

高等学校计算机科学与技术教材

基于 MATLAB 的 实用数值计算

石辛民 郝整清 编著

清华大学出版社
北京交通大学出版社

· 北京 ·

内 容 简 介

本书分两部分,第一部分紧扣数值计算介绍了 MATLAB 语言的基础知识:数值矩阵及其运算,字符串和符号矩阵,基本绘图和编程方法。第二部分介绍数值计算的基本内容:计算误差,代数方程及方程组的数值求解,插值法和数据拟合,数值积分和常微分方程初值问题数值解等。书中配有大量例题和适量的练习题,书末附有 MATLAB-7 的内容列表、习题参考答案及本书中使用的指令索引等。

与传统数值计算教材不同,本书把 MATLAB 语言和数值计算进行了有机结合,叙述简明易懂,内容详尽实用。本书既可以作为非计算数学类专业学生学习“数值计算(计算方法)”课程的教材,也可以作为学习用 MATLAB 进行数值计算的入门书,还可以供工程技术和科研人员阅读和参考。

版权所有,翻印必究。举报电话 010-62782989 13501256678 13801310933

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

本书防伪标签采用特殊防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

基于 MATLAB 的实用数值计算/石辛民,郝整清编著. —北京:清华大学出版社,北京交通大学出版社,2006.2

(高等学校计算机科学与技术教材)

ISBN 7-81082-645-X

I. 基... II. ①石... ②郝... III. 计算机辅助计算:数值计算-软件包, MATLAB-高等学校-教材 IV. TP391.75

中国版本图书馆 CIP 数据核字(2005)第 129964 号

责任编辑:吴嫦娥

出 版 者:清华大学出版社 邮 编:100084 电 话:010-62776969 <http://www.tup.com.cn>

北京交通大学出版社 邮 编:100044 电 话:010-51686414 <http://press.bjtu.edu.cn>

印 刷 者:北京鑫海金澳胶印有限公司

发 行 者:新华书店总店北京发行所

开 本:185×260 印 张:13 字 数:325 千字

版 次:2006 年 2 月第 1 版 2006 年 2 月第 1 次印刷

书 号:ISBN 7-81082-645-X/TP·247

印 数:1~4000 册 定 价:19.00 元

本书如有质量问题,请向北京交通大学出版社质监组反映。对您的意见和批评,我们表示欢迎和感谢。

投诉电话 010-51686043 51686008,传真 010-62225406, E-mail press@center.bjtu.edu.cn。

前 言

实验研究、理论分析和科学计算已经成为当代科学研究中不可或缺的三种主要手段。处于计算机时代的今天,科学计算是以数学模型为基础、以计算机和数学软件为工具进行的模拟研究,它是数学通向其他学科的桥梁,是当今盛行的计算机仿真技术的重要基石。

高等教育中如何培养学生科学计算能力已日益受到重视,许多理工类专业都开设有“数值计算”或“计算方法”课。但是,学习的目标却不尽相同:“计算数学”类专业的学生是为“研究”和“创造”算法而学习,而大多数理工科的学生则是为了“使用”算法,是想通过学习了解数值计算原理,学会选择算法和进行计算,为本专业的业务工作服务。本书的选材、安排均定位在“使用”算法上。

现今科学计算的研究方法是计算机和数学的有机结合,计算机已经成为科学计算必不可少的物质基础。数值计算确实需要理论上的指导,但落脚点必须是计算,学习数值计算就要自己会算,更应该会使用计算机算。在计算机高度发展和普及的今天,科技工作者科学算法意识的建立和计算能力的培养,必须在计算机环境下、在适当的平台上通过实际操作进行。非计算数学类专业中开设“数值计算”课程,只讲计算方法的理论,不用计算机进行实际演练,则有悖于“重视实用性和可操作性”的工程教育思想。出于这种考虑,本书尽量使数值计算理论与计算机软件应用紧密结合在一起,使学习数值计算理论与使用计算机实践操练有机结合在一起。

“工欲善其事,必先利其器”。在明确了数值计算必须与计算机相结合的原则后,选择实现“数值计算”的工具——计算机软件就显得非常重要。以前,很多人进行过数值计算和 Basic、Fortran 或 C 语言结合的教学试验,但是发现除计算机专业学生外,其他专业的学生还必须花大力气去学习计算机语言的编程,两者结合效果并不理想。从“使用”算法考虑,易学好用的 MATLAB 软件比 C 或 Fortran 等语言平台更适合本课程的教学要求。同时,要想在科学计算上与国际接轨,就必须学会 MATLAB。

本书首先对 MATLAB 基础作了介绍,内容包含 MATLAB 中有关数值计算、符号运算、绘图方法和编程基础等方面的基本内容,讲解都是围绕数值计算这个中心展开的。后面介绍了数值计算方面的内容:代数方程和方程组的数值求解、插值、拟合、数值积分、常微分方程等方面的计算理论,每部分都和 MATLAB 的实现进行了有机结合。介绍的计算方法理论都是基本的、经典的,需要的先修课程是高等数学和线性代数。学习计算方法理论的目的是要了解数值计算的基本原理,理解并熟悉科学和工程计算基本概念,树立起根据实际情况和具体需求选择算法的思想意识。

学习本书约需 40~70 学时,包含上机这个不可缺少的环节,使用时可根据需要决定内容的取舍。

本书既可以作为“数值计算”课程的教材,也可以作为学习 MATLAB 的入门书,同时对于需要进行数值计算或实验数据处理、作图的科技工作者,也可以从中找到所需要的内容。本书叙述简明易懂,内容详尽实用,通过自学和上机便可很快掌握其中的内容。

编者从 1999 年开始把数值计算和 MATLAB 结合在一起进行教学,每年都对编写的讲义做些修改,最终形成现在这本教材。周晓蕾高级实验师对本书的形成帮助极大,高丽丽教师及 1999 届以来的历届同学和都在本书的形成过程中提出过许多宝贵建议。此外,特别要提及的是北京交通大学出版社吴嫦娥编辑,在本书的出版中起了极大的推动作用,没有她的积极工作,本书不可能这么快就与读者见面。在此,对帮助过本书的所有人表示衷心感谢!

由于编者水平有限,恳请广大读者不惜赐教!

石辛民 郝整清
(E-mail: zushixm@126.com)
2006 年 1 月

目 录

第 1 章	MATLAB 预备知识	(1)
1.1	MATLAB 的基本功能	(1)
1.2	MATLAB 系统的安装、启动和退出	(2)
1.2.1	MATLAB 系统的安装	(2)
1.2.2	MATLAB 系统的启动和退出	(2)
1.3	MATLAB 的指令窗	(2)
1.3.1	指令窗中快捷按钮的功能	(3)
1.3.2	功能键的使用	(4)
1.3.3	在线查询方法	(4)
1.3.4	数据变量的删除、存储与调出	(8)
1.4	MATLAB 的范例演示窗	(8)
	练习题 1	(10)
第 2 章	MATLAB 语言基础	(11)
2.1	数值矩阵	(12)
2.1.1	永久性数值变量名	(12)
2.1.2	数值矩阵的创建	(12)
2.1.3	数值矩阵元素的标识与修改	(17)
2.1.4	数值矩阵的矩阵算法	(19)
2.1.5	数值矩阵的数组算法	(24)
	练习题 2.1	(29)
2.2	字符串和符号矩阵	(30)
2.2.1	字符串变量和函数求值	(30)
2.2.2	符号变量	(34)
2.2.3	符号矩阵的创建方法	(38)
2.2.4	符号矩阵元素的标识和删改	(39)
2.2.5	符号矩阵的运算	(40)
2.2.6	符号矩阵运算中的几个特有指令的应用	(41)
	练习题 2.2	(45)
2.3	基本绘图方法	(46)
2.3.1	绘图入门	(46)
2.3.2	二维图形的绘制	(48)
2.3.3	控制图形、画面的一些操作方法	(56)
2.3.4	三维图绘制初步	(57)

练习题 2.3	(63)
2.4 MATLAB 语言编程	(64)
2.4.1 MATLAB 的编辑调试窗	(64)
2.4.2 两类 M-文件	(65)
2.4.3 关系和逻辑运算	(67)
2.4.4 程序结构	(69)
2.4.5 两类 M-文件的转换	(74)
2.4.6 编程中的一些控制指令	(75)
练习题 2.4	(76)
第 3 章 误差和 MATLAB 的计算精度	(78)
3.1 误差	(78)
3.1.1 误差的来源	(78)
3.1.2 有关误差的一些概念	(79)
3.2 MATLAB 中的数值计算精度	(81)
3.2.1 浮点数及其运算特点	(81)
3.2.2 MATLAB 中的数值计算精度	(81)
3.3 设计算法的若干原则	(83)
3.3.1 算法的数值稳定性	(83)
3.3.2 设计算法的若干原则	(84)
练习题 3	(85)
第 4 章 求解非线性方程 $f(x)=0$	(86)
4.1 求解 $f(x)=0$ 的 MATLAB 符号法	(86)
4.2 求方程 $f(x)=0$ 数值解的基本方法	(88)
4.2.1 求实根的二分法原理	(88)
4.2.2 迭代法	(89)
4.2.3 切线法	(91)
4.2.4 割线法(弦截法)	(92)
4.3 方程 $f(x)=0$ 数值解的 MATLAB 实现	(93)
4.3.1 代数方程的求根指令 roots	(93)
4.3.2 求函数零点指令 fzero	(93)
4.4 求解非线性方程组数值解的迭代法	(96)
4.5 求方程组数值解的指令	(96)
练习题 4	(98)
第 5 章 求解线性代数方程组的直接法	(100)
5.1 线性代数方程组求解概论	(100)
5.1.1 线性代数方程组的矩阵表示	(100)

5.1.2	线性代数方程组解的性质	(100)
5.2	恰定线性代数方程组求解	(101)
5.2.1	克莱姆法则	(101)
5.2.2	高斯消去法	(102)
5.3	矩阵的三角分解	(104)
5.3.1	高斯消去法和三角矩阵	(104)
5.3.2	矩阵的三角分解	(104)
5.4	线性代数方程组数值解和矩阵三角分解的 MATLAB 实现	(105)
5.4.1	齐次线性代数方程组求解指令	(105)
5.4.2	求解非齐次线性代数方程组的 MATLAB 方法	(107)
5.4.3	矩阵分解指令	(111)
	练习题 5	(113)
第 6 章	求解线性代数方程组和计算矩阵特征值的迭代法	(115)
6.1	求解线性代数方程组的迭代法	(115)
6.1.1	迭代法的基本原理	(115)
6.1.2	雅可比迭代法	(115)
6.1.3	赛德尔迭代法	(117)
6.1.4	迭代法的敛散性	(117)
6.2	方阵特征值和特征向量的计算	(120)
6.2.1	方阵特征方程的求解	(120)
6.2.2	计算特征值和特征向量的迭代法	(121)
6.3	矩阵一些特征参数的 MATLAB 求算	(123)
6.3.1	求方阵特征值的有关指令	(123)
6.3.2	矩阵的正交三角分解指令 <code>qr</code>	(125)
6.3.3	计算范数和矩阵谱半径的指令	(127)
	练习题 6	(128)
第 7 章	插值法和数据拟合	(130)
7.1	多项式插值	(130)
7.1.1	代数多项式插值的基本原理	(131)
7.1.2	两种常见插值法	(131)
7.1.3	插值多项式的误差估计	(134)
7.2	分段三次插值和三次样条插值	(135)
7.2.1	分段三次 Hermite 插值	(136)
7.2.2	三次样条插值的基本原理	(136)
7.2.3	三次样条插值函数的一种具体推导方法	(137)
7.3	插值法在 MATLAB 中的实现	(138)
7.3.1	一元函数的插值(查表)指令 <code>interp1</code>	(139)

7.3.2 三次插值和三次样条插值指令	(139)
7.4 数据的曲线拟合	(141)
7.4.1 数据曲线拟合的最小二乘法	(141)
7.4.2 超定方程组的最小二乘解	(142)
7.5 多项式运算在 MATLAB 中的实现	(143)
7.5.1 多项式及其系数向量	(143)
7.5.2 多项式运算	(144)
7.6 曲线拟合在 MATLAB 中的实现	(146)
7.6.1 数据的多项式曲线拟合	(146)
7.6.2 多项式数据拟合应用的扩充	(148)
练习题 7	(149)
第 8 章 数值积分	(151)
8.1 计算积分的 MATLAB 符号法	(151)
8.2 牛顿-柯特斯求积公式	(154)
8.2.1 牛顿-柯特斯求积公式推导	(154)
8.2.2 牛顿-柯特斯求积公式的误差估计	(156)
8.3 几个低次牛顿-柯特斯求积公式	(157)
8.3.1 矩形求积公式	(158)
8.3.2 梯形求积公式	(158)
8.3.3 抛物线求积公式	(158)
8.4 复合求积公式及其 MATLAB 实现	(159)
8.4.1 复合矩形求积法及其 MATLAB 实现	(160)
8.4.2 复合梯形求积法及其 MATLAB 实现	(161)
8.5 变步长复合求积及其 MATLAB 实现	(163)
8.5.1 复合抛物线求积公式	(163)
8.5.2 变步长复合抛物线求积公式	(163)
8.5.3 求数值积分的指令 quad 和 quadl	(164)
练习题 8	(166)
第 9 章 常微分方程初值问题的数值解	(168)
9.1 求解常微分方程的 MATLAB 符号法	(168)
9.1.1 常微分方程的 MATLAB 符号表示法	(169)
9.1.2 求解常微分方程的符号法指令 dsolve	(169)
9.2 常微分方程数值解的基本原理	(171)
9.2.1 求常微分方程数值解的基本原理	(171)
9.2.2 泰勒展开法	(172)
9.2.3 龙格-库塔法	(173)
9.2.4 阿达姆斯法	(175)

9.3 常微分方程初值问题数值解的 MATLAB 实现	(176)
9.3.1 求常微分方程初值问题数值解的指令	(176)
9.3.2 ode23 使用方法举例	(177)
练习题 9	(183)
附录 A MATLAB-7 内容列表	(184)
附录 B MATLAB 指令索引	(186)
附录 C 部分练习题参考答案	(189)
参考文献	(197)

第 1 章 MATLAB 预备知识

本章介绍使用 MATLAB 的预备知识,对它的几个常用视窗,特别是对指令窗做了重点介绍。为了便于今后的学习,较详细地举例说明了几个常用在线查寻指令的使用方法。

1.1 MATLAB 的基本功能

当今世界上流行的 30 多个数学类软件中, MATLAB 语言处于数值计算型软件的主导地位,适用范围几乎涵盖了工程数学的各个方面。自从 1984 年美国 MathWorks 公司把它推向市场以来,由于功能强大,语法简单,使用方便,界面友好,很快便受到科技界的普遍欢迎。它的强大开放性,使得以它为基础又开发出了 20 多个专用工具箱,如信号处理、控制系统、神经网络等专用工具箱,这对于解决不同学科的专业计算和仿真设计问题提供了极大的方便,使它如虎添翼,应用范围更加广阔。该软件已经成为发达国家高等院校理工科大学、硕士生、博士生所必须掌握的软件,而它的多种工具箱也已成为科学研究和工程设计部门解决各种具体问题的一种标准软件。

MATLAB 是英文“MATRIX LABORATORY”(矩阵实验室)的缩写,该软件集计算、绘图和仿真于一身,其基本功能可以概括为下列 4 个方面。

(1) 数值计算(Numeric)

它具有庞大的数学函数库,可以进行工程数学中几乎所有的数值计算。对科学和工程中的各种数值计算、统计等各类数学问题都能提供方便快捷的计算方法。多数计算过程是调用一个指令就可以解决问题,就像使用计算器一样方便,它的计算精度可以任意选取,能满足各种不同的需求。

(2) 符号运算(Symbolic)

它具有对数学公式进行符号处理的功能,像因式分解、公式推导、求函数的导函数、幂级数展开及解微分方程组这类数学问题,只要用一条指令,调用一个 M-函数文件,就可以轻而易举地予以解决。

(3) 数据可视化(Graphic)

它具有很强的数据可视化能力,可以把计算所得的数据根据不同的情况和要求轻松地绘制出二维、三维图形,并可对图形的线条、色彩、视角等加以变换。它还可以用不同坐标系进行作图,绘出各种特殊的几何图形(如直方、柱状、网面等),把数据关系的特征形象化得淋漓尽致。

(4) 建模仿真(Simulink)

它具有动态系统建模、仿真和分析的集成环境。用户只要在视窗里通过简单的鼠标操作,就可以建立起直观的方框图式系统模型,比用传统软件包从微分方程、差分方程出发进行的建模更直观、方便和灵活。通过改变模块中的参数和模块间的连接方式,观察系统中发生的变化,结合仿真的结果,可对系统进行全面或局部的调试和修改,以求达到理想效果。

1.2 MATLAB 系统的安装、启动和退出

1.2.1 MATLAB 系统的安装

MATLAB 必须在 Windows 环境下安装。用光盘安装的基本步骤如下。

① 启动 Windows。

② 依次单击菜单【开始】→【控制面板】→【添加/删除】图标,然后按照显示出的提示,逐一操作进行安装。安装中可根据实际需要和硬盘空间大小,从显示出的信息上选择安装的具体条目内容(参见附录 A MATLAB-7 内容列表)。

注:鼠标操作术语意义——单击:按一下鼠标左键。

右击:按一下鼠标右键。

双击:连续快按两下鼠标左键。

拖动:按下鼠标左键同时移动鼠标,将屏幕界面中的对象移动到指定位置。

1.2.2 MATLAB 系统的启动和退出

1. 启动




打开计算机,进入 Windows 环境后,启动 MATLAB 的途径有两种:

① 若面板上有 MATLAB 小图标,则双击它即可进入;

② 单击屏幕左下角的小图标【开始】,再依次单击弹出的菜单【程序】→【MATLAB】项,即可进入。

2. 退出

在 MATLAB 界面下退出的途径有三种:

① 单击 MATLAB 界面右上角最右侧的关闭图标 (与它相邻的其他两个图标功能是最左的为使界面最小化,中间的为变换界面大小);

② 在指令窗中键入 `exit` 或 `quit` 指令后回车;

③ 依次单击菜单【File】→【Exit MATLAB】即可。

1.3 MATLAB 的指令窗

进入 MATLAB 后,它的任何界面上都有一行如下所示的主菜单:【File】(文件)、【Edit】(编辑)、【Debug】(调试)、【Desktop】(桌面)、【Window】(视窗)和【Help】(帮助)。单击【Desktop】则出现如图 1-1 所示的界面。

在图 1-1 界面上依次单击子菜单【Desktop Layout】(桌面设置)→【Command Window On-

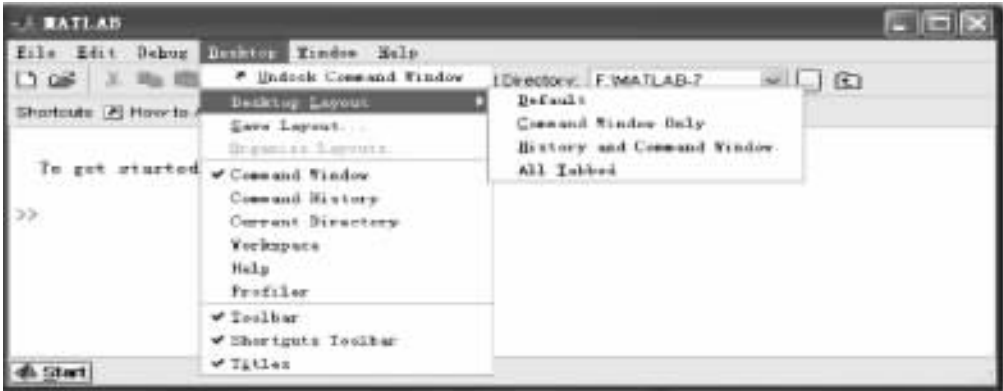


图 1-1 MATLAB 指令窗

ly】(只显指令窗) ,则显示出我们常用的指令窗 ,它是进行人机对话的主要窗口。


指令窗中的每个主菜单下都含有数量不等的子菜单 ,可以像刚才打开子菜单【Command Window Only】时一样依次单击各级子菜单 ,直到最后打开需要的界面。

