```

另外,还可以立即写出描述初值的 MATLAB 匿名函数为

```
>> u0=@(x)[1; 0];
```

有了这 3 个函数,选定 x 和 t 向量,则可以由下面的语句直接求解此偏微分方程,得出解 u_1 和 u_2 ,如图 7-28(a)、(b) 所示。

```
>> x=0:0.05:1; t=0:0.05:2; m=0; sol=pdepe(m,@c7mpde,u0,@c7mpbc,x,t);
surf(x,t,sol(:,:,1)), figure; surf(x,t,sol(:,:,2))
```

7.7.2 二阶偏微分方程的数学描述

除了前面介绍的一类偏微分方程外, MATLAB 语言还有自己的偏微分方程工具箱,可以比较规范地求解各种常见的二阶偏微分方程。这里将 MATLAB 偏微分方程工具箱可解

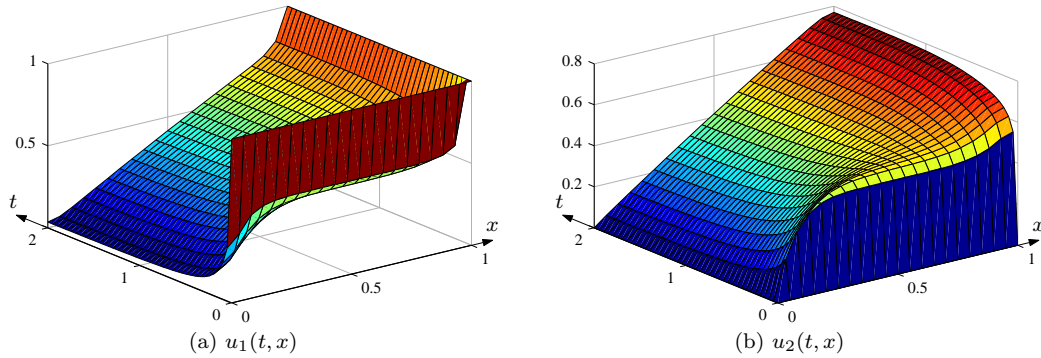


图 7-28 偏微分方程的解曲面

的二阶偏微分方程简单介绍一下,后面再介绍一个实用的偏微分方程求解界面 `pdetool`, 利用该程序界面可以容易地求解偏微分方程。

1. 椭圆型偏微分方程

椭圆型偏微分方程的一般表示形式为

$$-\operatorname{div}(c\nabla u) + au = f(\mathbf{x}, t) \quad (7-7-5)$$

其中,若 $u = u(x_1, x_2, \dots, x_n, t) = u(\mathbf{x}, t)$, ∇u 为 u 的梯度,则其定义为

$$\nabla u = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \dots, \frac{\partial}{\partial x_n} \right] u \quad (7-7-6)$$

散度 $\operatorname{div}(v)$ 的定义为

$$\operatorname{div}(v) = \left(\frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2} + \dots + \frac{\partial}{\partial x_n} \right) v \quad (7-7-7)$$

这样, $\operatorname{div}(c\nabla u)$ 可以更明确地表示成

$$\operatorname{div}(c\nabla u) = \left[\frac{\partial}{\partial x_1} \left(c \frac{\partial u}{\partial x_1} \right) + \frac{\partial}{\partial x_2} \left(c \frac{\partial u}{\partial x_2} \right) + \dots + \frac{\partial}{\partial x_n} \left(c \frac{\partial u}{\partial x_n} \right) \right] \quad (7-7-8)$$

若 c 为常数,则该项可以进一步化简为

$$\operatorname{div}(c\nabla u) = c \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \dots + \frac{\partial^2}{\partial x_n^2} \right) u = c\Delta u \quad (7-7-9)$$

其中, Δ 又称为 Laplace 算子。这样,椭圆型偏微分方程可以更简单地写成

$$-c \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \dots + \frac{\partial^2}{\partial x_n^2} \right) u + au = f(\mathbf{x}, t) \quad (7-7-10)$$

2. 抛物型偏微分方程

抛物型偏微分方程的一般形式为

$$d \frac{\partial u}{\partial t} - \operatorname{div}(c\nabla u) + au = f(\mathbf{x}, t) \quad (7-7-11)$$

根据上面的叙述,若 c 为常数,则该方程可以更简单地写成

$$d \frac{\partial u}{\partial t} - c \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2} \right) + au = f(\mathbf{x}, t) \quad (7-7-12)$$

3. 双曲型偏微分方程

双曲型偏微分方程的一般形式为

$$d \frac{\partial^2 u}{\partial t^2} - \operatorname{div}(c \nabla u) + au = f(\mathbf{x}, t) \quad (7-7-13)$$

若 c 为常数,则可以将该方程简化成

$$d \frac{\partial^2 u}{\partial t^2} - c \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2} \right) + au = f(\mathbf{x}, t) \quad (7-7-14)$$

从上面的 3 种类型方程可以看出,它们直接的区别在于 u 函数对 t 的导数阶次。如果对 t 没有求导,则可以理解为其值为常数,故称为椭圆型偏微分方程。如果取 u 对时间的一阶导数,则一阶导数与 u 对 \mathbf{x} 的二阶导数直接构成了抛物线关系,故称其为抛物型偏微分方程。如果对 t 取二阶导数,则可以称之为双曲型偏微分方程。

MATLAB 的偏微分方程工具箱采用的是有限元方法求解各种偏微分方程的。椭圆型偏微分方程求解中, c, a, d, f 均可以为给定函数的形式,但其他类型偏微分方程求解时,它们必须为常数。

4. 特征值型偏微分方程

MATLAB 可以直接求解的另一类偏微分方程为特征值型偏微分方程

$$-\operatorname{div}(c \nabla u) + au = \lambda du \quad (7-7-15)$$

对常数 c , 该方程还可以简化成

$$-c \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2} \right) + au = \lambda du \quad (7-7-16)$$

对比式 (7-7-16) 与式 (7-7-5) 可以发现,将前者等号右侧的 λdu 移动到方程的左侧,就可以变换成一般的椭圆型偏微分方程,所以该方程是椭圆型偏微分方程的一个特例。

7.7.3 偏微分方程的求解界面应用举例

1. 偏微分方程求解程序概述

MATLAB 偏微分方程工具箱提供了一个界面,可以求解二元偏微分方程 $u(x_1, x_2)$, 这时求解区域可以用该界面提供的绘制圆、椭圆、矩形及多边形等工具任意绘制,也可以由若干个这样简单绘制的集合进行并集、交集、差集等构成所需的求解区域。完成求解区域的绘制后,还可以用该界面提供的功能将原求解区域用三角剖分的形式自动绘制出网格。

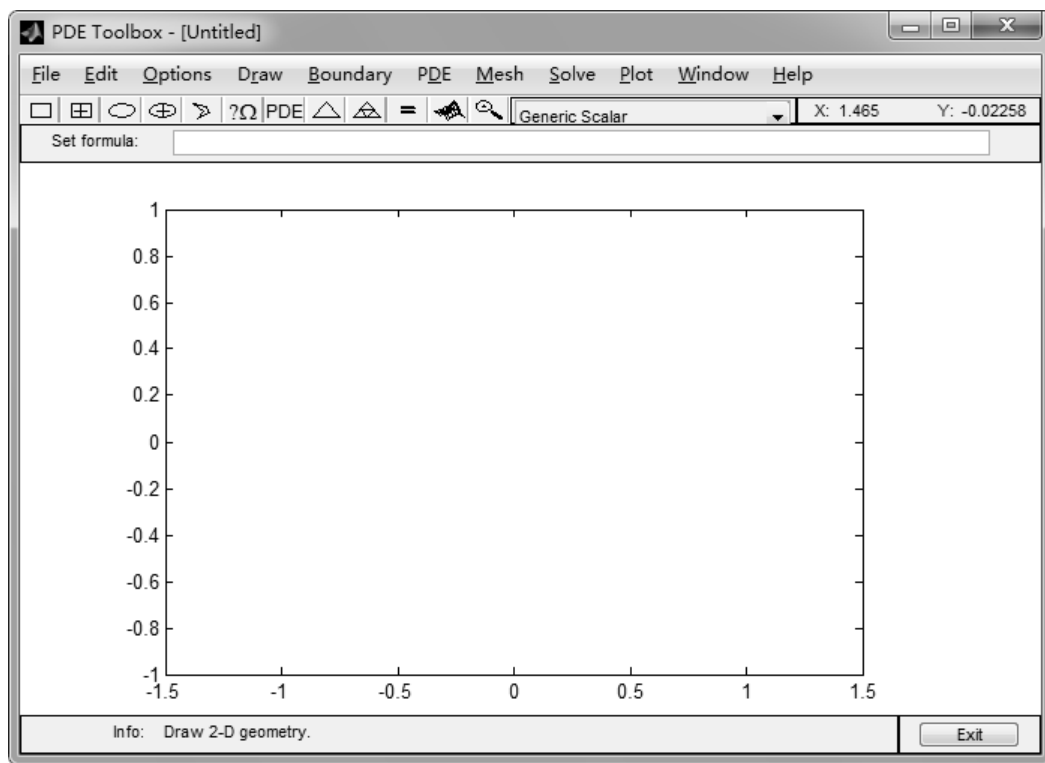


图 7-29 偏微分方程求解界面

在 MATLAB 提示符下键入 `pdetool`, 将启动偏微分方程求解界面, 如图 7-29 所示。

偏微分方程求解界面分为如下几个部分:

(1) **菜单系统**。偏微分方程工具箱有较全面的菜单系统, 其中大部分实用功能均可以由工具栏实现, 工具栏不能实现的部分多为一些工具箱设置与文件处理的功能。后面将根据实际需要介绍菜单系统的若干功能。

(2) **工具栏**。工具栏内各个按钮的详细内容如图 7-30 所示, 工具栏能实现从求解区域设定、微分方程参数描述、求解到结果表示在内的一整套实际功能。工具栏右侧的列表框还给出了 MATLAB 能直接求解的一些常用微分方程类型。

(3) **集合编辑 (Set formula)**。用户可以在求解区域用不同的几何形状画出若干集合, 而集合编辑区域允许用户用加减法等表示并、交和差集运算, 更准确地描述求解区域。

(4) **求解区域**。为该程序界面下部的区域, 用户可以在这个部分内绘制出问题的求解区域, 微分方程的解也可以在这个区域内用二维的形式表示出来。MATLAB 还支持三维表示, 但需要打开新的图形窗口。

2. 偏微分方程求解区域绘制

本节将通过例子演示在偏微分方程求解界面下描述求解区域的方法。首先用工具栏中提供的椭圆绘制和矩形绘制功能绘制出如图 7-31 (a) 所示的一些区域, 这样就可以在集合编辑栏目中将原来的内容修改为 $(R1+E1+E2)-E3$, 表示从矩形 $R1$, 椭圆 $E1$ 、 $E2$ 的并

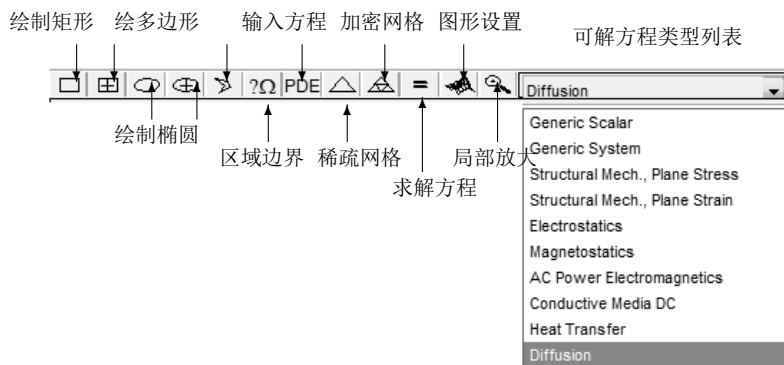


图 7-30 偏微分方程求解工具栏

集中剔除掉 E3。单击工具栏中 Ω 按钮就可以得到求解区域。选择 Boundary \rightarrow Remove All Subdomain Borders 菜单项,则将消除若干相邻区域中间的分隔线,得出如图 7-31 (b) 所示的区域图。

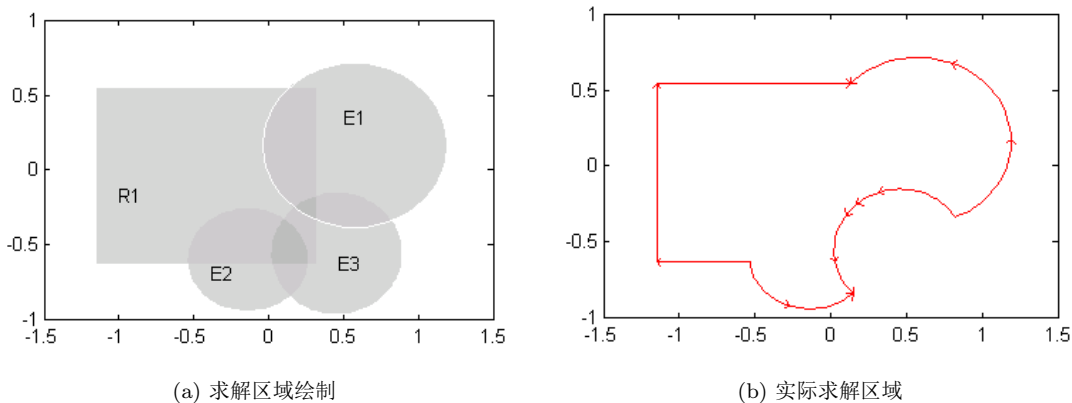


图 7-31 偏微分方程求解区域设置

有了求解区域,就可以单击 Δ 按钮将求解区域用三角形划分成若干网格,如图 7-32 (a) 所示。如果感觉到网格不够密,则可以单击右侧的按钮加密网格,可以得出如图 7-32 (b) 所示的更密的网格图。值得指出的是,一般情况下,网格越密,计算的结果越精确,但代价是计算时间则越长。

3. 偏微分方程边界条件描述

求解边界在偏微分方程界面下用 Ω 表示,数学上记作 $\partial\Omega$ 。一般地,在偏微分方程工具箱支持的边界条件包括 Dirichlet 条件和 Neumann 条件。下面分别介绍这两种边界条件。

(1) Dirichlet 条件。一般描述为

$$h\left(\mathbf{x}, t, u, \frac{\partial u}{\partial \mathbf{x}}\right) u|_{\partial\Omega} = r\left(\mathbf{x}, t, u, \frac{\partial u}{\partial \mathbf{x}}\right) \quad (7-7-17)$$

其中, $\partial\Omega$ 表示求解区域的边界。假设在边界上满足该方程,则只需给出 r 和 h 函数即可,这两个参数可以为常数,也可以为 \mathbf{x} 的函数,甚至可以是 u 、 $\partial u / \partial \mathbf{x}$ 的函数,为方便起见,一般

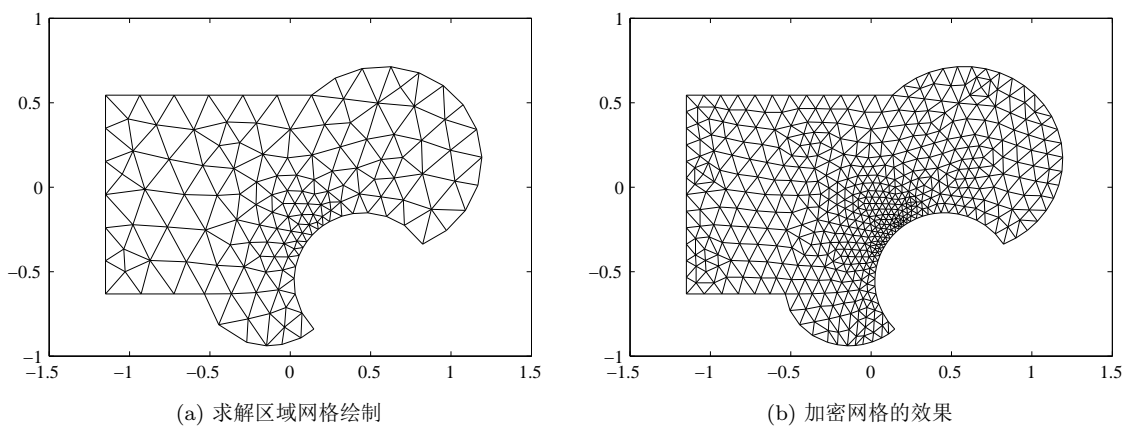


图 7-32 求解区域的网格生成

可以令 $h = 1$ 。后面将介绍 Dirichlet 边界条件的描述方式。

(2) **Neumann 条件**。其扩展形式为

$$\left[\frac{\partial}{\partial \mathbf{n}} (c \nabla u) + qu \right] \Big|_{\partial \Omega} = g \quad (7-7-18)$$

其中, $\partial u / \partial \mathbf{n}$ 为 \mathbf{x} 向量法向的偏导数。

选择 **Boundary** \rightarrow **Specify Boundary Conditions** 菜单, 将打开一个如图 7-33 所示的对话框, 用户可以在这个对话框中描述边界条件。如果想使得边界上各点的函数值为 0, 则可以将该对话框的 r 栏值设为 0 即可。

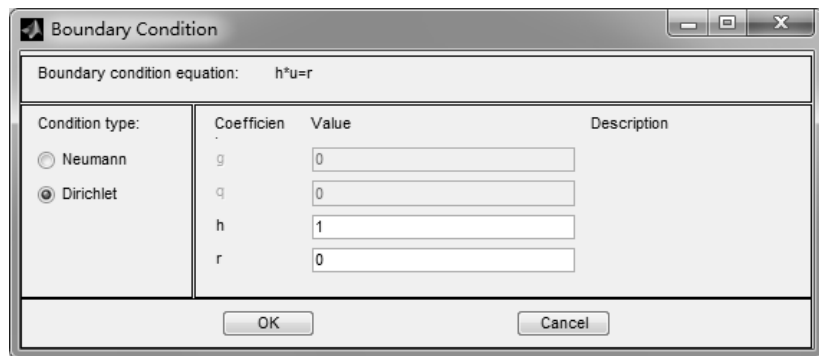


图 7-33 边界条件设置对话框

4. 偏微分方程求解举例

用前面的方法设置了求解区域和边界条件, 并选择了合适的偏微分方程后, 就可以单击工具栏的等号按钮 (=) 立即得出微分方程的解。下面将通过例子演示实际偏微分方程的求解全过程。

例 7-41 试求解双曲型偏微分方程 $\frac{d^2 u}{dt^2} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + 2u = 10$ 。

解 由给定的偏微分方程, 可以得出 $c = 1, a = 2, f = 10, d = 1$ 。这样单击偏微分方程界面工具

栏中的 PDE 图标,则将打开一个类似于图 7-34 的对话框,左侧有各种常见的偏微分方程类型。选择其中的 Parabolic 选项,就能将给定的偏微分方程的参数输入到该对话框中。

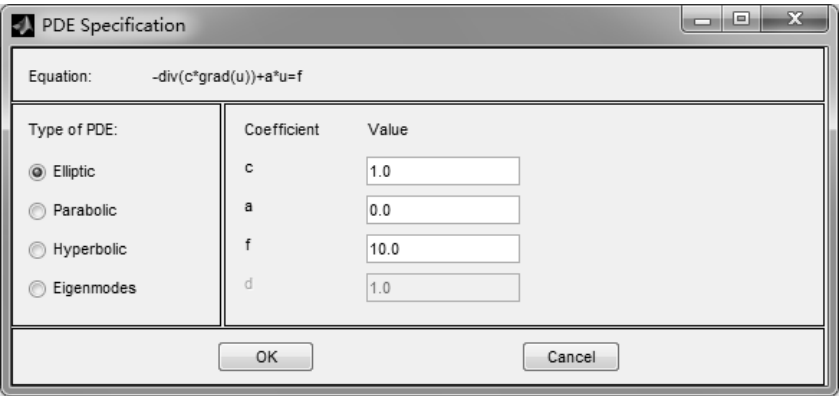


图 7-34 偏微分方程参数设置对话框

如果想求解该偏微分方程,则可以单击工具栏中的等号按钮,这样马上就能求出微分方程的解,如图 7-35 (a) 所示。其中,图形为伪色彩图形,其颜色表示 $u(x,y)$ 值。注意,这时给出的 $u(x,y)$ 的值为在 $t = 0$ 时的函数值,后面将介绍如何显示不同 t 下方程的解。

用户还可以修改微分方程的边界条件。例如,再得出图 7-33 所示的对话框,仍采用 Dirichlet 条件,令边界上所有的 u 值为 5,则可以将该对话框中 r 栏目的值填写为 5,这样再求解偏微分方程,将得出如图 7-35 (b) 所示的结果。

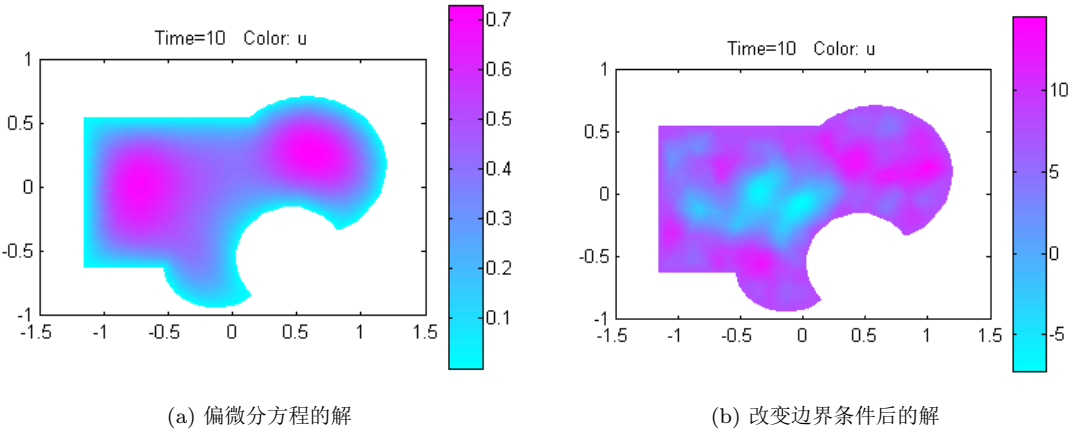


图 7-35 偏微分方程的解

微分方程的结果还可以用其他很多方式显示。单击工具栏中的三维图图标则将打开一个如图 7-36 所示的对话框,若再选择 Contour 则可以绘制等值线图,若选择 Arrows 选项,将计算并绘制引力线,选择这两个选项,将得出如图 7-37 (a) 所示的计算结果。

另外还应注意,在如图 7-36 所示的对话框中,Property 栏目的各个项目均有列表框,如第一项的默认值为 u ,表明所有的分析都是针对 $u(\cdot)$ 函数的,在绘图时显示的是 $u(x,y)$ 。如果想显示其他的内容,则可以单击右侧的 ▼,这样就可以打开列表框,从中选择其他的分析

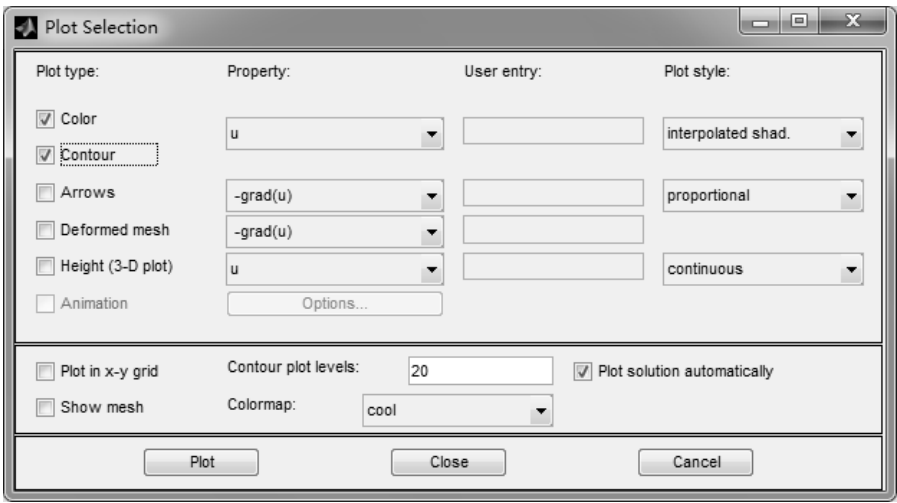


图 7-36 结果显示方式设置对话框

内容,直至选择用户自定义栏目。
若单独选择 Height (3d-plot),则将另外打开一个图形窗口,绘制出网格型三维图形,如图 7-37 (b) 所示。

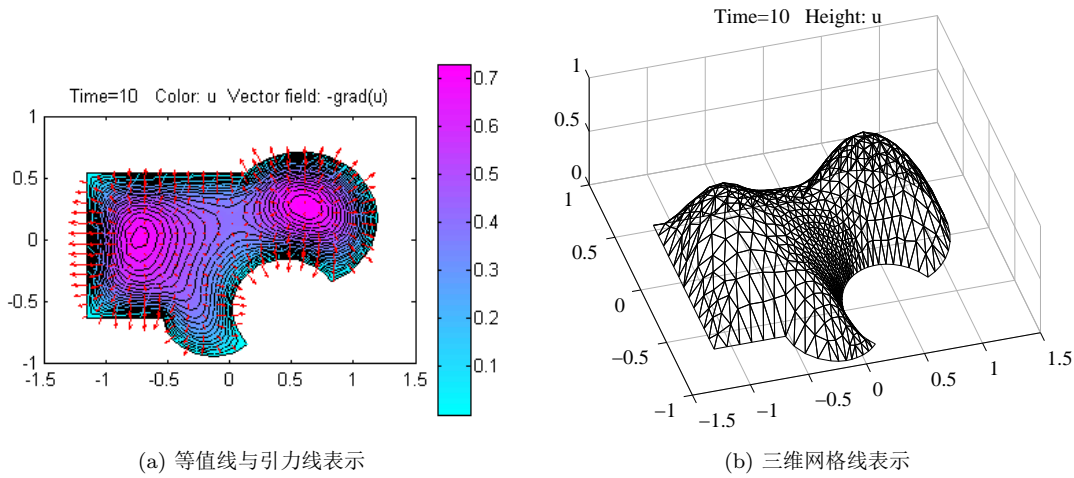


图 7-37 偏微分方程解的不同表现形式

5. 时变解的动画显示

偏微分方程程序默认的时间向量为 $t=0:10$,图 7-35 中得出的微分方程解也是在最终时刻 $t=10$ 的解。从双曲偏微分方程看,方程的解应该是时间 t 的函数,所以应该用动画形式显示出来。现在仍然以例 7-41 中给出的双曲偏微分方程为例,介绍如何将时变方程显示出来。

用户可以由 Solve → Parameters 菜单引出的对话框设置时间向量,例如在该栏目内填写 $0:0.1:4$,这样再进行微分方程求解则为这段时间的解。定义了该时间向量,由图 7-36 给出的对话框可以选择其中的动画 (Animation) 选项,单击 Options 按钮还可以

设置动画的播放速度,如用默认的 6fps(每秒 6 帧),这样就可以直接获得该微分方程解的动画了。用户可以用 Plot → Export Movie 菜单将动画输出到 MATLAB 工作空间,例如存成变量 M,则可以用 movie(M) 在 MATLAB 图形窗口中播放得出的动画,也可以用 movie2avi(M,'myavi.avi') 命令将动画存成 myavi.avi 文件,以备过后播放。

6. 函数参数的偏微分方程求解

前面介绍的偏微分方程 c, a, d, f 均为常数,而在实际遇到的偏微分方程中,经常需要这些量为函数的情况。偏微分方程工具箱目前能处理的问题是含有非线性系数的椭圆偏微分方程问题,在系数字符串中允许使用 x, y 直接表示微分方程中的 x_1, x_2 或 x, y ,用变量 ux 和 uy 表示 $\partial u / \partial x$ 和 $\partial u / \partial y$,这样就可以描述任意的非线性系数。下面将通过例子演示这种方程的求解方法。

例 7-42 假设偏微分方程为 $-\operatorname{div}\left(\frac{1}{1+|\nabla u|^2}\nabla u\right)+(x^2+y^2)u=e^{-x^2-y^2}$, 其中边界为 0, 试求解该偏微分方程。

解 观察该方程可以发现,它满足椭圆型偏微分方程,其中

$$c = \frac{1}{\sqrt{1 + \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2}}, \quad a = x^2 + y^2, \quad f = e^{-x^2 - y^2}$$

且在求解边界上 u 的值为 0。仍使用偏微分方程工具箱,打开如图 7-34 所示的对话框,选择椭圆型偏微分方程(Elliptic)选项,在 c 参数栏目填写 $1./\operatorname{sqrt}(1+ux.^2+uy.^2)$,在 a 和 f 栏目分别填写 $x.^2+y.^2$ 和 $\exp(-x.^2-y.^2)$;再打开 Solve → Parameters 对话框,从中选定 Use nonlinear solver 属性(注意,该属性只适用于椭圆型偏微分方程求解)。再单击工具栏内的等号,则可以求解该方程,得出如图 7-38(a)、(b)所示的解。

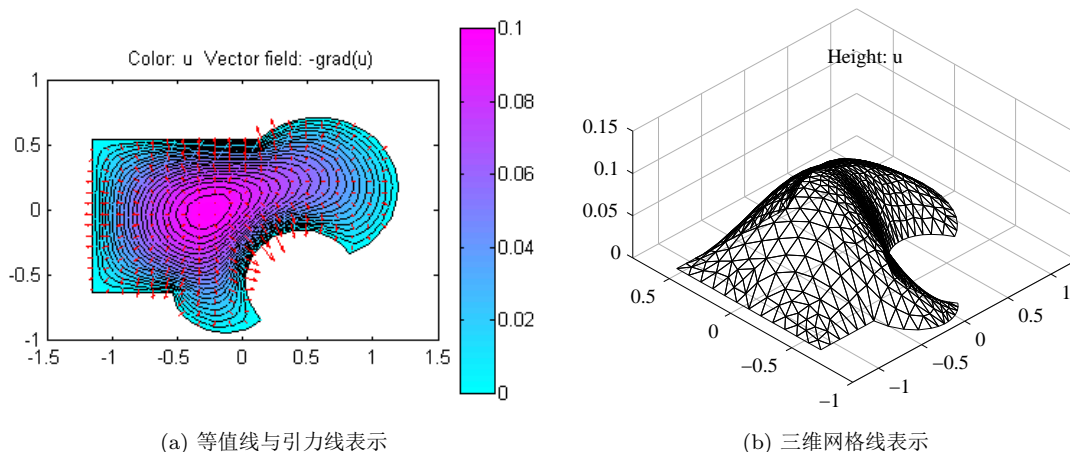


图 7-38 偏微分方程解的不同表现形式

7.8 基于 Simulink 的微分方程框图求解

前面介绍了微分方程的求解函数,如果某系统可以由微分方程描述,则可以通过前面介绍的方法直接求出微分方程的数值解。如果某微分方程是复杂系统中的一个部分,其输

入信号来自于前一个模块,则不能用前面介绍的方法求解微分方程,除非整个系统模型可以由单一的微分方程组描述出来。所以基于框图的仿真策略将是解决这类问题的最好方法,Simulink 是 MATLAB 下基于框图仿真方法的理想工具。本节将介绍各种微分方程的 Simulink 建模与仿真方法。

7.8.1 Simulink 简介

Simulink 环境是 1990 年前后由 The MathWorks 公司推出的产品,原名为 SimuLAB,1992 年改为 Simulink。其名字有两重含义,仿真(simu)与模型连接(link),表示该环境可以用框图的方式对系统进行仿真。可以利用这一有效的工具,用图形的方式描述各种各样的微分方程,从而求解相应的微分方程。

当然,Simulink 的功能远不止微分方程的求解,它还提供了各种可用于控制系统仿真的模块,支持一般的控制系统仿真。此外,它还提供了各种工程应用中可能使用的模块,如电机系统、机械系统、通信系统等模块集,直接进行建模与仿真研究。Simulink 的功能十分强大,可以借用其本身或模块集对任意复杂的系统进行仿真。相关内容可以参阅其手册^[9]和书籍^[10],限于本书篇幅,只能介绍和微分方程求解有关的内容。

本节先简单介绍微分方程建模可能用到的模块,然后通过例子演示微分方程的 Simulink 建模方法与求解方法。

7.8.2 Simulink 相关模块

在 MATLAB 命令窗口下给出下面的命令 `open_system('simulink')` 将打开如图 7-39 所示的模块组窗口。可见,该组窗口中提供了各类下一级的模块组,如输入信号源模块组 Sources、连续模块组 Continuous 等,每组的模块都是很丰富的,理论上可以建立任意复杂问题的仿真模型。

Simulink 下支持的模块不胜枚举,不可能在本节的篇幅内全部介绍,所以针对微分方程模块搭建问题,作者只选择了一些常用模块作为子模块库,用 `odegroup` 命令可以打开如图 7-40 所示的自定义模块集。

下面将介绍其中常用的模块:

(1) **输入输出端口(In1, out1)**。一般采用输出模块显示微分方程求解的结果,该模块将在 MATLAB 工作空间中产生变量 `yout`。此外,仿真信号还可以用示波器 Scope 直接显示。

(2) **时钟模块**。Clock 产生时间 t ,从而可以搭建时变微分方程模型。

(3) **常用输入模块**。可以用 Sine 模块产生正弦信号,用 Step 可以产生阶跃信号,而用 Constant 模块可以产生恒值信号。

(4) **积分器模块(Integrator)**。可以用其描述一阶导数,令常微分方程组的每个一阶导数项作为每个积分器模块的输入。例如,第 i 个积分器模块的输入端定义为 $x'_i(t)$,则其输出端自然就是 $x_i(t)$ 。若给出了一阶微分方程组,则积分器输入端的搭建就是整个微分方程组 Simulink 模型搭建的关键。对于线性高阶微分方程,还可以采用其中的传递函数模块 Transfer Fcn。

(5) **延迟模块(Transport Delay)**。可以得出其输入信号在 $t - \tau$ 时刻的值。该模块可以

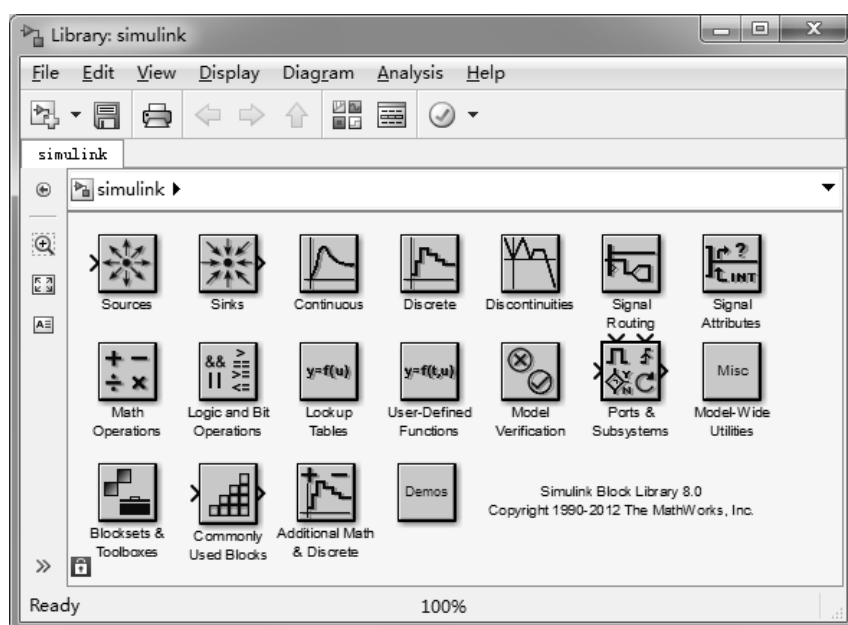


图 7-39 Simulink 环境的主窗口

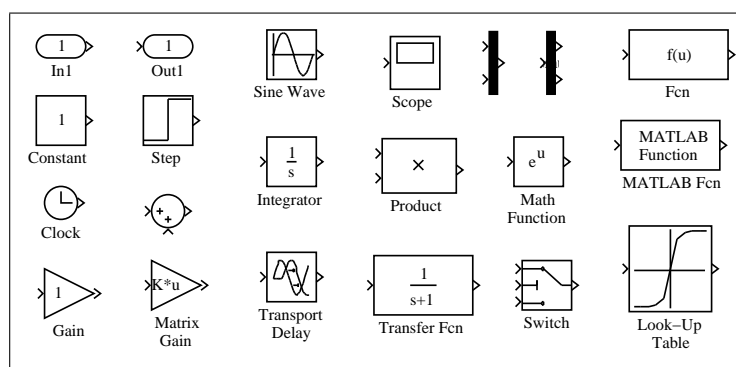


图 7-40 常用模块的自定义模块组

用于延迟微分方程的建模与求解。

(6) **增益模块 (Gain、Sliding Gain)**。这些增益模块都是建模中很有意义的增益模块,而它们的作用也各不相同。**Gain** 模块主要用于信号的放大,如果该模块的输入信号为 u ,则其输出为 Ku 。通过对话框设置还可以将其设置为矩阵增益模块。**Sliding Gain** 模块较有特色,它实际上是一个滚动杆,用户可以通过鼠标拖动的方式实时改变该模块的增益。

(7) **数学运算模块**。可以对其输入信号实现加减乘除等代数运算,也可以实现各种逻辑运算和比较运算。

(8) **数学函数模块**。可以对输入信号做模块指定的非线性运算,如三角函数运算、指数对数运算等。

(9) **信号向量化模块**。用混路模块 **Mux** 可以将若干路信号混成向量型的信号,用 **DeMux** 模块可以将向量型信号解出单路的信号。

7.8.3 微分方程的 Simulink 建模与求解

建立起微分方程的 Simulink 模型, 可以用 `sim()` 函数对其模型直接求解, 得出微分方程的数值解。`sim()` 函数的调用格式和 `ode45()` 等函数特别接近, 这里不详细介绍该函数的调用格式, 读者可以参考例子中的调用格式, 领略其语法结构。

本节将通过例子介绍微分方程的 Simulink 建模方法与微分方程求解问题, 首先介绍 Lorenz 方程的建模方法, 然后介绍一般的延迟微分方程建模, 最后用建模的方法解出前面不能求解的延迟微分方程。

例 7-43 考虑例 7-10 中给出的 Lorenz 方程的求解问题, 这里重写如下

$$\begin{cases} x_1'(t) = -\beta x_1(t) + x_2(t)x_3(t) \\ x_2'(t) = -\rho x_2(t) + \rho x_3(t) \\ x_3'(t) = -x_1(t)x_2(t) + \sigma x_2(t) - x_3(t) \end{cases}$$

其中, 设 $\beta = 8/3, \rho = 10, \sigma = 28$, 且各个状态变量的初值为 $x_1(0) = x_2(0) = 0, x_3(0) = 10^{-10}$, 试用 Simulink 搭建该模型, 并得出仿真结果。

解 前面已经介绍过, 用 MATLAB 语言可以编写一个函数, 描述原来的微分方程组模型, 然后用 `ode45()` 类函数就可以直接求解该方程。

用 Simulink 也可以描述出微分方程组的模型。具体的方法是: 考虑原方程中有状态变量的一阶导数项, 需要使用三个积分器, 用其输入端分别描述 $x_1'(t), x_2'(t), x_3'(t)$, 则其输出端自然就成了 $x_1(t), x_2(t), x_3(t)$, 这样就建立起了 Simulink 框图的核心模块框架, 如图 7-41 (a) 所示。双击积分器模块, 可以将状态变量初值填写进各个积分器模块。

在构造微分方程求解框架时, 定义了各个状态变量及其导数的信号, 利用混路器 Mux 模块, 可以定义出向量型的信号 $\mathbf{x}(t) = [x_1(t), x_2(t), x_3(t)]^T$, 该信号进入 Fcn 模块可以直接表示为该模块输入信号的 $\mathbf{u}(t) = [u_1(t), u_2(t), u_3(t)]^T$ 。这样, 考虑 Lorenz 方程的第一个式子, 可以在最下面的 Fcn 模块中填写 `-beta*u[1]+u[2]*u[3]`, 从而将该模块的输出直接连接到第一个积分器的输入端 dx_1/dt , 搭建起第一个微分方程式。用类似的方法可以建立起其他两个微分方程式, 用这样的方法最终可以构造出如图 7-41 (b) 所示的完整 Simulink 模型。为获得仿真结果, 可以将 $\mathbf{x}(t)$ 状态变量连接到输出端口。

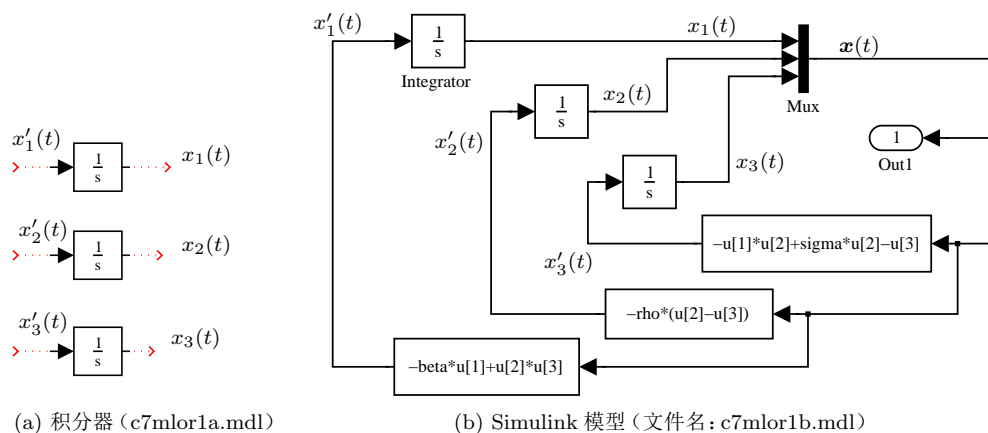


图 7-41 Lorenz 方程的 Simulink 建模

仿真模型建立起来之后,可以用下面的语句对该微分方程进行求解,并得出和图 7-2(a)、(b) 完全一致的仿真结果,这里不再重复给出。

```
>> beta=8/3; rho=10; sigma=28; [t,x]=sim('c7mlor1b',[0,100]); plot(t,x)
figure; plot3(x(:,1),x(:,2),x(:,3))
```

注意,这里的 β 、 ρ 和 σ 参数可以写入 MATLAB 工作空间,而无需作为附加参数在语句调用中给出,也可以将积分器的初值在积分器模块中设置成变量形式,无需改变 Simulink 模型本身就可以改变状态变量的初值。

应该指出,对于小规模问题来说,用 Simulink 建模的求解方式比用 `ode45()` 等函数的调用方式更复杂。但在解决大规模问题、模块化问题亦即复杂混连系统的问题时,用 Simulink 建模方式应该比简单的函数调用更合适。此外,对更复杂的时间延迟微分方程求解问题来说,采用 Simulink 建模的方法,可以解决用普通微分方程求解函数解决不了的问题。

例 7-44 考虑例 7-28 中介绍的延迟微分方程式,重写如下,试用 Simulink 搭建该微分方程模型,并得出其数值解。

$$\begin{cases} x'(t) = 1 - 3x(t) - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5) \\ y''(t) + 3y'(t) + 2y(t) = 4x(t) \end{cases}$$

解 该模型当然可以用 MATLAB 语言编写一个函数来表示,然后用延迟微分方程的求解函数 `dde23()` 直接求解,但这样的求解过程似乎不是很直观。

现在考虑第一个方程式,将 $-3x(t)$ 项移动到等号左侧,则可以将其变换成

$$x'(t) + 3x(t) = 1 - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5)$$

所以可以将该方程理解成 $x(t)$ 为传递函数模型 $1/(s+3)$ 的输出信号,而该函数的输入信号为 $1 - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5)$ 。第 2 个方程式可以理解为: $y(t)$ 是传递函数模块 $4/(s^2+3s+2)$ 的输出信号,而该模块的输入信号为 $x(t)$,在 $x(t)$ 信号和 $y(t)$ 信号上连接延迟模块 Transport Delay 可以得出这些信号的延迟。通过上面的分析,可以搭建出如图 7-42 所示的 Simulink 仿真模型。

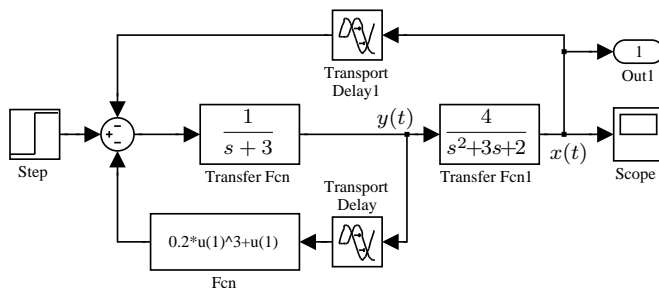


图 7-42 延迟微分方程的 Simulink 模型 (文件名: c7mdde2.mdl)

建立了仿真模型后,就可以用下面的语句求解该微分方程,并得出输出信号 $y(t)$ 的曲线。该结果和例 7-28 中给出的完全一致,这里不再重复给出。该结果还可以同时在示波器上显示出来。

```
>> [t,x]=sim('c7mdde2',[0,10]); plot(t,x)
```

当然,若不习惯使用传递函数模块,还可以假设 $x_1 = x, x_2 = y, x_3 = y'$,这样可以将原微分方程

模型变换成一阶状态方程模型

$$\begin{cases} x_1'(t) = 1 - x_1(t) - x_2(t-1) + 0.2x_1^3(t-0.5) - x_1(t-0.5) \\ x_2'(t) = x_3(t) \\ x_3'(t) = -4x_1(t) - 3x_3(t) - 2x_2(t) \end{cases}$$

给这3个状态变量选择3个积分器,则可以搭建出 Simulink 框图,也可以得出同样的结果。这里不给出具体的 Simulink 模型,读者可以按系统的要求自己搭建该模型。

例 7-45 现在考虑例 7-33 中定义的延迟微分方程,其中

$$\mathbf{A}_1 = \begin{bmatrix} -13 & 3 & -3 \\ 106 & -116 & 62 \\ 207 & -207 & 113 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0.03 & 0 \\ 0 & 0 & 0.04 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

试用 Simulink 搭建系统模型,并得出系统的仿真曲线。

解 该方程用 MATLAB 自身提供的 `ddensd()` 函数可以求解,所以这里考虑采用基于 Simulink 的框图形式求解该方程。在建模之前,可以用下面的语句输入已知的矩阵

```
>> A1=[-13,3,-3; 106,-116,62; 207,-207,113];
```

```
A2=diag([0.02,0.03,0.04]); B=[0; 1; 2];
```

再考虑原始的微分方程模型,已经存在一个状态向量 $\mathbf{x}(t)$,故可以安排一个积分器,使得其输出为 $\mathbf{x}(t)$,这样其输入端自然是 $\mathbf{x}'(t)$,可以分别给这两个信号连接延迟环节,并按实际情况设置延迟时间常数,则可以构造出 $\mathbf{x}(t-\tau_1)$ 和 $\mathbf{x}'(t-\tau_2)$ 信号,这样经过简单的处理就可以搭建出如图 7-43 所示的 Simulink 模型。

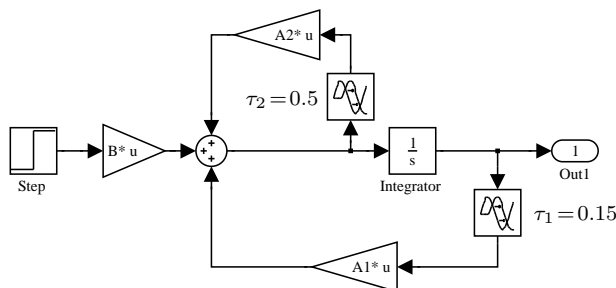


图 7-43 带有导数延迟的微分方程 Simulink 模型 (文件名: c7mdde3.mdl)

用下面的语句就可以求解该方程,并将各个状态变量绘制出来,如图 7-22 所示。

```
>> [t,x]=sim('c7mdde3',[0,8]); plot(t,x)
```

例 7-46 如果各个状态变量初始条件为零,试求解下面的变时间延迟微分方程。

$$\begin{cases} x_1'(t) = -2x_2(t) - 3x_1(t-0.2|\sin t|) \\ x_2'(t) = -0.05x_1(t)x_3(t) - 2x_2(t-0.8) \\ x_3'(t) = 0.3x_1(t)x_2(t)x_3(t) + \cos(x_1(t)x_2(t)) + 2\sin 0.1t^2 \end{cases}$$

解 和其他微分方程框图建模一样,需要用向量型积分器来定义状态变量向量 $\mathbf{x}(t)$,由 $\mathbf{x}(t)$ 、 $x_1(t-0.2|\sin t|)$ 、 $x_2(t-0.8)$ 、 t 六路信号构造 Mux 模块的输出向量,并编写出下面的 MATLAB 静态函数,且将其文件名填写到 Interpret MATLAB Function 模块

```
function y=c7fdde6(x)
y=[-2*x(2)-3*x(4); -0.05*x(1)*x(3)-2*x(5)+2;
    0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*x(6)^2)];
```

这样可以搭建起如图 7-44 所示的系统仿真框图。注意,在框图中,变延迟时间模型可以调用 Variable Time Delay 模块,让其第 2 路输入信号表示变时间延迟 $0.2|\sin t|$ 。对该系统进行仿真,将得出与图 7-19 中完全一致的结果。

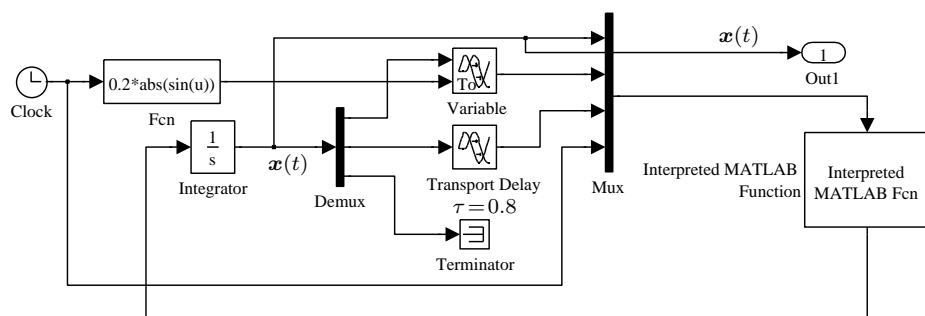


图 7-44 变时间延迟微分方程的 Simulink 模型 (模型名: c7mdde6.mdl)

如果微分方程的状态变量含有非零初值,且在 $t < 0$ 时状态变量的历史值为 0,则将积分器的初值直接设置成非零向量即可。例如,在此微分方程中将状态变量在 $t = 0$ 时刻的值设置成 $x_0 = [\sin 1, \cos 1, 1]^T$,则将得出与图 7-20(b) 完全一致的结果。

7.4.5 节介绍了线性微分方程在随机信号激励下的时域求解方法,该方法不能扩展到非线性微分方程的求解,所以可以考虑采用 Simulink 建模与仿真方法直接求解。随机输入模块可以采用 Sources 组中的 Band-Limited White Noise 模块表示,该模块需填写白噪声的方差与计算步长等信息。

例 7-47 重新考虑例 7-27 中给出的线性微分方程的求解问题。

解 根据原始的问题可以建立起如图 7-45 所示的仿真模型。选择采样周期 $T = 0.02\text{s}$,则可以启动仿真过程,然后给出下面的语句绘制输出信号的概率密度函数,得出的结果和图 7-17(b) 给出的基本一致,表明这样的仿真方法是可行的。

```
>> w=0.2; x=-2.5:w:2.5; y1=hist(yout,x); bar(x,y1/length(yout)/w);
x1=-2.5:0.05:2.5; v=0.6655; y2=1/sqrt(2*pi)/v*exp(-x1.^2/2/v^2);
line(x1,y2) % 叠印理论概率密度函数曲线
```

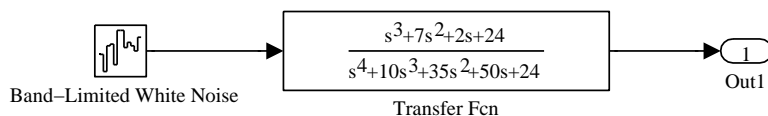


图 7-45 变时间延迟微分方程的 Simulink 模型 (模型名: c7mrand)

例 7-48 随机输入系统的建模与仿真。假设非线性系统的模型如图 7-46 所示,其中线性传递函

数和饱和非线性环节如下描述

$$G(s) = \frac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24}, \text{ 非线性环节 } \mathcal{N}(e) = \begin{cases} 2\text{sign}(e), & |e| > 1 \\ 2e, & |e| \leq 1 \end{cases}$$

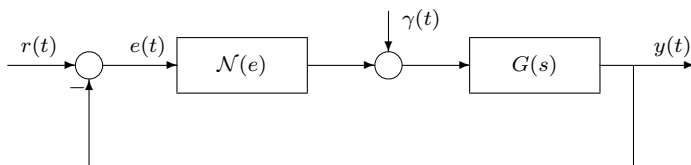


图 7-46 随机输入非线性系统框图

随机扰动信号 $\gamma(t)$ 为均值为 0, 方差为 3 的 Gauss 白噪声信号, 确定性输入信号 $r(t) = 0$ 。随机输入信号应该使用 Band-limited White Noise 模块, 而不能使用其他随机信号发生器模块。这样搭建起来的随机系统仿真模型如图 7-47 所示。注意, 应该采用定步长仿真方法对该系统进行仿真, 并将仿真步长设置成和 Band-limited White Noise 模块完全一致的值, 比如 0.01。此外, 随机系统的仿真一定要有足够多的仿真点才有意义, 所以这里选择 30000 个仿真点。

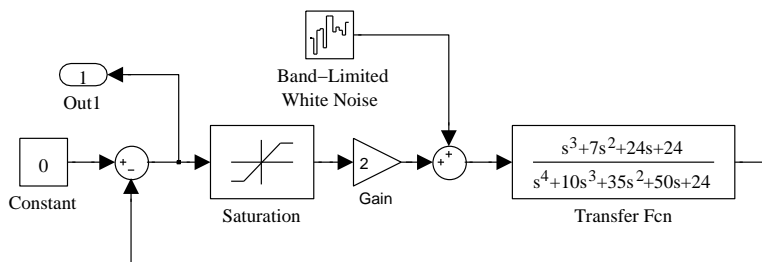


图 7-47 随机输入非线性系统仿真框图 (文件名: c7mnlrsys.mdl)

对该系统进行仿真, 则仿真结果将由 tout、yout 向量返回到 MATLAB 的工作空间, 给出下面语句将分别绘制出输出信号最后 500 个点的时域响应曲线和由仿真数据近似的 $e(t)$ 信号的概率密度直方图, 如图 7-48 (a)、(b) 所示。

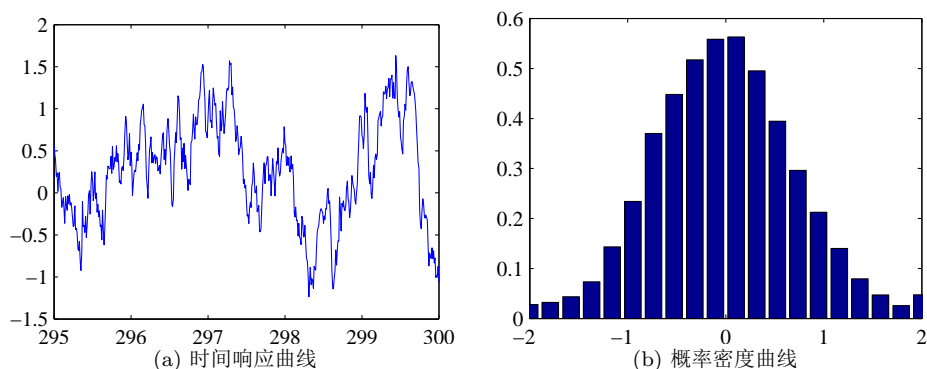
```
>> plot(tout(end-500:end), yout(end-500:end)); c=linspace(-2,2,20);
y1=hist(yout,c); figure; bar(c,y1/(length(tout)*(c(2)-c(1))))
```

7.9 习 题

1. 试求出下面线性微分方程的通解。

$$y^{(5)} + 13y^{(4)} + 64y'''(t) + 152y''(t) + 176y'(t) + 80y(t) = e^{-2t} [\sin(2t + \pi/3) + \cos 3t]$$

假设上述微分方程满足已知条件 $y(0) = 1, y(1) = 3, y(\pi) = 2, y'(0) = 1, y'(1) = 2$, 试求出满足该条件的微分方程的解析解。

图 7-48 系统 $e(t)$ 信号的仿真结果

2. 试求下面微分方程的通解

$$(1) \begin{cases} x'' - 2y'' + y' + x - 3y = 0 \\ 4y'' - 2x'' - x' - 2x + 5y = 0 \end{cases}, \quad (2) \begin{cases} 2x'' + 2x' - x + 3y'' + y' + y = 0 \\ x'' + 4x' - x + 3y'' + 2y' - y = 0 \end{cases}.$$

3. 试求解下面微分方程的通解以及满足 $x(0) = 1, x(\pi) = 2, y(0) = 0$ 条件下的解析解。

$$\begin{cases} x''(t) + 5x'(t) + 4x(t) + 3y(t) = e^{-6t} \sin 4t \\ 2y'(t) + y(t) + 4x'(t) + 6x(t) = e^{-6t} \cos 4t \end{cases}$$

4. 试求出下面的时变线性微分方程的解析解。

(1) Legendre 微分方程 $(1 - t^2)x'' - 2tx' + n(n+1)x = 0$,

(2) Bessel 微分方程 $t^2x'' + tx' + (t^2 - n^2)x = 0$ 。

5. 试求出微分方程 $y''(x) - (2 - 1/x)y'(x) + (1 - 1/x)y(x) = x^2e^{-5x}$ 的解析解通解, 并求出满足边界条件 $y(1) = \pi, y(\pi) = 1$ 的解析解。

6. 试求出下面微分方程组的解析解, 并利用 Laplace 变换求出其解。

$$\begin{cases} x''(t) + y''(t) + x(t) + y(t) = 0, \\ 2x''(t) - y''(t) - x(t) + y(t) = \sin t, \end{cases} \quad x(0) = 2, y(0) = 1, x'(0) = y'(0) = -1$$

7. 试求出下面微分方程的通解。

(1) $x''(t) + 2tx'(t) + t^2x(t) = t + 1$, (2) $y'(x) + 2xy(x) = xe^{-x^2}$, (3) $y''' + 3y'' + 3y' + y = e^{-t} \sin t$ 。

8. 试求解下面的非线性微分方程解析解。

(1) $y' = y^4 \cos x + y \tan x$, (2) $xy^2y' = x^2 + y^2, xy' + 2y + x^5y^3e^x = 0$ 。

9. 试求出下面方程的解析解, 并绘制出 (x, y) 的轨迹曲线。

$$\begin{cases} (2x'' - x' + 9x) - (y'' + y' + 3y) = 0, \\ (2x'' + x' + 7x) - (y'' - y' + 5y) = 0, \end{cases} \quad x(0) = x'(0) = 1, y(0) = y'(0) = 0.$$

10. 试求解下面的时变线性微分方程。

(1) $(x^2 - 2x + 3)y''' - (x^2 + 1)y'' + 2xy' - 2y = 0$,

(2) $x^2 \ln xy'' - xy' + y = 0$, (3) $(e^t + 1)y'' - 2y' - e^t y = 0$ 。

11. 极限环是二阶非线性常微分方程中一种常见的现象,对某些非线性微分方程来说,不论初始状态为何值,微分方程的相轨迹都将稳定在一条封闭的曲线上,该曲线称为微分方程的极限环。试绘制出下面微分方程的极限环,并对不同初值验证微分方程的相平面曲线确实收敛于极限环。

$$\begin{cases} x' = y + x(1 - x^2 - y^2) \\ y' = -x + y(1 - x^2 - y^2) \end{cases}$$

12. 考虑下面给出的非线性微分方程。文献 [11] 指出,该方程具有多个极限环, $r = 1/(n\pi)$, $n = 1, 2, 3, \dots$ 。用数值方法求解此方程并观察多极限环的情况。

$$\begin{cases} x' = -y + xf(\sqrt{x^2 + y^2}) \\ y' = x + yf(\sqrt{x^2 + y^2}) \end{cases}, \quad \text{式中}, f(r) = r^2 \sin(1/r)$$

13. 考虑下面给出的著名的 Rössler 化学反应方程组,选定 $a = b = 0.2$, $c = 5.7$, 且 $x_1(0) = x_2(0) = x_3(0)$, 绘制仿真结果的三维相轨迹,并得出其在 x - y 平面上的投影。在实际求解中建议将 a 、 b 、 c 作为附加参数,同样的方程若设 $a = 0.2$, $b = 0.5$, $c = 10$ 时,绘制出状态变量的二维图和三维图。

$$\begin{cases} x' = -y - z \\ y' = x + ay \\ z' = b + (x - c)z \end{cases}$$

14. Chua 电路方程是混沌理论中经常提到的微分方程 [12]

$$\begin{cases} x' = \alpha[y - x - f(x)] \\ y' = x - y + z \\ z' = -\beta y - \gamma z \end{cases}$$

其中, $f(x)$ 为 Chua 电路的二极管分段线性特性, $f(x) = bx + \frac{1}{2}(a - b)(|x + 1| - |x - 1|)$, 且 $a < b < 0$ 。试编写出 MATLAB 函数描述该微分方程,并绘制出 $\alpha = 15$, $\beta = 20$, $\gamma = 0.5$, $a = -120/7$, $b = -75/7$, 且初始条件为 $x(0) = -2.121304$, $y(0) = -0.066170$, $z(0) = 2.881090$ 时的相空间曲线。

15. Lotka-Volterra 扑食模型方程如下,试求解该微分方程,并绘制相应的曲线。

$$\begin{cases} x'(t) = 4x(t) - 2x(t)y(t), \\ y'(t) = x(t)y(t) - 3y(t), \end{cases} \quad x(0) = 2, y(0) = 3$$

16. 请给出求解下面微分方程的 MATLAB 命令,并绘制出 $y(t)$ 曲线。

$$y''' + ty y'' + t^2 y' y^2 = e^{-ty}, \quad y(0) = 2, \quad y'(0) = y''(0) = 0$$

试问该方程存在解析解吗? 选择四阶定步长 Runge-Kutta 算法求解该方程时,步长选择多少可以得出较好的精度,试与 MATLAB 语言给出的现成函数在速度、精度上进行比较。

17. 试选择状态变量,将下面的非线性微分方程组转换成一阶显式微分方程组,并用 MATLAB 对其求解,绘制出解的相平面或相空间曲线。

$$(1) \begin{cases} x'' = -x - y - (3x')^2 + y'^3 + 6y'' + 2t \\ y''' = -y'' - x' - e^{-x} - t \\ x(1) = 2, x'(1) = -4 \\ y(1) = -2, y'(1) = 7, y''(1) = 6 \end{cases}, \quad (2) \begin{cases} x'' - 2xz' = 3x^2yt^2 \\ y'' - e^y y' = 4xt^2z \\ z'' - 2tz' = 2te^{xy} \\ z'(1) = x'(1) = y'(1) = 2 \\ z'(1) = x(1) = y(1) = 3 \end{cases}.$$

18. 试用解析解和数值解的方法求解下面的微分方程组。

$$\begin{cases} x''(t) = -2x(t) - 3x'(t) + e^{-5t}, & x(0) = 1, x'(0) = 2 \\ y''(t) = 2x(t) - 3y(t) - 4x'(t) - 4y'(t) - \sin t, & y(0) = 3, y'(0) = 4 \end{cases}$$

19. 给定微分方程组如下, 且 $u(0) = 1, u'(0) = 2, v'(0) = 2, v(0) = 1$, 试选择一组状态变量, 将其变换成 MATLAB 语言能直接求解的微分方程组形式, 并绘制出 $u(t), v(t)$ 的轨迹曲线。

$$\begin{cases} u''(t) = -u(t)/r^3(t), \\ v''(t) = -v(t)/r^3(t), \end{cases} \quad \text{其中 } r(t) = \sqrt{u^2(t) + v^2(t)}$$

20. 已知微分方程如下^[13], 其中, $u_1(0) = 45, u_2(0) = 30, u_3(0) = u_4(0) = 0, g = 9.81$, 试求解此微分方程, 并绘制出各个状态变量的时间曲线。

$$\begin{cases} u'_1 = u_3 \\ u'_2 = u_4 \\ 2u'_3 + \cos(u_1 - u_2)u'_4 = -g \sin u_1 - \sin(u_1 - u_2)u_4^2 \\ \cos(u_1 - u_2)u'_3 + u'_4 = -g \sin u_2 + \sin(u_1 - u_2)u_3^2 \end{cases}$$

21. 试求出下面隐式微分方程的数值解, 已知 $x_1(0) = 1, x'_1(0) = 1, x_2(0) = 2, x'_2(0) = 2$, 并绘制出轨迹曲线。

$$\begin{cases} x'_1 x'_2 \sin(x_1 x_2) + 5x''_1 x'_2 \cos(x_1^2) + t^2 x_1 x_2^2 = e^{-x_2^2} \\ x''_1 x_2 + x''_2 x'_1 \sin(x_1^2) + \cos(x'_2 x_2) = \sin t \end{cases}$$

22. 下面的方程在传统微分方程教程中经常被认为是刚性微分方程。试用常规微分方程解法和刚性微分方程解法分别求解这两个微分方程的数值解, 并求出解析解, 用状态变量曲线比较数值求解的精度。

$$(1) \begin{cases} y'_1 = 9y_1 + 24y_2 + 5 \cos t - \frac{1}{3} \sin t, & y_1(0) = \frac{1}{3} \\ y'_2 = -24y_1 - 51y_2 - 9 \cos t + \frac{1}{3} \sin t, & y_2(0) = \frac{2}{3} \end{cases}, \quad (2) \begin{cases} y'_1 = -0.1y_1 - 49.9y_2, & y_1(0) = 1 \\ y'_2 = -50y_2, & y_2(0) = 2 \\ y'_3 = 70y_2 - 120y_3, & y_3(0) = 1 \end{cases}.$$

23. 考虑下面的化学反应系统的反应速度方程组, 该方程往往被认为是刚性方程。试采用 `ode45()` 对之求解, 观察是否能正确求解, 如果不能求解应该如何解决问题?

$$\begin{cases} y'_1 = -0.04y_1 + 10^4 y_2 y_3, \\ y'_2 = 0.04y_1 - 10^4 y_2 y_3 - 3 \times 10^7 y_2^2, \\ y'_3 = 3 \times 10^7 y_2^2, \end{cases} \quad \text{初值为 } y_1(0) = 1, y_2(0) = y_3(0) = 0$$

24. 试求出习题 5 中给出的微分方程边值问题数值解, 绘制出 $y(t)$ 曲线, 并和该习题得出的解析解比较精度。

25. 考虑 Van der Pol 方程 $y'' + \mu(y^2 - 1)y' + y = 0$, 试求解 $\mu = 1$, 且边值 $y(0) = 1, y(5) = 3$ 时方程的数值解。如果假设 μ 为自由参数, 试求出满足边值条件, 且满足 $y'(5) = -2$ 时方程的数值解及 μ 的值, 并绘图验证之。
26. 试用数值方法求解下面的偏微分方程, 并绘制出 u 函数曲面。
- $$\begin{cases} \partial^2 u / \partial x^2 + \partial^2 u / \partial y^2 = 0 \\ u|_{x=0, y>0} = 1, \quad u|_{y=0, x \geq 0} = 0 \\ x > 0, \quad y > 0 \end{cases}$$
27. 考虑简单的线性微分方程 $y^{(4)} + 3y''' + 3y'' + 4y' + 5y = e^{-3t} + e^{-5t} \sin(4t + \pi/3)$, 且方程的初值为 $y(0) = 1, y'(0) = y''(0) = 1/2, y'''(0) = 0.2$, 试用 Simulink 搭建起系统的仿真模型, 并绘制出仿真结果曲线。考虑上面的模型, 假设给定的微分方程变化成时变线性微分方程 $y^{(4)} + 3ty''' + 3t^2y'' + 4y' + 5y = e^{-3t} + e^{-5t} \sin(4t + \pi/3)$, 而方程的初值仍为 $y(0) = 1, y'(0) = y''(0) = 1/2, y'''(0) = 0.2$, 试用 Simulink 搭建起系统的仿真模型, 并绘制出仿真曲线。
28. 考虑延迟微分方程 $y^{(4)}(t) + 4y'''(t-0.2) + 6y''(t-0.1) + 6y'(t) + 4y'(t-0.2) + y(t-0.5) = e^{-t^2}$, 且在 $t \leq 0$ 时该方程具有零初始条件, 试分别用 Simulink 建模与 `dde23()` 函数求解的方式直接求解该微分方程, 并绘制出 $y(t)$ 曲线。

参考文献

- [1] Forsythe G E, Malcolm M A, Moler C B. Computer methods for mathematical computations. Englewood Cliffs: Prentice-Hall, 1977
- [2] Bogdanov A. Optimal control of a double inverted pendulum on a cart. Technical Report CSE-04-006, Department of Computer Science & Electrical Engineering, OGI School of Science & Engineering, OHSU, 2004
- [3] 武汉大学, 山东大学. 计算方法. 北京: 人民教育出版社, 1979
- [4] Liberzon D, Morse A S. Basic problems in stability and design of switched systems. IEEE Control Systems Magazine, 1999, 19(5):59-70
- [5] 孙增圻, 袁曾任. 控制系统计算机辅助设计. 北京: 清华大学出版社, 1998
- [6] Åström K J. Introduction to stochastic control theory. London: Academic Press, 1970
- [7] Shampine L F, Kierzenka J, Reichelt M W. Solving boundary value problems for ordinary differential equation problems in MATLAB with `bvp4c`, 2000
- [8] The MathWorks. Using MATLAB (MATLAB 用户手册), 2004
- [9] The MathWorks Inc. Simulink user's manual, 2007
- [10] 薛定宇, 陈阳泉. 基于 MATLAB/Simulink 的系统仿真技术与应用. 北京: 清华大学出版社, 2002
- [11] Enns R H, McGuire G C. Nonlinear physics with MAPLE for scientists and engineers. second edition. Boston: Birkhäuser, 2000
- [12] 张化光, 王智良, 黄伟. 混沌系统的控制理论. 沈阳: 东北大学出版社, 2003
- [13] Moler C B. Numerical computing with MATLAB. MathWorks Inc, 2004

第8章 数据插值、函数逼近问题的计算机求解

在科学与工程研究中经常会通过实验测出一些数据,根据这些数据对某种规律进行研究是数据插值与函数逼近所要解决的问题。可以将已知数据看成是样本点,所谓数据插值就是在样本点的基础上求出不在样本点上的其他点处的函数值。本章 8.1 节将介绍一维、二维甚至多维数据插值问题的求解方法,并介绍一种基于插值技术的求取数值积分的方法和离散数据的最优化问题求解方法。8.2 节将介绍两种常用的样条插值方式,三次分段多项式的插值方式和 B 样条插值方式,通过例子比较二者的不同,并介绍基于样条插值的数值微积分运算,还将演示该积分运算的结果优于 8.1 节介绍的方法。掌握了这两节就能较好地求解一维或多维数据的插值运算。

所谓函数逼近问题即由已知的样本点数据求取能对其有较好拟合效果的函数表达式的方法。最简单地,可以由多项式拟合更多的样本点,这样求解使得拟合误差极小化的多项式的系数即为多项式拟合或逼近所要解决的问题,8.3 节将介绍多项式拟合方法、多元函数的线性回归拟合方法与一般非线性函数的最小二乘参数拟合方法等。8.4 节将介绍一般给定函数的有理函数逼近方法,包括一般函数的连分式展开与逼近方法及一般函数的 Padé 近似方法等。8.5 节将介绍几种常用的特殊函数及曲线绘制。8.6 节还将介绍信号的相关分析、给定数据的快速 Fourier 变换技术、噪声滤波技术及滤波器设计等有关的信号处理入门知识及其 MATLAB 语言实现。

本章涉及的内容很多也可以由其他的非传统方法求解,如数据插值、拟合等内容将在 10.3 节中介绍用人工神经网络进行研究,而噪声滤波等内容将在 10.5 节中用小波变换的方式求解,有兴趣的读者可以阅读相关内容,并比较这些方法与本章介绍方法之间的优劣。

8.1 插值与数据拟合

8.1.1 一维数据的插值问题

1. 一维插值问题的求解

假设 $f(x)$ 是一维给定函数,函数本身未知,仅已知在相异的一组 N 个自变量 x_1, x_2, \dots, x_N 点处的函数值为 y_1, y_2, \dots, y_N , 这样一组采样点 $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ 又经常称为样本点,则由这些已知样本点的信息获得该函数在其他点上函数值的方法称为函数的插值。如果在这些给定点的范围内进行插值,称为内插,否则称为外插。如果从时间概念上理解这个问题,则对 x_N 以后点的插值又称为预报。

MATLAB 语言中提供了若干个插值函数,如一维插值函数 `interp1()`, 多项式拟合函数 `polyfit()` 等,还有大量的解决多维插值问题的函数。

一维插值函数 `interp1()` 的调用格式为 `y1 = interp1(x, y, x1, 'spline')`, 其中,

$\mathbf{x} = [x_1, x_2, \dots, x_N]^T, \mathbf{y} = [y_1, y_2, \dots, y_N]^T$ 两个向量分别表示给定的一组自变量和函数值数据, 可以用这两个向量来表示已知的样本点坐标, 且不要求 \mathbf{x} 向量为单调的。 \mathbf{x}_1 为用户指定的一组新的插值点的横坐标, 它可以是标量、向量或矩阵, 而得出的 \mathbf{y}_1 是在这一组插值点处的插值结果。这里给出的 'spline' 是本书推荐的插值方法选项, 表示采用三次样条插值的插值方法。除此之外, 还可以采用 'linear' (线性插值, 它在两个样本点间简单地采用直线拟合)、'nearest' (最近点等值方式) 和 'cubic' (三次 Hermite 插值, 当前版本的 MATLAB 中改为 'pchip') 等, 但从插值精度看建议使用 'spline' 选项。

例 8-1 假设已知的数据点来自函数 $f(x) = (x^2 - 3x + 5)e^{-5x} \sin x$, 试根据生成的数据进行插值处理, 得出较平滑的曲线。

解 根据给出的函数可以直接生成数据, 并绘制出如图 8-1(a) 所示的折线图。

```
>> x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x); plot(x,y,x,y,'o')
```

可以看出, 由这样的数据直接连线绘制出来的曲线十分粗糙, 可以再选择一组插值点, 然后直接调用 interp1() 函数进行插值近似。

```
>> x1=0:.02:1; y0=(x1.^2-3*x1+5).*exp(-5*x1).*sin(x1);
    y1=interp1(x,y,x1,'cubic'); y2=interp1(x,y,x1,'spline');
    y3=interp1(x,y,x1,'nearest'); y4=interp1(x,y,x1,'nearest');
    plot(x1,[y1',y2',y3',y4'],':',x,y,'o',x1,y0)
    e1=max(abs(y0(1:49)-y2(1:49))), e2=max(abs(y0-y3)), e3=max(abs(y0-y4))
```

分别选择各种拟合选项, 可以得出拟合结果与理论曲线, 它们之间的比较如图 8-1(b) 所示, 最大绝对误差分别为 $e_1 = 0.0177, e_2 = 0.0086, e_3 = 0.1598$ 。

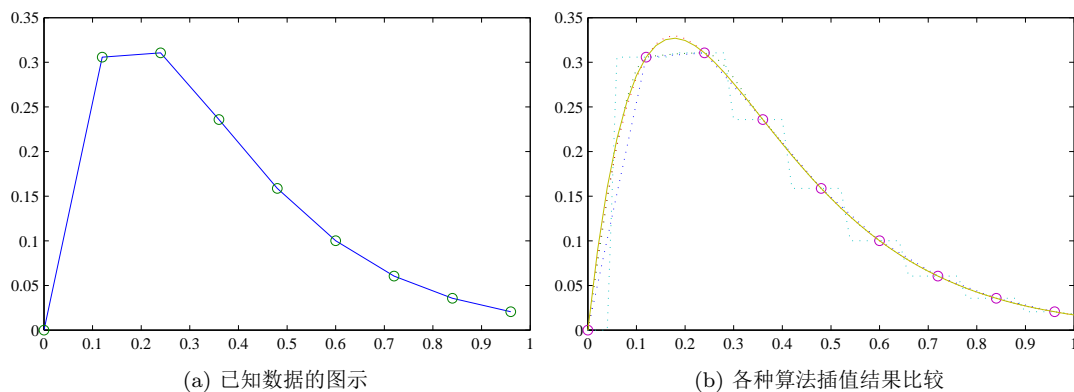


图 8-1 一维函数各种插值结果

可以看出, 默认的直线型拟合得到的曲线和图 8-1(a) 中的同样粗糙, 因为该方法就是对各个点的直接连线, 而 'nearest' 选项得出的拟合效果就更差了。采用 'cubic' 和 'spline' 选项的拟合更接近于理论值。事实上, 应用样条插值算法得出的插值十分逼近理论值, 甚至用肉眼难以分辨。所以样条函数插值在一维数据插值拟合中还是很有效的。样条插值还可以通过 spline() 函数和样条插值工具箱求出。

例 8-2 试编写一段程序, 允许用户利用插值方法手工绘制一条光滑的曲线。

解 在实际应用中经常需要用户自己选定几个点,然后就能绘制出一条光滑的曲线。选择点的方法可以由 MATLAB 下的 `ginput()` 函数实现,有了这些点,就可以编写出下面的函数,该函数即能实现所需的功能。在绘制图形时,若给出 `vis` 变量,则绘制的图形保留样本点处的圆圈,否则在绘制图形后删去圆圈。

```
function sketcher(vis)
x=[]; y=[]; i=1; h=[]; axes(gca);
while 1, [x0,y0,but]=ginput(1);
    if but==1, x=[x,x0]; y=[y,y0];
        h(i)=line(x0,y0); set(h(i),'Marker','o'); i=i+1; else, break
    end, end
if nargin==1, delete(h); end
xx=[x(1):(x(end)-x(1))/100:x(end)]; yy=interp1(x,y,xx,'spline'); line(xx,yy)
```

2. Lagrange 插值算法及应用

Lagrange 插值算法是一般代数插值教材中经常介绍的一类插值算法^[1],对已知的 x_i 、 y_i 点,可以求出 x 向量上各点处的插值为

$$\phi(x) = \sum_{i=1}^N y_i \prod_{j=1, j \neq i}^N \frac{x - x_j}{(x_i - x_j)} \quad (8-1-1)$$

根据上述算法,可以立即编写出相应的 MATLAB 函数为

```
function y=lagrange(x0,y0,x)
ii=1:length(x0); y=zeros(size(x));
for i=ii, ij=find(ii~=i); y1=1;
    for j=1:length(ij), y1=y1.*(x-x0(ij(j))); end
    y=y+y1*y0(i)/prod(x0(i)-x0(ij));
end
```

例 8-3 考虑一个著名的例子, $f(x) = 1/(1 + 25x^2)$, $-1 \leq x \leq 1$, 假设已知其中一些点的坐标,则可以采用下面的命令进行 Lagrange 插值,得出如图 8-2(a) 所示的插值曲线。

```
>> x0=-1+2*[0:10]/10; y0=1./(1+25*x0.^2);
    x=-1:.01:1; y=lagrange(x0,y0,x); % Lagrange 插值,对本例出现异常现象
    ya=1./(1+25*x.^2); plot(x,ya,x,y,'--')
```

由得出的插值曲线可见,用 Lagrange 插值得出的效果和精确值相差甚远,这种多项式阶次越高越发散的现象又称为 Runge 现象。所以对这个例子来说,传统的 Lagrange 算法失效。现在考虑 MATLAB 下的 `interp1()` 函数来解决同样的问题,通过下面的语句可以得出三次插值及样条插值的结果,并将各种插值结果与精确值绘制在相同的坐标系下,如图 8-2(b) 所示。可见,用 MATLAB 中提供的算法不存在 Runge 现象,一般可以放心大胆地直接使用。

```
>> y1=interp1(x0,y0,x,'cubic'); y2=interp1(x0,y0,x,'spline');
    plot(x,ya,x,y1,'--',x,y2,':')
```

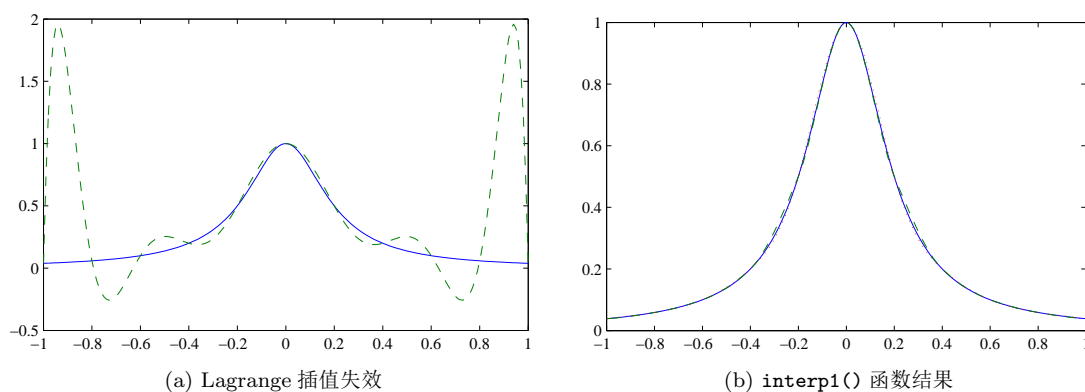


图 8-2 不同插值算法下的插值效果

3. 一维插值的预报问题求解

所谓预报,就是由现有的数据预测将来时刻的数据,比较常用的是由近些年的人口数预报将来某年的人口数,或已知近年来的产量预测未来某年的产量等。前面已经提及,预报问题又称为外插问题,可以使用 `interp1()` 函数直接求解,且使用 `'extrap'` 选项来表明预报问题。如果采用了 `'spline'` 选项则可以不使用 `'extrap'`,因为样条插值会自动处理预报问题。

例 8-4 例 2-29 给出了某省近些年的人口数据的 Excel 文件,以 5 年为步距构造出样本点,试以样本点为基础进行插值处理,得出 1949~2015 年的人口数量。

解 由于 Excel 文件包含了 1949~2011 年的人口数,所以 2009 年前的数据可以认为是内插,后面的数据属于预报。由下面的语句可以直接得出 1949~2015 年的人口数量插值结果,如图 8-3 所示。

```
>> X=xlsread('census.xls','B5:C67'); t=X(:,1); p=X(:,2);
t0=t(1:5:end); p0=p(1:5:end); t1=1949:2015;
y=interp1(t0,p0,t1,'spline','extrap'); plot(t,p,t1,y,t0,p0,'o')
```

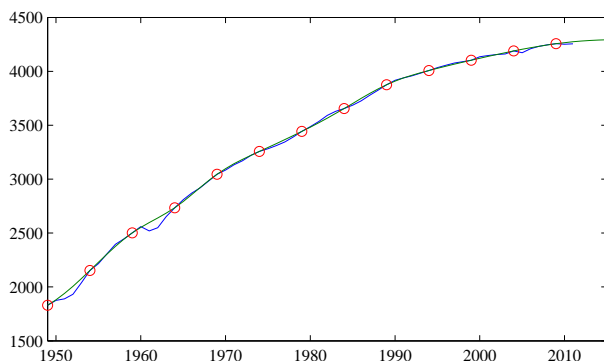


图 8-3 用插值和预报方法解决人口数量计算

人口数量预报问题是一个复杂的问题,采用人口动力学之类的动态模型描述可能更恰当,该模型还需要考虑其他的影响因素,如政策因素、自然灾害等,单纯采用普通的插值方法是不合适的。所以这里给出的预报只适合短时间之内的预报,不宜求解较长时间以后的预报问题。

8.1.2 已知样本点的定积分计算

3.7.1 节中给出了由样本点求解定积分的梯形方法及现成的 MATLAB 函数 `trapz()`。然而由例 3-50 中给出的结果看,若已知的样本点较稀疏,则得出的定积分近似将有很大的误差。如果被积函数用样条插值的方法从给定样本点直接求取,则可以编写出如下的 MATLAB 函数

```
function y=quadspln(x0,y0,a,b)
f=@(x)interp1(x0,y0,x,'spline'); y=integral(f,a,b);
```

该函数的调用格式为 $I = \text{quadspln}(x_0, y_0, a, b)$, 其中, x_0, y_0 为样本点构成的横纵坐标向量, a, b 为积分区间, 调用该函数将得出所需的定积分值。下面将通过例子演示该函数的应用。

例 8-5 试利用样条插值算法求解例 3-50 中给出的积分问题。

解 由原题知正弦函数积分的理论值为 2, 直接采用梯形法由数据求积分实际上是近似成折线与 x -轴围成区域的面积求取问题, 若步长较大, 则近似精度较差。这里将考虑在仅已知样本点的前提下利用插值方式描述被积函数, 进行积分求解的方法。由下面的语句即可得出定积分的值为 $I = 1.9999997$ 。

```
>> x0=0:pi/30:pi; y0=sin(x0); I=quadspln(x0,y0,0,pi)
```

可见, 这样的积分结果远比例 3-50 中用梯形法得出的结果精度高得多。如果给定的样本点更稀疏, 则下面可以由梯形法和插值法得出 $I_1 = 1.9835, I_2 = 2.0000$, 两种方法的优劣就更明显了。

```
>> x0=0:pi/10:pi; y0=sin(x0); I1=trapz(x0,y0), I2=quadspln(x0,y0,0,pi)
```

现在考虑更极端一点的例子, 即使已知再少的样本点, 例如在 $x \in [0, \pi]$ 区间内仅已知 5 个不均匀分布的样本点, 仍可以考虑采用插值和 `quad1()` 函数结合的方法求取积分值, $I_1 = 2.019, I_2 = 1.8416$ 。可见, 这时梯形法有很大的误差。可以给出如下的 MATLAB 语句

```
>> x0=[0,0.4,1 2,pi]; y0=sin(x0); % 生成样本点
plot(x0,y0,x0,y0,'o') % 绘制出的样本点折线如图 8-4(a) 所示
I1=quadspln(x0,y0,0,pi) % 大约有 1% 的相对误差, 应该说是相当精确的
I2=trapz(x0,y0) % 用 trapz() 函数将得出很大的相对误差 (7.9%)
```

事实上, 即使在这样稀疏的样本点下, 也可以用样条插值法得出相当好的拟合效果。用下面的 MATLAB 语句可以绘制出样条插值的结果与理论值之间的比较, 如图 8-4(b) 所示, 其中曲线的实线部分表示原函数, 虚线表示插值效果。

```
>> x=[0:0.01:pi,pi]; y0a=sin(x); y=interp1(x0,y0,x,'spline');
plot(x0,y0,x,y,':',x,y0a,x0,y0,'o')
```

例 8-6 仍然考虑例 3-51 中的振荡函数, 假设已知其中的 150 个数据点, 试采用 `quadspln()` 函数计算出该定积分的值, 并检验其精度。

解 因为这里假设原函数未知, 仅已知数据点, 所以用解析积分的算法是不可行的。若想求出该积分的数值解, 则可以给出下面的指令, 得出 $I = 0.0666722$ 。

```
>> x=[0:3*pi/2/200:3*pi/2]; y=cos(15*x); I=quadspln(x,y,0,3*pi/2)
```

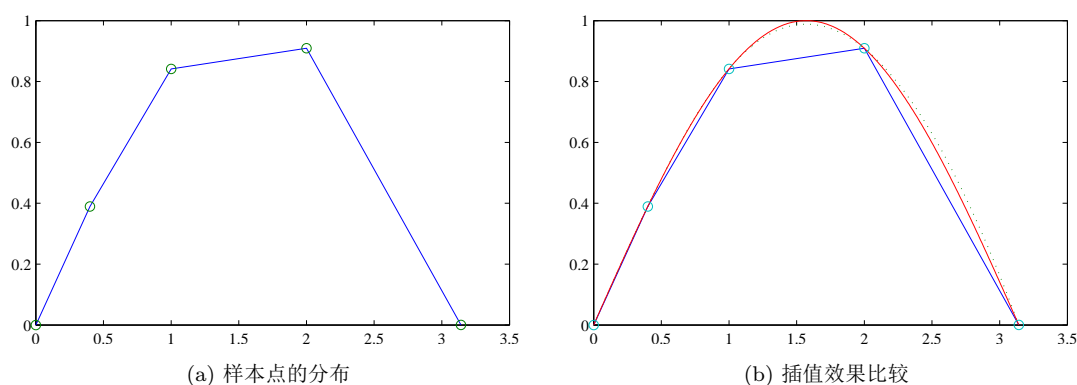


图 8-4 样本点极稀疏时的插值效果

可见,这样的结果还是很精确的。下面可以绘制出原始函数和插值曲线,如图 8-5 所示。可以看出,这样的曲线拟合效果还是很好的,从图形上和理论曲线基本看不出区别。

```
>> x0=[0:3*pi/2/1000:3*pi/2]; y0=cos(15*x0);
    y1=interp1(x,y,x0,'spline'); plot(x0,y0,x0,y1,':')
```

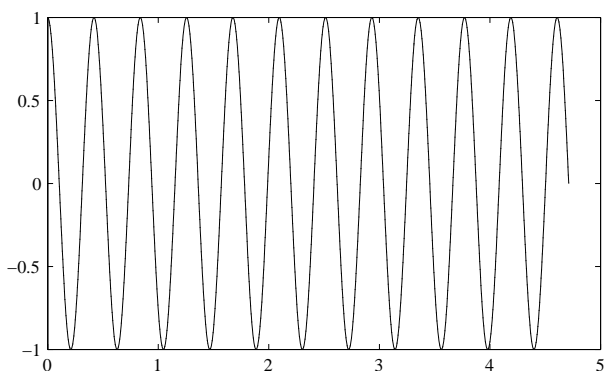


图 8-5 原型函数与样条插值曲线

对此例子来说,由于被积函数本身变化较大,给定的样本点相对较少,所以未能提供充足的信息量,来获得更高精度的积分值。若想进一步提高积分的精度,则唯一解决途径是提供更密的样本点。

8.1.3 二维网格数据的插值问题

MATLAB 下提供了二维插值的函数,如 `interp2()`,该函数的调用格式为

```
z1 = interp2(x0,y0,z0,x1,y1,'spline')
```

其中 x_0, y_0, z_0 为已知的数据,而 x_1, y_1 为由插值点构成的新的网格参数,返回的 z_1 矩阵为在所选插值网格点处的函数近似值。插值方法 'spline' 仍然为 'linear'、'cubic' 替换。和一元函数插值类似,其中最好的方法还是样条插值 'spline',本节仍将通过例子演示、比较各种算法。

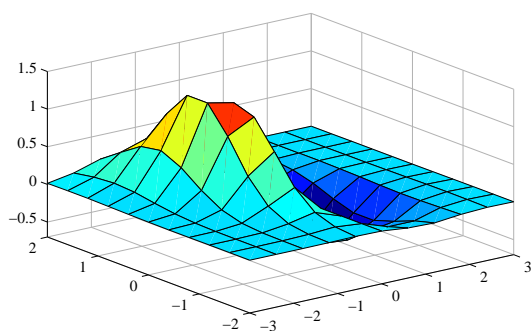
例 8-7 假设由二元函数 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$ 可以计算出一些较稀疏的网格数据,试根据这些数据对整个函数曲面进行各种插值拟合,并比较拟合结果。

解 考虑给出的二元函数,假设仅知其中较少的数据,则可以由下面的命令绘制出已知数据的网格图,如图 8-6(a) 所示。从图 8-6(a) 可以看出,由这些数据绘制的图形还是很粗糙的。

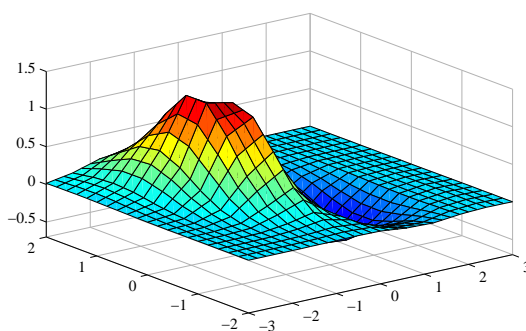
```
>> [x,y]=meshgrid(-3:.6:3, -2:.4:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
    surf(x,y,z), axis([-3,3,-2,2,-0.7,1.5])
```

选较密的插值点,则可以用下面的 MATLAB 语句采用默认的插值算法进行插值,得出的结果如图 8-6(b) 所示。

```
>> [x1,y1]=meshgrid(-3:.2:3, -2:.2:2);
    z1=interp2(x,y,z,x1,y1); surf(x1,y1,z1), axis([-3,3,-2,2,-0.7,1.5])
```



(a) 已知数据的图示



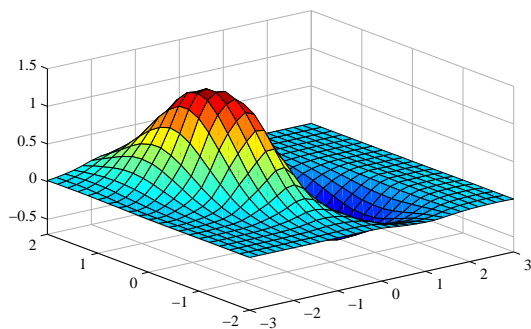
(b) 线性选项插值结果

图 8-6 二维函数插值比较

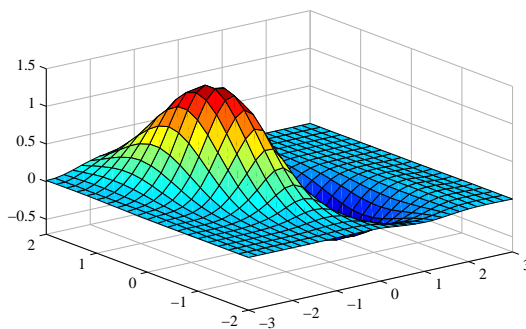
可以看出,默认的线性插值方法还原后的三维表面图在很多地方还是太粗糙。可以用下面的命令分别由立方插值选项和样条插值选项来进行插值,得出的结果如图 8-7 所示。

```
>> z1=interp2(x,y,z,x1,y1,'cubic'); z2=interp2(x,y,z,x1,y1,'spline');
    surf(x1,y1,z1), figure; surf(x1,y1,z2)
```

可以看出,这样的插值效果都是比较理想的。



(a) 立方插值算法



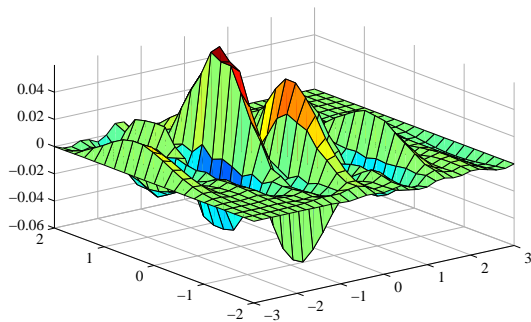
(b) 样条插值算法

图 8-7 二维函数其他插值结果比较

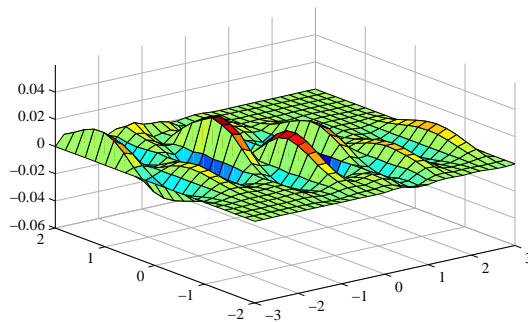
通过下面的误差分析还可以进一步比较两种算法,因为网格已知,故可以由已知函数计算出 z 的精确值,所以可以通过下面的语句求出两种算法得出的矩阵 z_1 和 z_2 与真值 z 之间误差的绝对值,分别如图 8-8(a)、(b) 所示。可以看出,选择样条方法的插值精度要远高于立方插值算法,所以在

实际应用中建议使用 'spline' 插值选项。

```
>> z=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1); % 新网格各点的函数值
surf(x1,y1,z-z1), figure; surf(x1,y1,z-z2)
```



(a) 立方插值算法



(b) 样条插值算法

图 8-8 二维函数的误差

8.1.4 二维散点分布数据的插值问题

通过上面的例子可以看出, MATLAB 提供的二维插值函数还是能较好地进行二维插值运算的。但该函数有一个重要的缺陷, 就是它只能处理以网格形式给出的数据, 如果已知数据不是以网格形式给出的, 则用该函数是无能为力的。在实际应用中, 大部分问题都是以实测的 (x_i, y_i, z_i) 散点给出的, 所以不能直接使用函数 `interp2()` 进行二维插值。

MATLAB 语言中提供了一个更一般的 `griddata()` 函数, 用来专门解决这样的问题。该函数的调用格式为 `z = griddata(x0, y0, z0, x, y, 'v4')`, 其中, x_0, y_0, z_0 是已知的样本点坐标, 这里并不要求是网格型的, 可以是任意分布的, 均由向量给出。 x, y 是期望的插值位置, 可以是单个点, 可以是向量或网格型矩阵, 得出的 z 的维数应该和 x, y 一致, 表示插值的结果。'v4' 选项是指采用 MATLAB 4.0 版本中提供的插值算法, 公认该算法效果较好, 但没有一个正式的名称, 所以这里用 'v4' 表明采用该算法。除了 'v4' 选项外, 还可以使用 'linear'、'cubic' 和 'nearest' 等算法, 但效果一般比 'v4' 差很多。

例 8-8 仍考虑原型函数 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$, 在 $x \in [-3, 3], y \in [-2, 2]$ 矩形区域内随机选择一组 (x_i, y_i) 坐标, 就可以生成一组 z_i 的值。以这些值为已知数据, 用一般散点数据插值函数 `griddata()` 进行插值处理, 并进行误差分析。

解 这里选择 200 个随机数构成的点, 则可以用下面的语句生成 x, y, z 向量, 但由于这些数据不是网格数据, 所以得出的数据向量不能直接用三维曲面的形式表示。但可以通过下面的语句将各个样本点在 $x-y$ 平面上的分布形式显示出来, 如图 8-9(a) 所示, 也可以绘制出样本点的三维分布, 如图 8-9(b) 所示。可以看出, 这些分布点还是比较均匀的, 但由此绘制的三维图形可读性很差, 所以需要对其进行插值处理, 得出可读性好的三维曲面表示。

```
>> x=-3+6*rand(200,1); y=-2+4*rand(200,1);
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); % 生成已知数据
plot(x,y,'x'), figure, plot3(x,y,z,'x')
```

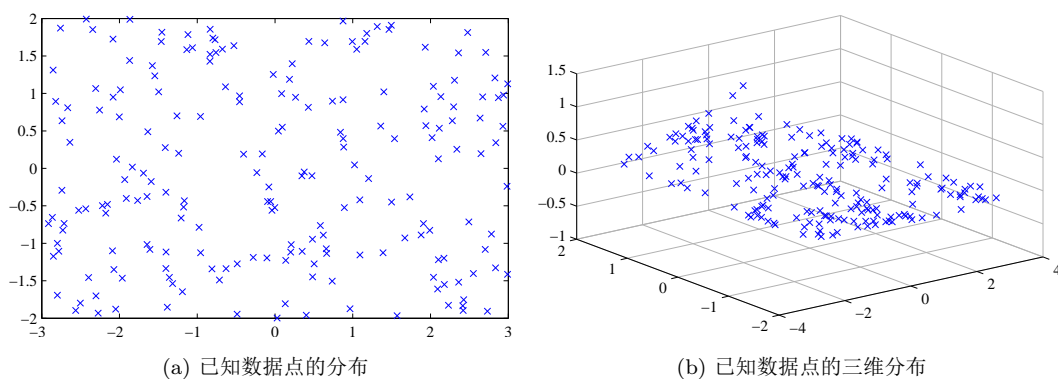



图 8-9 已知样本数据显示

仍选定按照例 8-7 中给出的方法生成网格矩阵,则可以用 'cubic' 和 'v4' 两种算法获得插值结果,还可以绘制出拟合后的曲面形式,分别如图 8-10 (a)、(b) 所示。可以看出,用 'v4' 算法得出的结果效果明显更好些,而用 'cubic' 插值算法得出的曲面在某些点上可能残缺不全。

```
>> [x1,y1]=meshgrid(-3:.2:3, -2:.2:2); z1=griddata(x,y,z,x1,y1,'cubic');
surf(x1,y1,z1), z2=griddata(x,y,z,x1,y1,'v4'); figure; surf(x1,y1,z2)
```

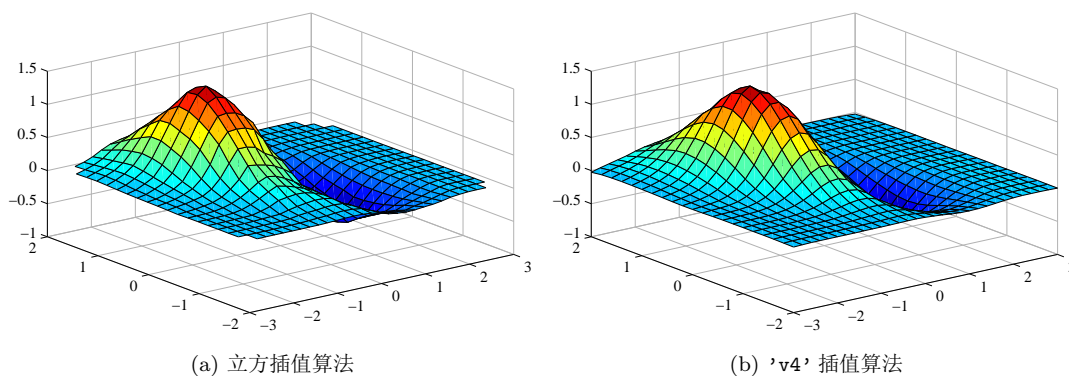


图 8-10 二维函数各种插值结果比较

还可以进一步进行误差分析。用下面的语句可以先计算出在新网格点处函数值的精确解,并用这些点和两种方法计算出来的误差,得出如图 8-11 (a)、(b) 所示的误差曲面。可见,用 'v4' 选项的插值结果明显优于立方插值算法,所以在实际应用中建议采用 'v4' 算法。

```
>> z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1); % 新网格各点的函数值
surf(x1,y1,z0-z1); figure; surf(x1,y1,z0-z2)
```

例 8-9 前面已经提及,给定的样本点在 x - y 平面分布较均匀,现在人为地剔除某区域的样本点,表明已知数据分布不均匀,这时再进行插值分析,观察插值效果。

解 由已知的 x, y, z 矩阵人为地剔除在以 $(-1, -1/2)$ 点为圆心,以 0.5 为半径的圆内的点,则可以采用下面语句

```
>> x=-3+6*rand(200,1); y=-2+4*rand(200,1); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
ii=find((x+1).^2+(y+0.5).^2>0.5^2); % 找出不满足条件的点坐标
x=x(ii); y=y(ii); z=z(ii); plot(x,y,'x')
```

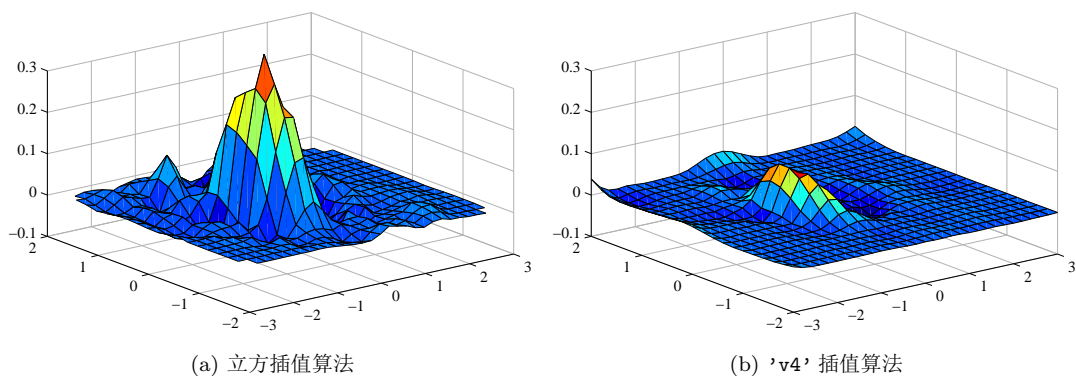


图 8-11 二维函数插值误差分析

```
t=[0:.1:2*pi,2*pi]; x0=-1+0.5*cos(t); y0=-0.5+0.5*sin(t); line(x0,y0)
```

这时将得出如图 8-12(a) 所示的样本点分布图,同时还叠印了圆。可见,在该圆内样本点确实均已经剔除。用新的样本点可以拟合出曲面,如图 8-12(b) 所示。可见,拟合效果还是很好的。

```
>> [x1,y1]=meshgrid(-3:.2:3, -2:.2:2); z1=griddata(x,y,z,x1,y1,'v4');  
surf(x1,y1,z1)
```

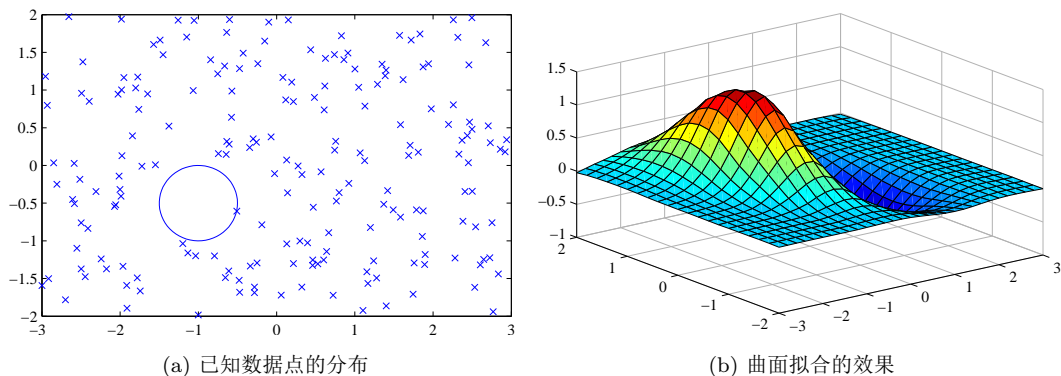


图 8-12 修改样本后的分布及拟合效果

现在可以进行误差分析了,用下面的语句可以得出误差,并绘制出误差曲面,如图 8-13(a) 所示,可以看出,尽管原始样本点数据被人为剔除掉了一个较大的区域,但拟合效果还是很好。

```
>> z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1); surf(x1,y1,z0-z1)
```

读者还可以给出下面的语句绘制出误差的等高线图,同时叠印出样本点分布,如图 8-13(b) 所示。从图中可以看出,原始样本点数据分布稀少的地方(在本例子中人为剔除样本点的区域和其他几个样本点稀少的区域)拟合效果不甚理想,其余部分拟合效果较好。

```
>> contour(x1,y1,z0-z1,30); hold on; plot(x,y,'x'); line(x0,y0)
```

由此可以得出结论,数据插值拟合效果的好坏在很大程度上取决于数据点的分布情况,如果某个区域的数据点分布较少,则很难通过插值的方式恢复该区域,因为这个区域已知的信息量是不足以高精度恢复数据的,所以为使得该函数的拟合精度较高,建议在数据采集时均匀地多选择一些点。

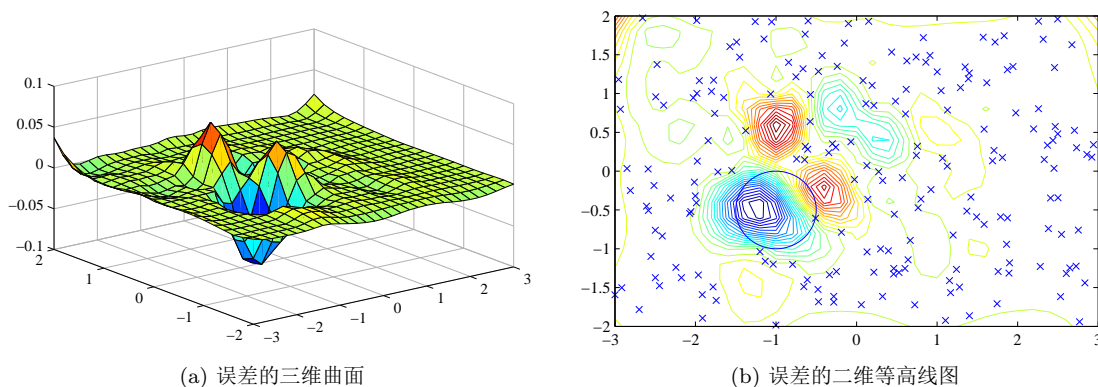


图 8-13 二维函数插值误差分析

8.1.5 高维插值问题

三维的网格数据生成仍然可以用 `[x,y,z] = meshgrid(x1,y1,z1)` 函数实现, 其中, x_1, y_1, z_1 为这三维所需要的分割形式, 应该是向量形式给出的, 返回的 x, y, z 为网格的数据生成, 均为三维数组。

n 维网格数据的生成还可以使用 `ndgrid()` 函数 `[x1,...,xn] = ndgrid(v1,...,vn)`, 其中, v_1, \dots, v_n 为这 n 维所需要的分割形式, 应该是向量形式给出的, 返回的 x_1, \dots, x_n 为网格数据生成的效果, 这时返回的 x_i 为 n 维数组。

`ndgrid()` 函数生成的数据格式与 `meshgrid()` 函数有些区别。首先, `meshgrid()` 函数只能用于二维或三维网格数据的生成, 而 `ndgrid()` 函数无此限制; 此外, 如果生成二维网格数据, `[x,y] = meshgrid(...)` 与 `[x1,y1] = ndgrid(...)` 在网格数据上满足 $x = x_1^T$, $y = y_1^T$ 。如果生成三维数据, `[x,y,z] = meshgrid(...)` 与 `[x1,y1,z1] = ndgrid(...)` 函数生成 z 与 z_1 完全一致, $x(:, :, i)$ 数组对每个 i 均为固定矩阵, x_1^T, y, y_1^T 数组也是一样, 且 $x_1(:, :, i) = x_1(:, :, i)^T$ 。除了网格点信息外, 样本点函数值也需同时做相应的转置处理。可以根据前面介绍的方法编写一个简单的转换函数实现两者的相互转换。

```
function [x1,y1,z1,v1]=mesh2nd(x,y,z,v)
switch nargin
case 3, x1=x.'; y1=y.'; z1=z.';
case 4, z1=z;
    for i=1:size(x,3),
        x1(:, :, i)=x(:, :, i).'; y1(:, :, i)=y(:, :, i).'; v1(:, :, i)=v(:, :, i).';
    end
    otherwise, error('Error in input arguments')
end
```

若已知按空间网格取的样本点, 则可以用 `interp3()` 函数或更一般的 `interpnn()` 函数进行插值运算。这些函数的调用格式和 `interp2()` 一致, 这里不详细介绍了。若已知样本点是散点形式给出, 则类似地可以调用 `griddata()` 函数对其进行插值拟合, 早期版本则应该使用 `griddata3()` 或 `griddatan()` 插值运算。

例 8-10 由例 2-38 给出的三元函数 $V(x, y, z) = \sqrt{x^x + y^{(x+y)/2} + z^{(x+y+z)/3}}$ 生成一组样本点数据,然后用插值方法得出拟合结果,并给出插值结果的四维表示与拟合误差。

解 先调用 meshgrid() 函数生成一组较稀疏的三维网格坐标点,则可以求出样本点处的函数值 V 。利用 vol_visual4d() 函数对比插值结果 V_1 和理论值 V_0 可见,二者得出的图形在已知区域边界附近稍有区别外绝大部分区域没有区别,得出的体视化切面图如图 8-14 所示。

```
>> [x,y,z]=meshgrid(0:0.3:2); [x0 y0 z0]=meshgrid(0:0.1:2);
V=sqrt(x.^x+y.^((x+y)/2)+z.^((x+y+z)/3));
V0=sqrt(x0.^x0+y0.^((x0+y0)/2)+z0.^((x0+y0+z0)/3));
V1=interp3(x,y,z,V,x0,y0,z0,'spline'); vol_visual4d(x0,y0,z0,V1)
```

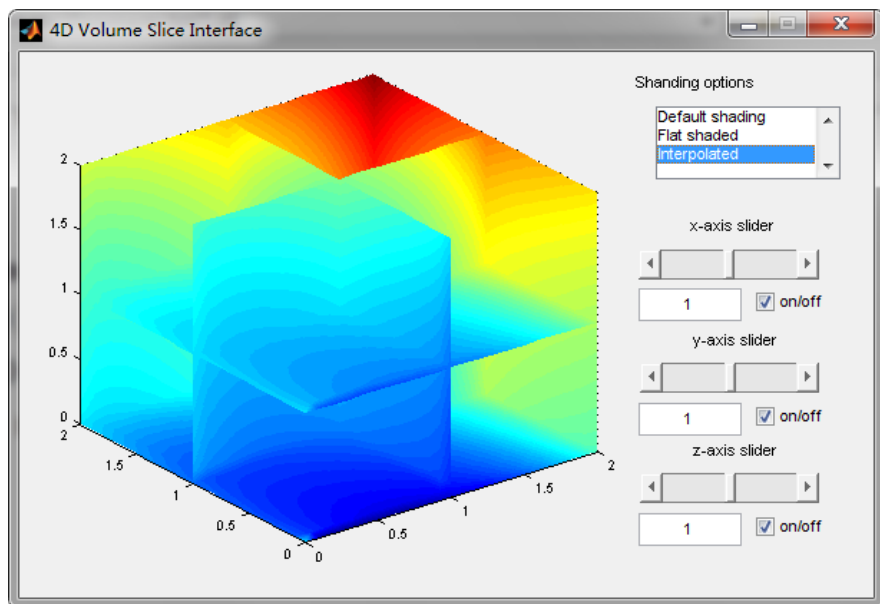


图 8-14 原函数理论值与插值结果的切面显示

8.1.6 基于样本数据点的离散最优化问题求解

在实际应用中,某一个需要优化的目标函数有时其原型是未知的,只有一些相应的、离散分布的样本数据点,这样,就可以采用样条插值或其他插值方法去拟合目标函数,从而优化这样的目标函数。下面将通过例子来演示这样的最优化问题求解方法。

例 8-11 重新考虑例 8-7 中的函数,假设已经测出了其中一些离散数据点,试根据这些离散点搜索对应函数的最小值,并检验所得出的结果。

解 仿照前面的例子,可以首先生成一些离散点,再由这些离散点通过插值方法构造目标函数的匿名函数,对该匿名函数进行优化,则可以得出最优解为 $x = 0.6069, y = -0.3085$ 。下面的语句还可以绘制出原目标函数的等高线,如图 8-15 所示。可见,这样得出的数值解是正确的。

```
>> x=-3+6*rand(200,1); y=-2+4*rand(200,1); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
```

```
f=@(p)griddata(x,y,z,p(1),p(2),'v4'); x=fminunc(f,[0,0]) % 优化搜索
[x0,y0]=meshgrid(-3:0.1:2,-2:0.1:2);
z0=(x0.^2-2*x0).*exp(-x0.^2-y0.^2-x0.*y0); contour(x0,y0,z0,30)
```

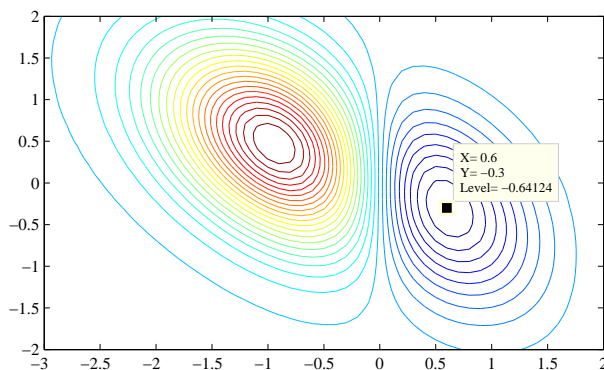


图 8-15 离散最优化解的图形显示

8.2 样条插值与数值微积分问题求解

前面介绍的插值函数是简单的插值算法。MATLAB 提供了一个样条插值工具箱,可以更好地求解样条插值问题。还可以借助样条数据结构容易地求解微积分问题。所以本节可以认为是 8.1 节以及 3.6 节和 3.7 节的拓展。

样条函数是函数逼近的一种方法,其中三次样条函数和 B 样条函数是两类常用的样条函数。下面首先分别介绍这两类样条函数的表示方法,然后介绍利用 MATLAB 的样条插值工具箱求解数值微分和数值积分的问题。

8.2.1 样条插值的 MATLAB 表示

1. 三次样条函数及其 MATLAB 表示

三次样条函数的定义是,已知平面上 n 个点 (x_i, y_i) ($i = 1, 2, \dots, n$), 其中, $x_1 < x_2 < \dots < x_n$, 这些点称为样本点。如果有某函数 $S(x)$ 满足下面 3 个条件,则称 $S(x)$ 为经过这 n 个点的三次样条函数。

- (1) $S(x_i) = y_i$ ($i = 1, 2, \dots, n$), 亦即该函数经过这些样本点。
- (2) $S(x)$ 在每个子区间 $[x_i, x_{i+1}]$ 上为三次多项式

$$S(x) = c_{i1}(x - x_i)^3 + c_{i2}(x - x_i)^2 + c_{i3}(x - x_i) + c_{i4}$$

- (3) $S(x)$ 在整个区间 $[x_1, x_n]$ 上有连续的一阶及二阶导数。

MATLAB 的样条插值工具箱中提供了 `csapi()` 函数来定义一个三次样条函数类,其调用格式很简单 `S = csapi(x, y)`, 其中, $\mathbf{x} = [x_1, x_2, \dots, x_n]$, $\mathbf{y} = [y_1, y_2, \dots, y_n]$ 为样本点,得出的 S 是一个三次样条函数对象,其成员变量包括子区间点、各个三次多项式系数等。

样条函数对象的插值结果可以由 `fnplt()` 绘制出来,对给定的向量 \mathbf{x}_p ,也可以由 `fnval()` 函数计算出来。这两个函数的调用格式为 `fnplt(S)` 和 `y_p = fnval(S, x_p)`, 其中得出的 \mathbf{y}_p 为 \mathbf{x}_p 上各点的插值结果。

例 8-12 试求出例 8-6 中给出的稀疏数据的三次样条插值结果。

解 由三次样条函数的调用语句可以立即得出给定数据的样条插值结果,并和原理论数据同时绘制出来,如图 8-16 所示。

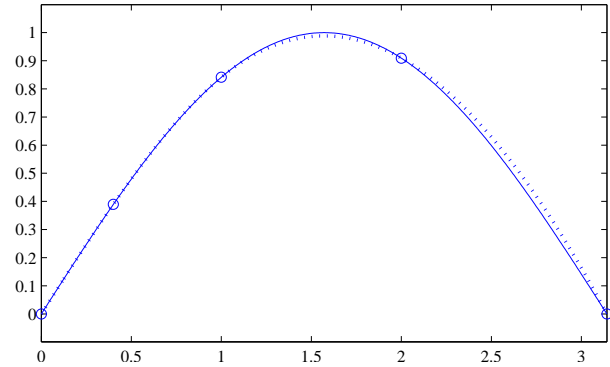


图 8-16 给定稀疏数据的三次样条插值效果

```
>> x0=[0,0.4,1 2,pi]; y0=sin(x0); sp=csapi(x0,y0), fnplt(sp,':');  
hold on, ezplot('sin(t)',[0,pi]); plot(x0,y0,'o')
```

其中得出的每一行为对应区间内的三次多项式系数,在表 8-1 中给出。例如,在 (0.4,1) 区间内,插值多项式可以表示为 $S_2(x) = -0.1627(x - 0.4)^3 - 0.1876(x - 0.4)^2 + 0.9245(x - 0.4) + 0.3894$ 。

表 8-1 分段三次多项式系数表

区间	c_0	c_1	c_2	c_3
(0,0.4)	-0.16265031	0.007585654	0.99653564	0
(0.4,1)	-0.16265031	-0.18759472	0.92453202	0.38941834
(1,2)	0.024435717	-0.48036529	0.52375601	0.84147098
(2, π)	0.024435717	-0.40705814	-0.36366741	0.90929743

例 8-13 试用三次样条插值的方法对例 8-1 中给出的数据进行拟合。

解 用下面的语句可以建立起描述已知数据的样条插值类,并得出各段三次多项式系数,由表 8-2 给出,根据该表可以用多项式方式计算出样条插值的值。

```
>> x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x); sp=csapi(x,y); fnplt(sp)  
c=[sp.breaks(1:4)' sp.breaks(2:5)' sp.coefs(1:4,:),... % 生成表 8-2 数据  
sp.breaks(5:8)' sp.breaks(6:9)' sp.coefs(5:8,:) ];
```

csapi() 函数还可以处理多个自变量的网格数据三次样条插值类,其调用格式为

```
S = csapi({x1,x2,...,xn},z)
```

其中, x_i 为自变量的网格标志, z 为网格数据的样本点,得出的 S 是三次样条函数对象。

例 8-14 试用三次样条插值方法得出例 8-7 中给出网格数据的样条插值拟合,并绘制出曲面。

解 用下面的语句自然就能得出样条插值对象 sp,并绘制出如图 8-17 所示的曲面。可见,这样的插值结果与 interp2() 函数得出的完全一致。

表 8-2 分段三次多项式样条插值系数表

分段 区间	三次多项式系数				分段 区间	三次多项式系数			
	c_1	c_2	c_3	c_4		c_1	c_2	c_3	c_4
(0, 0.12)	24.7396	-19.359	4.5151	0	(0.48, 0.6)	-0.2404	0.7652	-0.5776	0.1588
(0.12, 0.24)	24.7396	-10.4526	0.9377	0.3058	(0.6, 0.72)	-0.4774	0.6787	-0.4043	0.1001
(0.24, 0.36)	4.5071	-1.5463	-0.5022	0.3105	(0.72, 0.84)	-0.4559	0.5068	-0.2621	0.0605
(0.36, 0.48)	1.9139	0.07623	-0.6786	0.2358	(0.84, 0.96)	-0.4559	0.3427	-0.1601	0.03557

```
>> x0=-3:.6:3; y0=-2:.4:2; [x,y]=ndgrid(x0,y0); % 注意这里只能用ndgrid() 函数
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); % 否则生成的z矩阵顺序有问题
sp=csapi({x0,y0},z); fnplt(sp);
```

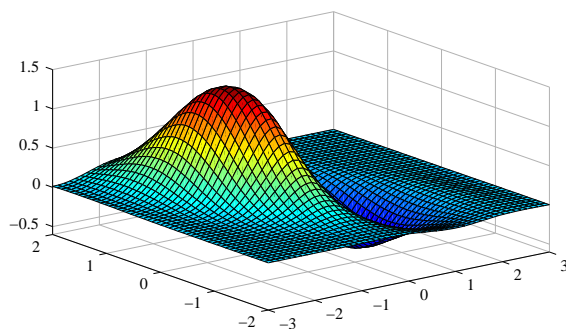


图 8-17 二维函数插值结果

注意,这里的 z 矩阵应该是基于用 `ndgrid()` 函数生成的 x 和 y 矩阵,而不能用 `meshgrid()` 函数生成,因为用其生成的 z 矩阵数据排列方式和样条插值工具箱不一致。如果原始数据是由 `meshgrid()` 格式表示的,则应该采用 `mesh2nd()` 函数先作相应的格式转换。

2. B 样条函数及其 MATLAB 表示

B 样条插值为另一类常用的样条函数,其数学描述方式比较不易理解,故这里只介绍该类样条函数对象的建立函数 `spapi()`,而不介绍其数学描述。若已知样本点数据向量 x 和 y ,则可以通过 `S = spapi(k,x,y)` 语句直接建立起 B 样条插值对象 S ,其中, k 为用户选定的 B 样条阶次。一般选择 $k=4,5$ 能得出较好的插值效果,对某些特定的问题适当提高 k 值能改善插值效果。

例 8-15 分别用 B 样条函数对例 8-12 和例 8-13 中给出的数据进行 5 次 B 样条函数拟合,并与三次分段多项式样条函数拟合的结果相比较。

解 先考虑例 8-12 中给出的数据,可以用下面的语句进行拟合,得出如图 8-18(a) 所示的拟合效果。其中的 B 样条插值效果几乎看不出和理论曲线的差异。

```
>> x0=[0,0.4,1 2,pi]; y0=sin(x0); ezplot('sin(t)',[0,pi]); hold on
sp1=csapi(x0,y0); fnplt(sp1,'--'); % 三次分段多项式样条插值
sp2=spapi(5,x0,y0); fnplt(sp2,':') % 5 次 B 样条插值
```

可见, 5 次 B 样条插值的效果远远优于三次分段多项式的拟合效果。对例 8-13 中给出的数据进行拟合, 将得出如图 8-18 (b) 所示的效果, B 样条亦远远优于三次样条插值。

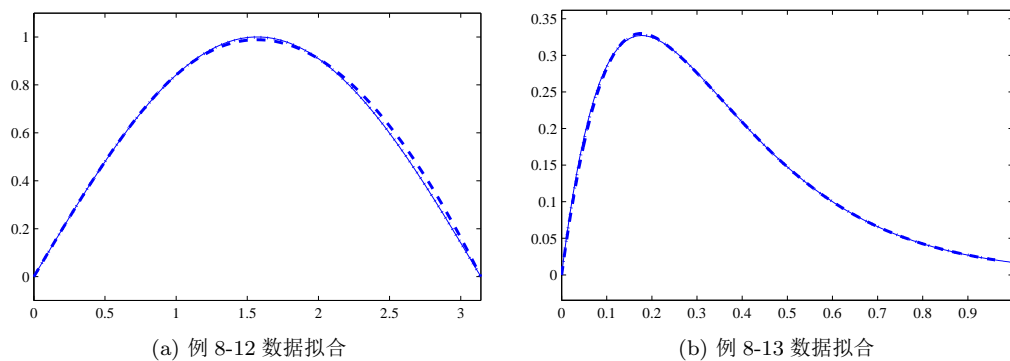


图 8-18 基于样条插值的曲线拟合效果

```
>> x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x);
    ezplot('x^2-3*x+5)*exp(-5*x)*sin(x)', [0,1]), hold on
    sp1=csapi(x,y); fnplt(sp1,'--'); sp2=spapi(5,x,y); fnplt(sp2,':')
```

8.2.2 基于样条插值的数值微积分运算

既然用样条插值的方法能由给定的数据进行曲线拟合, 且在给定数据较稀疏的情形拟合效果也是很理想的, 故可以利用插值对象对给定数据函数进行微积分运算。和 3.6 节与 3.7 节介绍的算法相比, 基于样条插值的数值微分算法有其特色, 更适用于给定样本数据较稀疏时的数值微分。从数值积分的角度看, 这里得出的数值积分是数值积分的函数, 亦即求取 $F(x) = \int_{x_0}^x f(t)dt$ 的值, 而不是单纯的定积分值, 其中 x_0 是用户指定的积分区域左端边界值。当然, 单纯的定积分也可以用积分函数得出, 即 $I = F(b) - F(a)$ 。

1. 基于样条插值的数值微分运算

基于样条函数的数值微分运算可以由 `fnder()` 函数直接计算出来, 调用格式如下

```
 $S_d = \text{fnder}(S, k),$  % 该函数可以求取  $S$  的  $k$  阶导数
 $S_d = \text{fnder}(S, [k_1, \dots, k_n]),$  % 可以求取多变量函数的偏导数
```

该函数的两种调用方法中, 前一种方法能直接求取 S 样条对象的 k 阶导数, 得出的结果仍然是样条对象 S_d 。后一种调用格式中, 可以对多变量样条对象进行偏导数求取。

例 8-16 考虑例 8-13 中给出的数据点, 试用三次分段多项式样条函数与 B 样条插值函数求出该函数的导数, 并与理论推导结果相比较。

解 可以用下面的语句生成原始数据, 并分别建立起三次分段多项式样条函数与 B 样条函数的数据类, 这样就可以调用 `fnder()` 函数求出该函数的导数, 并得出如图 8-19 所示的曲线。

```
>> syms x; f=(x^2-3*x+5)*exp(-5*x)*sin(x); ezplot(diff(f), [0,1]), hold on
    x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x);
```



```
sp1=csapi(x,y); dsp1=fnder(sp1,1); fnplt(dsp1,'--')
sp2=spapi(5,x,y); dsp2=fnder(sp2,1); fnplt(dsp2,':');
```

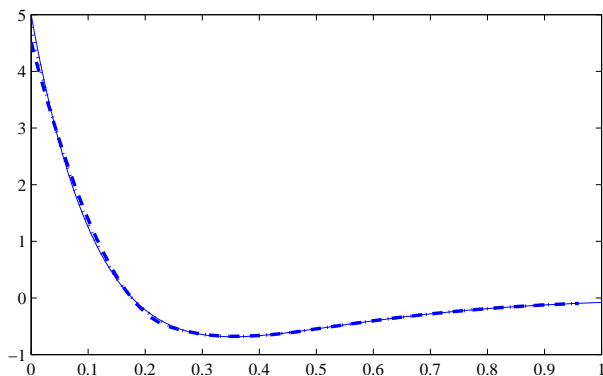


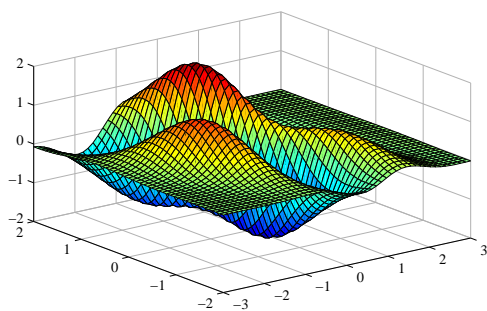
图 8-19 基于样条插值的函数数值微分结果

在图 8-19 中同时还绘制理论曲线。可见,用 B 样条拟合的数值微分结果是相当精确的,用三次分段多项式样条插值算法得出的微分效果也是很理想的,因为给出的数据点是很稀疏的,用 3.6 节中给出的算法无法得出这样的结果。

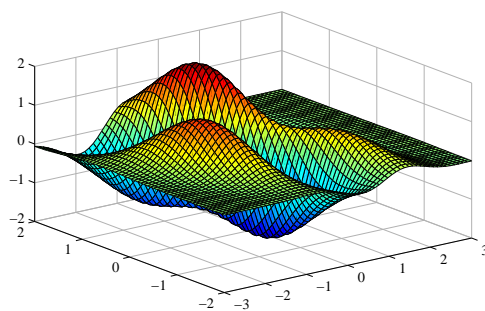
例 8-17 试由例 8-14 中给出的数据拟合 $\partial^2 z / (\partial x \partial y)$ 的曲面,并将得出的结果与解析解法绘制出的曲面相比较。

解 由下面给出的语句可以直接生成数据,进行 B 样条函数拟合,并对得出的结果进行求导,绘制出如图 8-20 (a) 所示的曲面。

```
>> x0=-3:.3:3; y0=-2:.2:2; [x,y]=ndgrid(x0,y0);
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
sp=spapi({5,5},{x0,y0},z); dspxy=fnder(sp,[1,1]); fnplt(dspxy)
```



(a) 样条拟合结果



(b) 理论推导结果

图 8-20 所需二阶偏导数的曲面表示

当然,下面的语句可以用理论的方法推导出所需的偏导数,并绘制出其曲面,如图 8-20 (b) 所示。可见,这样得出的曲面与样条插值得出的结果完全一致。由此可以看出,基于样条插值的数值偏导数算法函数是可靠的。

```
>> syms x y; z=(x^2-2*x).*exp(-x^2-y^2-x*y);
```

```
ezsurf(diff(diff(z,x),y),[-3 3],[-2 2])
```

2. 基于样条插值的数值积分运算

下面将介绍使用分段三次样条函数和五次 B 样条函数来逼近被积函数,从而求取该函数积分函数的方法。这里要介绍的方法和 8.1.2 节介绍的 `quadspln()` 函数是有区别的,`quadspln()` 函数介绍的是求取某一区间内的定积分值,而这里介绍的 `fnint()` 方法可以用来求取积分函数的值,当然也能用于求取定积分的值。积分函数可以由 $f_i = \text{fnint}(S)$ 求出,其中, f_i 为向量,返回 x 各点处的积分函数值,因为不定积分通常可以在积分结果上加一个常数,所以实际的不定积分值应该是该结果的上下平移。该不定积分结果还可以用于 $[a, b]$ 区间定积分的求解,即 $I = \text{diff}(\text{fnint}(S, [a, b]))$ 。

例 8-18 仍考虑例 8-5 中较稀疏的样本点,试用样条积分的方式求出定积分及积分函数。

解 下面的语句可以用两种形式建立起插值对象,用 `fnint()` 函数可以分别得出积分函数,并求出所需的定积分值。可见,这样得出的结果远远比例 8-5 中得出的结果精确得多,用 B 样条甚至能在极稀疏的样本点前提下得出相当高精度的定积分结果,其中 $I_1 = 2.01905234585838$, $I_2 = 1.99994177102332$ 。

```
>> x=[0,0.4,1 2,pi]; y=sin(x);
    sp1=csapi(x,y); a=fnint(sp1,1); xx=fnval(a,[0,pi]); I1=xx(2)-xx(1)
    sp2=spapi(5,x,y); b=fnint(sp2,1); xx=fnval(b,[0,pi]); I2=xx(2)-xx(1)
```

用下面的语句还可以绘制出积分函数的曲线,其中由 B 样条函数可以得出和解析解十分接近的积分函数,如图 8-21 所示。

```
>> ezplot('-cos(t)+2',[0,pi]); hold on; fnplt(a,'--'); fnplt(b,':')
```

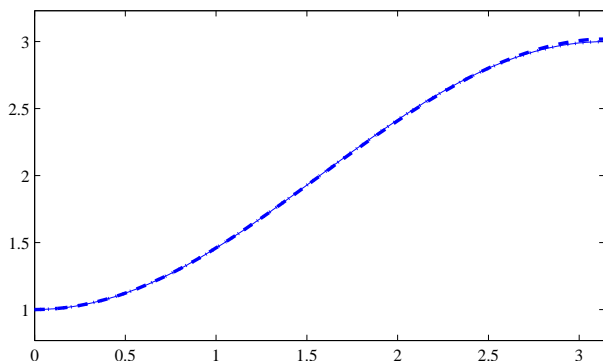


图 8-21 基于样条插值的函数数值积分结果

8.3 由已知数据拟合数学模型

前面介绍的插值方法主要用于求取未知点的函数值,并不能得出原函数的解析表达式,在实际应用中有时需要函数的数学表达式,所以本节侧重于由样本数据获得函数表达式的方法。本节首先介绍由样本数据获得多项式近似的方法,然后介绍多元函数的线性回归建模方法、一般非线性函数的最小二乘曲线拟合方法等。

8.3.1 多项式拟合

前面介绍的 Lagrange 插值就是一种多项式拟合。一般多项式拟合的目标是找出一组多项式系数 $a_i, i = 1, 2, \dots, n+1$, 使得多项式

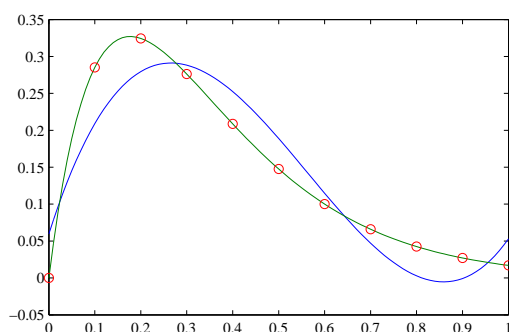
$$\psi(x) = a_1x^n + a_2x^{n-1} + \dots + a_nx + a_{n+1} \quad (8-3-1)$$

能够较好地拟合原始数据。和前面介绍的插值算法不同, 多项式拟合并不能保证每个样本点都在拟合的曲线上, 但能使得整体的拟合误差较小。多项式拟合可以通过 MATLAB 提供的 `polyfit()` 函数实现。该函数的调用格式为 `p = polyfit(x, y, n)`, 其中, x, y 为原始的样本点构成的向量, n 为选定的多项式阶次, 得出的 p 为多项式系数按降幂排列得出的行向量, 可以用符号运算工具箱中的 `poly2sym()` 函数将其转换成真正的多项式形式, 也可以使用 `polyval()` 函数求取多项式的值。下面将通过例子演示多项式拟合函数的使用方法。

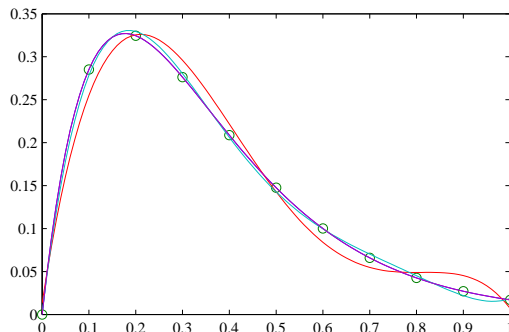
例 8-19 考虑例 8-1 中的样本点数据, 试用多项式拟合的方法在不同的阶次下进行拟合, 并观察拟合效果, 找出合适的阶次。

解 可以用下面的语句得出拟合该数据的 3 次多项式并绘制出拟合曲线, 如图 8-22 (a) 所示, 得出 $p_3(x) = 2.8400x^3 - 4.7898x^2 + 1.9432x + 0.05975$ 。

```
>> x0=0:.1:1; y0=(x0.^2-3*x0+5).*exp(-5*x0).*sin(x0); p3=polyfit(x0,y0,3)
x=0:.01:1; ya=(x.^2-3*x+5).*exp(-5*x).*sin(x);
y1=polyval(p3,x); plot(x,y1,x,ya,x0,y0,'o')
```



(a) 3 次多项式拟合



(b) 其他次数的多项式拟合

图 8-22 多项式拟合效果

从拟合结果可以看出, 效果还是相当差的, 一种很显然的解决方法就是增加拟合多项式的次数, 下面就不同的次数进行拟合, 最终得出如图 8-22 (b) 所示的拟合效果。

```
>> p4=polyfit(x0,y0,4); y2=polyval(p4,x); p5=polyfit(x0,y0,5);
y3=polyval(p5,x); p8=polyfit(x0,y0,8); y4=polyval(p8,x);
plot(x,ya,x0,y0,'o',x,y2,x,y3,x,y4)
```

从该例的拟合效果看, $n \geq 8$ 就能得出较好的结果, 这时拟合多项式为

$$p(x) = -8.26x^8 + 43.6x^7 - 102x^6 + 140.2x^5 - 125.3x^4 + 74.6x^3 - 27.7x^2 + 4.99x + 0.4 \times 10^{-6}$$

多项式拟合实际上相当于对已知函数用 Taylor 幂级数表示, 但 Taylor 幂级数展开的前提条件

是函数应该已知,这对实际的多项式拟合问题显得很苛刻。对本例来说,因为原函数是已知的,所以可以通过 Taylor 幂级数方法先展开该函数

```
>> syms x; y=(x^2-3*x+5)*exp(-5*x)*sin(x); p1=taylor(y,'Order',9)
```

可以得出多项式逼近的结果为 $p_1(x) = 5x - 28x^2 + 77.667x^3 - 142x^4 + 192.17x^5 - 204.96x^6 + 179.13x^7 - 131.67x^8$ 。比较该结果和上述多项式拟合的结果,可以发现二者是完全不同的,这样就可以得出结论,由多项式拟合的数据模型是不唯一的,即使两个多项式函数完全不同,在某一区域内其曲线将特别近似。所以有时进行多项式拟合时应该注意检验结果,比如得出的结果是否很平滑,而不应片面地比较多项式的系数是否一致。

例 8-20 重新考虑例 8-3 中的函数,试观察多项式拟合的效果。

解 多项式拟合的效果并不一定总是很精确的。考虑例 8-3 中的样本点,可以取不同的多项式阶次 n ,则使用如下语句获得多项式拟合,并绘制出拟合曲线,如图 8-23(a) 所示。

```
>> x0=-1+2*[0:10]/10; y0=1./(1+25*x0.^2); x=-1:.01:1; ya=1./(1+25*x.^2);
p3=polyfit(x0,y0,3); y1=polyval(p3,x); p5=polyfit(x0,y0,5);
y2=polyval(p5,x); p8=polyfit(x0,y0,8); y3=polyval(p8,x);
p10=polyfit(x0,y0,10); y4=polyval(p10,x);
plot(x,ya,x,y1,x,y2,x,y3,'-.',x,y3,'--',x,y4,':')
```

其实,该例子如果用 Taylor 幂级数展开效果将更差,用下面的语句可以得出 Taylor 幂级数展开式及拟合效果,并可以绘制出该多项式拟合的效果,如图 8-23(b) 所示。可以看出,这样拟合的结果是相当差的,甚至说是完全错误的。这时得出的拟合多项式为

$$p(x) = 1 - 25x^2 + 625x^4 - 15625x^6 + 390625x^8$$

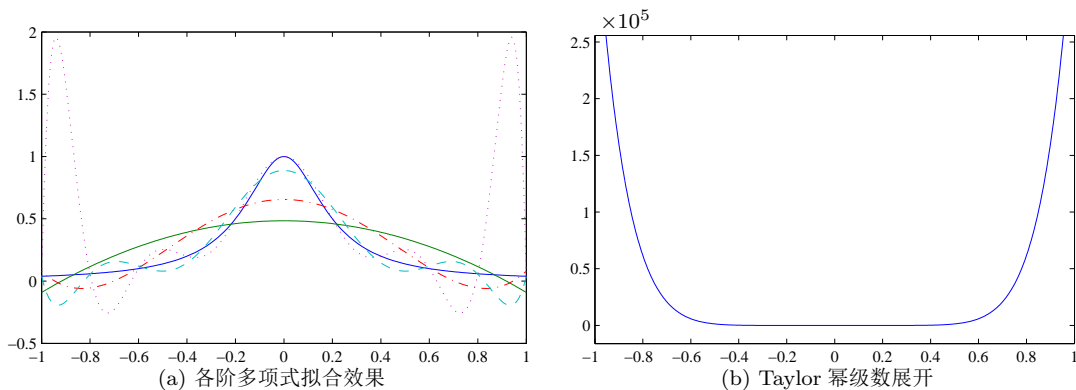


图 8-23 多项式拟合及 Taylor 幂级数展开

```
>> syms x; y=1/(1+25*x^2); p=taylor(y,x,'Order',10), ezplot(p,[-1 1])
```

8.3.2 函数线性组合的曲线拟合方法

假设已知某函数的线性组合为

$$g(x) = c_1 f_1(x) + c_2 f_2(x) + c_3 f_3(x) + \cdots + c_n f_n(x) \quad (8-3-2)$$

其中, $f_1(x), f_2(x), \dots, f_n(x)$ 为已知函数, c_1, c_2, \dots, c_n 为待定系数, 这时假设已经测出数据 $(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)$, 则可以建立起如下的线性方程

$$\mathbf{A}\mathbf{c} = \mathbf{y} \quad (8-3-3)$$

其中

$$\mathbf{A} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_M) & f_2(x_M) & \cdots & f_m(x_M) \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} \quad (8-3-4)$$

且 $\mathbf{c} = [c_1, c_2, \dots, c_n]^T$ 。故该方程的最小二乘解为 $\mathbf{c} = \mathbf{A} \backslash \mathbf{y}$ 。

例 8-21 假设测出了一组 (x_i, y_i) 由表 8-3 给出, 且已知函数原型为 $y(x) = c_1 + c_2 e^{-3x} + c_3 \cos(-2x)e^{-4x} + c_4 x^2$, 试用已知数据求出待定系数 c_i 的值。

表 8-3 实测数据

x_i	0	0.2	0.4	0.7	0.9	0.92	0.99	1.2	1.4	1.48	1.5
y_i	2.88	2.2576	1.9683	1.9258	2.0862	2.109	2.1979	2.5409	2.9627	3.155	3.2052

解 可以将表中数据直接拟合出曲线方程中的 c_i 参数。

```
>> x=[0,0.2,0.4,0.7,0.9,0.92,0.99,1.2,1.4,1.48,1.5]';
y=[2.88;2.2576;1.9683;1.9258;2.0862;2.109;2.198;2.541;2.9627;3.155;3.2052];
A=[ones(size(x)) exp(-3*x), cos(-2*x).*exp(-4*x) x.^2]; c=A\y; c1=c'
```

可以得出拟合参数 $\mathbf{c}^T = [1.22, 2.3397, -0.6797, 0.87]$, 将原 x 向量代入该原型函数

```
>> x0=0:0.01:1.5; B=[ones(size(x0)) exp(-3*x0), cos(-2*x0).*exp(-4*x0) x0.^2];
y1=B*c; plot(x0,y1,x,y,'x')
```

可以得出拟合曲线和已知数据点如图 8-24 所示, 可见拟合效果是令人满意的。

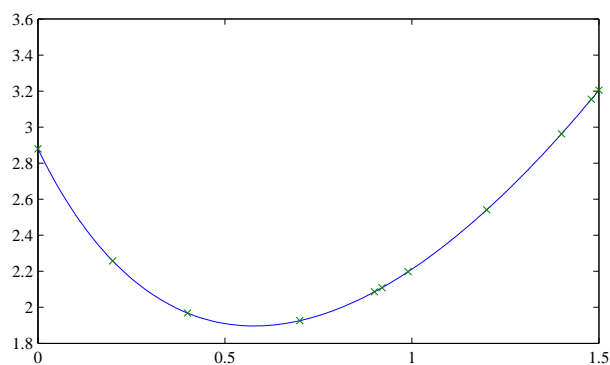


图 8-24 原始数据与拟合曲线

例 8-22 假设测出一组实际数据在表 8-4 中给出, 试对其进行函数拟合。

解 可以用下面的语句将表中给出的数据用曲线表示出来, 如图 8-25(a) 所示。

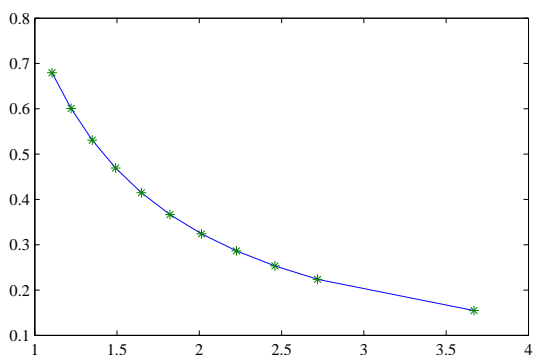
表 8-4 实测数据

x_i	1.1052	1.2214	1.3499	1.4918	1.6487	1.8221	2.0138	2.2255	2.4596	2.7183	3.6693
y_i	0.6795	0.6006	0.5309	0.4693	0.4148	0.3666	0.3241	0.2865	0.2532	0.2238	0.1546

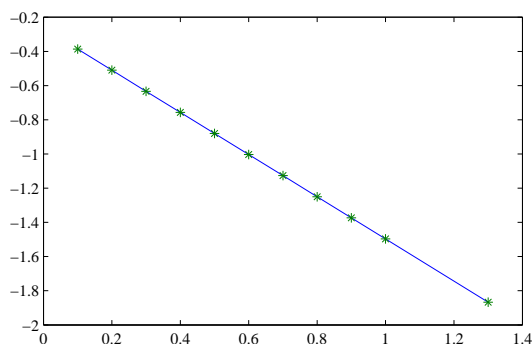
```
>> x=[1.1052,1.2214,1.3499,1.4918,1.6487,1.8221,2.0138,...
      2.2255,2.4596,2.7183,3.6693];
y=[0.6795,0.6006,0.5309,0.4693,0.4148,0.3666,0.3241,...
   0.2864,0.2532,0.2238,0.1546];
plot(x,y,x,y,'*')
```

在实际曲线拟合时,有时从 x, y 本身看不出它们之间的关系,则需要对数据进行可能的非线性变换,观察是否能得出线性关系。例如,可以对 x, y 分别进行对数变换,得出如图 8-25 (b) 所示的曲线,可见二者是线性的。

```
>> x1=log(x); y1=log(y); plot(x1,y1,x1,y1,'*')
```



(a) 曲线拟合



(b) 对数变换后的拟合

图 8-25 数据及拟合结果

这样用线性函数拟合的方法可以得出线性参数,使得 $\ln y = a \ln x + b$, 亦即 $y = e^b x^a$, 而系数 a 、 b 及 e^b 可以由下面的语句直接得出, $c = [-1.2339, -0.2630]^T$, 且 $e^b = 0.7687$, 亦即可以得出拟合函数 $y(x) = 0.7687x^{-1.2339}$ 。

```
>> A=[x1' ones(size(x1'))]; c=[A\y1']', exp(c(2))
```

例 8-23 多项式拟合可以认为是前面介绍的多函数线性组合的特例, 这样可以选择各个函数为 $f_i(x) = x^{n+1-i}, i = 1, 2, \dots, n$, 用该方法重新考虑例 8-19 中数据的多项式拟合问题并观察效果。

解 由上述的算法, 可以立即得出数据的多项式拟合结果, 和例 8-19 给出的结果完全一致。

```
>> x=[0:.1:2]'; y=(x.^2-3*x+5).*exp(-5*x).*sin(x); n=7; A=[];
for i=1:n+1, A(:,i)=x.^(n+1-i); end, c=A\y
```

8.3.3 最小二乘曲线拟合

假设有一组数据 $x_i, y_i, i = 1, 2, \dots, N$, 且已知这组数据满足某一函数原型 $\hat{y}(x) =$

$f(\mathbf{a}, x)$, 其中 \mathbf{a} 为待定系数向量, 则最小二乘曲线拟合的目标就是求出这一组待定系数的值, 可以定义出下面的最优化问题

$$J = \min_{\mathbf{a}} \sum_{i=1}^N [y_i - \hat{y}(x_i)]^2 = \min_{\mathbf{a}} \sum_{i=1}^N [y_i - f(\mathbf{a}, x_i)]^2 \quad (8-3-5)$$

MATLAB 的最优化工具箱中提供了 `lsqcurvefit()` 函数, 可以解决最小二乘曲线拟合的问题。该函数的调用格式为 `[a, Jm] = lsqcurvefit(Fun, a0, x, y, options)`, 其中 `Fun` 为原型函数的 MATLAB 表示, 可以是 M-函数或匿名函数, `a0` 为最优化的初值, `x`、`y` 为原始输入输出数据向量, `options` 则为最优化工具箱通用的控制模板。调用该函数则将返回待定系数向量 \mathbf{a} 以及在此待定系数下的目标函数的值 J_m 。

例 8-24 假设由下面的语句生成一组数据 x 和 y

```
>> x=0:.1:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
```

并已知该数据满足原型为 $y(x) = a_1 e^{-a_2 x} + a_3 e^{-a_4 x} \sin(a_5 x)$, 其中, a_i 为待定系数。采用最小二乘曲线拟合的目的就是获得这些待定系数, 使得目标函数的值为最小。

解 根据已知的函数原型, 可以编写出如下的匿名函数。建立起函数的原型, 则可以由下面的语句得出待定系数向量为 $\mathbf{c} = [0.12, 0.213, 0.54, 0.17, 1.23]$, 拟合残差为 1.7928×10^{-16} 。可以看出, 这样得出的待定系数精度较高。下面语句还可以绘制出拟合曲线与样本点, 如图 8-26 所示, 可见拟合精度很高。

```
>> f=@(a,x)a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x).*sin(a(5)*x);
[xx,res]=lsqcurvefit(f,[1,1,1,1,1],x,y)
x1=0: 0.01: 10; y1=f(xx,x1); plot(x1,y1,x,y,'o')
```

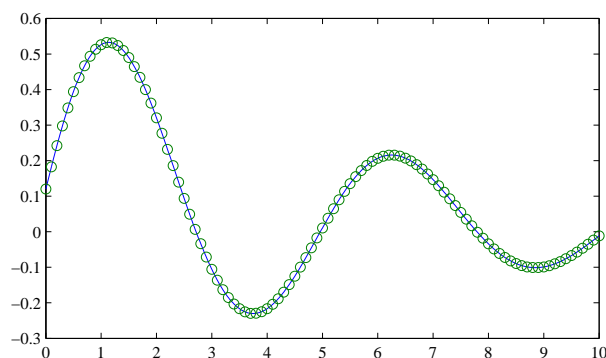


图 8-26 拟合效果比较

例 8-25 假设有一组实测数据由表 8-5 给出, 且已知该数据可能满足的原型函数为 $y(x) = ax + bx^2 e^{-cx} + d$, 试求出满足下面数据的最小二乘解 a 、 b 、 c 、 d 的值。

解 下面的语句可以输入已知的参数

```
>> x=0.1:0.1:1;
y=[2.3201,2.6470,2.9707,3.2885,3.6008,3.9090,4.2147,4.5191,4.8232,5.1275];
```


表 8-5 实测数据

x_i	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y_i	2.3201	2.6470	2.9707	3.2885	3.6008	3.9090	4.2147	4.5191	4.8232	5.1275

令 $a_1 = a, a_2 = b, a_3 = c, a_4 = d$, 这样, 原型函数可以写成 $y(x) = a_1x + a_2x^2e^{-a_3x} + a_4$, 可以用匿名函数描述。下面语句可以得出函数的待定参数 $\mathbf{a} = [3.1001, 1.5027, 4.0046, 2]^T$ 。注意, 本例若不采用循环则可能收敛不到真值。

```
>> f=@(a,x)a(1)*x+a(2)*x.^2.*exp(-a(3)*x)+a(4); a=[1;2;2;3];
while (1), [a,b,c,d]=lsqcurvefit(f,a,x,y); if d>0, break; end, end
```

用下面的语句还可以计算出各个点处的值, 可以将二者曲线绘制在同一坐标系下, 如图 8-27 所示。可见, 二者还是很接近的, 说明拟合效果较好。

```
>> y1=f(a,x); plot(x,y,x,y1,'o')
```

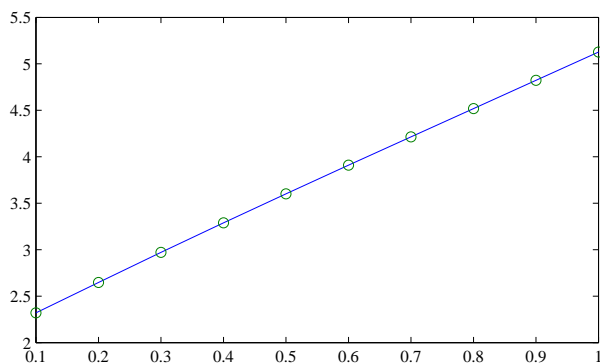


图 8-27 拟合效果比较

8.3.4 多变量函数的最小二乘函数拟合

如果某函数含有若干个自变量, 且已知其原型函数 $z = f(\mathbf{a}, x_1, x_2, \dots, x_m)$, 则仍然可以使用 `lsqcurvefit()` 函数来拟合参数 \mathbf{a} , 其中 $\mathbf{a} = [a_1, a_2, \dots, a_n]$ 。该函数仍需要用户编写一个匿名函数或 M-函数来描述原型函数, 然后调用 `lsqcurvefit()` 函数直接求解待定系数向量 \mathbf{a} , 下面将通过例子演示多变量函数最小二乘拟合的求解方法。

例 8-26 假设某三元函数的原型函数为 $z = a_1x^{a_2x} + a_3y^{a_4(x+y)} + a_5z^{a_6(x+y+z)}$, 且已知一组输入输出数据, 由文本文件 `c8data1.dat` 给出, 该文件的前三列为自变量 x, y, z , 第 4 列为返回向量, 试采用拟合方法得出待定系数 a_i 。

解 解决这类问题第一步仍然需要引入向量型的自变量 \mathbf{x} , 如令 $x_1 = x, x_2 = y, x_3 = z$, 并将给出的原型函数用匿名函数的形式描述出来。因为给出的数据是纯文本文件, 可以通过 `load()` 函数将其读入 MATLAB 工作空间, 用子矩阵提取的方法将输入矩阵 \mathbf{X} 和输出向量 \mathbf{v} 提取出来, 这样就可以用下面语句拟合出待定系数的值 $\mathbf{a} = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]$, 使得拟合误差的最小平方差最小, 其值为 1.0904×10^{-7} 。事实上, 文件中给出的数据是假设 $\mathbf{a} = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]$ 生成的, 所以用这类给出的拟合方法可以很精确地得出待定系数。


```
>> f=@(a,X)a(1)*X(:,1).^(a(2)*X(:,1))+a(3)*X(:,2).^(a(4)*(X(:,1)+X(:,2)))+...
    a(5)*X(:,3).^(a(6)*(X(:,1)+X(:,2)+X(:,3)));
XX=load('c8data1.dat'); X=XX(:,1:3); v=XX(:,4);
a0=[2 3 2 1 2 3]; [a,f,err,key]=lsqcurvefit(f,a0,X,v)
```

8.4 已知函数的有理式逼近方法

8.4.1 给定函数的连分式展开及基于连分式的有理近似

连分式是对函数或数值的一种很有效的近似形式。在数学上 $f(x)$ 经常可以表示为

$$f(x) = b_1 + \frac{(x-a)^{c_1}}{b_2 + \frac{(x-a)^{c_2}}{b_3 + \frac{(x-a)^{c_3}}{b_4 + \frac{(x-a)^{c_4}}{b_5 + \frac{(x-a)^{c_5}}{\dots}}}}} \quad (8-4-1)$$

其中, b_i 是常数, c_i 是有理数, a 是连分式展开的参考点。

MATLAB 语言及符号运算工具箱并未直接提供连分式展开的函数, 但 MuPAD 中提供了一组相关的函数, 我们编写了这些函数的接口函数 `contfrac()`, 由该函数可以直接得出给出函数或数值的连分式展开 (早期版本可以调用 Maple 中给出的 `cfrac()` 函数, 详见本书第 2 版), 该函数的调用格式为

```
cf = contfrac(f,n), [cf,r] = contfrac(f,n,'x=a')
```

其中, f 为原函数或常数的符号表达式, a 为展开的参考点 (默认值为 0, 该项填写为 x), n 是期望展开的级数。返回的 `cf` 是展开式的 MuPAD 表示, r 是展开式的有理式近似。如果 f 为常数, 则调用时只能返回 `cf`。该函数的清单如下

```
function [cf,r]=contfrac(f,varargin)
[n,str]=default_vals(6,'x=0',varargin{:});
if isnumeric(f), cf=feval(symengine,'contfrac',f,n);
else, cf=feval(symengine,'contfrac',f,str,n);
    if nargin==2, r=feval(symengine,'contfrac::rational',cf);
end, end
```

例 8-27 先观察一个常数的连分式近似问题, 试对 π 进行 20 级近似。

解 一个常数的连分式可以用下面的语句直接得出

```
>> cf=contfrac(pi,20), latex(cf)
```

返回变量 `cf` 包括一个系数向量 $c = [3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2]$, 并给出 π 的有理数近似为 $14885392687/4738167652$ (可以精确到 20 位有效数字)。由得出的系数向量, 可以写出 π 的连分式表示为

$$f(x) \approx \frac{1}{4} + \frac{\frac{x}{x}}{-\frac{12}{5} - \frac{\frac{x}{x}}{-\frac{25}{43} + \frac{\frac{x}{x}}{\frac{7396}{1685} - \frac{\frac{x}{x}}{\frac{2839225}{4863128} + \frac{\frac{x}{x}}{-\frac{44767468256}{2592461805} - \frac{\frac{x}{x}}{\frac{789048444603}{663225819208} - \dots}}}}}$$

由下面的语句可以得出前 8 级和 10 级连分式的有理多项式近似

```
>> [cf1,f8]=contfrac(f,8), [cf2,f10]=contfrac(f,10)
```

限于篇幅,这里只列出 $f_8(x)$ 的表达式为

$$f_8(x) \approx \frac{-8457130x^4 + 49735600x^3 - 118414380x^2 + 107698710x}{-5864273x^4 + 83147900x^3 + 294069480x^2 + 312380460x + 107698710}$$

$$f_{10}(x) \approx \frac{-170455846739x^5 + 472453225650x^4 + 3615529382220x^3 - 20275122684600x^2 + 28175852788020x}{2071713977216x^5 + 14187032489655x^4 + 58214153847990x^3 + 110354057230620x^2 + 92428288467480x + 28175852788020}$$

用下面的语句还可以得出 $(0, 2)$ 区间内的原始函数 $f(x)$ 和 $f_8(x)$ 的曲线,如图 8-28 (a) 所示。可见,拟合效果还是很理想的, $n = 10$ 时效果更好些,几乎无法区分原函数曲线和拟合曲线。若扩大拟合区域,令其为 $(0, 5)$,则可以得出如图 8-28 (b) 所示的拟合曲线,可见这样的拟合效果变差,需要进一步增加连分式级数,所以这样的方法有时不适合于大区域拟合。

```
>> ezplot(f,[0,2]), hold on; ezplot(f8,[0,2]); ezplot(f10,[0,2])
```

```
figure; ezplot(f,[0,5]), hold on; ezplot(f8,[0,5]); ezplot(f10,[0,5])
```

另外一种解决方法是选择 $x = 0.5$ 为参考点,这样就可以给出下面的语句

```
>> [cf1,f8]=contfrac(f,8,'x=0.5'), ezplot(f,[0,5]), hold on; ezplot(f8,[0,5]);
```

得出的 8 级连分式展开式为

$$f_1(x) \approx 0.086159 + \frac{x - 0.5}{-9.9242 + \frac{x - 0.5}{0.142877 + \frac{x - 0.5}{1.3 + \frac{x - 0.5}{-0.2942 + \frac{x - 0.5}{22.139 + \frac{x - 0.5}{0.15884 + \frac{x - 0.5}{-33.69 + \dots}}}}}}$$

从而得出其有理近似表达式为

$$f_8(x) \approx -\frac{5.79734386x^4 - 41.483839x^3 + 111.860576x^2 - 110.51792x + 0.004}{116.95748x^3 + 329.5053x^2 + 330.558218x + 110.44266}$$

8.4.2 有理式拟合 — Padé 近似

假设某函数 $f(s)$ 的幂级数展开可以表示为

$$f(s) = c_0 + c_1s + c_2s^2 + c_3s^3 + \dots = \sum_{i=0}^{\infty} c_i s^i \quad (8-4-2)$$

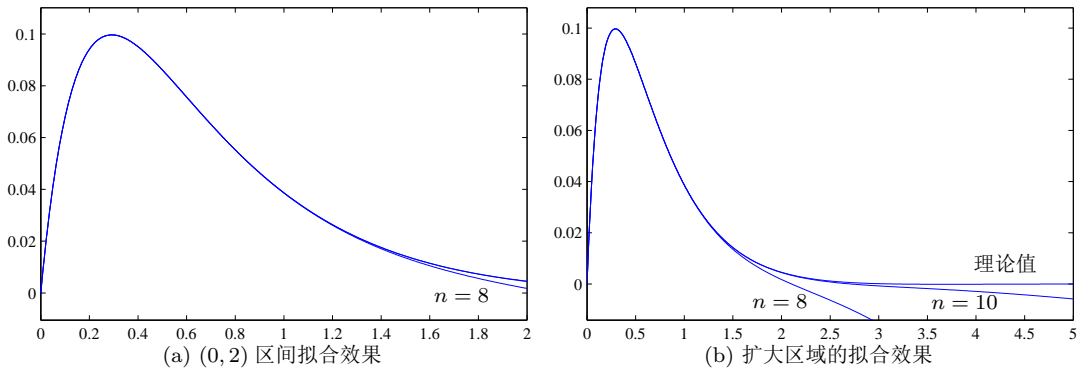


图 8-28 连分式拟合的效果比较

并假设 r/m 阶的 Padé 近似可以写成如下的有理函数形式

$$G_m^r(s) = \frac{\beta_{r+1}s^r + \beta_r s^{r-1} + \cdots + \beta_1}{\alpha_{m+1}s^m + \alpha_m s^{m-1} + \cdots + \alpha_1} = \frac{\sum_{i=1}^{r+1} \beta_i s^{i-1}}{\sum_{i=1}^{m+1} \alpha_i s^{i-1}} \quad (8-4-3)$$

式中, $\alpha_1 = 1$, $\beta_1 = c_1$ 。设 $\sum_{i=0}^{\infty} c_i s^i = G_m^r(s)$, 则可以写出如下的等式

$$\sum_{i=1}^{m+1} \alpha_i s^{i-1} \sum_{i=0}^{\infty} c_i s^i = \sum_{i=1}^{r+1} \beta_i s^{i-1} \quad (8-4-4)$$

对比等式中 s 相应次数的系数, 令相应的 s 项系数的值相等, 则 α_i , $i = 2, \cdots, m+1$ 和 β_i , $i = 2, \cdots, k+1$ 系数可以从下面的方程求解出来。

$$\mathbf{W}\mathbf{x} = \mathbf{w}, \quad \mathbf{v} = \mathbf{V}\mathbf{y} \quad (8-4-5)$$

其中

$$\mathbf{W} = \begin{bmatrix} c_{r+1} & c_r & \cdots & 0 & \cdots & 0 \\ c_{r+2} & c_{r+1} & \cdots & c_1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ c_{r+m} & c_{r+m-1} & \cdots & c_{m-1} & \cdots & c_{r+1} \end{bmatrix} \quad (8-4-6)$$

$$\mathbf{V} = \begin{bmatrix} c_1 & 0 & 0 & \cdots & 0 \\ c_2 & c_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_r & c_{r-1} & c_{r-2} & \cdots & c_1 \end{bmatrix} \quad (8-4-7)$$

且

$$\begin{aligned} \mathbf{x} &= [\alpha_2, \alpha_3, \cdots, \alpha_{m+1}]^T, \quad \mathbf{w} = [-c_{r+2}, -c_{r+3}, \cdots, -c_{m+r+1}]^T \\ \mathbf{v} &= [\beta_2 - c_2, \beta_3 - c_3, \cdots, \beta_{r+1} - c_{r+1}]^T, \quad \mathbf{y} = [\alpha_2, \alpha_3, \cdots, \alpha_{r+1}]^T \end{aligned} \quad (8-4-8)$$

可以证明^[2],若 Padé 近似的分子分母阶次相同或分母比分子高一阶,则该近似等效于 Cauchy II 型连分式近似。可以编写一个 MATLAB 函数 `padefcn()` 来计算给定 $f(x)$ 函数的 Padé 有理函数近似。该函数的清单如下