例如 ,为了改变屏幕上显示文字的行间距 ,可以在指令窗界面上依次单击菜单【file】→【preferences】 ,在弹出的对话框右侧上方 ,选中【Numeric display】下的 compact (紧凑) ,然后单击对话框中 按钮 ,就可以使得此后屏幕上的输出数据行间距较小 ,若选中 loose (稀疏)则屏幕上文字的行间距较大。这项功能也可以在指令窗中键入指令 `format compact` 或指令 `format loose` 来实现。

1.3.1 指令窗中快捷按钮的功能

指令窗中主菜单一行下面有一排小方图标 ,它们是一些常用子菜单的快捷按钮 ,单击它们和打开相应子菜单的功能一样 ,但操作比较快捷。现在把它们与相应的主、子菜单名称及其功能一并列在表 1-1 中。

表 1-1 指令窗中快捷按钮与相同功能菜单对照表









按钮图标	相应菜单名称		功能
	主菜单	子菜单	
	【File】(文件)	【New M-File】	创建新的 M-文件
		【Open File】	打开存盘的 M-文件
	【Edit】(编辑)	【Cut】	剪掉选定的内容
		【Copy】	把选定内容送入剪贴板
		【Paste】	粘贴已选定的内容
		【Undo】	撤销刚执行过的操作
		【Redo】	恢复刚撤销的操作
	Simulink(仿真)		浏览仿真模块库的内容
	GUIDE(指南)		创建和打开指南说明
	【Help】(帮助)	【MATLAB Help】	帮助了解软件内容

1.3.2 功能键的使用

在指令窗中用好功能键可以提高操作效率,现将常用的几个功能键列在表 1-2 中。

表 1-2 功能键的作用

功 能 键	功 能	功 能 键	功 能
 ()	调出上(下)一个指令		使光标移到行首
	使光标向左移动		使光标移到行尾
	使光标向右移动		删除光标所在行的全部内容
	翻到上一页		删掉光标左侧的字符
	翻到下一页		删掉光标右侧的字符
 + C	强行中止程序的运行	 + K	删除光标到行尾的内容

由表 1-2 可知,  和  两个功能键的作用与文字编辑处理时的功能完全不同。那里  和  键的作用是使光标上下移动,而这里则是调出已经输入过的指令。例如,在连续三行键入过三条指令后,现在想再次输入其中的第二条,就不必重新键入,只要连按两次功能键  ,若回车后又想使用先前键入的第三条指令,则再按两下功能键  。连续多次按动  键或  键,可不断显示出先前键入过的指令,免得重复键入相同的指令,从而提高了操作效率。

1.3.3 在线查询方法

MATLAB 的内容丰富、功能强大,指令、函数也特别多,学会在线查询是掌握 MATLAB 的捷径,也是学习它的基本内容之一。充分利用在线查询功能完全可以代替翻阅“用户手册”。

MATLAB 中常用的查询方法有两种:指令法和菜单法。

指令法就是在指令窗中键入查询帮助指令,回车后屏幕上就显示出需要了解的内容。

表 1-3 列出了一些常用的查询指令及其功能的简要说明。

表 1-3 常用查询、帮助指令表

指令使用格式	功 能 说 明
help 标识符	显示该“标识符”代表的信息,可以是总目录 topics(可省略)、函数库名、函数指令名称或工具箱名称等
lookfor 关键词	显示一批功能含有“关键词”的指令
type M-文件名	指明该“M-文件”的性质,或显示其全部内容
doc M-文件名	显示有关该“M-文件”的全部资料(与 demos 调出的相同)
who	显示工作空间中所有变量名称
whos a b	显示工作空间中 a、b(省略时为全部)变量的详细资料
what	显示当前目录下的全部 M-文件名称

续表

指令使用格式	功 能 说 明
dir	显示当前目录下的文件或子目录名称
ver	显示 MATLAB 的全部工具箱
path	显示或设置 MATLAB 的搜索路径
which M-文件名	显示该“M-文件”所在子目录的路径
cd	查询或改变路径

菜单法是在任何一个打开的窗口中单击主菜单【Help】(帮助),在显示出的下属子菜单中单击相关项目进行查询。它的使用像翻阅百科全书一样,按内容分类由章到节地逐级查找,最后单击需要查看的内容条目,就会看到详细的说明。

菜单法优点是不必记忆指令,可以像查字典一样从屏幕上边查边看,但较繁杂。人们经常使用的是指令法,它的使用较为简便,只是需要记住各个指令及其功能。下面举一些利用常用指令查询的例子,以便仿效。

例 1-1 指令 help 的应用举例。

解 (1) 在指令窗中键入 help (相当于键入 help topics),回车则得出系统中装入的函数库和工具箱名称(注:为了方便调用,加注了中文译文):

字符 结构)	matlab \general	——General purpose commands (通用命令)
	matlab \ops	——Operators and special characters (算子和特殊
	matlab \lang	——Programming language constructs (程序语言
	matlab \elmat	——Elementary matrices and matrix manipula- tion (基本矩阵及其操作)
	matlab \elfun	——Elementary math functions (初等函数)
	matlab \specfun	——Specialized math functions (特殊函数)
	matlab \matfun	——Matrix functions-numerical linear alge- bra (矩阵函数-线性代数)
	matlab \datafun	——Data analysis and Fourier transforms (数值分析和傅立叶变换)
	matlab \polyfun	——Interpolation and polynomials (插值和多 项式)
	matlab \funfun	——Function functions and ODE solvers (功能 函数和常微分方程)
	matlab \sparfun	——Sparse matrices (稀疏矩阵)
	matlab \graph2d	——Two dimensional graphs (二维图形)
	matlab \graph3d	——Three dimensional graphs (三维图形)
	matlab \specgraph	——Specialized graphs (特殊图)
	matlab \graphics	——Handle Graphics (手控图)
	matlab \uitools	——Graphical user interface tools (图形用户接口

工具)

matlab\strfun ——Character strings (字符串)
 matlab\iofun ——File input/output (文件输入/输出)
 matlab\timfun ——Time and dates (时间和日期)
 matlab\datatypes ——Data types and structures (数据类型/结构)
 matlab\winfun ——Windows Operating System Interface Files
 (DDE/ActiveX)
 (Windows 操作系统接口文件)
 matlab\demos ——Examples and demonstrations (举例和演示)
 ...

(2) 如果了解某个函数库中包含的函数指令,比如想了解“基本矩阵及操作库(matlab\elmat)”中的指令内容,键入:

```
help elmat
```

回车得出该库包含的子库及它所包含的指令:

Elementary matrices and matrix manipulation(基本矩阵及其操作)
 Elementary matrices.

zeros ——Zeros array
 ones ——Ones array
 eye ——Identity matrix
 repmat ——Replicate and tile array
 rand ——Uniformly distributed random numbers
 randn ——Normally distributed random numbers
 linspace ——Linearly spaced vector
 logspace ——Logarithmically spaced vector
 meshgrid ——X and Y arrays for 3-D plots
 ... ——Regularly spaced vector and index into ma-

trix

Basic array information.

size ——Size of matrix
 length ——Length of vector
 ndims ——Number of dimensions
 disp ——Display matrix or text
 isempty ——True for empty matrix
 isequal ——True if arrays are identical
 isnumeric ——True for numeric arrays
 islogical ——True for logical array
 logical ——Convert numeric values to logical

Matrix manipulation.

...

Special variables and constants.

...

Specialized matrices.

...

为节省篇幅,最后三个工具箱只列出了名称,省略了其中包含的函数指令。

(3) 如果了解某个函数指令,比如“disp”的功能和使用方法,键入:

```
help disp
```

回车则得出有关指令 disp 功能的说明:

```
DISP Display array.
```

```
DISP (X) displays the array, without printing the array name. In
```

```
all other ways it's the same as leaving the semicolon off an expression except that empty arrays don't display.
```

```
If X is a string, the text is displayed.
```

```
See also int2str, num2str, sprintf, rats, format.
```

```
Overloaded functions or methods (ones with the same name in other directories)
```

...

例 1-2 用帮助指令 type 查看 M-函数指令 rank 的具体程序。

解 要了解 rank 的原程序,可键入:

```
type rank
```

回车得出:

```
function r = rank(A,tol)
```

```
% RANK Matrix rank.
```

```
% RANK(A) provides an estimate of the number of linearly independent rows or columns of a matrix A.
```

```
% RANK(A,tol) is the number of singular values of A that are larger than tol.
```

```
% RANK(A) uses the default tol = max(size(A)) * eps(norm(A)).
```

```
%
```

```
% Class support for input A:
```

```
% float: double, single
```

```
% Copyright 1984 - 2004 The MathWorks, Inc.
```

```
% |SRevision: 5.11.4.3 |S|SDate: 2004/08/20 19:50:33 |S
```

```
s = svd(A);
```

```
if nargin == 1
```

```
tol = max(size(A')) * eps(max(s));
```

```
end
```



```
r = sum(s > tol);
```

由此可知,“rank”是求矩阵秩的指令,只要输入 rank(A),就可得出矩阵 A 的秩。

注:① 程序里行首的% 号表示该行的内容为注释文字,运行程序时不予执行。

② 注释部分的全空行起隔离作用,使用“help rank”则只能调出第一个空行之前的注释内容,而“type”则调出全部注释内容及程序内容,这是两个查询指令在功能上的差异。

例 1-3 cd 使用举例。

解 (1) 在指令窗中键入:

```
cd
```

回车得出:

```
F:\MATLAB-7\work
```

指出当前运行在 F 盘的 MATLAB-7 子目录下的 work 中。

(2) 在指令窗中键入:

```
cd ..
```

回车后将从当前运行目录往上靠一级(到 F:\MATLAB-7)。

要恢复到刚才的 work 中,则键入:

```
cd F:\MATLAB-7\work
```

回车。

1.3.4 数据变量的删除、存储与调出

1. 删除类指令

如果想擦掉指令窗屏幕上显示的所有内容,可键入擦除指令 clc,回车即可,此时擦去的变量依然还在“workspace(工作空间)”中,只要不关机就可以重新调出使用。“workspace”是每次开机后的临时存储目录。

想把键入的所有变量从工作空间中删除,可在指令窗中键入删除指令 clear。如果只想删除部分变量,则在 clear 后键入想要删除的变量名称,两个变量名称之间用空格间隔,不得添加任何符号。


2. 存储与调出指令

save——该指令可将 workspace 中的变量存入磁盘,以便重新开机后调出使用。

例如,欲将使用过的变量 a b c 以 w_01 的文件名存入 a 盘,则可键入:

```
save a:\w_01 a b c
```

回车即可。

此时若不写路径和文件名,回车后则将变量存入当前目录下(键入 cd 回车可查知当前目录名)。如键入 save w_01 a b c 回车则将变量 a b c 存入子目录 work 下的 w_01 文件中。这时单击指令窗上的快捷按钮 ,便可查看,重新开机后仍可调用。

load——该指令可将磁盘中的变量调到 workspace 中,以便在指令窗中调用。

例如,现在要使用 workspace 中没有、但存在于子目录 work 下 w_02 文件中的变量 e, f,

则可键入：

```
load w_02 e f
```

回车,即可将 e, f 调入工作空间。

在指令窗中使用过的指令(含用 `clc` 擦去的)都被保留在 Command History 窗口中,需要时可依次单击指令窗中的菜单【Desktop】→【Command History】进行查看。

1.4 MATLAB 的范例演示窗

MATLAB 中有一个范例演示窗口,它相当于一本软件使用方法说明书,附有许多丰富的例子。利用它可以帮助我们快速学习 MATLAB 和查询相关指令的使用说明。

在指令窗中键入 `demos` 指令后回车,或者单击主菜单【Help】下的子菜单【Demos】,屏幕上就出现如图 1-2 所示的范例演示窗。

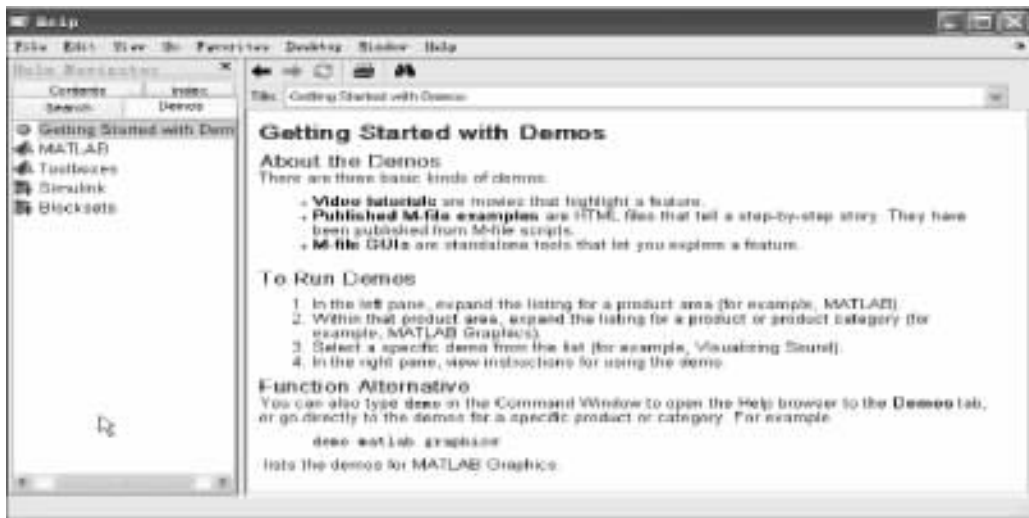


图 1-2 MATLAB 的演示窗口

该窗口左边有 4 项内容:Contents(内容)、Index(索引)、Search(查询)和 Demos(演示)。单击其中某一项,就会在左侧下方列出其下属的子菜单。

Contents 中包含着 MATLAB 所有内容的目录,单击某一项目录就在其下面列出该目录的下属子菜单,再单击其子菜单,其相应内容说明就显示在屏幕右侧,根据需要再在右侧打开相关的说明。

Index 中同样包含着 MATLAB 的所有内容的目录,不过目录是按其名称的头一个字母在英文字母表中的顺序排列的。单击 Index 后,下面便出现 26 个大写英文字母,单击某个字母就会将该字母打头的目录名称移列在前头。再单击要查看内容的目录名称,相关内容就会显示在屏幕右侧。


Search 是为用户查找相关内容设置的,单击 Search 后可在弹出的 Search for 对话框中填上待查看的内容,回车后就会将有关该内容子菜单在左侧全部列出,再单击欲查看的目录名称,就会在屏幕右侧显示出相关内容的说明。

Demos 集中了 MATLAB 各部分内容中的所有示例。

如果范例演示窗左边已经调出了许多演示子菜单,想使它们隐藏收回,可双击某一级菜单,这样就会使其下面已展现出的子菜单隐藏起来,恢复打开这些子菜单前的状态。

下面仅就 Demos 的使用作些说明,其他部分内容的调用与此类似。由图 1-2 可知, Demos 下属有 4 项主菜单:

MATLAB (MATLAB 基础部分)
 Toolboxes (工具箱)
 Simulink (仿真)
 Blocksets (模块)

双击其中某项菜单,就会在其下面显现出下一级子菜单,依次双击子菜单将逐级弹出更下一级的子菜单,直到出现左侧有小图标的条目,这是具体示例的名称。双击某项示例名称,右侧就显示出相关内容范例演示的说明。这时按照右侧图中给出的提示进行操作,便会出现具体示例的说明或图文并茂的演示。

例如,依次双击【MATLAB】→【Graphics】→【Vibrating Logo】,再点击屏幕右侧的 Run this demo(运行这个示例),就会出现图 1-3 所示的画面。图中右下角 Figure 1 里的动态画面是一个不断飘动的方格平面,背景的文字是关于它的说明。

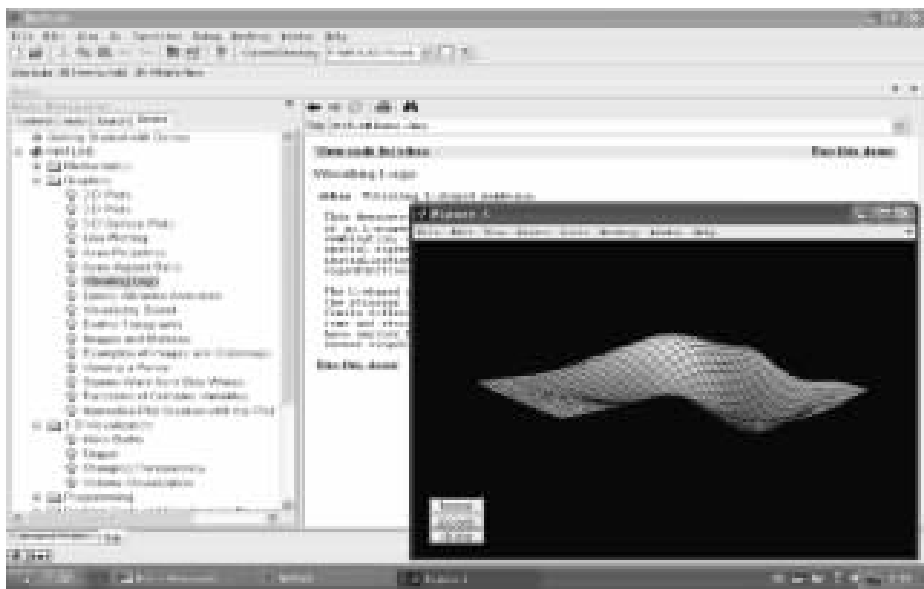


图 1-3 演示窗口中的动态平面

练习题 1

1. 熟悉 MATLAB 系统的进入和退出方法,并将书中例题自行操作一遍。
2. 用指令 help 调出“general”和“matfun”,了解这两个子目录的内容,并熟悉查询指令 help 的使用方法。
3. 用指令 help 和 type 分别查看指令“linspace”的说明和程序,比较这两个查询指

令的差异。

4. 在指令窗中每次键入下述指令之一 ,回车观察执行结果 :

$a = [4 \ 7 \ 9; 8.5 \ 6 \ 7]$, $t = 0:0.5:\pi$, $y1 = \sin(a)$, $y2 = \sin(t)$, $a * y1'$ 。

5. 用功能键调出第 4 题中矩阵 a 和第 3 题中的输入指令(不要重新录入)。

6. 将前面输入的变量 a , t , $y1$ 和 $y2$ 存入你的 a 盘中。重新启动 MATLAB 后 ,在指令窗中调出这些变量。

7. 打开 Command History 窗口 ,浏览你使用过的指令 ,然后关闭该窗口。

8. 熟悉演示指令 demos 的使用 ,通过“范例演示”窗 ,了解 MATLAB 关于矩阵的运算规则(逐级打开菜单【Demos】→【MATLAB】→【Mathematics】→【Basic Matrix Operations】)。

第2章 MATLAB 语言基础

本章介绍 MATLAB 的基础内容:数值计算、符号运算、绘图方法和编程入门。每部分都尽量介绍它们在数学中的应用,其中包括利用 MATLAB 进行的函数求值、矩阵运算和数学公式推导等方面的内容,并对以后常用的数据绘图和函数绘图也做了相当的介绍。这部分是 MATLAB 的重要基础,下面先介绍常用的几个基础概念。

1. 标识符

在 MATLAB 语言中把一串符号的组合叫字符串(Character Array 或 String),而把标志变量、常量或文件名称的特定字符串称为标识符(Identifier)。不是任何字符串都可以作为标识符的,作为标识符的字符串必须是由英文字母(大小写字母不等价,共计 52 个)、阿拉伯数字和下划线等符号组成的字符串,它的第一个符号必须得用英文字母;不符合这些规定的字符串,如“8ty”、“f(x)”、“k-q”、“文_01”等都不能作为标识符。

2. MATLAB 中的数据及变量类型

MATLAB 中有三种类型的基本数据,简介如下。

① 数值型数据,简称数值(Double Array 型数据):一般输入的数字均为数值数据,它包含实数、复数等。

② 字符串型数据,简称字符量(Char Array 型数据):用英文格式单引号加以界定的数字、字符、各种符号、表达式、方程式和汉字等。

③ 符号型数据,简称符号量(Sym Object 型数据):用 sym 或 syms 可以把字符、表达式、方程、矩阵等定义成数学符号,称为符号型数据,它的运算结果为数学表达式。

在指令窗中键入 class(a),回车即可得知已有变量 a 是哪种类型的数据。

3. 变量名及其赋值

代表上述三种数据的标识符称为变量名。让某个变量名等价地代表确定的数值、字符串或符号数据,就称为给该变量名赋值。变量名的赋值方法是用赋值符号“=”,将其右边的数据或已赋过值的变量名代表的数据,赋给左边的标识符(变量名),通用格式为

变量名(标识符) = 数据或已赋过值的变量名

此后,左边的变量名在参与各种运算或变换时完全等价地代表着右边的数据,直到给它赋予新的数据。

与三种类型的基本数据相对应有三类变量名,它们分别代表着数值、字符串和符号数据,其标识符分别被称为数值变量名、字符串变量名和符号变量名。

MATLAB 语言中各类数据的基本单元都是矩阵,因此与上述三类数据相对应便有数值矩阵、字符串矩阵和符号矩阵,它们是进一步参与运算、操作的基本单元。单个的元素、向量(元素序列)都可以看作是矩阵的特例,自然以矩阵为基本单元进行的创建、运算、变换等也适用

于单个数值、单个字符串和单个符号量及由它们构成的向量。本书以后的讨论都是针对矩阵进行的,但对于不同的数据类型及相应的矩阵,运算规律和使用的运算符号不尽相同,下面分别予以介绍。

2.1 数值矩阵

数值矩阵的基本形式是复数矩阵,一个数值、向量或实数矩阵都可以看作是复数矩阵的特例。为了不失一般性,下面的规则都是针对复数矩阵展开讨论的。

2.1.1 永久性数值变量名

在 MATLAB 中有一类数值变量名被称为永久性数值变量名,它们是系统事先定义了的,系统一旦启动它们就已存在,而且总是代表着固定的数值。常用的永久数值变量列在表 2-1 中。为了不引起混乱,一般不要再给它们赋予其他数值。

表 2-1 常用永久性变量名(预定义变量)

变 量 名	表示的意义	变 量 名	表示的意义
pi	圆周率 π	INF 或 Inf	正无穷大
eps	机器浮点运算误差限 ($2.220 4 \times 10^{-16}$)	i , j	虚数单位($i , j = \sqrt{-1}$)
ans	临时变量名,输出定义、运算结果时,用它代表未定义名称的变量	NaN	不定值(Not a Number),如 $\frac{0}{0}$, $\frac{\infty}{\infty}$, $0 \cdot \infty$ 。

2.1.2 数值矩阵的创建

1. 直接输入法

在 MATLAB 中,进行计算的基本单元是数值矩阵,由键盘直接输入是创建它的最基本方法。输入数值矩阵时,必须遵从以下规则。

- ① 矩阵的所有元素必须置于方括号“[]”内。
- ② 矩阵一行中元素之间用逗号“,”或空格分隔,矩阵相邻两行之间用分号“;”间隔或回车换行。
- ③ 如果输入内容太多,屏幕宽度不够或其他原因需要中途换行时,可键入三个连续的英文格式句号“...”称为续行号,回车后可继续键入后续的内容。续行号后的回车只是“换行”而不是“执行指令”。续行号不得加在矩阵一行中的两个元素之间。
- ④ 矩阵元素可以是实数、复数、向量、矩阵等数值量或其变量名,每行中元素个数必须相等。
- ⑤ 输完矩阵内容后加逗号“,”或回车,则显示出创建矩阵的内容;若在包含矩阵元素的右侧方括号外加有英文格式分号“;”,回车完成创建任务,但不显示矩阵内容。

⑥ 指令窗中同一行内输入几个矩阵或指令时,它们之间必须用逗号或分号间隔。

⑦ 完成矩阵输入后,无论光标在行内的什么位置,只要按回车键则“执行”该行的指令,不必非把光标移到行末再按回车(这里的回车作用是“执行”而不是“换行”)。

例 2-1 创建矩阵 $a = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \end{bmatrix}$,但不显示在屏幕上。

解 键入:

```
a = [1 2 3; 5 4 1]; % 逗号和空格等价
```

回车即可。为了验证它已在工作空间中,可键入:

```
a
```

回车得出:

```
a =
     1     2     3
     5     4     1
```

例 2-2 利用例 2-1 中的矩阵 a ,创建一个新矩阵 $a = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \\ 2i & 3-i & 9 \end{bmatrix}$ 。

解 新 a 中的前两行与例 2-1 中的 a 相同,故键入:

```
a = [a; 2i 3 - i 9]
```

回车得出:

```
a =
     1.0000     2.0000     3.0000
     5.0000     4.0000     1.0000
     0 + 2.0000i     3.0000 - 1.0000i     9.0000
```

创建复数矩阵时,除上述方法外也可以用实数矩阵和复数矩阵相加的方法。如对于本例的矩阵 a ,也可键入:

```
a = [a; 0 3 9] + [0 0 0; 0 0 0; 2i - i 0]
```

2. 创建特殊数值矩阵的指令输入法

对于某些特殊矩阵, MATLAB 中设有直接创建的专用指令,这给它们的创建、运算,特别是给编程带来很多方便。表 2-2 列出了一些创建特殊矩阵的专用指令。

表 2-2 创建特殊矩阵的专用指令

指令格式	功 能	指令格式	功 能
<code>zeros(n)</code>	输出 n 阶全零方阵	<code>rand(n)</code>	输出 n 阶均匀分布的随机方阵
<code>zeros(m,n)</code>	输出 $m \times n$ 阶全零矩阵	<code>rand(m,n)</code>	输出 $m \times n$ 阶均匀分布随机矩阵
<code>ones(n)</code>	输出 n 阶全 1 方阵	<code>randn(n)</code>	输出 n 阶正态分布随机方阵
<code>ones(m,n)</code>	输出 $m \times n$ 阶全 1 矩阵	<code>randn(m,n)</code>	输出 $m \times n$ 阶正态分布随机矩阵
<code>eye(n)</code>	输出 n 阶单位方阵; $n=1$ 时可以省略	<code>magic(n)</code>	输出 n 阶魔方阵(各行、列及两主对角线元素和均为 $(n^3 + n)/2$)
<code>diag(a,k)</code>	输出矩阵 a 主对角线右移 k 列时其元素构成的列向量。 $k=0$ 时可省略	<code>tril(a)</code> <code>(triu(a))</code>	输出矩阵 a 主对角线下(上)方元素构成的下(上)三角矩阵

例 2-3 创建一个 2×4 阶的全零矩阵。

解 键入：

```
zeros(2,4)
```

回车得出：

```
ans =  
    0    0    0    0  
    0    0    0    0
```

例 2-4 创建一个 2×3 阶随机矩阵(元素取值在 $[0,1]$ 间而无法事先确定的矩阵)。

解 键入：

```
as=rand(2,3)
```

回车得出：

```
as =  
    0.9501    0.6068    0.8913  
    0.2311    0.4860    0.7621
```

如果紧接着再一次键入 `rand(2,3)` ,得出的矩阵未必和它相同。

例 2-5 创建一个 3 阶魔方矩阵。

解 键入：

```
mf=magic(3)
```

回车得出：

```
mf =  
    8     1     6  
    3     5     7  
    4     9     2
```

该方阵的各行、各列及两个主对角线上元素之和都等于 $(n^3 + n)/2 = (3^3 + 3)/2 = 15$ 。

3. 变换矩阵结构的指令

在编写程序和一些特殊计算中 ,经常需要对已输入的矩阵结构作某种变换 ,即不改变矩阵中元素的总数和各元素的取值 ,仅使其位置发生变化。如使矩阵整体发生旋转、翻转等变换 ,称为变换矩阵结构 ,现将常用的几个变换矩阵结构指令列在表 2-3 中。

表 2-3 变换矩阵结构的常用指令

指令格式	功 能	指令格式	功 能
flipud (A)	输出矩阵 A 上下翻转后的矩阵	rot90 (A)	输出矩阵 A 逆时针旋转 90°后的矩阵
fliplr (A)	输出矩阵 A 左右翻转后的矩阵	reshape (A,m,n)	输出一个 $m \times n = k$ 阶矩阵 ,它是由矩阵 A 的 k 个元素重新排列构成的矩阵 ,重排前后各元素在矩阵中的序号不变
rot90 (A,k)	输出矩阵 A 沿逆时针方向旋转 k 个 90°的矩阵 k 为正负整数		

例 2-6 使 3 阶魔方矩阵左右翻转。

解 键入：

```
a=fliplr(magic(3))
```


回车得出：

```
a =
    6     1     8
    7     5     3
    2     9     4
```

例 2-7 将例 2-4 中的 2×3 阶矩阵 `as` 变成 3×2 阶矩阵。

解 键入：

```
reshape(as,3,2)
```

回车得出：

```
ans =
    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621
```

元素总数和各元素在矩阵中的序号(按先列后行排列的序号),变换前后不变。

4. 一些特殊向量(行矩阵)的创建

行(列)矩阵即向量,是矩阵的一个特例,它的创建方法和矩阵完全一样。但是,如果它的各元素取值有一定规律可循,这时可用下述的简便方法创建。

1) 等差数列型向量的创建

若向量各元素的取值构成一个等差数列,有下述两种简便的创建方法。

(1) 增量输入法。

输入格式为：

```
t = a:h:b 或 t = [a h b], t = (a h b)
```

① 输入参数 `a` 为初值 `b` 为终值 `h` 为公差(步长,增量),中间用冒号间隔。

② 公差 `h` 可正可负,省略时默认 `h = 1`。

③ 输出量 `t` 为一个等差数列: `a, a+h, a+2h, ..., b-h, b`, ($b^+ \leq b$ 且 $b-b^+ \leq h$)。

如键入：

```
t1 = 0 0.2 2.1
```

回车得出：

```
t1 = 0    0.2    0.4    0.6    0.8    1.0    1.2    1.4    1.6    1.8    2.0((2.1-2)
<0.2)
```

又如键入：

```
t2 = 1 8
```

回车得出：

```
t2 =
    1     2     3     4     5     6     7     8
```

(2) 指令输入法。

用线性等分指令 `linspace`,调用格式为：

$t = \text{linspace}(a, b, n)$

- ① 输入参数 a, b 分别为等差数列的初值和终值,即向量的第一和最末分量。
- ② 输入参数 n 是等分($b-a$)的节点数(含两端点),省略 n 时默认 $n=100$ 。
- ③ 输入参数间的逗号不能省略。
- ④ 输出参数 t 为一等差数列 $a, a + \frac{b-a}{n-1}, a+2\frac{b-a}{n-1}, \dots, b - \frac{b-a}{n-1}, b$ 。

如键入:

$t3 = \text{linspace}(1, 8.5, 8)$

回车得出:

$t3 =$

1.0000 2.0714 3.1429 4.2143 5.2857 6.3571 7.4286

8.5000

(3) 增量输入法与指令输入法的关系:

当 $(b-a)$ 可被 h 整除时,设 $t = (a:h:b) = \text{linspace}(a, b, n)$ 则 $h = (b-a)/(n-1)$ 或 $n = (b-a)/h + 1$ 。

2) 等比数列型向量的创建

向量各分量的取值构成等比数列时,可用对数等分指令 logspace 创建,调用格式为:

$q = \text{logspace}(\log_{10}(a), \log_{10}(b), n)$ 或 $q = \text{logspace}(as, bf, n)$

- ① 输入参数 a 为等比数列初值 b 为终值 n 为按等比数列划分时的节点数。
- ② 输出参数 q 是一个初值为 a 、终值为 b 的等比数列(注意,初终值不是 as 和 bf),共

有 n 项,公比为 $\left(\frac{b}{a}\right)^{\frac{1}{n-1}}$ 。

③ 省略 n 时 默认值 $n=50$ 。

④ $as = \log_{10}(a)$, $bf = \log_{10}(b)$, $\log_{10}(a)$ 是以 10 为底 a 的对数 $\lg a$ 在 MATLAB 中的表达式。

如键入:

$q = \text{logspace}(0, 1, 5)$ 或 $q = \text{logspace}(\log_{10}(1), \log_{10}(10), 5)$

回车得出:

$q =$

1.0000 1.7783 3.1623 5.6234 10.0000

后项与前项的比值均等于 $10/5.6234 = 5.6234/3.1623 = 3.1623/1.7783 = 1.7783 = \sqrt[4]{10}$ 。

当矩阵各元素取值之间构成等差或等比数列时,可以把上述特殊向量的创建方法与变换矩阵结构指令结合起来使用,这给某些特殊的元素较多矩阵输入带来很大方便。下面的例题就用了这种方法,它的元素并不多,只是为了说明运用这种方法的原理。

例 2-8 创建矩阵 $d = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 8 & 7 & 6 & 5 \end{bmatrix}$ 。

解 变换矩阵 d 中各元素的位置可以构成一个等差数列,反过来可以先输入一个等差数列向量,通过改变结构达到要求。据此原理键入:

$d = \text{rot90}(\text{reshape}(1:8, 4, 2), -1)$ 或 $d = \text{rot90}(\text{reshape}(1:8, 4, 2),$

3)

回车得出：

d =

4	3	2	1
8	7	6	5

5. 创建 n 个分量都等于 1 的向量

若键入 $g1 = \text{linspace}(1, 1, n)$, $\text{logspace}(0, 0, n)$ 或 $\text{ones}(1, n)$, 均可创建一个各项数值都是 1 的 n 维向量, 这在绘图、编程和某些特殊计算中经常用到。

6. 常数变量的赋值

一个常数可以看作一行一列的矩阵, 也可以看作是点向量, 它的赋值可以选用下述任何一种方法。假设给变量 $a2, b2, c2$ 分别赋以数值 4 3 6, 则可键入：

$$a2 = 4, b2 = (3), c2 = [6]$$

回车得出：

$$a2 = 4, b2 = 3, c2 = 6$$

2.1.3 数值矩阵元素的标识与修改

用一个符号代表已输入数值矩阵的单个或部分元素, 称为矩阵元素的标识。元素的基本标识方法如下所述。

1. 矩阵元素的标识方法

(1) 用符号 $a(p)$ 标识已输入矩阵 a 中序号为 p (正整数) 的元素。矩阵中元素的序号是把所有元素按“先列后行”排列时的顺序号。如 3×3 矩阵元素的序号排列方法为：

1	4	7
2	5	8
3	6	9

用一个冒号代替 p 时, 即 $a(:)$ 表示矩阵 a 的所有元素按上述规定顺序排成的列阵；

(2) 符号 $a(m, n)$ 表示矩阵 a 中第 m 行、第 n 列的一个元素 (m 和 n 均为正整数) 则：

① 若用冒号“:”代替 $a(m, n)$ 中的行号 m (或列号 n) 时, 表示 a 的第 n 列 (或第 m 行) 所有元素构成的列阵 (或行阵)；

② 若把 $a(m, n)$ 中的 m 换成一个行阵, 如让 $m = [p, q, r]$ (p, q, r 均为正整数) 即 $a(m, n) = a([p, q, r], n)$ 表示矩阵 a 的第 p, q, r 行、第 n 列元素构成的列阵；

③ 若把 $a(m, n)$ 中的 n 换成一个行阵, 如让 $n = [p, q, r]$ (p, q, r 均为正整数) 即 $a(m, n) = a(m, [p, q, r])$ 表示矩阵 a 的第 m 行、第 p, q, r 列元素构成的行阵；

④ 若把 $a(m, n)$ 中的 m (或 n) 换成“ $p:q$ ” ($p \leq q$) 即 $a(p:q, n)$ (或 $a(m, p:q)$) 表示矩阵 a 的第 p 到 q 行 (或列)、第 n 列 (或第 m 行) 的所有元素构成的列阵 (或行阵)；

⑤ 若把 $a(m, n)$ 中的 m 和 n 都换成行阵, 即 $a(m, n) = a([p, q, r], [w, s])$ (w, s 为正整数) 表示由矩阵 a 的第 p, q, r 行与第 w, s 列交点上元素构成的矩阵;

(3) 对于三维矩阵 A (相当于若干个平面矩阵一页一页的叠合, 每一页表示一个平面矩阵), 用 $A(:, :, n)$ 表示第 n 页的平面矩阵。

例 2-9 $a = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \\ 2i & 3-i & 9 \end{bmatrix}$, 下列变量各表示什么样的矩阵: $d1 = a(:, 3)$, $d2 = a(1:2, :)$ 和 $d3 = a([1, 3], [2, 3])$ 。

解 键入:

$a = [1 \ 2 \ 3; 5 \ 4 \ 1; 2i \ 3-i \ 9]; d1 = a(:, 3), d2 = a(1:2, :), d3 = a([1, 3], [2, 3])$

回车得出:

$d1 =$

3

1

9

$d2 =$

1 2 3

5 4 1

$d3 =$

2.0000 3.0000

3.0000 - 1.0000i 9.0000

例 2-10 已知 $A = \text{reshape}(1:8, 5, 2, 4)$ 和 $B = \begin{bmatrix} 1 & 3 & 5 & 7 & 3 \\ 2 & 4 & 6 & 8 & 2 \\ 7 & 5 & 3 & 1 & 1 \\ 8 & 6 & 4 & 2 & 4 \end{bmatrix}$, 如何用 A 组合

成 B ?