```
function [nP,dP]=padefcn(c,r,m)
w=-c(r+2:m+r+1)'; vv=[c(r+1:-1:1)'; zeros(m-1-r,1)];
W=rot90(hankel(c(m+r:-1:r+1),vv)); V=rot90(hankel(c(r:-1:1)));
x=[1 (W\w)']; y=[1 x(2:r+1)*V'+c(2:r+1)];
dP=x(m+1:-1:1)/x(m+1); nP=y(r+1:-1:1)/x(m+1);
```

例 8-29 试对 $f(x) = e^{-2x}$ 函数用有理函数近似。

解 可以选择不同的分母阶次,例如选择分子阶次为 0,并选择不同的分母阶次,则可以得出不同的 Padé 有理近似式,近似曲线如图 8-29 所示。

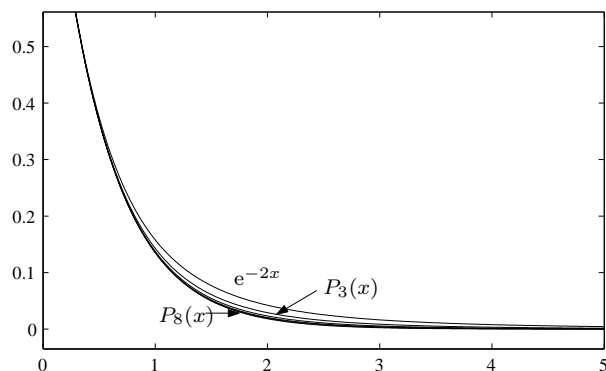


图 8-29 原始数据与拟合曲线

```
>> syms x; c=taylor(exp(-2*x),'Order',10); c=sym2poly(c);
c=c(end:-1:1); x=0:0.01:8; nd=[3:7]; plot(x,exp(-2*x));
for i=1:length(nd)
    [n,d]=padefcn(c,0,nd(i)); y=polyval(n,x)./polyval(d,x); line(x,y)
end
```

由图 8-29 可见,3 阶近似得出的效果尚可,如果增加阶次,会得出更好的效果,8 阶近似的结果还是很精确的。8 阶 Padé 近似表达式如下

$$P_8(s) = \frac{157.5}{x^8 + 4x^7 + 14x^6 + 42x^5 + 105x^4 + 210x^3 + 315x^2 + 315x + 157.5}$$

另外,采用上述算法,可以编写出符号运算版的 `padefcnsym()` 函数

```
function G=padefcnsym(f,r,m)
c=taylor(f,'Order',r+m+1); c=sym2poly(c); c=sym(c(end:-1:1));
w=-c(r+2:m+r+1)'; vv=[c(r+1:-1:1)'; zeros(m-1-r,1)];
W=rot90(hankelsym(c(m+r:-1:r+1),vv)); V=rot90(hankelsym(c(r:-1:1)));
X=[1 (W\w)']; y=[1 X(2:r+1)*V'+c(2:r+1)];
dP=X(m+1:-1:1)/X(m+1); nP=y(r+1:-1:1)/X(m+1);
syms x; G=poly2sym(nP,x)/poly2sym(dP,x);
```

例 8-30 用符号运算方式重新求解例 8-29 中的 Padé 近似模型。

解 给出下面语句,则能得出与前面完全一致的结果。

```
>> syms x; f=exp(-2*x); G=padefcnsym(f,0,8)
```

8.5 特殊函数及曲线绘制

在积分运算中,经常会发现某些函数是不可积的。例如,前面介绍被积函数 e^{-x^2} 不可积时,曾引入了特殊函数 $\text{erf}(\cdot)$ 来表示积分结果。在实际应用中,这类函数很多。常用的有 Γ -函数、 β -函数等。另外在后面将介绍的微分方程求解中,也会将某些不可解析求解的非线性代数方程与微分方程的解表示成特殊函数形式,如 Bessel、Legendre、Mittag-Leffler 函数等,本节将介绍这些常用的特殊函数,并给出这些函数的函数曲线。

8.5.1 Γ -函数

Γ -函数是下面无穷积分的解

$$\Gamma(\alpha) = \int_0^{\infty} e^{-t} t^{\alpha-1} dt \quad (8-5-1)$$

可以由分部积分法验证, $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$, 且 $\Gamma(1) = 1$ 。可见,若 α 为非负整数,则 $\Gamma(\alpha+1) = \alpha!$ 。 Γ -函数是阶乘在非整数 α 上的推广。若 α 为负整数,则 $\Gamma(\alpha+1)$ 趋于 $\pm\infty$ 。利用 MATLAB 语言, Γ -函数可以由 `y = gamma(x)` 直接求出, x 可以为向量,这样 `gamma()` 函数将得出 x 向量每个点上的 Γ -函数值。

可以通过 MATLAB 语言的 `gamma()` 函数直接验证下面的 Γ -函数性质

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}, \Gamma(\alpha)\Gamma(1-\alpha) = \frac{\alpha}{\sin \pi\alpha}, \Gamma(\alpha)\Gamma(-\alpha) = \frac{-\pi}{\alpha \sin \pi\alpha} \quad (8-5-2)$$

$$\Gamma\left(\frac{1}{2} + \alpha\right)\Gamma\left(\frac{1}{2} - \alpha\right) = \frac{\pi}{\cos \pi\alpha}, \lim_{\alpha \rightarrow \infty} \frac{\alpha^z \Gamma(\alpha)}{\Gamma(\alpha+z)} = 1, \text{Re} z > 0 \quad (8-5-3)$$

例 8-31 试绘制 $(-5, 5)$ 区间内 Γ -函数的曲线。

解 下面语句可以直接绘制出 Γ -函数曲线,如图 8-30 所示。因为 $\Gamma(\alpha)$ 在 $\alpha = 0, -1, -2, \dots$ 时趋于无穷大,所以为了使得出的图形含义更清晰,人为地减小了 y -轴显示的范围。

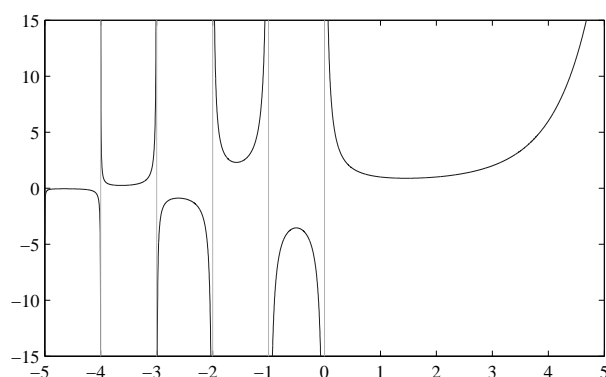
```
>> x=-5:0.002:5; plot(x,gamma(x)), ylim([-15,15])
```

例 8-32 某些积分问题可以利用 Γ -函数直接求解,试写出下面无穷积分的“解析解”。

$$I_1 = \int_0^{\pi/2} \sin^{2m-1} t \cos^{2n-1} t dt, I_2 = \int_0^{\infty} t^{x-1} \cos t dt, x > 0$$

解 下面语句可以直接求出这两个积分分别为 $I_1 = \frac{\Gamma(n)\Gamma(m)}{2\Gamma(n+m)}, I_2 = \frac{2^{x-1}\sqrt{\pi}\Gamma(x/2)}{\Gamma((1-x)/2)}$ 。

```
>> syms m n t z; syms x positive
I1=int(sin(t)^(2*m-1)*cos(t)^(2*n-1),t,0,pi/2)
I2=simple(int(t^(x-1)*cos(t),t,0,inf))
```

图 8-30 Γ -函数曲线

Γ -函数的值可以通过数值积分求解,也可以由下面无穷级数计算出来

$$\Gamma(x) = \frac{1}{x} e^{-\gamma x} \prod_{n=1}^{\infty} \left(\frac{n}{n+x} \right) e^{x/n} \quad (8-5-4)$$

其中 $\gamma \approx 0.57721566490153286$ 为 Euler γ 常数。

若 Γ -积分的上限不是无穷大,则可以定义不完整 Γ -函数为

$$\Gamma(\alpha, x) = \frac{1}{\Gamma(\alpha)} \int_0^x e^{-t} t^{\alpha-1} dt, \quad \alpha \geq 0 \quad (8-5-5)$$

MATLAB 语言提供了 `y = gammainc(x, α)` 函数来求取不完整 Γ -函数。

8.5.2 β -函数

由下面的积分可以定义出 β -函数

$$B(m, \alpha) = \int_0^1 t^{m-1} (1-t)^{\alpha-1} dt = \frac{\Gamma(m)\Gamma(\alpha)}{\Gamma(m+\alpha)}, \quad m, \alpha > 0 \quad (8-5-6)$$

由上面的定义和性质可见,例 8-32 中的 I_1 可以进一步简化为 $B(m, n)/2$ 。 β -函数可以通过 MATLAB 函数 `y = beta(m, x)` 直接计算。

例 8-33 试绘制出各种 m 值下的 β -函数曲线表示。

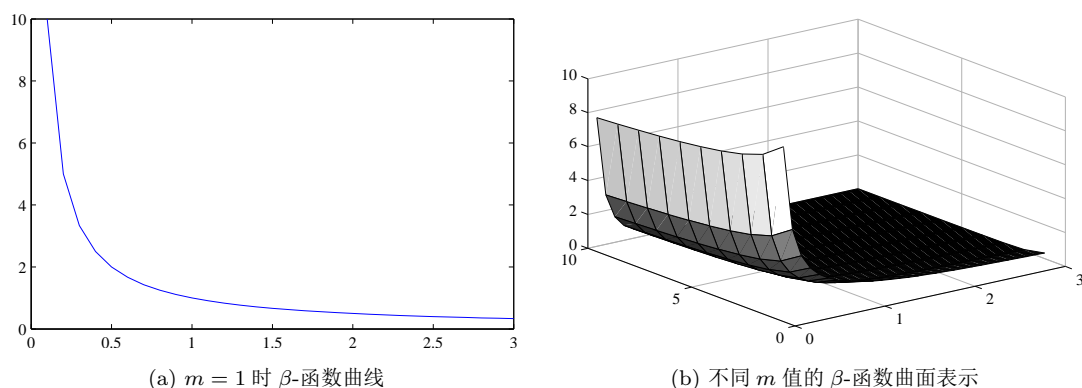
解 若 $m = 1$, 可以直接绘制出 β -函数曲线,如图 8-31(a) 所示。若取不同的 m 值,还可以得出 β -函数的曲面表示,经适当的视角变换可以得出如图 8-31(b) 所示的显示效果。

```
>> m=1; x=0.1:0.1:3; y=beta(m,x); plot(x,y); figure;
    m=1:10; Z=[]; for i=m, Z=[Z; beta(i,x)]; end; surf(x,m,Z)
```

8.5.3 Bessel 函数

考虑下面的 Bessel 微分方程

$$t^2 \frac{d^2 x}{dt^2} + t \frac{dx}{dt} + (t^2 - \lambda^2)x = 0 \quad (8-5-7)$$

图 8-31 β -函数曲线

若 λ 非整数,则采用幂级数求解方法,可以将该方程的通解写成

$$x(t) = C_1 J_\lambda(t) + C_2 J_{-\lambda}(t) \quad (8-5-8)$$

其中, C_1 和 C_2 为任意常数, $J_\lambda(t)$ 为第一类 λ 阶 Bessel 函数

$$J_\lambda(t) = \sum_{m=0}^{\infty} (-1)^m \frac{t^{\lambda+2m}}{2^{\lambda+2m} m! \Gamma(\lambda + m + 1)} \quad (8-5-9)$$

该定义也适用于 λ 为非负整数的情形。若 $\lambda = n$ 为正整数,第一类 Bessel 函数有如下性质

$$J_n(t) = (-1)^n J_{-n}(t), \quad \frac{J_n(x)}{dt} = \frac{n}{t} J_n(t) - J_{n+1}(t), \quad \int t^n J_{n-1}(t) dt = t^n J_n(t) \quad (8-5-10)$$

若 $\lambda = n$ 为整数,则 $J_n(t)$ 与 $J_{-n}(t)$ 线性相关,故方程的解不能用式 (8-5-8) 来表示,需要引入第二类 n 阶 Bessel 函数 (或称 Neumann 函数)

$$N_\lambda(t) = \frac{J_\lambda(t) \cos \lambda t - J_{-\lambda}(t)}{\sin \lambda t} \quad (8-5-11)$$

这时,取 $\lambda = n$,则式 (8-5-7) 的解可以改写成

$$x(t) = C_1 J_n(t) + C_2 H_n(t) \quad (8-5-12)$$

MATLAB 中提供的 `besselj()` 函数可以直接求取第一类 Bessel 函数的值,该函数的调用格式为 `y = besselj(λ , x)`,其中 λ 为阶次。第二类 Bessel 函数可以由 `bessely()` 函数直接求解,其格式与 `besselj()` 完全一致。MATLAB 还提供了第三类 Bessel 函数 (又称 Hankel 函数) 的计算函数 `besselh()`。

例 8-34 试用图形方式表示第一类 Bessel 函数曲线。

解 可以由下面语句绘制出各种 Bessel 函数曲线,如图 8-32 (a)、(b) 所示。注意,为更好地显示结果,这里给出的图形中曲线线形经过后处理。其他阶次的 Bessel 函数曲线建议仿照上面命令单独绘制,以增加其可读性。

```
>> lam=0; x=-10:0.1:10; y=besselj(lam,x); plot(x,y); figure;
    lam=-2:2; for i=lam, plot(x,besselj(i,x)); hold on; end
```

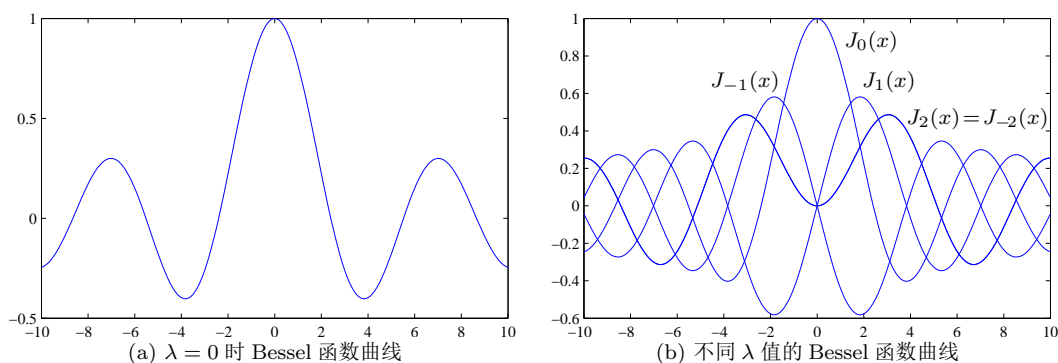



图 8-32 第一类 Bessel 曲线

8.5.4 Legendre 函数

考虑下面的 Legendre 微分方程

$$(1-t^2)\frac{d^2x}{dt^2} - 2t\frac{dx}{dt} + n(n+1)x = 0 \quad (8-5-13)$$

该方程没有解析解,所以可以通过引入 Legendre 函数将其解析解的通式写成

$$x(t) = C_1 P_n(t) + C_2 Q_n(t) \quad (8-5-14)$$

其中 C_1, C_2 为任意常数, $P_n(t)$ 和 $Q_n(t)$ 分别为如下定义的第一、二类 Legendre 函数

$$P_n(t) = \sum_{k=0}^{\infty} (-1)^k \frac{\Gamma(k+n+1)}{(k!)^2 \Gamma(n-k+1)} \left(\frac{1-t}{2}\right)^k, \quad |1-t| < 2 \quad (8-5-15)$$

$$Q_n(t) = \frac{1}{2} P_n(t) \ln \left(\frac{t+1}{t-1}\right) - \sum_{k=1}^n \frac{1}{k} P_{k-1}(t) P_{n-k}(t) \quad (8-5-16)$$

还可以将式(8-5-13)拓展,构造出一系列关联 Legendre 微分方程

$$(1-t^2)\frac{d^2x}{dt^2} - 2t\frac{dx}{dt} + \left[n(n+1) - \frac{m^2}{1-t^2}\right]x = 0 \quad (8-5-17)$$

这时,关联 Legendre 函数可以记作 $P_n^m(t)$,满足

$$P_n^m(t) = (-1)^m (1-t^2)^{m/2} \frac{d^m}{dt^m} P_n(t) \quad (8-5-18)$$

MATLAB 函数 `legendre()` 可以用来计算关联 Legendre 函数 $P_n^m(t)$,其调用格式为 `Y = legendre(n,x)`,这时 Y 为一个矩阵,其各行分别为 $P_n^0(x), P_n^1(x), \dots, P_n^n(x)$,该函数要求 $-1 < x < 1$ 。

例 8-35 Legendre 函数曲线可以由下面语句直接绘制出来,如图 8-33 所示。其他阶次的 Legendre 函数也可以仿照这里的命令直接绘制。

```
>> x=-1:0.04:1; Y=legendre(2,x); plot(x,Y)
```

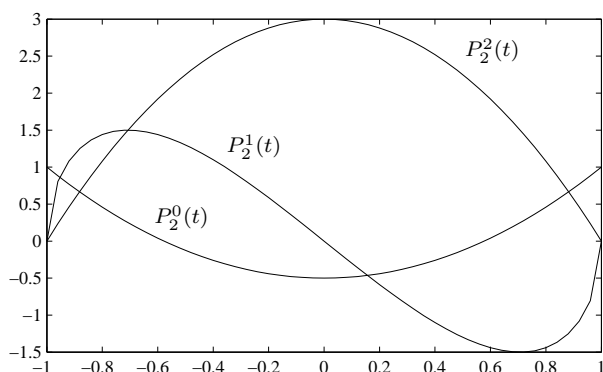


图 8-33 Legendre 函数曲线

8.5.5 Mittag-Leffler 函数

Mittag-Leffler 函数是简单指数函数的拓展。Mittag-Leffler 函数的最简单形式是由瑞典数学家 Magnus Gustaf Mittag-Leffler 于 1903 年提出的^[3], 又称其为单参数 Mittag-Leffler 函数, 后来又出现了双参数和多参数 Mittag-Leffler 函数。Mittag-Leffler 函数在分数阶微积分学中的作用和指数函数在整数阶微积分学中的作用相仿。这里将介绍各种 Mittag-Leffler 函数及计算。

最简单的 Mittag-Leffler 函数为单参数 Mittag-Leffler 函数, 其定义为

$$\mathcal{E}_{\alpha}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + 1)} \quad (8-5-19)$$

其中 α 为复数, 该无穷级数收敛的条件为 $\Re(\alpha) > 0$ 。

显然, 指数函数 e^z 是 Mittag-Leffler 函数的一个特例, 当 $\alpha = 1$ 时有

$$\mathcal{E}_1(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k+1)} = \sum_{k=0}^{\infty} \frac{z^k}{k!} = e^z \quad (8-5-20)$$

另外还可以推导出 $\alpha = 2, \alpha = 1/2$ 时

$$\mathcal{E}_2(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(2k+1)} = \sum_{k=0}^{\infty} \frac{(\sqrt{z})^{2k}}{(2k)!} = \cosh \sqrt{z} \quad (8-5-21)$$

$$\mathcal{E}_{1/2}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k/2+1)} = e^{z^2} (1 + \operatorname{erf}(z)) = e^{z^2} \operatorname{erfc}(-z) \quad (8-5-22)$$

考虑前面给出的单参数 Mittag-Leffler 函数。在分母 Γ -函数中将 1 替换成另一个自由变量 β , 则可以定义出如下的双参数 Mittag-Leffler 函数

$$\mathcal{E}_{\alpha, \beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + \beta)} \quad (8-5-23)$$

其中 α, β 为复数, 且使得无穷级数对任意复数 z 收敛的前提条件是 $\Re(\alpha) > 0, \Re(\beta) > 0$ 。若 $\beta = 1$, 则双参数 Mittag-Leffler 函数退化成单参数函数, 即

$$\mathcal{E}_{\alpha,1}(z) = \mathcal{E}_{\alpha}(z) \quad (8-5-24)$$

所以可以认为单参数函数是双参数函数的一个特例。

由 Mittag-Leffler 函数的定义还可以导出其他的特例, 如

$$\mathcal{E}_{1,2}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k+2)} = \frac{1}{z} \sum_{k=0}^{\infty} \frac{z^{k+1}}{(k+1)!} = \frac{e^z - 1}{z} \quad (8-5-25)$$

$$\mathcal{E}_{1,3}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k+3)} = \sum_{k=0}^{\infty} \frac{z^k}{(k+2)!} = \frac{1}{z^2} \sum_{k=0}^{\infty} \frac{z^{k+2}}{(k+2)!} = \frac{e^z - 1 - z}{z^2} \quad (8-5-26)$$

更一般地^[4]

$$\mathcal{E}_{1,m}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k+m)} = \frac{1}{z^{m-1}} \sum_{k=0}^{\infty} \frac{z^{k+m-1}}{(k+m-1)!} = \frac{1}{z^{m-1}} \left(e^z - \sum_{k=0}^{m-2} \frac{z^k}{k!} \right) \quad (8-5-27)$$

此外还可以推导出

$$\mathcal{E}_{2,2}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(2k+2)} = \frac{1}{\sqrt{z}} \sum_{k=0}^{\infty} \frac{(\sqrt{z})^{2k+1}}{(2k+1)!} = \frac{\sinh \sqrt{z}}{\sqrt{z}} \quad (8-5-28)$$

$$\mathcal{E}_{2,1}(z^2) = \sum_{k=0}^{\infty} \frac{z^{2k}}{\Gamma(2k+1)} = \sum_{k=0}^{\infty} \frac{z^{2k}}{(2k)!} = \cosh z \quad (8-5-29)$$

$$\mathcal{E}_{2,2}(z^2) = \sum_{k=0}^{\infty} \frac{z^{2k}}{\Gamma(2k+2)} = \frac{1}{z} \sum_{k=0}^{\infty} \frac{z^{2k+1}}{(2k+1)!} = \frac{\sinh z}{z} \quad (8-5-30)$$

给定函数 f 的单参数和双参数的 Mittag-Leffler 函数可以由下面的 MATLAB 函数直接求解, 相应的调用格式分别为

`F1 = mittag_leffler(α, z)` 或 `F1 = mittag_leffler([α, β], z)`

```
function f=mittag_leffler(aa,z)
aa=[aa 1]; a=aa(1); b=aa(2);
syms k; f=simple(symsum(z^k/gamma(a*k+b),k,0,inf));
```

注意, 在 MATLAB R2008a 及以前版本可以正常使用上述函数, 新版本中由于级数求和函数本身的限制, 该函数有时无法正确得出所需的结果, 建议在 MATLAB 早期版本下运行该函数。

例 8-36 试求出单参数 α 下的 Mittag-Leffler 函数的解析表达式。例如, 取 $\alpha = 1/3, 3, 4, 5, \dots$ 。

解 通过直接调用前面给出的 `mittag_leffler()` 函数的方法即可以由定义解析地求出所需的 Mittag-Leffler 函数

```
>> syms z; I1=mittag_leffler(1/sym(3),z), I2=mittag_leffler(3,z)
      I3=mittag_leffler(4,z), I4=mittag_leffler(5,z)
```

由上述命令可以直接得出如下的结果

$$\begin{aligned}\mathcal{E}_{1/3}(z) &= -\frac{e^{z^3}(-6\pi\Gamma(2/3) + \sqrt{3}\Gamma^2(2/3)\Gamma(1/3, z^3) + 2\Gamma(2/3, z^3)\pi)}{2\pi\Gamma(2/3)} \\ \mathcal{E}_3(z) &= \frac{1}{3}e^{\sqrt[3]{z}} + \frac{2}{3}e^{-\sqrt[3]{z}/2}\cos\left(\frac{\sqrt{3}}{2}\sqrt[3]{z}\right) \\ \mathcal{E}_4(z) &= \frac{1}{4}e^{\sqrt[4]{z}} + \frac{1}{4}e^{-\sqrt[4]{z}} + \frac{1}{2}\cos(\sqrt[4]{z}) \\ \mathcal{E}_5(z) &= \frac{1}{5}e^{\sqrt[5]{z}} + \frac{2}{5}e^{\cos(2\pi/5)\sqrt[5]{z}}\cos\left(\sin\left(\frac{2}{5}\pi\right)\sqrt[5]{z}\right) + \frac{2}{5}e^{-\cos(1\pi/5)\sqrt[5]{z}}\cos\left(\sin\left(\frac{1}{5}\pi\right)\sqrt[5]{z}\right)\end{aligned}$$

例 8-37 试求出双参数的 Mittag-Leffler 函数, 如 $\mathcal{E}_{4,1}(z)$ 、 $\mathcal{E}_{4,5}(z)$ 、 $\mathcal{E}_{5,6}(z)$ 、 $\mathcal{E}_{1/2,4}(z)$ 。

解 代入合适的 α 、 β 参数, 则可以给出下面语句直接求解

```
>> syms z, I5=mittag_leffler(4,1,z), I6=mittag_leffler(4,5,z)
      I7=mittag_leffler(5,6,z), I8=mittag_leffler(1/sym(2),4,z)
```

由上述命令可以得出

$$\begin{aligned}\mathcal{E}_{4,1}(z) &= \frac{1}{4}e^{\sqrt[4]{z}} + \frac{1}{4}e^{-\sqrt[4]{z}} + \frac{1}{2}\cos\sqrt[4]{z}, \quad \mathcal{E}_{4,5}(z) = -\frac{1}{4} + \frac{1}{4z}\left(e^{\sqrt[4]{z}} + e^{-\sqrt[4]{z}} + e^{j\sqrt[4]{z}} + e^{-j\sqrt[4]{z}}\right) \\ \mathcal{E}_{5,6}(z) &= -\frac{1}{z} + \frac{e^{\sqrt[5]{z}}}{5z}\left[1 + e^{(-1)^{2/5}} + e^{(-1)^{4/5}} + e^{(-1)^{1/5}} + e^{(-1)^{3/5}}\right] \\ \mathcal{E}_{1/2,4}(z) &= \frac{e^{z^2}}{z^6} - \frac{1}{z^6} - \frac{1}{z^4} - \frac{1}{2z^2} + \frac{ze^{z^2}\text{erf}(z)}{z^7} - \frac{8}{15\sqrt{\pi}z} - \frac{4}{3\sqrt{\pi}z^3} - \frac{2}{\sqrt{\pi}z^5}\end{aligned}$$

双参数 Mittag-Leffler 函数 $\mathcal{E}_{\alpha,\beta}(z)$ 的整数阶导数为

$$\frac{d^n}{dz^n}\mathcal{E}_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{(k+n)!}{k!\Gamma(\alpha k + \alpha n + \beta)} z^k \quad (8-5-31)$$

鉴于前面给出的公式可以编写出如下的 MATLAB 函数来求 Mittag-Leffler 函数及其整数阶导数的数值解。该函数采用了累加的方法, 该方法有时不收敛, 所以可以自动调用嵌入的 MLF() 函数来求解。该函数由斯洛伐克学者 Igor Podlubny 教授编写^[5], 数值稳定性高, 但求解速度很慢。本书给出的方法首先尝试累加的快速算法, 如果不收敛再调用 MLF() 函数, 既保证了函数的快速性, 又可以提高函数数值稳定性, 扩大适用范围。

```
function f=ml_func(aa,z,varargin)
aa=[aa,1,1,1]; a=aa(1); b=aa(2); c=aa(3); q=aa(4); f=0; k=0; fa=1;
[n,eps0]=default_vals({0,eps},varargin{:});
if n==0
while norm(fa,1)>=eps0
fa=gamma(k*q+c)/gamma(c)/gamma(k+1)/gamma(a*k+b) *z.^k; f=f+fa; k=k+1;
end
if any(~isfinite(f))
if c*q==1, f=MLF(a,b,z,round(-log10(eps0))); f=reshape(f,size(z));
else, error('Error: truncation method failed'); end, end
```

```

else, aa(2)=aa(2)+n*aa(1); aa(3)=aa(3)+aa(4)*n;
    f=gamma(q*n+c)/gamma(c)*ml_func(aa,z,0,eps0);
end

```

该函数的调用格式为

```

f=ml_func( $\alpha$ ,z,n, $\epsilon_0$ )      % 单参数函数  $\mathcal{E}_\alpha(z)$  的  $n$  阶导数
f=ml_func([ $\alpha$ , $\beta$ ],z,n, $\epsilon_0$ ) % 双参数函数  $\mathcal{E}_{\alpha,\beta}(z)$  的  $n$  阶导数

```

其中, ϵ_0 的默认值为 `eps`, n 的默认值为 0, 表示原函数。事实上该函数可以直接用于 3、4 参数的 Mittag-Leffler 函数的求解。

例 8-38 考虑例 8-37 给出的函数 $\mathcal{E}_{1/2,4}(z)$, 比较数值方法和解析方法得出的结果。

解 下面的语句可以求出该函数的解析解和数值解, 如图 8-34 所示。可见, 解析解法和数值解法得出的结果完全一致。对本例来说, 如果时间区间过大就会导致累加的数值算法不收敛, `ml_func()` 函数会自动调用嵌入的 `MLF()` 数值求解, 求解速度将明显减慢。

```

>> syms z, I8=mittag_leffler(1/sym(2),4,z);
    t=0:0.01:2; y=subs(I8,z,t); y1=ml_func([1/2,4],t); plot(t,y,t,y1)

```

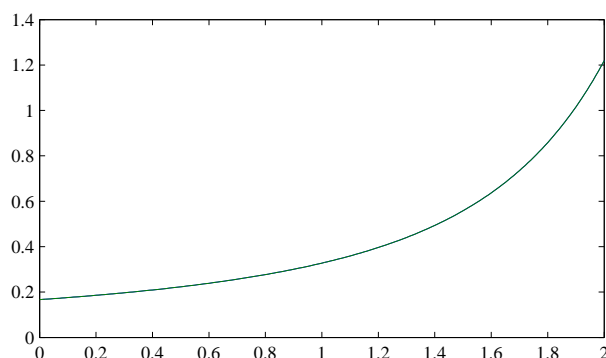


图 8-34 Mittag-Leffler 函数曲线

8.6 信号分析与数字信号处理基础

8.6.1 信号的相关分析

假设已知某确定性信号为 $x(t)$, 则其自相关函数的定义为

$$R_{xx}(\tau) = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T x(t)x(t+\tau)dt, \quad \tau \geq 0 \quad (8-6-1)$$

且相关函数为偶函数, 即 $R_{xx}(-\tau) = R_{xx}(\tau)$ 。若研究两个信号 $x(t)$ 和 $y(t)$, 则可以定义其互相关函数为

$$R_{xy}(\tau) = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T x(t)y(t+\tau)dt, \quad \tau \geq 0 \quad (8-6-2)$$

从前面介绍的微积分求解方法可知, 若这些函数都已知, 则可以通过 MATLAB 的符号运算工具箱直接求出自相关函数和互相关函数的解析表达式。

例 8-39 已知函数 $x(t) = A_1 \cos(\omega_1 t + \theta_1) + A_2 \cos(\omega_2 t + \theta_2)$, 试根据定义求出其自相关函数。

解 根据定义, 首先应该申明一些有关符号变量, 再定义出已知函数 $x(t)$, 这样, 就可以由下面的语句求出信号的自相关函数, 注意, MATLAB R2008a 及以前版本可以得出正确结果

```
>> syms A1 A2 w1 w2 t1 t2 t tau T; x=A1*cos(w1*t+t1)+A2*cos(w2*t+t2);
    Rxx=simple(limit(int(x*subs(x,t,t+tau),t,-T,T)/2/T,T,inf))
```

这样可以求出信号的自相关函数解析解为

$$R_{xx}(\tau) = \frac{1}{2}A_1^2 \cos(\omega_1 \tau) + \frac{1}{2}A_2^2 \cos(\omega_2 \tau)$$

假设在实验中测出两组数据, $x_i, y_i, (i = 1, 2, \dots, n)$, 则可以由下面的式子计算出两组数据的相关系数矩阵为

$$R_{xy} = \frac{\sqrt{\sum (x_i - \bar{x})(y_i - \bar{y})}}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}} \quad (8-6-3)$$

MATLAB 提供了 `corrcoef()` 函数, 可以求出已知 x, y 向量的相关系数矩阵 R 。该函数的调用格式为 $R = \text{corrcoef}(x, y)$ 或 $R = \text{corrcoef}([x, y])$ 。

例 8-40 试用原型函数 $y_1 = te^{-4t} \sin 3t$ 和 $y_2 = te^{-4t} \cos 3t$ 分别生成一组数据, 并由得出的数据求取其相关系数矩阵。

解 用下面的语句可以立即得出两个信号的相关系数矩阵为 $R = \begin{bmatrix} 1 & 0.4776 \\ 0.4776 & 1 \end{bmatrix}$ 。

```
>> x=0:0.01:5; y1=x.*exp(-4*x).*sin(3*x); y2=x.*exp(-4*x).*cos(3*x);
    R=corrcoef(y1,y2)
```

仍假设在实验中测出两组数据, $x_i, y_i, (i = 1, 2, \dots, n)$, 对这些离散点可以由下面的式子定义 x_i 序列的自相关函数

$$c_{xx}(k) = \frac{1}{N} \sum_{l=1}^{n-[k]-1} x(l)x(k+l), \quad 0 \leq k \leq m-1 \quad (8-6-4)$$

其中 $m < n$, 自相关函数是偶函数。类似地, 还可以定义出互相关函数

$$c_{xy}(k) = \frac{1}{N} \sum_{l=1}^{n-[k]-1} x(l)y(k+l), \quad 0 \leq k \leq m-1 \quad (8-6-5)$$

相关函数可以用来研究两个序列信号的相似性。MATLAB 提供了求取和绘制自相关函数和互相关函数的程序 `xcorr()`, 其调用格式为 $c_{xx} = \text{xcorr}(x, N)$ 、 $c_{xy} = \text{xcorr}(x, y, N)$, 其中, N 为 k 的最大取值, 可以忽略, 如果该函数不返回任何变量则将自动绘制自相关函数或互相关函数曲线。

例 8-41 仍假设已知由例 8-40 中函数计算出的数据, 试用数值方法求取自相关函数、互相关函数, 并和已知理论曲线进行比较。

解 先在 $t \in (0, 5)$ 区间生成一个时间向量, 则可以由原型函数直接计算出 x 向量和 y 向量, 调用现成的函数就可以得出它们的自相关函数和互相关函数, 如图 8-35(a)、(b) 所示。

```
>> t=0:0.01:5; x=t.*exp(-4*t).*sin(3*t); y=t.*exp(-4*t).*cos(3*t);
N=150; c1=xcorr(x,N); x1=[-N:N]; stem(x1,c1) % 实际绘图点选得更稀疏
figure; c1=xcorr(x,y,N); x1=[-N:N]; stem(x1,c1)
```

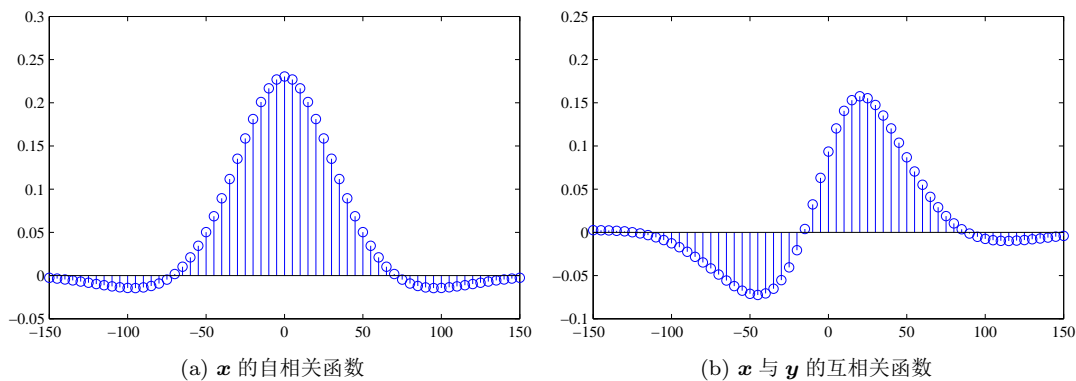


图 8-35 信号的相关函数分析

8.6.2 滤波技术与滤波器设计

例 8-42 在介绍滤波技术之前,考虑由曲线 $y(x) = e^{-x} \sin 5x$ 叠加标准差为 $\sigma = 0.05$ 的零均值的白噪声信号,绘制出噪声污染后的信号曲线。

解 下面语句可以得出如图 8-36 所示的曲线

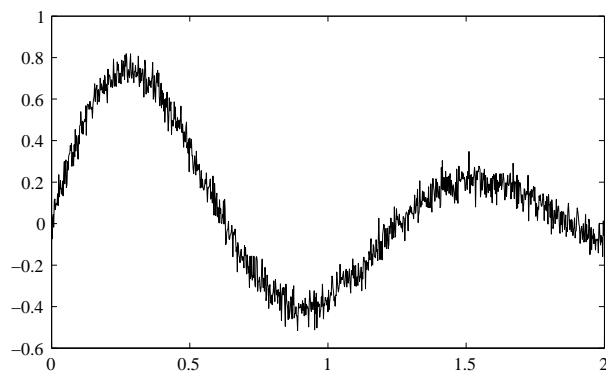


图 8-36 噪声污染的数据曲线

```
>> x=0:.002:2; y=exp(-x).*sin(5*x); r=0.05*randn(size(x)); y1=y+r; plot(x,y1)
```

对于图 8-36 中给出的被噪声污染的信号曲线,需要引入一种办法消除噪声。例如,可以引入滤波器来实现降低噪声的工作,得出平滑的曲线。

1. 线性滤波器的一般模型

线性滤波器模型的一般表达式为

$$H(z) = \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + \cdots + b_{n+1} z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_m z^{-m}} \quad (8-6-6)$$

假设输入信号为 $x(n)$, 则经过该滤波器后的输出信号可以由下面的差分方程表示为

$$y(k) = -a_1y(k-1) - \cdots - a_my(k-m) + b_1x(k) + b_2x(k-1) + \cdots + b_{n+1}x(k-n) \quad (8-6-7)$$

根据 n 和 m 的不同取值, 可以定义出 3 种常用的滤波器如下:

(1) **FIR 滤波器**。又称为有限长脉冲响应 (finite impulse response, FIR) 滤波器, 需要将式 (8-6-6) 中的 m 值设置成 $m = 0$, 这时 \mathbf{a} 为标量, 在控制领域也称移动平均模型 (moving average, MA), 这时用向量 \mathbf{b} 就可以表示该滤波器。

(2) **IIR 滤波器**。又称为全极点无限长脉冲响应 (infinite impulse response, IIR) 滤波器, 也称为自回归 (auto-regressive, AR) 模型, 这时 $n = 0$, 即 \mathbf{b} 为标量, 这样用 \mathbf{a} 即可以表示该滤波器。FIR 和全极点 IIR 滤波器相比, 一般达到同样要求所需的滤波器阶数较高, 但其优势是总可以设计出稳定的滤波器 [6]。

(3) **ARMA 滤波器**。又称为一般 IIR 滤波器和自回归移动平均 (auto-regressive moving average, ARMA) 模型, 要求 n 与 m 均不为 0, 可以用 \mathbf{a}, \mathbf{b} 两个向量表示该滤波器。

假设滤波器可以由 \mathbf{a}, \mathbf{b} 两个向量表示, 且假设需要过滤的信号为向量 \mathbf{x} , 则可以调用 `filter()` 函数直接计算出过滤后的信号向量 \mathbf{y} 为 $\mathbf{y} = \text{filter}(\mathbf{b}, \mathbf{a}, \mathbf{x})$ 。

从滤波器的作用又可以分为低通滤波器、高通滤波器和带通滤波器等。顾名思义, 低通滤波器是指那些允许低频信号顺利通过, 而高频信号被过滤的一类滤波器; 高通滤波器是指高频顺利通过, 而低频信号被过滤的滤波器; 带通滤波器是指某一个频段的信号被顺利通过, 而在这个频段带外的信号均被过滤的滤波器。它们在实际应用中各有其应用。

例如, 例 8-42 中给出信号中的噪声为高频的, 所以可以考虑设计一个低通滤波器, 即频率低时放大倍数接近于 1, 高频的信号经过接近于 0 的放大倍数后, 就基本可以被滤除。如果滤波器已知, 则用 `freqz()` 函数可以对滤波器进行放大倍数分析, 即

$$[\mathbf{h}, \mathbf{w}] = \text{freqz}(\mathbf{b}, \mathbf{a}, N)$$

其中, N 为分析的点数, 返回的 \mathbf{h} 为复数放大倍数, \mathbf{w} 为频率向量。复数放大倍数包含幅值与幅角等信息, 若只想获得放大倍数的幅值, 则可以用 `plot(w, abs(h))` 命令。

例 8-43 假设已经设计出

$$H(z) = \frac{1.2296 \times 10^{-6}(1+z^{-1})^7}{(1-0.7265z^{-1})(1-1.488z^{-1}+0.5644z^{-2})(1-1.595z^{-1}+0.6769z^{-2})(1-1.78z^{-1}+0.8713z^{-2})}$$

试得出该滤波器的放大倍数, 并观察滤波后的信号。

解 可以用下面的语句将滤波器的 \mathbf{b}, \mathbf{a} 向量输入到 MATLAB 环境中, 其中, 用 `conv()` 函数可以求取多项式乘积。下面的语句还可以绘制出滤波器的放大倍数曲线, 如图 8-37(a) 所示, 对低频信号的放大倍数接近 1, 不作过滤处理, 而对高频信号的放大倍数接近 0, 可以将噪声信号滤去, 得出所需的滤波信号。

```
>> b=1.2296e-6*conv([1 4 6 4 1],[1 3 3 1]); a=conv([1,-0.7265],...
    conv([1,-1.488,0.5644],conv([1,-1.595,0.6769],[1,-1.78,0.8713])));
x=0:0.002:2; y=exp(-x).*sin(5*x); r=0.05*randn(size(x)); y1=y+r;
[h,w]=freqz(b,a,100); plot(w,abs(h))           % 放大倍数绘制
figure; y2=filter(b,a,y1); plot(x,y1,x,y2)      % 滤波效果
```


经过滤波器后的滤波效果如图 8-37 (b) 所示。可见,该滤波器可以较好地对给定噪声信号进行过滤处理,但得出的滤波信号较原信号稍有延迟。

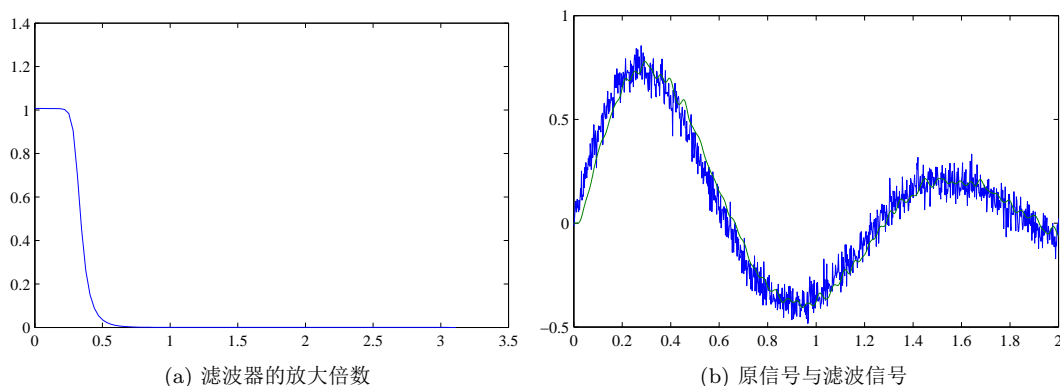


图 8-37 滤波器及滤波效果

2. 滤波器设计及 MATLAB 实现

从前面给出的例子可见,如果想较好地对噪声信号进行过滤,则需要用滤波器。滤波器的设计方法多种多样,在 MATLAB 的信号处理工具箱和滤波器设计工具箱中提供了多种滤波器设计的函数,可以直接调用。常用的有两种滤波器类型,如 Butterworth 滤波器和 Chebyshev 滤波器,可以分别用 `butter()` 函数、`cheby1()` 函数 (Chebyshev I 型滤波器) 及 `cheby2()` 函数 (Chebyshev II 型)。它们的调用格式为

$[b,a] = \text{butter}(n,\omega_n)$, $[b,a] = \text{cheby1}(n,r,\omega_n)$, $[b,a] = \text{cheby2}(n,r,\omega_n)$

其中, n 为滤波器的阶次,可以由用户选择,也可以用 MATLAB 的相应函数 (如 `buttord()` 等函数) 设置。 ω_n 为归一化的频率,定义为实际过滤的频率与信号 Nyquist 频率的比值。假设均匀步距采样的数据个数为 N 个,步距为 Δt ,则可以计算出基波频率 $f_0 = 1/\Delta t$ Hz,这时 Nyquist 频率的定义为 $f_0 N/2$ 。

例 8-44 仍考虑例 8-42 中的给定信号,试对阶次及不同的 ω_n 值,设计 Butterworth 滤波器并比较滤波效果。

解 仍选择 $\omega_n = 0.1$,用下面的语句则可以设计出不同阶次的 Butterworth 滤波器,并可以绘制出这些滤波器的放大倍数及滤波效果曲线,如图 8-38 (a)、(b) 所示。从滤波的结果可见,随着 n 的增加,滤波的效果越来越平滑,但延迟也将增大。

```
>> f1=figure; f2=figure;
    for n=5:2:20
        figure(f1); [b,a]=butter(n,0.1); y2=filter(b,a,y1); plot(x,y2); hold on
        figure(f2); [h,w]=freqz(b,a,100); plot(w,abs(h)); hold on
    end
```

选择阶次为 7,对不同的 ω_n 可以设计出 Butterworth 滤波器,并绘制出放大倍数及滤波效果曲线,如图 8-39 (a)、(b) 所示。从得出的曲线看,当 ω_n 增加时,延迟变小,但得出的滤波效果会明显变差, ω_n 太大时滤波没有什么效果。

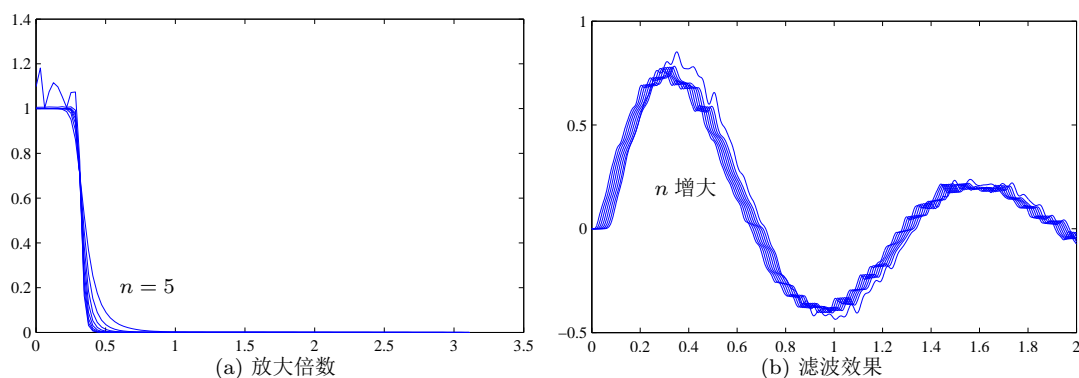


图 8-38 不同阶次下 Butterworth 滤波器放大倍数及滤波效果

```
>> for wn=0.1:0.1:0.7
    figure(f1); [b,a]=butter(7,wn); y2=filter(b,a,y1); plot(x,y2); hold on
    figure(f2); [h,w]=freqz(b,a,100); plot(w,abs(h)); hold on
end
```

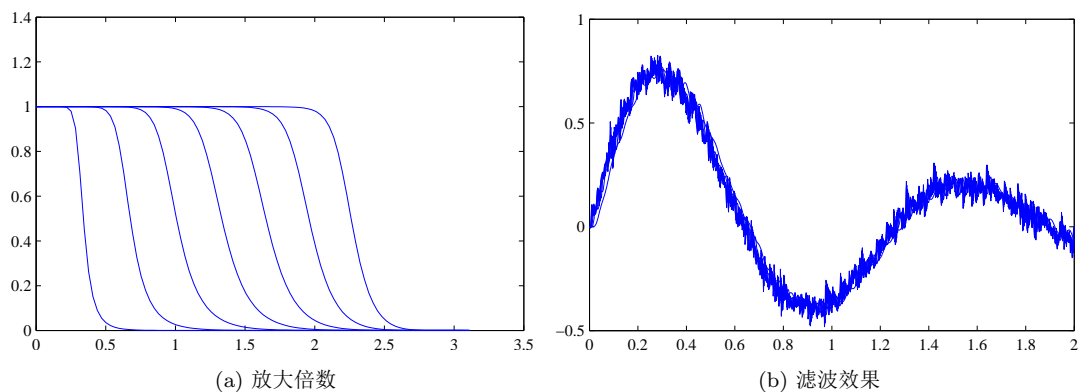


图 8-39 不同滤波频率下 Butterworth 滤波器放大倍数及滤波效果

若想设计高通滤波器,最简单的方法是可以利用 $1 - H(z^{-1})$ 直接设计出来,其中, $H(z^{-1})$ 为前面介绍的方法设计出的低通滤波器。另外, `butter()` 等函数也可以直接设计高通滤波器和带通滤波器。具体调用格式为

高通: `[b,a] = butter(n,wn, 'high')`, 带通: `[b,a] = butter(n,[w1,w2])`

8.7 习 题

1. 用 $y(t) = t^2 e^{-5t} \sin t$ 生成一组较稀疏的数据,并用一维数据插值的方法对给出的数据进行曲线拟合,并将结果与理论曲线相比较。
2. 用 $y(t) = \sin(10t^2 + 3)$ 在 $(0, 3)$ 区间内生成一组较稀疏的数据,用一维数据插值的方法对给出的数据进行曲线拟合,并将结果与理论曲线相比较。
3. 用 $f(x, y) = \frac{1}{3x^3 + y} e^{-x^2 - y^4} \sin(xy^2 + x^2 y)$ 原型函数生成一组网格数据或随机数据,分别拟合

出曲面,并和原曲面进行比较。

4. 假设已知一组数据,试用插值方法绘制出 $x \in (-2, 4.9)$ 区间内的光滑函数曲线,比较各种插值算法的优劣。

x_i	-2	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1	1.3
y_i	0.10289	0.11741	0.13158	0.14483	0.15656	0.16622	0.17332	0.1775	0.17853	0.17635	0.17109	0.16302
x_i	1.6	1.9	2.2	2.5	2.8	3.1	3.4	3.7	4	4.3	4.6	4.9
y_i	0.15255	0.1402	0.12655	0.11219	0.09768	0.08353	0.07015	0.05786	0.04687	0.03729	0.02914	0.02236

5. 假设已知实测数据由下表给出,试对 (x, y) 在 $(0.1, 0.1) \sim (1.1, 1.1)$ 区域内的点进行插值,用三维曲面的方式绘制出插值结果,并搜索出该函数的最小值。

y_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
0.1	0.83041	0.82727	0.82406	0.82098	0.81824	0.8161	0.81481	0.81463	0.81579	0.81853	0.82304
0.2	0.83172	0.83249	0.83584	0.84201	0.85125	0.86376	0.87975	0.89935	0.92263	0.94959	0.9801
0.3	0.83587	0.84345	0.85631	0.87466	0.89867	0.9284	0.96377	1.0045	1.0502	1.1	1.1529
0.4	0.84286	0.86013	0.88537	0.91865	0.95985	1.0086	1.0642	1.1253	1.1904	1.257	1.3222
0.5	0.85268	0.88251	0.92286	0.97346	1.0336	1.1019	1.1764	1.254	1.3308	1.4017	1.4605
0.6	0.86532	0.91049	0.96847	1.0383	1.118	1.2046	1.2937	1.3793	1.4539	1.5086	1.5335
0.7	0.88078	0.94396	1.0217	1.1118	1.2102	1.311	1.4063	1.4859	1.5377	1.5484	1.5052
0.8	0.89904	0.98276	1.082	1.1922	1.3061	1.4138	1.5021	1.5555	1.5573	1.4915	1.346
0.9	0.92006	1.0266	1.1482	1.2768	1.4005	1.5034	1.5661	1.5678	1.4889	1.3156	1.0454
1.0	0.94381	1.0752	1.2191	1.3624	1.4866	1.5684	1.5821	1.5032	1.315	1.0155	0.62477
1.1	0.97023	1.1279	1.2929	1.4448	1.5564	1.5964	1.5341	1.3473	1.0321	0.61268	0.14763

6. 假设已知一组实测数据在文件 c8pdat.dat 中给出,试通过插值的方法绘制出三维曲面。
7. 假设已知数据由文件 c8pdat3.dat 给出,其中数据的第 1~3 列分别为 x, y, z 坐标,第 4 列为测出的 $V(x, y, z)$ 函数值,试用三维插值方法对其进行插值。
8. 考虑函数 $f(x) = \frac{\sqrt{1+x} - \sqrt{x-1}}{\sqrt{2+x} + \sqrt{x-1}}$, 试在 $x=3:0.4:8$ 处生成一组样本点,采用分段三次样条和 B 样条分别对数据进行拟合,并由数据结果求取二阶导数,将得出的结果与理论曲线进行比较。
9. 已知如下的样本点 (x_i, y_i) 数据,试对其进行分段三次多项式样条插值。

x_i	1	2	3	4	5	6	7	8	9	10
y_i	244.0	221.0	208.0	208.0	211.5	216.0	219.0	221.0	221.5	220.0

10. 假设已知某三元函数 $V(x, y, z) = e^{x^2z+y^2x+z^2y} \cos(x^2yz + z^2yx)$, 可以通过该函数生成一些网格型样本点,试根据样本点进行拟合,并给出拟合误差。
11. 习题 4 和习题 5 中给出的数据分别为一元数据和二元数据,试用分段三次样条函数和 B 样条函数对其进行拟合,并求出它们相应函数的数值导数。
12. 重新考虑习题 4 中给出的数据,试考虑用多项式插值的方法对其数据进行逼近,并选择一个能较好拟合原数据的多项式阶次。

13. 假设习题 4 中给出的数据满足原型 $y(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$, 试用最小二乘法求出 μ, σ 的值, 并用得出的函数将函数曲线绘制出来, 观察拟合效果。
14. 试将常数 e 用连分式形式表示, 找出能较精确近似其值的项数。
15. 试用连分式展开和 Padé 近似方法分别求出下面函数的有理式近似表达式, 绘制图形观察拟合效果, 并求出具有较好拟合效果的有理式阶次。
 (1) $f(x) = e^{-2x} \sin 5x$, (2) $f(x) = \frac{x^3 + 7x^2 + 24x + 24}{x^4 + 10x^3 + 35x^2 + 50x + 24} e^{-3x}$ 。
16. 假设习题 5 中数据的原型函数为 $z(x, y) = a \sin(x^2 y) + b \cos(y^2 x) + cx^2 + dxy + e$, 试用最小二乘法识别出 a, b, c, d, e 的数值。
17. 试求解下面 Mittag-Leffler 函数, 绘制出相应的曲线并验证式 (8-5-25) 等。
 (1) $\mathcal{E}_{1,1}(z)$, (2) $\mathcal{E}_{2,1}(z)$, (3) $\mathcal{E}_{1,2}(z)$, (4) $\mathcal{E}_{2,2}(z)$ 。
18. 利用本章给出的 Mittag-Leffler 函数代码, 用数值方法验证下面几个等式。
 (1) $\mathcal{E}_{\alpha,\beta}(x) + \mathcal{E}_{\alpha,\beta}(-x) = 2\mathcal{E}_{\alpha,\beta}(x^2)$, (2) $\mathcal{E}_{\alpha,\beta}(x) - \mathcal{E}_{\alpha,\beta}(-x) = 2x\mathcal{E}_{\alpha,\alpha+\beta}(x^2)$,
 (3) $\mathcal{E}_{\alpha,\beta}(x) = \frac{1}{\Gamma(\beta)} + \mathcal{E}_{\alpha,\alpha+\beta}(x)$, (4) $\mathcal{E}_{\alpha,\beta}(x) = \beta\mathcal{E}_{\alpha,\beta+1}(x) + \alpha x \frac{d}{dx} \mathcal{E}_{\alpha,\beta+1}(x)$ 。
19. 假设已知函数 $f(t) = e^{-3t} \cos(2t + \pi/3) + e^{-2t} \cos(t + \pi/4)$, 试计算其自相关函数。
20. 试求出 Gauss 分布函数 $f(t) = \frac{1}{3\sqrt{2\pi}} e^{-t^2/3^2}$ 的自相关函数, 并用 MATLAB 函数生成一组满足 Gauss 分布的伪随机数, 用这些数据检验其自相关函数是否和理论值很接近。
21. 假设由下面的语句可以生成噪声污染的信号

```
>> t=0:0.005:5; y=15*exp(-t).*sin(2*t); r=0.3*randn(size(y)); y1=y+r;
```

 试求出该信号的 Nyquist 频率, 并选择滤波频率, 设计出 8 阶 Butterworth 滤波器, 使其能有效地滤除噪声, 又有较小的延迟。
22. 高通滤波器可以滤去低频的部分, 而保留高频的部分。试为习题 21 中给出的数据设计一个高通滤波器, 提取出噪声信号, 并和叠加上的实际噪声信号 r 相比较。

参考文献

- [1] 李岳生, 黄友谦. 数值逼近. 北京: 人民教育出版社, 1978
- [2] Bosley M J, Kropheller H W, Lees F P. On the relation between the continued fraction expansion and moments matching methods of model reduction. International Journal of Control, 1973, 18:461-474
- [3] Wikipedia. Mittag-Leffler function.
- [4] Podlubny I. Fractional differential equations. San Diego: Academic Press, 1999
- [5] Podlubny I. Mittag-Leffler function. MATLAB Central File ID: # 8738
- [6] The MathWorks Inc. Signal processing user's guide, 2007

第9章 概率论与数理统计问题的计算机求解

概率论与数理统计是实验科学中常用的数学分支,其问题的求解是很重要的,但有时也是很繁琐的,传统的方法经常需要用查询表格的方式解决。MATLAB 语言提供了专用的统计学工具箱,其中包含大量的函数,可以直接求解概率论与数理统计领域的问题。本章 9.1 节将介绍概率密度、概率分布函数的基本概念及公式,介绍常用概率分布的概率密度函数、概率分布函数的分布曲线绘制,还将介绍基于概率分布的概率运算,并将介绍各种常用分布的伪随机数生成方法,如均匀分布随机数、正态分布随机数、Poisson 分布、 Γ 分布、T 分布、F 分布等随机数发生函数,还通过例子介绍一般概率问题的计算方法。9.2 节将介绍一些统计量的计算函数,如数据的均值、方差、矩量、协方差等,并介绍多元随机变量的生成,还将通过例子介绍 Monte Carlo 方法在一类计算问题中的应用。9.3 节将介绍参数估计与区间估计算法及 MATLAB 实现,并介绍多元线性回归问题的求解及非线性函数的最小二乘拟合与求解估计方法。9.4 节将介绍假设检验方法,介绍正态分布的均值假设检验、正态性假设检验、给定分布函数的假设检验等。9.5 节侧重于方差分析问题及其 MATLAB 求解,介绍单因子方差分析、双因子方差分析和主成分分析方法等内容及基于 MATLAB 语言的求解方法。

9.1 概率分布与伪随机数生成

9.1.1 概率密度函数与分布函数概述

连续随机变量概率密度函数一般记作 $p(x)$, 概率密度函数满足

$$p(x) \geq 0, \text{ 且 } \int_{-\infty}^{\infty} p(x)dx = 1 \quad (9-1-1)$$

由概率密度函数可以定义概率分布函数

$$F(x) = \int_{-\infty}^x p(t)dt \quad (9-1-2)$$

概率分布函数 $F(x)$ 的物理意义是随机变量 ξ 满足 $\xi \leq x$ 发生的概率,该函数为单调递增函数,且满足

$$0 \leq F(x) \leq 1, \text{ 且 } F(-\infty) = 0, F(\infty) = 1 \quad (9-1-3)$$

若已知某概率分布函数 $f_i = F(x_i)$, 要求出 x_i 的值,在统计学的教材中给出了各种表格,可以查出所需的 x_i 值。因为概率分布函数是单调的,所以应该能查询出合适的 x_i 值,这样的问题称为逆分布函数问题。事实上,利用 MATLAB 语言的统计工具箱中提供的函数可以更容易、更精确地求出 x_i 的值。本节将介绍几种常用的概率分布形式,并介绍 MATLAB 的求解函数。

9.1.2 常见分布的概率密度函数与分布函数

MATLAB 的统计学工具箱中提供了大量函数名有规律的函数。例如,函数名前一部分为 **gam** 常用于表示和 Γ 分布有关的函数,这类关键词在表 9-1 中给出。函数名的后一部分为 **pdf** 的表示求取概率密度函数 (probability density function), **cdf** 表示累积分布函数 (cumulative distribution function), **inv** 表示逆分布函数, **rnd** 表示随机数生成函数, **stat** 表示矩阵、方差估计, **fit** 表示参数估计。学会了这样的组合,可以立即构造出所需的函数名。例如,对数正态分布的参数与区间估计函数显然是 **logn** 和 **fit** 组合出的名字,即 **lognfit()** 函数。这里将详细介绍其中几种常用的概率分布。

表 9-1 统计学工具箱中函数名关键词一览表

关键词	分布名称	有关参数	关键词	分布名称	有关参数	关键词	分布名称	有关参数
beta	β 分布	a, b	bino	二项分布	n, p	chi2	χ^2 分布	k
ev	极值分布	μ, σ	exp	指数分布	λ	f	F 分布	p, q
gam	Γ 分布	a, λ	geo	几何分布	p	hyge	超几何分布	m, p, n
logn	对数正态分布	μ, σ	mvn	多变量正态分布	μ, σ	nbin	负二项分布	ν_1, ν_2, δ
ncf	非零 F 分布	k, δ	nct	非零 T 分布	k, δ	ncx2	非零 χ^2 分布	k, δ
norm	正态分布	μ, σ	poiss	Poisson 分布	λ	rayl	Rayleigh 分布	b
t	T 分布	k	unif	均匀分布	a, b	wbl	Weibull 分布	a, b

除了前缀的描述方法之外, MATLAB 还提供了一些函数来统一处理概率统计运算,如 **pdf()** 用来计算分布的概率密度, **cdf()**、**icdf()** 用来计算概率分布函数与逆概率分布函数, **fitdist()** 函数用来求解某种分布的特征参数等,后面将详细介绍。

1. Poisson 分布

Poisson 分布是一类离散分布函数的概率分布,它要求 x 为非负整数。对正整数参数 λ 而言, Poisson 分布的概率密度函数定义为

$$p_p(x) = \frac{\lambda^x}{x!} e^{-\lambda}, \quad x = 0, 1, 2, 3, \dots \quad (9-1-4)$$

MATLAB 语言的统计工具箱提供了 **poisspdf()**、**poisscdf()** 和 **poissinv()** 函数,分别求取 Poisson 分布的概率密度函数、分布函数及逆概率分布的值。这些函数的调用格式为 **y = poisspdf(x, λ)**, **F = poisscdf(x, λ)**, **x = poissinv(F, λ)**, 其中, x 为选定的一组横坐标向量,应该由 $x = [0:k]$ 语句生成, y 为 x 各点处的概率密度函数的值, F 为 x 各点处的分布函数值。

如果采用 **pdf()** 等统一函数,前面介绍的函数可以由下面格式调用

y = pdf('poiss', x, λ), **F = cdf('poiss', x, λ)**, **x = icdf('poiss', F, λ)**

例 9-1 试分别绘制出 $\lambda = 1, 2, 5, 10$ 时 Poisson 分布的概率密度函数与分布函数曲线。

解 仿照前面的例子,可以由下面的语句直接对不同 λ 的值分别调用 **poisspdf()** 和 **poisscdf()** 函数,绘制出概率密度函数和分布函数曲线,如图 9-1 (a)、(b) 所示

```
>> x=[0:15]'; y1=[]; y2=[]; lam1=[1,2,5,10]; n=length(lam1);
    for i=1:n, y1=[y1,poisspdf(x,lam1(i))]; y2=[y2,poisscdf(x,lam1(i))]; end
```

```
stem(x,y1), line(x,y1), figure; plot(x,y2)
```

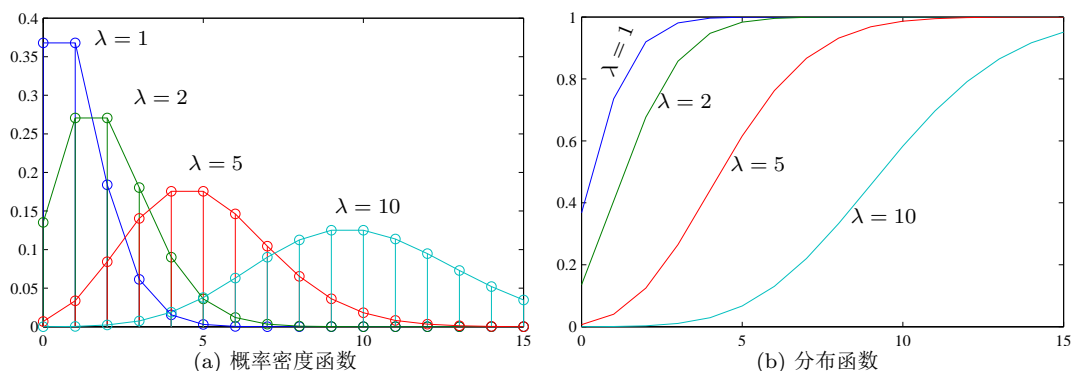


图 9-1 Poisson 分布的概率密度和分布函数曲线

2. 正态分布

正态分布的概率密度函数为

$$p_n(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/(2\sigma^2)} \quad (9-1-5)$$

其中, μ 和 σ^2 分别为正态分布的均值和方差, 可见概率密度是 μ 和 σ^2 的函数。MATLAB 语言的统计工具箱提供了相应的函数分别求取正态分布的概率密度函数与分布函数等的值。另外, 若已知概率分布值 F , 则可以求出相应的逆分布函数 x 值, 其调用格式为 $y = \text{normpdf}(x, \mu, \sigma)$, $F = \text{normcdf}(x, \mu, \sigma)$, $x = \text{norminv}(F, \mu, \sigma)$, 其中, x 为选定的一组横坐标向量, y 为 x 各点处的概率密度函数的值, F 为 x 各点处的分布函数值, 故这样得出的三个向量维数应该完全一致。 μ 为均值, σ 为标准差, 即方差的平方根。如果采用 $\text{pdf}()$ 等统一函数, 则可以用 $y = \text{pdf}('norm', x, \mu, \sigma)$, $F = \text{cdf}('norm', x, \mu, \sigma)$, $x = \text{icdf}('norm', F, \mu, \sigma)$ 命令直接求解。

例 9-2 试分别绘制出 (μ, σ^2) 为 $(-1, 1)$ 、 $(0, 0.1)$ 、 $(0, 1)$ 、 $(0, 10)$ 、 $(1, 1)$ 时正态分布的概率密度函数与分布函数曲线。

解 概率统计类教科书和数学手册中均绘制了某些 μ, σ^2 下的正态分布概率密度曲线和分布函数曲线。学习了 MATLAB 语言及前面介绍的有关函数, 读者可以用一个语句精确绘制出任意 μ, σ^2 组合下的正态分布概率密度曲线和分布函数曲线, 从而更好地利用工具高效解决概率统计问题。

这里给出的问题在 MATLAB 语言中是很容易求解的。首先在 $(-5, 5)$ 区间内构造一个横坐标向量 x , 再定义两个向量, 分别表示 μ 和 σ^2 的不同取值, 并求出相应的 σ , 这样就可以分别调用 $\text{normpdf}()$ 和 $\text{normcdf}()$ 函数, 绘制出概率密度函数和分布函数曲线, 如图 9-2(a)、(b) 所示。

```
>> x=[-5:.02:5]'; y1=[]; y2=[]; mu1=[-1,0,0,0,1]; sig1=sqrt([1,0.1,1,10,1]);
for i=1:length(mu1)
    y1=[y1,normpdf(x,mu1(i),sig1(i))]; y2=[y2,normcdf(x,mu1(i),sig1(i))];
end
plot(x,y1), figure; plot(x,y2)
```

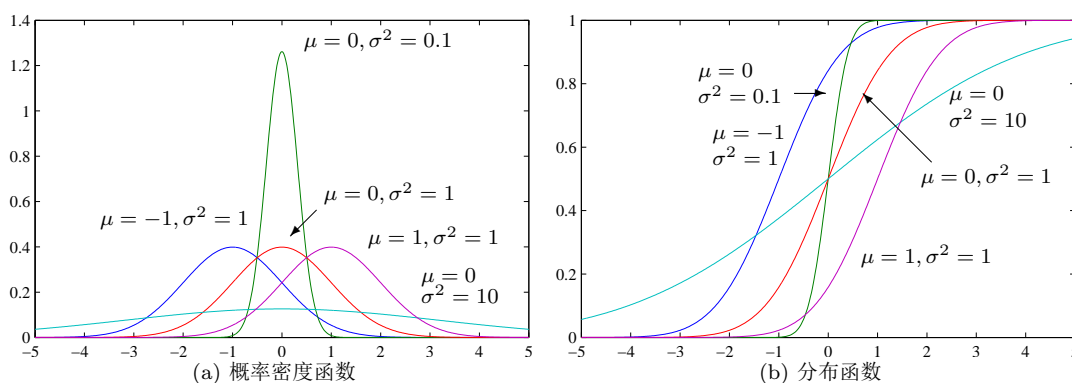


图 9-2 正态分布的概率密度和分布函数曲线

从得出的曲线可以看出,若 σ^2 相同,则曲线的形状相同,只是对不同的 μ 值进行平移即可。若 σ 不同,则曲线的形状不同, σ^2 的值越小,则概率密度曲线越陡。

$\mu=0, \sigma^2=1$ 的正态分布又称为标准正态分布,其数学表示为 $N(0,1)$ 。

3. F 分布

F 分布的概率密度函数为

$$p_F(x) = \begin{cases} \frac{\Gamma((p+q)/2)}{\Gamma(p/2)\Gamma(q/2)} p^{p/2} q^{q/2} x^{p/2-1} (p+qx)^{-(p+q)/2}, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (9-1-6)$$

F 分布的概率密度是参数 p, q 的函数,且 p, q 均为正整数。MATLAB 的统计工具箱提供了 `fpdf()`、`fcdf()` 和 `finv()` 函数,可以分别求取 F 分布的概率密度函数、分布函数与逆分布函数的值,其调用格式为 $\mathbf{y} = \text{fpdf}(\mathbf{x}, a, b)$, $\mathbf{F} = \text{fcdf}(\mathbf{x}, p, q)$, $\mathbf{x} = \text{finv}(\mathbf{F}, p, q)$, 其中 \mathbf{x} 为选定的一组横坐标向量, \mathbf{y} 为 \mathbf{x} 各点处的概率密度函数的值。如果采用 `pdf()` 等统一函数,则 $\mathbf{y} = \text{pdf}('f', \mathbf{x}, a, b)$, $\mathbf{F} = \text{cdf}('f', \mathbf{x}, p, q)$, $\mathbf{x} = \text{icdf}('f', \mathbf{F}, p, q)$ 。

例 9-3 试分别绘制出 (p, q) 对为 $(1,1)$ 、 $(2,1)$ 、 $(3,1)$ 、 $(3,2)$ 、 $(4,1)$ 时 F 分布的概率密度函数和分布函数曲线。

解 首先在 $(-0.1, 1)$ 区间内构造一个横坐标向量 \mathbf{x} ,再定义两个向量 \mathbf{p}_1 和 \mathbf{q}_1 ,分别调用 `fpdf()` 和 `fcdf()` 函数,绘制出概率密度函数和分布函数曲线,如图 9-3(a)、(b) 所示。

```
>> x=[-eps:-0.02:-0.05,0:0.02:1]; x=sort(x');
p1=[1 2 3 3 4]; q1=[1 1 1 2 1]; y1=[]; y2=[]; n=length(p1);
for i=1:n, y1=[y1,fpdf(x,p1(i),q1(i))]; y2=[y2,fcdf(x,p1(i),q1(i))]; end
plot(x,y1), figure; plot(x,y2)
```

4. T 分布

T 分布的概率密度函数为

$$p_T(x) = \frac{\Gamma((k+1)/2)}{\sqrt{k\pi} \Gamma(k/2)} (1+x^2/k)^{-(k+1)/2} \quad (9-1-7)$$

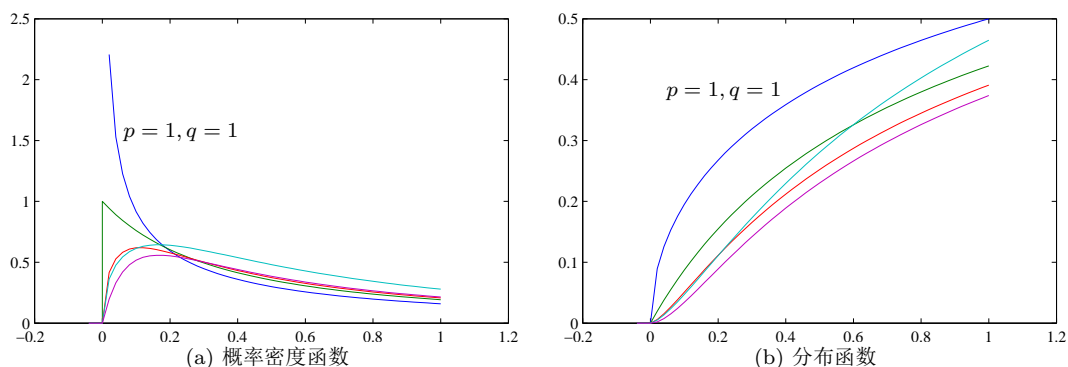


图 9-3 F 分布的概率密度和分布函数曲线

T 分布的概率密度是参数 k 的函数, 且 k 为正整数。MATLAB 语言的统计工具箱提供了 `tpdf()` 等函数, 可以分别求取 T 分布的概率密度函数、分布函数和逆分布函数的值。这些函数的调用格式为 $y = \text{tpdf}(x, k)$, $F = \text{tcdf}(x, k)$, $x = \text{tinv}(F, k)$, 其中, x 为选定的一组横坐标向量, y 为 x 各点处的概率密度函数的值。若采用 `pdf()` 等统一函数, 则 $y = \text{pdf}('t', x, k)$, $F = \text{cdf}('t', x, k)$, $x = \text{icdf}('t', F, k)$ 。

例 9-4 试分别绘制出 k 为 1、2、5、10 时 T 分布的概率密度函数与分布函数曲线。

解 首先在 $(-5, 5)$ 区间内构造一个横坐标向量 x , 再定义一个 k_1 向量, 这样就可以分别调用 `tpdf()` 和 `tcdf()` 函数, 绘制出概率密度函数和分布函数曲线, 如图 9-4(a)、(b) 所示。

```
>> x=[-5:0.02:5]'; k1=[1,2,5,10]; y1=[]; y2=[]; n=length(k1);
    for i=1:n, y1=[y1,tpdf(x,k1(i))]; y2=[y2,tcdf(x,k1(i))]; end
    plot(x,y1), figure; plot(x,y2)
```

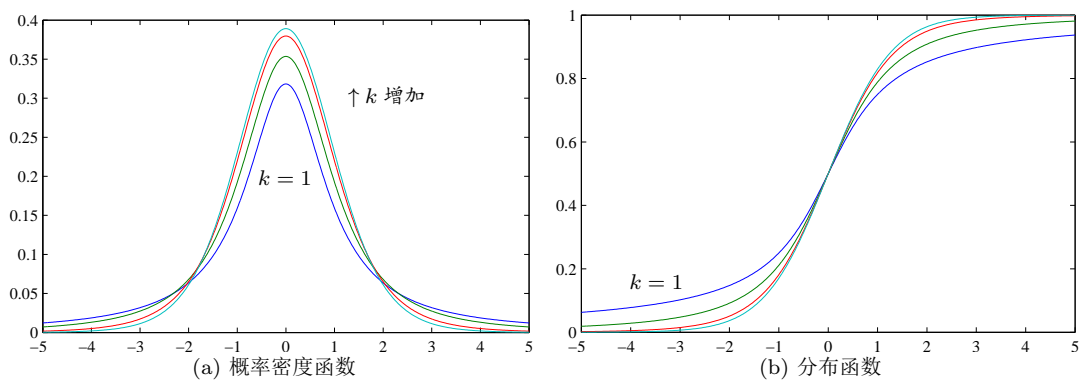


图 9-4 T 分布的概率密度和分布函数曲线

5. χ^2 分布

χ^2 分布的概率密度函数为

$$p_{\chi^2}(x) = \begin{cases} \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (9-1-8)$$

其中 k 为正整数。可以看出, χ^2 分布是一种特殊的 Γ 分布, 其中, $a = k/2, \lambda = 1/2$ 。 χ^2 分布的概率密度是参数 k 的函数。MATLAB 的统计工具箱提供了 `chi2pdf()`、`chi2cdf()` 和 `chi2inv()` 函数, 可以分别求取 χ^2 分布的概率密度函数与分布函数的值。这些函数的调用格式为 `y = chi2pdf(x, k)`, `F = chi2cdf(x, k)`, `x = chi2inv(F, k)`, 其中, x 为选定的一组横坐标向量, y 为 x 各点处的概率密度函数的值。如果选择统一函数, 则可以给出 `y = pdf('chi2', x, k)`, `F = cdf('chi2', x, k)`, `x = icdf('chi2', F, k)`。

例 9-5 试分别绘制出 k 为 1、2、3、4、5 时 χ^2 分布的概率密度函数与分布函数曲线。

解 在 $(-0.05, 1)$ 区间内构造一个横坐标向量 x , 定义一个向量, 表示 k_1 , 这样就可以分别调用 `chi2pdf()` 和 `chi2cdf()` 函数, 绘制出概率密度函数和分布函数曲线, 如图 9-5 (a)、(b) 所示。

```
>> x=[-eps:-0.02:-0.05,0:0.02:2]; x=sort(x'); k1=[1,2,3,4,5]; y1=[]; y2=[];
for i=1:length(k1)
    y1=[y1,chi2pdf(x,k1(i))]; y2=[y2,chi2cdf(x,k1(i))];
end
plot(x,y1), figure; plot(x,y2)
```

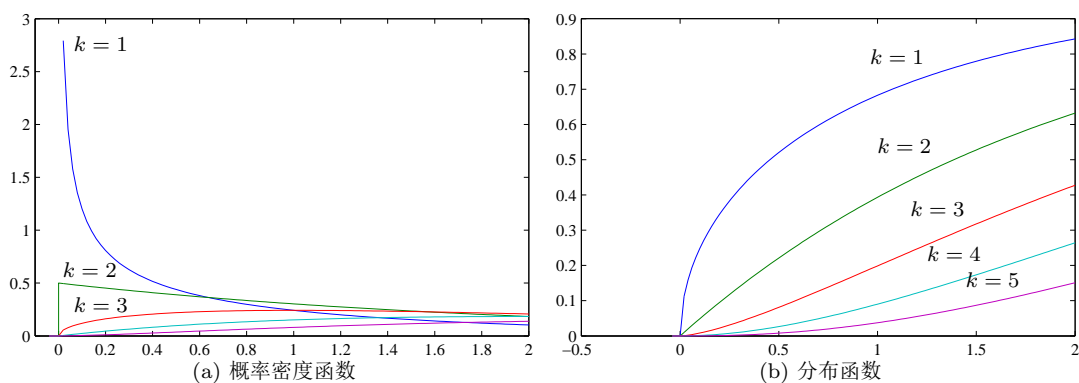


图 9-5 χ^2 分布的概率密度和分布函数曲线

6. Γ 分布

Γ 分布的概率密度函数为

$$p_{\Gamma}(x) = \begin{cases} \frac{\lambda^a x^{a-1}}{\Gamma(a)} e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (9-1-9)$$

Γ 分布的概率密度是参数 a, λ 的函数, MATLAB 语言统计工具箱提供了 `gampdf()`、`gamcdf()` 和 `gaminv()` 函数, 可以分别求取 Γ 分布的概率密度函数、分布函数及逆概率分布的值, 其调用格式为 `y = gampdf(x, a, lambda)`, `F = gamcdf(x, a, lambda)`, `x = gaminv(F, a, lambda)`, 其中, x 为选定的一组横坐标向量, y 为 x 各点处的概率密度函数的值。采用统一表示为 `y = pdf('gam', x, a, lambda)`, `F = cdf('gam', x, a, lambda)`, `x = icdf('gam', F, a, lambda)`。

例 9-6 试绘制出 (a, λ) 为 (1, 1)、(1, 0.5)、(2, 1)、(1, 2)、(3, 1) 时 Γ 分布的概率密度函数与分布函数曲线。

解 首先在 $(-0.5, 5)$ 区间内构造一个横坐标向量 x , 再定义两个向量, 分别表示 a 和 λ , 这样就可以分别调用相应的函数绘制出概率密度函数和分布函数曲线, 如图 9-6 (a)、(b) 所示。