解 键入:

$A = \text{reshape}(1:8, 5, 2, 4)$

回车得出:

$A =$

1 3 5 7

2 4 6 8

比较 A 和 B 可知 B 的 1 到 4 列前两行正好等于 A , 而后两行是把 A 左右翻转得到的矩阵。 B 的最后一列可以取自 A 的不同元素。据此分析可知, 若键入:

$B = [A, [A(3); A(2)]; \text{flip}(\text{r}(A), [A(1); A(4)])]$

回车即可得到题中的 B 。

2. 数值矩阵的增删修改

要想对已输入矩阵的部分或某个元素进行修改, 首先将欲修改部分加以标识, 然后再对被标识部分重新赋值。所以, 矩阵的增删修改, 实际上就是对被修改部分元素的标识和重新赋值。

例 2-11 把例 2-9 中矩阵 a 增加一个第 4 行, 其数值为 3 6 9。

解 键入:

```
a(4,:) = [3 6 9] 或 a = [a; [3 6 9]]
```

回车得出增加了第 4 行后的矩阵 a

```
a =
    1.0000    2.0000    3.0000
    5.0000    4.0000    1.0000
    0 + 2.0000i    3.0000 - 1.0000i    9.0000
    3.0000    6.0000    9.0000
```

例 2-12 使例 2-11 中矩阵 a 的第二行元素全部消失。

解 键入:

```
a(2,:) = []
```

回车得出(注: 空矩阵不是零矩阵):

```
a =
    1.0000    2.0000    3.0000
    0 + 2.0000i    3.0000 - 1.0000i    9.0000
    3.0000    6.0000    9.0000
```

例 2-13 把例 2-12 中矩阵 a 的第二列改为 2 5 9, 并增加一个第 4 列 0 5 7。

解 键入:

```
a(:,2) = [2;5;9]; a(:,4) = [0;5;7] (或 a(:,2) = [2;5;9]; a = [a[0;5;7]])
```

回车得出:

```
a =
    1.0000    2.0000    3.0000    0
    0 + 2.0000i    5.0000    9.0000    5.0000
    3.0000    9.0000    9.0000    7.0000
```

2.1.4 数值矩阵的矩阵算法

MATLAB 对数值矩阵提供了两种不同的运算方法: 矩阵算法和数组算法。

所谓“矩阵算法”就是把每一个矩阵都看作一个整体, 各种运算完全按照线性代数中的矩阵运算法则进行, 运算的书写形式和运算符号都与矩阵理论中的几乎完全一样。

所谓“数组算法”就是把每一个矩阵看作由其元素构成的一组数据(数组), 运算是在参与运算矩阵的对应元素之间进行的数与数的运算。这种算法方便于对大批数据的处理和一次求出多个函数值。这里只介绍数值矩阵的“矩阵算法”, 在 2.1.5 节中再介绍数值矩阵的“数组算

法”。

1. 数值矩阵维数的查验和矩阵的转置

1) 查验矩阵维数指令

进行矩阵运算之前,经常要考虑矩阵的维数,所以应知道如何用指令查验矩阵的维数。查验矩阵维数指令 `size` 的使用格式为:

`size(a)` 或 `size(a,r)`

- ① 输入参数 `a` 为待查的已输入矩阵或向量。
- ② 输入参数 `r` 可取 1 或 2。当 `r=1` 时输出 `a` 的行数;当 `r=2` 时输出矩阵 `a` 的列数。
- ③ 省略输入参数 `r`、仅有参数 `a` 时,输出二维向量 `(m,n)`,表示矩阵 `a` 是 $m \times n$ 维。

例如,欲查验例 2-10 中矩阵 `B` 的维数,可在指令窗中键入:

`size(B)`

回车得出:

```
ans =
     4     5
```

结果表明 `B` 是一个 4 行 5 列的矩阵。

若键入 `size(B(3,3))`,回车得出:

```
1     1
```

表明 `B(3,3)` 是一个 1 行 1 列矩阵,即一个数。

2) 求矩阵共轭转置的指令

- ① 使矩阵共轭转置的指令是“`'`”,使用格式为 `A'`,输出矩阵 `A` 的共轭转置矩阵 \bar{A}^T 。
- ② 使矩阵变成其共轭矩阵的指令是 `conj`,使用格式为 `conj(A)`,输出 `A` 的共轭矩阵 \bar{A} 。
- ③ 利用上述两个指令使矩阵仅发生转置的方法为:
 - 如果 `A` 为实数矩阵,用“`'`”就得出它的转置矩阵,即用 `A'` 就得出 A^T ;
 - 如果 `A` 为复数矩阵,用 `conj(A')` 或 `conj(A)'` 可得出 A^T 。

例 2-14 已知 $a = \begin{bmatrix} 1 & i & 3 \\ 9i & 2-i & 8 \\ 7 & 4 & 8+i \end{bmatrix}$,求 `a` 的共轭矩阵和共轭转置矩阵。

解 令 $a1 = \bar{a}$, $a2 = \bar{a}^T$,则在指令窗中键入:

`a=[1 i 3;9i 2-i 8;7 4 8+i]; a1=conj(a), a2=a'`

回车得出:

```
a1 =
    1.0000          0 - 1.0000i    3.0000
    0 - 9.0000i    2.0000 + 1.0000i    8.0000
    7.0000         4.0000         8.0000 - 1.0000i

a2 =
    1.0000          0 - 9.0000i    7.0000
    0 - 1.0000i    2.0000 + 1.0000i    4.0000
```

3.0000

8.0000

8.0000 - 1.0000i

2. 矩阵算法中的矩阵加、减和乘法运算

① 矩阵进行加、减、乘运算时,其运算符分别为“+”、“-”和“*”,用这些符号把参与运算的矩阵连接起来,回车便可得出计算结果。

② 矩阵进行加、减运算时,它们的维数必须相同,即行数、列数分别相等。

③ 两个矩阵相乘时,它们的内维数必须相等,即左矩阵的列数必须等于右矩阵的行数,用 $a * b$ 或指令 $mtimes(a, b)$ 。

④ 进行方阵 a 的 n 次幂 a^n 运算时,输入 a^n 或 $mpower(a, n)$,这时,若整数 $n > 0$,输出方阵等于 n 个 a 相乘的结果;若整数 $n < 0$,输出的方阵等于 n 个 a 连乘后的逆阵;若 n 是非整数,其数学意义属“矩阵分析”的范畴。

⑤ 矩阵与一个常数进行加、减运算在线性代数中是没有意义的,但在 MATLAB 中赋予如下规定:若 a 为矩阵, d 为常数,作 $a \pm d$ 的运算意义为

$$a \pm d = a \pm d * \text{ones}(\text{size}(a)),$$

即相当于矩阵 a 的每个元素都与数 d 进行一次加、减运算。

例 2-15 已知矩阵 $a = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \\ 2 & 5 & 9 \end{bmatrix}$, $b = \begin{bmatrix} 3 & 5 \\ 6 & 1 \\ 2 & 9 \end{bmatrix}$, $c = \begin{bmatrix} 6 & 1 & 2 \\ 5 & 2 & 8 \\ 4 & 7 & 1 \end{bmatrix}$, 求: $a + c, ab, ac, ca$

解 键入:

```
a=[1 2 3;5 4 1;2 5 9]; b=[3 5;6 1;2 9]; c=[6 1 2;5 2 8;4 7 1];
a1=a+c, a2=a*b, a3=a*c, a4=c*a
```

回车得出:

a1 =

```
7      3      5
10     6      9
6     12     10
```

a2 =

```
21     34
41     38
54     96
```

a3 =

```
28     26     21
54     20     43
73     75     53
```

a4 =

```
15     26     37
31     58     89
41     41     28
```

注 $a * c \neq c * a$ 矩阵乘积一般不遵从交换律。

例 2-16 已知 $a = \begin{bmatrix} 1 & 1-2i & 3 \\ 4+5i & 4 & 1 \\ 2 & 5 & 9i \end{bmatrix}$, 计算方阵 a 的幂乘: a^2 和 a^{-3} 。

解 键入:

```
a=[1 1-2i 3;4+5i 4 1;2 5 9i]; a5=a^2, a6=a^(-3)
```

回车得出:

```
a5 =
```

```
21.0000 - 3.0000 i    20.0000 - 10.0000 i    4.0000 + 25.0000 i
22.0000    +    25.0000 i    35.0000    -    3.0000 i    16.0000
```

```
+24.0000 i
```

```
22.0000 + 43.0000 i    22.0000 + 41.0000 i - 70.0000
```

```
a6 =
```

```
- 14.1688 + 1.0140 i    2.7604 - 1.6471 i    0.0249 - 4.4244 i
15.9971 + 14.1263 i    - 4.6676 - 1.2293 i    - 4.7685 + 4.6284 i
- 8.3077 + 5.4108 i    1.1303 - 1.9393 i    - 1.4797 - 2.7074 i
```

虽然 a^{-3} 和 $a^(-3)$ 等价, 但是为了查看方便, 习惯于写成 $a^(-3)$, 它相当于 $(a^* a^* a)^{-1}$ 。

例 2-17 已知 $d=5$, $a = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \\ 2 & 5 & 9 \end{bmatrix}$, 计算 $a - d$ (在线性代数中无意义)。

解 键入:

```
d=5; a=[1 2 3;5 4 1;2 5 9]; a-d
```

回车得出:

```
ans =
```

```
- 4    - 3    - 2
  0    - 1    - 4
- 3     0     4
```

3. 数值矩阵的求逆及矩阵算法中的除法

MATLAB 的矩阵算法中定义了两种除法: 左除和右除, 它们都是由逆阵引导出来的。

1) 求方阵逆阵的指令 inv

同维数方阵 a 、 b 满足 $ab = ba = E$ (E 为与 a 同维的单位阵), 则称 a 、 b 互为逆阵, 可记为 $b = a^{-1}$ 或 $a = b^{-1}$ 。在 MATLAB 中求方阵 a 逆阵指令 inv 的调用格式为 $inv(a)$ 。

① 输入的 a 必须是方阵;

② 输出 a 的逆阵 a^{-1} 。

2) 求矩阵伪逆阵的指令 pinv

对于奇异阵或长方阵 b , 把同时满足 $xbx = b$ 和 $bxb = x$ (Penrose 第一、二方程) 的矩阵 x 叫作 b 的伪逆阵, 记为 b^+ 或 $b\{1, 2\}$, MATLAB 中求伪逆阵 x 的指令为 $pinv(b)$ 。

3) 左除

解矩阵方程 $ax=b$ 可得 $x=a^{-1}b$ 在 MATLAB 中记为：

$$x = \text{inv}(a) * b \text{ 或 } x = a \backslash b \text{ 或 } \text{mldivide}(a, b)$$

即用 a^{-1} 左乘矩阵 b , 记为 $a \backslash b$ 称为 a 左除 b 。

4) 右除

解矩阵方程 $xa=b$ 可得 $x=ba^{-1}$ 在 MATLAB 中记为：

$$x = b * \text{inv}(a) \text{ 或 } x = b / a \text{ 或 } \text{mrdivide}(b, a)$$

即用 a^{-1} 右乘矩阵 b , 记为 b/a 称为 a 右除 b 。

虽然矩阵的左除和右除都可由矩阵求逆推出算法,但是考虑到计算精度和运算速度, MATLAB 中除法的实际计算并不用求逆阵后的相乘得出,而是通过矩阵的三角分解法求得。如计算 $x=a \backslash b$ 时,先用 $\text{l u}(a)$ 将 b 分解成两个三角阵 L 和 U 的乘积 $a=LU$,再求出 $y=L \backslash b$ 通过 $x=U \backslash y$ 求得 x 。

例 2-18 已知 $a = \begin{bmatrix} 5 & 1+2i & 8 \\ 4-5i & 6 & 1 \\ 3 & 5 & 9i \end{bmatrix}$, 求 a 的逆阵 $g=a^{-1}$ 。

解 键入：

$$a = [5 \ 1+2i \ 8; 4-5i \ 6 \ 1; 3 \ 5 \ 9i]; \quad g = \text{inv}(a)$$

回车得出：

$$g = \begin{bmatrix} -0.9538 + 0.3006i & 0.5672 + 0.9218i & -0.3696 - 0.7848i \\ 0.3121 - 0.9711i & -0.9364 - 0.0867i & 0.8728 + 0.1734i \\ 0.4393 - 0.1445i & -0.2591 - 0.3312i & 0.1652 + 0.2506i \end{bmatrix}$$

g 是 a 的逆阵,可作验证如下。

键入：

$$g * a \text{ 或 } a * g$$

回车得到的都是 3 阶单位阵。

例 2-19 已知 $a = \begin{bmatrix} -4 & -3 & -2 \\ 0 & -1 & -4 \\ -3 & 0 & 4 \end{bmatrix}$ 和 $b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, 分别求解矩阵方程 $ax=b$ 和 $ya=b^T$ 中的

x, y

解 由题设知 $x=a^{-1}b, y=b^T a^{-1}$, 键入：

$$b = [1 \ 2 \ 3]^T; \quad a = [-4 \ -3 \ -2; 0 \ -1 \ -4; -3 \ 0 \ 4]; \quad x = a \backslash b \quad (\text{或 } \text{mldivide}(a, b))$$

回车得出：

$$x = \begin{bmatrix} -3.5714 \\ 5.7143 \\ -1.9286 \end{bmatrix}$$

键入：

$y = b' / a$ 或 `mrdivide(b',a)`

回车得出：

$y =$
 $\begin{matrix} & -0.7857 & 0.3571 & 0.7143 \end{matrix}$

4. 矩阵函数

定义域和值域都属于方阵的函数称为矩阵函数 ,它有多种定义方法。当用方阵幂级数的和函数来定义矩阵函数时 ,方阵函数 $f(a) = \sum_{k=0} C_k a^k$,其中自变量 a 和函数值 $f(a)$ 都是 n 阶方阵 C_k 是常系数。表 2-4 列出了 MATLAB 中常用的矩阵函数求值指令。

表 2-4 常用矩阵函数求值指令

指令名	数 学 意 义	指令名	数 学 意 义
<code>expm(a)</code>	$e^a = 1 + \frac{a}{1!} + \frac{a^2}{2!} + \frac{a^3}{3!} + \dots$	<code>sqrtn(a)</code>	矩阵 a 的平方根
<code>logm(a)</code>	$\ln(a)$	<code>funm(a,@f)</code>	矩阵 a 的任意函数 $f(a)$

注 `funm(a,@f)` 中的 f 为任意矩阵函数 要把函数 f 用英文格式单引号界定或在 f 前加符号 `@`。

例 2-20 设方阵 $a = \begin{bmatrix} 4 & 1 & 7 \\ 4 & 7 & 9 \\ 1 & 5 & 3 \end{bmatrix}$,求 $\sin(a)$ 。

解 根据定义 $\sin(a) = \sum_{k=0} \frac{(-1)^k a^{2k+1}}{(2k+1)!} = \begin{bmatrix} 4 & 1 & 7 \\ 4 & 7 & 9 \\ 1 & 5 & 3 \end{bmatrix} - \frac{1}{3!} \begin{bmatrix} 4 & 1 & 7 \\ 4 & 7 & 9 \\ 1 & 5 & 3 \end{bmatrix}^3 + \frac{1}{5!} \begin{bmatrix} 4 & 1 & 7 \\ 4 & 7 & 9 \\ 1 & 5 & 3 \end{bmatrix}^5 + \dots$

在 MATLAB 中键入：

`a=[4 1 7;4 7 9;1 5 3]; sina=funm(a,@sin) 或 funm(a,'sin')`

回车即可求出：

$sina =$
 $\begin{matrix} 1.9448 & -1.9449 & 2.7397 \\ 0.3355 & 0.2606 & 0.8323 \\ -0.7679 & 1.3640 & -1.0660 \end{matrix}$

2.1.5 数值矩阵的数组算法

MATLAB 中的基本数据单位是复数矩阵 ,矩阵间的数组算法规定把复数矩阵看作一组数据 ,在两个矩阵的对应元素之间逐一按相联符号进行数值运算。定义数组算法方便于处理大批数据 ,能够一次算出多个函数值 ,也有利于数据作图。

1. 查验向量维数的指令

数据处理和绘图中经常用到两个向量对应元素间的运算 ,这就需要知道参与运算向量的

维数。下面介绍查验向量维数指令 `length` ,它的调用格式为 :

`length (a)`

- ① 输入参数 `a` 为向量时 ,则输出向量 `a` 的维数。
- ② 输入参数 `a` 为列阵(或行阵)时 ,输出 `a` 的列(或行)数。
- ③ 输入参数 `a` 是 $m \times n$ 阶矩阵时 ,则输出行数和列数的最大值 ,即 :
 $\max(m, n) = \max(\text{size}(a))$ 。

例 2-21 求向量 `t=0 0.2 2.1` 的维数。

解 键入 :

`t=0:0.2:2.1; length(t)`

回车得出 :

`ans =`

`11`

也可以键入 `size(t)` 来查验 ,回车可知 `t` 是 1×11 维矩阵。

2. 数值矩阵间数组算法的四则运算

(1) 矩阵之间进行数组运算时 ,加、减、乘、除运算定义为参与运算矩阵的对应元素之间的四则运算。因此 ,参与数组运算的矩阵维数必须严格相同 ,运算结果是一个与参与运算矩阵维数相同的矩阵。

(2) 数组矩阵的“矩阵加减”和“数组加减”定义完全相同 ,所以运算符无须区分。而数组算法的乘除运算与矩阵算法定义不同 ,使用的符号也不同 ,数组算法的乘除符号是在矩阵算法乘除符号前面加一个小黑点(英文句号)。

设 `a`、`b` 为两个同维矩阵 ,`s` 为常数 ,数组算法的乘除运算符和意义列在表 2-5 中。

表 2-5 数组算法的四则运算符、格式及意义

数组算法格式	算 法 意 义	数组算法格式	算 法 意 义
<code>a.*b</code>	<code>a</code> 与 <code>b</code> 的对应元素相乘	<code>a./b</code>	<code>a</code> 中各元素除以 <code>b</code> 中对应的元素
<code>a.^n</code>	<code>a</code> 中每个元素的 n 次方	<code>b.\a</code>	
<code>a./s, s.\a</code>	<code>a</code> 中各元素被 <code>s</code> 除	<code>s./a, a.\s</code>	<code>s</code> 被 <code>a</code> 中各元素除

例 2-22 已知 $a = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \\ 2 & 5 & 9 \end{bmatrix}$, $c = \begin{bmatrix} 6 & 1 & 2 \\ 5 & 2 & 8 \\ 4 & 7 & 1 \end{bmatrix}$,用数组算法计算 `a.*c`、`a.*c.^3` 和 `a.^3`

3。

解 键入 :

`a=[1 2 3;5 4 1;2 5 9];c=[6 1 2;5 2 8;4 7 1];ac=a.*c (或 c.*a)`

回车得出 :

`ac =`

6 2 6
25 8 8
8 35 9

键入：

```
ac1 = a * c
```

回车得出：

```
ac1 =
    28    26    21
    54    20    43
    73    75    53
```

键入：

```
a3 = a.^3, a4 = a^3
```

回车得出：

```
a3 =
     1     8    27
   125    64     1
     8   125   729

a4 =
   206    294    364
   238    318    364
   574    826   1032
```

例 2-23 已知矩阵 $a = \begin{bmatrix} 1 & 8 & 27 \\ 125 & 64 & 1 \\ 8 & 125 & 216 \end{bmatrix}$ 求 a 的平方及 a 中每个元素的立方根。

解 键入：

```
a = [1 8 27; 125 64 1; 8 125 216]; a^2
```

回车得出矩阵 a 的平方：

```
ans =
   1217    3895    5867
   8133    5221    3655
  17361   35064   46997
```

键入：

```
a.^(1/3)
```

回车得出矩阵 a 中每个元素的立方根：

```
ans =
   1.0000    2.0000    3.0000
   5.0000    4.0000    1.0000
   2.0000    5.0000    6.0000
```