```
>> x=[-0.5:0.02:5]'; y1=[]; y2=[]; a1=[1,1,2,1,3]; lam1=[1,0.5,1,2,1];
    for i=1:length(a1)
        y1=[y1,gampdf(x,a1(i),lam1(i))]; y2=[y2,gamcdf(x,a1(i),lam1(i))];
    end
    plot(x,y1), figure; plot(x,y2)
```

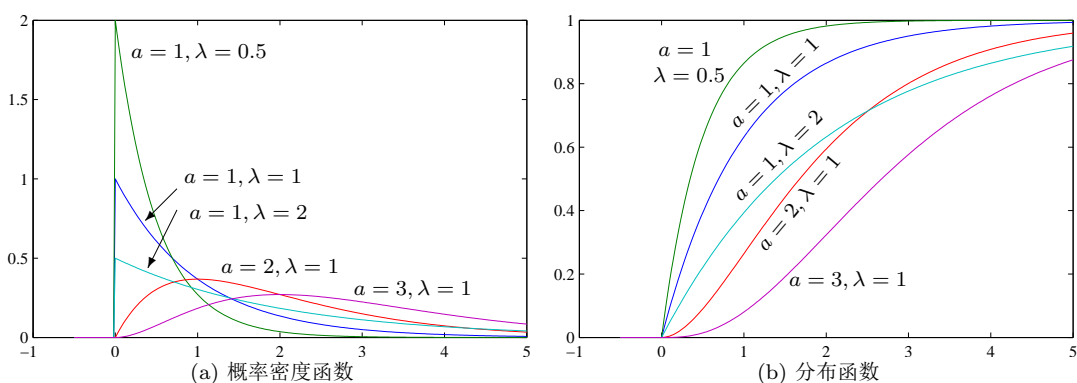


图 9-6 Γ 分布的概率密度函数和分布函数曲线

这里的概率密度图形稍有些问题, 因为绘图时横坐标步距选择为 0.02, 而因为在 -0.02 点处概率密度的值为 0, 则在 0 处的函数值为一个非零数值 p_1 , 所以在图形上看起来在 $x \leq 0$ 时函数值不等于 0。事实上, 在 $x = 0$ 处垂直上升为 p_1 , 在选择横坐标向量时改用下面的语句即可解决此问题。

```
>> x=[-eps:-0.02:-0.05,0:0.02:5]; x=sort(x');
```

7. Rayleigh 分布

Rayleigh 分布的概率密度函数为

$$p_r(x) = \begin{cases} \frac{x}{b^2} e^{-x^2/(2b^2)}, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (9-1-10)$$

该函数是 b 的函数, MATLAB 语言的统计工具箱提供了 `raylpdf()`、`raylcdf()` 和 `raylinv()`, 可以分别求取 Rayleigh 分布的概率密度函数、分布函数与逆分布函数的值。这些函数的调用格式为 `y = raylpdf(x,b)`, `F = raylcdf(x,b)`, `x = raylinv(F,b)`, 其中 x 为选定的一组横坐标向量, y 为 x 各点处的概率密度函数的值。如果采用统一函数, 则需要给出 `y = pdf('rayl',x,b)`, `F = cdf('rayl',x,b)`, `x = icdf('rayl',F,b)`。

例 9-7 试分别绘制出 b 为 0.5、1、3、5 时 Rayleigh 分布的概率密度函数与分布函数曲线。

解 首先在 $(-0.1, 5)$ 区间内构造一个横坐标向量 x , 再定义向量 b_1 , 这样就可以分别调用 `raylpdf()` 和 `raylcdf()` 函数, 绘制出概率密度函数和分布函数曲线, 如图 9-7 (a)、(b) 所示。

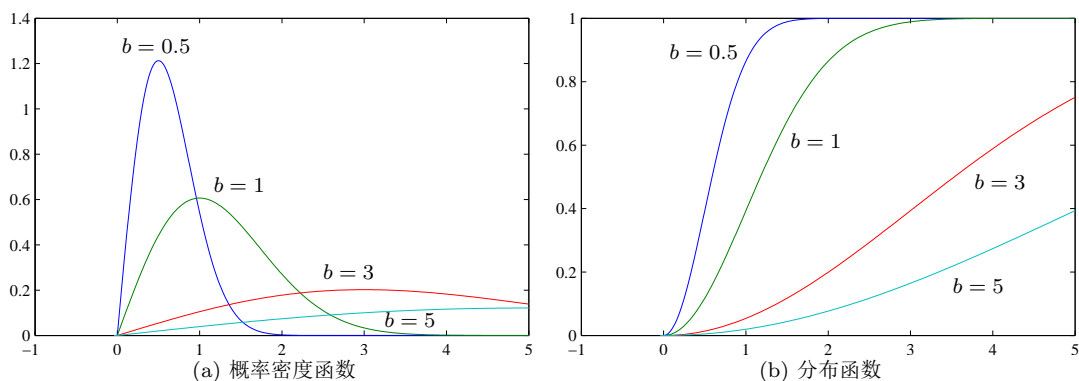


图 9-7 Rayleigh 分布的概率密度函数和分布函数曲线

```
>> x=[-eps:-0.02:-0.05,0:0.02:5]; x=sort(x'); b1=[.5,1,3,5]; y1=[]; y2=[];
    for i=1:length(b1), y1=[y1,raylpdf(x,b1(i))]; y2=[y2,raylcdf(x,b1(i))]; end
    plot(x,y1), figure; plot(x,y2)
```

MATLAB 语言的统计学工具箱还提供了其他各种分布的概率密度函数、分布函数及逆分布函数的函数,可以直接获得各种指定的随机变量分布求取问题。

9.1.3 概率问题的求解

前面介绍过,某随机变量 ξ 分布函数 $F(x)$ 的物理含义是随机变量 ξ 落入 $(-\infty, x)$ 区间的概率,故可以利用分布函数的概念求取满足条件的概率。例如,若想求出 ξ 落入区间 $[x_1, x_2]$ 的概率 $P[x_1 \leq \xi \leq x_2]$,则可以用两个分布函数之差求出。下面列出几个概率公式

$$\begin{cases} P[\xi \leq x] = F(x), & \xi \leq x \text{ 的概率} \\ P[x_1 \leq \xi \leq x_2] = F(x_2) - F(x_1), & x_1 \leq \xi \leq x_2 \text{ 的概率} \\ P[\xi \geq x] = 1 - F(x), & \xi \geq x \text{ 的概率} \end{cases} \quad (9-1-11)$$

其中,分布函数可以用前面介绍的分布函数求出。下面将通过例子演示概率问题的求解。

例 9-8 假设已知某随机变量 x 满足 Rayleigh 分布,且 $b=1$,试分别求出该随机变量 x 值落入区间 $[0.2, 2]$ 及区间 $[1, \infty)$ 的概率。

解 由概率分布函数可以容易地求出随机变量 x 落入 $(-\infty, 0.2]$ 区间和 $(-\infty, 2]$ 区间的概率,所以可以由下面的语句立即求出变量落入指定区间的概率为 0.8449。

```
>> b=1; p1=raylcdf(0.2,b); p2=raylcdf(2,b); P1=p2-p1
```

另外,由于能直接求出 $(-\infty, 1]$ 区间的概率,故可以求出 $x \geq 1$ 的概率为 0.6065。

```
>> p1=raylcdf(1,b); P2=1-p1
```

例 9-9 假设二维随机变量 (ξ, η) 的联合概率密度为 $p(x, y) = \begin{cases} x^2 + \frac{xy}{3}, & 0 \leq x \leq 1, 0 \leq y \leq 2 \\ 0, & \text{其他} \end{cases}$,

试求出 $P[\xi < 1/2, \eta < 1/2]$ 。

解 已知概率密度 $p(x, y)$,则可以通过积分求出 $P[\xi < x_0, \eta < y_0]$ 的值为 $5/192$ 。

```
>> syms x y; f=x^2+x*y/3; P=int(int(f,x,0,1/2),y,0,1/2)
```

此处积分下限直接取 0 而不是 $-\infty$ 是因为联合概率密度函数的值在自变量为负的时候为 0, 故不直接写出其积分。

例 9-10 假设某两地 A、B 间有 6 个交通岗, 在各个交通岗处遇到红灯的概率均相同, 为 $p = 1/3$, 且中途遇到红灯的次数满足二项分布 $B(6, p)$ [6], 试求出某人从 A 地出发到达 B 地至少遇到一次红灯的概率。若选择不同的 p 值, 试再绘制出至少遇到一次红灯的概率曲线。

解 由于已知在每个交通岗处遇到红灯的概率密度满足二项分布, 可以由 `binopdf()` 或 `pdf()` 计算。记某人遇到红灯的次数为 x , 则 x 的取值可以为 0, 1, 2, \dots , 6, 相应的概率密度可以如下求出

```
>> x=0:6; y=binopdf(x,6,1/3)
```

上述语句得出 $y = [0.0878, 0.2634, 0.3292, 0.2195, 0.0823, 0.0165, 0.0014]$ 。可见, 只遇到一次红灯的概率为 0.2634。至少遇到一次红灯的概率可以由两种方法求出, 其一是 1 减去不遇红灯的概率(即遇到次数为 0 的概率), 另一种是将 y 向量中第一项以外的全部项加起来, 由下面语句可知, 至少遇到一次红灯的概率为 $p = 91.22\%$ 。

```
>> P=1-y(1) % 或 P=sum(y(2:end))
```

如果二项分布参数 p 发生变化, 则可以用循环结构重新计算出概率曲线, 如图 9-8 所示。

```
>> p0=0.05:0.05:0.95; y=[];
for p=p0, y=[y 1-binopdf(0,6,p)]; end, plot(p0,y,1/3,P,'o')
```

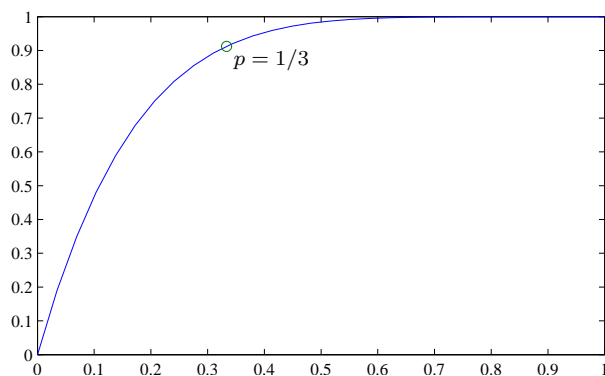


图 9-8 参数 p 变化时至少遇到一次红灯的概率

9.1.4 随机数与伪随机数

在科学研究和统计分析中经常要用到随机数据。随机数的生成通常有两类方法, 其一是依赖一些专用的电子元件发出随机信号, 这种方法又称为物理生成法; 另一类是通过数学的算法, 仿照随机数发生的规律计算出随机数, 由于产生的随机数是由数学公式计算出来的, 所以这类随机数又称为“伪随机数”。

伪随机数至少有两个优点: 首先, 若选择相同的随机数种子, 则随机数是可以重复的, 这样就创造了重复实验的条件; 其次, 随机数满足的统计规律可以人为地选择, 例如可以自由地选择均匀分布、正态分布、Poisson 分布等, 来满足人们的需要。

在 4.1 节中介绍了 `rand()` 和 `randn()` 两个函数,可以分别生成均匀分布和正态分布伪随机数,并介绍了如何生成任意区间的均匀分布伪随机数及任意给定均值、方差的伪随机数生成方法。除了这两类伪随机数之外,在应用中还需要其他各类随机数,如前面介绍的各种概率密度函数对应的随机数。在 MATLAB 环境中,可以用统计工具箱中的语句生成所需的随机数,具体的命令为

```
A = gamrnd(a,λ,n,m)    % 生成  $n \times m$  的  $\Gamma$  分布的伪随机数矩阵
B = chi2rnd(k,n,m)    % 生成  $\chi^2$  分布的伪随机数
C = trnd(k,n,m)      % 生成 T 分布的伪随机数
D = frnd(p,q,n,m)    % 生成 F 分布的伪随机数
E = raylrnd(b,n,m)    % 生成 Rayleigh 分布的伪随机数
```

更一般地,可以使用统一函数 `random()` 来生成一般的伪随机数,该函数的调用格式为 `N = random(类型,参数,n,m)`,其中“类型”与“参数”与表 9-1 中给出的一致,例如,上述的 **A**、**C** 矩阵可以通过 `A = random('gam',a,λ,n,m)`、`C = random('t',k,n,m)` 直接生成,使得不同分布的随机数生成方法更规范。

例 9-11 令 $b = 1$, 试生成 30000×1 个 Rayleigh 分布的随机数,并用直方图检验生成数据的概率分布情况,和理论曲线进行比较。

解 由前面的语句可知,由 `raylrnd()` 函数可以立即生成 30000×1 的随机数向量。人为定义一个向量 `xx`,可以用 `hist()` 函数找出随机数落入各个子区间点的个数,并由此拟合出生成数据的概率密度,用 `bar()` 函数表示出来。用下面语句可以实现上述功能,并将拟合直方图与理论概率密度在同一坐标系下绘制出来,如图 9-9 所示。可见,生成的数据能较好地满足指定的分布。

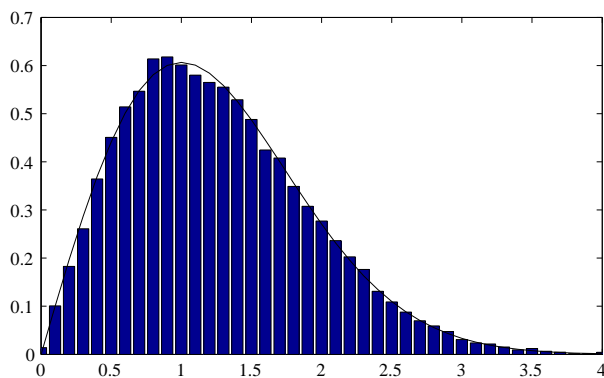


图 9-9 Rayleigh 分布函数与生成数据分布

```
>> b=1; p=random('rayl',1,30000,1); xx=0:.1:4; yy=hist(p,xx);
yy=yy/(30000*0.1); bar(xx,yy), y=raylpdf(xx,1); line(xx,y)
```

新版本 MATLAB 提供了 `rng()` 函数来控制伪随机数的生成方式,由 `s = rng` 命令可以获得当前伪随机数的控制变量,在生成新随机数前调用 `rng(s)` 则可以生成与前面相同的伪随机数,从而达到重复试验的目的。此方法适用于各种伪随机数生成函数。

9.2 统计量分析

9.2.1 随机变量的均值与方差

假设连续随机变量 x 的概率密度函数为 $p(x)$, 则可以如下定义该变量的数学期望 $E[x]$ 和方差 $D[x]$ 为

$$E[x] = \int_{-\infty}^{\infty} xp(x)dx, \quad D[x] = \int_{-\infty}^{\infty} (x - E[x])^2 p(x)dx \quad (9-2-1)$$

利用 MATLAB 符号运算工具箱中的积分函数可以求出这两个重要的统计量。

例 9-12 试用积分方法求取 Γ 分布 ($a > 0, \lambda > 0$) 的均值与方差。

解 利用 MATLAB 的符号运算工具箱可以立即得出 $m = a/\lambda, s = a/\lambda^2$ 。

```
>> syms x; syms a lam positive
p=lam^a*x^(a-1)/gamma(a)*exp(-lam*x); m=int(x*p,x,0,inf)
s=simple(int((x-1/lam*a)^2*p,x,0,inf))
```

假设在实际中测出一组样本数据 $x_1, x_2, x_3, \dots, x_n$, 则该随机变量的均值和方差分别定义为

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \hat{s}_x^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (9-2-2)$$

由统计学理论可以证明, 这样定义的 \hat{s}_x^2 方差是有偏的, 所以在实际应用中经常采用无偏的方差, 即

$$s_x^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (9-2-3)$$

并称 $s_x \geq 0$ 为标准差。

若已知一组随机变量样本数据构成的向量 $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]^T$, 则可以直接使用 MATLAB 函数 `mean()`、`var()` 和 `std()` 求出该向量各个元素的均值、方差和标准差。这 3 个函数的调用格式为 `m = mean(x)`, `s2 = var(x)`, `s = std(x)`, 这 3 个函数和其他函数还可以处理 \mathbf{x} 为矩阵的形式。具体的解释是, 对矩阵 \mathbf{x} 的每个列向量进行均值、方差和标准差分析就可以得出一个行向量。若想将矩阵或多维数组 \mathbf{x} 全部元素进行统计分析, 最简单的格式是 `m = mean(x(:))`。

例 9-13 试生成一组 30000 个正态分布随机数, 使其均值为 0.5, 标准差为 1.5, 试分析这些数据实际的均值、方差和标准差。如果减小随机变量个数, 会有什么结果?

解 可以用下面的语句生成所需的随机数, 并求出该变量的均值为 0.4879, 方差为 2.2748, 标准差为 1.5083。可见, 这样得出的数据均值和方差与理论值比较接近。

```
>> p=random('norm',0.5,1.5,30000,1); mean(p), var(p), std(p)
```

若减小随机数个数, 例如选择 300 个随机数, 则可以由以下的语句得出新生成随机数的均值为 0.4745, 方差为 1.9118, 标准差为 1.3827。可见, 得出的随机数标准差与理论值相差较大, 所以在进行较精确的统计分析时不能选择太少的样本点。

```
>> p=random('norm',0.5,1.5,300,1); mean(p), var(p), std(p)
```

前面介绍过各种常见的分布函数,如正态分布、 Γ 分布等,如果给定了分布,则可以用 MATLAB 统计学工具箱中的现成函数,如 `normstat()` 或 `gamstat()` 等函数直接求出该分布的均值和方差,分布类型标识后加后缀 `stat`,这样就可以构造出一类求取均值和方差的函数。例如, `gamstat()` 函数的调用格式为 `[\mu, \sigma^2] = gamstat(a, \lambda)`, 返回的变量为相关分布的均值和方差,还可由 `[\mu, \sigma^2] = fittest('gam', a, \lambda)` 求解。

例 9-14 试求出 Rayleigh 分布 ($b = 0.45$) 的均值与方差。

解 由于需要求解 Rayleigh 分布,所以需要使用的函数名应该为 `raylstat()`, 可以通过下面的语句直接求出该分布的均值为 $m = 0.5640$, 方差为 $s = 0.0869$ 。

```
>> [m,s]=raylstat(0.45)
```

9.2.2 随机变量的矩

假设 x 为连续随机变量,且 $p(x)$ 为其概率密度函数,则可以由下面的式子定义出该变量的 k 阶原点矩和中心矩为

$$\nu_k = \int_{-\infty}^{\infty} x^k p(x) dx, \quad \mu_k = \int_{-\infty}^{\infty} (x - \mu)^k p(x) dx \quad (9-2-4)$$

可见, $\nu_1 = E[x]$, $\mu_2 = D[x]$ 。

例 9-15 考虑例 9-12 中 Γ 分布的原点矩和中心矩,并由前几项结果总结一般规律。

解 先用下面的语句求解原点矩

```
>> syms x; syms a lam positive; p=lam^a*x^(a-1)/gamma(a)*exp(-lam*x);
    for n=1:5, m=simple(int(x^n*p,x,0,inf)), end
```

得出的结果分别为

$$\nu_{1 \sim 5} = \frac{a}{\lambda}, \frac{a}{\lambda^2}(a+1), \frac{a}{\lambda^3}(a+1)(a+2), \frac{a}{\lambda^4}(a+1)(a+2)(a+3), \frac{a}{\lambda^5}(a+1)(a+2)(a+3)(a+4)$$

可以总结为

$$\nu_k = \frac{1}{\lambda^k} a(a+1)(a+2) \cdots (a+k-1) = \frac{1}{\lambda^k} \prod_{i=0}^{k-1} (a+i) = \frac{\lambda^{-k} \Gamma(a+k)}{\Gamma(a)}$$

```
>> syms k; m=simple(int((x)^k*p,x,0,inf))
```

同样,可以通过下面的语句求出原问题的中心矩(建议使用早期版本)

```
>> for n=1:7, s=simple(int((x-1/lam*a)^n*p,x,0,inf)), end
```

各个中心矩的数学表示如下,但好像这样的积分问题没有规律性的化简结果。

$$\mu_{1 \sim 7} = 0, \frac{a}{\lambda^2}, \frac{2a}{\lambda^3}, \frac{3a(a+2)}{\lambda^4}, \frac{4a(5a+6)}{\lambda^5}, \frac{5a(3a^2+26a+24)}{\lambda^6}, \frac{6a(35a^2+154a+120)}{\lambda^7}$$

若给定的随机数为一些样本点 x_1, x_2, \cdots, x_n , 则该随机变量的 k 阶原点矩与中心矩的定义分别为

$$A_k = \frac{1}{n} \sum_{i=1}^n x_i^k, \quad B_k = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^k \quad (9-2-5)$$

MATLAB 语言的统计学工具箱提供了 `moment()` 函数, 可以求出向量 x 的中心高阶矩, 但没有直接函数可以求出原点矩。其实, 可以用下面的语句求出给定随机向量 x 的 r 阶原点矩与中心矩为 $A_k = \text{sum}(x.^k)/\text{length}(x)$, $B_k = \text{moment}(x, k)$ 。

例 9-16 仍考虑前面的随机数, 可以用下面的语句得出随机数的各阶矩为

```
>> A=[]; B=[]; p=random('norm',0.5,1.5,30000,1); n=1:5;
    for r=n, A=[A, sum(p.^r)/length(p)]; B=[B,moment(p,r)]; end
```

由数值方法可以求出其各阶原点矩分别为 0.5081、2.5155、3.5457、18.8911、40.7912, 中心矩分别为 0、2.2689、0.0133、15.2391、0.0865。

由下面的语句还可以求出各阶矩的理论值, 分别为 $A_1^T = [1/2, 5/2, 7/2, 149/8, 653/16]$, $B_1^T = [0, 9/4, 0, 243/16, 0]$ 。可以看出, 从生成的数据求出的各阶矩和理论值的拟合程度也是很好的。

```
>> syms x; A1=[]; B1=[]; a=-inf; b=inf;
    p=1/(sqrt(2*sym(pi))*3/2)*exp(-(x-1/2)^2/(2*(3/2)^2));
    for i=1:5, A1=[A1,int(x^i*p,x,a,b)]; B1=[B1,int((x-1/2)^i*p,x,a,b)]; end
```

9.2.3 多变量随机数的协方差分析

假设随机数 $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$ 为二维随机变量对 (x, y) 的样本, 则可以分别定义出二维样本的协方差 s_{xy} 与二维样本的相关系数 η 为

$$s_{xy} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}), \quad \eta = \frac{s_{xy}}{s_x s_y} \quad (9-2-6)$$

由上述的式子还可以定义出矩阵 C 为

$$C = \begin{bmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{bmatrix} \quad (9-2-7)$$

式中, $c_{xx} = s_x^2$, $c_{xy} = c_{yx} = s_{xy}$, 该矩阵称为协方差矩阵 (covariance matrix)。

多个随机变量的协方差矩阵可以由上述定义扩展出来。MATLAB 中提供了一个专门求解多元随机变量协方差均值的函数 `cov()`。该函数的调用格式为 $C = \text{cov}(X)$, 其中, X 的各列均表示不同的随机变量的样本值。若 X 是向量, 则得出的是其方差, 否则将返回协方差矩阵 C 。

例 9-17 试用 MATLAB 语言产生 4 个满足标准正态分布的随机变量, 并求出其协方差矩阵。

解 用 MATLAB 给出的 `randn()` 函数可以生成一个标准正态分布随机数的矩阵。该矩阵有 4 列, 表示 4 个不同的随机数变量。该矩阵有 30000 行, 表示每个随机数变量均取 30000 个样本点。这样, 由下面的语句可以立即得出这 4 个随机数变量的协方差矩阵为

$$R = \begin{bmatrix} 1.0064 & 0.0012589 & 0.004726 & -0.00051901 \\ 0.0012589 & 1.004 & -0.00094545 & 0.004802 \\ 0.004726 & -0.00094545 & 1.011 & -0.011895 \\ -0.00051901 & 0.004802 & -0.011895 & 0.99476 \end{bmatrix}$$

可见, 该矩阵是对称矩阵, 趋近于理论上的单位阵。

```
>> p=randn(30000,4); R=cov(p)
```

9.2.4 多变量正态分布的联合概率密度函数及分布函数

假设有 n 个正态分布的随机变量 $\xi_1, \xi_2, \dots, \xi_n$, 它们的均值分别为 $\mu_1, \mu_2, \dots, \mu_n$, 可以构成一个均值向量 $\boldsymbol{\mu}$, 这些变量的协方差矩阵为 $\boldsymbol{\Sigma}^2$, 可以按下面的方式构造出随机数向量为 $\boldsymbol{x} = [x_1, x_2, \dots, x_n]^T$, 这样就可以定义出这些随机变量的联合概率密度为

$$p(x_1, x_2, \dots, x_n) = \frac{1}{\sqrt{2\pi}} \boldsymbol{\Sigma}^{-1} e^{-\frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Sigma}^{-2} \boldsymbol{x}} \quad (9-2-8)$$

MATLAB 语言的统计学工具箱中提供了 `mvnpdf()` 函数, 利用该函数可以计算出多变量正态分布的联合概率密度值。该函数的调用格式为 `p = mvnpdf(X, μ, Σ2)`, 其中, \boldsymbol{X} 为 n 列的矩阵, 表示各个随机变量的取值, 每一列表示一个随机变量, $\boldsymbol{\mu}$ 为每个随机变量均值构成的向量, $\boldsymbol{\Sigma}^2$ 为这些随机变量的协方差矩阵, 这样生成的 \boldsymbol{p} 矩阵为列向量, 表示每个随机变量组合的联合概率密度函数。

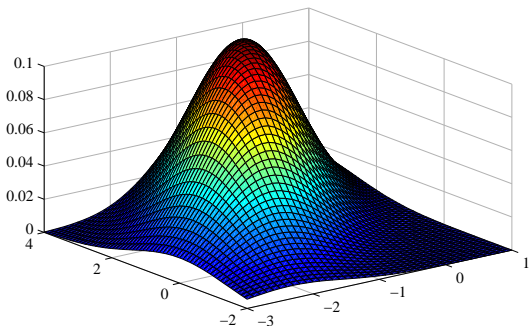
例 9-18 试绘制出均值为 $\boldsymbol{\mu} = [-1, 2]^T$, 协方差矩阵为 $\boldsymbol{\Sigma}^2 = [1, 1; 1, 3]$ 的二维正态分布的联合概率密度函数。若协方差矩阵的非对角线元素为 0, 试绘制出新的联合概率密度函数。

解 由于 `mvnpdf()` 函数只支持一个双列矩阵来表示 \boldsymbol{X} , 所以应该用适当的转换方法将其转换成两个列向量, 再构成两列的矩阵, 由该矩阵就可以求出联合概率密度向量, 将该向量用 `reshape()` 函数还原成矩阵形式, 最后用三维网格图的形式显示出来, 如图 9-10 (a) 所示。

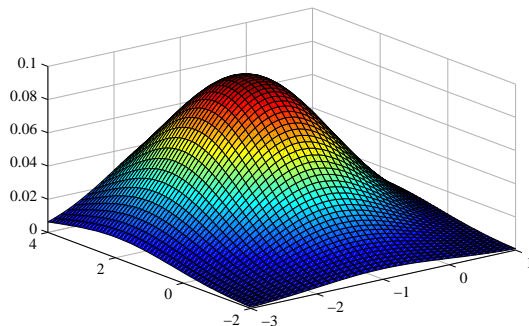
```
>> mu1=[-1,2]; Sigma2=[1 1; 1 3]; % 输入均值向量和协方差矩阵
[X,Y]=meshgrid(-3:0.1:1,-2:0.1:4); xy=[X(:) Y(:)]; % 产生网格数据并处理
p=mvnpdf(xy,mu1,Sigma2); P=reshape(p,size(X)); % 求取联合概率密度
surf(X,Y,P) % 绘制联合概率密度的三维表面图
```

对协方差矩阵进行处理, 则可以消除协方差矩阵的非对角元素。重新执行下面的语句可以计算出新的联合概率密度函数, 如图 9-10 (b) 所示。

```
>> Sigma2=diag(diag(Sigma2)); % 消除协方差矩阵的非对角元素
p=mvnpdf(xy,mu1,Sigma2); P=reshape(p,size(X)); surf(X,Y,P)
```



(a) 原协方差矩阵



(b) 对角协方差矩阵

图 9-10 二维正态分布的联合概率密度函数

MATLAB 的统计学工具箱还提供了 `mvnrnd()` 函数,用于产生多变量正态分布随机数。该函数的调用格式为 `R = mvnrnd(μ , Σ^2 , m)`,该函数可以生成 m 组满足多变量正态分布的随机变量,返回的 R 为 $m \times n$ 矩阵,每一列表示一个随机变量。

例 9-19 观察例 9-18 中给出的两种二维正态分布的伪随机数分布情况。

解 下面的语句可以得出两种分布下 2000 个点在 x - y 平面上的分布情况,如图 9-11 (a)、(b) 所示。可见,若协方差矩阵为对角矩阵,则两个随机变量之间没有必然联系,所以从分布图看不出随机变量的分布偏向性,而协方差矩阵不是对角矩阵时,随机变量明显有偏向性

```
>> mu1=[-1,2]; Sigma2=[1 1; 1 3]; R1=mvnrnd(mu1,Sigma2,2000);
    plot(R1(:,1),R1(:,2),'o'), Sigma2=diag(diag(Sigma2)); figure;
    R2=mvnrnd(mu1,Sigma2,2000); plot(R2(:,1),R2(:,2),'o')
```

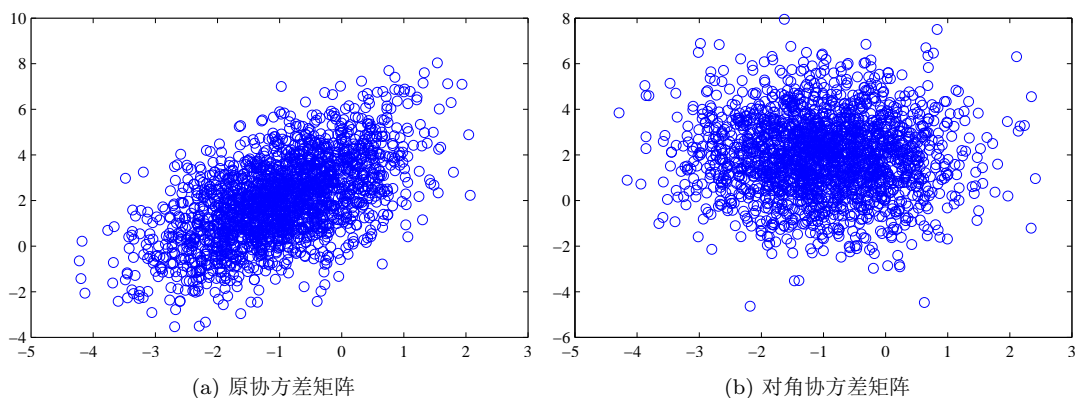


图 9-11 二维正态分布随机数分布情况

9.2.5 基于 Monte Carlo 法的数学问题求解

Monte Carlo 法是通过大量实验来求取随机变量近似值的一种常用的方法,在现代科学研究中经常用来求解一些建模困难的问题。这里将通过例子演示该方法的入门知识。

考虑图 9-12 (a) 中给出的示意图。假设正方形的边长为 1,可见,四分之一圆的面积是 $\pi/4$,其面积和正方形面积的比是 $\pi/4:1$,换句话说,如果产生一个均匀分布的随机数,它落到四分之一圆的概率为 $\pi/4$ 。生成 N 组随机数 x_i 和 y_i ,使其均为区间 $[0, 1]$ 内均匀分布的随机数。这样,落入四分之一圆,即满足 $x_i^2 + y_i^2 \leq 1$ 条件的点的个数为 N_1 ,则对大量的实验数据,有 $N_1/N \approx \pi/4$,即 $\pi \approx 4N_1/N$ 。如果 N 足够大,则可以通过前面的式子近似求出 π 的值。

例 9-20 试用 Monte Carlo 法近似求出 π 的值。

解 参照前面介绍的算法,可以由下面语句求出 π 的值

```
>> N=100000; x=rand(1,N); y=rand(1,N); i=(x.^2+y.^2)<=1; N1=sum(i); p=N1/N*4
```

这样得出 $\pi \approx 3.1418$ 。再进一步增加 N 的值,可以改进求解的精度,但应该注意,用这样的方法不可能得出绝对精确的 π 值。

考虑积分问题 $\int_a^b f(x) dx$ 。若 $f(x) \geq 0$,则基于 Monte Carlo 方法的示意图如图 9-12

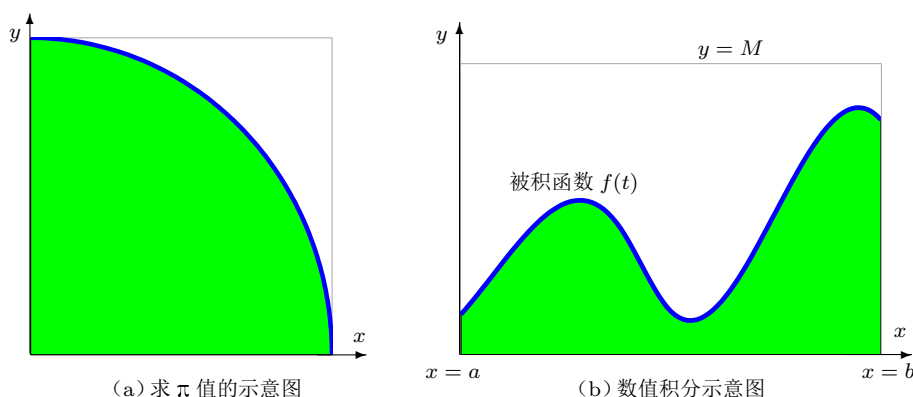


图 9-12 Monte Carlo 法近似的示意图

(b) 所示。仍考虑生成 N 组随机数 x_i 和 y_i , 它们分别为区间 $[a, b]$ 和 $(0, M)$ 上均匀分布的随机数。记满足不等式 $y_i \leq f(x_i)$ 的点的个数为 N_1 , 则可以得出, $\frac{N_1}{N} \approx \frac{1}{M(b-a)} \int_a^b f(x) dx$, 这样可以近似得出^[1]

$$\int_a^b f(x) dx \approx \frac{M(b-a)N_1}{N} \quad (9-2-9)$$

例 9-21 试用 Monte Carlo 法计算积分 $\int_1^3 [1 + e^{-0.2x} \sin(x+0.5)] dx$ 。

解 可以由下面语句求出积分值为 $p = 2.7343$, 事实上, 该积分的精确结果为 $I = 2.7439$ 。

```
>> f=@(x)1+exp(-0.2*x).*sin(x+0.5); a=1; b=3; M=2; N=100000;
    x=a+(b-a)*rand(N,1); y=M*rand(N,1); i=y<=f(x); N1=sum(i); p=M*N1*(b-a)/N
    syms x; I=vpa(int(1+exp(-0.2*x)*sin(x+0.5),x,a,b))
```

值得指出的是, 用这样的方法只能求解满足 $f(x) \geq 0$ 或 $f(x) \leq 0$ 的问题, 否则应该采用改进的 Monte Carlo 方法来求解。

9.3 数理统计分析方法及计算机实现

9.3.1 参数估计与区间估计

若实测一组数据 $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, 且已知这些数据满足某种分布, 如正态分布, 则可以用 MATLAB 语言统计学工具箱中的函数 `normfit()` 由极大似然法估计出该分布的均值 μ 及方差 σ^2 , 且同时估计出其置信区间 $\Delta\mu$ 及 $\Delta\sigma^2$ 。该函数的调用格式为

$$[\mu, \sigma^2, \Delta\mu, \Delta\sigma^2] = \text{normfit}(\mathbf{x}, P_{ci})$$

其中, P_{ci} 为用户指定的置信度, 如 95%, 利用该值, 此函数会自动调用 `norminv()` 函数求出相关值, 这样就可以得出所需的参数。可以预见, P_{ci} 的值越大 (亦即越趋近于 1), 则得出的置信区间将越小, 亦即得出的结果越接近于真值。

类似于其他概率分布的密度函数等, 该工具箱还提供了其他分布的函数, 如 Γ 分布的参数估计函数 `gamfit()`、Rayleigh 分布的参数估计函数 `raylfit()`、均匀分布的参数估计函

数 `unifit()`、Poisson 分布的参数估计函数 `poissfit()` 等,这些函数的调用格式很接近,在此不详细介绍。除此之外,在新版的 MATLAB 下还可以采用统一的拟合函数 `fitdist()`。

例 9-22 试用 `gamrnd()` 函数生成一组 $a = 1.5, \lambda = 3$ 的伪随机数,用参数估计的方法以不同的置信度进行估计,比较估计结果。

解 假设生成一组 30000 个数据,并选择置信度为 90%、92%、95%、98%,可以用下面的语句得出不同置信度下的参数估计结果

```
>> p=gamrnd(1.5,3,30000,1); Pv=[0.9,0.92,0.95,0.98]; A=[];
    for i=1:length(Pv)
        [a,b]=gamfit(p,Pv(i)); A=[A; Pv(i),a(1),b(:,1)',a(2),b(:,2)'];
    end
```

为了便于理解,下面用表格的形式将得出的结果显示出来,如表 9-2 所示,表格的全部数据为上述程序中 A 矩阵。可见,置信度选择不同,不会影响参数估计值,但会影响到置信区间的大小。事实上,在一般应用中通常选择 95% 的置信度。

表 9-2 不同置信度下的参数估计结果

置信度	a 参数估计结果			λ 参数估计结果		
	\hat{a}	a_{\min}	a_{\max}	$\hat{\lambda}$	λ_{\min}	λ_{\max}
90%	1.506500556	1.505099132	1.507901979	2.991117941	2.987797191	2.994438691
92%	1.506500556	1.505380481	1.507620631	2.991117941	2.988463861	2.993772021
95%	1.506500556	1.505801226	1.507199886	2.991117941	2.98946084	2.992775042
98%	1.506500556	1.506220978	1.506780134	2.991117941	2.990455465	2.991780417

现在考虑随机数数目的不同选择,考虑选择 300、3000、30000、300000、3000000 个随机数,则可以通过下面语句计算出 95% 置信度下参数估计与置信区间的变化。同样,为了更好地显示估计结果,将以列表的形式显示,如表 9-3 所示。

表 9-3 不同随机数个数的参数估计结果

随机数个数	a 参数估计结果			λ 参数估计结果		
	\hat{a}	a_{\min}	a_{\max}	$\hat{\lambda}$	λ_{\min}	λ_{\max}
300	1.548677954	1.540991679	1.55636423	2.91172985	2.896265076	2.927194623
3000	1.476057561	1.473908973	1.47820615	3.040589493	3.035438607	3.04574038
30000	1.503327455	1.502624743	1.504030167	2.976242793	2.974591027	2.977894559
300000	1.509546583	1.509323617	1.50976955	2.984774009	2.984252596	2.985295421
3000000	1.498005677	1.497935817	1.498075536	3.006048895	3.005882725	3.006215065

```
>> num=[300,3000,30000,300000,3000000]; A=[];
    for i=1:length(num), p=gamrnd(1.5,3,num(i),1)
        [a,b]=gamfit(p,0.95); A=[A; num(i),a(1),b(:,1)',a(2),b(:,2)'];
    end
```

由得出的表格可见,当随机数生成中选择的点较少时,随机数参数的估计效果也不理想,所以在生成随机数时不宜生成太少的点,这在前面均值、方差分析与分布函数分析中已经介绍过了。但也不是随机数点选择得越多越好,因为随机数点选得太多,也不会使参数估计显著提高精度,所以在一般计算中选择 30000 个点即可。

9.3.2 多元线性回归与区间估计

假设输出信号 y 为 n 路输入信号 x_1, x_2, \dots, x_n 的线性组合

$$y = a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n \quad (9-3-1)$$

其中, a_1, a_2, \dots, a_n 为待定系数。现在假设已经进行了 m 次实验,由实际测得

$$\begin{aligned} y_1 &= x_{11}a_1 + x_{12}a_2 + \dots + x_{1n}a_n + \varepsilon_1 \\ y_2 &= x_{21}a_1 + x_{22}a_2 + \dots + x_{2n}a_n + \varepsilon_2 \\ &\vdots \\ y_m &= x_{m1}a_1 + x_{m2}a_2 + \dots + x_{mn}a_n + \varepsilon_m \end{aligned} \quad (9-3-2)$$

则可以建立起如下的矩阵方程

$$\mathbf{y} = \mathbf{X}\mathbf{a} + \boldsymbol{\varepsilon} \quad (9-3-3)$$

式中, $\mathbf{a} = [a_1, a_2, \dots, a_n]^T$ 为待定系数向量,因为每次实验的观测数据可能有误差,故不能完全满足式(9-3-1),每一个方程右端均有误差 ε_k ,所以 $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m]^T$ 为误差构成的向量, $\mathbf{y} = [y_1, y_2, \dots, y_m]^T$ 为各个观测值,且 \mathbf{X} 为测出的自变量值,即

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \quad (9-3-4)$$

假设目标函数选择为使得残差的平方和最小,即 $J = \min \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}$,则可以得出线性回归模型的待定系数向量 \mathbf{a} 的最小二乘估计为

$$\hat{\mathbf{a}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (9-3-5)$$

由第4章中介绍的线性代数知识可知, MATLAB 语言可以由下面的语句求出最小二乘解,即 $\mathbf{a} = \text{inv}(\mathbf{X}' * \mathbf{X}) * \mathbf{X}' * \mathbf{y}$,或更简单地, $\mathbf{a} = \mathbf{X} \backslash \mathbf{y}$ 。MATLAB 语言的统计学工具箱还提供了多变量线性回归参数估计与置信区间估计的函数 `regress()`,可以求出所需的估计结果。该函数的调用格式为 $[\hat{\mathbf{a}}, \mathbf{a}_{ci}] = \text{regress}(\mathbf{y}, \mathbf{X}, \alpha)$,其中 $1 - \alpha$ 为用户指定的置信度,可以选择为 0.02、0.05 或其他的值。

例 9-23 假设线性回归方程为 $y = x_1 - 1.232x_2 + 2.23x_3 + 2x_4 + 4x_5 + 3.792x_6$, 试生成 120 组随机输入值 x_i , 计算出输出向量 y 。以这些信息为已知, 观察是否能由最小二乘方法得出待定系数 a_i 的估计值, 并得出置信区间。

解 本例用于演示线性回归的方法及 MATLAB 实现, 实际应用中应该采用实测数据。由下面的语句可以生成所需的矩阵 X 和向量 y , 并用最小二乘计算公式得出待定系数向量 a 的估计为 $a_1 = [1, -1.232, 2.23, 2, 4, 3.792]^T$ 。

```
>> a=[1 -1.232 2.23 2 4, 3.792]'; X=0.01*round(100*randn(120,6));
y=0.0001*round(10000*X*a); [a,aint]=regress(y,X,0.02)
```

可见, 因为输出值完全由精确计算得出, 所以线性回归参数估计的误差为 1.067×10^{-5} , 可以忽略。用 `regress()` 函数还可以计算出 98% 置信度的置信区间分别为

$$a = \begin{bmatrix} 1 \\ -1.232 \\ 2.23 \\ 2 \\ 4 \\ 3.792 \end{bmatrix}, \quad a_{\text{int}} = \begin{bmatrix} 1 & 1 \\ -1.232 & -1.232 \\ 2.23 & 2.23 \\ 2 & 2 \\ 4 & 4 \\ 3.792 & 3.792 \end{bmatrix}$$

```
>> yhat=y+sqrt(0.5)*randn(120,1); [a,aint]=regress(yhat,X,0.02)
errorbar(1:6,a,aint(:,1)-a,aint(:,2)-a)
```

假设观测的输出数据被噪声污染, 则可以给输出样本叠加上 $N(0, 0.5)$ 区间的正态分布噪声, 这时可以用下面语句进行线性回归分析, 得出待定系数向量的估计参数及置信区间, 用 `errorbar()` 函数还可以用图形绘制参数估计的置信区间, 如图 9-13 (a) 所示。新的估计结果为

$$a = \begin{bmatrix} 0.9296 \\ -1.1392 \\ 2.2328 \\ 1.9965 \\ 4.0942 \\ 3.7160 \end{bmatrix}, \quad a_{\text{int}} = \begin{bmatrix} 0.7882 & 1.0709 \\ -1.2976 & -0.9807 \\ 2.0960 & 2.3695 \\ 1.8752 & 2.1178 \\ 3.9494 & 4.2389 \\ 3.5719 & 3.8602 \end{bmatrix}$$

减小噪声的方差, 假设方差为 0.1, 则可以得出新噪声下参数估计的结果, 如图 9-13 (b) 所示。显然估计出的参数更精确。

```
>> yhat=y+sqrt(0.1)*randn(120,1); [a,aint]=regress(yhat,X,0.02);
errorbar(1:6,a,aint(:,1)-a,aint(:,2)-a)
```

9.3.3 非线性函数的最小二乘参数估计与区间估计

假设有一组数据 $x_i, y_i, i = 1, 2, \dots, N$, 且已知这组数据满足某一函数原型 $\hat{y}(x) = f(a, x)$, 其中 a 为待定系数向量, 但由于误差的影响, 可能存在误差, 故该函数应该严格写成 $\hat{y}(x) = f(a, x) + \varepsilon$, 其中 ε 称为残差, 这样可以引入目标函数

$$I = \min_a \sum_{i=1}^N [y_i - \hat{y}(x_i)]^2 = \min_a \sum_{i=1}^N [y_i - f(a, x_i)]^2 \quad (9-3-6)$$

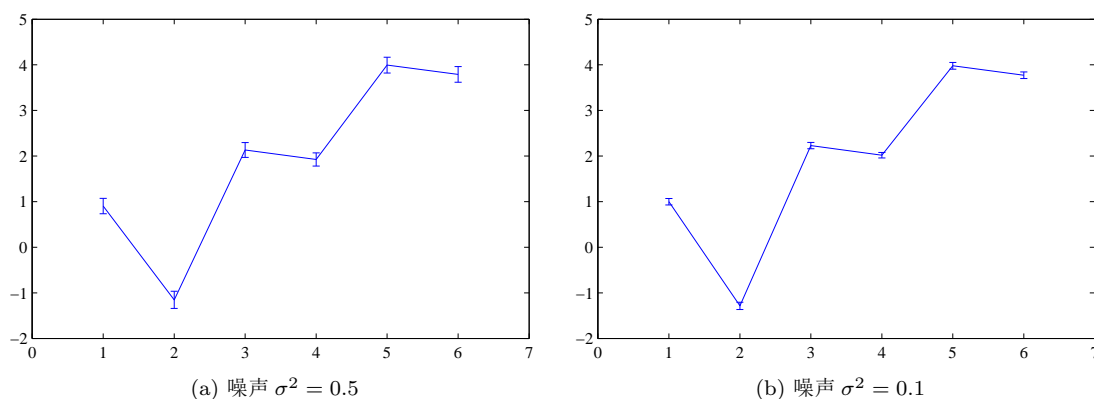


图 9-13 参数估计及置信区间图形表示

用某种数学算法求出使得目标函数最小的参数 \mathbf{a} 。将估计出的 \mathbf{a} 代入原型函数,则可以得出残差 $\varepsilon_i = y_i - f(\mathbf{a}, x_i)$ 。类似于 8.3.3 节中介绍的最小二乘拟合,本节中介绍基于 Gauss-Newton 算法的最小二乘拟合及其 MATLAB 语言 `nlinfit()`。所不同的是,此函数同时还可以求出残差对估计参数向量 \mathbf{a} 的 Jacobi 向量 \mathbf{j}_i 。这些可以用于非线性参数的区间估计函数 `nlparci()`,得出 95% 置信度下的区间估计结果。这些函数的调用格式为

```
[a,r,J] = nlinfit(x,y,fun,a0) % 最小二乘拟合
c = nlparci(a,r,J)           % 由置信度为 95% 的置信区间
```

其中, \mathbf{x} 和 \mathbf{y} 为实测数据; `fun` 为原型函数,可以由 M-函数表示也可以由匿名函数表示; \mathbf{a}_0 为参数估计的初值。可见,该函数调用中的输入参数与 `lsqcurvefit()` 函数完全一致。该参数返回的 \mathbf{a} 向量为估计出的参数, \mathbf{r} 为此参数下的残差构成的向量, \mathbf{J} 为各个 Jacobi 行向量构成的矩阵。得出了这些信息,就可以将其用于置信区间的估计,得出置信度为 95% 的置信区间 \mathbf{c} 。下面将通过例子演示非线性函数参数估计与置信区间估计的问题求解。和前面介绍的 `lsqcurvefit()` 函数一样,该函数同样适用于多变量函数的参数估计与区间估计。

例 9-24 试用参数估计的方法重新求解例 8-24 中给出的最小二乘拟合问题,得出 95% 置信度的置信区间,并在实测信号上叠加均匀分布的噪声信号再进行参数与区间估计。

解 假设原型函数为 $y(x) = a_1 e^{-a_2 x} + a_3 e^{-a_4 x} \sin a_5 x$, 其中 a_i 为待定系数。可以由匿名函数直接描述此原型函数,表示成 $y = f(\mathbf{a}, x)$, 这样就可以人为指定一组 x_i 值并得出相应的 y_i 值,调用 `nlinfit()` 函数就可以得出参数估计,而 `nlparci()` 函数可以获得置信区间,估计的结果为 $\mathbf{a} = [0.1200, 0.2130, 0.5400, 0.1700, 1.2300]^T$, 和理论值完全一致。

```
>> f=@(a,x)a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x).*sin(a(5)*x);
x=0:0.1:10; y=f([0.12,0.213,0.54,0.17,1.23],x); format long
[a,r,j]=nlinfit(x,y,f,[1;1;1;1;1]), ci=nlparci(a,r,j)
```

该函数的拟合结果比 `lsqcurvefit()` 函数的默认控制结果精确得多,但因为本函数不允许给出精度控制选项,所以也不能得出更精确的结果。可见这样得出的置信区间较小,结果比较精确。

现在假设给样本点数据 y_i 叠加上 $[0, 0.02]$ 区间的随机数,则可以给出如下的语句,得出由新样本数据估计出的参数及置信区间分别为

$$a = \begin{bmatrix} 0.12281531581639 \\ 0.17072641296744 \\ 0.55113088779121 \\ 0.17347639675132 \\ 1.2291686258648 \end{bmatrix}, \quad c_i = \begin{bmatrix} 0.11857720435195 & 0.12705342728083 \\ 0.16221631527879 & 0.17923651065609 \\ 0.54465309442893 & 0.55760868115349 \\ 0.17055714192171 & 0.17639565158094 \\ 1.22755955648343 & 1.23077769524618 \end{bmatrix}$$

```
>> y=f([0.12,0.213,0.54,0.17,1.23],x)+0.02*rand(size(x));
[a,r,j]=nlinfit(x,y,f,[1;1;1;1;1]), ci=nlparci(a,r,j)
errorbar(1:5,a,ci(:,1)-a,ci(:,2)-a)
```

这样可以绘制出参数估计及其置信区间,如图 9-14 所示。

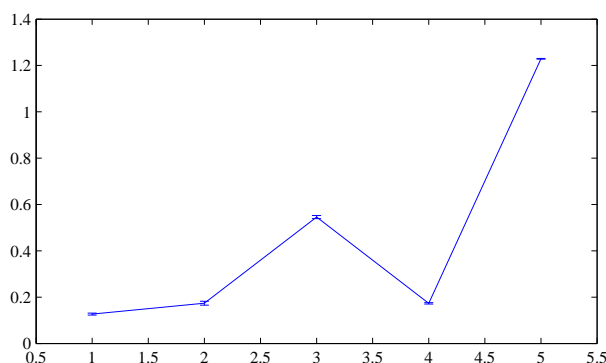


图 9-14 参数估计及置信区间图形表示

例 9-25 试利用 `nlinfit()` 函数求解多变量非线性回归问题。假设非线性函数为

$$f(\mathbf{a}, \mathbf{x}) = (a_1 x_1^3 + a_2) \sin(a_3 x_2 x_3) + (a_4 x_2^3 + a_5 x_2 + a_6)$$

解 假设 a_i 的值均为 1, 可以用下面的语句定义出函数 f , 产生一组数据 \mathbf{X} , 则可以计算出一组输出值作为观测数据。

```
>> a=[1;1;1;1;1;1]'; X=0.01*round(100*rand(120,4)); % 保留两位小数
f=@(a,x)(a(1)*x(:,1).^3+a(2)).*sin(a(3)*x(:,2).*x(:,3))+...
(a(4)*x(:,3).^3+a(5)*x(:,3)+a(6));
y=0.0001*round(10000*f(a,X));
```

由这些观测数据可以用非线性回归参数估计函数求出 a_i 的值, 并绘制出原观测数据与拟合数据, 如图 9-15 (a) 所示, 可见拟合结果比较好。从估计的参数看, 所得出的结果也是较精确的。

```
>> [ahat,r,j]=nlinfit(X,y,f,[0;2;3;2;1;2]); ahat
y1=f(ahat,X); plot([y y1]), ci=nlparci(ahat,r,j)
errorbar(1:6,ahat,ci(:,1)-ahat,ci(:,2)-ahat)
```

得出的估计向量及置信区间分别为

$$\mathbf{a} = \begin{bmatrix} 0.9999 \\ 0.9999 \\ 1.0001 \\ 1 \\ 0.9999 \\ 1 \end{bmatrix}, \quad \mathbf{c}_i = \begin{bmatrix} 0.9997 & 1.0001 \\ 0.9997 & 1.0001 \\ 0.9998 & 1.0003 \\ 1 & 1.0001 \\ 0.9999 & 1 \\ 1 & 1 \end{bmatrix}$$

和前面的例子一样,用 `nlparci()` 函数也能求出置信区间,也可以用图形表示 95% 置信度的置信区间,如图 9-15 (b) 所示。

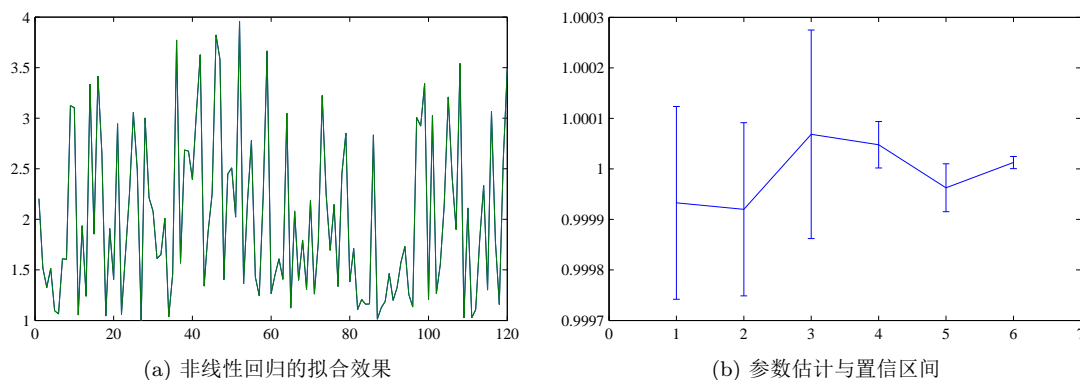


图 9-15 多元非线性回归拟合与参数估计

9.4 统计假设检验

9.4.1 统计假设检验的概念及步骤

先假设总体具有某种统计特征(如具有某种参数或遵从某种分布),然后再检验这个假设是否可信,这种方法称为统计假设检验方法。统计假设检验在统计学中是有重要地位的。例如,有人提出这样的假设,某灯泡厂生产的某种型号的灯泡平均寿命在 3000h 以上,如何检验这个假设是否正确。该方法的确切检验方法,即将所有灯泡使用到烧坏为止显然是没有意义的,而应该采用统计假设检验的方法对该假设进行检验。下面将通过例子来演示统计假设检验实现的步骤。

例 9-26 已知某产品的平均强度为 $\mu_0 = 9.94\text{kg}$, 现在改变制作方法, 并从新产品中随意抽取 200 件, 算得它们的平均强度为 $\bar{x} = 9.73\text{kg}$, 标准差 $s = 1.62\text{kg}$, 问制作方法的改变对强度有无显著影响 [2]?

解 解决这样的问题需要采用统计假设检验, 具体的步骤如下:

(1) 引入两个命题

$$\begin{cases} \mathcal{H}_0: \mu = \mu_0 & \text{即无显著改变} \\ \mathcal{H}_1: \text{拒绝 } \mathcal{H}_0 \text{ 假设} \end{cases}$$

(2) 选取统计量

$$u = \frac{\sqrt{n}(\bar{x} - \mu_0)}{s} \quad (9-4-1)$$

该统计量满足标准正态分布 $N(0, 1)$ 。对本例来说, 可以计算出统计量为 $u = 1.8332$ 。

```
>> n=200; mu0=9.94; xbar=9.73; s=1.62; u=sqrt(n)*(mu0-xbar)/s
```

(3) 给出显著性水平, 由于统计检验毕竟不是确切性检验, 所以无论接受还是拒绝该假设都有可能出现错误。引入 α 的意义是判定出现“取伪”错误的概率。由于研究的是随机问题, 当然不可能令 $\alpha = 0$ 。一般经常取 $\alpha = 5\%$ 或 $\alpha = 2\%$, 用语言表示即为“可以有 95% 或 98% 的把握接受或拒绝该假设”。

(4) 有了 α 值, 则可以用逆正态分布函数求出 $K_{\alpha/2}$ 的值, 使得

$$\int_{-K_{\alpha/2}}^{K_{\alpha/2}} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx < 1 - \alpha \quad (9-4-2)$$

该步骤可以用 MATLAB 求解, 可以得出不同 α 值下的 $K_{\alpha/2}$ 值, 由 (α_i, K_i) 对表示, 分别为 (0.01, 2.5758)、(0.02, 2.3263)、(0.03, 2.1701)、(0.04, 2.0537)、(0.05, 1.96)。

```
>> alpha=[0.01:0.01:0.05]; K=norminv(1-alpha/2,0,1); [alpha' K']
```

(5) 由式 (9-4-1) 计算出统计量 u 的值, 若 $|u| < K_{\alpha/2}$, 则不拒绝 \mathcal{H}_0 假设, 否则拒绝该假设。该步骤用 MATLAB 求解为 `abs(u) < Kα/2`。从比较表达式结果可见, 在 $\alpha = 0.01 \sim 0.05$ 的显著性水平下均可以接受假设 \mathcal{H}_0 , 认为改进方法后无显著变化, 但增大 α 的值, 如这里的 $\alpha = 0.07, 0.09$, 则应该拒绝该假设, 然而这一结论的可靠性将较低, 因为出现错误的概率比较大。

9.4.2 假设检验问题求解

前面的例子介绍了假设检验的 MATLAB 求解。其实, MATLAB 的统计学工具箱中还提供了多个假设检验的函数, 例如正态分布均值的假设检验、正态分布性假设检验和任意分布函数的假设检验等。下面介绍这些检验和 MATLAB 统计学工具箱实现。

1. 正态分布的均值假设检验

已知某组数据符合正态分布规律, 且已知其标准差为 σ 。假设其均值为 μ , 则可以采用 MATLAB 统计学工具箱的 `ztest()` 函数对该假设进行 Z 假设检验。该函数的调用格式为 `[H, s, μ_{ci}] = ztest(X, μ, σ, α)`, 其中, H 为假设检验的结论, 当 $H = 0$ 时表示不拒绝 \mathcal{H}_0 假设, 否则表示拒绝该假设, s 为该检验的显著性水平, μ_{ci} 为其均值的置信区间。

若未知正态分布的标准差, 也可以采用 T 检验法对其进行均值假设检验, 调用 `ttest()` 函数对某正态分布的均值进行检验, 其调用格式为 `[H, s, μ_{ci}] = ttest(X, μ, α)`。

例 9-27 试用正态分布随机数函数生成一组随机数, 并对该随机数进行均值假设检验。

解 假设先由 MATLAB 语句生成一组 400 个 $N(1, 2^2)$ 的正态分布随机数, 由于已知标准差为 2, 可以引入假设 $\mathcal{H}_0: \mu = 1$, 这样可以由下面的 MATLAB 语句进行检验, 得出 $H = 0, p = 0.4359, ci = [0.845, 1.3105]$, 故可以接受该假设。

```
>> r=normrnd(1,2,400,1); [H,p,ci]=ztest(r,1,2,0.02)
```

现在试将假设设置为 $\mathcal{H}_0: \mu = 0.5$, 则可以给出如下语句, 得出 $H = 1, p = 7.51 \times 10^{-9}$, 表示应该拒绝 \mathcal{H}_0 假设。这里得出的置信区间和前面得出的仍然完全一致。

```
>> [H,p,ci]=ztest(r,0.5,2,0.02)
```

若认为标准差未知, 则可以采用 T 检验对假设 $\mathcal{H}_0: \mu = 1$ 进行检验, 假设检验可以由下面

的 MATLAB 语句直接得出,由于得出的 $H = 0$, $p = 0.4517$,故表示可以接受该假设。置信区间为 $c_i = [0.8364, 1.3195]$ 。

```
>> [H,p,ci]=ttest(r,1,0.02)
```

2. 正态分布假设检验

判定某变量是否为正态分布的传统方法是采用正态概率纸的形式实现的,这时的假设 \mathcal{H}_0 表示待检验的分布是正态分布。其实,这样的过程完全可以用计算机来实现。MATLAB 统计学工具箱中提供了 `jbtest()` 和 `lillietest()` 两个函数,分别实现 Jarque-Bera 与 Lilliefors 假设检验算法^[3],可以直接由随机样本判定该分布是否为正态分布。这两个函数的调用格式为

```
[H,s] = jbtest(X,α)      % Jarque-Bera 检验
[H,s] = lillietest(X,α)  % Lilliefors 检验
```

例 9-28 某工厂生产一种白炽灯,其流明为随机变量 ξ ,假设 ξ 满足正态分布 $N(\mu, \sigma^2)$,现从产品中随机抽取 120 个样本,其指标(流明数)在表 9-4 中给出,试检验正态分布的假设是否正确。

表 9-4 白炽灯流明实测数据表(例子及数据来源:文献[4])

216	203	197	208	206	209	206	208	202	203	206	213	218	207	208	202	194	203	213	211
193	213	208	208	204	206	204	206	208	209	213	203	206	207	196	201	208	207	213	208
210	208	211	211	214	220	211	203	216	224	211	209	218	214	219	211	208	221	211	218
218	190	219	211	208	199	214	207	207	214	206	217	214	201	212	213	211	212	216	206
210	216	204	221	208	209	214	214	199	204	211	201	216	211	209	208	209	202	211	207
202	205	206	216	206	213	206	207	200	198	200	202	203	208	216	206	222	213	209	219

解 输入该数据,可以调用现成的 `jbtest()` 函数或 `lillietest()` 函数,均能得出 $H = 0$, $p = 0.7281$,表示可以接受该假设,亦即给出的数据满足正态分布。

```
>> X=[216,203,197,208,206,209,206,208,202,203,206,213,218,207,208,...
      202,194,203,213,211,193,213,208,208,204,206,204,206,208,209,...
      213,203,206,207,196,201,208,207,213,208,210,208,211,211,214,...
      220,211,203,216,224,211,209,218,214,219,211,208,221,211,218,...
      218,190,219,211,208,199,214,207,207,214,206,217,214,201,212,...
      213,211,212,216,206,210,216,204,221,208,209,214,214,199,204,...
      211,201,216,211,209,208,209,202,211,207,202,205,206,216,206,...
      213,206,207,200,198,200,202,203,208,216,206,222,213,209,219];
[H,p]=jbtest(X,0.05)
```

确定了该数据为正态分布数据,则可以直接用前面介绍的正态分布拟合函数 `normfit()` 求解问题,得出该分布的均值为 208.8167,其置信区间为 $[207.6737, 209.9596]$,方差为 6.3232,其置信区间为 $[5.6118, 7.2428]$ 。

```
>> [mu1,sig1,mu_ci,sig_ci]=normfit(X,0.05)
```

例 9-29 试用统计学工具箱生成一组 Γ 分布数据,用现成函数验证其是否为正态分布数据,显然这些数据不是正态分布的,所以假设检验结果应该是 1。

解 给出下面的语句,先用 MATLAB 语句生成一组 Γ 分布的随机数,然后调用 `jbtest()` 函数立即得出结果为 $H = 1, p = 0$,即 \mathcal{H}_0 应该被拒绝。

```
>> r=gamrnd(1,3,400,1); [H,p,c,d]=jbtest(r,0.05)
```

3. 其他分布的 Kolmogorov–Smirnov 检验

前面介绍的 Jarque–Bera 与 Lilliefors 假设检验算法和 MATLAB 函数只能用于检验某分布是否为正态分布,却不能用于其他分布的检验。Kolmogorov–Smirnov 检验是检验任意已知分布函数的一种有效的假设检验算法。MATLAB 的统计学工具箱中提供了 `kstest()` 函数,实现了该算法,其调用格式为 `[H,s]=kstest(X,cdfun,α)`,其中, `cdfun` 为两列的均值,第 1 列为自变量,第 2 列应该为要检验的分布函数在自变量处的值。在构造 `cdfun` 时可以用现成分布函数求取,也可以自己按需要检验的分布函数编写,所以可以用此算法检验是否为任意给定的分布。

例 9-30 试对例 9-29 中生成的随机数进行假设检验。该随机数满足 Γ 分布。

解 首先假设其满足 Γ 分布,则由 `gamfit()` 函数可以得出两个参数 $a = 1.0456, \lambda = 3.2868$ 。

```
>> r=gamrnd(1,3,400,1); alam=gamfit(r)
```

这样就能构造出 Γ 分布的分布函数为 `gamcdf(sort(r),alam(1),alam(2))`。将其代入 `kstest()` 函数,就可以对前面的假设进行检验,得出 $H = 0, p = 0.87724898430408$ 。

```
>> r=sort(r); [H,p]=kstest(r,[r gamcdf(r,alam(1),alam(2))],0.05)
```

由于 $H = 0$,所以可以认为通过假设检验,表明前面生成的数据确实满足 Γ 分布,且其参数为 $\hat{a} = 1.0456, \hat{\lambda} = 3.2868$ (真值为 $a = 1, \lambda = 3$)。

9.5 方差分析与主成分分析

9.5.1 方差分析

方差分析 (analysis of variance, ANOVA) 是英国统计学家兼遗传学家 Ronald Fischer 提出的一种分析方法,在医学研究、科学试验和现代工业质量控制等众多领域有着广泛的应用。方差分析技术是假设检验的拓展。考虑有 N 组样本,假设这些样本的均值是相同的,即作出如下假设

$$\mathcal{H}_0: \mu_1 = \mu_2 = \cdots = \mu_N \quad (9-5-1)$$

如果采用前面介绍的假设检验方法,则需要对这些样本进行两两假设检验,这样非常麻烦和不便,故应该引入新的方差分析方法进行分析与检验。

试验样本的影响方式不同,则采用方差分析方法也不同,一般常用单因子 (one-way)、双因子 (two-way) 和 n 因子 (N -way) 方法。下面将分别介绍各种形式下的方差分析方法及其 MATLAB 实现。

1. 单因子方差分析

顾名思义,单因子方差分析就是指对一些观察来说,只有一个外界因素可能对观测的现象产生影响。假设需要研究 N 种药物对某病症的疗效,可以采用这样的方法。将病人随机地分成 N 组,每组有 m 个病人,这样将每个病人的疗效观测指标(如治愈需要的天数)记作 $y_{i,j}$,其中下标 i 表示第 i 组($i = 1, 2, \dots, N$), j 表示某组内病人的编号($j = 1, 2, \dots, m$)。现在仿照 MATLAB 的冒号表达式记号,记第 i 组的所有病人观测指标为 $y_{i,:}$,或各组的第 j 个病人观测指标为 $y_{:,j}$,则这样得出的表示均为向量。还可以引入平均值的概念,例如用 $\bar{y}_{i,:}$ 表示第 i 组内病人观测指标的平均值,用 $\bar{y}_{:,j}$ 表示所有组内所有病人观测指标的平均值,则可以构造出如表 9-5 所示的标准方差分析表,根据该表格中给出的数据找出所需的规律。

表 9-5 单因子方差分析表

方差来源	平方和	自由度	均 方	F	p 值
因子效应	$SSA = \sum_i n_i \bar{y}_{i,:}^2 - N \bar{y}_{:,,:}^2$	$I - 1$	$MSSA = SSA / (I - 1)$	$MSSA / MSSE$	$p = P(F_{I-1, N-I} > c)$
随机误差	$SSE = \sum_i \sum_k y_{i,k}^2 - \sum_i n_i \bar{y}_{i,:}^2$	$N - I$	$MSSE = SSE / (N - I)$		
和	$SST = \sum_i \sum_k y_{i,k}^2 - N \bar{y}_{:,,:}^2$	$N - 1$			

上面所采用的药物作为分组的依据,称为因子(factor),它们的差异(如这里采用药物的不同)称为因子的水平。因为这里始终将药物作为影响观测指标的因素,故称为单因子分析。这里仍然使用假设检验的方法进行方差分析,假设的 \mathcal{H}_0 为各组的平均观测指标是相同的。表格中比较重要的数值是最后两列, Fisher 分布的值 F 和置信度为 c 时的概率值 p , 概率值可以通过逆分布函数求出。如果得出的概率值 $p < \alpha$, $1 - \alpha$ 为置信度,则应该拒绝假设 \mathcal{H}_0 , 否则不拒绝假设。

MATLAB 的统计学工具箱提供了 `anova1()` 函数,可以用于对给出的数据进行单因子方差分析。该函数的调用格式为 `[p,tab,stats] = anova1(X)`, 其中, X 为需要分析的数据,该数据应该为一个 $m \times n$ 矩阵,每一列对应于随机分配的一个组的测试数据,这样就会返回概率 p , 方差表数据 `tab`, 其内容如表 9-5 所示, `stats` 为统计结果量,为结构体变量,包括每组的均值等信息。该函数还将自动打开两个 MATLAB 图形窗口,一个按表 9-5 的形式显示出该表的内容,另一个图形窗口将显示盒式图。

例 9-31 设有 5 种治疗某病的药物,要比较它们的疗效,假定将 30 个病人随机地分成 5 组,每组 6 人,令每组病人使用同一种药物,并记录病人从使用药物开始到痊愈的时间,如表 9-6 所示,试评价疗效有无显著差异。

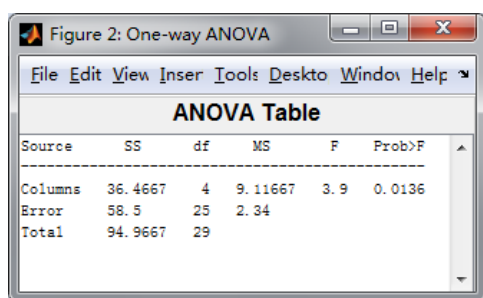
表 9-6 治愈天数的实验数据表(例子及数据来源:文献 [5])

病人编号	药物 1	药物 2	药物 3	药物 4	药物 5	病人编号	药物 1	药物 2	药物 3	药物 4	药物 5
1	5	4	6	7	9	2	8	6	4	4	3
3	7	6	4	6	5	4	7	3	5	6	7
5	10	5	4	3	7	6	8	6	3	5	6

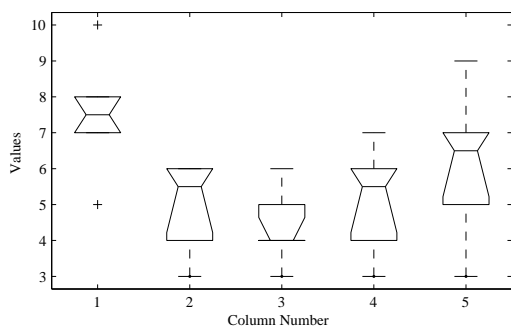
解 根据给出的表格,可以按规则立即建立起 A 矩阵,先求出各列的均值,并对各组数据进行单因子方差分析,得出如下的方差分析结果, $m = [7.5, 5, 4.3333, 5.1667, 6.1667], p = 0.0136$ 。

```
>> A=[5,4,6,7,9; 8,6,4,4,3; 7,6,4,6,5; 7,3,5,6,7; 10,5,4,3,7; 8,6,3,5,6];
m=mean(A), [p,tbl,stats]=anova1(A)
```

同时, `anova1()` 函数还将自动打开两个图形窗口,分别绘制出方差分析表和盒式图,如图 9-16 (a)、(b) 所示。由于得出的概率值 $p = 0.0136 < \alpha$, 其中 $\alpha = 0.02$ 或 0.05 , 故应该拒绝给出的假设,认为这些药物确实对治愈时间有显著影响。该结果和文献中由统计软件 SAS 求出的结果完全一致。事实上,从得出的盒式图可以看出,第 3 种药物的治愈时间显然低于第 1 种药物。



(a) 单因子方差表界面



(b) 盒式图

图 9-16 单因子方差分析结果

2. 双因子方差分析

如果有两种因子可能影响到某现象的统计规律,则应该引入双因子方差分析的概念。这时观测量 y 可以表示为一个三维数组 $y_{i,j,k}$, 表示第 1 个因子取第 i 个水平,第 2 个因子取第 j 个水平时,组内第 k 个对象的观测指标。

根据双因子的特点,可以引入 3 个假设如下:

$$\begin{cases} \mathcal{H}_1: \alpha_1 = \alpha_2 = \cdots = \alpha_I, & \alpha_i \text{ 为第 1 因子单独作用的效应} \\ \mathcal{H}_2: \beta_1 = \beta_2 = \cdots = \beta_J, & \beta_j \text{ 为第 2 因子单独作用的效应} \\ \mathcal{H}_3: \gamma_1 = \gamma_2 = \cdots = \gamma_{IJ}, & \gamma_k \text{ 为两个因子同时作用的效应} \end{cases} \quad (9-5-2)$$

对双因子方差分析问题,可以构造出如表 9-7 所示的分析表格。其中,交互效应的 SSAB 的值可以由式 (9-5-3) 求出。

$$SSAB = K \sum_{ij} \bar{y}_{i,j,:}^2 - JK \sum_i \bar{y}_{i,:,:}^2 - IK \sum_j \bar{y}_{:,j,:}^2 + IJK \bar{y}_{::,:}^2 \quad (9-5-3)$$

另外,3 个概率的定义及意义为

$$\begin{cases} p_A = P(F_{[I-1, IJ(K-1)]} > c_1), & \text{若 } p_A < c_1 \text{ 则拒绝假设 } \mathcal{H}_1 \\ p_B = P(F_{[J-1, IJ(K-1)]} > c_2), & \text{若 } p_B < c_2 \text{ 则拒绝假设 } \mathcal{H}_2 \\ p_{AB} = P(F_{[(I-1)(J-1), IJ(K-1)]} > c_3), & \text{若 } p_{AB} < c_3 \text{ 则拒绝假设 } \mathcal{H}_3 \end{cases} \quad (9-5-4)$$

表 9-7 双因子方差分析表

方差来源	平方和	自由度	均 方	F	p 值
主效应 A	$SSA=JK \sum_i \bar{y}_{i,:}^2 - IJK\bar{y}_{:,,:}^2$	$I - 1$	$MSSA=\frac{SSA}{I - 1}$	$MSSA/MSSE$	p_A
主效应 B	$SSB=IK \sum_j \bar{y}_{:,j}^2 - IJK\bar{y}_{:,,:}^2$	$J - 1$	$MSSB=\frac{SSB}{J - 1}$	$MSSB/MSSE$	p_B
交互效应	SSAB 见式 (9-5-3) 中的定义	$(I - 1)(J - 1)$	$MSSAB=\frac{SSAB}{(I - 1)(J - 1)}$	$MSSAB/MSSE$	p_{AB}
随机误差	$SSE=\sum_{ijk} y_{i,j,k}^2 - K \sum_i \sum_j \bar{y}_{i,j,:}^2$	$IJ(K - 1)$	$MSSE=\frac{SSE}{IJ(K - 1)}$		
和	$SST=\sum_{ijk} y_{i,j,k}^2 - IJK\bar{y}_{:,,:}^2$	$IJK - 1$			

求解双因子方差分析问题的 MATLAB 统计学工具箱函数为 `anova2()`, 其调用格式与单因子方差分析函数 `anova1()` 很相近, 为 `[p,tab,stats] = anova2(X)`。

例 9-32 为比较 3 种松树在 4 个不同地区的生长情况有无差别, 在每个地区对每种松树随机地选择 5 株, 测量它们的胸径, 得出的数据在表 9-8 中给出 (第 3 种树在第 4 地区的第 1 个数值 16, 原数据为 18, 但和后面分析结果对不上, 故改), 试判定树种或地区对松树的生长有无影响。

表 9-8 松树数据 (例子及数据来源: 文献 [5])

松树种类	地 区																			
	1					2					3					4				
1	23	15	26	13	21	25	20	21	16	18	21	17	16	24	27	14	17	19	20	24
2	28	22	25	19	26	30	26	26	20	28	19	24	19	25	29	17	21	18	26	23
3	18	10	12	22	13	15	21	22	14	12	23	25	19	13	22	16	12	23	22	19

解 因为要分析树种和地区两个因素对松树生长的影响, 所以需要采用双因子方差分析方法。按下面的方式将表中数据输入到 MATLAB 环境, 然后调用 `anova2()` 函数, 得出如图 9-17 所示的方差分析表格, 该表格与文献 [5] 中的结果完全一致。

```
>> B=[23,15,26,13,21,25,20,21,16,18,21,17,16,24,27,14,17,19,20,24;  
      28,22,25,19,26,30,26,26,20,28,19,24,19,25,29,17,21,18,26,23;  
      18,10,12,22,13,15,21,22,14,12,23,25,19,13,22,16,12,23,22,19];  
anova2(B',5);
```

ANOVA Table					
Source	SS	df	MS	F	Prob>F
Columns	355.6	2	177.8	9.68	0.0003
Rows	49.65	3	16.55	0.9	0.4478
Interaction	106.4	6	17.733	0.97	0.4588
Error	882	48	18.375		
Total	1393.65	59			

图 9-17 双因子方差分析表格

从得出的结果看,由于 p_A 的值很小,所以应该拒绝 \mathcal{H}_1 假设。可以认为, A 因子对观测现象有显著影响,得出结论为树种对观测树的胸径有显著影响。

```
>> C=[]; for i=1:3, for j=1:4, C(i,j)=mean(B(i,[1:5]+(j-1)*5)); end, end
      C=[C; mean(C)]; C=[C mean(C)']
```

计算出的各个均值为

$$C = \begin{bmatrix} 19.6 & 20 & 21 & 18.8 & 19.85 \\ 24 & 26 & 23.2 & 21 & 23.55 \\ 15 & 16.8 & 20.4 & 18.4 & 17.65 \\ 19.533 & 20.933 & 21.533 & 19.4 & 20.35 \end{bmatrix}$$

可见,树种 2 的树胸径最大,树种 3 的最小(见 C 矩阵的各行)。由于另外两个概率 p_B 和 p_{AB} 的值很大,所以没有理由拒绝另外两个假设。故得出结论:地区对树的胸径无显著影响,不同区域对不同树种的胸径观测结果也无显著影响。

3. 多因子方差分析

类似于前面介绍的双因子方差分析,用 MATLAB 语言的统计学工具箱还可以进行三因子甚至多因子的方差分析,可以采用 `manova1()` 函数进行多因子方差分析,这里不再介绍该分析。

9.5.2 主成分分析

主成分分析(principal components analysis, PCA)是现代统计分析中的一种有效方法。假设某一个现象受多个因素同时影响,则可以考虑采用主成分方法,由大量实测数据中识别出到底哪些因素对其发生起主要的作用,通过这样的方法可以忽略掉次要的因素,将原来问题的维数降下来,从而简化原来问题的分析。

假设某一事件的发生可能受 n 个因素 x_1, x_2, \dots, x_n 影响,而实测数据共有 M 组,这样可以假设这些数据由一个 $M \times n$ 矩阵 \mathbf{X} 表示。记该矩阵的每一列的均值为 \bar{x}_i , $i = 1, 2, \dots, n$,则主成分分析方法的一般步骤为:

(1) 调用 `corr()` 函数,由矩阵 \mathbf{X} 可以建立起 $n \times n$ 协方差矩阵 \mathbf{R} ,使得

$$r_{ij} = \frac{\sqrt{\sum_{k=1}^M (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)}}{\sqrt{\sum_{k=1}^M (x_{ki} - \bar{x}_i)^2 \sum_{k=1}^M (x_{kj} - \bar{x}_j)^2}} \quad (9-5-5)$$

(2) 由 \mathbf{R} 矩阵可以分别得出特征向量 \mathbf{e}_i 和对应的排序特征值 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$,特征向量矩阵的每一列也都进行了相应的归一化,即 $\|\mathbf{e}_i\| = 1$ 或 $\sum_{j=1}^n e_{ij}^2 = 1$ 。这样的运算可以通过 `eig()` 直接获得,然而得出的特征值是按照升序排列的,应该反序,所以需要用到 `fliplr()` 函数处理特征向量矩阵。

(3) 计算如下定义的主成分贡献率和累计贡献率

$$\text{主成分贡献率: } \gamma_i = \frac{\lambda_i}{\sum_{k=1}^n \lambda_k}, \quad \text{累计贡献率: } \delta_i = \frac{\sum_{k=1}^i \lambda_k}{\sum_{k=1}^n \lambda_k} \quad (9-5-6)$$

如果前 s 个特征值的累计贡献率大于某个预期的指标, 如 85%~95%, 则可以认为这 s 个因素是原问题的主成分, 这时, 原来的 n 维问题就可以简化成 s 维问题了。

(4) 建立新变量指标 $\mathbf{Z} = \mathbf{XL}$, 即

$$\begin{cases} z_1 = l_{11}x_1 + l_{21}x_2 + \cdots + l_{n1}x_n \\ z_2 = l_{12}x_1 + l_{22}x_2 + \cdots + l_{n2}x_n \\ \vdots \\ z_n = l_{1n}x_1 + l_{2n}x_2 + \cdots + l_{nn}x_n \end{cases} \quad (9-5-7)$$

其中变换矩阵第 i 列的系数 l_{ji} 可以如下计算 $l_{ji} = \sqrt{\lambda_i} e_{ji}$ 。这时, 主成分分析方法可以由得出的矩阵系数 l_{ij} 直接分析。通常情况下, 如果取前 s 个成分作主成分, 则 \mathbf{L} 矩阵的 s 列以后各值应该趋于 0, 这样, 式 (9-5-7) 中后 $n-s$ 个 z 变量就可以忽略, 由一组 m 个状态变换后的新变量

$$\begin{cases} z_1 = l_{11}x_1 + l_{21}x_2 + \cdots + l_{n1}x_n \\ \vdots \\ z_s = l_{1s}x_1 + l_{2s}x_2 + \cdots + l_{ns}x_n \end{cases} \quad (9-5-8)$$

就可以表示原问题, 即在适当的线性变换下, 原来的 n 维问题就可以简化成 s 维问题。

假设已知某物理量受若干个因素影响, 而这些因素的值可以由传感器测出。但在实验研究中, 往往这些传感器测出的量包含冗余信息, 可以通过主成分分析的方法构造出一组新的数据, 将高维的问题简化成低维问题。下面将通过例子介绍主成分分析方法及应用。

例 9-33 假设某三维曲线上的样本点由 $x = t \cos 2t, y = t \sin 2t, z = 0.2x + 0.6y$ 直接生成, 试用主成分分析的方法对其降维处理。

解 可以由 MATLAB 语句生成一组数据, 并将结果用三维曲线表示出来, 如图 9-18(a) 所示。