3. 数值矩阵间数组乘法和矩阵乘法的差异

数值矩阵间的“数组算法”是 MATLAB 中特设的，初学者容易和“矩阵算法”混淆，它们间的差异集中表现在乘除运算里，现举例说明如下。

$$\text{设 } a = [a_1 \quad a_2 \quad a_3] \quad b = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \\ b_5 & b_6 \end{bmatrix} \quad c = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix} \text{ 和 } d = \begin{bmatrix} d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 \\ d_7 & d_8 & d_9 \end{bmatrix}, \text{MAT-}$$

LAB 运算中：

① $a * b = [a_1 b_1 + a_2 b_2 + a_3 b_3 \quad a_1 b_2 + a_2 b_4 + a_3 b_6]$,也可进行 $a * c, a * d, c * b, c * d$ 运算 ,但不能进行 $b * a, c * a$ 和 $d * a$ 等运算 ,因为矩阵乘法运算要求两矩阵内维数必须相等；

② 可以进行 $c * d$ 或 $d * c$ 运算 ,但 $c * d \neq d * c$,因为矩阵乘法不遵从交换律；

③ 可以进行 $c . * d$ 或 $d . * c$ 运算 ,但不能进行 $a . * b, a . * c$ 和 $a . * d$ 等运算 ,因为矩阵的数组乘法要求两矩阵维数必须完全相同；

④ 可以做 $c . * d$ 和 $d . * c$ 运算 ,且 $c . * d = d . * c$,矩阵的数组乘法是遵从交换律的；

⑤ $a.^3 = [a_1^3 \quad a_2^3 \quad a_3^3]$,也可以进行 $b.^3, c.^3, d.^3, c.^3, d.^3$ 运算。但 $c.^3 \neq c.^3, d.^3 \neq d.^3$,不能进行 $a.^3 \quad b.^3$ 运算 ,因为矩阵乘方要求矩阵必须是方阵。

4. 数组算法中的基本初等函数运算

由于数组算法是在矩阵的对应元素之间进行的数与数的运算 ,所以数组算法的函数运算也就是对每个元素的函数运算。设 $X = \{x_p\}$ 是一个数值矩阵 , $f(\quad)$ 表示任意函数 ,则函数 $Y = \{y_p\} = f(X)$ 就是一个与 X 同维的矩阵 ,它的元素 $y_p = f(x_p)$ 。

表 2-6 中列出了部分基本初等函数的求值指令 ,其中 x, x_1 和 x_2 均为数值矩阵 , m 和 n 为数值。

表 2-6 常用数组算法中的基本初等函数表

函数指令	数学意义	函数指令	数学意义	函数指令	数学意义
$\sin(x)$	$\sin x$	$\cot(x)$	$\cot(x)$	$\text{sign}(x)$	数 x 的正负号
$\text{asin}(x)$	$\arcsin x$	$\sinh(x)$	双曲正弦 $\text{sh } x$	$\text{round}(x)$	四舍五入取整
$\cos(x)$	$\cos x$	$\cosh(x)$	双曲余弦 $\text{ch } x$	$\text{fix}(x)$	输出靠近零的整数
$\sec(x)$	$\sec x$	$\text{real}(x)$	x 的实部	$\text{floor}(x)$	输出靠向 - 的整数
$\tan(x)$	$\tan x$	$\text{imag}(x)$	x 的虚部	$\text{ceil}(x)$	输出靠向 的整数
$\text{atan}(x)$	$\arctan x$	$\text{abs}(x)$	x 绝对值或模	$\text{rem}(x_1, x_2)$ 或 $\text{mod}(x_1, x_2)$	输出矩阵 x_1 各元素除以 x_2 (x_2 为数或与 x_1 同维矩阵)对应元素的余数
$\exp(x)$	e^x	$\text{angle}(x)$	复数 x 的相角	$\text{median}(x)$	输出 x 各列元素的中间值
$\log(x)$	$\ln x$	$\text{pow2}(x)$	2^x	$\text{prod}(m:n)$	$m(m+1) \dots (n-1)n, (m < n)$
$\log_2(x)$	$\log_2 x$	$\text{max}(x)$	x 各列元素中最大值	$\text{factorial}(x)$	$x!$, x 可为非负整数矩阵, 这时输出各元素的阶乘
$\log_{10}(x)$	$\log_{10} x$	$\text{mean}(x)$	x 各列元素之平均值		
$\text{sqrt}(x)$	\sqrt{x}	$\text{sum}(x)$	x 各列元素之和		

利用数组算法的性质可以将多个自变量作成一个矩阵或向量 X ,利用 $Y = f(X)$ 的函数关系一次就能求出多个自变量(X 的元素)的函数值 ,如果 X 是 $m \times n$ 阶矩阵 , $Y = f(X)$ 一次则可求出 $m \times n$ 个自变量的函数值 $y_p = f(x_p) (p = 1, 2, \dots, m \times n)$ 。

基本初等函数用四则运算组合起来就可以构成任意初等函数 ,用 MATLAB 对这种初等函

数求值的方法将在“2.2 字符串和符号矩阵”节中介绍。

例 2-24 分别求出当 $x=1\ 2\ 3\ 5\ 4\ 9$ 时 \sqrt{x} 和 2^x 的值。

解 键入：

```
b=[1 2 3 5 4 9]; sqrt(b)
```

回车得出 \sqrt{x} 的值：

```
ans =  
1.0000    1.4142    1.7321    2.2361    2.0000    3.0000
```

键入：

```
pow2(b)
```

回车得出 2^x 的值：

```
ans =  
2     4     8    32    16   512
```

例 2-25 求 C_{10}^5 , C_8^3 , C_7^2 和 C_6^4 , C_m^n 表示 m 个元素中取 n 个的组合。

解 由于 $C_m^n = \frac{A_m^n}{P_n} = \frac{m(m-1)(m-2)\dots[m-(n-1)]}{1 \times 2 \times 3 \times \dots \times n}$, (A_m^n 为 m 个元素中取 n 个的排

列 P_n 为 n 个元素的全排列) 所以在指令窗中键入：

```
C1=prod(10-(5-1):10)./factorial(5)
```

回车得出：

```
C1 =  
252,
```

同理键入：

```
C2=prod(8-(3-1):8)./factorial(3)...,
```

可分别算出：

```
C2=56 ,C3=21 ,C4=15
```

又因为 $C_m^n = C_m^{m-n} = \frac{m!}{n!(m-n)!}$, 可以用下述指令一次求出 4 个组合的值。

键入：

```
a=[5 3 2 4];b=[10 8 7 6];C=factorial(b)./(factorial(a).*factorial(b-a))
```

回车得出：

```
C =  
252    56    21    15
```

表明 $C_{10}^5=252$, $C_8^3=56$, $C_7^2=21$, $C_6^4=15$ 。

也可键入 $C=factorial(b)./(factorial(a).factorial(b-a))$, 一次求出 4 个组合的值。

5. 向量的点积和叉积

在向量运算中,经常要计算两个向量的点积(数量积)和叉积(向量积)。MATLAB 中对这两种运算设有专用指令。

1) 求向量数量积指令

计算向量的数量积(点积)指令 `dot` 的使用格式为：

`dot(a,b)`

- ① 输入参数 a b 必须是两个维数相等的向量；
- ② 输出向量 a b 的内积(数量积 线性代数中记为 $[a, b]$)是一个数；
- ③ 输入参量 a b 均为三维向量时, 输出向量 a b 的数量积, 即点积 $a \cdot b$ 。

2) 求向量的向量积的指令

计算向量的向量积(叉积)的指令 `cross` 的使用格式为：

`c=cross(a,b)`

① a 和 b 是三维向量时, 输出 c 是一个三维向量, 等于 a 和 b 的向量积, 即叉积 $c = a \times b$ 。

② a b 是同维数矩阵, 且行或列必有一项是三维时, 输出 c 为一个矩阵, 它是把矩阵 a , b 视为多个三维向量, 分别计算其向量积的结果。

例 2-26 已知向量 $a = 0:0.2:3$, 向量 b 起于 0 终于 5, 被等分的分量数与向量 a 相同。求 a 与 b 的内积 $a \cdot b$ 。

解 键入：

```
a=0:0.2:3; b=linspace(0,5,length(a)); dot(a,b) (或 a*b')
ans =
    82.6667
```

例 2-27 已知 $a = 3.2i + 5.1j + 6k$, $b = 7.1i + 4j + 2.4k$, $c = 3i + 5j + 6k$, $d = 8i + 7j + 4k$, 求向量积 $a \times b$ 和 $c \times d$ 。

解 可以键入：

`cross(a,b)` 和 `cross(c,d)` 分两次求出, 但也可以用下述方法一次求出。

键入：

```
a=[3.2 5.1 6]; b=[7.1 4 2.4]; c=[3 5 6]; d=[8 7 4]; A=[a;c]; B=[b;d]; cross(A,B)
```

回车得出：

```
ans =
    -11.7600    34.9200   -23.4100
    -22.0000    36.0000   -19.0000
```

即 $a \times b = -11.7600i + 34.9200j - 23.4100k$; $c \times d = -22.0000i + 36.0000j - 19.0000k$

练习题 2.1

1. 已知 $a = \begin{bmatrix} 5 & 12 & 47 \\ 13 & 41 & 2 \\ 9 & 6 & 71 \end{bmatrix}$, $a_2 = \begin{bmatrix} 12 & 9 \\ 6 & 15 \\ 7 & 21 \end{bmatrix}$, 在 MATLAB 中计算 $a_1 * a_2$, $a_1(:, 1:2)$

$* a_2$, a_1^2 和 $a_1(:).^2$ 。

2. 对题 1 中的矩阵 a 和 a_2 , 可否在 MATLAB 中进行下述运算并说明原因 $a_1(1,:)^2$ 、

$a1(1,:).^2$ 、 $a2^2$ 和 $a2.^2$ 。

3. 将 $a2$ 左旋 90 度作为 $a1$ 的第 4、5 行构成 $b1$ ，再把 $b1$ 的第 4 行第 1 列的元素和第 5 行第 3 列的元素，分别赋给 $c1$ 和 $c2$ 。求 $b1$, $c1$, $c2$ 及 $b1(3:4,:) * a2$ 。

4. 计算 $a1 + 5a1' - E$ 和 $a1^3 - \text{rot90}(a1)^2 + 6E$, E 是与 $a1$ 同维的单位方阵。

5. 利用特殊向量创建方法和变换矩阵结构指令输入矩阵 $A = \begin{bmatrix} 25 & 17 & 9 & 1 \\ 27 & 19 & 11 & 3 \\ 29 & 21 & 13 & 5 \\ 31 & 23 & 15 & 7 \end{bmatrix}$ 。

6. 设 $s=5$ ，计算 $s - a1$, $s * a1$, $s ./ a1$ 和 $a1 ./ s$ ，比较它们的差异。

7. 从下式的计算中能否找出规律？ $a1/a2$, $a1 \setminus a2$, $a1/a2'$ 和 $a2 \setminus a1$, $a2/a1$, $a2'/a1$

8. 已知矩阵 $b = \begin{bmatrix} 2 & 1 & 4 & -2 & -1 & -4 \\ 3 & 0 & -1 & -3 & 0 & 1 \\ 2 & 3 & 4 & -2 & -3 & -4 \\ 4 & 2 & 8 & 6 & 3 & 12 \\ 6 & 0 & -2 & 9 & 0 & -3 \\ 4 & 6 & 8 & 6 & 9 & 12 \end{bmatrix}$ ，在 MATLAB 中求 b^{-1} , b^5 , $b^T b$ 。

9. 已知 $c = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$ ，在 MATLAB 中求 c^{-4} , $(c^3)^{-1}$, $(3c + 5c^{-1})/5$ 。

10. 已知 $a = \begin{bmatrix} 1 & i & 3 \\ 9i & 2-i & 8 \\ 7 & 4 & 8+i \end{bmatrix}$ ，求 a 的转置矩阵。

11. 已知 $a = \begin{bmatrix} 1 & i & 3 \\ 9i & 2-i & 8 \\ 7 & 4 & 8+i \end{bmatrix}$, $b = \begin{bmatrix} 2 & 5 \\ 3 & 1 \\ 7 & 9 \end{bmatrix}$ ，可否在 MATLAB 中进行下述运算： $a * b$ 、

$b * a$ 、 $a ./ b$ 、 $a.^3$ 、 $a.^3 ./ b$ 和 $b.^2$ ？为什么？

12. 已知 $abc = \begin{bmatrix} -2.57 & 8.87 \\ -0.57 & 3.2-5.5i \end{bmatrix}$ ，用取值函数求出 $m1 = \text{sign}(abc)$, $m2 = \text{round}(abc)$, $m3 = \text{floor}(abc)$ 和 $m4 = \text{imag}(abc)$ ，理解其数学意义。

13. 已知 $x = [1 \ 4 \ 3 \ 2 \ 0 \ 8 \ 10 \ 5]^T$, $y = [8 \ 0 \ 0 \ 4 \ 2 \ 1 \ 9 \ 11]^T$ ，求它们的内积 $[x, y]$ (可以用矩阵积和点积指令两种方法)。

14. 已知 $a = 3.82 i + 5.71 j + 9.62 k$, $b = 7.31 i + 6.42 j + 2.48 k$ ，求数量积 $a \cdot b$ 和向量积 $a \times b$ 。

2.2 字符串和符号矩阵

在数学、物理、电路、力学等许多基础学科和工程应用中，常常要进行公式的推演变换、代

数方程的符号解、微积分的解析解等,处理这类问题就属于符号运算。符号运算是对方程或代数式中的数学符号进行的运算操作,运算结果是数学表达式而不是数值。例如高等数学中有求函数的导函数和求函数导数在某一点取值的问题,前者属于符号运算,后者属于数值计算。MATLAB 开发了符号运算工具包(Symbolic Math Toolbox),可以进行代数式的因式分解、展开和化简,函数的幂级数展开,求微积分及微分方程组的解析解等多种运算。

另外,MATLAB 中的符号运算是按有理数的运算方式进行的,如对未知量 x 、有理数(如 $1/3$)或超越数(如 π)等的运算都没有有效数字位数的限制,也没有舍入误差,所以相当精确,这种精确结果可以转换成任意精度的数值解,这对数值计算来说,是一个难能可贵的优点。

2.2.1 字符串变量和函数求值

字符串在数据处理、造表和函数求值运算中都很有用,为了按照要求的格式进行结果的显示,就需要熟悉 MATLAB 中关于字符串的操作。

1. 字符串及其显示

1) 字符串和字符串变量名

用单引号界定的排成序列的各种符号,包括数字、英文、汉字、横线、括号及表达式、方程式等,都被看作是字符串(也称字符串数据或字符量)。用赋值号“=”可以把它赋给某个标识符,这样的标识符称为字符串变量名,简称字符名。

例如,键入:

```
s1 = 'hello'
```

回车得出:

```
s1 =  
hello
```

这里,标识符 `s1` 是字符串变量名, `hello` 是字符串变量。

2) 字符串的输出显示指令

MATLAB 中有几个常用的输出显示指令,可以使字符串或数据按一定格式显示。

(1) 字符串或数据显示指令

最常用的指令是 `disp`,它的使用格式为:

```
disp(ZS)
```

① 输入参数 `ZS` 可以是数值变量、字符串变量(或它们的变量名)和被界定的字符串,但每次只能有一个内容,各变量之间用空格分开。

② `ZS` 若是字符串,回车后照原样显示(包括字符间的空格大小);`ZS` 若是数值变量名,则显示它代表的数值。

(2) 空字符(空格)输出显示指令

字符或数据间的空格大小,除在字符串间留有空格外,也可以利用空字符指令 `blanks` 控制,它的使用格式为:

```
blanks(n)
```

① 该指令作为 `disp` 指令的输入参数,可夹杂在数据或字符串中间使用;

- ② 输入参数 n 为整数 ,表示遇到 `blanks(n)` 时输出 n 个字符的空格 ;
- ③ 仅输入 `disp(blanks(n)')` ,回车后将使光标下移 n 行。

例 2-28 造一张函数表 ,显示 $\sin t, e^t, \ln t$ 在 $t \in [0.1, \pi/4]$ 间的取值 , t 取值间隔为 0.2 ,并显示出这张函数表的表头。

解 键入 :

```
s1 = ' sin t ' ; s2 = ' exp ( t ) ' ; s3 = ' ln t ' ; t = [ 0.1 : 0.2 : pi / 4 ]  
' ; % 输入数据  
s = '      t   sin t   exp ( t )   ln t ' ; % 显示表头 ,字符串间预留  
空格  
disp ( s ) , disp ( [ t sin ( t ) exp ( t ) log ( t ) ] ) % 显示数据
```

回车得出 :

t	sin t	exp (t)	ln t
0.1000	0.0998	1.1052	- 2.3026
0.3000	0.2955	1.3499	- 1.2040
0.5000	0.4794	1.6487	- 0.6931
0.7000	0.6442	2.0138	- 0.3567

也可以键入 :

```
disp ( [ blanks ( 4 ) ' t ' blanks ( 9 ) s1 blanks ( 5 ) s2 blanks ( 3 )  
s3 ] ) , disp ( [ t sin ( t ) exp ( t ) log ( t ) ] )  
代替“显示表头”和“显示数据”两句 ,表头中的空格用空字符指令方法显示 ,回车得出与上面  
相同的结果。
```

(3) 格式化数据显示指令

为了使输出的数据按规定的格式(表示方式、小数点位数等)显示 ,常用指令 `sprintf` ,它的使用格式为 :

```
sprintf ( ' Z ' , S1 , S2 , ... )
```

- ① 输入参数的第一部分 'Z' 由两部分组成 (这两部分可以交错混合排列) :
- 打算作为字符串显示的内容 (字符、文字、符号等) ;
 - 控制第二部分数据 $S1, S2, \dots$ 显示形式的“格式符” 格式符及其意义列在表 2-7 中 ,两个格式符之间空格的大小就是数据显示时的间距。

表 2-7 控制数据显示的格式符及其意义

格 式 符	显 示 格 式	格 式 符	显 示 格 式
% e	指数格式	% g	自动选择指数或小数中较短的格式
% f	小数格式	% m . n f	数据共占 m 个字符宽度 ,显示 n 位小数
% d	十进制整数格式	\n	换行符

- ② 输入参数的第二部分 $S1, S2, \dots$ 是打算输出的数据或数值变量名称 ,它们之间用逗号分开。
- ③ 回车后输出 'Z' 中的字符串和 $S1, S2, \dots$ 代表的数据 ,数据按 'Z' 中与其排序对应的格

式符要求的形式显示,如果格式符少于数据个数,格式符可循环使用。

例2-29 分两行输出自然数 e , $\ln 5$ 和最小浮点数 eps 的符号(即表头)和数值。

解 键入:

```
shju=[exp(1) log(5) eps];
```

键入:

```
zfch=sprintf(' e ln5 eps\n %9.3f; %f; %e;',
```

```
shju)
```

回车得出:

```
zfch =
```

```
 e ln5 eps
```

```
2.718; 1.609438; 2.220446e-016;
```

键入上述指令时,适当调节 e , $\ln 5$ 和 eps 间的空格大小,以便使表头和数据对齐。

例2-30 用指令 `sprintf` 显示出“ $e = 2.718\ 282e + 000$; $e = 2.718\ 282$; $e = 2.718\ 3$; $e = 2.718\ 28$; $e = 2.718\ 282e + 000$ ”。