```
>> t=[0:0.1:3*pi]'; x=t.*cos(2*t); y=t.*sin(2*t); z=0.2*x+0.6*y;
X=[x y z]; R=corr(X); [e,d]=eig(R), d=diag(d), plot3(x,y,z)
```

从得出的图形上可以看出, 该曲线应该位于某一个平面上, 所以可以考虑引入坐标变换, 将原来的三维图形压缩在某一个二维的平面上。由上述语句得出的结果为

$$\mathbf{R} = \begin{bmatrix} 1 & -0.0789 & 0.2536 \\ -0.0789 & 1 & 0.9443 \\ 0.2536 & 0.9443 & 1 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 0.2306 & -0.9641 & 0.1314 \\ 0.6776 & 0.256 & 0.6894 \\ -0.6983 & -0.0699 & 0.7124 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0 \\ 1.0393 \\ 1.9607 \end{bmatrix}$$

可见, 这样得出的 \mathbf{d} 向量是按照升序排列的, 而不是期望的按照降序排列的, 所以应该对其进行反序处理, 同时对 \mathbf{e} 矩阵进行左右翻转, 并最终得出 \mathbf{L} 矩阵。由于前两个特征值的值较大, 第 3 个特征值趋于 0, 可见, 保留两个变量即可以有效地研究原始问题。由下面的语句

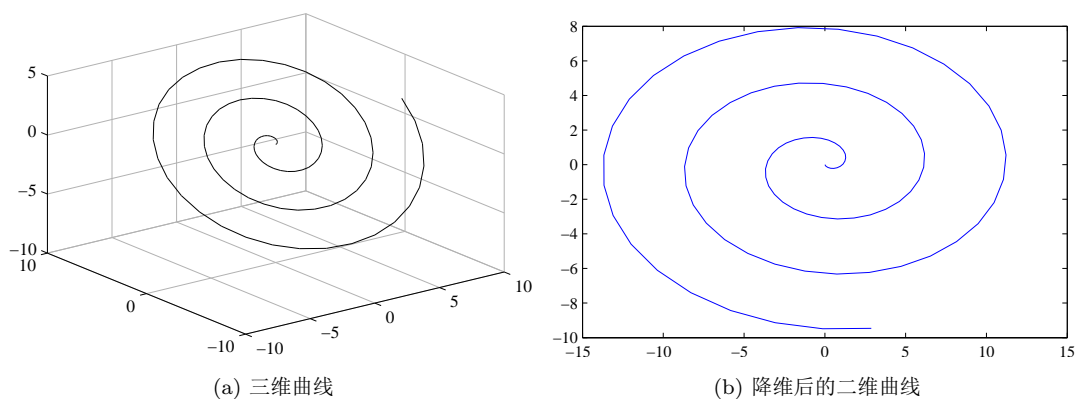


图 9-18 三维曲线及主成分分析降维效果

```
>> d=d(end:-1:1); e=flipplr(e); D=[d'; d'; d']; L=real(sqrt(D)).*e
      Z=X*L; figure; plot(Z(:,1),Z(:,2))
```

可以得出

$$L = \begin{bmatrix} 0.184 & -0.9829 & 0 \\ 0.9653 & 0.261 & 0 \\ 0.9975 & -0.0713 & 0 \end{bmatrix}$$

即引入新坐标系

$$\begin{cases} z_1 = 0.1840x + 0.9653y + 0.9975z \\ z_2 = -0.9829x + 0.2610y - 0.0713z \end{cases}$$

就可以将原三维问题降为二维问题。降维后的二维曲线如图 9-18 (b) 所示,可见,这样得出的二维图形可以将原三维空间上的一个平面提取出来,该平面包含原图的全部信息。

9.6 习 题

- 假设已知 Rayleigh 分布的概率密度函数为 $p_r(x) = \begin{cases} \frac{x}{b^2} e^{-x^2/(2b^2)}, & x \geq 0 \\ 0, & x < 0 \end{cases}$, 试用解析推导的方法求出该分布的分布函数、均值、方差、中心矩和原点矩。生成一组满足 Rayleigh 分布的伪随机数,用数值方法检验得出的解析结果是否正确。
- 某次外语考试抽样调查结果表明,学生外语考试成绩近似服从正态分布,且其均值为 72 分,并已知超过 96 分的人数占总数的 2.3%,试求出考生外语成绩介于 60 与 80 之间的概率。
- 试生成满足正态分布 $N(0.5, 1.4^2)$ 的 30000 个伪随机数,对其均值和方差进行验证,并用直方图的方式观察其分布与理论值是否吻合。若改变直方图区间的宽度会得出什么结论?
- 假设通过实验测出某组数据如下表,试用 MATLAB 对这些数据进行检验。
 - 若认为该数据满足正态分布,且标准差为 1.5,请检验该均值为 0.5 的假设是否成立。
 - 若未知其方差,试再检验其均值为 0.5 的假设是否成立。
 - 试对给出数据的正态性进行检验。

-1.7908	1.5803	1.5924	2.7278	-0.7177	-1.8152	2.8943	0.4704	-1.5161	0.7403	2.3831	2.3258
0.0903	2.0033	0.4887	0.9925	-2.5004	1.047	-0.0521	-0.8056	0.8041	4.6585	-1.1251	1.9318
3.9223	0.3238	-0.1215	1.0887	2.9135	-2.3273	-2.9145	5.3067	0.1872	-0.1190	-1.1234	3.4477
0.41351	2.5006	3.372	3.2303	-1.1022	-0.2812	0.5219	-0.0796	-2.1176	5.4782	0.0473	1.236
3.2618	5.6959	4.6927	-0.1180	0.4746	-1.6181	0.6606	-2.6714	3.1634	3.8942	0.4540	-1.0142
-1.0665	1.6804	-0.6758	0.2005	0.4982	-2.1428	1.2122	4.4827	0.4653	-3.8764	1.1275	0.1640
0.5169	0.4735	0.7327	-2.3586	-0.0612	-1.7976	1.6246	1.2325	1.7065	-3.2812	2.8812	-5.0103
-1.2615	2.5546	-1.3172	-3.2431	1.3923	-0.4038	3.3757	-2.0178	-1.112	-0.7905	1.8988	1.5649
1.8206	0.6259	2.031	-1.083	-0.0940	0.8908	-0.7326	1.8958	-0.9750	0.0819	-3.4389	0.7631
-0.0652	-1.9909	-4.8203	1.132	-3.2440	0.2387	1.0868	3.357	1.2073	0.5201	2.0690	-0.8300

- 某研究者对随机抽取的一组保险丝进行了实验,测出使保险丝烧断的电流值为 10.4、10.2、12.0、11.3、10.7、10.6、10.9、10.8、10.2、12.1 A,假设这些值满足正态分布,试在置信水平 $\alpha \leq 0.05$ 的条件下求出这些保险丝的熔断电流及其置信区间。
- 假设在某固定气压下对水的沸点进行多次测试,得出一组数据为 113.53、120.25、106.02、101.05、116.46、110.33、103.95、109.29、93.93、118.67°C,并假设它们满足正态分布,试求出置信水平 $\alpha \leq 0.05$ 的条件下,该气压下水沸点的总体方差的置信区间。
- 甲、乙两位化验员独立地对某种聚合物的含氮量用相同的方法各取了 10 次测定,其测定值的修正样本方差 S_1^{*2} 、 S_2^{*2} 依次为 0.5419、0.6050,求总体方差比 σ_1^2/σ_2^2 置信度为 0.90 的置信区间。假定测定值总体服从正态分布。
- 假设测出某随机变量的 12 个样本为 9.78、9.17、10.06、10.14、9.43、10.60、10.59、9.98、10.16、10.09、9.91、10.36,试求其方差及方差的置信区间。
- 10 个失眠者服用 A、B 两种药后,延长睡眠时间由下表给出,试判定两种药物对失眠的疗效有无显著差异。

A	1.9	0.8	1.1	0.1	-0.1	4.4	5.5	1.6	4.6	3.4
B	0.7	-1.6	-0.2	-1.2	-0.1	3.4	3.7	0.8	0	2

- 假设两个随机变量 A、B 的样本点如下,试判定二者是否有显著差异。

A	10.42	10.48	7.98	8.52	12.16	9.74	10.78	10.18	8.73	8.88	10.89	8.1
B	12.94	12.68	11.01	11.68	10.57	9.36	13.18	11.38	12.39	12.28	12.03	10.8

- 假设测出一组输入值 x_1, x_2, x_3, x_4, x_5 和输出值 y 如下表,且已知 $y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5$,试用线性回归方法估计出 a_i 的值及其置信区间。

x_1	8.11	9.25	7.63	7.89	12.94	10.11	7.57	9.92	7.74	7.3	9.48	11.91
x_2	2.13	2.66	0.83	1.54	1.74	0.79	0.68	2.93	2.01	1.35	2.81	2.23
x_3	-3.98	0.68	-1.42	0.96	0.28	-3.37	-4.58	-2.15	-2.66	-3.69	-1	0.98
x_4	-6.55	-6.85	-6.25	-5.34	-6.85	-7.2	-6.12	-6.07	-5.51	-6.6	-6.15	-6.43
x_5	5.92	7.54	5.39	4.65	6.47	5.1	6.04	5.37	6.54	6.55	5.8	3.95
y	27.676	38.774	23.314	23.828	35.154	21.779	25.516	29.845	32.642	28.443	31.5	23.554

12. 假设测出一组输入值 x_i 和输出值 y_i 如下表, 且已知原型函数为 $f(x) = a_1 e^{-a_2 x} \cos(a_3 x + \pi/3) + a_4 e^{-a_5 x} \cos(a_6 x + \pi/4)$, 试估计出 a_i 的值及其置信区间。

x	1.027	1.319	1.204	0.684	0.984	0.864	0.795	0.753	1.058	0.914	1.011	0.926
y	-8.880	-5.964	-7.106	-8.691	-9.251	-9.922	-9.890	-9.636	-8.588	-9.728	-9.023	-9.661

13. 假设测出一组输入值 x_1, x_2, x_3, x_4, x_5 和输出值 y 如下表, 且已知 $y = e^{-a_1 x_1} \sin(a_2 x_2 + a_3 x_3) + x_4^3 \cos(a_4 x_5)$, 试估计出 a_i 的值及其置信区间。

x_1	8.11	9.25	7.63	7.89	12.94	10.11	7.57	9.92	7.74	7.3	9.48	11.91
x_2	2.13	2.66	0.83	1.54	1.74	0.79	0.68	2.93	2.01	1.35	2.81	2.23
x_3	-3.98	0.68	-1.42	0.96	0.28	-3.37	-4.58	-2.15	-2.66	-3.69	-1	0.98
x_4	-6.55	-6.85	-6.25	-5.34	-6.85	-7.2	-6.12	-6.07	-5.51	-6.6	-6.15	-6.43
x_5	5.92	7.54	5.39	4.65	6.47	5.1	6.04	5.37	6.54	6.55	5.8	3.95
y	22.126	250.16	-144.11	-152.07	234.09	-318.04	54.401	-136.8	132.03	229.26	-19.048	-145.83

14. 假设可以通过实验测出如下表所示的数据, 且假设这些数据满足 $y(t) = c_1 e^{-5t} \sin(c_2 t) + (c_3 t^2 + c_4 t^3) e^{-3t}$, 试根据这些数据求出 c_i 参数的估计值与置信区间。

-0.2163	0.1201	1.8787	2.7393	2.7238	4.5219	5.0833	4.9699	5.5947	5.9073	6.0663	6.8166
6.4115	8.0106	7.0286	7.2988	7.8903	7.4742	7.4594	7.1308	7.7132	6.8981	7.9065	8.3289
7.1251	7.8416	7.9701	6.4669	6.4553	7.3657	6.7779	7.2148	7.1647	6.9958	7.1645	6.7303
6.8659	5.5421	6.005	5.8074	4.9543	5.7555	4.9696	6.077	4.8393	5.3799	5.1003	4.4062
3.6602	4.5961	4.0026	4.6994	4.5325	5.0136	4.3541	3.6301	4.0379	3.2414	3.637	3.5258
3.4556	3.2048	3.8218	2.2502	3.3167	3.4682	3.306	3.1518	2.8077	3.053	2.928	2.447
2.3194	2.2955	1.6433	2.2031	2.3206	2.3618	2.871	1.9203	2.3557	2.3935	2.4159	1.4025
1.9591	1.928	1.2625	1.3541	2.2263	1.5807	1.8039	1.6166	1.2197	1.2236	1.6922	0.9634
1.7978	1.6616	0.9371	1.1868	0.6982	0.1643	1.7327	0.9551	1.3536	1.2832	1.1538	0.6187
0.6252	0.8904	0.4639	0.5088	1.7534	1.0259	0.3708	0.9407	0.3794	0.2517	0.7789	1.3697
0.9413	1.1895	0.2620	0.6006	0.6850	0.1953	0.1281	1.2397	0.7663	0.4249	1.1374	0.8377
0.2146	1.3671	0.8302	1.4132	1.2313	0.3089	-0.0061	0.5926	0.4531	0.1861	0.8465	1.3317
0.3043	0.1444	0.435	0.88024	0.1123	-0.0704	0.4614	0.4798	0.7464	0.1975	0.41194	0.2611
0.4307	1.2299	0.1511	-0.2271	0.6736	-0.0204	0.4419	0.1029	0.6771	0.6788	0.2935	-0.6387
0.4480	1.1715	0.8777	-0.4282	0.3163	0.00854	-0.1691	-0.2801	0.4755	0.1106	0.3473	0.1298
0.0756	0.4880	0.6612	1.3478	-0.3922	-0.2301	0.7950	0.0762	-0.4245	0.4190	1.0331	0.6057

15. 设从 A、B 两个不同的地区各取得某种植物的样品 12 个, 测得植物中铁元素含量 ($\mu\text{g/g}$) 的数据如下表, 假定已经知道这种植物中铁元素含量为正态分布, 且分布的方差是不受地区影响的, 检验这两个地区该种植物中铁元素含量的分布是否相同。

地区 A	11.5	18.6	7.6	18.2	11.4	16.5	19.2	10.1	11.2	9	14	15.3
地区 B	16.2	15.2	12.3	9.7	10.2	19.5	17	12	18	9	19	10

16. 一批由同种原料织成的布, 用不同的染整工艺处理, 每台进行缩水率试验, 目的是考察不同的工艺对布的缩水率是否有显著影响。现采用 5 种不同的染整工艺, 每种工艺处理 4 块布样, 测得缩

水率的百分数见下表。试判定染整工艺对缩水率有无显著影响。

布样		染整工艺数据					布样		染整工艺数据				
1	4.3	6.1	6.5	9.3	9.5		2	7.8	7.3	8.3	8.7	8.8	
3	3.2	4.2	8.6	7.2	11.4		4	6.5	4.2	8.2	10.1	7.8	

17. 抽查某地区 3 所小学五年级男学生的身高由下表给出,问该地区这 3 所小学五年级男学生的平均身高是否有显著差别 ($\alpha = 0.05$)?

学校	实测身高数据					
1	128.1	134.1	133.1	138.9	140.8	127.4
2	150.3	147.9	136.8	126	150.7	155.8
3	140.6	143.1	144.5	143.7	148.5	146.4

18. 下表记录了 3 位操作工分别在 4 台不同机器上操作的日产量,试检验
- (1) 操作工之间的差异是否显著?
- (2) 机器之间的差异是否显著?
- (3) 交互作用是否显著 ($\alpha = 0.05$)?

机 器	操 作 工									机 器	操 作 工								
	1			2			3				1			2			3		
M_1	15	15	17	19	19	16	16	18	21	M_3	15	17	16	18	17	16	18	18	18
M_2	17	17	17	15	15	15	19	22	22	M_4	18	20	22	15	16	17	17	17	17

参考文献

[1] Landau D P, Binder K. A guide to Monte Carlo simulations in statistical physics. Cambridge, MA: Cambridge University Press, 2000

[2] 《数学手册》编写组. 数学手册. 北京:人民教育出版社,1979

[3] Conover W J. Practical nonparametric statistics. New York: Wiley, 1980

[4] 中山大学数学力学系. 概率论与数理统计. 北京:人民教育出版社,1980

[5] 陆璇. 应用统计. 北京:清华大学出版社,1999

[6] 赵选民,师义民. 概率论与数理统计典型题分析解集. 西安:西北工业大学出版社,1999

第 10 章 数学问题的非传统解法

前面各章系统介绍了高等应用数学各个领域的数学问题计算机辅助求解方法。近几十年来,科学家们仿照人类思维方式或其他自然科学的研究成果,发展出了很多新的分支,用来解决数学和其他应用科学领域的问题。例如,仿照人类思维和语言规则提出的模糊逻辑和模糊推理,仿照生物神经网络提出的人工神经网络,仿照生物遗传学及进化过程的“适者生存”规律提出的遗传算法和进化理论等。这些理论在自动控制学科及其他科学与工程领域均有很好的应用前景。在本章 10.1 节中将首先介绍经典集合论问题的 MATLAB 语言求解方法,然后引入模糊集合的概念并介绍基于 MATLAB 语言的模糊集合与模糊推理的实现方法。10.2 节引入粗糙集的概念并介绍粗糙集的基本理论,然后介绍其在条件约简等领域的应用。10.3 节引入人工神经网络的数学表示及前馈式神经网络结构,介绍利用 MATLAB 语言神经网络结构设置、训练及网络泛化的全过程,利用 MATLAB 神经网络工具箱直接求解数据拟合问题的方法。10.4 节实现引入遗传算法、粒子群算法等基本概念和解题步骤,介绍其在无约束最优化与有约束最优化问题中的应用,并通过例子介绍利用 MATLAB 语言现成的工具求解最优化问题的方法。10.5 节介绍小波理论和小波分析概述,并介绍如何用 MATLAB 语言的小波工具箱求解噪声滤波等。10.6 节还将深入介绍分数阶微积分问题的求解方法。本章简要介绍这些理论的基本概念,但均侧重于介绍用 MATLAB 语言或相应的工具箱如何求解这些问题的方法。

10.1 集合论、模糊集与模糊推理

10.1.1 经典可枚举集合论问题及 MATLAB 求解

集合论是现代数学的基础。所谓集合,就是一些事物的全体,而其中每一个事物均称为集合中的一个元素。若事物 a 是集合 A 中的一个元素,则记 $a \in A$,称为 a 属于 A 。若 b 不是 A 集合中的元素,则记 $b \notin A$ 。所谓可枚举集合,就是该集合中的所有元素均可以一一列出的集合。在 MATLAB 中用向量或单元数组的形式就可以表示这样的集合。例如,下面的语句均可以表示集合,集合定义中可以使用重复元素。

```
>> A=[1 2 3 5 6 7 9 3 4 11], B=1 2 3 5 6 7 9 3 4 11 % 两种方法均可表示集合  
C='ssa','jsjhs','su','whi','kjshd','kshk' % 字符串集合,可以为人名等
```

MATLAB 语言提供了集合定义与基本运算函数。在表 10-1 中列出了进行集合运算的函数及解释,用这些函数可以对集合进行操作,这些函数还可以嵌套使用,建立较复杂的集合运算。遗憾的是,这些函数不能用于符号表达式的集合运算。

例 10-1 假设给定 3 个集合 $A = \{1, 4, 5, 8, 7, 3\}$, $B = \{2, 4, 6, 8, 10\}$, $C = \{1, 7, 4, 2, 7, 9, 8\}$, 试演示集合的各种运算,并验证这些集合满足分配律 $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ 。

表 10-1 MATLAB 下集合运算的函数

运算名称	MATLAB 语句	集合运算描述
并集运算	<code>A = union(B,C)</code>	求两个集合 B 、 C 的并集,数学记号为 $A = B \cup C$,集合运算后的结果重新排序
差集运算	<code>A = setdiff(B,C)</code>	求两个集合 B 和 C 的差集,记作 $A = B \setminus C$,亦即从集合 B 中剔除 C 中的元素剩下的元素,结果被重新排序
交集运算	<code>A = intersect(B,C)</code>	求两个集合 B 和 C 的交集,即 $A = B \cap C$,重新排序
异或运算	<code>A = setxor(B,C)</code>	求两个集合 B 和 C 的异或运算,即从 $B \cup C$ 中剔除 $B \cap C$,数学表示为 $A = (B \cup C) \setminus (B \cap C)$,结果被重新排序
唯一运算	<code>A = unique(B)</code>	将 B 集合中的重复元素剔除,得出的是唯一的元素集合,结果被排序
属于判定	<code>key = ismember(a,B)</code>	判定 a 是否为 B 集合中的元素,如果是则返回 <code>key</code> 值为 1,否则返回 0,记作 $\text{key}=a \in B$ 。其实,在属于关系中, a 也可以为矩阵,这时返回的 <code>key</code> 为和 a 一样维数的矩阵,在满足属于关系的元素处为 1,否则为 0

解 由给出的条件可以立即输入已知的 A 、 B 、 C 这 3 个集合,然后调用集合运算的命令即可以得到 $D = [1, 2, 4, 7, 8, 9]$, $E = [1, 2, 3, 4, 5, 6, 7, 8, 10]$, $F = [4, 8]$,这些结果应该不难理解。

```
>> A=[1,4,5,8,7,3]; B=[2,4,6,8,10]; C=[1,7,4,2,7,9,8]; % 集合定义
D=unique(C) % 求解唯一运算,可见从 C 中剔除了重复的 7
E=union(A,B), F=intersect(A,B) % 求出并集和交集
```

给出如下命令,则可以发现分配律左侧的集合与右侧的集合求差集,得出的结果为空集,由此验证了分配律的正确性。

```
>> G=setdiff(intersect(union(A,B),C),union(intersect(A,C),intersect(B,C)))
```

现在可以演示 `ismember()` 函数在集合运算中的应用,由语句 `E = ismember(A,B)` 可以得到 $E = [0, 1, 0, 1, 0, 0]$,表明 A 集合中的第 2 和第 4 元素属于 B 集合,因为这些位置处测试结果的价值为 1。所以,可以用语句 `G = A(ismember(A,B))` 提取出 A 集合中属于 B 的元素,即 $G = [4, 8]$ 。

例 10-2 假设 A 集合为字符串组 $\{\text{'skhsak'}, \text{'ssd'}, \text{'ssfa'}\}$, B 集合为 $\{\text{'sdsd'}, \text{'ssd'}, \text{'sssf'}\}$,试求它们的并集与交集,令 $C = \{\text{'jsg'}, \text{'sjjfs'}, \text{'ssd'}\}$,试验证分配律

$$(A \cap B) \cup (C \cap B) = (A \cup C) \cap B$$

解 字符串构成的集合可以用单元数组的形式表示,也可以进行集合运算,所以直接用下面的语句求出它们的并集为 $F = \{\text{'sdsd'}, \text{'skhsak'}, \text{'ssd'}, \text{'ssf'}, \text{'sssf'}\}$,交集为 $D = \{\text{'ssd'}\}$,且 E 为空集。

```
>> A='skhsak','ssd','ssfa'; B='sdsd','ssd','sssf'; F=union(A,B) % 并集
D=intersect(A,B) % 交集
C='jsg','sjjfs','ssd'; % 可以由下面的集合运算验证分配律
E=setdiff(union(intersect(A,B),intersect(C,B)),intersect(union(A,C),B))
```

子集与集合包含等概念是集合论中很重要的概念。所谓集合包含即集合 A 中所有的元素均为集合 B 的元素,记作 $A \subseteq B$,称为 B 包含 A ,又称 A 是 B 的子集。若 $B \setminus A$ 非空,则称严格包含,记作 $A \subset B$ 。MATLAB 中并未直接提供集合包含或子集的函数,但可以通过

下面的命令判定包含和严格包含。

```
key = all(ismember(A,B)), % key=1 则  $A \subseteq B$ , 即判定  $A$  所有元素均为  $B$  元素
key = all(ismember(A,B)) & (length(setdiff(B,A))>0), % key=1 则  $A \subset B$ 
```

例 10-3 考虑例 10-1 中的 E, F 集合, 试判定 $F \subset E$ 是否满足, 并由 A 集合验证集合的自反律, 亦即 $A \subseteq A$ 。

解 可以用下面的语句进行判定, 得出 $\text{key}=1$ 。

```
>> A=[1,4,5,8,7,3]; B=[2,4,6,8,10];
E=union(A,B); F=intersect(A,B); key=all(ismember(F,E))
```

事实上, $F = A \cap B$, $E = A \cup B$, 所以当然 $E \subset F$ 。还可以验证 $A \subseteq A$, 亦即自反律。

```
>> key=all(ismember(A,A)) & (length(setdiff(A,A))>0); %  $A \not\subset A$ 
key1=all(ismember(A,A)); [key,key1] %  $A \subseteq A$  当然成立
```

例 10-4 Goldbach 猜想是尚未严格证明的最古老的数论问题。该猜想为: 任何大于 2 的偶数均能分解为两个质数的和。试用集合运算的方法验证小于 2000 的偶数均满足该猜想。

解 对有限偶数来说, 可以由某范围内的两质数所有的可能的和构造出一个集合, 然后判定是否有限偶数均属于该集合, 如果不属于该集合的偶数为空集, 则可以得出结论: 测试的偶数均满足 Goldbach 猜想。根据上述思路, 可以给出下面的 MATLAB 语句

```
>> iA=1:1040; iA=iA(isprime(iA)); c=[]; for i=iA, c=[c i+iA]; end
c=unique(c); c1=4:2:2000; c2=ismember(c1,c); key=c1(c2==0)
```

可见, 这样得出的集合 key 为空集, 故可以得出结论, $[4, 2000]$ 的偶数均满足 Goldbach 猜想。值得指出的是, 这样的方法并不适合于大偶数的验证。目前已由并行计算机验证的最大范围为 $[4, 10^{19}]$, 没有发现不满足该猜想的偶数^[1]。

10.1.2 模糊集合与隶属度函数

由经典集合论可见, 一个事物 a 要么属于集合 A , 要么不属于集合 A , 没有其他的属于关系。在现代科学与工程应用中, 经常会出现模糊的概念, 亦即某一事物 a 以一定程度属于集合 A , 该程度记作 $\mu_A(a)$, 称为隶属度函数, 其取值范围为 $\mu_A(a) \in [0, 1]$ 。该思想是模糊集合理论的基础。

模糊集合的概念是控制论专家 Lotfi A Zadeh 教授于 1965 年引入的^[2]。目前模糊逻辑已经广泛地应用于理、工、农、医等各种领域^[3]。在自动控制领域中模糊控制也是很有吸引力的研究方向。

例 10-5 Zadeh 教授给出了年老与年轻的模糊表示及隶属度函数, 假设论域 $U = [0, 120]$, 则

$$\mu_O(u) = \begin{cases} 0, & 0 \leq u \leq 50, \\ \left[1 + \left(\frac{u-50}{5}\right)^{-2}\right]^{-1}, & 50 < u \leq 120, \end{cases} \quad \mu_Y(u) = \begin{cases} \left[1 + \left(\frac{u-25}{5}\right)^{-2}\right]^{-1}, & 0 \leq u \leq 25 \\ 0, & 25 < u \leq 120 \end{cases}$$

这两个隶属度函数可以由下面语句直接求出并绘制出来, 如图 10-1 所示。

```
>> u=0:0.1:120; mu_o=1./(1+((u-50)/5).^(-2)).*(u>50);
mu_y=1./(1+((u-25)/5).^(-2)).*(u<25); plot(u,mu_y,u,mu_o)
```

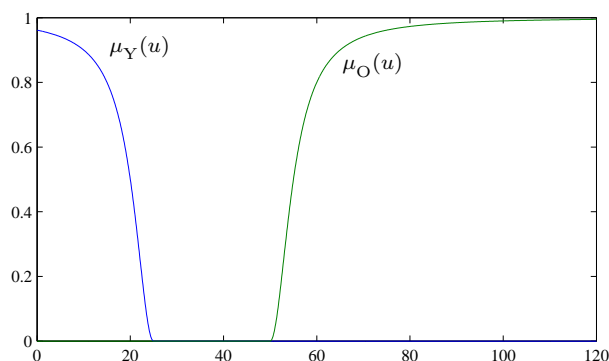


图 10-1 年老与年轻的隶属度函数曲线

在本书前面的介绍中实际上也使用了模糊的概念,例如变步长方法中关于误差的描述是当“误差较大时……”,只不过在实际处理时没有使用模糊的方法去处理,而直接使用了确定性方法解决问题。

这里不加解释地直接引入文献 [4] 给出的示意图来表示精确性与意义性,如图 10-2 所示。可以看出,现实世界中的事物并非都是越精确越好。Zadeh 教授指出,当问题的复杂性增加时,精确的描述将失去意义,而有意义的描述将失去精度。

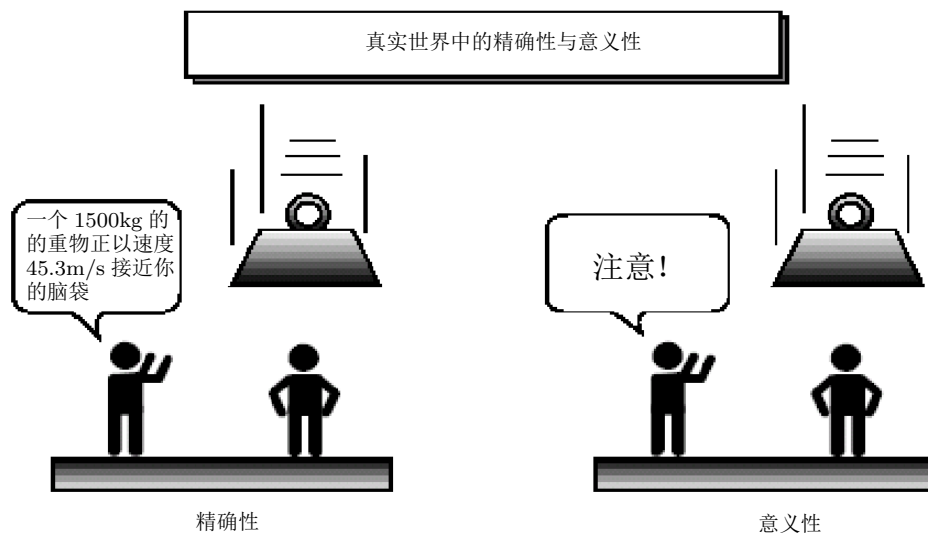


图 10-2 在现实世界中精确性与意义性示意图 (文献 [4])

在实际模糊集合与模糊推理系统中,可以选择各种各样的隶属度函数,下面将列出常用的隶属度函数。

(1) 钟形隶属度函数。钟形隶属度函数的数学表达式为

$$f(x) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}} \quad (10-1-1)$$

MATLAB 模糊逻辑工具箱中提供了函数 `gbellmf()`, 可以求出隶属度函数的值。该函数的调用格式为 `y = gbellmf(x, [a, b, c])`, 其中 x 为任意给定的自变量值。调用此函数则可以求出 x 处的隶属度函数值 y 。

例 10-6 试绘制出不同参数组合下的钟形隶属度函数曲线。具体的方法是, 先选定 x 向量, 再分别改变 a 、 b 、 c 的值, 可以得出如图 10-3 所示的隶属度函数曲线, 从得出的曲线可以观察出隶属度函数对 a 、 b 、 c 参数的依赖关系。

```
>> x=[0:0.05:10]'; y=[]; a0=1:5; b=2; c=3;
for a=a0, y=[y gbellmf(x,[a,b,c])]; end
y1=[]; a=1; b0=1:4; c=3; for b=b0, y1=[y1 gbellmf(x,[a,b,c])]; end
y2=[]; a=2; b=2; c0=1:4; for c=c0, y2=[y2 gbellmf(x,[a,b,c])]; end
plot(x,y); figure; plot(x,y1); figure; plot(x,y2)
```

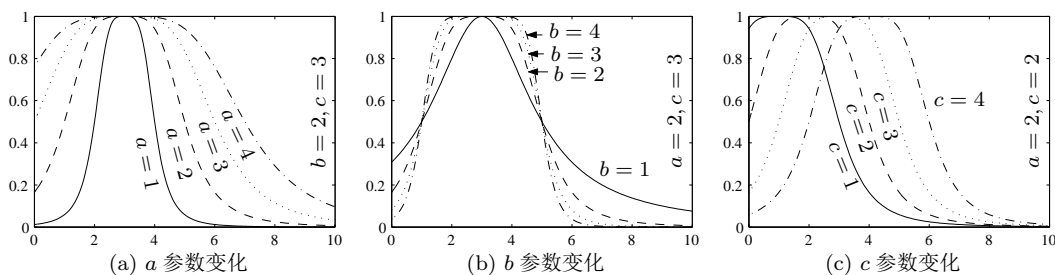


图 10-3 钟形隶属度函数曲线

从得出的曲线形状可以看出, 当其他参数不变, 只修改 a 值时, 若 a 值小则曲线形状很窄, 增大 a 值则曲线变宽, b 参数增大将增加上升段和下降段的陡度, c 参数只能用于平移曲线, 不改变曲线的形状, 可以通过这些参数的组合有意识地得出合适的隶属度函数。

(2) Gauss 隶属度函数。Gauss 隶属度函数的数学表达式为

$$f(x) = e^{-(x-c)^2/(2\sigma^2)} \quad (10-1-2)$$

MATLAB 模糊逻辑工具箱中提供了 `gaussmf()` 函数, 可以求取 Gauss 隶属度函数的值。该函数的调用格式为 `y = gaussmf(x, [σ, c])`。

例 10-7 不同 c 和 σ 参数的 Gauss 隶属度函数可以通过下面的语句绘制出来, 如图 10-4 所示。该函数实际上和第 9 章定义的正态分布概率密度函数形状是一致的。可以看出, 当 c 变化时, 隶属度函数曲线形状不变, 只作左右平移, σ 增大时曲线变宽。

```
>> x=[0:0.05:10]'; y=[]; c0=1:4; s=3; for c=c0, y=[y gaussmf(x,[s,c])]; end
y1=[]; c=5; sig0=1:4; for sig=sig0, y1=[y1 gaussmf(x,[sig,c])]; end;
plot(x,y); figure; plot(x,y1)
```

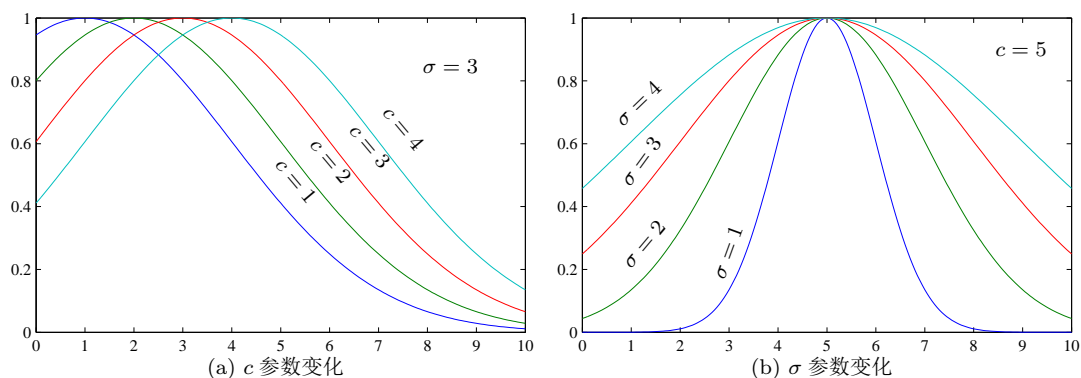


图 10-4 Gauss 隶属度函数曲线

(3) **Sigmoid 型隶属度函数**。Sigmoid 型隶属度函数的数学表达式为

$$f(x) = \frac{1}{1 + e^{-a(x-c)}} \quad (10-1-3)$$

该隶属度函数可以用 MATLAB 函数 `sigmf()` 求出 $y = \text{sigmf}(x, [a, c])$ 。

例 10-8 Sigmoid 函数在 a 和 c 变量的不同取值下隶属度函数形状如图 10-5 所示。可见, 当 c 参数增加或减小时, Sigmoid 函数向右或向左进行平移, 而隶属度函数的形状不变, 当 a 参数增大或减小时, 曲线变得更陡或更平缓。另外应该注意, 该函数是单值的, 故可以用于最右侧区间的隶属度函数描述, 最左侧区间的隶属度函数可以用 $1 - f(x)$ 来表示。

```
>> x=[0:0.05:10]'; y=[]; c0=1:4; a=3; for c=c0, y=[y sigmf(x,[a,c])]; end
    y1=[]; c=5; a0=1:2:7; for a=a0, y1=[y1 sigmf(x,[a,c])]; end
    plot(x,y); figure; plot(x,y1)
```

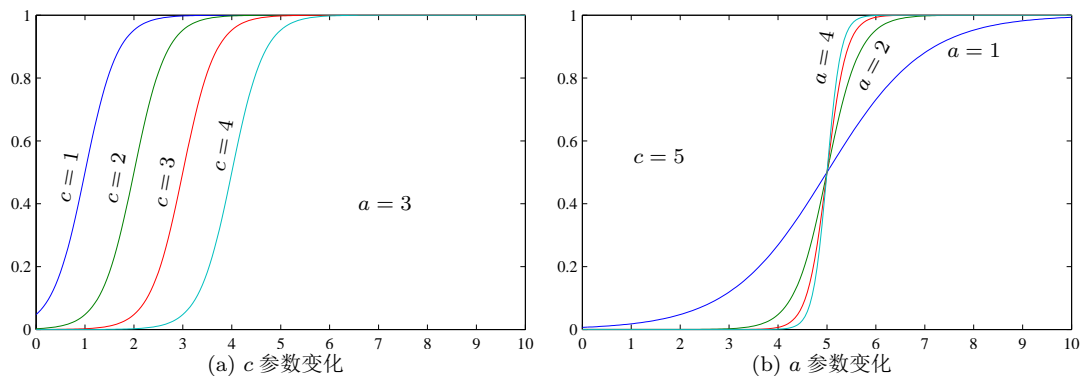


图 10-5 Sigmoid 隶属度函数曲线

隶属度函数可以由 MATLAB 模糊逻辑工具箱中提供的隶属度函数编辑界面进行编辑。在 MATLAB 提示符下键入 `mfedit` 命令就可以打开隶属度函数编辑界面, 如图 10-6 所示。其中给出了 3 个隶属度函数的原型, 用户可以通过界面中的选项设置各种隶属度函数, 可以由对话框右下栏目中的内容对当前隶属度函数的形状和参数进行编辑, 也可以通过鼠标在隶属度函数示意图上可视地修改隶属度函数的参数。

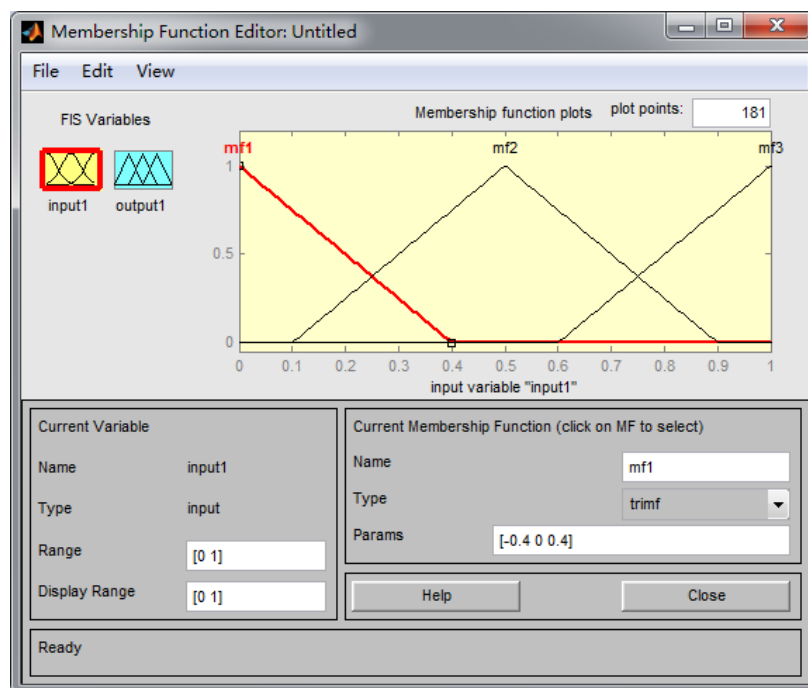
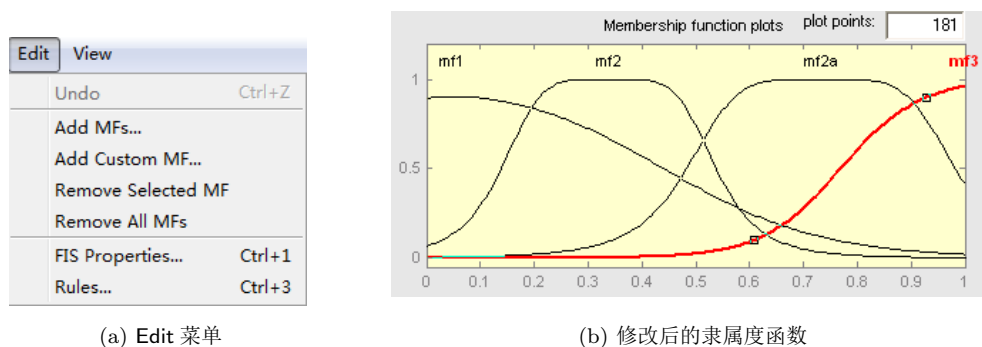


图 10-6 隶属度函数图形编辑界面

如果想再添加一个隶属度函数,则可以选择 **Edit** → **Add custom MF** 菜单,如图 10-7 (a) 所示,设置完成后就可以在编辑区域内添加一个隶属度函数,对这个新添加的隶属度函数可以按前面的方式进行修改,例如可以改变成如图 10-7 (b) 所示的形式。



(a) Edit 菜单

(b) 修改后的隶属度函数

图 10-7 隶属度函数的编辑结果

10.1.3 模糊推理系统及其 MATLAB 求解

用模糊逻辑工具箱中提供的 `newfis()` 函数可以构建出模糊推理系统的数据结构,其调用格式为 `fis = newfis(name)`,其中,FIS 为模糊推理系统 fuzzy inference system 的缩写,`name` 为字符串,表示模糊推理系统的名称,通过该函数可以建立起结构体 `fis`,其内容包括模糊的与、或运算,解模糊算法等,这些属性可以由 `newfis()` 函数直接定义,也可以事后定义。定义了模糊推理系统 `fis` 后,可以调用 `addvar()` 函数来添加系统的输入和输出变

量。该函数的调用格式为

```
fis = addvar(fis,'input',iname,vi)    % 定义一个输入变量 iname
fis = addvar(fis,'output',oname,vo)    % 定义一个输出变量 oname
```

其中, v_i 及 v_o 为输入或输出变量的取值范围,亦即最小值与最大值构成的行向量。通过这样的方法可以进一步定义 **fis** 的输入输出情况,每个变量的隶属度函数可以用 **addmf()** 函数定义,也可以用 **mfedit()** 定义。

例 10-9 假设某模糊推理系统有两个输入变量 ip_1 和 ip_2 ,并有一个输出变量 op ,且假设 ip_1 的取值范围为 $(-3,3)$,分为 3 个区间,隶属度函数选择为钟形函数;输入信号 ip_2 的取值范围为 $(-5,5)$,分为 3 个区间,隶属度函数选择为 Gauss 型函数;输出信号 op 的取值范围为 $(-2,2)$,隶属度函数为 Sigmoid 型函数,则可以用下面的语句构造模糊推理系统原型,并用 **fuzzy()** 函数编辑此模糊推理系统。由 **fuzzy()** 函数可以打开模糊推理系统的程序界面,如图 10-8 所示。

```
>> fff=newfis('c10mfis');                % 建立模糊推理系统模型
fff=addvar(fff,'input','ip1',[-3,3]);    % 定义第一路输入
fff=addvar(fff,'input','ip2',[-5,5]);    % 定义第二路输入
fff=addvar(fff,'output','op',[-2,2]);    % 定义输出信号
fuzzy(fff)                               % 用 fuzzy() 函数可视地编辑模糊推理系统
```

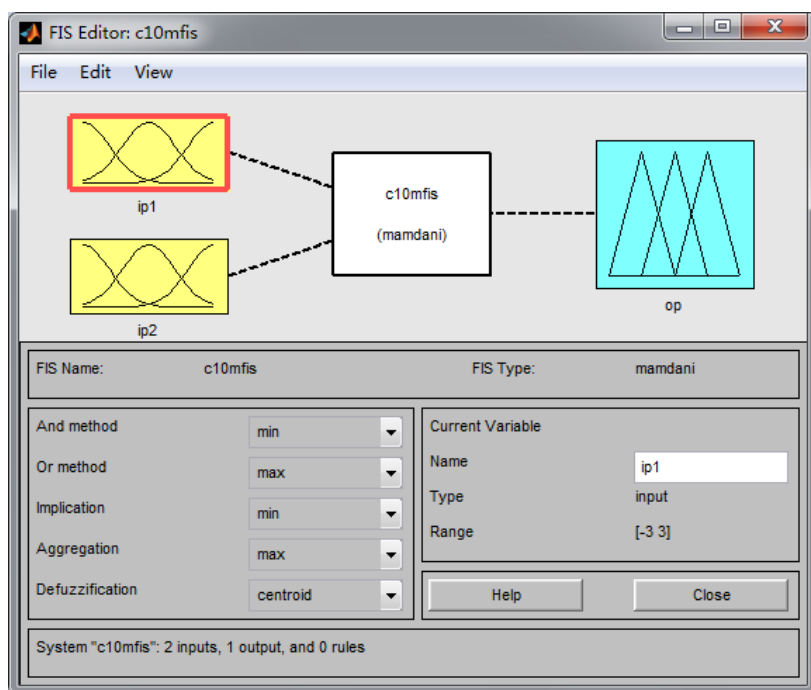
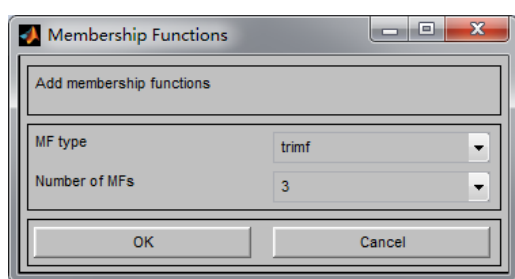


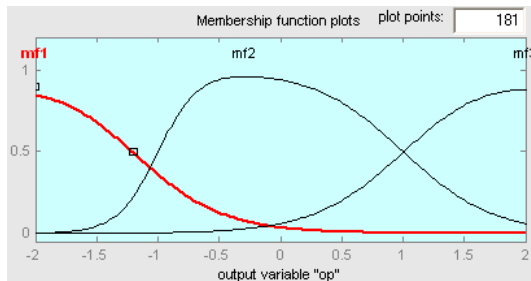
图 10-8 模糊推理系统编辑界面

在得出的界面下,选择 **Edit** → **Membership functions** 菜单项,打开如图 10-6 所示的隶属度编辑界面。在得出的界面上选择 ip_1 图标,再选择 **Edit** → **Add MFs** 菜单项,打开如图 10-9(a) 所示的对话框,可以通过该对话框定义各个信号的隶属度函数。例如,通过编辑得出如图 10-9(b) 所示的输出

隶属度函数。



(a) 隶属度函数设置对话框



(b) 修改后的输出变量隶属度函数

图 10-9 隶属度函数的编辑结果

1. 模糊化

若将某信号用 3 个隶属度函数表示,则一般对应的物理意义是“很小”、“中等”与“很大”,若分为 5 段,则可以表示为“很小”、“较小”、“中等”、“较大”和“很大”,一个精确的信号可以通过这样一组隶属度函数模糊化,变成模糊信号。

2. 模糊规则

如果将多路信号均模糊化,则可以用 **if, then** 型语句表示出模糊推理关系。例如,若输入信号 ip_1 “很小”,且输入信号 ip_2 “很大”,则设置“很大”的输出信号 op ,这样的推理关系可以表示成 **if ip_1 为“很小” and ip_2 为“很大”, then op = “很大”**。

模糊推理规则可以通过 `ruleedit()` 函数生成的界面来设定,也可以从 `mfedit()` 函数界面的 **Edit → Rules** 菜单项编辑模糊推理规则,这将打开如图 10-10 所示的对话框。用该对话框可以逐条将推理规则输入到系统中,每设定一条规则后,单击 **Add rule** 按钮,将规则添加到规则库中。如果想删除某条规则,则选中该规则,然后单击 **Delete rule** 按钮即可。

编辑后的规则还可以单击 **Change Rule** 按钮进行修改。完成了模糊规则的编辑,则可以单击 **Close** 按钮关闭编辑窗口。模糊推理规则可以由 **View → Surface** 菜单项进行处理,得出如图 10-11 (a) 所示的三维图形,表明从输入信号到输出信号的映射关系。

模糊规则还可以更简单地用数据向量表示,多行向量可以构成多条模糊规则矩阵。每行向量有 $m + n + 2$ 个元素, m, n 分别为输入变量和输出变量的个数,其中前 m 个元素表示输入信号的隶属度函数序号,次 n 个元素对应输出信号的隶属度函数序号,第 $m + n + 1$ 表示输出的加权系数,最后一个元素表示输入信号的逻辑关系,1 表示逻辑“与”,2 表示逻辑“或”。例如对图 10-10 中的第 3 条逻辑关系,若用数据向量的形式可以表示为 $[3, 2, 1, 1, 1]$,当然用界面处理模糊规则矩阵更简洁方便。

若用前面的规则生成一个规则矩阵 R ,则可以由下面的命令直接补加到模糊推理系统 `fis` 原有的规则后面,即 `fis = addrule(fis, R)`。

3. 解模糊化

通过模糊推理可以得出模糊输出量 op ,此模糊量可以通过指定的算法精确化,亦称解模糊化(defuzzification)。模糊逻辑工具箱提供了多种解模糊化的算法,可以由图 10-8 所示的对话框 **Defuzzifications** 栏目,即对话框中如图 10-11 (b) 所示的部分选择解模糊化算法。

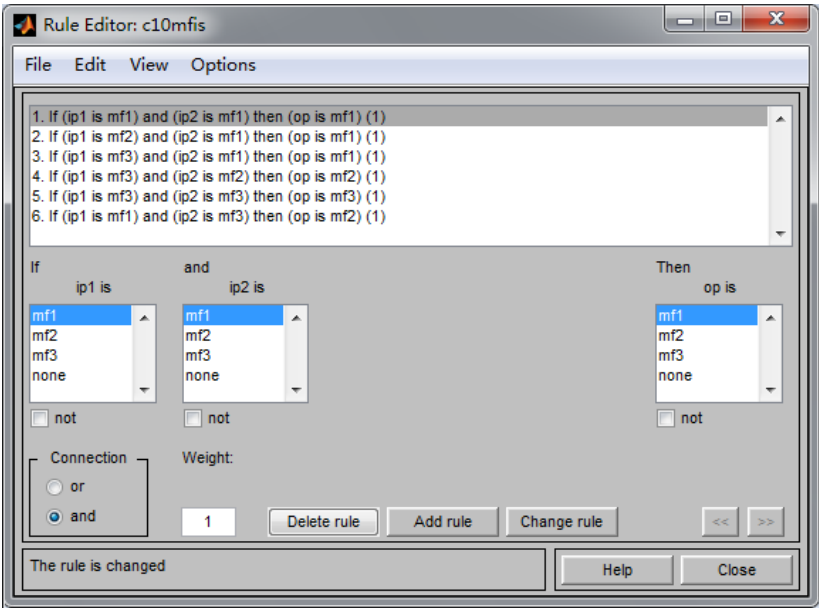


图 10-10 模糊规则编辑对话框

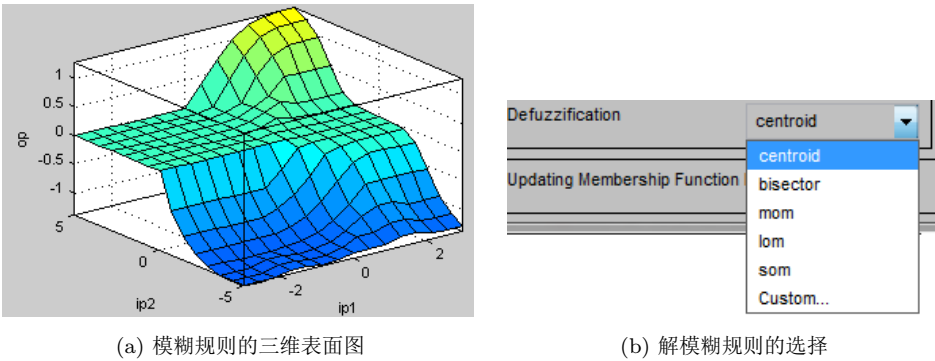


图 10-11 模糊规则图形表示

按照上述方式就可以建立起模糊推理系统的数据结构。可以由 File 菜单对模型进行处理,例如可以用 File → Export → To Disk 菜单项将其存成文件,后缀名为 fis。用户可以将前面编辑的模糊推理系统存储成 c10mfis.fis 文件。该工作还可以通过 writefis() 函数完成。还可以由 File → Export → To Workspace 菜单项将其存入 MATLAB 工作空间,存储时应该给出变量名。

模糊推理问题还可以用 MATLAB 函数 evalfis() 求解, $y = \text{evalfis}(X, \text{fis})$, 其中, X 为矩阵,其各列为各个输入信号的精确值,evalfis() 函数会对用户定义的模糊推理系统 fis 计算这些输入信号的模糊化结果,用该系统进行模糊推理,并将结果进行解模糊化,得出相应的精确输出信号 y 。

例 10-10 假设已经按上述方式建立起了模糊推理模型,在 x - y 平面内的 $(-3, -5) \sim (3, 5)$ 区域内进行网格分割,试用此模糊推理系统绘制出输出的三维曲面。

解 采用下面的语句可以先读入前面建立的模糊推理系统,并对感兴趣的 x - y 平面区域进行网格分割,将网格数据转换成列向量,再由 `evalfis()` 函数求出曲面的 z 坐标值,这样就可以用下面的语句绘制出三维曲面,如图 10-12 所示。

```
>> fff=readfis('c10mfis.fis');           % 读入模糊推理系统文件
    [x,y]=meshgrid(-3:.2:3,-5:.2:5);      % 进行网格分割
    x1=x(:); y1=y(:); z1=evalfis([x1 y1],fff); % 模糊推理
    z=reshape(z1,size(x)); surf(x,y,z)     % 绘制曲面
```

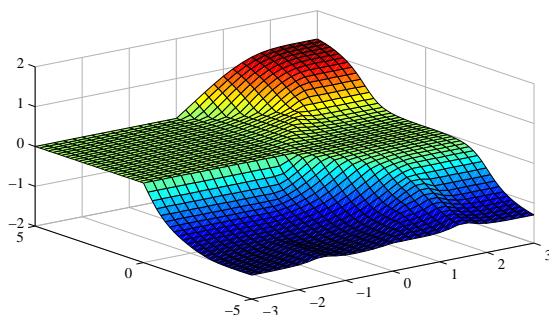


图 10-12 由模糊推理得出的输出曲面

10.2 粗糙集理论与应用

10.2.1 粗糙集理论简介

粗糙集 (rough set) 是波兰数学家 Zdzisław Pawlak 为开发自动规则生成系统及研究软计算问题于 1982 年提出的。20 世纪 90 年代初,人们才逐渐认识到粗糙集的重要性。1991 年 Pawlak 教授出版了专著,奠定了严密的数学基础^[5]。基于粗糙集的知识理论由于不需要预先给定某些特征或属性的数量,可从现有的数据出发给出知识的简化和相对简化、范畴的简化和相对简化方法,为处理不精确、不完全信息提供一种更符合人类认知的知识理论。粗糙集理论是一种处理不精确、不确定与不完全数据的新的数学方法。它能有效地分析和处理不精确、不一致、不完整等各种不完备信息,并从中发现隐含的知识,揭示潜在的规律。由于它在机器学习与知识发现、数据挖掘、决策支持与分析、专家系统、归纳推理、模式识别、知识约简、信息计算等方面的应用突出,现已成为一个热门的研究领域。

10.2.2 粗糙集的基本概念

设 $X, Y \in U$, R 是定义在 U 上的等价关系,则集合 X 关于 R 的下近似集定义为

$$\underline{\mathcal{R}}(X) = \bigcup \{Y \in U/R : Y \subseteq X\} \quad (10-2-1)$$

其中, $\underline{\mathcal{R}}(X)$ 是根据现有知识判断肯定属于 X 的对象组成的最大的集合,称为正区,记为 $\text{Pos}(X)$ 。类似地,也可以定义出集合 X 关于 R 的上近似集为

$$\overline{\mathcal{R}}(X) = \bigcup \{Y \in U/R : Y \cap X \neq \emptyset\} \quad (10-2-2)$$

其中, ϕ 表示为空集。 $\overline{\mathcal{R}}(X)$ 是由所有集合 X 相交非空的等效类的并集, 是那些可能属于 X 的对象组成的最小集合。

由上面的定义可以再给出边界集 c 定义为 $c = \text{Bnd}(X) = \overline{\mathcal{R}}(X) - \underline{\mathcal{R}}(X)$ 。如果 $\text{Bnd}(X)$ 是空集, 则称 X 关于 R 是清晰的; 反之, 若 $\text{Bnd}(X)$ 非空, 则称 X 为关于 R 的粗糙集。

例 10-11 假设玩具积木集合 $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, 具有“颜色 R_1 ”、“形状 R_2 ”、“体积 R_3 ”这 3 种属性, 且 $R_1 = \{0, 1, 2\}$, 分别对应红色、黄色和绿色, “形状”的属性值取为 $R_2 = \{0, 1, 2\}$, 分别对应方、圆、三角形。“体积”的属性关系, 可以取为 $R_3 = \{0, 1\}$, 分别对应于“大的物体”和“小的物体”。

按照这样的属性关系, 假设红色的积木有 $\{x_1, x_2, x_7\}$, 绿色的有 $\{x_3, x_4\}$, 黄色的有 $\{x_5, x_6\}$, 则可以写出 $U|R_1 = \{\{x_1, x_2, x_7\}, \{x_3, x_4\}, \{x_5, x_6\}\}$ 。

10.2.3 信息决策系统

信息决策系统 T 可以表示为 $T = (U, A, C, D)$, 其中, U 是对象的集合, 即论域, A 是属性集合, 如果属性集 A 可以分为条件属性集 C 和决策属性集 D , 即 $C \cup D = A, C \cap D = \phi$, 则该信息系统称为决策系统或决策表。

粗糙集理论中使用决策表来描述论域中对象。它是一张二维表格, 每一行描述一个对象, 每一列描述对象的一种属性。属性分为条件属性和决策属性, 论域中的对象根据条件属性的不同, 被划分到具有不同决策属性的决策类。表 10-2 为一张信息系统决策表的例子, $U = \{x_1, x_2, \dots\}$ 为对象集, $C = \{s_1, \dots, s_m\}$ 为条件属性集, $D = \{d_1, \dots, d_k\}$ 为决策属性集, f_{ij} 表示第 i 个对象的第 j 个条件属性值, g_{ij} 是第 i 个对象的第 j 个决策属性值。

表 10-2 信息系统决策表

论域	C				D		
U	s_1	s_2	\dots	s_m	d_1	\dots	d_k
x_1	f_{11}	f_{12}	\dots	f_{1m}	g_{11}	\dots	g_{1k}
x_2	f_{21}	f_{22}	\dots	f_{2m}	g_{21}	\dots	g_{2k}
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x_n	f_{n1}	f_{n2}	\dots	f_{nm}	g_{n1}	\dots	g_{nk}

表 10-3 例 10-12 信息系统决策表

论域	C 属性				D
U	颜色 a	形状 b	大小 c	价位 d	销量
1	1	0	1	1	1
2	1	0	0	0	1
3	0	0	1	0	0
4	1	1	0	1	0
5	1	1	1	2	2
6	2	1	0	2	2
7	2	2	0	2	2

例 10-12 考虑例 10-11 中给出的玩具积木论域集合, 假设有 4 个相关属性, 颜色属性 a、形状属性 b、大小属性 c 和价位属性 d, 且已知 $x_{1,2,4,5}$ 为黄色的, x_3 为红色的, $x_{6,7}$ 为绿色的; $x_{1,2,3}$ 为方形的, $x_{4,5,6}$ 为圆形的, x_7 为三角形的; $x_{1,3,5}$ 为大玩具, 其余为小玩具; $x_{2,3}$ 价位较低, $x_{1,4}$ 价位中等, $x_{5,6,7}$ 价位较高。从实际销售情况看, $x_{3,4}$ 销售很好, $x_{1,2}$ 销售一般, 而 $x_{5,6,7}$ 销售较差, 试列出该问题的信息系统决策表。

解 设颜色属性中用 $\{0, 1, 2\}$ 分别表示红色、黄色和绿色, 形状属性中 $\{0, 1, 2\}$ 分别表示方形、圆形和三角形, 大小属性中 $\{0, 1\}$ 分别表示玩具的小和大, 价位属性中 $\{0, 1, 2\}$ 分别表示较低、中等和较高, 在决策属性中 $\{0, 1, 2\}$ 分别表示销售较好、中等和较差, 那么根据给出的表格可以立即得出信

息系统决策表,如表 10-3 所示。粗糙集理论的一个重要应用是对已知各个条件进行约简,找出哪些属性对玩具销售情况影响大,哪些没有影响。

信息系统决策表在 MATLAB 下可以由一个矩阵 S 来表示,其中前面各列表示 C 属性,后面各列表示 D 属性,这时上近似集 $\overline{\mathcal{R}}(X)$ 和下近似集 $\underline{\mathcal{R}}(X)$ 可以分别由自编的 `rsupper()` 和 `rslower()` 函数求出。其内容如下

```
function w=rslower(y,a,T)
z=ind(a,T); w=[]; [p,q]=size(z);
for u=1:p,
    zz=setdiff(z(u,:),0); if ismember(zz,y), w=cat(2,w,zz); end
end

function w=rsupper(y,a,T)
z=ind(a,T); w=[]; [p,q]=size(z);
for u=1:p
    zz=setdiff(z(u,:),0); zzz=intersect(zz,y);
    if length(zzz)~=0, w=cat(2,w,zz); end
end
```

这两个函数共用的支持函数 `ind()` 用于求取不可分辨关系,后面将给出其定义与 MATLAB 实现。计算边界集的函数可以采用 MATLAB 自带的 `setdiff()` 函数。这样,下近似集、上近似集和边界集可以分别通过下面的函数调用求取。

```
Sl = rslower(X,a,S)    % 求出下近似集 Sl
Su = rsupper(X,a,S)    % 求出上近似集 Su
Sd = setdiff(Su,Sl)    % 利用 MATLAB 的差集函数可以求出边界集 Sd
```

例 10-13 假设论域 $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$, 关系为 $R = \{R_1, R_2\}$, 且

$U/R_1 = \{\{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}\{x_9, x_{10}\}\},$

$U/R_2 = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6, x_7\}\{x_8, x_9, x_{10}\}\},$

若 $X = \{x_1, x_2, x_3, x_4, x_5\}$, 试求出集合 X 的上近似集和下近似集。

解 根据题中已知条件,分别简记 U/R_1 和 U/R_2 中的 3 个子集为 $\{0, 1, 2\}$, 则可以建立起表 10-4 中给出的信息系统决策表 S 。注意,这里为排版方便起见,将信息系统决策表进行了旋转。选择集合 $X = \{1, 2, 3, 4, 5\}$, 并选择决策表中的第 1、2 列构成 a 向量,则可以由下面语句计算出集合 X 的上下近似集和边界集。

表 10-4 信息系统决策表

论域 X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
U/R_1 关系	0	0	0	0	1	1	1	1	2	2
U/R_2 关系	0	0	0	1	1	1	1	2	2	2

```
>> S=[0,0; 0,0; 0,0; 0,1; 1,1; 1,1; 1,1; 1,2; 2,2; 2,2];
X=[1,2,3,4,5]; a=[1,2]; S1=rslower(X,a,S) % 下近似集
```

$S_2 = \text{rsupper}(X, a, S)$, $S_d = \text{setdiff}(S_2, S_1)$ % 上近似集和边界集

可以得出, $S_1 = [1, 2, 3, 4]$, $S_2 = [1, 2, 3, 4, 5, 6, 7]$, $S_d = [5, 6, 7]$ 。从决策表可见, $\{U/R_1, U/R_2\}$ 构成的集合总共有 $\{0, 0\}$ 、 $\{0, 1\}$ 、 $\{1, 1\}$ 、 $\{1, 2\}$ 、 $\{2, 2\}$, 选择的样本 $X = \{1, 2, 3, 4, 5\}$, 涉及的 $\{U/R_1, U/R_2\}$ 集合只有 $\{0, 0\}$ 、 $\{0, 1\}$ 和 $\{1, 1\}$, 所以, 肯定属于样本集合 X 的只有 $\{1, 2, 3, 4\}$, 亦即 $\{x_1, x_2, x_3, x_4\}$, 因为和 x_5 一样具有映射关系 $\{1, 1\}$ 的还有 x_6, x_7 , 所以可以得出下近似集为 $\{x_1, x_2, x_3, x_4\}$ 。类似地, 可能属于样本集合 X 的样本是 X 的上近似集, 由于 x_6, x_7 和 X 集合中的 x_5 都为 $\{1, 1\}$, 所以上近似集中除了下近似集中的样本外还应该包括 x_5, x_6, x_7 , 这 3 个样本亦为边界集。此外, 由于边界集非空, 所以属于粗糙集。

在信息系统中, 对于每个属性子集 $R \subseteq A$, 不可分辨关系为

$$\text{Ind}(R) = \{(x, y) \in U \times U : r \in R : r(x) = r(y)\} \quad (10-2-3)$$

显然, $\text{Ind}(R)$ 是一个等价关系, 在不产生混淆的情况下可以用 R 代替 $\text{Ind}(R)$ 。不可分辨关系的 MATLAB 函数可以如下编写

```
function aa=ind(a,x)
[p,q]=size(x); [ap,aq]=size(a); z=1:q;
tt=setdiff(z,a); x(:,tt(size(tt,2):-1:1))=-1;
for r=q:-1:1, if x(1,r)==-1, x(:,r)=[]; end, end
for i=1:p, v(i)=x(i,:)*10.^(aq-[1:aq]'); end
y=v'; [yy,I]=sort(y); y=[yy I];
[b,k,l]=unique(yy); y=[l I]; m=max(l); aa=zeros(m,p);
for ii=1:m, for j=1:p, if l(j)==ii, aa(ii,j)=I(j); end, end, end
```

10.2.4 粗糙集数据处理问题的 MATLAB 求解

1. 利用粗糙集理论的约简

目前社会已经进入信息时代, 人们获得信息越来越容易。但大量的未处理的信息使人们陷入“数据灾难”、“决策灾难”, 导致要么“疲于应付”, 要么“弃之不理”。对解决这类问题的研究, 一般称为“从数据库中发现知识”与“数据挖掘”。

信息系统约简主要是使信息量减少, 它将一些无关或多余的信息忽略掉, 而不影响其原有的决策功能。可以设想将约简后的信息重新组合而产生新的决策规则, 这类决策规则的前提信息和结论信息可能不同于约简前的任何一条决策规则, 但它们能经推理而得到相同或相近的结果。因此这样的研究成果对数据挖掘以及数据库的进一步应用将产生新的影响。

所谓约简, 即不含多余属性并保证分类正确的最小条件属性集。一个信息决策表可能同时存在几个约简。关系等价族 R 中所有不可约去的关系称为核, 由它构成的集合称为 R 的核集, 记成 $\text{Core}(R)$ 。

这里不详细介绍约简的具体算法, 只介绍依据约简算法编写的几个 MATLAB 函数, 如 $\text{redu}()$ 、 $\text{core}()$ 等。关于约简算法的详细讨论请见文献 [6]。

假设信息系统决策表由矩阵 S 表示, 向量 c 和 d 分别为条件属性 C 和决策属性 D 的

编号,则从 C 属性中相关的列中直接使得决策属性 D 成立的最少列的求取可以由约简函数 $\text{redu}()$ 找出。这些函数的调用格式为

```
 $y = \text{redu}(c,d,S)$     % 条件约简,找出从  $C$  属性中选定条件推出  $D$  的最小集合
 $y = \text{core}(c,d,S)$     % 求取从  $C$  属性中选定条件推出  $D$  的核集
```

2. 粗糙集理论在信息约简中的应用举例

前面介绍了约简的基本概念和约简问题的 MATLAB 求解函数,这些函数在本书配套程序包中给出,可以直接使用。下面将通过两个实际应用的例子^[6]来演示基于粗糙集运算的信息约简方法及 MATLAB 求解。

例 10-14 0~9 这 10 个数字的显示一般采用 7 段数码管来实现,这 7 段数码管排序如图 10-13(a) 所示。其中某段数码管发光则记为 1,否则记为 0,这样每个数字显示对应的真值表如表 10-5 所示。试用粗糙集的方法对其进行约简,找出不必要的数码管。

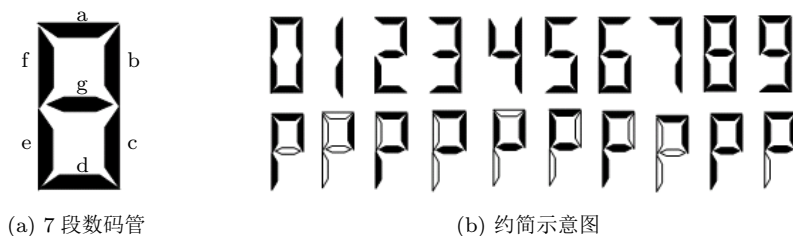


图 10-13 数码管显示及数码管约简结果

表 10-5 数码管显示真值表

数码 X	C 属性							D 属性	数码 X	C 属性							D 属性
	a	b	c	d	e	f	g			a	b	c	d	e	f	g	
0	1	1	1	1	1	1	0	0	5	1	0	1	1	0	1	1	5
1	0	1	1	0	0	0	0	1	6	1	0	1	1	1	1	1	6
2	1	1	0	1	1	0	1	2	7	1	1	1	0	0	0	0	7
3	1	1	1	1	0	0	1	3	8	1	1	1	1	1	1	1	8
4	0	1	1	0	0	1	1	4	9	1	1	1	1	0	1	1	9

解 从人类对数字的直观理解看,这 7 段数码管当然全是必要的,缺少哪段都不易被人准确辨认出来。但如果考虑用计算机来识别数字,就不一定完全遵循数字的直观显示了,可以有一种内部的映射。若去掉某一段数码管后不影响辨认这 10 个数字,则得出的新映射关系就可以理解成原始 7 段数码管形式的一个约简。利用下面的 MATLAB 语句可以立即得出下面的约简结果,若最后一个语句调用 $\text{redu}()$ 函数也将得到同样的结果。

```
>> C=[1,1,1,1,1,1,0; 0,1,1,0,0,0,0; 1,1,0,1,1,0,1; 1,1,1,1,0,0,1;
      0,1,1,0,0,1,1; 1,0,1,1,0,1,1; 1,0,1,1,1,1,1; 1,1,1,0,0,0,0;
      1,1,1,1,1,1,1; 1,1,1,1,0,1,1];
D=[0; 1; 2; 3; 4; 5; 6; 7; 8; 9]; X=[C D];
c=1:7; d=8; Y=core(c,d,X) % 其中 1~7 列为  $C$  属性, 8 列为  $D$  属性
```

可见, $Y = [1, 2, 5, 6, 7]$ 段数码管是不能约简掉的, 而 3、4(即图 10-13(a) 中的 c、d 段) 是可有可无的, 去掉它们并不影响辨认数码管显示的数字, 可以用图 10-13(b) 中的映射关系辨认数字。这样做虽然对人工辨认没有什么好处, 但对机器辨认数字无疑会很方便, 对机器视觉和书写体数字识别等领域是很有用途的。

例 10-15 SARS 是 2003 年给全球带来恐慌的疾病, 其准确诊断是很困难的。这里给出从报刊提取出的一些数据, 构成表 10-6, 试利用粗糙集理论对给出的 12 个条件进行约简, 找出辅助诊断的最主要的条件。这里的数据有些不确切, 数据样本也不完全, 所以不能真正用于临床诊断。

表 10-6 SARS 患者和正常人若干检测指标表

	C 属性													D
U	干咳	呼吸困难	血液检测	高烧 38°C	X 射线	浓痰	白细胞多	寒战	肌肉酸痛	乏力	胸膜痛	头痛	SARS	
1	1	1	1	1	0	0	0	0	1	1	0	1	1	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	1	0	1	0	0	0	0	0	0	1	0	0	0	
4	0	0	0	1	1	1	1	0	1	0	1	1	0	
5	1	0	0	1	1	1	1	1	0	1	1	0	0	
6	0	1	0	1	1	1	1	1	1	0	0	1	0	
7	1	0	0	0	1	1	1	0	0	1	1	1	0	
8	1	1	1	1	0	0	0	0	1	1	0	1	1	
9	1	0	1	1	1	0	0	0	1	1	0	1	1	
10	1	1	1	1	0	0	0	0	1	1	0	1	1	
11	1	0	1	1	1	0	0	0	1	1	0	1	1	
12	1	0	1	1	1	0	0	0	1	1	0	1	1	

解 根据题意, 可以给出如下命令来进行条件约简, 最后得出的条件为 $Y = [3, 4]$, 表示第 3 和第 4 列是诊断 SARS 的重要因素, 亦即“血液检测呈阳性”和“高烧 38°C”。

```
>> D=[1; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1; 1; 1];
C=[1,1,1,1,0,0,0,0,1,1,0,1; 0,0,0,0,0,0,0,0,0,0,0,0,0;
1,0,1,0,0,0,0,0,0,1,0,0; 0,0,0,1,1,1,1,0,1,0,1,1;
1,0,0,1,1,1,1,1,0,1,1,0; 0,1,0,1,1,1,1,1,1,0,0,1;
1,0,0,0,1,1,1,0,0,1,1,1; 1,1,1,1,0,0,0,0,1,1,0,1;
1,0,1,1,1,0,0,0,1,1,0,1; 1,1,1,1,0,0,0,0,1,1,0,1;
1,0,1,1,1,0,0,0,1,1,0,1; 1,0,1,1,1,0,0,0,1,1,0,1];
Y=redu(1:12,13,[C D])
```

10.2.5 粗糙集约简的 MATLAB 程序界面

基于粗糙集约简的理论和方法, 编写了 MATLAB 程序界面。在 MATLAB 提示符下键入 rsdav3 命令则将启动该程序界面, 得出如图 10-14 所示的对话框, 用户可以由其中的 Browse 按钮读入信息系统决策表, 给出 C 属性和 D 属性所需的列号, 则可以进一步进行分析。例如, 单击 Redu 按钮可以进行约简, 结果将在 Results 栏目显示出来。

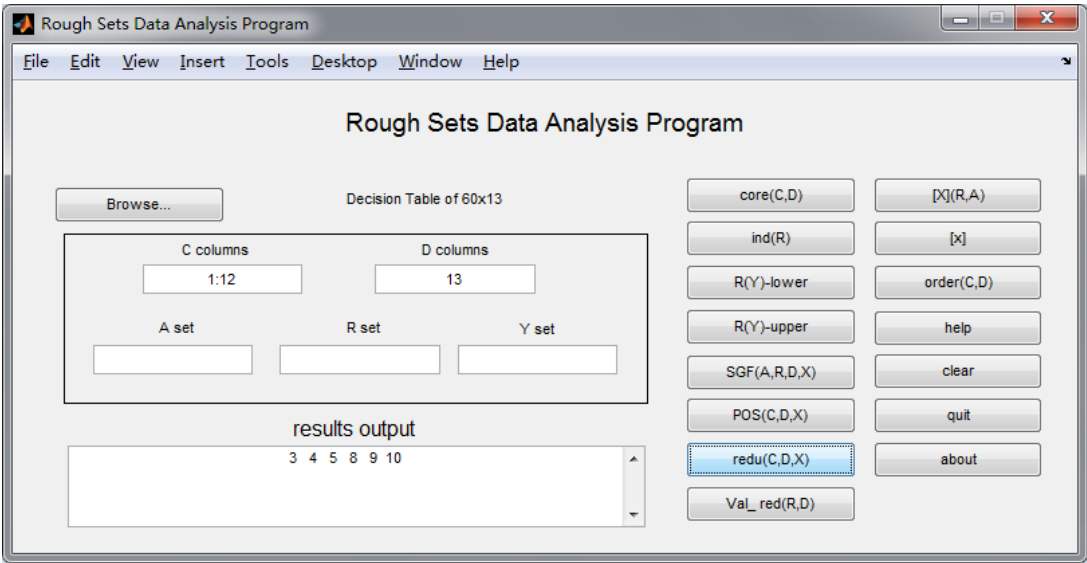


图 10-14 粗糙集数据分析程序界面

10.3 人工神经网络及其在数据拟合中的应用

人工神经网络是在对复杂的生物神经网络研究和理解的基础上发展起来的。人脑是由大约 10^{11} 个高度互连的单元构成,这些单元称为神经元,每个神经元约有 10^4 个连接 [7]。仿照生物的神经元,可以用数学方式表示神经元,引入人工神经元的概念,并由神经元的互连可以定义出不同种类的神经网络。限于当前的计算机水平,人工神经网络不可能有人脑那么复杂。本节将首先介绍人工神经元和人工神经网络的数学结构,然后介绍神经网络的建立、训练与泛化的概念以及 MATLAB 语言的神经网络工具箱在解决这些问题中的应用,最后介绍一个用户界面来利用神经网络求解数据拟合中的问题。

10.3.1 神经网络基础知识

1. 前馈神经网络结构与应用

单个人工神经元的数学表示形式如图 10-15 所示。其中, x_1, x_2, \dots, x_n 为一组输入信号,它们经过权值 w_i 加权后求和,再加上阈值 b ,则得出 u_i 的值,可以认为该值为输入信号与阈值所构成的广义输入信号的线性组合。该信号经过传输函数 $f(\cdot)$ 可以得出神经元的输出信号 y 。

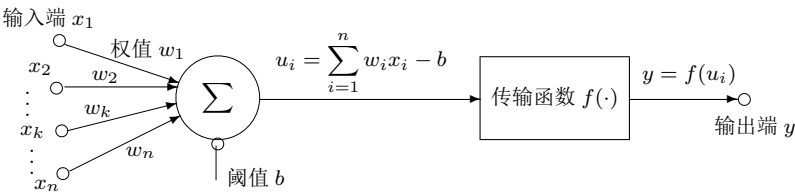


图 10-15 神经元的基本结构

在神经元中,权值和传输函数是两个关键的因素。权值的物理意义是输入信号的强度,若涉及多个神经元则可以理解成神经元之间的连接强度。神经元的权值 w_i 应该通过神经元对样本点反复的学习过程而确定,而这样的学习过程在神经网络理论中又称为训练。传输函数又称为激励函数,可以理解成对 u_i 信号的非线性映射,一般的传输函数应该为单值函数,使得神经元是可逆的。常用的传输函数有 Sigmoid 函数和对数 Sigmoid 函数,它们的数学表达式分别为

$$\begin{aligned} \text{Sigmoid 函数 } f(x) &= \frac{2}{1 + e^{-2x}} - 1 = \frac{1 - e^{-2x}}{1 + e^{-2x}} \\ \text{对数 Sigmoid 函数 } f(x) &= \frac{1}{1 + e^{-x}} \end{aligned} \quad (10-3-1)$$

当然也可以使用简单的饱和函数和阶跃函数作为传输函数。下面将通过例子介绍各类传输函数的形状及基于 MATLAB 神经网络工具箱的绘制方法。

例 10-16 试绘制各种常用的传输函数曲线。

解 用下面的语句可以直接绘制出 Sigmoid 函数的曲线,如图 10-16 所示。

```
>> x=-2:0.01:2; y=tansig(x); plot(x,y)
```

用 `logsig()` 语句取代前面的 `tansig()` 函数则得出对数 Sigmoid 函数曲线。另外,由其他函数可以绘制出其传输函数曲线,如图 10-16 所示,同时标出了绘制这些传输函数的 MATLAB 函数名。

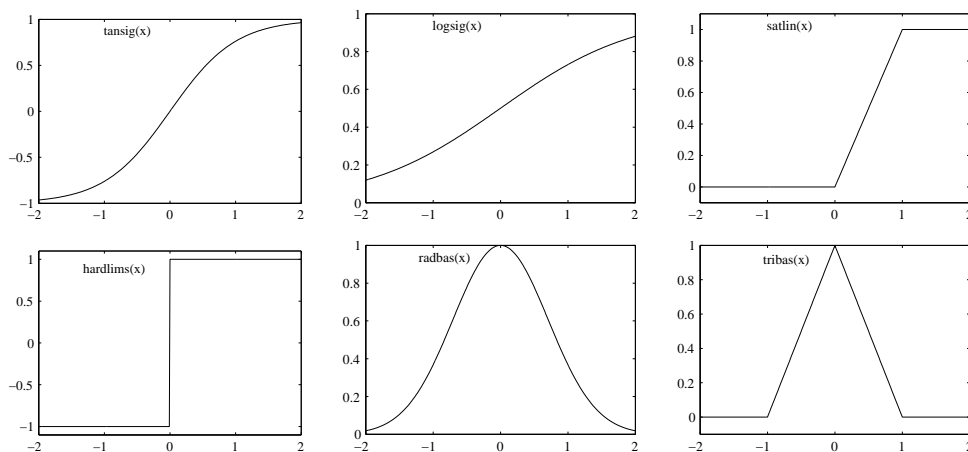


图 10-16 各种传输函数曲线

由若干个神经元相互连接,可以构成网络,称为神经网络。由于连接方式的不同,神经网络的类型也不同。这里仅介绍前馈神经网络,因为其权值训练中采用误差逆向传播的方式,所以这类神经网络更多地称为反向传播(backward propagation, BP)神经网络。BP 网的基本网络结构如图 10-17 所示。这类神经网络层数的定义和一般神经网络教材稍有差异,图 10-17 中给出的最后一个隐层实际上本身就是输出层,所以实际隐层数为 $k-1$ 。

以两层网络为例, $k=2$, m 为输出端子的个数, n 为输入端子路数, p 为隐层节点个数。

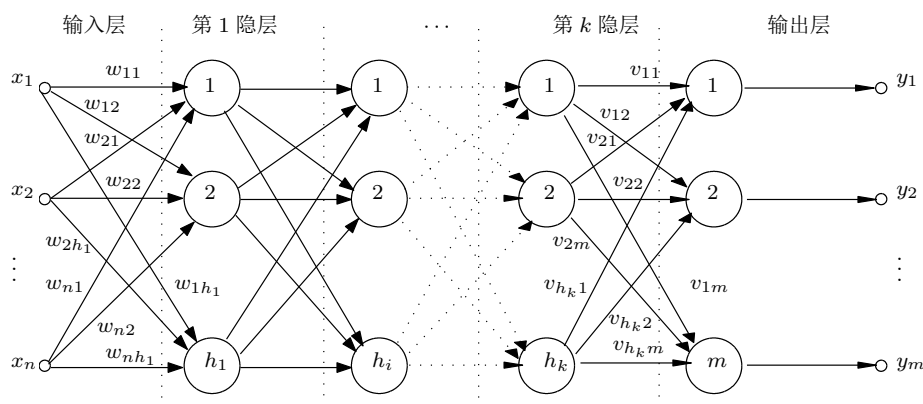


图 10-17 前馈型神经网络的基本结构

这时，神经网络示意图如图 10-18 所示。隐层节点在传输函数前后的值分别为

$$u_j = \sum_{i=1}^n w_{ij}x_i + b_{1j}, \quad u'_j = F_1(u_j), \quad j = 1, \dots, p \quad (10-3-2)$$

其中 b_{1j} 为隐层节点的阈值，而输出层传输函数前后的信号分别为

$$y'_j = \sum_{i=1}^p v_{ji}u'_i + b_{2j}, \quad y_j = F_2(y'_j), \quad j = 1, \dots, m \quad (10-3-3)$$

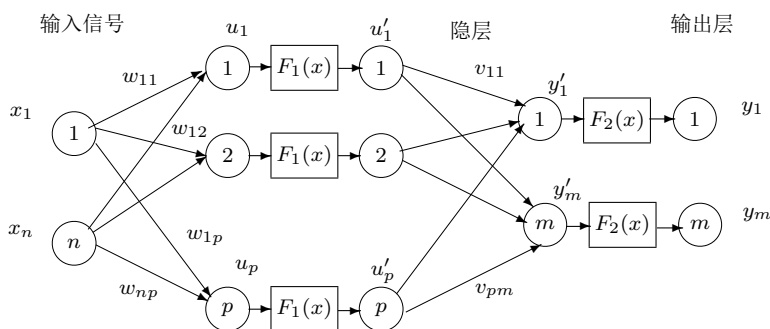


图 10-18 单个隐层的前馈神经的基本结构

如果已知 N 组用于训练的实际样本点数据，其输入和输出之间关系对照如下

$$\begin{array}{ccccccc} x_{11} & x_{12} & \cdots & x_{1n} & \Rightarrow & \hat{y}_{11} & \cdots & \hat{y}_{1m} \\ x_{21} & x_{22} & \cdots & x_{2n} & \Rightarrow & \hat{y}_{21} & \cdots & \hat{y}_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} & \Rightarrow & \hat{y}_{N1} & \cdots & \hat{y}_{Nm} \end{array} \quad (10-3-4)$$

神经网络要解决的问题是通过已知数据，反复训练神经网络，得到两层的加权量 w_{ij} 、 v_{ij} 和阈值 b_{ij} ，使得神经网络的计算输出信号 y_i 与实际期望输出信号 \hat{y}_i 的误差最小。一种

较合适的方式就是使得误差的平方和最小,即

$$\min_{\mathbf{W}, \mathbf{V}} \sum_{l=1}^N \sum_{i=1}^m (y_{li} - \hat{y}_{li})^2 \quad (10-3-5)$$

其中下标 l 为样本组数。

对于这样的无约束最优化问题,可以采用反复地求导、解方程来得出决策矩阵 \mathbf{W} 和 \mathbf{V} ,或采用共轭梯度法等算法来搜索最优值。给出权值 \mathbf{W}_{ij} 和 \mathbf{V}_{ij} 的初值 \mathbf{W}_{ij}^0 和 \mathbf{V}_{ij}^0 ,则可以通过下面的递推算法修正权值,得出 [8]

$$\mathbf{W}_{ij}^{l+1} = \mathbf{W}_{ij}^l + \beta e_j^l a_i^l, \quad \mathbf{V}_{jt}^{l+1} = \mathbf{V}_{jt}^l + \alpha d_t^l \gamma_j^l \quad (10-3-6)$$

其中, $i = 1, \dots, n; j = 1, \dots, p; t = 1, \dots, m; \alpha$ 和 β 为人为指定的速度常数。中间变量 a_i^l 、 γ_j^l 、 e_j^l 、 d_t^l 可以迭代求出。

利用 MATLAB 语言的神经网络工具箱提供的现成函数和神经网络类,建立一个前馈的 BP 神经网络模型还是很容易的,可以使用 `feedforwardnet()` 函数,该函数具体的调用格式为 `net = feedforwardnet(h, f)`, 其中, $\mathbf{h} = [h_1, h_2, \dots, h_k]$ 为各隐层节点个数向量, f 为训练用函数,默认值为 'trainlm', 表示采用 Levenberg-Marquardt 反向传播训练算法。给了该命令就可以构造出神经网络数据对象 `net`, 该对象的一些重要属性在表 10-7 中给出。

表 10-7 神经网络对象的常用属性

属性名	数据类型	属性说明	默认参数
<code>net.IW</code>	单元数组	输入层和隐层加权, 其中 <code>net.IW{1}</code> 为输入层的加权矩阵, <code>net.IW{i+1}</code> 为第 i 隐层的加权矩阵	随机
<code>net.numInputs</code>	整型	输入路数, 可以由输入、输出样本点的维数自动计算出来	
<code>net.numLayers</code>	整型	隐层数, 可以由 <code>feedforwardnet()</code> 函数调用时自动确定	
<code>net.LW</code>	单元数组	输入层和隐层加权, 其中 <code>net.IW{1}</code> 为输入层的加权矩阵, <code>net.IW{i+1}</code> 为第 i 隐层的加权矩阵	随机
<code>net.trainParam.epochs</code>	整型	最大训练回合数, 当误差准则满足, 即使未训练到此步骤也将停止训练, 返回训练结果	100
<code>net.trainParam.lr</code>	实型	自学习的学习率	0.01
<code>net.trainParam.goal</code>	实型	训练误差准则, 当误差小于此值时停止训练	0
<code>net.trainFcn</code>	字符串	训练算法, 可选 'traingcf' (共轭梯度法)、'train' (批处理训练算法)、'traingdm' (带动量的梯度下降算法)、'trainlm' 等	'train'

前馈神经网络下更经常使用的两个函数是 `fitnet()` 和 `patternnet()`, 它们的调用格式与 `feedforwardnet()` 函数完全一致, 分别适用于数据拟合与模式识别。下面通过例子演示神经网络对象的建立。

例 10-17 试建立两个前馈型神经网络, 第 1 个网络含有一个隐层, 该层有 8 个节点, 第 2 个网络含有两个隐层, 其第 1 层有 4 个节点, 第 2 隐层有 6 个节点。

解 这两个前馈网络由下面的语句即可以直接建立起来

```
>> net1=fitnet(5); net2=fitnet([4 6]);
```

除了神经网络结构之外,还可以用下面的语句格式直接设定其他参数,如

```
net.trainParam.epochs = 300, net.trainFcn = 'traingdm'
```

2. 神经网络的训练与泛化

若建立了神经网络模型 **net**,则可以调用 **train()** 函数对神经网络参数进行训练。该函数的调用格式为 **[net,tr,Y₁,E]=train(net,X,Y)**,其中,变量 **X** 为 $n \times M$ 矩阵, n 为输入变量的路数, M 为样本的组数,**Y** 为 $m \times M$ 矩阵, m 为输出变量的路数,**X**、**Y** 分别存储样本点的输入和输出数据。由样本点数据进行训练,则可得出训练后的神经网络 **net**,且可以返回其他相关的内容,**tr** 为结构体数据,返回训练的相关跟踪信息,**tr.epochs** 为训练回合数,**tr.perf** 为各步训练中目标函数的值。**Y₁** 和 **E** 矩阵分别返回由神经网络计算出的输出和误差矩阵。在训练过程中将每隔 25 步自动显示一次训练指标。训练结束后还可以用 **plotperf(tr)** 语句绘制出目标值曲线。该界面将自动显示如图 10-19 所示的训练界面。除了训练出神经网络模型之外,该界面还提供了一些按钮,显示训练的中间结果曲线。

如果在给出的最大训练回合数下无法得出满足要求的网络,则将给出错误的信息提示。用户可以再调用该函数一次,这时将以上次的训练结果加权矩阵为初值继续训练,用户可以循环调用该语句。如果误差在几次循环后仍无显著改善,则说明网络结构有问题,应该修改网络结构。

神经网络训练完成后,可以利用该网络对样本区域内的其他输入量求解其输出值,这种求值的方法称为神经网络的仿真或泛化 (generalization),可以理解为利用神经网络进行数据拟合,对新的输入点数据 **X₁** 调用 **sim()** 函数进行泛化,得出这些输入点处的输出矩阵 **Y₁**,且 **Y₁ = sim(net,X₁)**。

神经网络是否成功不在于对样本点本身拟合误差的大小,关键在于其泛化效果。如果对样本点以外的其他输入点均有较好的拟合,则说明该神经网络结构合理,否则,训练出来的神经网络没有应用价值。下面将通过例子来演示神经网络及其在数据拟合中的应用及神经网络控制参数对训练的影响。

例 10-18 考虑用例 8-24 中给出的数据,试用神经网络对其进行拟合。

解 可以用下面的语句输入样本点数据,并选择前馈神经网络。设有 2 个隐层,因为最后一个隐层实际上为输出层,所以其节点个数应该与输出路数一致,故节点数为 1。现在令第 1 隐层节点个数为 5,则可以用下面的语句进行神经网络训练,得出如图 10-19 所示的训练界面。选择更密集的输入数据进行泛化,则可以得出如图 10-20 所示的泛化效果。可见,这样得出的拟合效果是令人满意的,和理论曲线之间看不出任何差异。

```
>> x=0:.5:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
x0=[0:0.1:10]; y0=0.12*exp(-0.213*x0)+0.54*exp(-0.17*x0).*sin(1.23*x0);
net=fitnet(5); net.trainParam.epochs=1000; % 设置最大步数
[net,b]=train(net,x,y); % 训练神经网络
figure; y1=sim(net,x0); plot(x,y,'o',x0,y0,x0,y1,':'); %网络泛化
```

可以用下面的语句显示出神经网络的权值

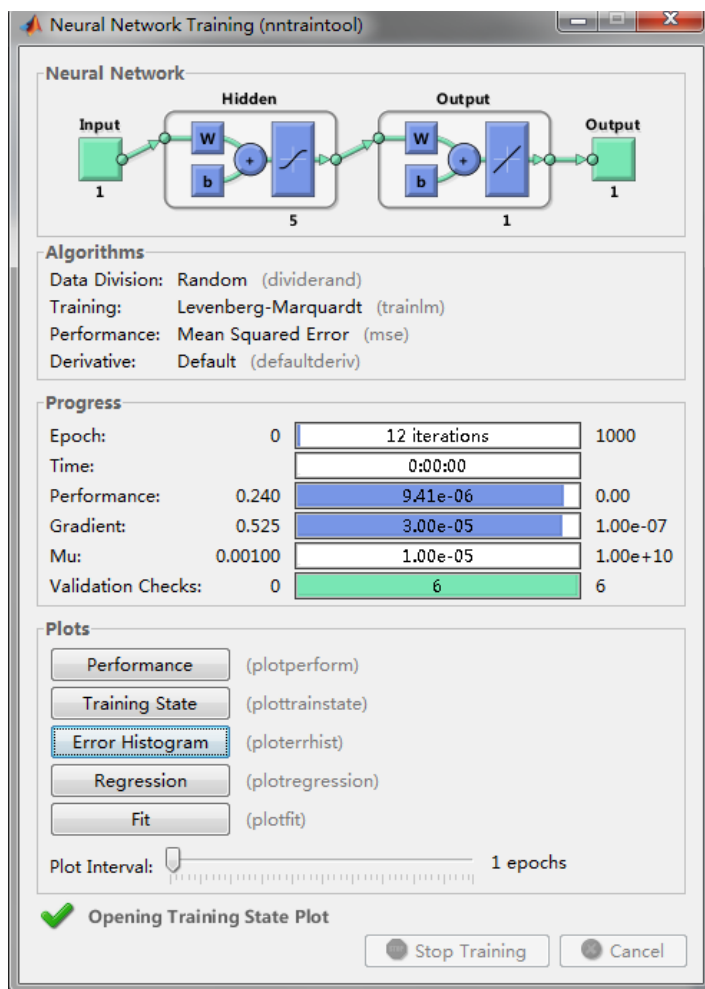


图 10-19 神经网络拟合界面

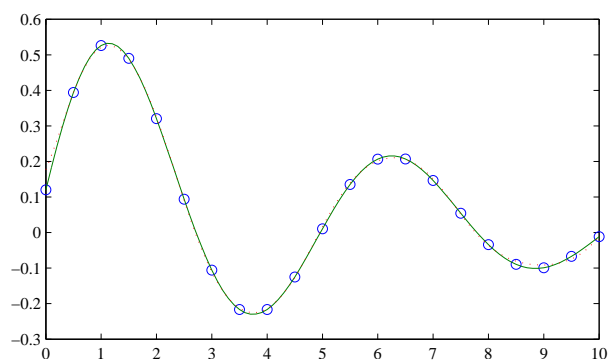


图 10-20 神经网络拟合效果

```
>> w1=net.IW{1}, w2=net.LW{2,1} % 隐层权值和输出层权值
```

得出输入层到隐层的加权为 $w_1^T = [-6.4589, -5.8420, -5.1430, 4.9068, -7.3987]$, 而隐层到输出层

的权值为 $w_2 = [-0.6343, 0.5038, -0.8231, -1.3635, -0.8936]$ 。

选择不同的训练算法,将得出不同的误差曲线,如图 10-21 (a) 所示,还可以得出拟合曲线,如图 10-21 (b) 所示。

```
>> n1=fitnet(5,'traincgf'); n1.trainParam.epochs=500; [n1,b1]=train(n1,x,y);
    n2=fitnet(5,'traingdx'); n2.trainParam.epochs=500; [n2,b2]=train(n2,x,y);
    semilogy(b. epoch,b.perf,b1. epoch,b1.perf,'--',b2. epoch,b2.perf,':')
    y2=sim(n1,x0); y3=sim(n2,x0); figure; plot(x0,y1,x0,y2,'--',x0,y3,':')
```

从训练效果看,用其他算法很多步数难以达到的训练效果用 Levenberg-Marquardt 算法可以较少步就能得出满意的效果。其他训练算法得出的拟合效果也不是很理想,所以建议采用默认的 Levenberg-Marquardt 训练算法。

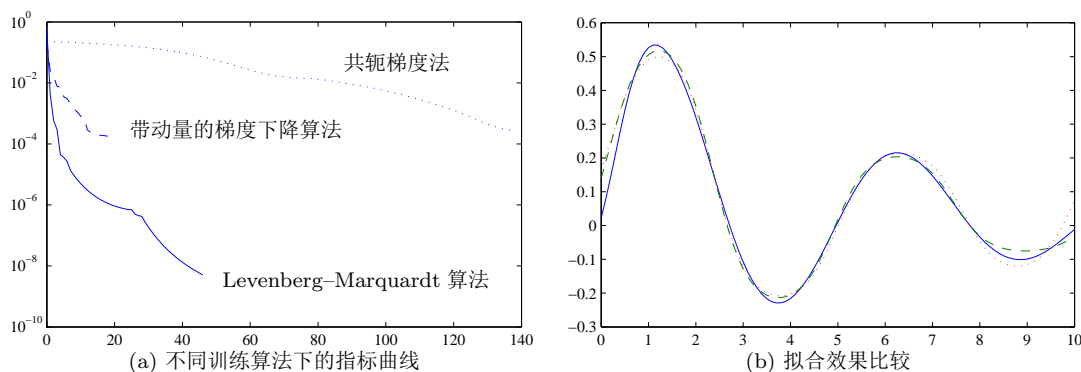


图 10-21 不同训练算法及拟合结果

若增加隐层节点个数,例如增加到 15,则可以重新建立神经网络,并进行训练,从训练结果看,只用 50 步就能得出极小的拟合误差,如图 10-22 (a) 所示。得出的曲线拟合结果如图 10-22 (b) 所示。

```
>> n3=fitnet(15); n3.trainParam.epochs=100; [n3,b3]=train(n3,x,y);
    semilogy(b3. epoch,b3.perf); figure; y4=sim(n3,x0); plot(x0,y4,x,y,'o')
```

从得出的拟合结果看,似乎无限增大隐层节点个数就可以改进拟合效果。其实不然,增大节点个数会改善对样本点的拟合,但对其他点的函数拟合将得出如图 10-22 (b) 所示的结果,亦即神经网络泛化出现了问题,故不能无限增加节点个数。不过节点个数如何选择至今没有公认的解析方法,只能根据实际情况用试凑方式选择。

例 10-19 试用神经网络对例 8-7 中给出的二元函数进行曲面拟合。

解 先考虑用下面的语句输入样本数据,选择两个隐层网络,每层均有 10 个节点,这样就可以用下面的语句对该网络进行训练,并得出如图 10-23 (a) 所示的泛化结果。

```
>> [x,y]=meshgrid(-3:.6:3, -2:.4:2); x=x(:)'; y=y(:)';
    z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); % 注意这三个变量均应为行向量
    net=fitnet([10,10]); net.trainParam.epochs=1000;
    [net,b]=train(net,[x; y],z); % 训练神经网络
    [x2,y2]=meshgrid(-3:.1:3, -2:.1:2); x1=x2(:)'; y1=y2(:)';
    figure; z1=sim(net,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
```

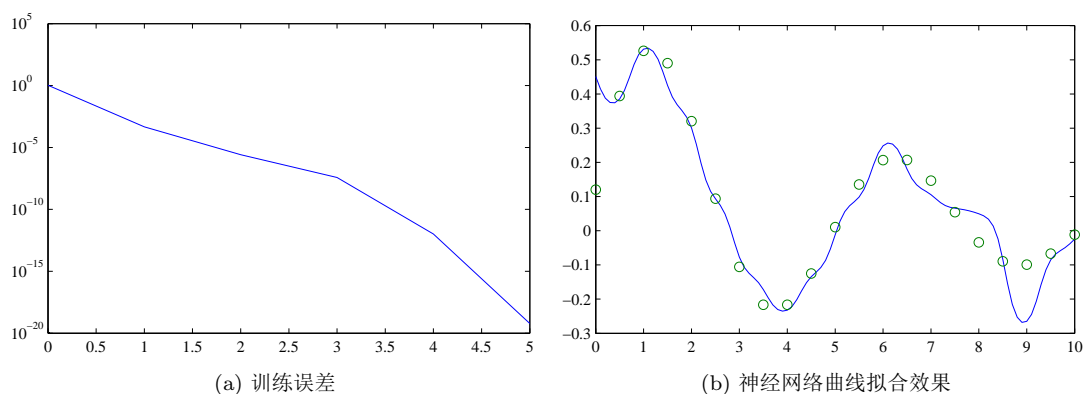


图 10-22 隐层节点个数选为 15 时神经网络拟合效果

这样的泛化效果不是很理想,部分点处有较大的波动,现在设定两个隐层都有 20 个节点,则可以得出如图 10-23 (b) 所示的泛化效果。可见,泛化效果恶化,说明节点数选择过多。从总体拟合效果看,神经网络直接拟合效果比前面介绍的插值方法差。

```
>> net=fitnet([20,20]); [net,b]=train(net,[x; y],z); % 训练神经网络
figure; z1=sim(net,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
```

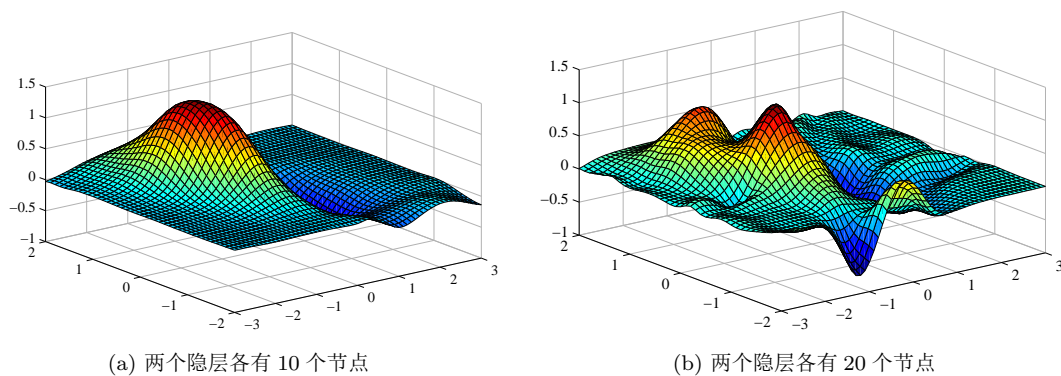


图 10-23 不同结构的拟合效果

如果选择单隐层网络或三隐层的网络,则可以得出如图 10-24 (a)、(b) 所示的拟合效果。

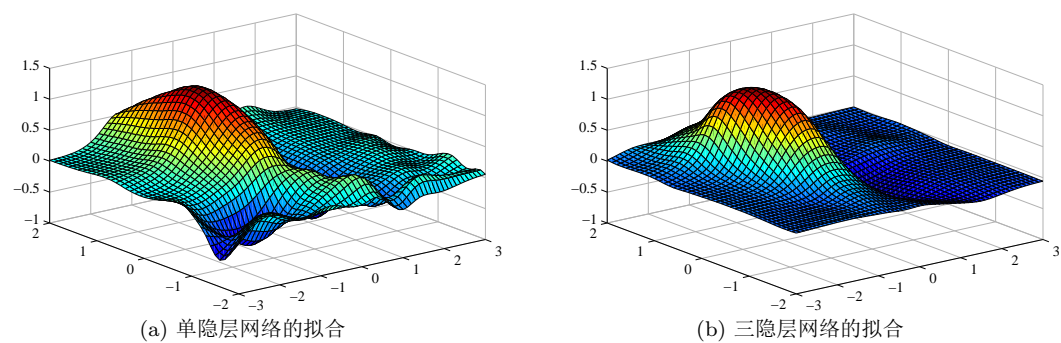


图 10-24 神经网络的二元函数拟合结果

```
>> net=fitnet(25); net.trainParam.epochs=1000; [net,b]=train(net,[x; y],z);
figure; z1=sim(net,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
N=fitnet([10 10 5]); N.trainParam.epochs=1000; [N,b]=train(N,[x; y],z);
figure; z1=sim(N,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
```

在神经网络研究中,经常有人引用这样的理论结果,即三层 BP 网络可以按任意精度逼近给定函数。不过从这样普通的例子看,节点个数的选择对拟合效果至关重要,而节点个数究竟如何选择至今没有任何公认的方法,只能通过试凑的方法去选择。本例试用了神经网络工具箱中提供的全部训练算法,也无法得到接近于样条插值拟合效果的神经网络模型。

10.3.2 径向基网络结构与应用

径向基函数(radial basis function, RBF)是一类特殊的指数函数,其数学描述为

$$\psi(\mathbf{x}) = e^{-b\|\mathbf{x}-\mathbf{c}\|} = e^{-b(\mathbf{x}-\mathbf{c})^T(\mathbf{x}-\mathbf{c})} \quad (10-3-7)$$

其中, \mathbf{c} 为聚类中心点,而 $b>0$ 为调节聚类效果的参数。在神经网络工具箱中,`radbas()`函数可以计算出标准径向基函数 $y_i = e^{-x_i^2}$ 的曲线参数,而简单的计算语句能求出式(10-3-7)中的对应关系。

径向基网络是一类特殊的神经网络结构。考虑图 10-18 中给出的一般三层前馈网络结构,如果隐层的传输函数 $F_1(x)$ 为径向基函数,输出层的传输函数 $F_2(x)$ 为线性函数,则此结构的网络称为径向基网络。

例 10-20 试绘制不同参数 (c, b) 下的径向基函数曲线。

解 取中心点 $c = -2, 0, 2$,并假设 $b = 1$,则径向基函数曲线可以由下面的语句得出,如图 10-25 (a)所示。可见,这些曲线形状完全相同,不同的是它们有 c 单位的平移。

```
>> x=-4:0.1:4; cc=[-2,0,2]; b=1;
for c=cc, y=exp(-b*(x-c).^2); plot(x,y); hold on; end
```

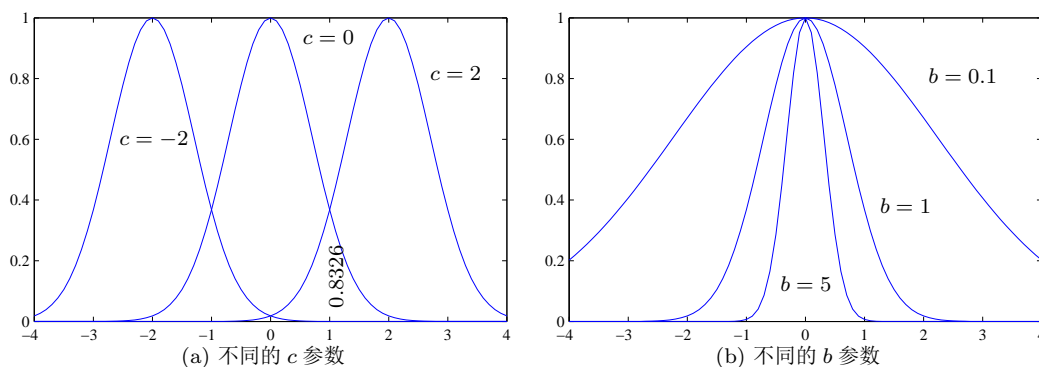


图 10-25 不同参数下径向基函数曲线

选择中心点 $c = 0$,并假设 $b = [0.1, 1, 5]$,则径向基函数曲线如图 10-25 (b)所示。

```
>> x=-4:0.1:4; bb=[0.1,1,5]; c=0;
for b=bb, y=exp(-b*(x-c).^2); plot(x,y); hold on; end
```

虽然网络结构都是前馈型的,但径向基网络的训练方式不是采用反向误差传播实现的,所以径向基网络不是 BP 网络。从 MATLAB 的神经网络工具箱使用看,径向基网络的使用要简单得多,只用两个函数 `newrbe()` 和 `sim()` 就可以实现神经网络的建立、训练和泛化全过程。下面还是通过例子来演示这些函数的使用。

例 10-21 考虑用例 10-18 中给出的数据,试用径向基神经网络对其进行拟合。

解 用下面的语句输入样本点数据,用简单的语句对其泛化,立即可以得出如图 10-26 所示的泛化效果。可见,这样得出的拟合效果是令人满意的,和理论曲线之间看不出任何差异。从曲线拟合数据效果看,RBF 网络明显优于前面介绍的 BP 网络。这里采用的隐层节点个数为 21,两层的权值可以由 `net.IW{1}` 和 `net.LW{1}` 读出。

```
>> x=0:.5:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
    x0=[0:0.1:10]; y0=0.12*exp(-0.213*x0)+0.54*exp(-0.17*x0).*sin(1.23*x0);
    net=newrbe(x,y); y1=sim(net,x0); % 建立、训练并泛化神经网络
    plot(x,y,'o',x0,y0,x0,y1,':');
```

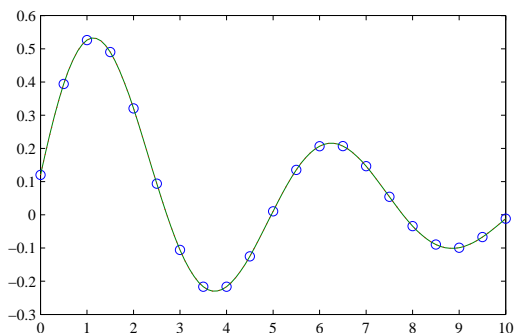


图 10-26 一维曲线拟合效果

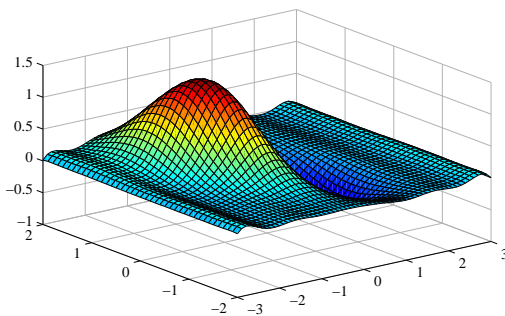


图 10-27 二维曲面拟合效果

例 10-22 试用 RBF 神经网络拟合例 10-19 中的二维曲面。

解 前面曾经演示过,若采用一般 BP 网络,不能很好地拟合本例中的二维曲面。下面给出用 RBF 网络结构重新进行拟合,可以给出下面的语句,得出的拟合效果在图 10-27 给出。可见,这样得出的拟合效果远远优于 BP 网络,虽然略差于二维样条插值效果。径向基函数网络自动选择的隐层节点个数为 121。

```
>> [x,y]=meshgrid(-3:.6:3, -2:.4:2); x=x(:)'; y=y(:)';
    z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); % 注意这三个变量均应为行向量
    net=newrbe([x; y],z); % 建立径向基网络模型
    [x2,y2]=meshgrid(-3:.1:3, -2:.1:2); x1=x2(:)'; y1=y2(:)';
    z1=sim(net,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
```


10.3.3 神经网络界面

MATLAB 的神经网络工具箱提供了一个可以直接使用的程序。在 MATLAB 命令窗口中给出 `nntool` 命令, 打开一个如图 10-28 所示的图形用户界面, 可以用该界面建立所需的神经网络模型, 并可以由已知数据对该网络进行训练、仿真。下面将通过例子演示神经网络图形界面的使用。

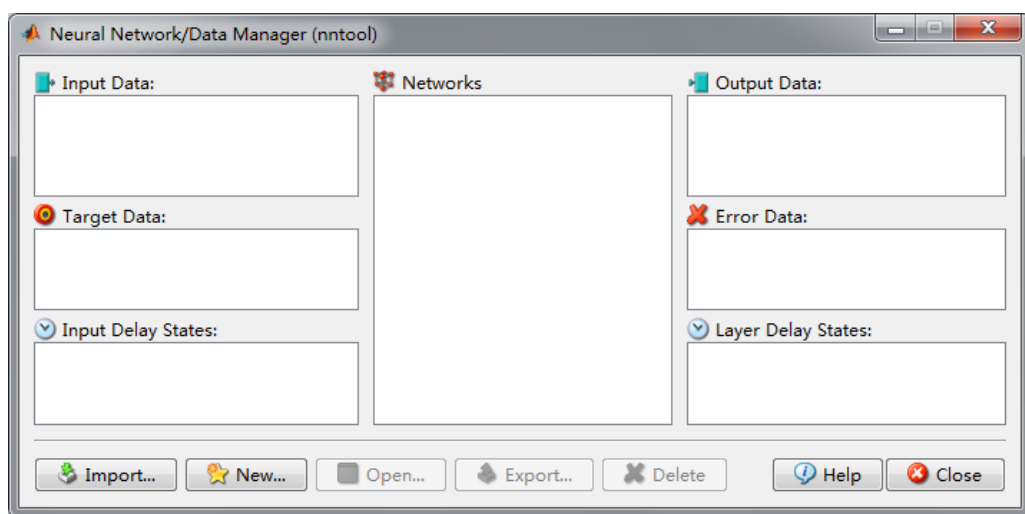


图 10-28 神经网络应用界面

例 10-23 考虑用例 10-18 中给出的数据和神经网络拟合效果, 试利用神经网络工具箱的 `nntool` 界面完成同样的拟合。

解 先输入已知数据, 然后启动 `nntool`, 得出如图 10-28 所示的程序界面。可以通过这个界面对给定的数据进行神经网络拟合。

```
>> x=0:.5:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);  
x0=[0:0.1:10]; y0=0.12*exp(-0.213*x0)+0.54*exp(-0.17*x0).*sin(1.23*x0);  
nntool    % 启动神经网络拟合界面
```

如果想利用神经网络拟合系统, 首先需要将数据输入到此界面。单击 `Import` 按钮将弹出如图 10-29 所示的对话框。将中间栏目下的 `x`、`x0` 两个现有的 MATLAB 工作空间变量作为输入变量 (右面栏目的 `Inputs`) 输入, 将变量 `y` 作为目标变量 (亦即 `Targets`) 输入到界面中。

导入了所需的数据, 单击主界面中的 `New Network` 按钮来选择神经网络的结构, 这样得出如图 10-30 所示的界面。其中默认的网络结构是前馈型反向传播网络结构 (Feedforward Backprop)。保持各个默认的网络结构, 将网络层数 (Number of layers) 设置成 2, 在下面的列表框中分别选择不同的节点数, 第 1 层选择 8 个节点数, 第 2 层选择 1 个节点。在第 1 层中选择传输函数 (Transfer Function) 为 `Logsig`, 第 2 层选择传输函数为 `Pureline`, 单击 `Create` 按钮就可以确定神经网络结构, 关闭此窗口。

神经网络结构确定后, 单击 `View` 即可以显示神经网络结构, 如图 10-31 所示。

现在可以训练神经网络了。单击 `Train` 按钮, 得出如图 10-32 所示的对话框。在对话框中需要输入

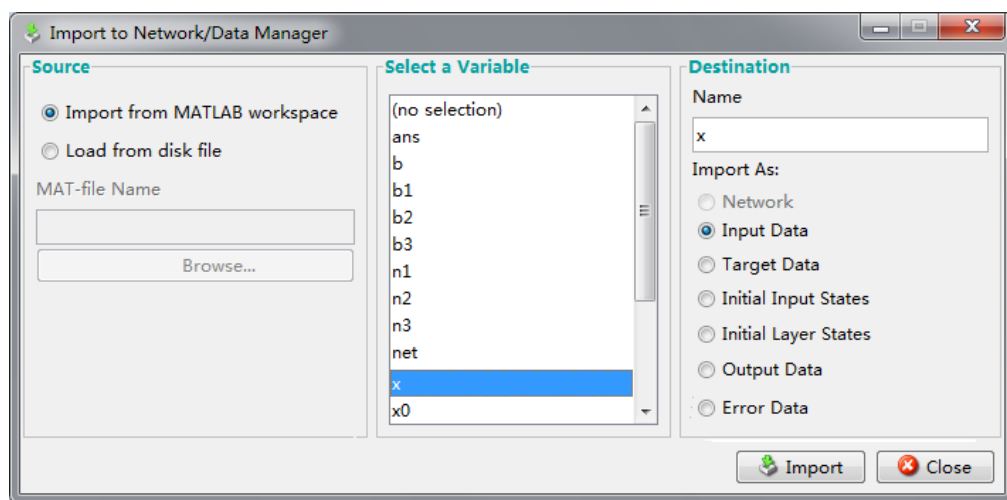


图 10-29 数据输入界面

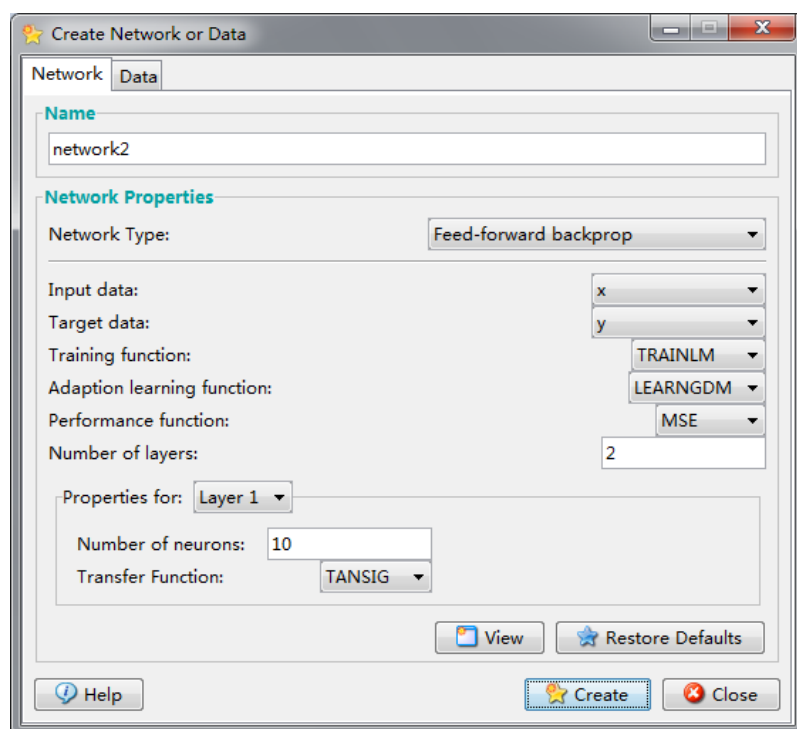


图 10-30 神经网络结构设置对话框

训练用已知数据,如在 Inputs 和 Targets 栏目分别填写 x 和 y。另外,还可以单击 Training Parameters 标签指定神经网络训练的控制参数,得出如图 10-33 所示的对话框,例如将训练回合数 epochs 设置为 1000,这时程序将自动训练网络参数,当误差指标满足时会自动停止训练,得出所需的参数。

单击 Train Network 按钮就可以开始训练,得出如图 10-21 (a) 所示的训练误差曲线。可见,这样的训练误差很小,将训练得出的网络模型输出 (Export) 到 MATLAB 环境中,这时将由下面的语句

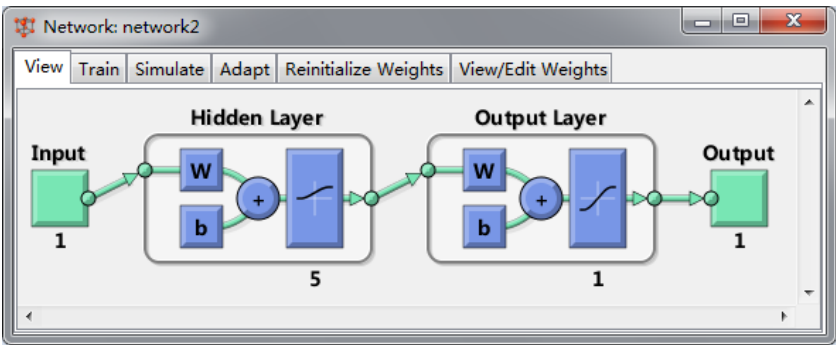


图 10-31 神经网络结构

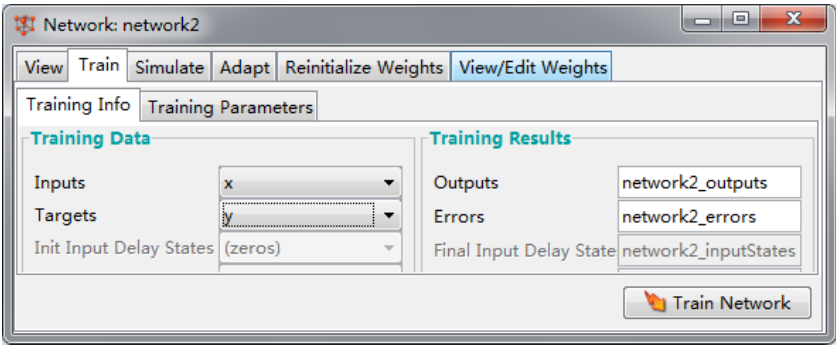


图 10-32 神经网络训练参数设置对话框

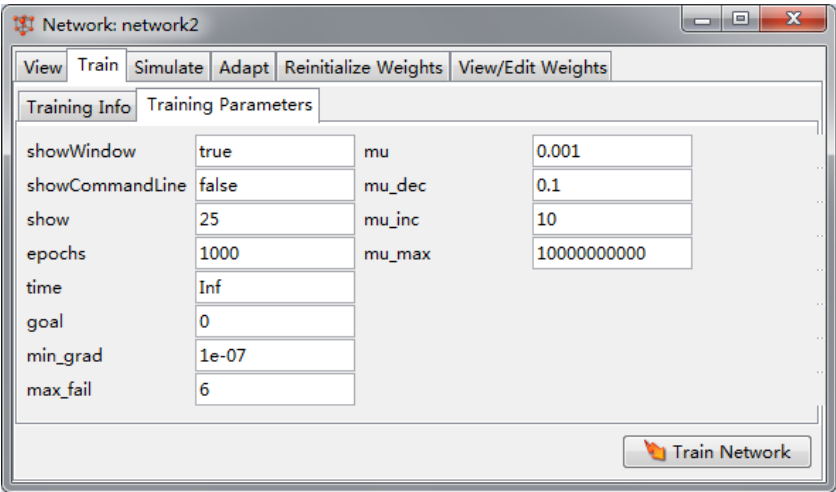


图 10-33 网络训练控制参数对话框

绘制出曲线拟合与泛化结果,如图 10-21 (b) 所示。

```
>> y1=sim(network1,x0); plot(x,y,'o',x0,y0,x0,y1,':')
```

10.4 进化算法及其在最优化问题中的应用

遗传算法是基于进化论在计算机上模拟生命进化机制而发展起来的一类新的最优化算法,它根据适者生存、优胜劣汰等自然进化规则搜索和计算问题的解^[9]。该领域最早是由美国 Michigan 大学的 John Holland 于 1975 年提出的。遗传算法的基本思想是,从一个代表最优化问题解的一组初值开始进行搜索,这组解称为一个种群,种群由一定数量、通过基因编码的个体组成,其中每一个个体称为染色体,不同个体通过染色体的复制、交叉或变异又生成新的个体,依照适者生存的规则,个体也在一代一代进化,通过若干代的进化最终得出条件最优的个体。本节以一个较好的 MATLAB 工具箱——遗传算法最优化工具箱为主要工具,先介绍遗传算法的基本概念,然后通过例子介绍其在求解最优化问题中的应用。本节还将介绍粒子群优化算法及应用。

10.4.1 遗传算法的基本概念及 MATLAB 实现

MATLAB 提供了全局优化工具箱(早期版本名为遗传算法与直接搜索工具箱),可以较好地解决与遗传算法相关的各类问题。但早期版本没有官方的工具箱,只有两个基于 MATLAB 语言的免费遗传算法工具箱。其一,英国 Sheffield 大学自动控制与系统工程系 Peter Fleming 教授与 Andrew Chipperfield 开发的遗传算法工具箱,实现了各种基本运算,说明书齐全,调用格式更类似于最优化工具箱中的函数;另一个,美国北 Carolina 州立大学 Christopher Houck、Jeffery Joines 和 Michael Kay 开发的遗传算法最优化工具箱(GAOT),其函数 `gaopt()` 可以直接解决最优化问题^①。由于 GAOT 工具箱流传较广,已经能比较容易地解决最优化问题,所以这里还是建议采用该免费工具箱来解决基于遗传算法的最优化问题。

简单遗传算法的一般步骤为:

(1) 选择 N 个个体构成初始种群 P_0 ,并求出种群内各个个体的函数值。染色体可以用二进制数组表示,也可以用实数数组来表示,种群可以由随机数生成函数建立。其实使用遗传算法求解函数 `gaopt()`,则会自动生成所需的初始种群 P_0 。

(2) 设置代数 $i = 1$,即设置其为第 1 代。

(3) 计算选择函数的值,所谓选择即通过概率的形式从种群中选择若干个个体的方式。遗传算法最优化工具箱提供了 3 个选择函数,其中 `roulette()` 实现轮盘选择算法,`normGeomSelect()` 函数实现归一化几何选择方法,`tournSelect()` 实现锦标赛形式的选择方式。`normGeomSelect()` 函数为默认选择函数。

(4) 通过染色体个体基因的复制、交叉、变异等创造新的个体,构成新的种群 P_{i+1} ,其中复制、交叉和变异都有相应的 MATLAB 函数,`gaopt()` 函数选择其中默认的方法进行这样的处理,构成新的种群。

(5) $i = i + 1$,若终止条件不满足,则转移到步骤(3)继续进化处理。

^① 该工具箱的主函数名为 `ga()`,但该函数名与 MATLAB 的全局优化工具箱中主函数同名,故这里将其改名为 `gaopt()`,本书所附的代码也做了相应的修改。

和传统最优化算法比较,遗传算法主要有以下几点不同^[10]:

(1) 不同于从一个点开始搜索最优解的传统的最优化算法。遗传算法从一个种群开始对问题的最优解进行并行搜索,所以更利于全局最优解的搜索,但遗传算法需要指定各个自变量的范围,而不像最优化工具箱中可以使用无穷区间的概念。

(2) 遗传算法并不依赖于导数信息或其他辅助信息来进行最优解搜索,而只由目标函数和对应于目标函数的适应度水平来确定搜索的方向。

(3) 遗传算法采用的是概率性规则而不是确定性规则,所以每次得出的结果不一定完全相同,有时甚至会有较大的差异。

本书研究的遗传算法使用的种群和个体表示均采用实值,这和经典遗传算法中采用的编码二进制值是不同的,所以无需对其进行编码和解码运算,且其求解问题的精度比二进制编码形式要高^[11]。

10.4.2 遗传算法在求解最优化问题中的应用举例

这里将首先介绍遗传算法最优化工具箱中的 `gaopt()` 函数在求解最优化问题中的应用。之所以使用该函数是因为该函数调用简单。即使对遗传算法理解不多,甚至不知道染色体如何选择、如何进行交叉和变异、如何进行选择等关于遗传算法的最基本知识,但利用 MATLAB 语言描述出目标函数,就可以得出最优解。和最优化工具箱不同,`gaopt()` 函数能求解的问题是最大值问题,所以在编写目标函数时应该注意。`gaopt()` 函数最基本的调用格式为 `[a,b,c] = gaopt(bound,fun)`,其中,`bound = [xm,xM]` 为求解区间下界 x_m 和上界 x_M 构成的矩阵,`fun` 为字符串,表示用户编写的目标函数,其写法与最优化工具箱的目标函数写法相近,但结构不完全相同,后面将通过例子来描述。返回的 `a` 为搜索的结果向量,由搜索出的最优 `x` 向量与目标函数构成。`b` 为搜索的最终种群。`c` 为搜索中间过程参数表,其第 1 列为代数,后面各列分别为该代最好的个体与目标函数的值,可以认为是寻优的中间结果。当然该函数还有其他的调用格式,例如需要用户自己声明遗传算法的编码、选择等信息,设定初始种群、交叉率、变异率,还可以选择最大的代数。

MATLAB 的全局优化工具箱也提供了遗传算法求解函数 `ga()`,其调用格式与最优化工具箱中的相应函数类似,可以同样利用遗传算法处理最优化问题。该函数的调用格式为 `[x,f,flag,out] = ga(fun,n,opts)`,其中,`fun` 为描述目标函数的 MATLAB 函数,其格式与最优化工具箱一致,`n` 为自变量个数,`opts` 为遗传算法控制选项,可以调用 `gaoptimset()` 函数设置各种选项。例如,用其 `Generations` 属性可以设定最大允许的代数,`InitialPopulation` 属性可以设置初始种群,用 `PopulationSize` 属性可以给定种群的规模,用 `SelectionFcn` 属性可以定义选择函数等。函数调用结束后,返回的 `x` 为搜索的结果,若返回的 `flag` 大于 0,则表示求解成功,否则求解出现问题。

全局优化工具箱还提供了 `optimtool()` 函数(早期版本 `gatoool()`),该函数将打开如图 10-34 所示的遗传算法程序界面。可以用可视的方式设置遗传算法的各类参数,如初始种群、交叉变异、选择函数等可以通过列表框与编辑框直接填写。单击 `Start` 按钮就可以开始遗传算法搜索,得出最优结果。

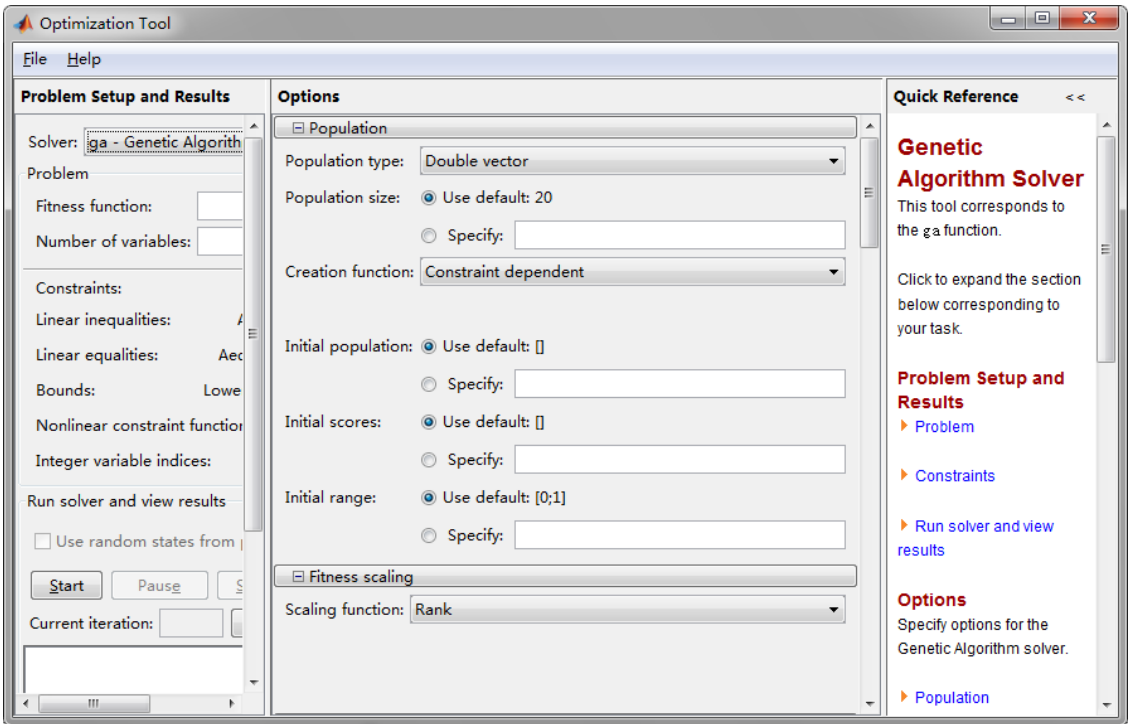


图 10-34 遗传算法工具箱的设计界面

例 10-24 考虑一个简单的一元函数最优化问题求解, $f(x) = x \sin 10\pi x + 2$, $x \in (-1, 2)$, 试求出 $f(x)$ 取最大值时 x 的值。

解 用下面的语句可以绘制出求解区间内的目标函数的曲线, 如图 10-35 所示。可以看出, 该曲线为振荡曲线, 存在很多极值点。

```
>> ezplot('x*sin(10*pi*x)+2', [-1,2])
```

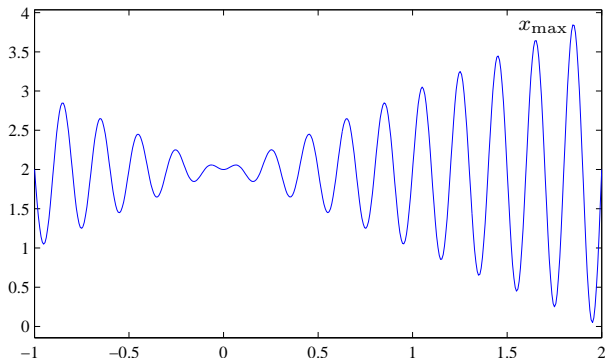


图 10-35 目标函数的曲线表示

因为最优化工具箱的搜索函数需要给出初值, 所以对不同的初值可能得出不同的搜索结果, 例如可以给出如下的语句试测不同初值, 得出的结果如表 10-8 所示。

```
>> f=@(x)-x.*sin(10*pi*x)-2; v=[]; xx=[-1:0.8:1.5,1.5:0.1:2];
    for x=xx, x1=fmincon(f,x,[],[],[],[],-1,2); v=[v; x,x1,f(x1)]; end
```

可见,随意选择一个初值很难得出全局最优解,故用传统的寻优方式不一定能得出满意的结果。

表 10-8 不同初值 x_0 下搜索到的最优解及目标函数值

x_0	搜索解 x_1	目标函数 $f(x_1)$	x_0	搜索解 x_1	目标函数 $f(x_1)$	x_0	搜索解 x_1	目标函数 $f(x_1)$
-1	-1	-2	1.4	1.450698306	-3.450349229	1.7	1.250809692	-3.250405044
-0.2	-0.6515538221	-2.650777689	1.5	0.2539684622	-2.251997259	1.8	1.850547452	-3.850273767
0.6	0.6515537909	-2.650777689	1.6	1.65061375	-3.650306929	1.9	0.4522326498	-2.451120675

利用遗传算法函数 `gaopt()`,完全选择默认选项,则可以编写一个文件描述目标函数如下

```
function [sol,y]=c10mga1(sol,options)
x=sol(1); y=x.*sin(10*pi*x)+2;
```

这样,完全用默认参数调用遗传算法工具箱中的 `gaopt()` 函数,而不对其编码等做任何指定,则可以得出结果为^① $x = 1.8506$, $f = 3.8503$ 。

```
>> [a,b,c,d]=gaopt([-1,2], 'c10mga1'); a,c
```

其 c 参数单独显式的可读性不佳,可以将其表示成表 10-9 的形式。可见,对此例来说,从第 13 代开始就可以得出较精确的结果,通过 100 代的搜索,可以得出相当高精度的寻优结果。

表 10-9 遗传算法搜索中间结果

代	x	$f(x)$	代	x	$f(x)$	代	x	$f(x)$
1	1.866429742	3.623276856	8	1.855582779	3.827116063	19	1.850630071	3.850267532
2	1.857329778	3.808304445	9	1.855325444	3.829420164	21	1.850569848	3.850273309
3	1.856754984	3.815102341	10	1.846815256	3.837579341	⋮	⋮	⋮
5	1.856200451	3.821095566	11	1.852559605	3.84657337	98	1.850547236	3.850273767
7	1.855683776	3.826178928	13	1.850438152	3.85026285	100	1.850547466	3.850273767

由最优化工具箱中的 `fmincon()` 可知,初值选择为 1.8 是较精确的结果。这样,可以由下面的语句比较由该初值得出的最优解与遗传算法工具箱默认参数得出的最优解。

```
>> x0=1.8; x1=fmincon(f,x0,[],[],[],[],-1,2,'',ff); f1=f(x1), f2=f(a(1))
```

从比较得出的结果可见,遗传算法得出的 x 值对应的函数值更小一些,因此可以认为该方法对本例中的函数更适用。

现在考虑一个更大的求解空间,假设 $x \in (-1, 20)$, 可以用 `ezplot()` 函数绘制出目标函数的曲线,如图 10-36 (a) 所示。用最优化工具箱中的 `fmincon()` 函数将更难得出全局最优解,因为不能容易地给出搜索的初值。但用遗传算法则没有这样的问题,只需给出如下语句即可得出 $x = 19.4501$, $f = 21.4500$ 。

```
>> [a,b,c,d]=gaopt([-1,20], 'c10mga1')
```

^① 前面已经指出,遗传算法中有随机性因素,所以调用该算法每次得出的结果均不相同,但大的趋势应该是一样的。阅读本部分时不要指望实际得出的结果和书上的完全一致。

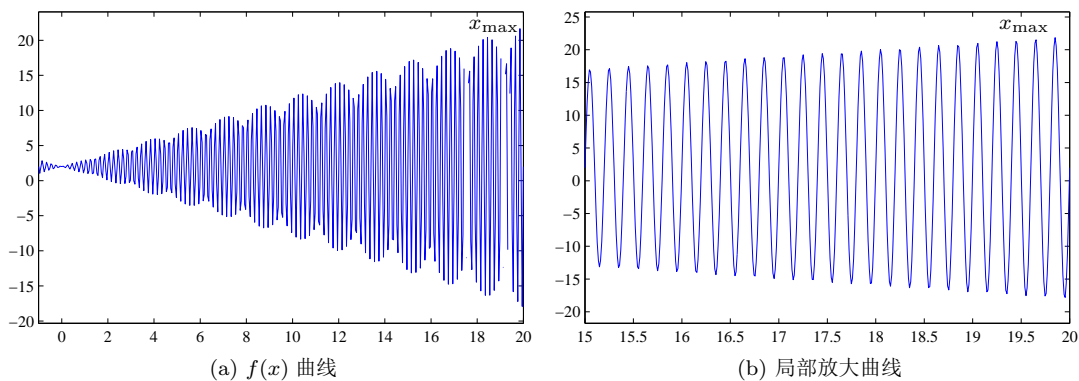


图 10-36 变换区间后的目标函数曲线

其 c 参数可以表示成表 10-10 的形式。

表 10-10 遗传算法搜索中间结果

代	x	$f(x)$	代	x	$f(x)$	代	x	$f(x)$
1	18.04650679	19.93794526	13	18.8529686	20.77103967	29	19.45008417	21.45001617
3	18.05080536	20.04502802	17	19.05252045	20.99282365	40	19.45004021	21.45002469
6	18.05063177	20.04707655	20	19.0522703	21.00383083	44	19.45005193	21.45002605
11	18.45004062	20.4500256	22	19.45056616	21.44748956	100	19.45005208	21.45002605

其实,将该曲线局部放大,将得出如图 10-36 (b) 所示的表示形式。可见,这个区间内使得函数取最大值的 x 值应该大于 19.5,所以这样用遗传算法得出的并非全局最优值,而为次最优值。由此可见,遗传算法也不能保证获取全局最优值。

现在改变遗传算法的求解区间,则可以由如下命令得出 $x = 19.8501, f = 21.8500$ 。寻优过程的中间结果在表 10-11 中给出。

```
>> [a,b,c,d]=gaopt([12,20], 'c10mga1')
```

表 10-11 遗传算法搜索中间结果

代	x	$f(x)$	代	x	$f(x)$	代	x	$f(x)$
1	19.04416318	20.72488654	15	19.45002088	21.4500167	35	19.85009565	21.85000603
6	19.447683	21.39618429	17	19.84813102	21.81392715	37	19.8500877	21.85001236
10	19.44775156	21.39925388	19	19.84823094	21.81758539	43	19.85007289	21.85002085
11	19.45087121	21.44358629	20	19.84985188	21.84963699	56	19.85006859	21.8500225
12	19.4495212	21.44732089	27	19.85024638	21.84965177	63	19.85005257	21.8500255
13	19.44965169	21.44848728	28	19.84987418	21.84971911	100	19.85005104	21.85002552
14	19.44974647	21.44912956	29	19.85012215	21.84997599			

例 10-25 试求函数 $f(x) = (x_1 + x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$ 的最小值。

解 可以按照遗传算法工具箱编写如下函数,描述最优化问题的目标函数。注意,这里应该描述成目标函数的最大值。

```
function [S,f]=c10mga3(S,options)
```



```
x=S(1:4); f=-(x(1)+x(2))^2-5*(x(3)-x(4))^2-(x(2)-2*x(3))^4-10*(x(1)-x(4))^4;
```

使用遗传算法函数 `gaopt()`, 并设定自变量的求解范围为 $-1 \leq x_i \leq 1, i = 1, 2, 3, 4$, 则可以由下面的函数求解最优化问题, 得出如下的结果, 且部分中间结果如表 10-12 所示。事实上, 该函数的精确解为 $x_1 = x_2 = x_3 = x_4 = 0$, 但用遗传算法不能搜索到该点, 只能得出次最优值 $x = [0.05512, -0.05428, 0.01487, 0.01509]$, 这时的目标函数值为 $f = -0.000076$ 。

```
>> [a,b,c,d]=gaopt([-1,1; -1 1; -1 1; -1 1], 'c10mga3'); x=a(1:4), f=a(5)
```

表 10-12 遗传算法搜索部分中间结果

代	x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
1	-0.58664839284	0.60535582738	0.028407550371	-0.073658816991	-0.83549958111
4	-0.01182138153	-0.048357871289	0.21168424008	0.30791546876	-0.20395460867
9	-0.02498777978	-0.033384575447	0.11281202834	0.10206772219	-0.011090989881
13	-0.02498777978	0.0042179390678	0.11281202834	0.10206772219	-0.0060176098015
18	-0.0083633004824	0.013245084818	0.02411260887	0.023000817182	-4.118625464910 ⁻⁵
24	-0.0082541665169	0.0050231349179	0.02411260887	0.023000817182	-2.964632067910 ⁻⁵
28	-0.0082541665169	0.008935952083	0.02411260887	0.023000817182	-1.857090777210 ⁻⁵
37	-0.00067214193564	0.0015648793181	0.02398676301	0.024507332569	-1.081027149610 ⁻⁵
40	-0.002743469815	0.0034270462778	0.024021142751	0.024095768707	-9.646255329910 ⁻⁶
83	-0.00079248828595	0.0026278066001	0.021397761566	0.02118842351	-8.525152175210 ⁻⁶
85	-0.0015381099099	0.0027951007597	0.021397761566	0.02118842351	-7.02692040410 ⁻⁶
89	-0.0019644292217	0.0015052840869	0.019625896238	0.019461658039	-4.483255974310 ⁻⁶
100	-0.0017892249183	0.0019037994674	0.019465754048	0.019467860996	-3.934738981710 ⁻⁶

上述求解结果显然误差很大, 其最主要的原因是最大允许的代数(默认值为 100)太小。另外, 求解的区间太小, 使得得出解的可信度降低。也就是说, 允许的求解区域是 $[-1, 1]$, 如果选择更大的求解区域将得出更不精确的结果。

现在考虑 `gaopt()` 函数稍复杂一点的调用格式, 如下:

```
[x,b,c,d] = gaopt(bound,fun,p,v,P0,fun1,n)
```

其中, p 可以给目标函数增加附加参数, 这些参数必须和目标函数对应, v 为精度及显示控制向量, P_0 为初始种群, `fun1` 为终止函数的名称, 默认值为 'maxGenTerm', n 为最大的允许代数。当然还有其他调用参数的用户设置格式, 如选择函数及参数、变异函数及参数等, 在这里不具体介绍, 读者可以参见文献 [11]。

例 10-26 仍考虑例 10-25 中给出的最优化问题, 试设置更多的允许代数, 观察寻优的结果, 和最优化搜索算法得出结果比较在精度上的差异。

解 考虑新的求解区域, $-1000 \leq x_i \leq 1000$, 由于求解范围增大, 采用默认的代数 100 会有问题, 所以可以考虑用新介绍的求解格式。显然, 目标函数由 x 就能唯一确定, 而无需附加参数, 故令 $p=[]$ 即可。同样, 因为不想改变初始种群, 控制向量等, 可以将它们设置成空向量。这样, 若想指定 2000 代为最大进化代数, 则可以给出如下的命令, 得出的中间结果如表 10-13 所示。下面语句搜索到的结果为 $x = [-0.00094, 0.00084, -0.01565, -0.01569]$ 。耗时大约 29 s。

```
>> tic, xmM=[-ones(4,1),ones(4,1)]*1000;
[a,b,c,d]=gaopt(xmM, 'c10mga3', [], [], [], 'maxGenTerm', 2000);
```

```
x=a(1:4), dd=[c(1:100:end,:); c(end,:)], toc
```

表 10-13 遗传算法搜索部分中间结果

代	x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
1	299.261366622	-139.695834899	-75.5914398821	272.888321091	-5487787.82563
211	259.334708142	-133.958552709	-63.9282289435	255.140011267	-529224.38675
418	207.950553013	-107.100526442	-50.7673718192	204.4965359	-338351.978321
633	183.944398697	121.408588645	63.0882570778	183.712206401	-166507.956154
828	30.5439763104	62.3298177457	31.7096211527	32.1807061178	-8699.82405036
1012	25.9003394774	41.1500210753	21.4199283474	24.9266965028	-4574.37877164
1211	19.218615649	26.8881841847	14.8795410924	20.0214636561	-2330.11994456
1390	6.33506176436	8.75276639595	5.30792816069	5.98777957032	-242.14756723
1559	0.317019478936	0.133356773142	0.42062335921	0.624193867527	-0.750183675915
1825	0.00172291111621	-0.000400676613735	-0.0233512636645	-0.0234295772108	-1.03776×10^{-5}
2000	-0.000937885010771	0.000843474097882	-0.0156533824812	-0.0156919023378	-1.55859×10^{-6}

可见,在 1390 代左右的误差都一直很大,所以用默认的 100 代处理不了这样大求解区间的问题,必须增大代数,然而若达到较高求解精度,则再进一步增加最大允许的代数也不会显著改善求解的精度了。

利用 MATLAB 的遗传算法求解函数 $ga()$,则可以编写出如下的目标函数。这样,调用 $ga()$ 函数可以得出结果为 $x = [0.009578, -0.01671, 0.01027, 0.01073]$ 。

```
>> f=@(x)(x(1)+x(2))^2+5*(x(3)-x(4))^2+(x(2)-2*x(3))^4+10*(x(1)-x(4))^4;
x=ga(f,4)
```

还可以人为指定一些搜索参数,如进化代数选择为 2000,种群选择为 80 个体,而交叉函数选择为启发式算法等,可以得出更精确的结果为 $x = [-0.23, 0.23, -0.94, -0.94] \times 10^{-3}$,耗时 7s。

```
>> ff=gaoptimset; ff.Generations=2000; ff.PopulationSize=80;
ff.CrossoverFcn=@crossoverheuristic; x=ga(f,4,ff)
```

现在考虑用第 6 章的无约束最优化求解功能,可以由如下的命令直接求出原问题的最优解为 $x = [0.0304, -0.0304, -0.7534, -0.7534] \times 10^{-6}$,耗时仅为 0.75s,其计算精度明显高于遗传算法的结果,且求解时间大大减少。

```
>> f=@(x)(x(1)+x(2))^2+5*(x(3)-x(4))^2+(x(2)-2*x(3))^4+10*(x(1)-x(4))^4;
ff=optimset; ff.MaxIter=10000; ff.TolX=1e-7;
tic, x=fminsearch(f,10*ones(4,1),ff); toc; f(x)
```

10.4.3 遗传算法在有约束最优化问题中的应用

前面已经通过例子介绍过,遗传算法可以用于无约束最优化问题的求解。但在实际科学研究中经常要求解有约束的最优化问题,例如第 6 章中分别介绍了利用传统的数学方法求解线性规划、二次型规划及一般非线性规划的方法。经典的遗传算法不能直接用于有约束最优化问题的求解。如果约束中有等式约束,则可以通过等式求解的方式将其中若干个自变量用其他自变量表示。不等式约束则可以通过惩罚函数方法转换到目标函数中,这样可以将

原始问题转换成无约束最优化问题,可以参见文献 [12],或直接将不满足等式约束时的目标函数人为设置成小数,迫使搜索算法脱离这样的 x 值。下面通过例子演示遗传算法在求解有约束线性规划问题中的应用。

例 10-27 试用遗传算法求解线性规划问题。

$$\begin{aligned} & \min (x_1 + 2x_2 + 3x_3) \\ & \mathbf{x} \text{ s.t. } \begin{cases} -2x_1 + x_2 + x_3 \leq 9 \\ -x_1 + x_2 \geq -4 \\ 4x_1 - 2x_2 - 3x_3 = -6 \\ x_{1,2} \leq 0, x_3 \geq 0 \end{cases} \end{aligned}$$

解 由等式约束可以解出 $x_3 = (6 + 4x_1 - 2x_2)/3$ 。可见,原来三元最优化问题可以转换成二元最优化问题。由上面的推导,可以用下面的 MATLAB 函数描述目标函数为

```
function [sol,y]=c10mga4(sol,options)
x=sol(1:2); x=x(:); x(3)=(6+4*x(1)-2*x(2))/3;
y1=[-2 1 1]*x; y2=[-1 1 0]*x;
if (y1>9 | y2<-4 | x(3)<0), y=-100; else, y=-[1 2 3]*x; end
```

其中用 x_1, x_2 数据计算出 x_3 的值,并判定约束条件是否满足,在不满足约束条件时人为地将目标函数的值设置成 -100 ,有意排除这些个体,这样就能由下面语句较好地解决有约束最优化问题。

```
>> [a,b,c]=gaopt([-1000 0; -1000 0], 'c10mga4', [], [], [], 'maxGenTerm', 1000);
x=a(1:2); x(3)=(6+4*x(1)-2*x(2))/3, f=a(3)
```

这里只得出了 $x_1 = -6.9999, x_2 = -10.9999$, 而 $x_3 = (6 + 4x_1 - 2x_2)/3 = 0.0000553$ 。遗传算法的中间结果由表 10-14 给出。

表 10-14 遗传算法搜索部分中间结果

代	x_1	x_2	$f(x)$	代	x_1	x_2	$f(x)$
1	-269.8650965	-377.0311139	-100	57	-1.45193811	0	1.25969055
90	-1.59065611	-0.6689306314	1.953280549	123	-1.860366329	-0.8104298101	3.301831646
186	-5.897126844	-9.4606018	23.48563422	356	-6.388432983	-10.04209934	25.94216492
613	-6.889962856	-10.8068347	28.44981428	676	-6.902971784	-10.80664291	28.51485892
823	-6.925942985	-10.87574207	28.62971492	1080	-6.963837954	-10.92786854	28.81918977
1397	-6.990366303	-10.98077174	28.95183151	1768	-6.997345129	-10.99471444	28.98672565
1849	-6.997368852	-10.99494982	28.98684426	1952	-6.999503469	-10.99926216	28.99751735
1994	-6.999896151	-10.99985241	28.99948076	2000	-6.999916701	-10.99991635	28.9995835

其实,该问题用线性规划函数可以立即得出更精确的结果 $x = [-7, -11, 0]^T$ 。

```
>> f=[1 2 3]; A=[-2 1 1; 1 -1 0]; B=[9; 4]; Aeq=[4 -2 -3]; Beq=-6;
x=linprog(f,A,B,Aeq,Beq,[-inf;-inf;0],[0;0;inf]); x', fm=f*x
```

类似于一般最优化函数, `ga()` 函数还可以直接处理有约束最优化问题,该函数的调用格式为 `[x,a,key]=ga(f,n,A,B,Aeq,Beq,xm,xM,f1)`, 其中参数与非线性规划求解函数完全一致,不过似乎该函数只能得出原问题的可行解,并不能得出问题的最优解。`ga()`

函数同样可以求解由结构体型变量描述的最优化问题。

例 10-28 试用 `ga()` 函数重新考虑例 10-27 中的线性规划问题

解 可以给出下面的求解语句,不过每次语句调用都得出截然不同的结果。经过检验可以发现,每次得出的解都满足约束条件,但每次的目标函数值是不同的,说明每次得到的都是可行解,但不能得出全局最优解,所以使用该函数应该慎重。

```
>> f=@(x)[1 2 3]*x(:); A=[-2 1 1; 1 -1 0]; B=[9; 4]; Aeq=[4 -2 -3]; Beq=-6;
    xm=[-inf;-inf;0]; xM=[0;0;inf]; [x a k]=ga(f,3,A,B,Aeq,Beq,xm,xM)
```

如果由结构体型变量描述最优化问题,则该问题可以由下面语句求解

```
>> P.fitnessfcn=@(x)[1 2 3]*x(:); P.nvars=3; P.Aineq=[-2 1 1; 1 -1 0];
    P.Bineq=[9; 4]; P.Aeq=[4 -2 -3]; P.Beq=-6; P.lb=[-inf;-inf;0];
    P.ub=[0;0;inf]; P.solver='ga'; P.options=gaoptimset; [x a k]=ga(P)
```

从最优化问题求解的方法看,最优化工具箱中的函数一次只能搜索到一个解,对非凸性问题来说往往可能找到一个局部最优值,而用遗传算法则可以同时从一组初值点出发,有可能找到更好的局部最优值甚至全局最优值,但其求取最优值算法的精度和速度均不是很理想。在实际求解问题中,可以考虑采用这样的策略,先用遗传算法初步定出较好最优值所在的大概位置,然后以该位置为初值,调用最优化工具箱中函数快速、准确地求出该最优值。

10.4.4 粒子群优化算法与求解

粒子群优化 (particle swarm optimization, PSO) 算法是文献 [13] 提出的一种进化算法,该算法是受生物界鸟群觅食的启发而提出的搜索食物,即最优解的一种方法。假设某个区域内有一个食物(全局最优值),有位于随机初始位置的若干个鸟(或粒子),每一个粒子有到目前为止自己的个体最优值 $p_{i,b}$,整个粒子群有到目前为止群体的最优值 g_b ,这样每个粒子可以根据下面的式子更新自己的速度和位置

$$v_i(k+1) = \phi(k)v_i(k) + \alpha_1\gamma_{1i}(k)[p_{i,b} - x_i(k)] + \alpha_2\gamma_{2i}(k)[g_b - x_i(k)] \quad (10-4-1)$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (10-4-2)$$

其中 γ_{1i}, γ_{2i} 为 $[0, 1]$ 区间内均匀分布的随机数, $\phi(k)$ 为惯量函数, α_1 和 α_2 为加速常数。

文献 [14] 给出了一个基于粒子群优化算法的 MATLAB 工具箱 [15], 调用格式为

```
[sol,tr] = pso_Trelea_vectorized(fun,n,v_M,[x_m,x_M],key,options)
```

其中 `fun` 为描述目标函数的 MATLAB 函数名,求解函数不支持匿名函数。 n 是 \mathbf{x} 向量的维数,这两个量是求解最优化问题必须提供的,其他的变量是可选的;变量 v_M 是最大允许的速度,其默认值为 4; $\mathbf{x}_m, \mathbf{x}_M$ 是每个变量允许的上界与下界向量,默认为 ± 100 ; `key` 为极值类型选择,默认的 0 为最小值,取 1 表示求最大值;选项 `options` 为结构体控制变量;返回的 $(n+1) \times 1$ 列向量 `sol` 由两个部分构成,前 n 个元素为搜索到的 \mathbf{x} 向量,第 $n+1$ 个元素是目标函数的最优值;返回的 `tr` 记录了寻优的中间结果。主函数基于 Trelea 算法 [16] 实现了粒子群优化功能。该函数调用了神经网络工具箱,并将寻优的中间训练过程用图形显示出来,比较直观。该函数还支持 Clerc 算法 [17]。

这里给出的函数是“向量化”版本,允许一次性运行一组粒子向量的目标函数求值,故其效率远远高于非向量化的版本。在目标函数的描述上与传统的最优化目标函数有不同之处。前面介绍的目标函数中 \mathbf{x} 为向量,每个元素对应 x_i 的当前值,而这里的 \mathbf{x} 为矩阵,其第 i 列为各个粒子的 x_i 值,所以在原来目标函数程序中用的 $x(i)$ 变量应该修改为 $x(:,i)$,且相应地应该进行点运算。下面将用实际例子演示最优化求解方法。

例 10-29 试用粒子群算法求解例 10-25 中给出的无约束最优化问题。

解 编写一个 MATLAB 函数描述目标函数

```
function f=c10mpso1(x)
f=(x(:,1)+x(:,2)).^2 + 5*(x(:,3)-x(:,4)).^2 +...
(x(:,2)-2*x(:,3)).^4 + 10*(x(:,1)-x(:,4)).^4;
```

注意,在目标函数中的向量化表示和点运算,如果去掉这些表示,程序将无法运行。可以按照遗传算法工具箱编写如下函数,描述最优化问题的目标函数。这样可以由下面命令求解该最优化问题

```
>> x=pso_Trelea_vectorized('c10mpso1',4); x(1:4)
```

由前面的命令得出的最优解近似值为 $\mathbf{x} = [-0.00059, 0.00059, 0.00227, 0.00227]^T$, 解的图示由图 10-37 给出。

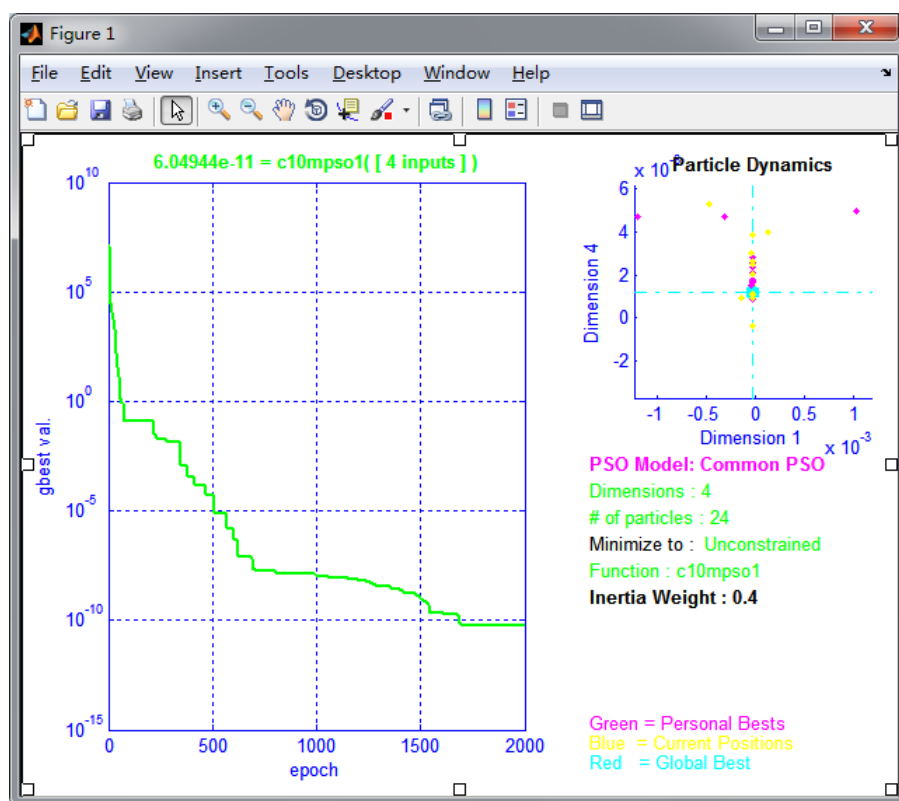


图 10-37 粒子群优化算法寻优结果的图示

例 10-30 重新考虑例 10-27 中给出的有约束最优化问题,试用粒子群算法求解该问题。

解 类似于前面基于遗传算法程序的编写方法,注意这里采用的是向量化的模式描述目标函数,

可以编写出如下的函数

```
function y=c10mpso4(x)
x1=x(:,1); x2=x(:,2); x3=(6+4*x1-2*x2)/3; x=[x x3]';
y1=[-2 1 1]*x; y2=[-1 1 0]*x; y=[1 2 3]*x;
ii=find(y1>9|y2<-4|x3'<0); y(ii)=100; y=y(:);
```

由下面的语句求解该问题,得出精确的最优解 $\mathbf{x}^T = [-7, -11, 0]$ 。该函数调用后将得出类似于图 10-37 所示的搜索结果。

```
>> x=psotrelea_vectorized('c10mpso4',2); [x(1:2); (6+4*x(1)-2*x(2))/3]
```

10.4.5 其他全局优化算法

MATLAB 的全局优化工具箱还提供了其他的全局优化函数,如模拟退火算法函数 `simulannealbnd()`、模式搜索函数 `patternsearch()` 等,其中 `simulannealbnd()` 函数可以求解边界约束的无约束最优化问题,其格式为 `$\mathbf{x} = \text{simulannealbnd}(f, \mathbf{x}_0, \mathbf{x}_m, \mathbf{x}_M)$` ,模式搜索函数可以求解一般的非线性规划问题,与 `fmincon()` 函数格式完全一致。

例 10-31 重新考虑例 10-27 中给出的线性规划问题。

解 由前面给出的遗传算法函数 `ga()` 并不能很好地求解有约束最优化问题,所以可以考虑由模式搜索函数直接求解原问题,得出 $\mathbf{x} = [-6.99906, -10.9981, 1.5546 \times 10^{-7}]$,得出的目标函数为 -28.9953 ,可见模式搜索寻优函数可以很好地求解有约束最优化问题。

```
>> f=@(x)[1 2 3]*x(:); A=[-2 1 1; 1 -1 0]; B=[9; 4];
Aeq=[4 -2 -3]; Beq=-6; xm=[-inf;-inf;0]; xM=[0;0;inf];
[x a k]=patternsearch(f,rand(3,1),A,B,Aeq,Beq,xm,xM)
```

例 10-32 试求出下面最优化问题的全局最优解。

$$\begin{aligned} \min \quad & \sin(3xy) + (x-0.1)(y-1) + x^2 + y^2 \\ \text{s.t.} \quad & \begin{cases} -1 \leq x \leq 3 \\ -3 \leq y \leq 3 \end{cases} \end{aligned}$$

解 该目标函数的曲面表示在图 10-38 中给出,可见,得出的曲面凹凸不平,所以求取全局最优解比较困难,因为全局最优解是否能得到全凭是否恰当地选择初始搜索点。

```
>> [x,y]=meshgrid(-1:0.1:3,-3:0.1:3);
z=sin(3*x.*y+2)+(x-0.1).*(y-1)+x.^2+y.^2; surf(x,y,z);
```

求解原始问题需要引入变量 $x_1 = x, x_2 = y$,原问题可以改写为

$$\begin{aligned} \min \quad & \sin(3x_1x_2) + (x_1-0.1)(x_2-1) + x_1^2 + x_2^2 \\ \text{s.t.} \quad & \begin{cases} -1 \leq x_1 \leq 3 \\ -3 \leq x_2 \leq 3 \end{cases} \end{aligned}$$

若采用传统非线性规划问题求解方法,则可以写出匿名函数来表示目标函数,这样从一个随意选取的初值可能得出解为 $\mathbf{x}^T = [0.9299, 0.6577]$,目标函数为 0.3742。

```
>> f=@(x)sin(3*x(1)*x(2)+2)+(x(1)-0.1)*(x(2)-1)+x(1)^2+x(2)^2;
```

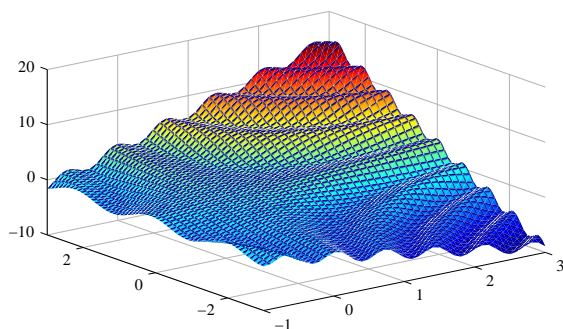


图 10-38 目标函数的三维表面图

```
x0=rand(1,2); x=fmincon(f,x0,[],[],[],[],[-1;-3],[3;3]), f(x)
```

若采用模拟退火函数 `simulannealbnd()` 函数,可以得出的最优解为 $x = 1.22698$, $y = -0.91747$,且目标函数为 -0.79502867 。可见,由传统最优化方法得出的解是局部最优解,其目标函数比这里得出的目标函数大得多。