解 键入:

```
sprintf('e=%d;e=%f;e=%6.4f;e=%g;e=%e',exp(1),exp(1),exp(1),exp(1),exp(1))
```

回车得出:

```
ans =
```

```
e = 2.718282e + 000; e = 2.718282; e = 2.7183; e = 2.71828; e = 2.718282e + 000
```

指令中的“ $e =$ ”,“;”属于字符串,输出时被原样输出显示,同时也显示出数据。

该例表明指令 `sprintf` 可以同时输出要显示的字符串和数据,而指令 `disp` 却无此功能。

2. 自定义函数求值

把基本初等函数按照需要用有限个四则运算符号连接起来,就可以构成任意初等函数,即自定义函数。MATLAB 中可以用数值或字符变量表达式、内联函数指令、自己编写的M-函数文件等多种方法自定义函数,然后进行函数的求值运算。

1) 用数值变量表达式自定义函数

给数值变量名赋值后,用 MATLAB 数组算法的四则运算符号把这些代表数值矩阵的数值变量名连接成函数表达式(即自定义函数),回车便可得出自定义函数的取值。

创建函数表达式时必须注意:

- ① 表达式中的四则运算必须用数组算法符号,否则就成为矩阵函数;
- ② 表达式两端不得加引号,否则整个表达式将被定义成字符串;
- ③ 这种函数表达式的使用是“一次性”的,再次使用时得重新输入;
- ④ 在输入表达式前必须先给数值变量名赋值。

2) 用字符变量表达式自定义函数

把函数表达式定义成字符串表达式(自定义函数),给字符串赋以数值后,通过数值转换指令 eval 将字符串表达式转换成数值,从而得出函数值。创建字符串表达式时注意:

- ① 表达式两端必须加单引号界定,使它被定义成字符串表达式;
- ② 表达式中的四则运算必须用数组算法符号,否则就成为矩阵函数;
- ③ 将表达式赋值给一个标识符,便被保存在工作空间中,关机前可以多次调用;
- ④ 可以先输入“字符串表达式”,后给字符串变量名赋值。

3) 用内联函数指令自定义函数

内联函数指令为:

inline

若定义名为 fun2_1 函数,其格式如下:

fun2_1 = inline(字符串表达式)

- ① 输入参数“字符串表达式”可以是函数字符串表达式,或代表它的标识符;
- ② 该指令定义了名称为 fun2_1 的内联函数,函数名称可选用任意标识符;
- ③ 函数表达式用数组算法时,定义的是函数矩阵(矩阵的元素是函数),若用矩阵算法则定义的是矩阵函数(自变量是矩阵的函数);
- ④ 调用该内联函数的格式为 fun2_1(a),回车输出将数值矩阵 a 代入函数的结果;
- ⑤ 内联函数被保存在工作空间中,关机前可以多次调用。

例 2-31 已知 $A = (a_{ij})_{2 \times 3} = \begin{bmatrix} 3\pi & 5 & 2 \\ 7 & 10 & 5 \end{bmatrix}$,用三种方法求 $y_{ij} = f(a_{ij}) = a^2 + \sin^3 a - e^{-a}$ 的值。

解 (1) 用数值变量定义函数

键入:

$A = [3 * \pi \ 5 \ 2; 7 \ 10 \ 5]; f1 = A.^2 + \sin(A).^3 - \exp(-A)$ % 必须在 f1 之前给

A 赋值

回车得出:

f1 =

88.8264	24.1115	4.6165
49.2827	99.8389	24.1115

表明:

$$Y = f(A) = \begin{bmatrix} f(3\pi) & f(5) & f(2) \\ f(7) & f(10) & f(5) \end{bmatrix} = \begin{bmatrix} 88.8264 & 24.1115 & 4.6165 \\ 49.2827 & 99.8389 & 24.1115 \end{bmatrix}$$

矩阵相等即对应元素相等,所以得出:

$$f(3\pi) = 2 \times (3\pi)^2 + \sin^3(3\pi) - e^{-3\pi} = 88.8264$$

$$f(5) = 2 \times (5)^2 + \sin^3(5) - e^{-5} = 24.1115$$

.....

(2) 用字符变量定义函数

键入:

f12 = 'A.^2 + sin(A).^3 - exp(-A)'; % 定义字符串表达式,然后再给 A

赋值

```
A=[3* pi 5 2;7 10 5 ]; f12_A=eval(f12) % 或 eval('A.^2+sin(A).^3 - exp(-A)')
```

回车得出：

```
f12_A =
    88.8264    24.1115    4.6165
    49.2827    99.8389    24.1115
```

(3) 用内联函数指令定义函数

先定义内联函数 ,键入：

```
f2=inline('x.^2+sin(x).^3-exp(-x)')
```

回车得出：

```
f2 =
    Inline function:
    f2(x)=x.^2+sin(x).^3-exp(-x) % 定义了内联函数 f2(x)
```

再代入自变量值 ,键入 `A=[3* pi 5 2;7 10 5]`; `f2(A)` ,回车便会得出与前相同的结果。

关机前用上述(2)、(3)两种方法计算函数值 ,就像代公式一样方便。但是一旦关机 ,用这两种方法编辑的函数就不复存在了。对于需要多次调用的函数 ,可以用后边的编程方法把它编辑成 M-文件并予存盘 ,这样就能像使用 MATLAB 中的固有指令一样不受关机的影响。

2.2.2 符号变量

1. 符号变量和符号表达式的创建

像数学物理公式中的代数符号一样 ,用于表示代数变量的标识符叫符号量。符号量和符号表达式是通过专用指令 `syms` 和 `sym` 来创建的 ,代表符号量的标识符称为符号变量名。

1) 用 `syms` 创建符号量

`syms` 是创建符号量的便捷指令 ,它的使用格式为：

```
syms a1 a2 a3...flag1
```

① 输入参数 `a1 a2 a3...` 均为需要定义成符号量的单个字符或标识符 ,一次可以定义多项。输入参数 `a1 a2 a3...` 不能是数字、函数表达式或方程式。

② 输入参数 `a1 a2 a3...flag1` 均不用加引号界定 ,它们之间必须用空格分开。

③ 参数 `flag1` 可根据需要选取下述字符串之一：

`positive`——定义成正实型符号变量；

`real`——定义成实符号量(在数学问题的讨论中有时需要区分实型和正实型)；

`unreal`——定义成复数符号变量。

若省略 `flag1` 则默认为选取“`unreal`”。

④ 数字、运算符和被 `syms` 定义的符号量被运算符连接成的表达式叫作符号表达式。

2) 用 `sym` 创建符号量、符号表达式等

`sym` 指令不仅可以创建符号量 , 而且还能直接创建符号表达式等 , 使用格式为 :

标识符 = `sym`(A, flag)

① 输入参量 A 可以是数字或字符串 , 当然也可以是字符串变量名、字符表达式或字符方程式 , 甚至可以是界定成字符串的汉字 , 但每次只能定义一项。

② 回车后 A 将被定义成符号量并赋值给赋值号左边的标识符 , 输入 `a1 = sym('a1')` 与输入 `syms a1` 等价。

③ 若 A 为字符串时 , flag 可选取的内容与 `syms` 中 flag1 完全相同 , 但选定的 flag1 内容必须用单引号界定 , 此时省略 flag , 则默认取 'unreal'。

④ 若 A 为数字时 : flag 可取 :

'd' —— 定义成最接近的十进制浮点精确表示形式 ;

'e' —— 定义成带估计误差的有理表示形式 ;

'f' —— 定义成十六进制浮点表示形式 ;

'r' —— 定义成最接近的有理表示形式 , 如 p/q 等。

此时省略 flag , 则默认为取 'r'。

注 : 键入 `s1 = 'hello'` 和 `s2 = sym(s1)` 得出的结果形式一样 , 但变量类型不同 , 键入 `s3 = sym('s1')` 和 `s2` 所得结果不同 , 键入 `sym(hello)` 则是错误的 , 这些都是为什么 ?

例 2-32 把 `s1` , `s2` 和 `s3` 分别定义成正实型、实型和复数型符号变量 , 并予查验。

解 首先定义 `s1` , `s2` 和 `s3` , 为此键入下述指令 :

```
syms s1 positive, s2 = sym('s2', 'real'); s3 = sym('s3');
```

回车。

为了查验与定义是否相符 , 分别求出 `s1` , `s2` 和 `s3` 的实部和虚部 , 键入 :

```
s1r = real(s1), s1m = imag(s1), s2r = real(s2), s2m = imag(s2), s3r = real(s3),  
s3m = imag(s3)
```

回车得出 :

```
s1r =
```

```
s1
```

```
s1m =
```

```
0
```

(由 `s1r` 和 `s1m` 可知 `s1` 是正实型符号量)

```
s2r =
```

```
s2
```

```
s2m =
```

```
0
```

(由 `s2r` 和 `s2m` 可知 `s2` 是实符号量)

```
s3r =
```

```
1/2 * s3 + 1/2 * conj(s3)
```

```
s3m =
```

```
1/2 * i * (s3 - conj(s3))
```

(由 `s3r` 和 `s3m` 可知 `s3` 是复数型符号量)

例 2-33 在指令窗中键入 `s3 = 1/2` , `s4 = '1/2'` , `s5 = sym(1/2)` , 它们的属性如何 ?

解 键入：

```
s3 = 1/2, s4 = '1/2', s5 = sym(1/2)
```

回车得出：

```
s3 =
    0.5000
s4 =
    1/2
s5 =
    1/2
```

再键入：

```
whos s3 s4 s5
```

回车得出：

Name	Size	Bytes	Class
s3	1x1	8	double array(双精度数值)
s4	1x3	6	char array
s5	1x1	130	sym object

s4 和 s5 虽然形式一样,但其类型性质不同。

例 2-34 用标识符 ad 代表 $ax^2 + c$ 符号表达式,用 af 代表 $ax^3 + bx^2 - c = 5$ 符号方程。

解 键入：

```
ad = sym('a* x^2 + c')
```

回车得出：

```
ad =
    a* x^2 + c
```

(也可以键入 `syms a x c, as = a* x^2 + c` 完成上述任务)

键入：

```
af = sym('a* x^3 + b* x^2 - c = 5')
```

回车得出：

```
af =
    a* x^3 + b* x^2 - c = 5
```

2. 查询变量类型指令

了解一个变量的类型是经常遇到的问题,用指令 `whos` 查出的是变量性质的详细资料:名称、维数、字节数和类型。若仅想知道某个变量 `a` 的类型,则可以用指令 `class(a)`,它只给出 `a` 的类型,简洁方便。例如,想知道例 2-33 中的 `s3`, `s4` 和 `s5` 的数据类型,可键入：

```
s3 = class(s3), s4 = class(s4), s5 = class(s5)
```

回车得出：

```
s3 =
    double
```

```
s4 =
    char
s5 =
    sym
```

3. 注意字符量和符号量的差异

符号型和字符串型数据变量容易被混淆, 因为符号运算工具箱中有些指令的参数既可以用符号型数据, 又可以用字符串型数据, 但也有一些指令的参数必须得用符号型数据。对此必须查看指令说明, 在使用中逐步掌握。

例如, 键入:

```
c1 = '1/2';    c2 = sym(3/7);    b = c1 + c2
```

回车得出:

```
b =
    13/14
```

键入:

```
c1 = class(c1), c2 = class(c2), b = class(b)
```

回车得出:

```
c1 =
    char
c2 =
    sym
b =
    sym
```

加法运算中把数值形式的字符量和符号量都作为符号量处理, 不加区分。又如对于求导函数指令 `diff`, 其输入参量同样既可以用符号表达式, 又可以用字符表达式。例如, 键入:

```
f = 'sin(x)^2';    f1 = sym(f);    dfdx = diff(f)    或    dfdx = diff(f1)
```

回车得出同样结果:

```
dfdx =
    2* sin(x)* cos(x)
```

这里 `f` 是字符变量名, 而 `f1` 是符号变量名。

但是, 级数求和指令 `symsum(s, 'n', h, k)` 就不一样(参数 `s` 是通项表达式, `n` 为级数的项数, `h` 和 `k` 分别为求和的起始和终止项数)。它的输入参数 `s` 必须得用符号表达式, 而不能用字符表达式。如键入:

```
s = 'x/(1+2^n)';    s1 = sym(s);    symsum(s, 'n', 1, 3)
```

回车得出:

```
??? Function 'symsum' is not defined for values of class 'char'.
```

指出字符串不适用于 `symsum` 函数。若把 `s` 改为 `s1`, 键入:

```
symsum(s1, 'n', 1, 3)
```


回车得出：

```
ans =
    29/45 * x
```

2.2.3 符号矩阵的创建方法

以符号数据、符号变量或符号表达式 (Symbolic expression) 等为元素构成的矩阵叫符号矩阵。它的创建方法有以下几种。

1. 直接创建法

首先定义一些符号量、符号变量、符号表达式或符号方程,然后跟创建数值矩阵的方法和规则一样,把它们作为矩阵的元素键入方括号内,就构成了符号矩阵。但是符号矩阵的显示格式与数值矩阵不同,每行元素都被显示在一个方括号内,两个元素之间还有用于间隔的逗号。

例如,键入：

```
syms c1 c2 c3, B=[c1 3*c2; c3-cos(c1),c2]
```

回车得出：

```
B =
[      c1,      3 * c2 ]
[ c3 - cos(c1),      c2 ]
```

2. 用 sym 指令创建符号矩阵

与创建符号表达式或符号方程的方式类似,用 sym 创建符号矩阵的格式为：

```
sym(A)
```

① 输入参量 A 可以是数字矩阵、字符串矩阵、符号量矩阵,或者是它们的变量名；

② 矩阵 A 的元素也可以是字符串或符号量构成的表达式或方程,两个元素之间最好用逗号分隔,以防止对空格的误识别。

例 2-35 创建符号矩阵 $f_j = \begin{bmatrix} \frac{a}{\sin x} & \cos x \\ b - \frac{x}{5} & a \sin x \end{bmatrix}$ 和 $d1 = \begin{bmatrix} \frac{1}{3} & 5.12 & 6.45 \\ \sin 4 & \sqrt{3} & 9 \end{bmatrix}$ 。

解 (1) 创建 f_j 键入：

```
fj=sym('[a/sin(x),cos(x);b-x/5,a*sin(x)]')
```

回车得出：

```
fj =
[a/sin(x), cos(x)]
[b - x/5, a*sin(x)]
```

(2) 创建 $d1$ 键入：

```
d=[1/3 5.12,6.45;sin(4),sqrt(3),9]
```

回车得出：

```
d =
    0.3333    5.1200    6.4500
   -0.7568    1.7321    9.0000
```

键入：

```
d2 = sym(d) % 把数值矩阵定义为符号矩阵
```

回车得出：

```
d2 =
    [1/3, 128/25, 129/20]
    [-6816670871723694*2^(-53), sqrt(3), 9]
```

键入：

```
d1 = sym(' [1/3,5.12,6.45;sin(4),sqrt(3),9 ]') % 把字符串矩阵定义为
```

符号矩阵

回车得出：

```
d1 =
    [1/3, 5.12, 6.45]
    [sin(4), sqrt(3), 9]
```

2.2.4 符号矩阵元素的标识和删改

1. 元素的标识

让一个标识符完全代表符号矩阵中的一个或部分元素,叫符号矩阵的元素标识,它和数值矩阵中元素的标识方法完全一样。

2. 元素的替换修改

符号矩阵的替换修改方法与数值矩阵中完全一样。此外,符号矩阵的替换还可用替换指令 subs 进行,其使用格式为：

```
subs(B,old,new)
```

该指令表示把符号矩阵 B 中指定的内容 old,用新量 new 代替。

例 2-36 创建符号矩阵 $\begin{bmatrix} \sin(bx) & a+b \\ e^{ax} & \sqrt{x} \end{bmatrix}$, 并将它的第一行第一列元素换成 $v2c$ 。

解 键入：

```
g = sym(' [a* sin(b* x), a+b; exp(a* x), sqrt(x) ]'); % 创建符
```

号矩阵

```
g(1,1) = 'v2c' 或 g = subs(g,g(1,1),'v2c') % 替换 g 的第一行第一
```

列元素

回车得出：

```
g =
    [ v2c, a+b ]
```

`[exp(a * x), x^(1/2)]`

例 2-37 用 3 阶魔方矩阵代替代数表达式 $a^2 * \sin(x)$ 中的 a 。

解 键入：

```
syms a x, subs(a^2 * sin(x), a, magic(3))
```

回车得出：

```
ans =
    [ 64 * sin(x),      sin(x), 36 * sin(x) ]
    [  9 * sin(x), 25 * sin(x), 49 * sin(x) ]
    [16 * sin(x), 81 * sin(x),  4 * sin(x) ]
```

这里 符号量 a 被 3 阶魔方阵所替换 原来的符号表达式变换成符号矩阵。

2.2.5 符号矩阵的运算

1. 符号矩阵的四则运算

符号矩阵进行加、减、乘、除运算时,同样分为“矩阵算法”和“数组算法”,其运算符号和意义跟数值矩阵中的完全一样。只是关系运算中只有“等于”关系一个概念,运算符有“等于(=)”、“不等于(\sim)”,而没有“大于”、“小于”等其他比较概念。

例 2-38 若 $\mathfrak{d} = \begin{bmatrix} \frac{a}{b} & \sin a \\ b^3 & 5 \end{bmatrix}$, $\mathfrak{z} = \begin{bmatrix} \frac{4}{b} & \cos b \\ a^2 & 8 \end{bmatrix}$, 计算 $\mathfrak{d} + \mathfrak{z}$ 和矩阵 \mathfrak{d} 与 \mathfrak{z} 的矩阵算

法积和数组算法积。

解 键入：

```
s1=sym(' [a/b, sin(a); b^3, 5 ] '); s2=sym(' [4/b cos(b); a^2 8 ] '); s1+s2
```

回车得出：

```
ans =
    [ a/b + 4/b, sin(a) + cos(b) ]
    [ b^3 + a^2,      13 ]
```

键入：

```
ss=s1 * s2, sss=s1. * s2
```

回车得出：

```
ss =
    [ 4 * a/b^2 + sin(a) * a^2, a/b * cos(b) + 8 * sin(a) ]
    [ 4 * b^2 + 5 * a^2, b^3 * cos(b) + 40 ]
sss =
    [ 4 * a/b^2, sin(a) * cos(b) ]
    [ b^3 * a^2,      40 ]
```

2. 符号矩阵的求逆

设 B 为非奇异符号矩阵,求逆指令与数值矩阵中的完全一样,用 $\text{inv}(B)$ 得出 B^{-1} 。

例 2-39 计算例 2-38 中矩阵 $s1$ 的逆阵和 $s1$ 被 $s2$ 除的结果。

解 求 $s1$ 的逆阵,键入:

```
inv(s1)
```

回车得出:

```
ans =
```

```
[5/(5*a - sin(a)*b^4)*b, -sin(a)/(5*a - sin(a)*b^4)*b]
[-b^4/(5*a - sin(a)*b^4), a/(5*a - sin(a)*b^4)]
```

键入:

```
s1/s2
```

回车得出:

```
ans =
```

```
[a*(a*b*sin(a)-8)/(cos(b)*a^2*b-32), (cos(b)*a-4*sin(a))/(cos(b)*a^2*b-32)]
[b*(5*a^2-8*b^3)/(cos(b)*a^2*b-32), (cos(b)*b^4-20)/(cos(b)*a^2*b-32)]
```