```
>> x=simulannealbnd(f,rand(2,1),[-1;-3],[3;3]), f(x)
```

10.4.6 求取精确的全局最优解

从前面的演示例子基本可以得出这样的结论:传统的最优化方法可能得出精确的最优解,然而,经常可能得到的是局部最优解而不是全局最优解,而进化方法通常能得出全局最优解,但解的精度可能很低。在实际最优化问题求解中可以考虑将二者的优势结合起来,得到精确的全局最优解。例如,可以采用进化方法首先获得全局最优解的不精确近似值,以该值为初始点,采用传统最优化求解方法搜索出精确的全局最优值。

例 10-33 试求出例 10-32 中的最优化问题的精确全局最优解。

解 例 10-32 中曾经给出问题的全局最优解,只不过该解不一定很精确。由该结果为初值,就可以由下面语句直接得出原问题的精确全局最优解为 $x = 1.2255333$, $y = -0.918287$,这时的目标函数值为 -0.79503377 。

```
>> f=@(x)sin(3*x(1)*x(2)+2)+(x(1)-0.1)*(x(2)-1)+x(1)^2+x(2)^2;
x0=simulannealbnd(f,rand(3,1),[-1;-3],[3;3])
ff=optimset; ff.TolX=1e-20; ff.TolFun=1e-20;
x=fmincon(f,x0,[],[],[],[],[-1;-3],[3;3],[],ff), f(x)
```

10.4.7 基于遗传算法的混合整数规划求解

遗传算法求解函数 `ga()` 还可以用于混合整数规划问题的求解,其完整的语句调用格式为 `[x,a,key]=ga(f,n,A,B,[],[],x_m,x_M,f1,options,v_i)`,其中,由 `ga()` 函数求解混合整数规划问题,目前只能求解不含有等式约束的问题,非线性约束 f_1 函数也不能含有等式约束。

也可以用结构体描述原始问题再调用 `ga()` 函数求解,这时需要将目标函数赋给成员变量 `fitnessfcn`,将整数的决策变量编号赋给 `IntCon`,其他成员变量将通过例子演示。

例 10-34 试用遗传算法重新求解例 6-35 中的非线性整数规划问题。

解 该例子通过引入适当的变换已经改写成标准的整数规划问题,下面重新给出该问题

$$\begin{aligned} \min \quad & 2y_1^2/16 + y_2^2/100 - 4y_1 - y_2 \\ \text{s.t.} \quad & \begin{cases} y_1^2/16 - 6y_1/4 + y_2/10 - 11 \leq 0 \\ -y_1y_2/40 + 3y_2/10 + e^{y_1/4-3} - 1 \leq 0 \\ y_2 \geq 30 \end{cases} \end{aligned}$$

非线性约束已经由 `c6mdisp.m` 给出。下面的语句可以用来直接求解整数规划问题,得出的结果是 $y = [16, 50]$,与原例子中得出的完全一致。

```
>> clear P; P.fitnessfcn=@(y)2*y(1)^2/16+y(2)^2/100-4*y(1)-y(2);
P.nonlcon=@c6mdisp; P.IntCon=[1,2]; P.lb=[-200; 30]; P.ub=[200; 200];
P.solver='ga'; P.options=gaoptimset; P.nvars=2; [y,a key c]=ga(P)
```

10.5 小波变换及其在数据处理中的应用

Fourier 变换是信号处理中一种重要的手段,但因其本身的局限性(例如,它只能将时域波形变换成频域表示,完全失去了与原来时域信号的对应关系),所以对某些特定的信号进行处理不是很理想。前面已经介绍过,Fourier 变换将给定的信号展开成不同频率的正弦信号之和。对平稳的信号来说,用 Fourier 变换的方式可以对信号进行较好的分析,但对非平稳的信号和暂变的信号,不适合使用 Fourier 变换进行分析。因为 Fourier 变换会略去重要的暂态信息,因此需要引入其他的变换方式解决这样的问题,如短时 Fourier 变换。20 世纪 80 年代逐渐兴起的小波分析技术弥补了这方面的不足,目前正越来越广泛地应用于数据与信号处理以及图像处理等领域。

10.5.1 小波变换及基小波波形

所谓小波(wavelet),是指均值为 0 的一类波形。小波分析是将原来的信号分解为基小波波形经过平移与比例变化后的一系列波形。本节将介绍连续、离散小波变换的基本内容,并介绍几种常用的基小波函数波形。

1. 连续小波变换

连续小波变换的变换公式为

$$\mathcal{W}_{a,b}^{\psi}[f(t)] = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} f(t) \overline{\psi_{a,b}(t)} dt = W_{\psi}(a,b) \quad (10-5-1)$$

其中

$$\psi_{a,b}(t) = \psi\left(\frac{t-b}{a}\right), \text{ 且 } \int_{-\infty}^{\infty} \psi(t) dt = 0 \quad (10-5-2)$$

$\psi(t)$ 称为基小波, $\psi_{a,b}(t)$ 为基小波通过平移、比例缩放构成的小波信号。

例 10-35 假设“墨西哥帽”基小波函数由 $\psi(t) = \frac{1-t^2}{\sqrt{2\pi}}e^{-t^2/2}$ 给出,试绘制出不同 a, b 值变换下的小波函数。

解 因为基小波函数已给出,故可以用符号运算工具箱表示该函数,并用 `ezplot()` 函数将其绘制出来。利用符号运算工具箱中给出的 `subs()` 函数则可以将 t 变量替换成小波函数 $\psi_{a,b}(t)$ 所需的形式,并在原来坐标系下绘制出不同 a, b 参数下的小波函数曲线,分别如图 10-39 (a)、(b) 所示。

```
>> syms t; f=(1-t^2)*exp(-t^2/2)/sqrt(2*pi);
    ezplot(f,-4,4), hold on; % 绘制基小波,并保护坐标系不被刷新
    ezplot(subs(f,t,t-1),-4,4); ezplot(subs(f,t,t+1),-4,4) % 小波平移
    figure; ezplot(f,-4,4), hold on;
    ezplot(subs(f,t,t/2),-4,4); ezplot(subs(f,t,2*t),-4,4) % 小波缩放
```

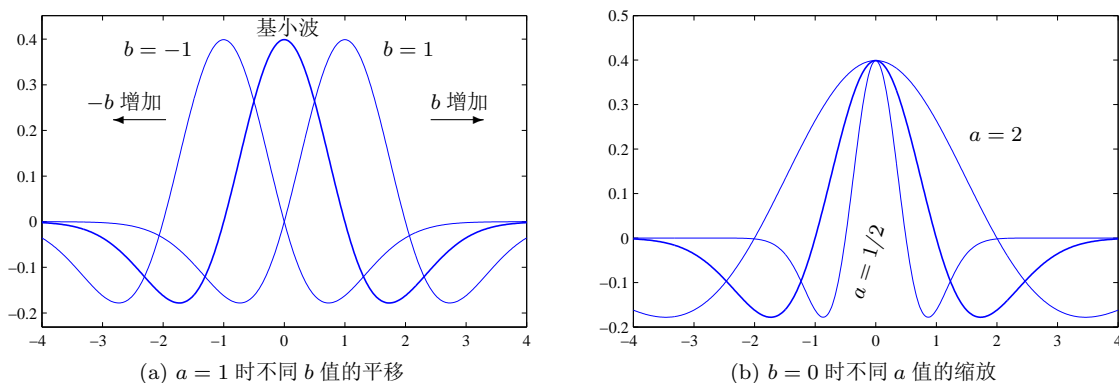


图 10-39 墨西哥帽基小波在不同 a, b 下的波形

从上面的例子可以看出, b 参数将向左右方向平移基小波信号, a 参数起到扩展或压缩基小波的作用。若 $a < 1$, 将压缩基小波信号的宽度, 形成新的小波信号。小波分析是对各个 a, b 组合计算出系数, 然后将这些小波信号乘以相应的系数再叠加起来, 重构出原信号。和其他积分变换一样, 重构原信号又称为小波反变换, 或小波反演。

小波反变换的定义为

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_\psi(a, b) \psi_{a,b}(t) da db \quad (10-5-3)$$

其中

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega \quad (10-5-4)$$

用连续小波变换的系数计算可以由 `cwt()` 函数完成。该函数的调用格式为

```
Z = cwt(y, a, 基小波名称) % 计算小波系数矩阵 Z
Z = cwt(y, a, 基小波名称, 'plot') % 直接绘制小波系数绝对值图
```

其中, 基小波名称在后面将详细介绍, 这里只使用墨西哥帽函数, 其名称标记为 'mexh'。

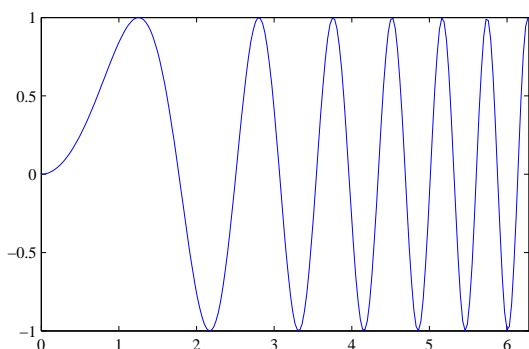
例 10-36 试对信号 $f(t) = \sin t^2$ 进行连续小波分解,并绘制出其系数图。

解 可以由下面语句生成 $t \in [0, 2\pi]$ 区间内的数据并绘制出时域数据曲线,如图 10-40 (a) 所示。

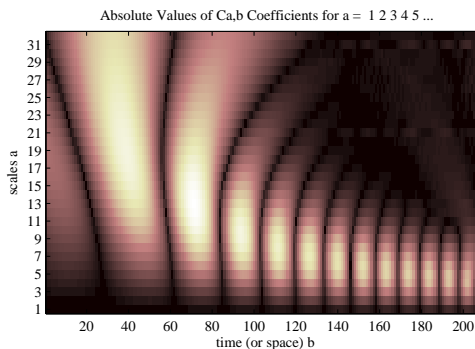
```
>> t=0:0.03:2*pi; y=sin(t.^2); plot(t,y)
```

选择 'mexh' 基小波作为模板,可以绘制出小波系数 $W_\psi(a, b)$ 的图形,如图 10-40 (b) 所示。

```
>> a=1:32; Z=cwt(y,a,'mexh','plot'); % 绘制绝对值图
```



(a) 已知时域函数曲线



(b) 连续小波变换系数

图 10-40 连续小波变换

还可以用下面的命令绘制小波系数的三维表面图,如图 10-41 所示。

```
>> surf(t,a,Z); shading flat; axis([0 2*pi,0,32,min(Z(:)) max(Z(:))])
```

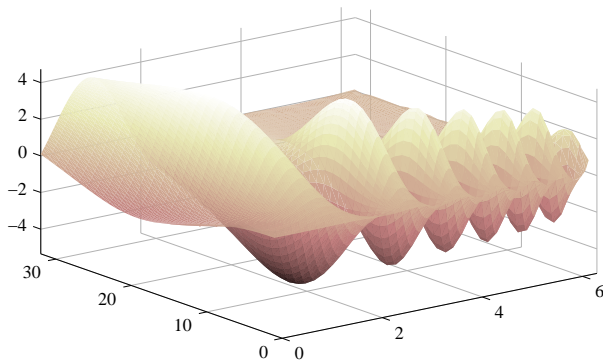


图 10-41 连续小波变换系数的三维表面图表示

2. 离散小波变换

若 $f(t)$ 信号取其离散值 $f(k)$,且选择基小波函数 $\psi(t)$,则结果平移与缩放的小波函数为 $\psi_{a,b}(t) = \sqrt{2}\psi(2^m t - n)$,其离散形式可以写成 $\psi_{a,b}(k) = \sqrt{2}\psi(2^m k - n)$,这时可以定义出离散信号的小波变换为

$$\mathcal{W}_{n,m}^\psi[f(k)] = \sqrt{2} \sum_k f(k) \overline{\psi(2^m k - n)} = W_\psi(m, n) \quad (10-5-5)$$

离散小波反演公式为

$$f(k) = \sum_m \sum_n W_{n,m}(k) \psi_{m,n}(k) \quad (10-5-6)$$

MATLAB 的小波工具箱中给出了 `dwt()` 函数,可以对给定的数据进行一次离散小波变换。该函数的调用格式为 `[cA,cD] = dwt(x,fun)`,其中, x 为原始数据, fun 为选择的基小波函数名,可以为前面介绍的 'mexh' 函数,还可以是后面将介绍的其他基小波波形。结果离散小波变换得出的 cA 是能近似描述原波形的小波系数,而 cD 为信号的细节信息,通常前者对应于低频,后者对应于高频的部分。 cA 和 cD 的长度均为原向量 x 长度的一半。

还可以调用 `$\hat{x} = idwt(cA,cD,fun)$` 函数进行离散小波反变换,还原出 \hat{x} 向量。

例 10-37 生成一组被噪声污染的信号数据,试对其进行离散小波分解,并对结果进行离散小波反变换,反演出原函数,再观察反演结果。

解 仿照例 10-36 中给出的信号模型 $f(t) = \sin t^2$,在其基础上叠加标准差为 0.1 的白噪声信号,则可以用下面的语句生成波形曲线,如图 10-42 所示。可见,该曲线被噪声污染较严重。通过离散小波变换,可以在同一图形窗口内绘制出近似波形和细节波形。可以看出,这样得出的波形在一定程度上降低了噪声,如果需要较好地降噪则需要多次进行小波变换。

```
>> x=0:0.002:2*pi; y=sin(x.^2); r=0.1*randn(size(x));
    y1=y+r; subplot(211); plot(x,y1), [cA,cD]=dwt(y1,'db4'); % 离散小波变换
    subplot(223), plot(x(1:length(cA)),cA);
    subplot(224), plot(x(1:length(cD)),cD) % 绘制近似和细节信号
```

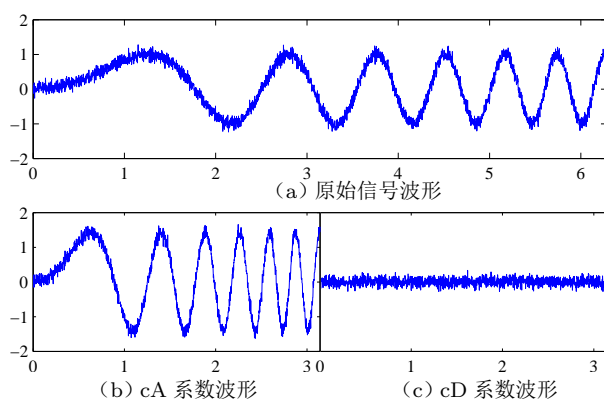


图 10-42 给定信号的小波分解

对得出的 cA 和 cD 向量还可以进行离散小波反变换。经过和原信号比较,得出的信号基本上还原了原始信号,误差达 10^{-11} 数量级。

```
>> y2=idwt(cA,cD,'db4'); norm(y1-y2) % 检验反变换对信号的还原精度
```

3. 小波工具箱中提供的基小波函数

小波工具箱中提供了大量的基小波模板,如 Haar 小波、Daubechies 族小波、墨西哥帽小波、Bior 族小波等,可以直接调用。用 `wavemngr()` 函数即可列出允许使用的基小波名称。该函数可以由下面的格式调用 `wavemngr('read',1)`。例如,Haar 小波可以选择名称 'haar', Daubechies 族小波可以有 'db1'、'db2' 等模板, Bior 族小波可以有 'bior1.3'、'bior2.4' 等,墨西哥帽小波可以选择 'mexh' 等。小波模板数据可以由 `wavefun()` 来计

算。该函数的调用格式为

```
[ $\psi$ , $x$ ] = wavefun(fun, $n$ ),    % Gauss、墨西哥帽等基小波函数
[ $\phi$ , $\psi$ , $x$ ] = wavefun(fun, $n$ ), % Daubechies 族、Symlets 族等正交基小波函数
[ $\phi_1$ , $\psi_1$ , $\phi_2$ , $\psi_2$ , $x$ ] = wavefun(fun, $n$ ), % Bior 族等基小波函数
```

其中, n 为迭代次数,其默认值为 8。 ψ 为基小波, ϕ 为小波导数,而 Bior 小波中 ϕ 、 ψ 向量的下标 1 表示用于小波分解,2 用于小波重建。

例 10-38 选择 Daubechies 6 小波('db6'),试绘制出不同阶次下的基小波波形。

解 用下面的语句可以立即绘制出该小波在 2、4、6、8 迭代次数下的波形,如图 10-43 所示。可见,当迭代次数选为 8 时能得出相当平滑的波形,所以一般可以选择的迭代次数为 8。

```
>> [a,y,x]=wavefun('db6',2); subplot(141), plot(x,y)
    [a,y,x]=wavefun('db6',4); subplot(142), plot(x,y)
    [a,y,x]=wavefun('db6',6); subplot(143), plot(x,y)
    [a,y,x]=wavefun('db6',8); subplot(144), plot(x,y)
```

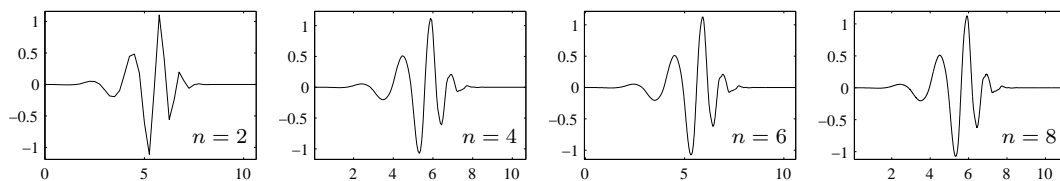


图 10-43 不同迭代次数下的 Daubechies 6 基小波函数波形

例 10-39 试绘制出常用基小波波形。

解 下面的语句可以绘制出一些常用的基小波波形,如图 10-44 所示。其中,'db1' 小波实际上就是 Haar 小波。还可以看出, Daubechies 族小波在 'db6' 取值时较平滑, Symlets 族小波在 'sym6' 时较平滑。

```
>> subplot(5,4,1), [a,y,x]=wavefun('db1'); plot(x,y), % 同样绘制其他 db 小波
    subplot(5,4,9), [a,y,x]=wavefun('sym2'); plot(x,y), % 同样绘制其他 sym 小波
    subplot(5,4,13), [a,y,x]=wavefun('coif2'); plot(x,y)
    subplot(5,4,15), [y,x]=wavefun('gaus2'); plot(x,y)
    subplot(5,4,17), [a,y,b,c,x]=wavefun('bior1.3'); plot(x,y) % 以下略
```

10.5.2 小波变换技术在信号处理中的应用

与 Fourier 变换技术以及基于该技术的频域方法类似,小波技术也可以用于信号处理和二维信号处理(如图像处理),且可以显示出传统频域分析方法难以实现的特性。本节将介绍基于小波变换技术的信号分解与重建方法及 MATLAB 实现,并将通过信号噪声过滤的例子来演示小波在降噪中的应用。

通过小波变换方法对某给定信号进行分解,可以将给定信号 S 分解成两个部分,即 cA_1 和 cD_1 ,这时得出的 cA_1 和 cD_1 信号的数据量均为原数据 S 的一半,且 cA_1 保留原信号的

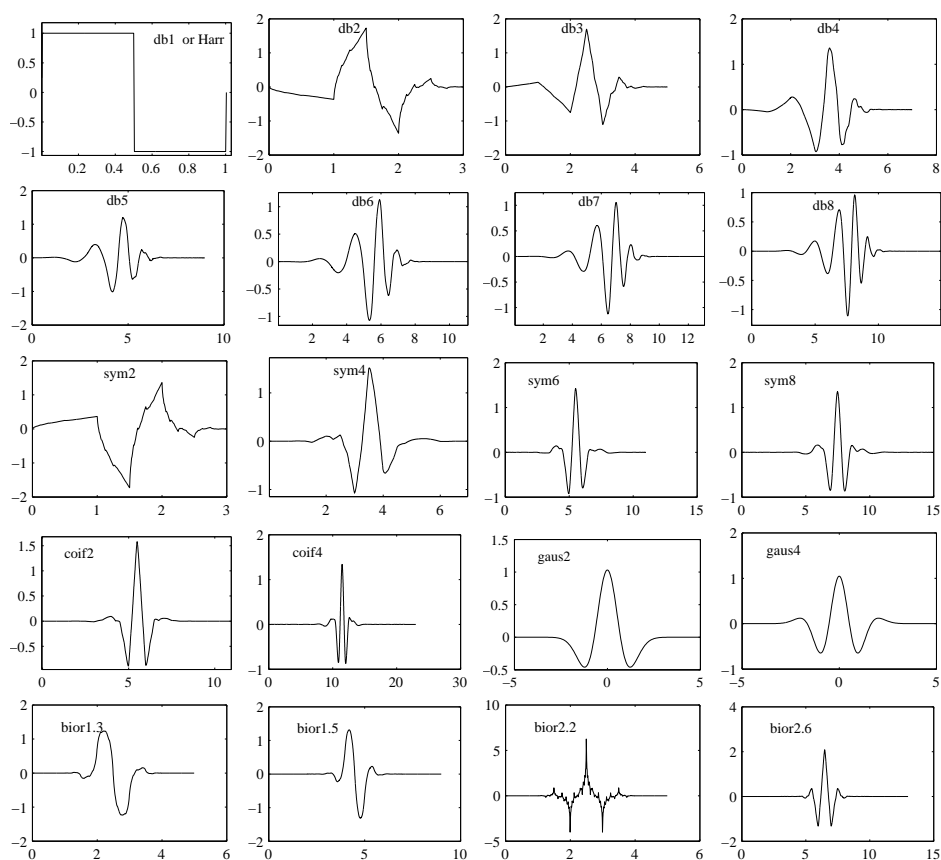


图 10-44 常用基小波波形

低频信息或近似信息,而 cD_1 保留该信号的高频信息或细节信息。从信号的噪声过滤的角度看, cA_1 信号有效的成分多,而 cD_1 多属于噪声信号。对 cA_1 信号再进行一步小波分解,则得出 cA_2 、 cD_2 。对 cA_2 再进行分解则得出 cA_3 和 cD_3 ,如此还可以再进行多步分解,分解的过程如图 10-45 (a) 所示。和前面介绍的单级小波分解类似,各个 cA 序列称为近似系数,而 cD 段称为细节系数。

MATLAB 的小波分析工具箱提供了 `wavedec()` 函数,可以用于一维信号的小波分解。该函数的调用格式为 `[C,L] = wavedec(x,n,fun)`,其中, x 为原始信号, n 为分解的步数,如取 $n = 3$, fun 为所选基小波的名称,如 'db6',分解后可以得出 C 和 L 两个向量,其组成形式如图 10-45 (b) 所示,即 C 向量是按照如图 10-45 (b) 所示的顺序将这些段短向量接成的和 x 等长度的向量,且每个子段的长度由 L 向量相应元素给出。

由分解后的 C 和 L 向量提取近似系数 cA 和细节系数 cD 可以分别由 `appcoef()` 和 `detcoef()` 函数实现。其调用格式分别为

```
cAn = appcoef(C,L,fun,n); % 提取近似系数
cDi = detcoef(C,L,i); % 提取第 i 段细节系数
```

由得出的近似系数和细节系数重建原信号则可以略去部分噪声信息,信号重建的函数可以

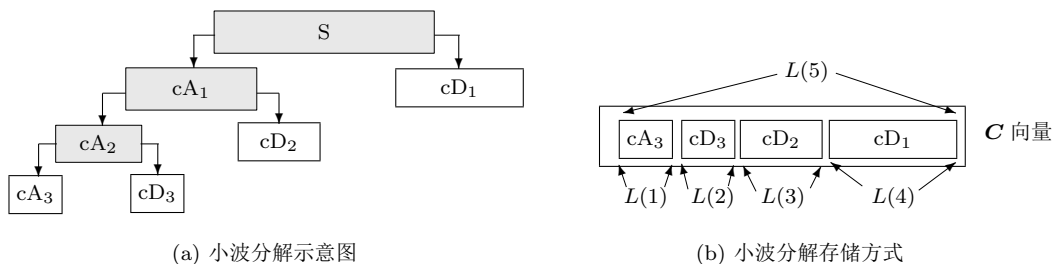


图 10-45 小波分解示意图

使用 `wrcoef()`。该函数的调用格式为 $\hat{x} = \text{wrcoef}(\text{类型}, C, L, \text{fun}, n)$ ，其中，“类型”可以选择为 'a' 和 'd'，用来确定是利用近似小波系数还是利用细节系数来进行原信号重建，若选择了近似系数，则可以较好地解决信号降噪问题。

例 10-40 对例 10-37 中的数据进行 3 次小波分解，试用各种基小波函数对其进行降噪处理，并比较这些基小波函数对降噪效果的影响。

解 由例 10-37 中给出的数据信号可以绘制出如图 10-42 (a) 所示的原信号波形曲线。

```
>> x=0:0.002:2*pi; y=sin(x.^2); r=0.1*randn(size(x)); y1=y+r; plot(x,y1)
```

对给定的数据进行 3 次小波分解，则可以用下面的语句绘制出相关各个子信号的波形，如图 10-46 所示，其中作者根据需要对各个坐标系的宽度进行了手工调整。可见，每次分解都能滤去一部分噪声，故最终得出的 cA_3 含有的噪声成分较少。

```
>> [C,L]=wavedec(y1,3,'db6'); cA3=C(1:L(1)); subplot(141), plot(cA3)
dA3=C(L(1)+1:sum(L([1 2]))); subplot(142), plot(dA3)
dA2=C(sum(L(1:2))+1:sum(L(1:3))); subplot(143), plot(dA2)
dA1=C(sum(L(1:3))+1:sum(L(1:4))); subplot(144), plot(dA1)
```

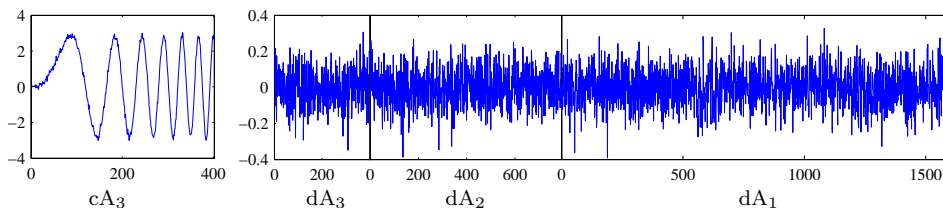


图 10-46 小波分解的结果

现在仍采用 'db6' 基小波，则可以由下面的语句绘制出滤波后的近似波形，如图 10-47 (a) 所示。若采用 'db2' 基小波，则可以得出如图 10-47 (b) 所示的滤波近似波形。从得出的结果看，对本例来说，采用两种基小波在滤波效果上没有显著的差异。

```
>> A3=wrcoef('a',C,L,'db6',3); plot(A3); figure
[C,L]=wavedec(y1,3,'db2'); A3=wrcoef('a',C,L,'db2',3); plot(A3)
```

其实，用下面的语句还可以得出另外两种常用基小波下降噪的效果，和如图 10-47 (a)、(b) 所示的结果很接近。故对本例来说，降噪效果仍无显著差异。

```
>> [C,L]=wavedec(y1,3,'bior2.6'); A3=wrcoef('a',C,L,'bior2.6',3); plot(A3)
```

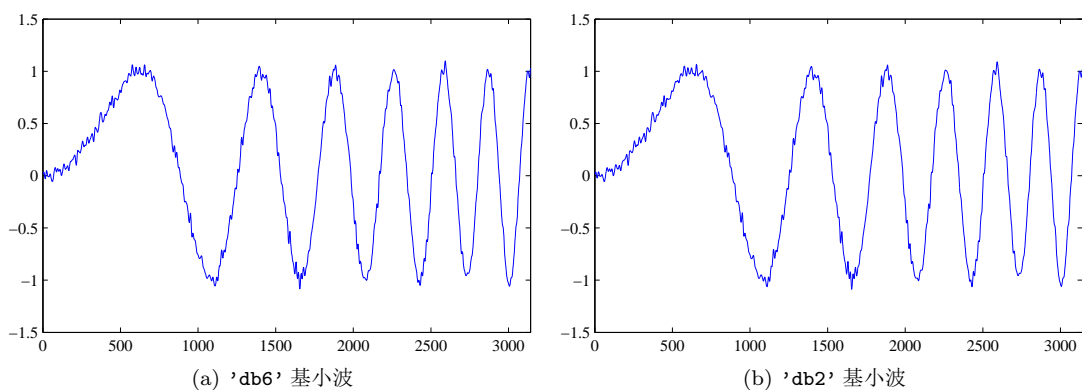


图 10-47 不同基小波下小波分析降噪效果

```
[C,L]=wavedec(y1,3,'coif4'); A3=wrcoef('a',C,L,'coif4',3); plot(A3)
```

例 10-41 重新考虑例 8-43 中的数字滤波问题,试用小波对其进行滤波,并比较滤波效果。

解 用下面语句可以重复例 8-43 中给出的滤波器滤波结果,同样,采用 4 级 'db6' 小波,也可以进行降噪,这些降噪效果在图 10-48 中给出,图中有延迟的曲线为第 8 章的滤波方法得出的。可见,小波降噪方法不会产生数字滤波器那样的时间延迟,滤波效果也明显好于滤波器。

```
>> b=1.2296e-6*conv([1 4 6 4 1],[1 3 3 1]); a=conv([1,-0.7265],...
    conv([1,-1.488,0.5644],conv([1,-1.595,0.6769],[1,-1.78,0.8713])));
x=0:0.002:2; y=exp(-x).*sin(5*x); r=0.05*randn(size(x)); y1=y+r;
y2=filter(b,a,y1); % 例 8-43 中给出的滤波方法
[C,L]=wavedec(y1,4,'db6'); A4=wrcoef('a',C,L,'db6',4);
plot(x,y,x,y2,x,A4) % 原来污染信号与污染信号的两种滤波效果
```

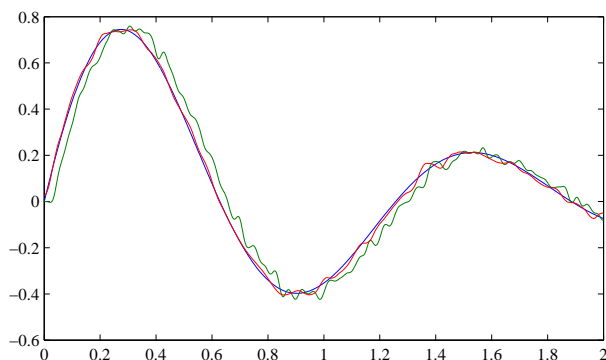


图 10-48 给定信号的滤波效果比较

10.5.3 小波问题的程序界面

小波工具箱还提供了求解一维、二维小波变换问题的图形用户界面。在 MATLAB 命令窗口中输入 `wavemenu` 命令可以启动该程序,得出如图 10-49 所示的图形界面。如果想解决一维小波变换问题,则单击 Wavelet 1-D (一维小波分析) 按钮,该程序将引导用户输入数据

文件,选择小波并进行小波分析的全过程。该界面使用起来较容易,故不在此处详细介绍了。

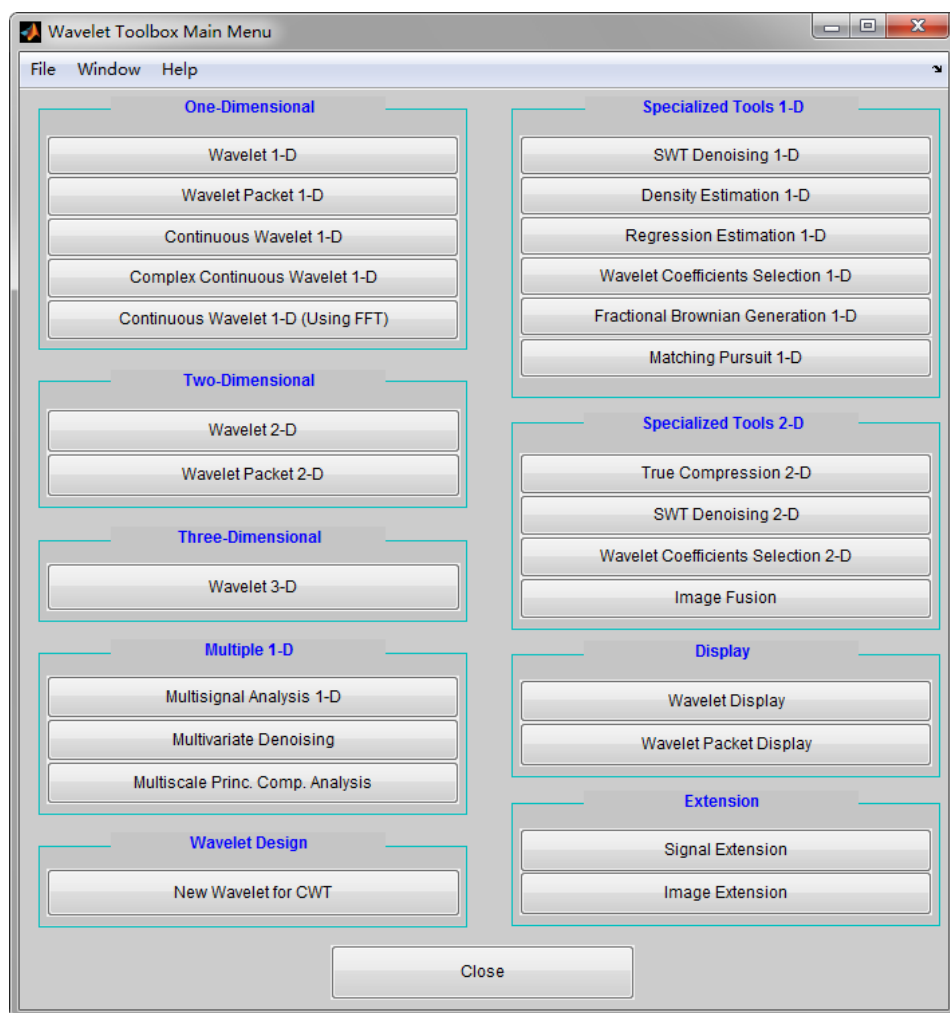


图 10-49 小波分析程序界面

10.6 分数阶微积分学问题求解及应用

第 3 章详细介绍了微积分学的相关内容,其他一些章节还陆续介绍了基于 MATLAB 语言的微积分问题的数值解和解析解法。一般地, $\mathrm{d}^n y/\mathrm{d}x^n$ 表示 y 对 x 的 n 阶导数,但若 $n = 1/2$ 时会怎么样呢? 这是 300 多年以前法国著名数学家 Guillaume François Antoine L'Hôpital 问过微积分学创造者之一 Gottfried Wilhelm Leibniz 的一个问题^[18],这标志着分数阶微积分研究的开始。严格说来,“分数阶”(fractional-order)一词是误用,正确的应该非整数阶(non-integer-order),因为 $\sqrt{2}$ 也可以用作微积分的阶次,而它不是分数。然

而,“分数阶”一词流传很广,在这一领域的研究者中也一直使用分数阶一词,所以本书也沿用该关键词。分数阶微积分理论建立至今已经有 300 年的历史了,但早期主要侧重于理论研究,近年来在很多领域都已经开始应用分数阶微积分学理论,例如在自动控制领域出现了分数阶控制理论等新的分支。本节将介绍分数阶微积分的定义及各种计算方法,并介绍分数阶线性及非线性微分方程的求解方法,还将以分数阶传递函数为例,介绍在 MATLAB 语言下面向对象的编程方法。

10.6.1 分数阶微积分的定义

在分数阶微积分理论发展过程中,出现了很多种函数的分数阶微积分的定义,如由整数阶微积分直接扩展而来的 Cauchy 积分公式、Grünwald–Letnikov 分数阶微积分定义、Riemann–Liouville 分数阶微积分定义以及 Caputo 定义等,本节将先介绍这些定义及其等效关系,再给出分数阶微积分的各种性质。

分数阶微积分有各种各样的定义,其中常用的定义为:

(1) **分数阶 Cauchy 积分公式**。该公式从简单整数阶积分直接扩展而来

$$\mathcal{D}^\gamma f(t) = \frac{\Gamma(\gamma+1)}{2\pi j} \int_C \frac{f(\tau)}{(\tau-t)^{\gamma+1}} d\tau \quad (10-6-1)$$

其中, C 为包围 $f(t)$ 单值与解析开区域的光滑曲线。

(2) **Grünwald–Letnikov 分数阶微积分定义**。该定义为

$${}_a^{\text{GL}} \mathcal{D}_t^\alpha f(t) = \lim_{h \rightarrow 0} \frac{1}{h^\alpha} \sum_{j=0}^{[(t-a)/h]} (-1)^j \binom{\alpha}{j} f(t-jh) \quad (10-6-2)$$

其中, $\binom{\alpha}{j}$ 为二项式系数,其计算方法后面将介绍。

(3) **Riemann–Liouville 分数阶微积分公式**。其分数阶积分的定义为

$${}_a^{\text{RL}} \mathcal{D}_t^{-\alpha} f(t) = \frac{1}{\Gamma(\alpha)} \int_a^t (t-\tau)^{\alpha-1} f(\tau) d\tau \quad (10-6-3)$$

其中, $0 < \alpha < 1$, 且 a 为初值,一般可以假设零初始条件,即令 $a = 0$,这时微分记号可以简写成 ${}^{\text{RL}} \mathcal{D}_t^{-\alpha} f(t)$,或省去角标 RL。Riemann–Liouville 定义是目前最常用的分数阶微积分定义。特别地, \mathcal{D} 左右侧的下标分别表示积分式的下界和上界^[19]。

由这样的积分还可以定义出分数阶微分。假设分数阶 $n-1 < \beta \leq n$,则分数阶微分为

$${}_a^{\text{RL}} \mathcal{D}_t^\beta f(t) = \frac{d^n}{dt^n} \left[{}_a \mathcal{D}_t^{-(n-\beta)} f(t) \right] = \frac{1}{\Gamma(n-\beta)} \frac{d^n}{dt^n} \left[\int_a^t \frac{f(\tau)}{(t-\tau)^{\beta-n+1}} d\tau \right] \quad (10-6-4)$$

(4) **Caputo 分数阶微分定义**。Caputo 分数阶微分定义为

$${}_0^{\text{C}} \mathcal{D}_t^\alpha y(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{y^{(m+1)}(\tau)}{(t-\tau)^\alpha} d\tau \quad (10-6-5)$$

其中, $\alpha = m + \gamma$, m 为整数, $0 < \gamma \leq 1$ 。类似地, Caputo 分数阶积分定义为

$${}_0^{\text{C}} \mathcal{D}_t^\gamma y(t) = \frac{1}{\Gamma(-\gamma)} \int_0^t \frac{y(\tau)}{(t-\tau)^{1+\gamma}} d\tau, \quad \gamma < 0 \quad (10-6-6)$$

10.6.2 不同分数阶微积分定义的关系与性质

可以证明^[20], 对很广一类实际函数来说, Grünwald–Letnikov 分数阶微积分定义及 Riemann–Liouville 分数阶微积分定义是完全等效的。Caputo 定义和 Riemann–Liouville 定义的区别主要表现在对常数求导的定义上, 前者对常数的求导是有界的(为 0), 而后者求导是无界的, Caputo 定义更适用于分数阶微分方程初值问题的描述。

若函数 $y(t)$ 的初值非零, 且 $\alpha \in (0, 1)$, 则比较 Caputo 和 Riemann–Liouville 定义可见

$${}_C^{\alpha} \mathcal{D}_t^{\alpha} y(t) = {}^{\text{RL}} \mathcal{D}_t^{\alpha} (y(t) - y(t_0)) \quad (10-6-7)$$

其中, 常数 $y(t_0)$ 的导数为 ${}^{\text{RL}} \mathcal{D}_t^{\alpha} y(t_0) = y(t_0)(t - t_0)^{-\alpha}/\Gamma(1 - \alpha)$, 这时可以推导出 Caputo 微分定义和 Riemann–Liouville 定义之间的关系为

$${}_C^{\alpha} \mathcal{D}_t^{\alpha} y(t) = {}^{\text{RL}} \mathcal{D}_t^{\alpha} y(t) - \frac{y(t_0)(t - t_0)^{-\alpha}}{\Gamma(1 - \alpha)} \quad (10-6-8)$$

更一般地, 如果阶次 $\alpha > 1$, 记 $m = [\alpha]$, 则

$${}_C^{\alpha} \mathcal{D}_t^{\alpha} y(t) = {}^{\text{RL}} \mathcal{D}_t^{\alpha} y(t) - \sum_{k=0}^{m-1} \frac{y^{(k)}(t_0)}{\Gamma(k - \alpha + 1)} (t - t_0)^{k-\alpha} \quad (10-6-9)$$

且前面介绍的 $0 \leq \alpha \leq 1$ 是上述公式的一个特例。

若 $\alpha < 0$, 前面已经指出, Riemann–Liouville 分数阶积分定义和 Caputo 积分定义是完全相同的, 所以在实际应用中二者可以混用。

这里不加证明地给出分数阶微积分的性质^[21]:

- (1) 解析函数 $f(t)$ 的分数阶导数 ${}_0 \mathcal{D}_t^{\alpha} f(t)$ 对 t 和 α 都是解析的。
- (2) $\alpha = n$ 为整数时, 分数阶微分与整数阶微分的值完全一致, 且 ${}_0 \mathcal{D}_t^0 f(t) = f(t)$ 。
- (3) 分数阶微积分算子为线性的, 即对任意常数 a, b , 有

$${}_0 \mathcal{D}_t^{\alpha} [af(t) + bg(t)] = a {}_0 \mathcal{D}_t^{\alpha} f(t) + b {}_0 \mathcal{D}_t^{\alpha} g(t) \quad (10-6-10)$$

- (4) 分数阶微积分算子满足交换律, 并满足叠加关系

$${}_0 \mathcal{D}_t^{\alpha} \left[{}_0 \mathcal{D}_t^{\beta} f(t) \right] = {}_0 \mathcal{D}_t^{\beta} \left[{}_0 \mathcal{D}_t^{\alpha} f(t) \right] = {}_0 \mathcal{D}_t^{\alpha+\beta} f(t) \quad (10-6-11)$$

函数的分数阶积分表达式的 Laplace 变换为

$$\mathcal{L} \left[\mathcal{D}_t^{-\gamma} f(t) \right] = s^{-\gamma} \mathcal{L}[f(t)] \quad (10-6-12)$$

在 Riemann–Liouville 定义下, 函数分数阶微分的 Laplace 变换为

$$\mathcal{L} \left[{}^{\text{RL}} \mathcal{D}_t^{\alpha} f(t) \right] = s^{\alpha} \mathcal{L}[f(t)] - \sum_{k=1}^{n-1} s^k \left[{}_0 \mathcal{D}_t^{\alpha-k-1} f(t) \right]_{t=0} \quad (10-6-13)$$

特别地,若函数 $f(t)$ 及其各阶导数的初值均为 0,则 $\mathcal{L}[{}_0\mathcal{D}_t^\alpha f(t)] = s^\alpha \mathcal{L}[f(t)]$ 。

Caputo 定义下函数积分的 Laplace 变换与 Riemann-Liouville 定义下的完全一致,但 Caputo 定义下函数微分的 Laplace 变换满足

$$\mathcal{L} [{}_0^C\mathcal{D}_t^\gamma f(t)] = s^\gamma F(s) - \sum_{k=0}^{n-1} s^{\gamma-k-1} f^{(k)}(0) \quad (10-6-14)$$

10.6.3 分数阶微积分的计算方法

1. 用 Grünwald-Letnikov 定义求解分数阶微分

求解分数阶微积分最直接的数值方法是利用 Grünwald-Letnikov 定义的方法

$${}_a\mathcal{D}_t^\alpha f(t) = \lim_{h \rightarrow 0} \frac{1}{h^\alpha} \sum_{j=0}^{[(t-a)/h]} (-1)^j \binom{\alpha}{j} f(t-jh) \approx \frac{1}{h^\alpha} \sum_{j=0}^{[(t-a)/h]} w_j^{(\alpha)} f(t-jh) \quad (10-6-15)$$

其中, $w_j^{(\alpha)} = (-1)^j \binom{\alpha}{j}$ 为函数 $(1-z)^\alpha$ 的多项式系数,该系数还可以更简单地由下面的递推公式直接求出

$$w_0^{(\alpha)} = 1, \quad w_j^{(\alpha)} = \left(1 - \frac{\alpha+1}{j}\right) w_{j-1}^{(\alpha)}, \quad j = 1, 2, \dots \quad (10-6-16)$$

假设步长 h 足够小,则可以用式(10-6-15)直接求出函数数值微分的近似值,并可以证明^[20],该公式的精度为 $o(h)$ 。所以,利用 Grünwald-Letnikov 定义可以立即编写出下面的函数来求取给定函数的分数阶微分函数。

```
function dy=glfdiff(y,t,gam)
h=t(2)-t(1); dy(1)=0; y=y(:); t=t(:);
w=1; for j=2:length(t), w(j)=w(j-1)*(1-(gam+1)/(j-1)); end
for i=2:length(t), dy(i)=w(1:i)*[y(i:-1:1)]/h^gam; end
```

该函数的调用格式为 $y_1 = \text{glfdiff}(y, t, \gamma)$, 其中, y, t 分别为给定函数的采样值与时刻值构成的向量。要求 t 为等间距向量,且 γ 为分数阶导数的阶次,这样得出的 y_1 向量为函数的分数阶导数。

例 10-42 在整数阶微积分理论的框架下,我们知道,常数的各阶微分均等于零,一阶积分为斜线,高阶积分分别为二次曲线、三次曲线等。试求出常数的分数阶微积分。

解 由下面语句可以先构造常数信号向量 y ,再调用 `glfdiff()` 函数则可以直接得出函数的分数阶微积分曲线,如图 10-50 所示,可见,常数信号的分数阶微分与整数阶是有很大区别的。

```
>> t=0:0.01:1.5; gam=[-1 -0.5 0.3 0.5 0.7]; y=ones(size(t)); dy=[];
for a=gam, dy=[dy; glfdiff(y,t,a)]; end, plot(t,dy)
```

例 10-43 考虑一个初值非零的函数 $f(t) = e^{-t} \sin(3t+1), t \in (0, \pi)$, 试求出其分数阶导数。

解 如果试图采用 Fourier 级数的方法来求解该函数的分数阶微分,则会出现很多问题,所以这里只考虑由 Grünwald-Letnikov 定义来计算其分数阶微分函数。分别选择计算步长 $T = 0.01$ 和

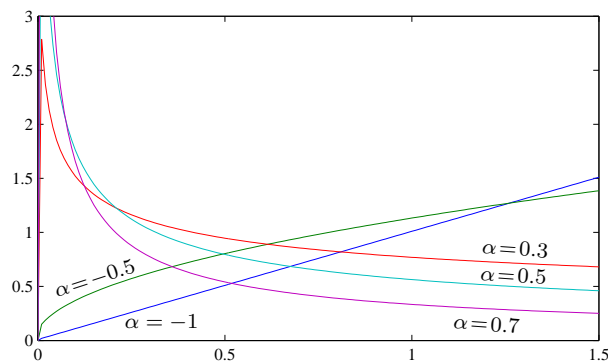


图 10-50 常数的分数阶微积分

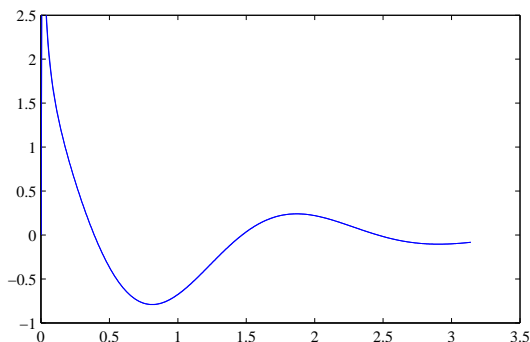
$T = 0.001$, 则可以得出在这两个计算步长下计算出的 0.5 阶导数函数曲线, 如图 10-51 (a) 所示。可见, 在 t 接近 0 的区域外二者是很接近的。对本函数而言, 选择 $T = 0.01$ 可以足够精确地求出函数的分数阶微分。

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1); dy=glfdiff(y,t,0.5); plot(t,dy);
```

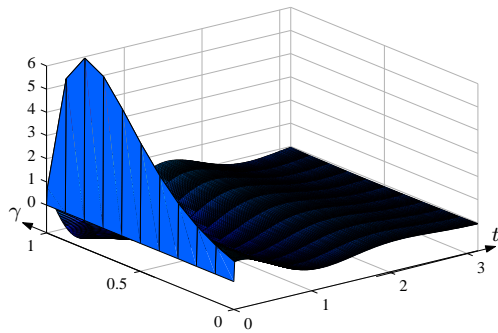
```
t=0:0.01:pi; y=exp(-t).*sin(3*t+1); dy=glfdiff(y,t,0.5); line(t,dy)
```

对不同的 γ 选值, 可以调用下面的语句绘制出分数阶导函数的三维图, 如图 10-51 (b) 所示。

```
>> Z=[]; t=0:0.01:pi; y=exp(-t).*sin(3*t+1);
for gam=0:0.1:1, Z=[Z; glfdiff(y,t,gam)]; end
surf(t,0:0.1:1,Z); axis([0,pi,0,1,-1.2,6])
```



(a) 不同计算步长的结果比较



(b) 分数阶微分曲面

图 10-51 函数的分数阶微分

例 10-44 试用不同定义求取函数 $f(t) = \sin(3t + 1)$ 的 0.75 阶微分, 并比较得出的结果。

解 由 Cauchy 定义, 可以立即求出 0.75 阶微分为 ${}_0\mathcal{D}_t^{0.75} f(t) = 3^{0.75} \sin\left(3t + 1 + \frac{0.75\pi}{2}\right)$, 而用

Grünwald-Letnikov 定义的微分可以由 `glfdiff()` 函数得出。这样, 由下面语句可以绘制出由这两个定义得出的分数阶微分曲线, 如图 10-52 所示。

```
>> t=0:0.01:pi; y=sin(3*t+1); y1=3^0.75*sin(3*t+1+0.75*pi/2);
y2=glfdiff(y,t,0.75); plot(t,y1,t,y2)
```

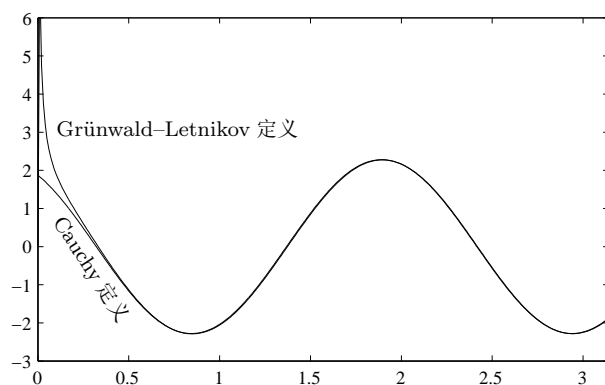


图 10-52 不同定义下的分数阶微分曲线

比较两种定义得出的结果,可见用 Cauchy 积分公式定义的算法没有初始突变过程,在 $t=0$ 区域以外二者几乎是一致的。从得出的结果很难断定哪个定义是正确的,因为这完全取决于 $t \leq 0$ 时 y 的值是什么。若 $y=0$,则显然在 $t=0^+$ 时刻 y 的值从 0 突变到 $\sin 1$,所以这时分数阶微分值应该为 ∞ ,突变的影响亦将持续一段时间,故 Grünwald-Letnikov 定义能正确反映该函数的分数阶微分变化,而不适合用 Cauchy 积分公式;若在 $t \leq 0$ 时原函数仍然满足函数 $y(t) = \sin(3t+1)$,则在 $t=0^+$ 时函数没有突变,则更适合使用 Cauchy 积分公式。

2. Caputo 微积分定义的数值计算

由前面的介绍可见, Caputo 分数阶积分与 Grünwald-Letnikov 定义完全一致,所以可以采用 `glfdiff()` 函数直接求解。若 $\alpha > 0$,可以通过式 (10-6-9) 计算出 Caputo 分数阶微分,这样可以编写出如下的 MATLAB 函数

```
function dy=caputo(y,t,gam,L,vec)
t0=t(1); dy=glfdiff(y,t,gam); if nargin<=3, L=10; end
if gam>0, m=ceil(gam); if gam<=1,vec=y(1); end
    for k=0:m-1, dy=dy-vec(k+1)*(t-t0).^(k-gam)./gamma(k+1-gam); end
    yy1=interp1(t(L+1:end),dy(L+1:end),t(1:L),'spline'); dy(1:L)=yy1;
end
```

该函数的调用格式为 $\mathbf{y}_1 = \text{caputo}(\mathbf{y}, t, \alpha, \mathbf{y}_0, L)$, 其中, $\alpha \leq 0$, 将直接返回 Grünwald-Letnikov 积分结果; 若 $\alpha < 1$, 则由 \mathbf{y} 向量提取 $y(t_0)$ 的值; 若 $\alpha > 1$, 则应该给出 $y(t)$ 及各阶导数的初值向量, 即 $\mathbf{y}_0 = [y(t_0), y'(t_0), \dots, y^{(m-1)}(t_0)]$, 且 $m = \lceil \alpha \rceil$ 。在数值计算中 Caputo 定义下分数阶导数的前若干项可能有较大的误差, 所以实际应用中需要对前 L 项可进行插值处理, 更好地逼近理论结果, L 的默认值为 0, 随着导数阶次增高, 需要加大 L 的取值。

例 10-45 重新考虑例 10-44 的函数 $f(t) = \sin(3t+1)$, 试绘制不同定义下 0.3 阶、1.3 阶、2.3 阶导数曲线。

解 可见在 $t=0$ 时刻函数 $f(t)$ 的初值为 $\sin 1$, 故可以得出二者之差为 $d(t) = t^{-0.3} \sin 1 / \Gamma(0.7)$, 这样可以由下面语句计算出 Grünwald-Letnikov 定义和 Caputo 定义下的函数曲线, 如图 10-53(a)

所示。可见,在初值非零时,二者差异还是很大的。

```
>> t=0:0.01:pi; y=sin(3*t+1); d=t.^(-0.3)*sin(1)/gamma(0.7);
y1=glfdiff(y,t,0.3); y2=caputo(y,t,0.3); plot(t,y1,t,y2,'--',t,d,':')
```

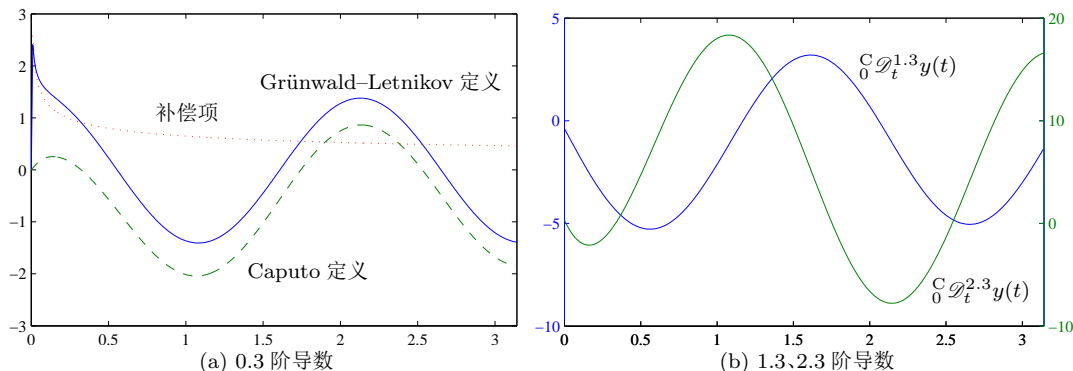


图 10-53 不同定义下的分数阶微分曲线

因为需要求出 ${}_0^C \mathcal{D}_t^{2.3} y(t)$, 所以需要事先求出 $y'(0), y''(0)$ 的值, 这些值可以通过符号运算方法求出并转换成双精度变量。下面语句可以求出函数的 1.3、2.3 阶导数, 为得到较精确的结果, 前若干项需要插值处理, 最终得出如图 10-53(b) 所示的曲线。

```
>> syms t; y=sin(3*t+1); y00=sin(1); y10=double(subs(diff(y,t),t,0));
y20=double(subs(diff(y,t,2),t,0)); t=0:0.01:pi; y=sin(3*t+1);
y1=caputo(y,t,1.3,[y00 y10],15); y2=caputo(y,t,2.3,[y00,y10,y20],40);
plotyy(t,y1,t,y2)
```

3. Oustaloup 滤波算法

前面介绍的各种分数阶微分运算的前提是被微分函数 $f(t)$ 为已知函数, 但在实际应用中该信号经常是无法预先知道的, 因为这些信号可能来自别的系统环节, 所以应该采用其他形式来求取分数阶微分, 例如通过构造滤波器的方式来对信号进行数值微积分处理。

信号的滤波器可以有连续和离散两种形式, 分别用来拟合 Laplace 变换算子 s^γ 和 Fourier 变换算子 $(j\omega)^\gamma$ 。从效果上看, 函数的分数阶数值微分相当于原来信号需要通过这样的滤波器得出的输出信号。

文献 [21] 中列出了多种连续滤波器的实现算法。这里只介绍其中的 Oustaloup 算法 [22]。假设选定的拟合频率段为 (ω_b, ω_h) , 则可以构造出连续滤波器的传递函数模型为

$$G_f(s) = K \prod_{k=1}^N \frac{s + \omega'_k}{s + \omega_k} \quad (10-6-17)$$

其中, 滤波器零极点和增益可以由式 (10-6-18) 直接求出, 为

$$\omega'_k = \omega_b \omega_u^{(2k-1-\gamma)/N}, \quad \omega_k = \omega_b \omega_u^{(2k-1+\gamma)/N}, \quad K = \omega_h^\gamma, \quad \text{其中 } \omega_u = \sqrt{\omega_h/\omega_b} \quad (10-6-18)$$

根据上述算法, 可以直接编写出如下的函数, 设计连续滤波器。这样, 若 $y(t)$ 信号通过滤波器进行过滤, 则可以认为输出信号是 $\mathcal{D}_t^\gamma y(t)$ 的近似。

```
function G=ousta_fod(gam,N,wb,wh)
k=1:N; wu=sqrt(wh/wb);
wkp=wb*wu.^((2*k-1-gam)/N); wk=wb*wu.^((2*k-1+gam)/N);
G=zpk(-wkp,-wk,wh^gam); G=tf(G);
```

该函数的调用格式为 $G_1 = \text{ousta_fod}(\gamma, N, \omega_b, \omega_h)$, 其中, γ 为分数阶的阶次, N 为滤波器的阶次, ω_b 和 ω_h 分别为用户选定的拟合频率下限和上限, 一般在该区域内能较好地拟合分数阶微分算子, 而其外的区域将和微分算子相差很多。

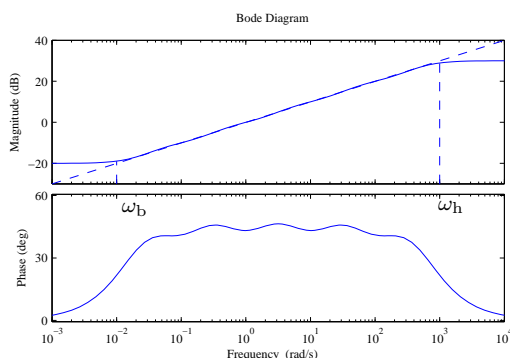
例 10-46 假设 $\omega_b = 0.01, \omega_h = 1000 \text{ rad/sec}$, 试设计出连续滤波器, 对 $f(t) = e^{-t} \sin(3t + 1)$ 信号计算 0.5 阶微分。

解 可以用下面的语句设计出滤波器

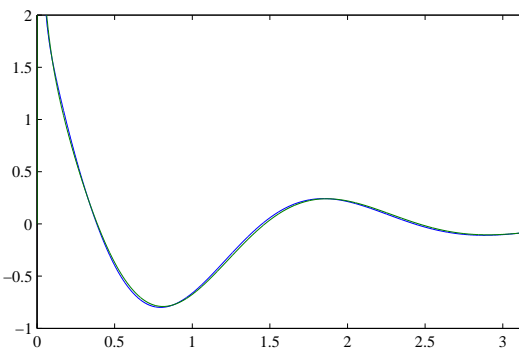
$$G(s) = \frac{31.62s^5 + 6248s^4 + 1.122 \times 10^5 s^3 + 1.996 \times 10^5 s^2 + 3.514 \times 10^4 s + 562.3}{s^5 + 624.8s^4 + 3.549 \times 10^4 s^3 + 1.996 \times 10^5 s^2 + 1.111 \times 10^5 s + 5623}$$

```
>> G=ousta_fod(0.5,5,0.01,1000), bode(G)
```

上面的语句还可以直接绘制出该滤波器的 Bode 图, 如图 10-54 (a) 所示, 在该图中还叠印了直线, 表示 $(j\omega)^\gamma$ 的理论值。由下面的语句还可以绘制出由滤波器计算出来的分数阶微分曲线, 同时也将绘制出由 Grünwald-Letnikov 定义计算出来的分数阶微分曲线, 如图 10-54 (b) 所示。可见, 由滤波器计算出来的分数阶微分结果还是很精确的。



(a) Oustaloup 滤波器 Bode 图



(b) 分数阶导数曲线

图 10-54 函数的分数阶导数

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1);
y1=lsim(G,y,t); y2=glfdiff(y,t,0.5); plot(t,y1,t,y2)
```

当然, 用该算法还可以在更大的频率范围内拟合分数阶微分函数, 这时需要适当增大拟合的阶次。下面给出在 $(10^{-4}, 10^4)$ 频段内的拟合效果, 如图 10-55 所示。可见, 对这样大的频率范围, 不再适合 $N = 2$ 的近似, 而应该采用更大的 N 值, 如 $N = 4$ 。

```
>> G=ousta_fod(0.5,5,1e-4,1e4); G1=ousta_fod(0.5,7,1e-4,1e4);
G2=ousta_fod(0.5,9,1e-4,1e4); G3=ousta_fod(0.5,11,1e-4,1e4);
bode(G,'-',G1,'--',G2,':',G3,'-.')
```

4. 改进的 Oustaloup 滤波器方法

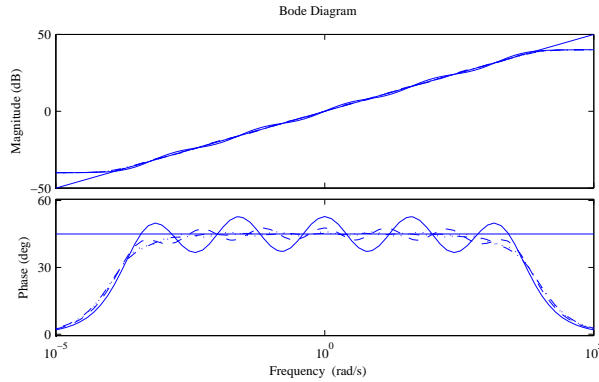


图 10-55 不同阶次下的滤波器近似效果

从前面演示的 Oustaloup 滤波器的拟合范围看,在 ω_h 和 ω_b 边界附近,拟合效果是相当不理想的,而整个拟合的效果,尤其是相位拟合结果是很差的,所以应该考虑对其进行改进,例如引入如下的滤波器 [23]

$$s^\gamma \approx \left(\frac{d\omega_h}{b}\right)^\gamma \left(\frac{ds^2 + b\omega_h s}{d(1-\gamma)s^2 + b\omega_h s + d\gamma}\right) \prod_{k=-N}^N \frac{s + \omega'_k}{s + \omega_k} \quad (10-6-19)$$

其中

$$\omega'_k = (d\omega_b/b)^{(\gamma-2k)/(2N+1)}, \quad \omega_k = (b\omega_h/d)^{(\gamma+2k)/(2N+1)} \quad (10-6-20)$$

通常可以选择加权参数 $b = 10, d = 9$ 。根据上述算法可以编写出下面的 MATLAB 函数

```
function G=new_fod(r,N,wb,wh,b,d)
if nargin==4, b=10; d=9; end
mu=wh/wb; k=-N:N; w_kp=(mu).^((k+N+0.5-0.5*r)/(2*N+1))*wb;
w_k=(mu).^((k+N+0.5+0.5*r)/(2*N+1))*wb; K=(d*wh/b)^r;
G=zpk(-w_kp',-w_k',K)*tf([d,b*wh,0],[d*(1-r),b*wh,d*r]);
```

其调用格式为 $G_f = \text{new_fod}(\gamma, N, \omega_b, \omega_h, b, d)$, 其中 b, d 可以忽略。当前的 newfod() 函数本身有局限性,要求 $\omega_h \omega_b = 1$, 否则拟合效果可能不理想。

例 10-47 重新考虑例 10-46 中的问题,选择 $\omega_b = 0.001, \omega_h = 1000$, 试观察新的滤波逼近和分数阶导数求取效果。

解 采用两种不同的滤波器,则可以得出它们的 Bode 图,如图 10-56(a) 所示,而分数阶微分结果的比较如图 10-56(b) 所示。可见,无论从幅值还是相位的拟合效果看,改进方法得出的结果明显优于传统的 Oustaloup 滤波器。

```
>> G1=ousta_fod(0.5,2,0.001,1000); G2=new_fod(0.5,2,0.001,1000);
bode(G1,'-',G2,'--'), figure; t=0:0.001:pi; y=exp(-t).*sin(3*t+1);
y1=lsim(G1,y,t); y2=lsim(G2,y,t); y0=glfdiff(y,t,0.5); plot(t,y1,t,y2,t,y0)
```

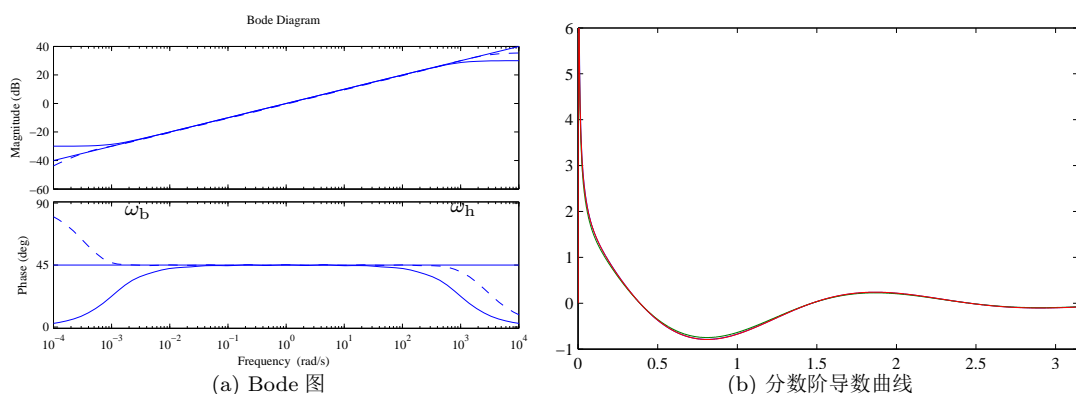



图 10-56 不同滤波器下函数的分数阶导数

10.6.4 分数阶线性微分方程的求解方法

分数阶线性微分方程的一般形式为^[20]

$$\begin{aligned} a_n \mathcal{D}_t^{\beta_n} y(t) + a_{n-1} \mathcal{D}_t^{\beta_{n-1}} y(t) + \cdots + a_1 \mathcal{D}_t^{\beta_1} y(t) + a_0 \mathcal{D}_t^{\beta_0} y(t) \\ = b_1 \mathcal{D}_t^{\gamma_1} u(t) + b_2 \mathcal{D}_t^{\gamma_2} u(t) + \cdots + b_m \mathcal{D}_t^{\gamma_m} u(t) \end{aligned} \quad (10-6-21)$$

其中初值为零的微分方程称为 Riemann–Liouville 微分方程,若初值非零则称为 Caputo 微分方程,本节将侧重于介绍两类微分方程的数值解方法。如果初始条件均为零,该线性微分方程还可以用下面的分数阶传递函数直接描述

$$G(s) = \frac{b_1 s^{\gamma_1} + b_2 s^{\gamma_2} + \cdots + b_m s^{\gamma_m}}{a_1 s^{\beta_1} + a_2 s^{\beta_2} + \cdots + a_{n-1} s^{\beta_{n-1}} + a_n s^{\beta_n}}. \quad (10-6-22)$$

1. 一类分数阶线性系统时域响应解析解方法

类似于整数阶函数的部分分式展开法,求解一类线性系统时域响应解析解可以通过引入 Mittag–Leffler 函数来获得。如果微分方程右侧只含有输入信号本身,则由 n 项构成的分数阶微分方程的解可以表示为

$$\begin{aligned} y(t) = & \frac{1}{a_n} \sum_{m=0}^{\infty} \frac{(-1)^m}{m!} \sum_{\substack{k_0+k_1+\cdots+k_{n-2}=m \\ k_0 \geq 0, \cdots, k_{n-2} \geq 0}} (m; k_0, k_1, \cdots, k_{n-2}) \\ & \prod_{i=0}^{n-2} \left(\frac{a_i}{a_n} \right)^{k_i} t^{(\beta_n - \beta_{n-1})m + \beta_n + \sum_{j=0}^{n-2} (\beta_{n-1} - \beta_j)k_j - 1} \\ & \mathcal{E}_{\alpha, \beta}^{(m)} \left(-\frac{a_{n-1}}{a_n} t^{\beta_n - \beta_{n-1}} \right) \end{aligned} \quad (10-6-23)$$

式中, $\mathcal{E}_{\alpha, \beta}(x)$ 为式(8-5-23)中定义的两参数 Mittag–Leffler 函数, m 为整数。

一般系统的时域响应解析解表达式(10-6-23)有时过于复杂,所以这里只考虑一个特殊

模型 $G(s) = 1/(a_2 s^{\beta_2} + a_1 s^{\beta_1} + a_0)$ 的阶跃响应解^[24]

$$y(t) = \frac{1}{a_2} \sum_{k=0}^{\infty} \frac{(-1)^k \hat{a}_0^k t^{-\hat{a}_1 + (k+1)\beta_2}}{k!} \mathcal{E}_{\beta_2 - \beta_1, \beta_2 + \beta_1 k + 1}^{(k)} \left(-\hat{a}_1 t^{\beta_2 - \beta_1} \right) \quad (10-6-24)$$

其中 $\hat{a}_0 = a_0/a_2, \hat{a}_1 = a_1/a_2$ 。可以用累加方法编写出下面的阶跃响应数值求解函数

```
function y=ml_step(a0,a1,a2,b1,b2,t,eps0)
y=0; k=0; ya=1; a0=a0/a2; a1=a1/a2; if nargin==6, eps0=eps; end
while max(abs(ya))>=eps0
    ya=(-1)^k/gamma(k+1)*a0^k*t.^((k+1)*b2).*...
        ml_func([b2-b1,b2+b1*k+1],-a1*t.^(b2-b1),k,eps0);
    y=y+ya; k=k+1;
end
y=y/a2;
```

该函数的调用格式为 **$y = \text{ml_step}(a_0, a_1, a_2, \beta_1, \beta_2, t, \epsilon_0)$** 。

例 10-48 试求出零初值分数阶微分方程 $\mathcal{D}^{0.8}y(t) + 0.75\mathcal{D}^{0.4}y(t) + 0.9y(t) = u(t)$, 其中输入信号为阶跃函数。

解 显然, $a_0 = 0.9, a_1 = 0.75, a_2 = 1, \beta_1 = 0.4, \beta_2 = 0.8$, 这样由下面的语句可以得出原系统阶跃响应的数值解, 得出的曲线如图 10-57 所示。

```
>> t=0:0.001:5; y=ml_step(0.9,0.75,1,0.4,0.8,t); plot(t,y)
```

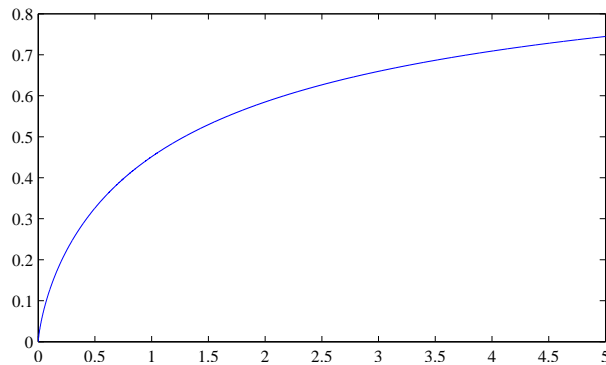


图 10-57 方程解的曲线

从前面实际例子的仿真可见, 由基于 Grünwald–Letnikov 定义的分数阶微分计算得出的阶跃响应计算量比 Mittag–Leffler 函数的算法计算量明显小, 适用面广很多, 所以在后面的研究中侧重于该方法的使用。

2. 零初值分数阶线性微分方程的解法

如果输入和输出信号 $y(t)$ 和 $u(t)$ 及其导函数在初始时刻的值均为零, 等号右侧只有输入信号 $\hat{u}(t)$ 本身, 则微分方程可以简化为

$$a_n \mathcal{D}_t^{\beta_n} y(t) + a_{n-1} \mathcal{D}_t^{\beta_{n-1}} y(t) + \cdots + a_1 \mathcal{D}_t^{\beta_1} y(t) + a_0 \mathcal{D}_t^{\beta_0} y(t) = \hat{u}(t) \quad (10-6-25)$$

其中, $\hat{u}(t)$ 可以由某函数及其分数阶微分构成, 可以事先计算出来

$$\hat{u}(t) = b_1 \mathcal{D}_t^{\gamma_1} u(t) + b_2 \mathcal{D}_t^{\gamma_2} u(t) + \cdots + b_m \mathcal{D}_t^{\gamma_m} u(t) \quad (10-6-26)$$

为后面叙述方便, 不妨对微分方程式作假设 $\beta_n > \beta_{n-1} > \cdots > \beta_1 > \beta_0 > 0$ 。若需要研究的微分方程出现下面两种特殊情况, 则需要先对其变换, 再进行求解。

(1) 若研究的微分方程各个阶次不满足上述大小关系, 则可以先进行排序。

(2) 若涉及负的 β_i 值, 则原来的方程为分数阶微积分方程, 需要选择引入新的变量 $z(t) = \mathcal{D}_t^{\beta_0} y(t)$, 这样就可以将原来的微积分方程变换成关于 $z(t)$ 的微分方程了。

考虑式 (10-6-15) 中给出的 Grünwald-Letnikov 定义, 用离散方法可以将其改写成

$${}_a \mathcal{D}_t^{\beta_i} y(t) \approx \frac{1}{h^{\beta_i}} \sum_{j=0}^{[(t-a)/h]} w_j^{(\beta_i)} y_{t-jh} = \frac{1}{h^{\beta_i}} \left[y_t + \sum_{j=1}^{[(t-a)/h]} w_j^{(\beta_i)} y_{t-jh} \right] \quad (10-6-27)$$

其中, $w_0^{(\beta_i)}$ 可以由下面的递推公式得出

$$w_0^{(\beta_i)} = 1, \quad w_j^{(\beta_i)} = \left(1 - \frac{\beta_i + 1}{j} \right) w_{j-1}^{(\beta_i)}, \quad j = 1, 2, \cdots \quad (10-6-28)$$

代入式 (10-6-25), 则可以直接推导出微分方程数值解为

$$y_t = \frac{1}{\sum_{i=0}^n \frac{a_i}{h^{\beta_i}}} \left[\hat{u}_t - \sum_{i=0}^n \frac{a_i}{h^{\beta_i}} \sum_{j=1}^{[(t-a)/h]} w_j^{(\beta_i)} y_{t-jh} \right] \quad (10-6-29)$$

现在考虑式 (10-6-21) 中给出的一般形式。如果先对等号右侧的函数 $u(t)$ 求分数阶导数, 则显然可以将原方程变换成右侧为 $\hat{u}(t)$ 的形式, 这样套用上述公式就可以求出一般微分方程的数值解。在实际编程运算中, 先求导可能导致计算误差, 故可以考虑先求在 $u(t)$ 激励下的 $\hat{y}(t)$, 再对得出的 $\hat{y}(t)$ 按照等号右侧的方式求导。对线性系统来说这个方法是完全等效的。基于这个算法, 可以编写出一个 `fode_sol()` 来实现任意输入的零初值分数阶线性微分方程的数值解法。

```
function y=fode_sol(a,na,b,nb,u,t)
h=t(2)-t(1); D=sum(a./[h.^na]); nT=length(t); vec=[na nb]; W=[];
D1=b(:)./h.^nb(:); nA=length(a); y1=zeros(nT,1); W=ones(nT,length(vec));
for j=2:nT, W(j,:)=W(j-1,:).*(1-(vec+1)/(j-1)); end
for i=2:nT,
    A=[y1(i-1:-1:1)]'*W(2:i,1:nA); y1(i)=(u(i)-sum(A.*a./[h.^na]))/D;
end
for i=2:nT, y(i)=(W(1:i,nA+1:end)*D1)'*[y1(i:-1:1)]; end
```

该函数的调用格式为 `y = fode_sol(a, na, b, nb, u, t)`, 其中时间向量和输入点向量分别由 t 和 u 给出。注意, 当计算点需要很多时, 这样的求解方法可能比较慢。

例 10-49 试用数值方法求解下面的零初值分数阶线性微分方程并绘制输出函数曲线。

$$\mathcal{D}_t^{3.5}y(t) + 8\mathcal{D}_t^{3.1}y(t) + 26\mathcal{D}_t^{2.3}y(t) + 73\mathcal{D}_t^{1.2}y(t) + 90\mathcal{D}_t^{0.5}y(t) = 90\sin t^2$$

解 由给出的方程可以写出 \mathbf{a} 和 \mathbf{n} 向量,从而直接调用编写的 `fode_sol()` 函数得出该微分方程的解,用绘图语句可以绘制出输出和输入信号的曲线,如图 10-58 所示。为提高得出数值解的精度,通常需要选择较小的 h 值,这里得出的结果精度较高,再进一步减小 h 的值,例如选择 $h = 0.001$,则得出的仿真结果与图中给出的结果看不出任何区别。

```
>> a=[1,8,26,73,90]; n=[3.5,3.1,2.3,1.2,0.5];
    t=0:0.002:10; u=90*sin(t.^2); y=fode_sol(a,n,1,0,u,t);
    subplot(211), plot(t,y); subplot(212), plot(t,u)
```

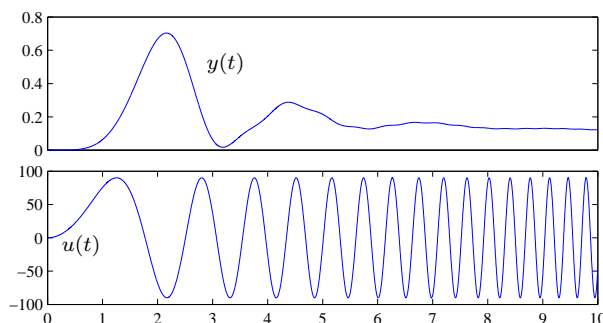


图 10-58 方程解及输入信号

3. 非零初值 Caputo 微分方程的数值求解

如果微分方程中输入、输出变量及其各阶导数的初值非零,则前面使用的方法不能求取方程的数值解,需要使用 Caputo 定义下微分方程的求解方法。

考虑下面给出的 Caputo 线性分数阶微分方程的一般形式

$$a_n {}^C\mathcal{D}_t^{\beta_n}y(t) + a_{n-1} {}^C\mathcal{D}_t^{\beta_{n-1}}y(t) + \cdots + a_1 {}^C\mathcal{D}_t^{\beta_1}y(t) + a_0 {}^C\mathcal{D}_t^{\beta_0}y(t) = \hat{u}(t) \quad (10-6-30)$$

为方便起见,假设 $\beta_n > \beta_{n-1} > \cdots > \beta_1 > \beta_0 \geq 0$ 。等号右侧只含有 $\hat{u}(t)$ 函数。如果实际方程等号右侧含有输入信号 $u(t)$ 的分数阶导数,则可以仿照前面的方法先将其线性组合 $\hat{u}(t)$ 计算出来。

如果 $m = \lceil \beta_n \rceil$,则唯一求解该分数阶微分方程应该已知 m 个初始值, $y(0), y'(0), \cdots, y^{(m-1)}(0)$ 。这样,可以引入辅助变量 $z(t)$

$$z(t) = y(t) - y(0) - y'(0)t - \cdots - y^{(m-1)}(0)t^{m-1} \quad (10-6-31)$$

这时 $z(t)$ 信号及其前 $m-1$ 阶导数的初值均为 0。对上式稍加变换,则

$$y(t) = z(t) + y(0) + y'(0)t + \cdots + y^{(m-1)}(0)t^{m-1} \quad (10-6-32)$$

因为 $z(t)$ 信号及各阶导数的初值均为零,显然 ${}^C\mathcal{D}_t^{\beta_i}z(t) = {}^{RL}\mathcal{D}_t^{\beta_i}z(t)$, 多项式 $y(0) + y'(0)t + \cdots + y^{(m-1)}(0)t^{m-1}$ 的 β_i 阶 Caputo 导数可以通过下面的函数直接求出

```
function s=poly2caputo(a,r), syms u tau;
s=int(diff(poly2sym(a,'tau'),ceil(r))/(u-tau)^(r-ceil(r)+1))...
/gamma(ceil(r)-r),tau,0,u);
```

该函数的调用格式为 $s = \text{poly2caputo}(a, \beta)$, 其中, $a = [y^{(m-1)}, \dots, y'(0), y(0)]$ 为初始条件向量, β 为分数阶微分的阶次, 返回的 s 为多项式分数阶导数符号表达式, 自变量为 u 。

借助于前面给出的补偿函数, 可以将原微分方程变换为

$$\begin{aligned} a_n^{\text{RL}} \mathcal{D}_t^{\beta_n} y(t) + a_{n-1}^{\text{RL}} \mathcal{D}_t^{\beta_{n-1}} y(t) + \dots + a_1^{\text{RL}} \mathcal{D}_t^{\beta_1} y(t) + a_0^{\text{RL}} \mathcal{D}_t^{\beta_0} y(t) \\ = \hat{u}(t) - \sum_{i=0}^n a_i^{\text{C}} \mathcal{D}_t^{\beta_i} \left[y(0) + y'(0)t + \dots + y^{(m-1)}(0)t^{m-1} \right] \end{aligned} \quad (10-6-33)$$

仿照 `fode_sol()` 函数, 可以编写出下面的 Caputo 微分方程求解函数, 该函数目前只能处理右端只含有 $u(t)$ 的微分方程。该函数的调用格式为 $y = \text{fode_caputo}(a, n_a, y_0, u, t)$, 其中, u 为输入信号的采样点, t 为时间向量。