2.2.6 符号矩阵运算中的几个特有指令的应用

符号矩阵与数值矩阵中的基本函数运算大体相同,只是有关对数的运算有所不同,符号运算中只有以 e 为底的自然对数 $\log()$ (数学中的 \ln) 运算,没有其他数为底的对数运算。下面介绍几个符号矩阵特有的运算指令,它们常用于公式推导。

1. 因式分解、展开、合并指令

1) 因式分解

因式分解指令 factor 的使用格式为:

```
factor(S)
```

输出对 S 进行因式分解的结果,其中 S 是符号矩阵、符号表达式或有理分式。

例 2-40 对矩阵 $\begin{bmatrix} \frac{x}{x^2-5x-6} & \frac{2x}{x^2-2x+1} \\ \frac{x^2+x}{x^2+2x+1} & \frac{x+1}{x^2+x} \end{bmatrix}$ 的各元素进行因式分解。

解 键入:

```
k=sym(' [x/(x^2-5*x-6), 2*x/(x^2-2*x+1); (x^2+x)/(x^2+2*x+1), (x+1)/(x^2+x)]');
```

```
factor(k)
```

回车得出：

```
ans =  
[x/(x+1)/(x-6), 2*x/(x-1)^2]  
[x/(x+1), 1/x]
```

`factor(s)` 中的 `s` 也可以是代数式或有理分式, 如对 $s1 = 3x^2 - 10x + 8$ 进行因式分解, 可键入:

```
syms x, s1 = 3 * x^2 - 10 * x + 8; s1 = factor(s1)
```

回车得出：

```
s1 =  
(3 * x - 4) * (x - 2)
```

2) 代数式展开

代数式展开指令 `expand` 的使用格式是：

```
expand(S)
```

输出对符号矩阵 `S` 的每个元素进行代数式展开的结果 `S` 当然也可以是一个代数式。

例 2-41 将符号矩阵 $A = \begin{bmatrix} (x+1)^3 & \sin(x+y) \\ e^{x+y} & \cos(x+y) \end{bmatrix}$ 的各元素展开。

解 键入：

```
A = sym('[(x+1)^3, sin(x+y); exp(x+y), cos(x+y)]'); expand  
(A)
```

回车得出：

```
ans =  
[x^3 + 3 * x^2 + 3 * x + 1, sin(x) * cos(y) + cos(x) * sin(y)]  
[exp(x) * exp(y), cos(x) * cos(y) - sin(x) * sin(y)]
```

3) 同幂项系数合并

同幂项系数合并指令为：

```
collect(S, 'v')
```

输出对符号矩阵 `S` 各元素按“`v`”的同幂次项系数进行合并的结果, 省略“`v`”时默认变量为“`x`”, `S` 也可以是一个代数式。

例 2-42 将 $x^2y + xy - x^2 - 2x$ 和 $-\frac{1}{4}xe^{-2x} + \frac{3}{16}e^{-2x}$ 两个代数式进行同幂系数合并。

解 键入：

```
syms x y, collect(x^2 * y + y * x - x^2 - 2 * x)
```

回车得出：

```
ans =  
(y - 1) * x^2 + (y - 2) * x
```

键入：

```
f = -1/4 * x * exp(-2 * x) + 3/16 * exp(-2 * x); collect(f, exp(-2
```

* x))

回车得出：

```
ans =
(- 1/4 * x + 3/16) * exp(- 2 * x)
```

2. 求函数极限和导数指令

1) 求函数极限指令

limit(F,x,a,'right'或'left')

① 输出函数符号表达式 $F(x)$ 当自变量 $x \rightarrow a$ 时的极限 $\lim_{x \rightarrow a} F(x)$ 。

② 省略 a 时默认 $a = 0$ 。

③ 输入参数选 right 表示 $x \rightarrow a$ 的右极限 $\lim_{x \rightarrow a+0} F(x)$ 选 left 表示 $x \rightarrow a$ 的左极限

$\lim_{x \rightarrow a-0} F(x)$ 。省略这最后一个输入参数时 表示左右极限相等。

例 2-43 求 $\lim_{x \rightarrow a} \frac{\sqrt[m]{x} - \sqrt[m]{a}}{x - a}$ 。

解 键入：

```
syms x m a, limit((x^(1/m) - a^(1/m))/(x - a), x, a)
```

回车得出：

```
ans =
a^(1/m)/a/m
```

即

$$\lim_{x \rightarrow a} \frac{\sqrt[m]{x} - \sqrt[m]{a}}{x - a} = \frac{\sqrt[m]{a}}{ma}$$

2) 求导函数指令

diff(S,'v',n)

① 输入参数 S 为函数、函数向量、函数矩阵的符号表达式或字符表达式；

② 输入参量 v 为求导的指定自变量，省略时默认为 x, t 等约定俗成的自变量；

③ 输入参量 n 为求导阶数，缺省时默认 $n = 1$ ，即求一阶导函数；

④ 输出量为函数 S 对变量 v 的 n 阶导函数；

⑤ 由于求导函数时可以指定自变量，因此该指令也可用于求偏导函数。

例 2-44 已知二元函数 $z(x, y) = xye^{\sin(\pi xy)}$ ，求 $\frac{\partial z}{\partial x}$ 和 $\frac{\partial z}{\partial y}$ 。

解 在指令窗中键入：

```
syms x y, z = x * y * exp(sin(pi * x * y)); diff(z)
```

回车得出：

```
ans =
```

```
y * exp(sin(pi * x * y)) + x * y^2 * cos(pi * x * y) * pi * exp(sin(pi * x * y))
```

即 $\frac{\partial z}{\partial x} = ye^{\sin(\pi xy)} [1 + \pi xy \cos(\pi xy)]$ 。上述指令 diff(z) 中，可不写指定自变量 x 。

再键入：

```
diff(z, 'y')
```

回车得出：

```
ans =
```

```
x*exp(sin(pi*x*y))+x^2*y*cos(pi*x*y)*pi*exp(sin(pi
```

即 $\frac{\partial z}{\partial y} = x e^{\sin(\pi xy)} [1 + \pi xy \cos(\pi xy)]$ 。上述指令 `diff(z, 'y')` 中,指定自变量 y 不能省略。

例 2-45 已知 $A = (a_{ij}) = \begin{bmatrix} \ln z & \sin z \\ -\cos z & e^{2z} \end{bmatrix}$, 求出每个元素对 z 的一、二阶导数。

解 键入：

```
syms z, A=[log(z),sin(z); -cos(z),exp(2*z)]; diff(A,z)
```

回车得出：

```
ans =
```

```
[ 1/z, cos(z) ]
[ sin(z), 2*exp(2*z) ]
```

再键入：

```
diff(A,z,2)
```

回车得出：

```
ans =
```

```
[ -1/z^2, -sin(z) ]
[ cos(z), 4*exp(2*z) ]
```

3. 级数求和

级数求和指令的使用格式为：

```
symsum(s,n,n0,nk)
```

① 输入参量 s 为级数通项的符号表达式。

② 输入参数 n 是通项中被认定的项数变量。当缺省 n 时,通项表达式 s 中的项数变量一般默认是“ x ”,若变量较多则可用 `findsym(s)` 查询哪个量是项数变量；

③ n_0, n_k 为项数的取值范围。 n_0 可取小数,但步长总是为 1,缺省 n_0, n_k 时,默认 $n_0 = 1, n_k = n - 1$ 。

④ 输出量是通项为 s 的级数第 n_0 项到第 n_k 项之和。

例 2-46 求 $\sum_{n=0}^{\infty} (-1)^n \frac{x^{n+1}}{n+1}$ 和 $\sum_{x=1}^{k-1} x^2$ 。

解 键入：

```
syms x n, symsum((-1)^n*x^(n+1)/(n+1),n,0,inf)
```

回车得出：

```
ans =
```

```
log(1+x)
```

即

$$\sum_{n=0}^{\infty} (-1)^n \frac{x^{n+1}}{n+1} = \log(1+x)$$

键入：

```
syms k, symsum(k^2, 1, k-1) 或 symsum(k^2)
```

回车得出：

```
ans =  
1/3 * k^3 - 1/2 * k^2 + 1/6 * k
```

即

$$\sum_{k=1}^{k-1} x^2 = \frac{1}{3}k^3 - \frac{1}{2}k^2 + \frac{1}{6}k$$

4. 一元函数的泰勒级数展开

将一个函数 f 展开成泰勒级数的专用指令是 `taylor` ,调用格式为：

```
taylor(f,n,'v',a)
```

① 输入参量 f 为待展开函数的符号表达式 ,不可省略；

② 输入参量 n 取正整数时 ,函数 f 被展开成最高幂次为 $(n-1)$ 的幂级数；

③ 输入参数 v 是被指定的变量名称 ,缺省时默认变量为 x 或 t , f 中只有一个可视为变量的符号量时 ,可省略指定变量名 v ；

④ 输入参数 a 表示函数 f 在 $v=a$ 点被展开 ,即展成 $(x-a)$ 的幂级数；

⑤ 缺省 a 时默认 $a=0$,函数在 $v=0$ 点展开 ,即函数 f 被展成麦克劳林级数 ,此时若省略 n 默认 $n=6$,函数 f 被展成 x 最高为 5 次幂的幂级数。

例 2-47 把 e^{-x} 展开成麦克劳林级数 , x 的最高次数取为 5。

解 键入：

```
syms x, taylor(exp(-x))
```

回车得出：

```
ans =  
1 - x + 1/2 * x^2 - 1/6 * x^3 + 1/24 * x^4 - 1/120 * x^5
```

即
$$e^{-x} \approx 1 - x + \frac{1}{2}x^2 - \frac{1}{3!}x^3 + \frac{1}{4!}x^4 - \frac{1}{5!}x^5 = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \frac{1}{24}x^4 - \frac{1}{120}x^5$$

例 2-48 把 $\ln(xy)$ 展开成 $(y-1)$ 的 5 次幂级数。

解 键入：

```
syms x y, taylor(log(x*y),6,'y',1)
```

回车得出：

```
ans =  
log(x) + y - 1 - 1/2 * (y-1)^2 + 1/3 * (y-1)^3 - 1/4 * (y-1)^4 + 1/5 *  
(y-1)^5
```

即
$$\ln(xy) = \ln x + y - 1 - \frac{1}{2}(y-1)^2 + \frac{1}{3}(y-1)^3 - \frac{1}{4}(y-1)^4 + \frac{1}{5}(y-1)^5$$

练习题 2.2

1. 利用数组函数指令造一张表,显示 x 从 0.01 到 $\pi/4$ 、以步长 0.1 变化时,下述几个函数的取值 $\cos x$, $\ln x$, x^3 和 \sqrt{x} 。

2. 求当 $t=0.5$, $3e$ 和 1.2 时 $f(t)=t^5 - \frac{3}{t^3} + t\sin(t)e^{-t} - 97$ 的值。

3. 求当 $x = -2.3$, -0.24 , 1 , $\pi/4$, $e/8$ 时, $f_1(x) = 2x^3 - \frac{1}{3x^2} + \frac{5\sin x}{x^2}e^x - 3$, $f_2(x) = 2x^9 - \frac{1}{9x^2} + \frac{5\sin x}{x^2}e^x - 9$ 和 $f_3(x) = 4x^3 - \frac{3}{3x^4} + \frac{5\sin x}{x^4}e^x - 9$ 的值。

4. 已知矩阵 $\mathbf{a} = \begin{bmatrix} \sin x - \cos x & \ln x^2 \\ e^{2x} & x^2 + 5 \end{bmatrix}$, $\mathbf{a} = \begin{bmatrix} \cos x & 1 - \ln x^2 \\ e^{-x} & -x^2 \end{bmatrix}$, 在 MATLAB 中计算 \mathbf{a}^{-1} , $\mathbf{a}1 + \mathbf{a}2$, $\mathbf{a}1\mathbf{a}2$, $\mathbf{a}1/\mathbf{a}2$ 。

5. 将第4题中 $\mathbf{a}1$ 里的第1行第2列元素换成 $1 + \ln x^2$ 、第2行2列元素换成 x^2 重新计算。

6. 将 $(x+1)^3 + (x-1)^3 - 2x^3$ 展开成幂函数。

7. 求 $f(x) = \ln \cos\left(\frac{1}{x}\right) + (x^2 - 2x + 3)e^{-x} + \sin x^3$ 的一、二阶导数。

8. 求下列函数的全微分 ① $z = \sin(x \cdot \cos(y))$ ② $u = \arcsin \frac{z}{\sqrt{x^2 + y^2}}$ 。

9. 计算当 $n=10$ 和无穷大时 $\sum_{x=1}^n \frac{a}{x^2}$ 的值。

10. 计算当 $x=3$ 和 5 时 $\sum_{n=1}^5 \frac{x^{n+1}}{n+1}$ 的值。

11. 将 $f(x) = \sin x^2 + \cos x \ln(x+a)$ 和 $\operatorname{sh} x$ 展成麦克劳林级数和 $(x-3)$ 的4次幂级数。

2.3 基本绘图方法

2.3.1 绘图入门

1. 图形窗基础知识









MATLAB 能绘制二维、三维等各种图形,还可以控制绘图的点型、线型和颜色,控制图形的视角、光照等品质,画面能充分表现出数据间的函数关系及图形的特征。

用 MATLAB 软件绘图非常轻松自如 ,只要在指令窗中键入绘图指令 ,回车后便在图形窗中得到相应的图形。如在指令窗中键入 `:[x,y,z]=sphere(30); surf(x,y,z), box` 回车 ,则屏幕上就出现图 2-1 所示的 MATLAB 图形窗 ,并画出了执行上述指令的图形。

由图 2-1 可见 ,图形窗的上面有两排菜单 ,第一排为主菜单 :【File】(文件)、【Edit】(编辑)、【View】(查看)、【Insert】(插入)、【Tools】(工具)、【Desktop】(桌面)、【Window】(窗口)、【Help】(帮助)。单击其中任何一项 ,就会被激活而弹出下属的子菜单 ,根据需要单击某项子菜单 ,便可实现其功能 ,从而可以对图形进行控制、修饰、改动 ,还可以在图面上加画直线、箭头、方框等 ,并能加写注释文字。图形窗的右上角有三个小图标 ,其用法与指令窗中的一样 ,单击它们可使窗口最小化、变换大小或被关闭。

第二排有 14 个小图标 ,它们是子菜单的快捷按钮 ,单击它们相当于打开主菜单中的某些子菜单。但是 ,单击他们的操作要比逐级调用相应子菜单方便很多 ,表 2-8 中列出了几个常用的快捷按钮及其相应的子菜单 ,并对其功能做了简要说明。

表 2-8 图形窗快捷按钮功能表

快捷按钮小图标	对应的菜单		功能简要说明
	主 菜 单	子 菜 单	
	【File】 (文件)	【New Figure】	创建新图形
		【Open File】	打开图形文件
		【Save Figure】	将图形存盘
		【Print Figure】	打印图形
	【Tools】 (工具)	【Edit Plot】	图形编辑
		【Zoom In】	局部放大
		【Zoom Out】	缩小被放大的曲线
		【Rotate 3D】	三维旋转

2. 图形窗的编号及分割

① 为了区别和保留不同画面 ,可以使用指令 `figure(n)` 对画面进行编号。该指令的输入

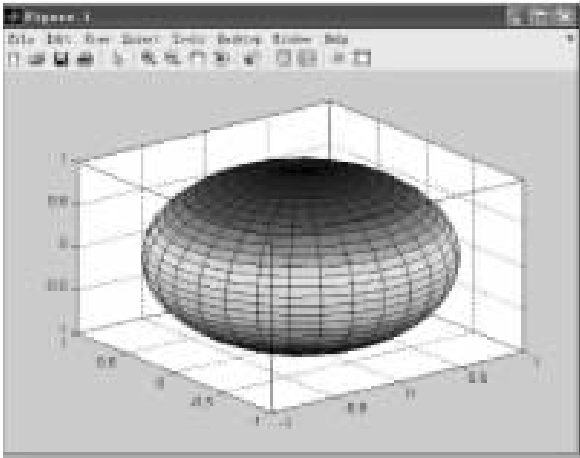



图 2-1 MATLAB 图形窗

参数 n 取正整数,表示图的序号。每当键入绘图指令前,先键入图形窗编号指令 `figure(n)`,此后的曲线就画在第 n 幅画面上,直到重新键入图形编号指令。



在指令窗中键入新的绘图指令并予以另一编号时,会新建一个图形窗,旧图会被新图遮盖。图号用小图标  保留在 Windows 任务栏内,想调出它时可单击该小图标。



② 为了在图形窗中同时显示出几个画面,MATLAB 中设有分割绘图窗口指令,它的调用格式是 `subplot(m,n,p)`,其中的参数 m, n, p 均为正整数。

键入该指令后,将把整个屏幕按 m 行 n 列分成 $m \times n$ 个子图,此后的绘图指令将在第 p 幅子图中画出。序号 p 是按先左右、后上下,即先行后列的顺序把 $m \times n$ 个图进行排列时的序号。画图前应该先键入 `subplot(m,n,p)`,指明在哪幅子图中绘制。

若想恢复全屏绘图,必须键入 `subplot(1,1,1)`。

3. 图形的存盘与调出

想把画好的图形存盘,可单击图形窗中的快捷按钮 ,在弹出对话框 `Save as` 左下角“文件名(N):”右侧栏内,把 `untitled` 改为所给图形起的文件名,单击  按钮,即自动加上扩展名“.fig”进行了存盘。

想调出已存盘的图形,可单击图形窗快捷按钮 ,在弹出的对话框里单击需调出的图形文件名,再单击  按钮即可。

2.3.2 二维图形的绘制

1. 数据绘图指令

`plot` 是 MATLAB 内部的平面数据绘图指令,基本调用格式有三种:

```
plot(X, 'S')
```

```
plot(X,Y, 'S')
```

```
plot(X1,Y1, 'S1', X2,Y2, 'S2', ..., Xn,Yn, 'Sn'),
```

下面分别予以介绍。

1) `plot(X, 'S')`

① 输入参数 X 为实向量时,绘出以 X 各分量的序号为横坐标、分量值为纵坐标的连线(折线)平面图。输入参数 X 为复向量时,则绘出以 X 各分量的实部数值为横坐标,虚部数值为纵坐标的连线平面图。


② 输入参数 X 为 $m \times n$ 阶实数矩阵时,相当于 n 个列向量,绘制 n 条折线,每条折线对应于 X 的一列,矩阵元素值为折线拐点的纵坐标、元素的行序号为横坐标。

③ 输入参数 X 为 $m \times n$ 阶复数矩阵时,相当于 n 个列向量,绘制 n 条折线。 X 的每列元素对应于一条线,元素实部值为折线拐点的横坐标,虚部值为纵坐标。

④ 输入参数 S 是修饰曲线的标记符号,它包含标志拐点的点型符号、曲线的线型符号和色型符号,分别标记数据点位置、曲线类型和曲线色彩;三者排序任意,两者之间不用分隔,它们可以部分或全部缺省,三者置于一对单引号内(见表 2-9)。

表 2 -9 标记曲线的点符、线符和色符

点标志符	点型名称	线标志符	线型名称	色标志符	颜色名称
.	点	-	实 线	r	红
o	圈	- -	虚 线	m	品 红
X	叉	- .	点 划 线	y	黄
+	十 字	:	点 线	g	绿
*	星 花			c	青
S	方 框			b	蓝
D	菱 形			k	黑
V	下尖三角形			w	白
^	上尖三角形	<div>注 1. 点符、线符和色符排列顺序不限 ,不用间隔 ,可同时标出 ,亦可部分或全部缺省 ;</div> <div>2. 色标志符缺省时默认为蓝色 ;</div> <div>3. 点、线标志符都缺省时默认为实线。</div>			
>	右尖三角形				
<	左尖三角形				
H	类六星、矛头				
P	准 五 星				