```
function [y,z]=fode_caputo(a,na,y0,u,t)
h=t(2)-t(1); D=sum(a./[h.^na]); nT=length(t); nb=0; b=1; vec=[na nb]; W=[];
D1=b(:)./h.^nb(:); nA=length(a); y1=zeros(nT,1); W=ones(nT,length(vec));
for i=1:length(a), u=u-a(i)*subs(poly2caputo(y0,na(i)),'u',t); end
for j=2:nT, W(j,:)=W(j-1,:).*(1-(vec+1)/(j-1)); end
for i=2:nT,
    A=[y1(i-1:-1:1)]'*W(2:i,1:nA); y1(i)=(u(i)-sum(A.*a./[h.^na]))/D;
end
z=y1'; y=z+polyval(y0,t);
```

例 10-50 重新考虑例 10-49 的线性分数阶微分方程, 如果该方程是 Caputo 微分方程, 且已知初始条件为 $y(0) = 1, y'(0) = -1, y''(0) = 2, y'''(0) = 3$, 试重新求解该方程。

解 由给出的初始条件构造出初始条件向量, 则可以调用下面的语句直接求解 Caputo 微分方程, 得出的解函数如图 10-59 所示。

```
>> a=[1,8,26,73,90]; n=[3.5,3.1,2.3,1.2,0.5]; t=0:0.001:10; u=90*sin(t.^2);
y0=[3 2 -1 1]; y=fode_caputo(a,n,y0,u,t); plot(t,y)
```

10.6.5 基于框图的非线性分数阶微分方程近似解法

如果给出的非线性分数阶微分方程是非线性的微分方程, 特别地, 该微分方程是整个系统中的一部分, 则用常规求解方法不能得出原问题的数值解, 必须使用基于框图的求解方法。前面介绍过, 可以考虑 Oustaloup 滤波器或其他改进形式的滤波器, 用高阶整数阶模块逼近原始的分数阶算子, 这样就可以搭建起非线性分数阶微分方程的求解框图, 最终得出微分方程的数值解。本节将分别介绍一般零初值问题和非零初值 Caputo 微分方程的数值求解方法。

1. 零初值非线性分数阶微分方程的求解

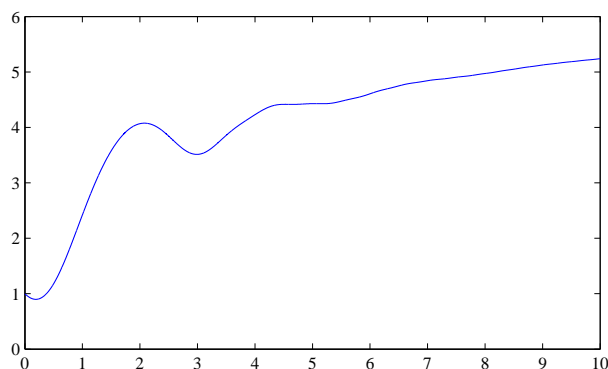
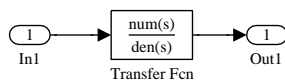


图 10-59 Caputo 方程的数值解

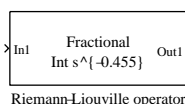
由前面的内容可见,对未知信号进行分数阶微分数值运算的一种有效途径是采用 Oustaloup 算法设计连续滤波器对信号进行滤波处理。另外,考虑到该滤波器分子和分母阶次一致,可能导致在仿真过程中出现代数环,所以应该在其后面再接一个低通滤波器,将其截止频率设置为 ω_h ,这样可以建立起如图 10-60 (a) 所示的分数阶微分器模块,通过适当选择频段和阶次可以较好地近似分数阶微分的效果。注意,虽然 Oustaloup 算法设计的滤波器理论上可以求取任意阶次的分数阶微积分,但从数值微积分精度看,该滤波器更适合求取 1 阶以内的分数阶微积分,所以应该将高阶微积分先进行整数阶微积分运算,再对结果进行滤波处理。注意,这里的方法只适用于零初值问题的求解,非零初值问题后面将介绍。

利用 Simulink 的模块封装技术^[25],可以将该模型进行封装,得出如图 10-60 (b) 所示的分数阶微分器模块。双击该模块则可以得出如图 10-60 (c) 所示的对话框,允许用户填写设计 Oustaloup 滤波器所需的参数。在模块封装初始化栏目应该填写下面的语句,以便在使用模块前先自动设计出滤波器,并根据阶次正确显示图标。

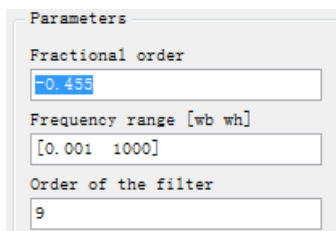
```
wb=ww(1); wh=ww(2); G=ousta_fod(gam,n,wb,wh);
num=G.num{1}; den=G.den{1}; str='Fractional\ n';
if isnumeric(gam)
    if gam>0, str=[str, 'Der s^' num2str(gam)];
    else, str=[str, 'Int s^{ ' num2str(gam) ' }']; end
else, str=[str, 'Der s^gam']; end
```



(a) 分数阶微分滤波器



(b) 封装模块



(c) 分数阶微分器参数对话框

图 10-60 Riemann-Liouville 分数阶算子模块

在实际仿真过程中,由于搭建起来的系统一般为刚性系统,所以在选择求解算法时应该选择 ode15s 或 ode23tb 等,因为这些算法可以保证较高的计算效率和精度。下面将通过例子演示该模块在分数阶微分方程近似求解中的应用。

例 10-51 试用滤波器的思想求取例 10-49 中分数阶线性微分方程的数值解,并与该例中所用方法得出的结果进行比较。

解 求解分数阶线性微分方程问题不如例 10-49 中给出的方法直观。在求解之前,需要引入辅助变量 $z(t) = \mathcal{D}_t^{0.5}y(t)$,这样,原来的微分方程可以直接变换成下面的形式

$$z(t) = \sin t^2 - \frac{1}{90} [\mathcal{D}_t^3 z(t) + 8\mathcal{D}_t^{2.6} z(t) + 26\mathcal{D}_t^{1.8} z(t) + 73\mathcal{D}_t^{0.7} z(t)]$$

根据该方程可以搭建起如图 10-61 所示的 Simulink 仿真框图。对该框图进行仿真,则可以得出该微分方程的数值解。将两种方法得出的数值解在同一坐标系下绘制,则得出如图 10-62 所示的曲线。从曲线上基本看不出两者的差别,表明此算法可以以很高精度求解线性方程。

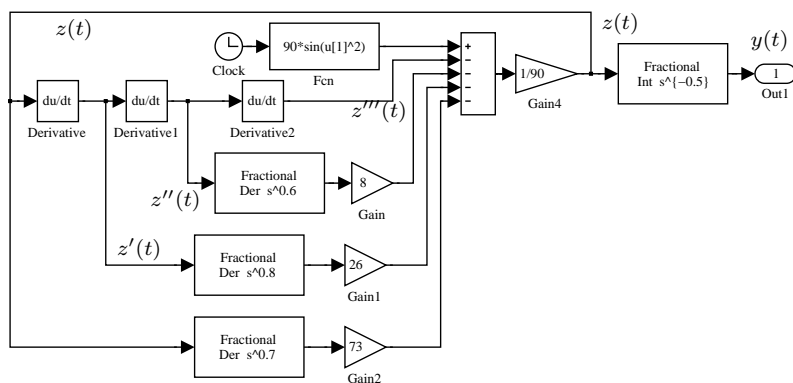


图 10-61 微分方程求解的 Simulink 框图 (文件名: c10fode1.mdl)

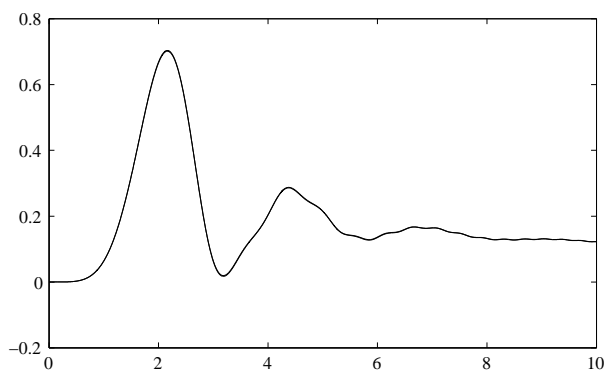


图 10-62 两种数值解方法比较

例 10-52 试用近似方法求解下面的分数阶非线性微分方程

$$\frac{3\mathcal{D}^{0.9}y(t)}{3 + 0.2\mathcal{D}^{0.8}y(t) + 0.9\mathcal{D}^{0.2}y(t)} + |2\mathcal{D}^{0.7}y(t)|^{1.5} + \frac{4}{3}y(t) = 5\sin 10t$$

解 根据方程本身,可以容易地写出 $y(t)$ 函数的显式表达式为

$$y(t) = \frac{3}{4} \left[5 \sin 10t - \frac{3 \mathcal{D}^{0.9} y(t)}{3 + 0.2 \mathcal{D}^{0.8} y(t) + 0.9 \mathcal{D}^{0.2} y(t)} - |2 \mathcal{D}^{0.7} y(t)|^{1.5} \right]$$

根据得出的 $y(t)$ 可以绘制出如图 10-63 (a) 所示的仿真模型。从得出的仿真模型可见, 信号的各个分数阶微分信号可以由前面设计的模块获得, 因此仿真的精度取决于滤波器对微分的拟合效果, 选择不同的拟合频段和滤波器阶次对求解精度将有一定的影响。图 10-63 (b) 对不同的滤波器频段、阶次组合进行了比较, 得出的结果基本一致, 误差稍大的曲线是由 $\omega_b = 0.001, \omega_h = 1000, n = 5$ 得出的。所以对此例来说, 选择 $n = 9$ 并选择适当的频段得出的结果几乎完全一致。

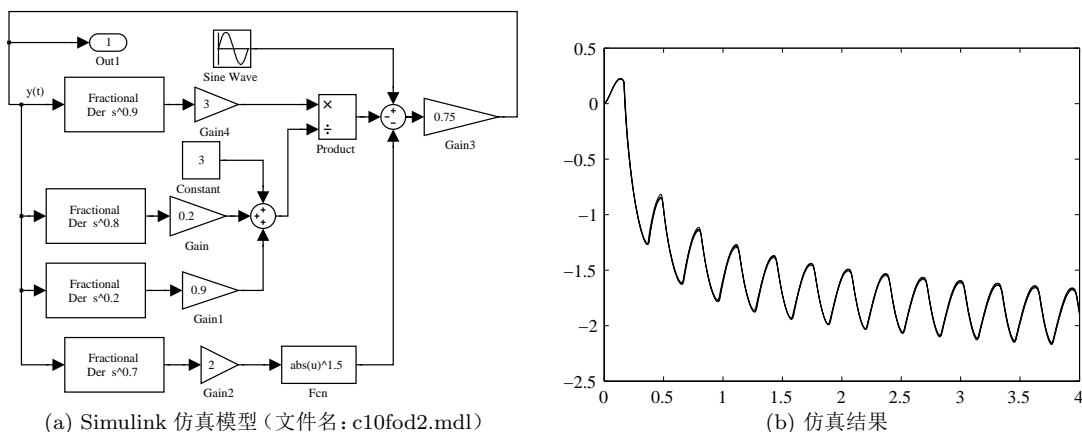


图 10-63 非线性分数阶微分方程的 Simulink 描述及仿真结果

2. 非零初值的 Caputo 微分方程数值解法

类似于前面介绍的 Riemann–Liouville 微分算子的模块封装方法, 同样可以设计出 Caputo 算子的封装模块, 如图 10-64 所示。

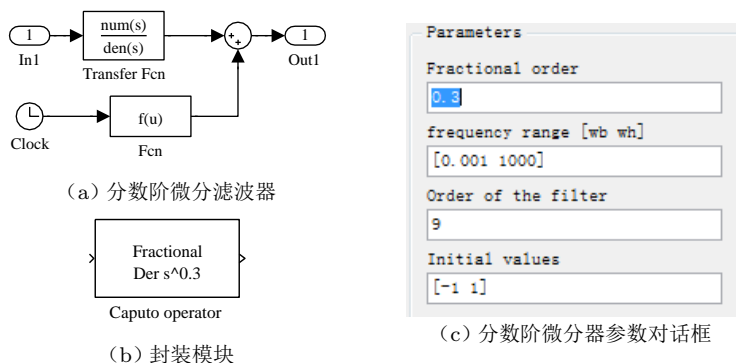


图 10-64 Caputo 分数阶算子模块

该模块的初始化栏目的内容修改成

```
wb=ww(1); wh=ww(2); G=ousta_fod(gam,n,wb,wh); G1=tf(G);
num=G1.num{1}; den=G1.den{1}; str='Fractional\''n';
strmodel=[get_param(gcs,'Name'),'/' get_param(gcb,'Name') '/Fcn'];
set_param(strmodel,'Expr',char(poly2caputo(a,gam)));
```



```

if isnumeric(gam)
    if gam>0, str=[str, 'Der s^' num2str(gam) ];
    else, str=[str, 'Int s^{ ' num2str(gam) '}']; end
else, str=[str, 'Der s^gam']; end

```

其中,函数 `poly2caputo()` 用于填写 `Fcn` 模块的补偿函数,其内容前面已经给出。

假设分数阶微分方程中的最高阶次 α 满足 $m = \lceil \alpha \rceil$,则需要已知 m 个初始值, $y(0)$, $y'(0)$, $y''(0), \dots, y^{(m-1)}(0)$ 。可以引入辅助变量

$$z(t) = y(t) - y(0) - y'(0)t - y''(0)t^2 - \dots - y^{(m-1)}(0)t^{m-1} \quad (10-6-34)$$

这样,关于 $y(t)$ 信号的 Caputo 微分方程可以直接变换成关于 $z(t)$ 信号的 Riemann–Liouville 微分方程,该方程可以由前面介绍的基于框图的方法直接求解。下面通过例子演示这类微分方程的数值求解方法。

例 10-53 试求解下面 Caputo 定义下的微分方程^[26]

$${}_0^C \mathcal{D}_t^{1.455} y(t) = -t^{0.1} \frac{\mathcal{E}_{1.445}^{1.545}(-t)}{\mathcal{E}_{1.445}^{1.445}(-t)} e^t y(t) {}_0^C \mathcal{D}_t^{0.555} y(t) + e^{-2t} - [{}_0^C \mathcal{D}_t^1 y(t)]^2$$

其中, $y(0) = 1, y'(0) = -1$, 且已知其解析解为 $y = e^{-t}$ 。

解 引入辅助变量 $z(t) = y(t) - y(0) - y'(0)t = y(t) - 1 + t$, 则由 $y(t) = z(t) + 1 - t$ 可以将原方程转换成关于 $z(t)$ 的零初值非线性分数阶微分方程

$${}_0^C \mathcal{D}_t^{1.455} [z(t) + 1 - t] = -t^{0.1} e^t \frac{\mathcal{E}_{1.445}^{1.545}(-t)}{\mathcal{E}_{1.445}^{1.445}(-t)} y(t) {}_0^C \mathcal{D}_t^{0.555} [z(t) + 1 - t] + e^{-2t} - [{}_0^C \mathcal{D}_t^1 [z(t) + 1 - t]]^2$$

由式 (10-6-9) 可知, 零初值信号 $z(t)$ 满足 ${}_0^C \mathcal{D}^\alpha z(t) = {}_0^{\text{RL}} \mathcal{D}^\alpha z(t)$ 。另外, 由 Caputo 定义可见, 因为求取左侧的 ${}_0^C \mathcal{D}_t^{1.455} [z(t) + 1 - t]$ 需要对函数本身求 2 阶导数, 增广的 $1 - t$ 项的二阶导数为 0, 这样, ${}_0^C \mathcal{D}_t^{1.455} [z(t) + 1 - t] = {}_0^{\text{RL}} \mathcal{D}_t^{1.455} z(t)$ 。原来的 Caputo 微分方程可以转化为下面的形式

$${}_0^{\text{RL}} \mathcal{D}_t^{1.455} z(t) = -t^{0.1} e^t \frac{\mathcal{E}_{1.445}^{1.545}(-t)}{\mathcal{E}_{1.445}^{1.445}(-t)} y(t) {}_0^C \mathcal{D}_t^{0.555} y(t) + e^{-2t} - [z'(t) - 1]^2$$

方程内部的 ${}_0^C \mathcal{D}_t^{0.555} y(t)$ 可以由前面介绍的 Caputo operator 模块直接表示。由前面给出的推导可以立即建立起该分数阶微分方程的 Simulink 仿真模型, 如图 10-65 所示。另外, 方程中用到了 Mittag–Leffler 函数, 需要将其用 S-函数的形式表示出来

```

function [sys,x0,str,ts]=sfun_mls(t,x,u,flag,a)
switch flag
case 0, sizes=simsizes;
    sizes.NumContStates=0; sizes.NumDiscStates=0; sizes.NumOutputs=1;
    sizes.NumInputs=1; sizes.DirFeedthrough=1; sizes.NumSampleTimes=1;
    sys=simsizes(sizes); x0=[]; str=[]; ts=[-1 0];
case 3, sys=ml_func(a,u);
case {1,2,4,9}, sys=[];
otherwise, error(['Unhandled flag=',num2str(flag)]);
end

```

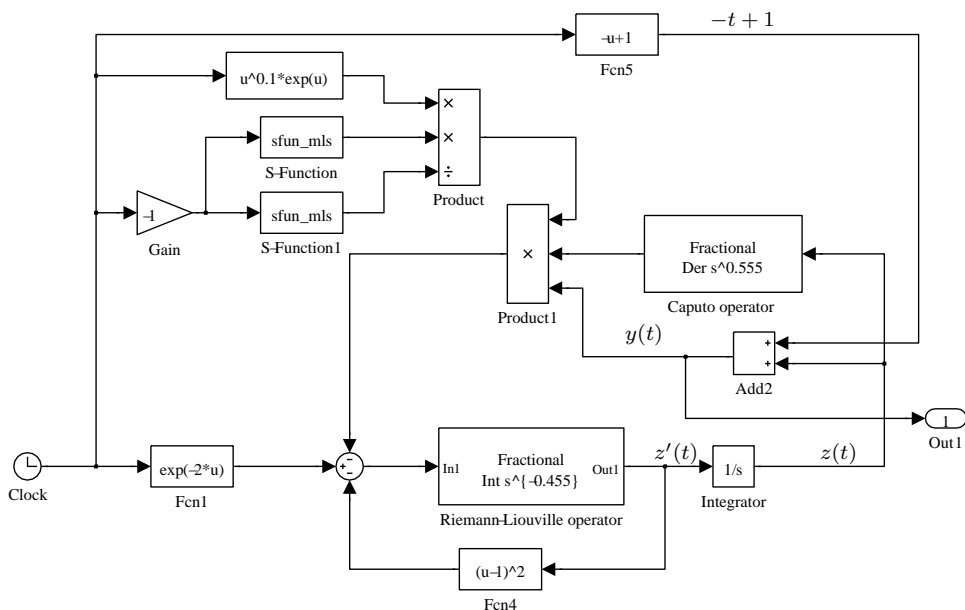


图 10-65 非线性分数阶微分方程的 Simulink 描述 (文件名: c10mcaputo)

对该模型进行仿真即可求解原始的非线性 Caputo 微分方程, 得出的结果如图 10-66 所示, 可见, 结果很接近理论值 e^{-t} 。

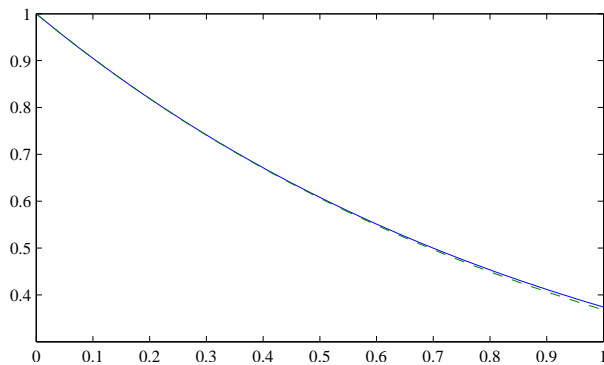


图 10-66 非线性分数阶微分方程的结果与理论值比较

10.6.6 分数阶传递函数模型与分析

考虑线性分数阶微分方程模型式 (10-6-21), 如果输入信号 $u(t)$ 和输出信号 $y(t)$ 的各阶导数初值均为零, 由 Laplace 变换的性质, 再引入 T 秒的时间延迟, 则可以建立起如下的分数阶传递函数模型

$$G(s) = \frac{b_1 s^{\gamma_1} + b_2 s^{\gamma_2} + \cdots + b_m s^{\gamma_m}}{a_1 s^{\eta_1} + a_2 s^{\eta_2} + \cdots + a_{n-1} s^{\eta_{n-1}} + a_n s^{\eta_n}} e^{-Ts} \quad (10-6-35)$$

和我们熟知的整数阶传递函数相比, 这里的分数阶模型除了分子和分母多项式的

系数外,还需要已知各项的阶次,这样可以用 4 个向量和一个延迟标量来唯一地描述式(10-6-35)中的分数阶传递函数模型。可以考虑在 MATLAB 下建立一个 FOTF 类(即分数阶传递函数)来专门描述这样的模型。再仿照控制系统工具箱中系统的建模与分析方法,为 FOTF 对象编写重载函数,使得分数阶系统的建模与分析像整数阶系统那样简单、直观。

1. FOTF —— 分数阶传递函数类的创建

若想建立一个 MATLAB 的类,则需给它起个名字,如分数阶传递函数的类名可以取作 FOTF,这样我们可以建立一个文件夹 @fotf。为使得该类能正常运行,文件夹中至少需要两个文件: `fotf.m` 文件用于定义这个类,而 `display.m` 文件用于显示该类的内容,下面具体介绍这两个必要文件。

(1) **FOTF 类定义文件编写**。应该编写 `fotf.m` 文件来定义分数阶传递函数类,其中支持各种调用方法和转换方法

```
function G=fotf(a,na,b,nb,T)
if nargin==0,
    G.a=[]; G.na=[]; G.b=[]; G.nb=[]; G.ioDelay=0; G=class(G,'fotf');
elseif isa(a,'fotf'), G=a;
elseif nargin==1 & isa(a,'double'), G=fotf(1,0,a,0,0);
elseif isa(a,'tf') | isa(a,'ss'),
    [n,d]=tfdata(tf(a),'v'); nn=length(n)-1:-1:0;
    nd=length(d)-1:-1:0; G=fotf(d,nd,n,nn,a.ioDelay);
elseif nargin==1 & a=='s', G=fotf(1,0,1,1,0);
else, ii=find(abs(a)<eps); a(ii)=[]; na(ii)=[];
    ii=find(abs(b)<eps); b(ii)=[]; nb(ii)=[];
    if nargin==5, G.ioDelay=T; else, G.ioDelay=0; end
    G.a=a; G.na=na; G.b=b; G.nb=nb; G=class(G,'fotf');
end
```

这样,在 MATLAB 命令窗口中给出命令 `G = fotf(a, na, b, nb, T)` 就可以建立一个分数阶传递函数对象,其中, $\mathbf{a}=[a_1, a_2, \dots, a_n]$, $\mathbf{b}=[b_1, b_2, \dots, b_m]$, $\mathbf{n}_a=[\beta_1, \beta_2, \dots, \beta_n]$, $\mathbf{n}_b=[\gamma_1, \gamma_2, \dots, \gamma_m]$ 分别表示分子、分母多项式的系数与阶次向量, T 为延迟常数。若系统不含延迟则可以省去该变元。类似于整数阶传递函数的定义,还可以用 `s = fotf('s')` 命令来定义一个分数阶式算子 s 。依照本函数定义,还可以用 `G = fotf(k)` 命令将常数转换成 FOTF 对象。如果 G 是控制系统工具箱中的 LTI 对象,则用 `G = fotf(G)` 可以将其转换成 FOTF 对象。

(2) **对象显示函数的编写**。在此文件夹下必须编写的另一个函数是 `display.m`,其清单如下,该函数在一个 FOTF 对象建立后用于自动显示该对象的内容,其中 `simple()` 函数为模型化简的重载函数,后面将介绍。

```
function display(G)
G=simple(G); strN=polydisp(G.b,G.nb); strD=polydisp(G.a,G.na);
nn=length(strN); nd=length(strD); nm=max([nn,nd]);
disp([char(' '*ones(1,floor((nm-nn)/2))) strN], ss=[]);
```

```

T=G.ioDelay; if T>0, ss=[' exp(-' num2str(T) 's)']; end
disp([char('-'*ones(1,nm)), ss]);
disp([char(' '*ones(1,floor((nm-nd)/2))) strD])
function strP=polydisp(p,np)
if length(np)==0, p=0; np=0; end, P=''; [np,ii]=sort(np,'descend'); p=p(ii);
for i=1:length(p), P=[P,'+',num2str(p(i)),'s^',num2str(np(i)),'']; end
P=P(2:end); P=strrep(P,'s^0',''); P=strrep(P,'+-','-' );
P=strrep(P,'^1',''); P=strrep(P,'+1s','+s'); strP=strrep(P,'-1s',' -s');
nP=length(strP); if nP>=2 & strP(1:2)=='1s', strP=strP(2:end); end

```

例 10-54 试将 FOTF 模型 $G(s) = \frac{0.8s^{1.2} + 2}{1.1s^{1.8} + 1.9s^{0.5} + 0.4} e^{-0.5s}$ 输入到 MATLAB 工作空间。

解 可以由下面的 MATLAB 语句直接输入,这样就可以在 MATLAB 下建立一个分数阶传递函数对象 G , 显示从略。

```
>> G=fotf([1.1,1.9,0.4],[1.8,0.5,0],[0.8,2],[1.2,0],0.5)
```

值得指出的是,一定要将这些文件放置于指定的文件夹中,不要随便放置,更不要直接放置到工作文件夹中,否则可能影响其他的同名函数。

2. FOTF 对象的连接

整数阶模块可以通过加法、乘法和 `feedback()` 函数定义 LTI 模型的并联、串联和反馈系统模型。仿照这样的思想,可以编写出下面的重载函数。这些文件依旧放置到 `@fotf` 文件夹中。这里的大部分函数取自文献 [27],但从内容和适用范围上做了很大完善与扩充。

(1) **FOTF 对象的乘法函数**, $G = G_1 * G_2$, 用于计算两个 FOTF 模块 $G_1(s)$ 、 $G_2(s)$ 的串联,具体的算法为

$$G(s) = G_1(s)G_2(s) = \frac{N_1(s)N_2(s)}{D_1(s)D_2(s)} \quad (10-6-36)$$

```

function G=mtimes(G1,G2)
G1=fotf(G1); G2=fotf(G2); na=[]; nb=[]; a=kron(G1.a,G2.a); b=kron(G1.b,G2.b);
for i=1:length(G1.na), na=[na,G1.na(i)+G2.na]; end
for i=1:length(G1.nb), nb=[nb,G1.nb(i)+G2.nb]; end
G=simple(fotf(a,na,b,nb,G1.ioDelay+G2.ioDelay));

```

(2) **加法函数**, $G = G_1 + G_2$, 计算模块的并联

$$G(s) = G_1(s) + G_2(s) = \frac{N_1(s)D_2(s) + N_2(s)D_1(s)}{D_1(s)D_2(s)} \quad (10-6-37)$$

```

function G=plus(G1,G2)
G1=fotf(G1); G2=fotf(G2); na=[]; nb=[];
if G1.ioDelay==G2.ioDelay
    a=kron(G1.a,G2.a); b=[kron(G1.a,G2.b),kron(G1.b,G2.a)];
    for i=1:length(G1.a), na=[na G1.na(i)+G2.na]; nb=[nb, G1.na(i)+G2.nb]; end
    for i=1:length(G1.b), nb=[nb G1.nb(i)+G2.na]; end
    G=simple(fotf(a,na,b,nb,G1.ioDelay));
else, error('cannot handle different delays'); end

```

(3) 负反馈函数, $G = \text{feedback}(G_1, G_2)$, 得出两个 FOTF 模块的负反馈反馈连接总模型, 如果为正反馈结构, 则 G_2 由 $-G_2$ 取代

$$G(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)} = \frac{N_1(s)D_2(s)}{D_1(s)D_2(s) + N_1(s)N_2(s)} \quad (10-6-38)$$

```
function G=feedback(F,H)
F=fotf(F); H=fotf(H); na=[]; nb=[];
if F.ioDelay==H.ioDelay
    b=kron(F.b,H.a); a=[kron(F.b,H.b), kron(F.a,H.a)];
    for i=1:length(F.b), nb=[nb F.nb(i)+H.nb]; na=[na,F.nb(i)+H.nb]; end
    for i=1:length(F.a), na=[na F.na(i)+H.na]; end
    G=simple(fotf(a,na,b,nb,F.ioDelay));
else, error('cannot handle different delays'); end
```

(4) 简单支持函数, $\text{uminus}()$ 函数求 $G_1(s) = -G(s)$, 允许使用 $G_1 = -G$ 命令, 也可以用 $G = \text{inv}(G_1)$ 函数求 $G(s) = 1/G_1(s)$, $\text{minus}()$ 函数求 $G(s) = G_1(s) - G_2(s)$, 允许使用 $G = G_1 - G_2$ 命令. 还可以用 $\text{eq}()$ 函数判定两个分数阶传递函数 G_1, G_2 是否相等, 允许使用 $\text{key} = G_1 == G_2$, 若相等则返回 key 为 1。

```
function G=uminus(G1), G=G1; G.b=-G.b;
function G=inv(G1), G=fotf(G1.b,G1.nb,G1.a,G1.na,-G1.ioDelay);
function G=minus(G1,G2), G=G1+(-G2);
function key=eq(G1,G2), G=G1-G2; key=(length(G.nb)==0|norm(G.b)<1e-10);
```

(5) 右除函数, $G = G_1/G_2$ 可以求出 $G(s) = G_1(s)/G_2(s)$

```
function G=mrdivide(G1,G2)
G1=fotf(G1); G2=fotf(G2); G=G1*inv(G2); G.ioDelay=G1.ioDelay-G2.ioDelay;
if G.ioDelay<0, warning('block with positive delay'); end
```

(6) 幂函数, $G = G_1 \sim n$, 如果 G_1 为分数阶传递函数则要求 n 为整数, 否则, 此函数只能处理 G_1 为 Laplace 算子乘方的情形。

```
function G1=mpower(G,n)
if n==fix(n),
    if n>=0, G1=1; for i=1:n, G1=G1*G; end
    else, G1=inv(G^(-n)); end, G1.ioDelay=n*G.ioDelay;
elseif G==fotf(1,0,1,1), G1=fotf(1,0,1,n);
else, error('mpower: power must be an integer.');
```

(7) 化简函数, $G = \text{simple}(G)$, 主要用于合并分子与分母中的同类项, 不能消除相同位置的零极点, 其子函数 $\text{polyuniq}()$ 用于合并分数阶多项式的同类项, 是 $\text{simple}()$ 函数的底层内部函数, 不能直接调用。

```
function G=simple(G1)
[a,n]=polyuniq(G1.a,G1.na); G1.a=a; G1.na=n; na=G1.na;
```

```

[a,n]=polyuniq(G1.b,G1.nb); G1.b=a; G1.nb=n; nb=G1.nb;
if length(nb)==0, nb=0; G1.nb=0; G1.b=0; end
nn=min(na(end),nb(end)); nb=nb-nn; na=na-nn;
G=fotf(G1.a,na,G1.b,nb,G1.ioDelay);
function [a,an]=polyuniq(a,an)
[an,ii]=sort(an,'descend'); a=a(ii); ax=diff(an); key=1;
for i=1:length(ax)
    if ax(i)==0, a(key)=a(key)+a(key+1); a(key+1)=[]; an(key+1)=[];
    else, key=key+1; end
end

```

例 10-55 试将分数阶 PID 控制器 $G_c(s) = 5 + 2s^{-0.2} + 3s^{0.6}$ 模型输入到 MATLAB 工作空间。

解 先定义一个 FOTF 算子 s , 则可以直接输入模型

```
>> s=fotf('s'); Gc=5+2*s^(-0.2)+3*s^0.6
```

例 10-56 试将下面的复杂的分数阶传递函数模型输入到 MATLAB 工作空间。

$$G(s) = \frac{(s^{0.3} + 3)^2}{(s^{0.2} + 2)(s^{0.4} + 4)(s^{0.4} + 3)}$$

解 对复杂模型来说只能先定义 FOTF 算子 s , 然后用简单的语句输入该模型, 得出一般的 FOTF 表示

$$G(s) = \frac{s^{0.6} + 6s^{0.3} + 9}{s + 2s^{0.8} + 7s^{0.6} + 14s^{0.4} + 12s^{0.2} + 24}$$

```
>> s=fotf('s'); G=(s^0.3+3)^2/(s^0.2+2)/(s^0.4+4)/(s^0.4+3)
```

例 10-57 假设典型单位负反馈控制系统的模型为

$$G(s) = \frac{0.8s^{1.2} + 2}{1.1s^{1.8} + 0.8s^{1.3} + 1.9s^{0.5} + 0.4}, \quad G_c(s) = \frac{1.2s^{0.72} + 1.5s^{0.33}}{3s^{0.8}}$$

试求出闭环系统的分数阶传递函数模型。

解 先将反馈系统的两个组成部分输入到 MATLAB 环境, 再给出下面语句连接系统

```
>> G=fotf([1.1,0.8 1.9 0.4],[1.8 1.3 0.5 0],[0.8 2],[1.2 0]);
Gc=fotf([3],[0.8],[1.2 1.5],[0.72 0.33]); G0=feedback(G*Gc,1)
```

这样可以得出系统的闭环模型为

$$G_0(s) = \frac{0.96s^{1.59} + 1.2s^{1.2} + 2.4s^{0.39} + 3}{3.3s^{2.27} + 2.4s^{1.77} + 0.96s^{1.59} + 1.2s^{1.2} + 5.7s^{0.97} + 1.2s^{0.47} + 2.4s^{0.39} + 3}$$

利用这里建立的 FOTF 对象还可以进一步编写出线性分数阶系统的分析与设计函数, 函数名和调用格式在表 10-15 中列出, 详见文献 [28]。

10.7 习 题

1. 考虑一个餐馆小费付费问题^[4]。假设平均小费为 15% 消费, 试根据服务水平 (例如, 可以分为

表 10-15 线性分数阶系统的分析函数

函数调用	函数功能与参数解释
$y = \text{step}(G, t)$	计算分数阶系统 G 的阶跃响应; 如果不返回变量则自动绘制阶跃响应曲线
$y = \text{lsim}(G, u, t)$	计算分数阶系统 G 的任意输入 u 的响应; 如果不返回变量则自动绘制时域响应曲线。
$\text{key} = \text{isstable}(G)$	判定分数阶系统的稳定性
$H = \text{bode}(G, w)$	Bode 频域响应数据的计算, 若不返回参数则直接绘制 Bode 图, 类似函数还有 <code>nyquist()</code> 、 <code>nichols()</code> 等, 调用格式均完全一致。
$n = \text{norm}(G, k)$	计算分数阶传递函数 G 的范数, 默认的 k 为 2, 计算 \mathcal{H}_2 范数, 若 $k = \text{inf}$, 计算 \mathcal{H}_∞ 范数

好、中、差或更详细的分段)和食物质量(也可以根据实际情况分成若干段)建立起小费确定的模糊推理系统。

2. 已知如下表的样本点 (x_i, y_i) 数据, 试利用神经网络理论在 $x \in (1, 10)$ 求解绘制出样本对应的函数曲线。还可以尝试不同的神经网络结构和训练算法, 将基于神经网络的曲线拟合结果和前面介绍的分段三次多项式插值的算法进行比较。

x_i	1	2	3	4	5	6	7	8	9	10
y_i	244.0	221.0	208.0	208.0	211.5	216.0	219.0	221.0	221.5	220.0

3. 假设已知实测数据由下表给出, 试利用神经网络对 (x, y) 在 $(0.1, 0.1) \sim (1.1, 1.1)$ 区域内的点进行插值, 并用三维曲面的方式绘制出基于神经网络的插值结果。

y_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
0.1	0.83041	0.82727	0.82406	0.82098	0.81824	0.8161	0.81481	0.81463	0.81579	0.81853	0.82304
0.2	0.83172	0.83249	0.83584	0.84201	0.85125	0.86376	0.87975	0.89935	0.92263	0.94959	0.9801
0.3	0.83587	0.84345	0.85631	0.87466	0.89867	0.9284	0.96377	1.0045	1.0502	1.1	1.1529
0.4	0.84286	0.86013	0.88537	0.91865	0.95985	1.0086	1.0642	1.1253	1.1904	1.257	1.3222
0.5	0.85268	0.88251	0.92286	0.97346	1.0336	1.1019	1.1764	1.254	1.3308	1.4017	1.4605
0.6	0.86532	0.91049	0.96847	1.0383	1.118	1.2046	1.2937	1.3793	1.4539	1.5086	1.5335
0.7	0.88078	0.94396	1.0217	1.1118	1.2102	1.311	1.4063	1.4859	1.5377	1.5484	1.5052
0.8	0.89904	0.98276	1.082	1.1922	1.3061	1.4138	1.5021	1.5555	1.5573	1.4915	1.346
0.9	0.92006	1.0266	1.1482	1.2768	1.4005	1.5034	1.5661	1.5678	1.4889	1.3156	1.0454
1	0.94381	1.0752	1.2191	1.3624	1.4866	1.5684	1.5821	1.5032	1.315	1.0155	0.62477
1.1	0.97023	1.1279	1.2929	1.4448	1.5564	1.5964	1.5341	1.3473	1.0321	0.61268	0.14763

4. 假设通过实验测出一系列数据, 可以列出一个 60×13 的表格, 由文件 `c10rsdat.txt` 给出, 其中每行为一个样本, 前 12 列每列对应一个条件, 最后一列表示某事件是否发生的标志, 试用粗糙集的方法找出前 12 个条件中哪些条件对事件的发生起着重要的作用。
5. De Jong 最优化问题^[10]是一个富有挑战性的最优化基准测试问题, 其目标函数为

$$J = \min_{\mathbf{x}} \mathbf{x}^T \mathbf{x} = \min_{\mathbf{x}} (x_1^2 + x_2^2 + \cdots + x_{20}^2)$$

若 $-512 \leq x_i \leq 512, i = 1, 2, \dots, 20$, 试用遗传算法得出其最优化问题的解, 并用普通的无约束最优化算法函数 `fminunc()` 求解同样的问题, 比较两种方法所需的时间和精度。显然, 该问题的全局最优解为 $x_1 = x_2 = \cdots = x_{20} = 0$ 。

6. 假设由下面的语句可以生成噪声污染的信号:

```
>> t=0:0.005:5; y=15*exp(-t).*sin(2*t);
r=0.3*randn(size(y)); y1=y+r;
```

试用小波分解与小波重建方法对该信号进行滤波处理,并和第8章习题中的结果进行比较。

7. 试利用遗传算法求解下面的有约束最优化问题,并和传统数值方法进行比较。

$$\begin{aligned} \min \quad & \frac{1}{2 \cos x_6} \left[x_1 x_2 (1 + x_5) + x_3 x_4 \left(1 + \frac{31.5}{x_5} \right) \right] \\ \text{s.t.} \quad & \begin{cases} 0.003079 x_1^3 x_2^3 x_5 - \cos^3 x_6 \geq 0 \\ 0.1017 x_3^3 x_4^3 - x_5^2 \cos^3 x_6 \geq 0 \\ 0.09939(1+x_5)x_1^3 x_2^2 - \cos^2 x_6 \geq 0 \\ 0.1076(31.5+x_5)x_3^3 x_4^2 - x_5^2 \cos^2 x_6 \geq 0 \\ x_3 x_4 (x_5 + 31.5) - x_5 [2(x_1 + 5) \cos x_6 + x_1 x_2 x_5] \geq 0 \\ 0.2 \leq x_1 \leq 0.5, 14 \leq x_2 \leq 22, 0.35 \leq x_3 \leq 0.6 \\ 16 \leq x_4 \leq 22, 5.8 \leq x_5 \leq 6.5, 0.14 \leq x_6 \leq 0.2618 \end{cases} \end{aligned}$$

8. 给定信号 $f(t) = e^{-3t} \sin(t + \pi/3) + t^2 + 3t + 2$, 试利用定义计算出该函数的 0.2 阶微分信号及 0.7 阶积分信号, 并将结果信号用曲线表示出来。
9. 分别为习题 8 给出的信号设计出连续滤波器, 并对该信号进行分数阶微积分运算, 和前面介绍的较精确的数值结果相比较, 研究所用方法的精度。
10. 假设已知分数阶线性微分方程为^[20]

$$0.8 \mathcal{D}_t^{2.2} y(t) + 0.5 \mathcal{D}_t^{0.9} y(t) + y(t) = 1, y(0) = y'(0) = y''(0) = 0$$

试求该微分方程的数值解。若将微分阶次 2.2 近似成 2, 0.9 阶近似成 1 阶, 则可以将该微分方程近似为整数阶微分方程, 试比较整数阶近似的计算精度。

11. 试求解下面的零初值分数阶非线性微分方程, 其中 $f(t) = 2t + 2t^{1.545}/\Gamma(2.545)$ 。如果该方程是 Caputo 微分方程, 且已知 $y(0) = -1, y'(0) = 1$, 试重新求解该方程。

$$\mathcal{D}^2 x(t) + \mathcal{D}^{1.455} x(t) + \left[\mathcal{D}^{0.555} x(t) \right]^2 + x^3(t) = f(t)$$

12. 设分数阶非线性微分方程由图 10-67 中的 Simulink 模型描述, 试写出该微分方程的数学表达式, 并绘制出输出信号 $y(t)$ 。
13. 试求解 Bagley-Torvik 方程^[26] $Ay''(t) + B\mathcal{D}^{3/2}y(t) + Cy(t) = C(t+1), y(0) = y'(0) = 1$, 并验证该方程的解与常数 A, B, C 的取值无关。

参考文献

- [1] Weisstein E W. Goldbach conjecture. From MathWorld — A Wolfram Web Resource. <http://mathworld.wolfram.com/GoldbachConjecture.html>
- [2] Zadeh L A. Fuzzy sets. Information and Control, 1965, 8:338–353
- [3] 汪培庄. 模糊集合论及其应用. 上海: 上海科学技术出版社, 1983

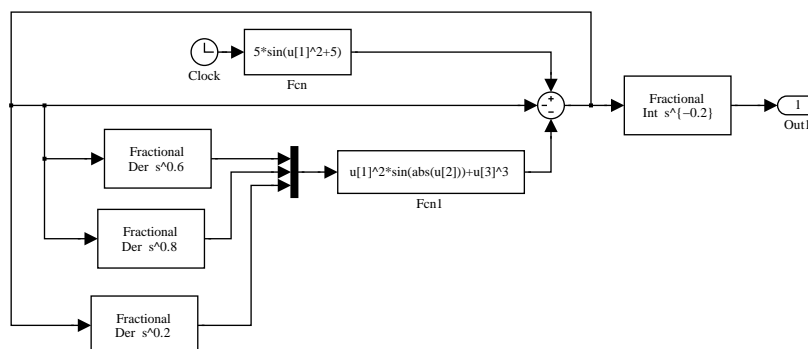


图 10-67 非线性分数阶微分方程的 Simulink 描述 (文件名: c10mfode4.mdl)

- [4] The MathWorks Inc. Fuzzy logic toolbox user's manual, 2007
- [5] Pawlak Z. Rough sets — theoretical aspects of reasoning about data. Boston, USA: Kluwer Academic Pub., 1991
- [6] 张雪峰. 粗糙集数据分析系统应用平台的研究与程序开发. 沈阳: 东北大学硕士论文, 2004
- [7] Hagan M T, Demuth H B, Beale M H. Neural network design. PWS Publishing Company, 1995 (戴葵等译. 神经网络设计. 北京: 机械工业出版社, 2002)
- [8] 王旭, 王宏, 王文辉. 人工神经网络原理与应用. 沈阳: 东北大学出版社, 2000
- [9] 邵军力, 张景, 魏长华. 人工智能基础. 北京: 电子工业出版社, 2000
- [10] Chipperfield A, Fleming P. Genetic algorithm toolbox user's guide. Department of Automatic Control and Systems Engineering, University of Sheffield, 1994
- [11] Houck C R, Joines J A, Kay M G. A genetic algorithm for function optimization: a MATLAB implementation, 1995
- [12] 王小平, 曹立明. 遗传算法 — 理论、应用与软件实现. 西安: 西安交通大学出版社, 1998
- [13] Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks. Perth, Australia, 1995, 1942–1948
- [14] Birge B. PSOt, a particle swarm optimization toolbox for MATLAB. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. Indianapolis, 2003, 182–186
- [15] Birge B. Particle swarm optimization toolbox. MATLAB Central File ID: #7508, 2003
- [16] Trelea I C. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters, 2003, 85(6):317–325
- [17] Clerc M, Kennedy J. The particle swarm: explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 2002, 6(1):58–73
- [18] Vinagre B M, Chen Y Q. Fractional calculus applications in automatic control and robotics. 41st IEEE CDC, Tutorial workshop 2, Las Vegas. 2002
- [19] Hilfer R. Applications of fractional calculus in physics. Singapore: World Scientific, 2000
- [20] Podlubny I. Fractional differential equations. San Diego: Academic Press, 1999

-
- [21] Petráš I, Podlubny I, O'Leary P. Analogue realization of fractional order controllers. TU Košice: Fakulta BERG, 2002
 - [22] Oustaloup A, Levron F, Nanot F, et al. Frequency band complex non integer differentiator: characterization and synthesis. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2000, 47(1):25–40
 - [23] Xue D, Zhao C N, Chen Y Q. A modified approximation method of fractional order system. Proceedings of IEEE Conference on Mechatronics and Automation. Luoyang, China, 2006, 1043–1048
 - [24] Podlubny I. Fractional-order systems and $PI^{\lambda}D^{\mu}$ -controllers. IEEE Transactions on Automatic Control, 1999, 44(1):208–214
 - [25] 薛定宇, 陈阳泉. 基于MATLAB/Simulink的系统仿真技术与应用. 北京:清华大学出版社, 2002
 - [26] Cachan J, Groningen F, Paris B. The Analysis of Fractional Differential Equations. New York: Springer, 2010
 - [27] Monje C A, Chen Y Q, Vinagre B M, Xue D, Feliu V. Fractional-order systems and controls — fundamentals and applications. London: Springer, 2010
 - [28] 薛定宇. 控制系统计算机辅助设计——MATLAB 语言与语言 (第 3 版). 北京:清华大学出版社, 1998

MATLAB 函数名索引

本书涉及大量的 MATLAB 函数与作者编写的 MATLAB 程序、模型,为方便查阅与参考,这里给出重要的 MATLAB 函数调用语句的索引,其中黑体字页码表示函数定义和调用格式页。标注为 * 的为作者编写的函数,标注为 † 的为作者编写的重载函数。

A

abs 29 33 56 71 81 135 **163**
acos 135
addrule **387**
addvar **386**
all 20 122 **381**
angle **163**
anova1 **370**
anova2 372
any 20 29 122 336
any_matrix* **103** 104 110
apolloeq* 251~253
appcoef **426**
are **134** 187
asin 135
assignin 188
assumptions **15**
atan 55 135
atan2 171
atanh 73
axes 303
axis 48 81 247 248 307 422 432

B

balreal 267
bar **35** 268 295 296 354
besselh/besselj/besselk **332**
beta **331**
binopdf 353
bintprog **210** 211
binvar 218
biograph **228** 229
BNB20_new* **207 208** 209~212

bode †451
break 24 25 121 188 204 303 324
butter **341** 342
bvp5c **278** 279
bvpinit 278 279

C

c10mcaputo.mdl* 446
c10mga1/3/4* 411 412 415
c10mpso1/4* 417 418
c2d 267
c3ffun* 84
c6exinl* 209
c6exmcon* **203**
c6exnls* 204
c6fun3* 196
c6mdisp* 210 420
c6mmibp* 212
c7impode* **261**
c7mdde2/3/6.mdl* 293~295
c7mlor1a.mdl*/c7mlor1b.mdl* 292 293
c7mnlrsys.mdl* 296
c7mrand.mdl* 295
caputo* **433** 434
cdf **346** 347~351
ceil **22** 208 433 441
char 149 238 239 444 447
charpoly **108** 110
cheby1/cheby2 **341**
chi2pdf/chi2cdf/chi2inv **350**
chi2rnd **354**
chol/cholsym* **119 120**
clabel 44

class 447
 collect 20 58 69 108 140 148
 comet **35**
 comet3 247
 compan/companysym* **101** 102
 compass **35**
 cond **125**
 conj 135 **163**
 contfrac* **325** 326 327
 contour 41 **43** 44 60 81 193 310 313
 contour3 **42** **44** 45 195
 contourf **44** 45
 conv 30 340 427
 convs* 30
 core* **393**
 corr 374
 corrcoef **338**
 cos 135 140 148 154~156 172
 cov **357**
 cplxgrid **164** 165
 cplxmap/cplxmap1* **164** **165**
 cplxroot **164**
 csapi **313** **314** 315~318
 cumsum 150
 cwt **421**

D

dblquad 88
 dde23 **268** 269 270
 ddensd **272** 273
 ddesd **270** 271 272
 dec2mat 216 217
 decic **262** 264
 default_vals* 70 88 103 188 325 336
 delete 43 303
 det **104** 135
 detcoef **426**
 diag **99** 102 139 140 201 267 273 294
 diagm* **99** 136 138
 diff **57** 58~61 239 259 316 318 450
 diff_ctr* 79 **80**
 diff_eq* **174**
 dijkstra* **230** 231
 disp 157 161 447
 display †447
 dlyap 131
 doc 10 12 60

double 14 **102** 106 131 170 218 219 434
 dsolve **238** 239~242 275
 dwt **422** 423

E

eig **114** **115** 121 122 163 374
 elseif **24** 28 29 102 108 171 447 449
 eps 14 29 71 72 85 124 171 188
 eq †449
 error 56 59 61 70 102 108 132 230 311
 errorbar **35** 363 365
 eval 103 150 184 185 213
 evalfis **388** 389
 exp 135 137 140 147 148 150 151
 expand 20 21
 expm **136** 137 138 241 257
 eye **98** 108~112 116 176
 ezcontour/ezcontourf 41 42
 ezimplot3 45
 ezmesh 41
 ezplot **36** 54 58 67 68 70 73 163
 ezplot3 45
 ezsurf 41 42 318

F

factor 20 **22**
 factorial 29 55 170
 feasp **216** 217
 feather **35**
 feedback †449 †450
 feedforwardnet **398**
 feval **55** 56 168 169 244 325
 fft/fft2/fftn 156
 fgoalattain **226**
 figure 45 90 164 265 308
 fill **35** 75 85
 fill3 40
 filter **340** 341 342
 find 20 75 90 139 163 170 197 206
 findsum* 27 28
 fitnet 399 401~403
 fix **22** 449
 fliplr **18** 121 150 375
 flipud **18**
 floor **22** 29 121 135 156 448
 fmincon **202** 203~205 220 410 411 419
 fminimax **225** 226

fminsearch **192** 194 414
 fminsearchbnd **196** 197
 fminunc **192** **193** 195 196 313
 fnder **316** 317
 fnint **318**
 fnplt **313** 314~318
 fnval **313** 318
 fode_caputo* **441**
 fode_sol* **439** 440
 for **23** 24 25 28~30 48
 format 72 110 168 171 364
 fotf* **447** 448~450
 fourier **152** 153
 fpdf/fcdf/finv **348**
 freqz **340** 341 342
 frnd **354**
 fseries* **70** 71
 fsolve **186** 187 188 261
 full **103** 227
 funm/funmsym* **137** 139 **140**
 fuzzy 386

G

ga **409** 414 **415** 416 **419** 420
 gamfit 361 369
 gamma 150 **330** 335~337 356 433 434 438
 441
 gammainc **331**
 gampdf/gamcdf/gaminv **350** 351
 gamrnd **354** 361 368 369
 gamstat **356**
 gaopt **409** 411 412 **413** 415
 gaoptimset 414
 gatool 409
 gaussmf **383**
 gbellmf **383**
 gcd **22** **167**
 get **33** 49 229
 get_param 444
 getedgesbynodeid 229
 getlmis **216** 217
 gevp **216**
 ginput 37 303
 glfdiff* **431** 432~436
 gradient **81**
 graphshortestpath **228** 229
 grid 32 40

griddata **308** 309 310 313
 griddata3/griddatan 311

H

hankel/hankelsym* **100** 102 108 329
 hessian **63**
 hilb 22 **100** 104~106 110 111
 hist **35** 268 295 296 354
 hold 32 40 49 54 58 60 67 68 70 73

I

icdf **346** 347~351
 idwt **423**
 if **24** 25 28 29 59 61 70 87
 ifft/ifft2/ifftn 156
 ifourier **152** 153
 ilaplace **147** 149
 imag 122 135 **163** 171
 impldiff* **61** 62
 ind* 391 392
 Inf 200 230 231 330 335 338 355
 inline 30 84
 int **64** **65** 66
 int2str 103 213
 int8/int16/int32 15
 integral **84** 85~87 150 305
 integral2 **88**
 integral3 **91**
 interp1 150 **301** 302~306 433
 interp2 **306** 307
 interp3/interp4 311 312
 intersect **380** 381 391
 intfunc* **87**
 intfunc2* **88** 89 90
 intvar 218
 inv_pendulum* **255**
 inv_z* **162**
 invhilb **100** 110 111
 INVLPAP **149**
 inv **110** 127 133 137 140 260 362 †449
 isa 447
 isfinite 108 336
 ismember **380** **381** 391
 isnumeric 150 275 325 442 445
 isprime **22** 381
 isstable †451
 iztrans **160** 174 176

J
jacobian **62** 63 196 205
jbtest 368
jordan **123** 137 139

K
kron **130** 131 132 448 449
kstest **369**

L
lagrange* **303**
laplace **147** 148
lasterr **14** 26
lastwarn **14**
latex 21 58 325
lcm **22**
legendre **333**
length 30 391 423 431 439 441 448~450
lillietest **368**
limit **54** 55 57 73 166
line 182 303 310 329 347 354 432
linineq* 134 135
linprog 3 **198** 199 200 221 222 224 415
linprog.c* **222** 223
linspace 83 87 88 150 **278** 279 296
lmiterm **216** 217
lmivar **216** 217
ln_shooting* **275** 276
load **38** 324 325
log 73 135 157 212 322
log10 135 336
loglog **35**
logm **136**
lorenz1* **248**
lsim †451
lsqcurvefit **323** 324 325 364
lsqlin **223**
lsqnonlin **220**
lu/lusym* **117** **118**
lyap/lyapsym* **129** **132** 133
lyap2lmi* **213** **214**

M
maple **154** **158** 159
max 188 197 259 302 392 438 447
mean 267 **355** 356 370 372
mesh **41**
mesh2nd* 311 315

meshgrid **41** 42~44 46~49 193 195 197
mfedit 386 387
min 102 230 252 259 422 450
mincx **216**
minus †449
mittag_leffler* **335** 336
ml_func* **336** **337**
ml_step* **438**
MLF 336
moment **357**
more_sols* **188** 189 190
mpower †449
mrdivide †449
mtimes †448
mvnpdf **358**
mvnrnd 359
myhilb*/my_fact*/my_fibo* 28 29

N
NaN **14** 90 197
nargin 27~29 87 102 132 170 433 436
438
nargout 28 70 325
ndgrid 206 207 **311** 315 317
new_fod* 436
newfis **385** 386
newrbe 404
nlbound* 277
nlinfit **364** 365
nlparci **364** 365
nntool **404** 405
norm 19 80 **106** 107 109~111 †451
normfit **360** 368
normpdf/normcdf/norminv **347** 367
normrnd 367
null **127** 128
num_laplace* **150** 151
num2str 442 445 448
numden 20 62 170

O
ode15i **262** 264
ode15s 3 257 260 261 263
ode45 **246** 247~252 255 256 258 259
odeset **247** 252 255 257~260 263 264 270
ones **98** 102 103 208 230 321 322
open_system **290**

opt_con1*/opt_con2* 202 203
 opt_fun2* 205
 optimset **186** 187 188 193 195 196 199
 optimtool 409
 orth **116** 117

P

padefcn*/padefcnsym* **329** 330
 paradiff* **59**
 paretofront **224**
 partfrac* **168** 169
 partfrac1* 170
 patternnet 398
 patternsearch 418
 pause 48
 pcode **31**
 pdebc **280**
 pdefun **280**
 pdepe **281**
 pdetool **282**
 pdf **346** 347~351
 pfrac* **171**
 pi **14** 15 17 32~34 40 55
 piecewise* **55** **56** 86 155
 pinv **112** 113 127 128
 plot **31** 32~34 54 80 163 264
 plot3 **40** 247 248 293 308 374
 plotperf **399**
 plotyy 34 434
 plus †448
 poisspdf/poisscdf/poissinv **346** 347
 polar 34 **35**
 poly **107** 108 109
 poly1* **108** 109
 poly2caputo* **441** 444
 poly2sym 2 **110** 174 329 441
 polycoef* **108**
 polyfit **319** 320
 polyval **109** 319 320 329 441
 polyvalm/polyvalmsym* **109** 110
 prod 29 165 166 170 172 213 303
 pso-Trelea_vectorized **416** 417 418

Q

quad 84
 quad2dggen **89** 90
 quadgk 84 86

quadl 84 86 305
 quadndg **91** 92
 quadprog **201**
 quadspln* **305**
 quadv 84 150
 quiver **35** 60 81

R

rand **98** 121 170 226 279 308 309 359
 randn **98** 267 339 340 344 357 363
 random **354** 355~357
 rank **105** 106 113 121 124 127 128
 rat **22** 168
 raylcdf/raylinv/raylpdf **351** 352
 raylrnd **354**
 raylstat 356
 readfis 389
 real 122 135 150 156 **163** 171 375
 realjordan* 122 124
 redu* **393** 394
 regress **362** 363
 rem **22** 175
 reshape 130~132 256 336 358 389
 residue **167** 168 171
 ric_de* **256**
 rk4* **244** 253 258
 rng **354**
 roots 2 114
 rot90 **18** 101 329
 rotate **47** 48 49
 round **22** 336 363 365
 rref **112** 128
 rsdsv3* **394**
 rslower*/rsupper* **391**
 ruleedit 387

S

save **38**
 sc2d* **267**
 sdpsettings 218
 sdpvar **217** 218 219
 semilogx/semilogy **35** 401
 set **33** 186 193 218 219 229 303
 set_param 444
 setdiff **380** 381 391 392
 setlmis **216** 217
 setxor **380**

-
- sfun_mls* 445 446
 shading 43 197 422
 sigmf **384**
 sign 33 56
 sim 293 294 **399** 401~404 406
 simple **20** 21 59~62 64 69 †447 †449
 simulannealbnd **418** 419
 sin 32 34~36 **135** 241
 sinml* **137**
 size 98 99 102 108~111 118
 sketcher* **303**
 slice **48** 49
 solve 2 170 **183** **184** 185 187 191 254
 solvesdp **218** 219
 sort 71 72 90 170 206 207 448 450
 spapi **315** 316~318
 sparse **103** **227** 229~231
 sqrt 14 19 42 48 54 57 75 85 135 138
 sqrtm 139
 ss 267
 stairs **35** 175
 std 356
 stem **35** 156 162 174 339 347
 stem3 40
 step †451
 strrep 150 448
 subplot 35 46 80 175 423 424 440
 subs **21** 56 59 60 62 65 70 148 149 163
 sum 24 150 264 353 357 359 360 426 439
 surf **41** 43 197 281 307~310 389
 surfc/surfl 41
 svd 16 **125** 267
 switch/case **25** 26 79 311 445
 switch_sys* 265
 sym 3 22 58 72 **102** 103 106 107 111
 sym2poly **110** 329
 symprod **73** 74
 syms **15** 21 54~57 166
 symsum **72** 73 74 161 335
- \mathcal{T}*
- tan 32 33 55 277
 tansig 396
 taylor **67** 68 **69**
 tf 267
 tfdata 447
 tic/toc 24 29 58 83 86 91 92 104
- title 32
 tpdf/tcdf/tinv **349**
 trace **105** 108
 train **399** 401~403
 trapz **82** 83 305
 triplequad 91
 trnd **354**
 try/catch **26** 56 132 188
 ttest **367**
- \mathcal{U}*
- uint8/uint16/uint32 15
 uminus †449
 union **380** 381
 unique **380** 381 392
- \mathcal{V}*
- vander/vandersym* **101** **103** 109
 varargin 30 56 70 88 99 103 325 336
 varargout 30 70
 view **46** 90 229
 vol_visual4d* **49** 312
 vpa **15** 65 85~87 90
- \mathcal{W}*
- warning 449
 waterfall 41
 wavedec **425** 426 427
 wavefun **423** 424
 wavemenu **427**
 wavemngr **423**
 while **23** 24 27 121 122 137 188 303
 wrcoef **426** 427
 writefis 388
- \mathcal{X}*
- xcorr **338** 339
 xlabel 32
 xlim 46 156
 xlsread **38** 39 304
 xlswrite 38
 xor **20**
- \mathcal{Y}*
- ylabel 32
 ylim 194 330
- \mathcal{Z}*
- zeros **98** 102 108 109 118 137 177 189
 zlim 43 60 165 195
 ztest **367**
 ztrans **160** 161 174 176

术语索引

0-1 线性规划 210 211

A

Abel–Ruffini 定理 2 238

Adams 算法 243

ARMA 滤波器 340

B

B 样条 313 315~318

白噪声 265~267 295 296 339 423

半对数图 35 36

半正定矩阵 120

包含 380

饱和非线性 33 56 296 396

被积函数 63 64 75 77 92 306

Bessel 函数 158 331 332

Bessel 微分方程 297 331

β 分布 346

β -函数 330 331

闭环控制 245 450

比较运算 13 20 291

闭式解 4

边界集 390~392

边界条件 238 274 275 280 281

变精度算法 15 148

变量替换 21 63 147 170 173 192 204 209 420

边权值 227 230 231

变时间延迟方程 268 270~272 294 295

边值问题 10 237 273~280

表面图 41~43 307 358 422

标准差 339 347 355 366 367 423

标准正态分布 98 267 348 357 366

并集 283 285 380 390

并联连接 178 448

病态矩阵 125

Bode 图 435~437 451

补偿函数 441 445

不定积分 63~65 318

不定式 14 52 73 197

部分分式展开 9 145 167~172 437

不可分辨关系 391 392

不完整 Γ -函数 331

不稳定 255 265

Butterworth 滤波器 341 342

C

采样周期 156 173 177 267 295

参数方程 40 59 60 74 76~78

参数估计 345 346 360~366

Caputo 定义 429 444 445

Caputo 微分方程 441 444~447

Cauchy 积分公式 429 432 433

Cayley–Hamilton 定理 109

测度 106 124 128

侧视图 46 47

查表法 159 345

差分方程 4 9 145 173~177 340

差集 283 284 380 391

插值 42 83 150 151 243 301~319 403 404 434

Chebyshev 滤波器 341

乘方 18 449

程序调试 5 26 27

χ^2 分布 346 349 350 354

Cholesky 分解 97 119 120 267

重积分 53 63 65 66 82 88~92 152

重极限 53 56 57

重奇点 165 166 168 172

重特征值 114 115 122 123 138 139

重载 102 111 447 448
 初始搜索点 187 193 194 198 202 220 278 418
 初值问题 178 237 242~248 273~278 430 441
 Chua 电路 298
 传递函数 150 151 267 290 293 296 434
 串联连接 178 448
 传输函数 395~397 403 405
 次最优解 206 207
 Clerc 算法 416
 粗糙集 3 10 379 389~394
 存在性 4

D

打靶法 274~277
 待定系数 238 321~324 362~364
 代码保密 31
 代数环 442
 代数余子式 2 104 105 134
 代数运算 13 17~19 291
 带通滤波器 340 342
 单边极限 53~55
 单纯形法 198
 单位矩阵 97 98 108 112 118 121 216
 单因子方差分析 345 369~371
 单元数组 15 30 379 380 398
 倒立摆 254~256
 Daubechies 族小波 423 424
 等高线 41~45 195 310~312
 递归方法 2 29 59
 低通滤波器 340 342 442
 递推方法 431
 递推算法 107 175 398
 点运算 18 19 21 40 41 48 84 109 416 417
 Dijkstra 算法 227 229 230
 定步长算法 11 83 86 245 253 258 296
 定步长 Runge-Kutta 算法 11 244 245 298
 定积分 4 63~66 75 82~89 91 305 306 318
 Dirichlet 条件 285 287
 动态规划 9 181 227~231
 对称矩阵 97 100 112 114 119 120 129 130
 212 213 216 217 256 357

对角矩阵 97 99 114 121 122 125 266 359
 对数 135 136 139 291 322
 对数图 35
 对数正态分布 346
 对象 15 33 37 446~450
 多变量函数 53 56 66 68 69 316 324 325 364
 多变量正态分布 346 358 359
 多解方程 187~190
 多目标规划 9 181 219~226
 多目标线性规划 221~223
 多维数组 15 98 355
 多项式方程 2 170 181 183~186 238
 多项式拟合 301 319 320 322
 多因子方差分析 373
 多纵轴 34

E

EISPACK 4 5 7 97 114 115
 二次型 133
 二次型规划 181 201 202 414
 二分法 51
 二项分布 346 353
 二项式系数 429
 Euclid 距离 231
 Euler 常数 66 73 74 331
 Euler 公式 138
 Euler 算法 237 242 243
 Excel 文件 38 39 304

F

F 分布 345 346 348
 泛化 379 395 399 401 402 404 406
 反馈连接 449
 范数 97 106 109~116 188 218 220 451
 反向传播 396 398 405
 翻转 17 18 121 375
 方差分析 10 345 361 369~373
 仿射函数 214
 方位角 45 46
 仿真框图 295 296 443
 非对称矩阵 120 130
 非零初值 160 178 271 295 440~442

非满秩矩阵 105 106 113
 非奇异矩阵 106 111 116 121 131 132 260 264
 非线性规划 181 201~205 414 415 418 419
 非线性环节 296
 非线性回归 365 366
 非线性矩阵方程 181 187
 非线性微分方程 237 241 242 247 255 295 429
 非运算 19
 非整数阶 又见 分数阶
 分布函数 又见 概率分布函数 349~351
 分部积分法 63 330
 分段函数 33 43 44 47 55 56 74 85~87
 分块矩阵 99 214 216
 分配律 379 380
 分数阶 149 151 334 379 428
 分数阶传递函数 见 FOTF
 分数阶微分方程 430 437 440 441 443 445 446
 分数阶微积分 428~450
 分枝定界法 181 207 218
 封闭曲线积分 145 171~173
 FFT 又见 快速 Fourier 变换
 Fibonacci 数列 6 29
 FIR 滤波器 又见 有限长脉冲响应 340
 FOTF 类 446~450
 Fourier 变换 145 151~156 159 420 424 434
 Fourier 级数 53 66 69~72 431
 Frobinius 范数 107
 复变函数 3 9 162~171
 浮点运算 14
 负定矩阵 212
 附加变量 246 248
 俯视图 46 90
 复数特征值 121 122 163 170
 辅助变量 440 443 445

\mathcal{G}

改进 Oustaloup 滤波器 435 436
 概率 3 345 353 359 366 367 370~372 408
 概率分布函数 345~349
 概率密度函数 43 268 295 345~349
 Γ 分布 345 346 350 351 354~356 360 368 369

Γ -函数 52 330 331 334 429 430
 刚性方程 3 237 250 257~259 443
 高阶导数 57 59 61 148 254
 高通滤波器 340 342 344
 Gauss-Kronrod 算法 86
 Gauss-Newton 算法 364
 Gauss 白噪声 265 296
 Goldbach 猜想 381
 共轭复数 17 135 163
 共轭梯度 398 401
 共轭转置 又见 Hermite 转置 116 120
 Grünwald-Letnikov 定义 429~435 438 439
 关联矩阵 227 229~231
 惯量函数 416
 广义逆矩阵 97 112 113
 广义特征值 115 215 216
 广义 Lyapunov 方程 又见 Sylvester 方程 132
 归一化 341 373 408

\mathcal{H}

Haar 小波 423 424
 Hadamard 乘积 18
 函数逼近 301 313
 函数调用 16 27 30 35 41 59 61 67 72 82
 行列式 2 97 103~105 107 116 120 141
 Hankel 变换 145 156 158 159
 Hankel 函数 又见 Bessel 函数 332
 Hankel 矩阵 50 97 99 100 102 108 111 218
 核集 392 393
 Heaviside 函数 152 153
 Henon 引力线 51
 Hermite 转置 又见 共轭转置 17 116 120
 Hess 矩阵 63
 Hilbert 矩阵 2 22 28 97 100 104 105 110 111
 Hilbert 逆矩阵 100 111
 后向差分公式 78 79
 互相关函数 337~339
 化简 2 20 21 58 62 64 66 69 109 356
 化零空间 127
 坏条件矩阵 100 125
 回合数 398 399 405

- 混沌 298
 混合整数规划 181 205~207 211 419
 火柴杆图 35 36 40
 或运算 19 385
- \mathcal{I}
- IIR 滤波器 又见 无限长脉冲响应 340
 inline 函数 30 84 85 91
- \mathcal{J}
- Jacobi 矩阵 62 63 246 279
 Jacobi 算法 114
 Jarque-Bera 假设检验 368 369
 迹 97 105 116
 基础解系 97 127
 奇点 145 146 165 166 168 169 171 172
 积分变换 3 145~159 421
 集合 161 283 284 379~381
 激励函数 又见 传输函数
 级数求和 9 24 53 66 72~74 335
 计算步长 11 242~245 252 266 295
 计算机数学语言 1~4 6 8 9 13 53 82 97
 极限 9 53~57 73 167 170
 极限环 298
 基小波 420~427
 极小极大问题 224~226
 基准测试问题 232 451
 极坐标 34 35 164
 假频 156
 假设检验 345 366~370
 加速常数 416
 间断点 55 85
 降噪 423 424 426 427
 交集 283 380
 阶乘 29 49 330
 截断 326
 结构体 184 186 193 196 199 201 203 416 419
 解模糊 385 387~389
 阶梯图 35 36
 截止频率 442
 进化算法 407~416
 近似解 91 243
- 近似系数 424~426
 径向基函数 403 404
 径向基网络 403 404
 Jordan 变换 97 122~124
 Jordan 标准型 122
 Jordan 矩阵 122 123 136 137 139
 局部最优 181 193 194 204 416 419
 矩阵乘法 18
 矩阵分解 97 104 117
 矩阵函数 59 97 123 136~140
 矩阵三角函数 138
 矩阵微分方程 254~256
 矩阵指数 136 137 139
 卷积 146 152 160
 决策变量 181 186 199 204~211 215 217 218
 决策属性 390 392 393
 绝对误差限 84
 均匀分布 74 98 345 346 353 354 359 360 364
- \mathcal{K}
- 开关结构 13 23 25 26
 开环控制 245
 可行解 197 211 213 215~218 224 226 415
 可枚举集合 379
 Kolmogorov-Smirnov 假设检验 369
 空集 380 381 390
 控制系统工具箱 7 129 131 132 134 218 447
 Kronecker 乘积 130 132
 块对角矩阵 99
 快速 Fourier 变换 4 145 155 156 301
- \mathcal{L}
- L'Hôpital 法则 93
 Lagrange 插值 303 304 319
 Lagrange 方程 254
 Lambert W 函数 52
 LaPACK 5 6 97
 Laplace 变换 145~151 170 171 430 431 434
 Laplace 反变换 145~147 149
 L^AT_EX 21 38 58 147
 Legendre 函数 333
 类 15 447~450

累积误差 243 244
 累极限 56 57 92
 类 Riccati 方程 187 189
 Levenberg–Marquardt 训练算法 398 401
 Leverrier–Faddeev 递推算法 107
 离散点 223 224 312 338
 离散化 266 267
 离散 Fourier 变换 156
 离散 Fourier 正余弦变换 145 155
 离散 Lyapunov 方程 131~133
 隶属度函数 381~387
 粒子群优化 379 408 416~418
 连分式 301 325~327 329
 联合概率密度 352 353 358
 Lilliefors 假设检验 368 369
 林士谔–Bairstrow 算法 2
 零初值问题 160 237 442
 零矩阵 97 98 127
 LINPACK 4 5 7 97
 留数 9 145 165~167 170~172
 Lorenz 方程 247 248 292
 LTI 模型 447 448
 鲁棒控制工具箱 212 216 217
 滤波器 301 339~342 426 434~436 441~443
 LU 分解 104 117~119
 论域 381 390 391
 逻辑变量 19
 逻辑运算 13 19 20 56 291
 Lyapunov 不等式 213 214
 Lyapunov 方程 97 129~133

M

M-函数 80 88 91 143 186 188 202 220
 Maclaurin 级数 66 67
 满秩矩阵 105 114
 冒号表达式 13 17 369
 Maple 语言 1 8 97 153 154 157~159 325
 Mathematica 语言 1 8
 Mellin 变换 145 157 158
 Mellin 反变换 157
 幂级数 66 67 94 137 138 143 162 320 327

幂零矩阵 139 141
 面向对象 429
 Mittag–Leffler 函数 52 330 334~337 437 438
 模糊规则 387
 模糊化 387 388
 模糊集合 3 379 381 382
 模糊逻辑工具箱 383~385 387
 模糊推理 2 10 379 382 385~389
 模拟退火算法 418
 模式搜索算法 418
 墨西哥帽小波 420 421 423
 Monte Carlo 方法 345 359 360
 Moore–Penrose 广义逆矩阵 112 113 128
 目标规划问题 226
 目标函数 3 181 186 191~199 201~212 215
 218~227 312 323 362 364 399 409~419
 MuPAD 语言 8 55 56 153 168 325

N

n -因子方差分析 又见 多因子方差分析
 NAG 软件包 4 5
 内核函数 16 104 105 156
 Neumann 函数 又见 Bessel 函数 332
 Neumann 条件 285 286
 Newton–Raphson 迭代法 51 276
 逆分布函数 345~349 351 352
 逆矩阵 97 110~112
 匿名函数 30 45 84 85 87~89 91 92 416 419
 NIT 见 数值积分工具箱
 Nyquist 频率 341 344

O

偶函数 337 338
 Oustaloup 滤波器 434~436 441 442

P

Padé 近似 301 327~330
 抛物型偏微分方程 282 283
 Pareto 解集 223 224
 偏导数 60~63 205 286 316 317
 偏微分方程工具箱 237 280~285 289
 Poisson 分布 345~347 353
 PSO 又见 粒子群优化 416

Q

齐次方程 127 142
 奇异矩阵 104 105 110~113 115~117 122~125
 260 263
 奇异值 5 106 107 124~126
 奇异值分解 16 97 105 124~126 266
 前馈 379 396~399 403~405
 前向差分公式 78
 切换微分方程 237 264 265
 切面 48~50 312
 穷举方法 181 206 207 209 210 224
 求和 24 82 172 335 395
 区间估计 362~364
 区间函数 55
 曲面积分 9 53 76~78
 曲线积分 9 53 74~76 145 171~173
 曲线拟合 306 316 322 401 402 404
 全局最优解 181 193~195 204 206 207 209 211
 232 408 410 411 415 418 419
 权值 228~230 395 396 398~401 404

R

染色体 408 409
 Rayleigh 分布 346 351 352 354 356 360
 人工神经网络 3 10 301 379 395~406
 任意矩阵 103 139
 Riccati 不等式 216 217
 Riccati 方程 97 133 134 187 188 215 217
 Riccati 微分方程 256 257
 Riemann–Liouville 定义 429~431 437 442~445
 Riemann 曲面 163~165
 Romberg 算法 82
 Rosenbrock 函数 195 196 208
 Rössler 方程 298
 Runge–Kutta–Fehlberg 算法 246 259
 Runge–Kutta 算法 3 243 253 258 268 298
 Runge 现象 303

S

三次方根 19 165
 三次样条 302 313~316 318
 散点 164 308 311 312

散度 282
 三对角矩阵 99
 三角分解 又见 LU 分解 97 104 117
 三维隐函数 45
 Schur 补 214 215 217
 Schur 分解 132 187
 上近似集 389 391 392
 舍入误差 243 244
 神经网络 见 人工神经网络
 神经网络工具箱 379 395~398 403~405 416
 时变差分方程 175
 视角 13 45~47 331
 试探结构 13 26
 适应度 409
 受控对象 267
 数据结构 13~15 123 163 313 388
 数据挖掘 389 392
 属于 379~381 389
 数值分析 1~4 10 11 53 72 81~86 244 253
 数值积分 53 81~92 149~151 301 313 316 318
 数值积分工具箱 89
 数值微分 9 53 78~81 313 316 317 431 434
 数值线性代数 4 8 97
 数值秩 105
 数值 Laplace 变换 150
 数值 Laplace 反变换 149 150
 双重极限 57 92
 双精度 14 30 72 85 102 106 123~125 434
 双曲型偏微分方程 283 286
 双线性变换 21 163 179
 双因子方差分析 345 371~373
 四维图形 13 48 49 312
 死循环 188
 Sigmoid 函数 384 386 396
 Simpson 算法 82
 Stein 方程 131 132
 Stiff 方程 又见 刚性方程 257
 随机变量 345 352 355~359
 随机数矩阵 97 98 354
 Sylvester 方程 97 131~133

\mathcal{T}

T 分布 345 346 348 349
 T 检验 367
 Taylor 级数 53 66~69 71 79 266 319 320
 特解 127 238 239
 特殊函数 10 65 301 330~337
 特殊矩阵 97 98 100 102
 特征多项式 97 101 107~109 114
 特征向量 5 97 114 115 373
 特征值 97 105~107 113~116 121~124 167 168
 238 373
 特征值型偏微分方程 283
 梯度 60 80 81 181 186 195 196 205 282 401
 体视化 13 48 312
 体视化数据 48 49
 梯形法 82 83 305
 条件数 105 125
 条件属性 390 392
 条件约简 379 393 394
 条件转移结构 13 24 25 43
 条形图 35
 统计学工具箱 345 346 352 356~358 360 362
 通解 97 238 239 332
 通项公式 73 74
 Trelea 算法 416
 椭圆型偏微分方程 282 283 289

 \mathcal{V}

Van der Pol 方程 3 242 249 250 257
 Vendermonde 矩阵 97 100~102 109

 \mathcal{W}

网格数据 41 43 48 80 81 224 306 308 311
 314 358 389
 网格图 41 46 285 307 358
 伪代码 31
 微分代数方程 11 246 257 262~264
 微分方程 237~300 330 429
 伪逆 又见 Moore-Penrose 广义逆矩阵 112
 伪随机数 98 99 267 345 353 354 359 361
 唯一解 97 126 130 132
 唯一性 4

误差限 14 35 86 89 105 112 186 243 249
 无穷级数 66 72~74 149 155 331 334 335
 无限长脉冲响应 340
 无向图 227 230 231 235
 无约束最优化 190~196 379 398 414 417 418

 \mathcal{X}

细节系数 424~426
 稀疏矩阵 97 103 227~229 231
 下近似集 389 391 392
 线性代数方程 5 97 126~129 132 181
 线性规划 3 198~201 206 208 212 215 218 221
 223 414 415 418
 线性矩阵不等式 9 134 135 181 212~217
 线性组合 127 320 322 362 395 440
 显著性水平 366 367
 相伴矩阵 97 101 102 121
 相对误差限 84 86 249 252 270
 相关函数 337~339
 相轨迹 247 298
 香蕉函数 又见 Rosenbrock 函数 195
 向量化 24 291 416 417
 相平面 249 250 265 298
 相容初始条件 262 264
 相似变换 97 114 116 121
 小波变换 301 420~428
 小波反变换 又见 小波反演 421
 小波反演 421 422
 小波工具箱 379 422 423 427
 小波系数 421 422 426
 协方差矩阵 266 267 357~359 373
 信息决策系统 390~392
 序列求积 9 73 74
 旋转 18 19 37 43 45 47 48 391
 循环结构 7 23~25 43 47 87 149 154 157 175
 177 188 203 204 221 353
 训练 379 395~399 401~407 416

 \mathcal{Y}

YALMIP 工具箱 212 217~219
 延迟微分方程 3 268~273 291~294
 严格包含 380

- 样本点 82 150 151 301~306 308~315 318~320
 323 355~357 364 374 396~399 401 404
 仰角 46
 样条插值 53 83 301~308 312~318 403 404
 样条插值工具箱 302 313 315
 样条函数对象 313~315
 么矩阵 97 98 124
 遗传算法 10 194 379 407~419
 遗传算法最优化工具箱 409
 异或运算 20 380
 隐层 396~399 401~403
 隐层节点 396~398 401 402 404
 隐函数 13 35 36 61 62 181 182
 引力线 35 51 60 81 287~289
 因式分解 20 169 170
 隐式微分方程 237 257 260~262 264
 映射 69 145 146 163 164 387 392~394 396
 友矩阵 又见 相伴矩阵 101
 有理式近似 325~330 344
 有限长脉冲响应 340
 有限元法 4
 有限 Fourier 变换 又见 离散 Fourier 正余弦变换
 有向图 227~231 235
 有约束最优化 9 181 186 197~202 379 414~418
 与运算 19 385
 源程序 5 8 26 31 115 149
 原点矩 356 357
 原函数 57 63~66 89 153 320 325 423
 原型函数 261 306 308 321~324 338 364
 约简 379 389 391~394
 约束条件 197 213 215 219 222~227 415

 \mathcal{L}
 z 变换 9 145 159~163 173 174 176
 z 反变换 159~162 173
 真值表 393
 正定矩阵 115 120 130 213
 正规矩阵 120
 正交基 116
 正交矩阵 116 125 266
 正视图 46 47
 整数线性规划 205 206
 正态分布 98 345~348 353 355 356 358 360
 363 367~369 383
 秩 97 105 106 113 116 124 125 127 128
 直方图 35 36 40 268 296 354
 置换 103 104 118
 置换矩阵 117 118
 质量矩阵 246 264
 质数 22 23 381
 置信度 360~365 370
 置信区间 360~368
 质因数分解 22
 中立型延迟微分方程 237 268 270 272 273
 种群 408 409 413~415
 中心差分算法 53 79 80
 中心矩 356 357
 主元素 117 118
 主子行列式 120
 转置 17 129 184 269 311
 状态变量 175 176 237 242~249 251 253~258
 260~263 266~273 278 279 292~295
 状态空间 7 241
 准解析解 181 183~186 238
 字符串 14 15 20 21 38 45 56 149 184 208
 238 240 245 380 385 398 409
 子矩阵 13 17 99 104 105 120 324
 自相关函数 337~339
 最大公约数 22 167 169
 最大值 17 191 197 199 222 224 386 409 411
 最短路径 227~231
 最佳妥协解 222
 最小二乘 18 97 128 220 223 301 318 321~323
 345 362~364
 最小二乘曲线拟合 322~325
 最小公倍数 22
 最优化工具箱 192 193 195 198 199 201 202
 207 225 226 323 408~411 416
 左右极限 又见 单边极限 54