⑤ 画实线时可以在参数 S 后加写参数“'linewidth', n ”,用以控制线条宽度 n 标志线宽,也可在绘图窗中单击快捷按钮,用其菜单【Edit】进行编辑。

例 2-49 (1) 以向量 $y = (1 \ 2 \ 5 \ 4.5 \ 3 \ 6 \ 1)$ 各分量值为纵坐标,分量序号为横坐标,绘制顺序连线图。

(2) 在同一张图上绘制三个向量,其分量值纵坐标为 $y_1 = (1 \ 8 \ 1 \ 5)$ $y_2 = (5 \ 3 \ 4 \ 9)$, $y_3 = (3 \ 5 \ 3 \ 4)$ 横坐标为其分量序号的顺序连线图。

解 (1) 由于 y 为向量,可以直接用 `plot` 指令画出。

键入:

```
y1=[1 2 5 4.5 3 6 1]; % 输入向量 y1 的数据
subplot(1,2,1), plot(y1,'linewidth',2), grid
```

回车得出图 2-2。

(2) 在同一图上画出三个向量 y_1, y_2, y_3 代表的三条折线,先用给出的数据构成一个 3

列矩阵,即 4×3 阶矩阵 $x_2 = \begin{bmatrix} 1 & 5 & 3 \\ 8 & 3 & 5 \\ 1 & 4 & 3 \\ 5 & 9 & 4 \end{bmatrix}$ 。在指令窗中键入:

```
x2=[1,5,3;8,3,5;1,4,3;5,9,4]; subplot(1,2,2), plot(x2), xlabel('x'),
ylabel('y')
```

回车得出图 2-3。

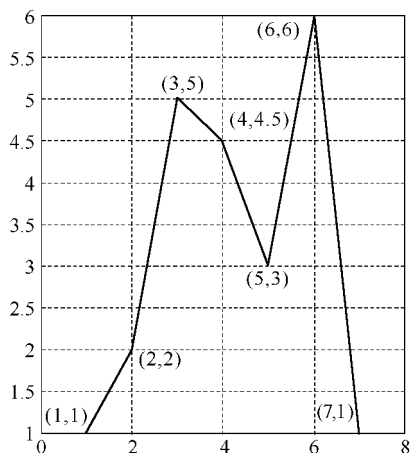


图 2-2 例 2-49(1)的一条折线图

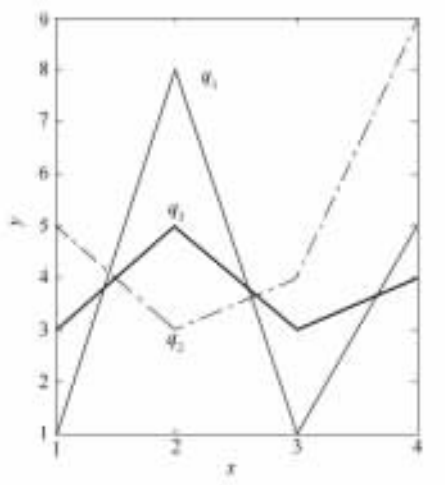


图 2-3 例 2-49(2)的三条折线图

注 ① `grid` 为在图中加画网格线指令;

② `xlabel('x')` 和 `ylabel('y')` 为加注坐标指令,把输入的字符串分别加注在 x 和 y 坐标轴旁。

2) `plot(X,Y,'S')`

(1) 若 X, Y 是同维向量,则绘制出 X 分量值为横坐标, Y 中与其对应分量值为纵坐标的

连线平面图。

(2) 若 X 是向量, Y 是矩阵, 且 X 维数与矩阵 Y 的行(或列)数相等, 则绘制出多条不同颜色的曲线, 曲线条数等于 Y 的列(或行)数。曲线(实为折线)拐点的横坐标为 X 的分量, 纵坐标为 Y 中与 X 分量对应的值。

(3) 若 X, Y 是同维矩阵时, 则以 X 的列元素值为横坐标, Y 中对应列元素值为纵坐标绘制一条曲线, 曲线条数等于矩阵列数。

MATLAB 是根据数据描点绘制曲线的, 如果不是根据已知数据而是根据已知函数绘图, 则绘图前必须把自变量 x 离散化, 即取 $x = x_1, x_2, x_3, \dots$, 函数 $y = f(x)$ 的表达式中变量之间必须用数组算法符号连接, 这样才能使 y 相应地离散化, 从而可用 `plot` 画出 $y = f(x)$ 的曲线。

例 2-50 用 `plot` 画出 $x = f(t) = e^{-t} \sin(t)$ 的曲线, $t \in [0, 3\pi]$ 。

解 先将自变量 t 离散化, 设取 $t = 0, 0.2, 0.4, \dots, 3\pi$, 函数 $x = f(t)$ 用数组算法符号连接。

键入:

```
t=0:.2:3*pi; x=exp(-t).*sin(t); plot(t,x)
```

回车得出图 2-4。

例 2-51 一次画出两条函数曲线 $\sin t$ 和 $e^{-t} \cos t$, $t \in [0, 3\pi]$, t 取值间隔为 0.02。

解 键入:

```
t=0:.02:3*pi; x=[sin(t);cos(t).*exp(-t)]; t=size(t),x=size(x),
plot(t,x)
```

回车得出图 2-5 和 t, x 的维数。

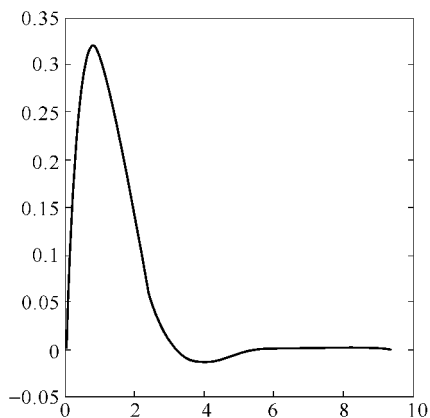


图 2-4 函数 $x = f(t) = e^{-t} \sin(t)$ 曲线

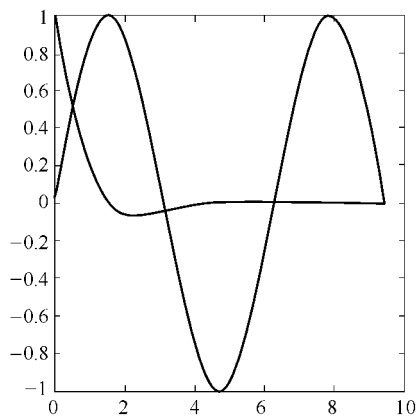


图 2-5 函数 $\sin t$ 和 $e^{-t} \cos t$ 曲线

$t =$

1 472 (t 是 472 维向量)

$x =$

2 472 (x 是 2×472 维矩阵)

若 t 的取值间隔较大, 即 t 的维数很小时, 光滑曲线就会显出其折线的本来面貌。

3) `plot(X1,Y1, 'S1',X2,Y2, 'S2',...,Xn,Yn, 'Sn')`

相当于一次同时调用 n 条 `plot(X,Y,S)` 指令,输入参量的要求与 `plot(X,Y,S)` 相同。

例 2-52 在同一图上画出下列三条函数曲线 $y_1 = 10\cos x, x \in [0, 3\pi]$; $y_2 = e^{\pi - 3x}, x \in [0.5, 8]$; $y_3 = \frac{6}{x}, x \in [1, 9]$ 。

解 三个函数的定义域不同,可以在各自的定义域内离散自变量,也可以都在定义域最大者中离散自变量。这里采用前一方法离散自变量。键入:

```
x1=0:0.2:3*pi;y1=10*cos(x1); x2=0.5:0.3:8;y2=exp(pi-3*x2);
x3=1:0.3:9;y3=6./x3; subplot(2,1,1);
plot(x1,y1,'r*- ',x2,y2,'+g:',x3,y3,'- .sk'),...
legend('10cosx','exp(pi-3x)','6./x')
```

回车得出图 2-6(a)。

例 2-53 画出下列分段函数的曲线 $y(x) = \begin{cases} 0 & -1 \leq x < 0 \\ 1 & 0 \leq x < 1.5 \end{cases}$ 。

解 键入:

```
subplot(2,2,3),plot([-1 0],[0 0],[0 0],[0 1],[0 1],[1 1],'linewidth',3),...
axis([-1.2 1.5 -0.5 2]),
```

回车得出图 2-6(b)。

例 2-54 用复数作图法画一个单位圆。

解 键入:

```
subplot(2,2,4),t=0:pi/100:2*pi; y=exp(i*t);plot(y),axis('square')
```

回车得出图 2-6(c)。

注 ① legend 为线型标注指令,详见“控制图形、画面的一些操作方法”相关内容。

② axis([xmin,xmax,ymin,ymax])是规定图形 x 轴和 y 轴坐标范围指令。

③ 用 plot 画直线时,既可以像例 2-53 那样只使用两个端点的坐标,也可以用数据描点绘图方法,这时可以键入下述指令同样可得图 2-6(b):

```
x1 = -1:0.1:0; y2 = 0:0.1:1; x3 = 0.01:0.1:1.1;
```

```
plot(x1,0*x1,0*y2,y2,x3,x3./x3,'linewidth',3),axis([-1.2 1.5 -0.5 1.5]);
```

若分段函数是曲线则只能用数据描点绘图方法绘制。

④ 图 2-6 这类并列组合图是用 subplot 绘制出的,这是一种不同于平常等分组合的绘图窗口分割方法,从中可以体会 subplot 灵活运用的技巧。

⑤ 例 2-54 中的语句 axis('square')表示坐标横轴和纵轴取值相等。

2. 函数绘图指令

数据绘图指令 plot 在提供大量数据组时能够把数据间的函数关系形象表示出来。这种使数据可视化的方法在实验数据的处理中非常有用,但是在绘制函数图时需要对函数进行分点取值非常麻烦。下面介绍的函数绘图指令则是根据解析函数直接绘图的,可根据函数自动分割取点,绘制出解析函数曲线,常用于解析函数的分析、观察和研究。

1) 解析函数绘图指令 fplot

使用格式为:

```
fplot('fun',lims,'S',tol)
```

① 用单引号界定的输入参数 fun,是解析函数字符串表达式、内联函数或 M-函数文件名。fun 可以是一个函数,也可以是元素为函数的向量(行矩阵)。

② 输入参数 lims 规定了绘图区间,用矩阵表示:lims=[a,b,c,d],即自变量 x 和函

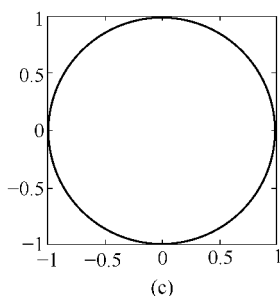
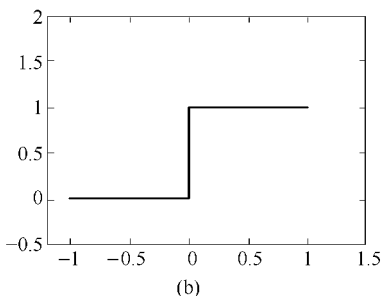
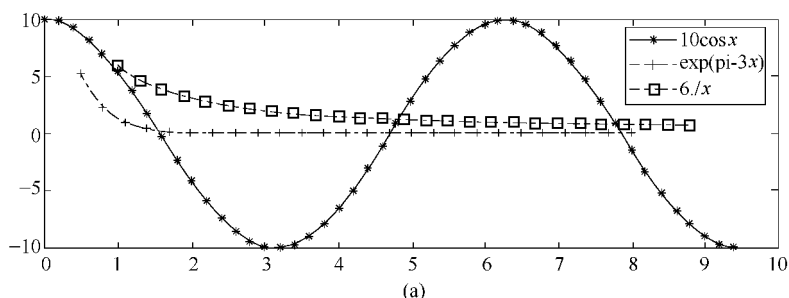


图 2-6 指令 subplot 绘图示例(例 2-52 ~ 例 2-54)

数 y 的取值范围分别是 $x \in [a, b]$ $y \in [c, d]$ 通常 c, d 被省略。

③ 输入参数 S 用于修饰曲线 与 `plot` 指令中用法一样。

④ 输入参数 `tol` 规定函数取值的相对误差。经常被缺省 默认值为 $2e-3$ 。

⑤ `fun` 是函数行矩阵或向量时 绘出的几条曲线的取值区间和线型是相同的。

注 ① 输入该指令的函数表达式是解析式 式中不用数组算法符号。

② 画图虽然也用描点法 但 x 的取值间隔是随函数曲线的曲率自动调节的 曲率大(曲率半径小)处间隔小 曲率小处间隔大。这种自适应地取值使绘制的曲线光滑、美观 还可以减少取点数目。

例 2-55 分别绘制 $[-\pi, \pi]$ 间的正弦函数曲线和 $e^{-x} \sin x$ 函数曲线。

解 “`sin`” 是正弦函数的 M-函数文件名 可以直接作为输入参数使用。

键入：

```
fplot('sin',[-pi pi], 'r +')
```

回车得图 2-7。

可以看出 x 取值间隔与曲线曲率有关 数据点的疏密程度随曲线曲率的增大而变大。

再键入：

```
fplot('exp(-x)*sin(x)',[-pi pi])
```

回车得出图 2-8。

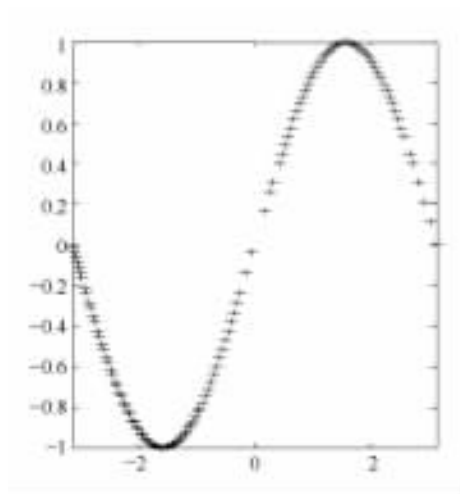


图 2-7 正弦函数曲线图

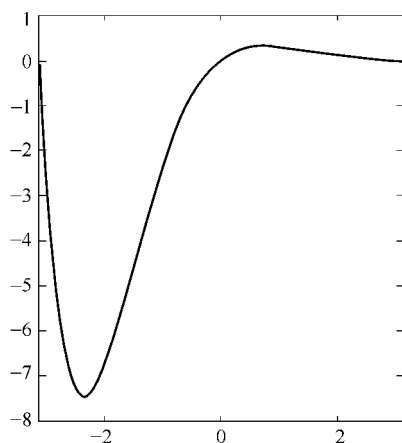


图 2-8 函数 $e^{-x} \sin x$ 曲线图

注 指令中 `fun` 为两个函数的乘积 $\exp(-x) * \sin(x)$ 它们的整体不是 M-函数文件 不能用 `sin` 代替 `sin(x)` 用 `exp` 代替 `exp(-x)` 因此必须写出自变量 x 不能写成 `-exp * sin`。

例 2-56 在同一幅画面上绘制出 $y = e^x$ $y = 3\sin x$ 和 $y = x^2$ 三条函数曲线 $x \in [-\pi, \pi]$ 。

解 用给出的三条函数曲线解析表达式构成一个行阵 用 `fplot` 一次就能画出三条曲线。

键入：

```
fplot('[exp(x),3*sin(x),x^2]',pi*[-1 1 -1 1])
```

回车。

这时得出的三条曲线线型完全相同仅颜色不同,为了加以区分,用绘图快捷按钮【Edit】和【Insert】中的相关功能及指令 `legend('exp(x)', '3sin(x)', 'x^2')`,最终加工修饰成图 2-9。

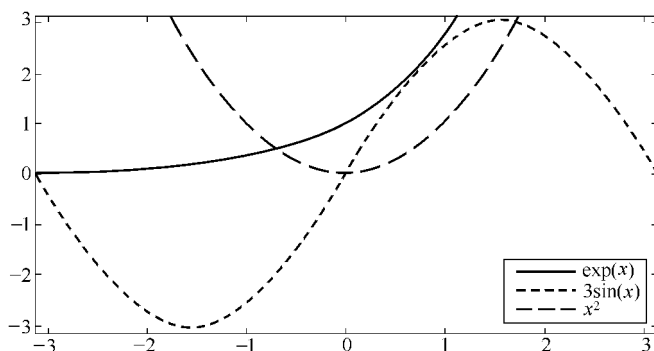


图 2-9 函数 e^x 、 $3\sin x$ 和 x^2 曲线

2) 隐函数绘图指令 `ezplot`

使用格式为：

`ezplot('func', lims)`

① 输入参数 'func' 可以是字符表达式、内联函数名或 M-函数文件名。

② 输入参数 func 为一元函数 $f(x)$ 时,输出 $y = f(x)$ 的几何图形。这时指令后面可以不用括号和引号。但函数的第一个符号不得是括号,不能加写输入参数 lims,默认绘图范围为 $[-2\pi, 2\pi]$ 。

③ 输入参数 func 为二元函数表达式 $f(x, y)$ 时,输出方程 $f(x, y) = 0$ 的几何图形,即绘制隐函数曲线。变量取值范围由 $\text{lims} = [a, b, c, d]$ 确定,表示 $x \in [a, b]$, $y \in [c, d]$ 。省略 c, d 时,默认 y 与 x 的取值范围相同。

④ 输入参数 func 为参数方程 $\begin{cases} x(t) = 0 \\ y(t) = 0 \end{cases}$ 时,func 写成“'x(t)', 'y(t)'”,按参数方程绘出参变量 $t \in \text{lims} = [a, b]$ 的函数曲线。

⑤ 输入参数 lims 用于规定自变量的取值范围 $[a, b]$,省略时默认自变量 $x \in [-2\pi, 2\pi]$ 。

⑥ 该指令每次只能绘制一条曲线,在绘出函数图的同时自动在图的上侧加注函数解析式,下侧加注自变量名称,曲线的色型、线型无法控制。

例 2-57 绘出余弦函数 $\cos x$ 和 $f(t) = t^3 + \frac{2e^t}{1-t^2} - t$ $t \in [-4, 4]$ 的曲线。

解 由于指令 `ezplot` 每次只能画出一条曲线,现分别画在两个图上。

键入：

`ezplot cos`

回车得出图 2-10。`cos` 是余弦的 M-函数文件名,可以省略自变量 x 。

再键入：

```
ezplot('t^3+2*exp(t)/(1-t^2)-t',[-4,4])
```

回车得出图 2-11。

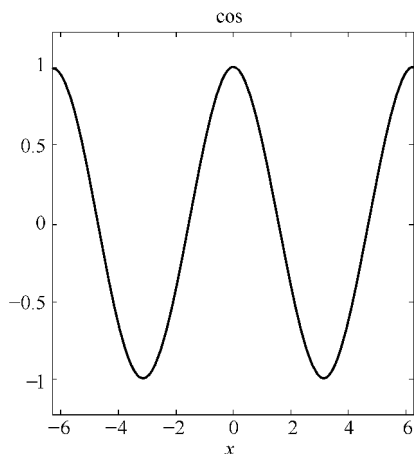


图 2-10 余弦函数曲线

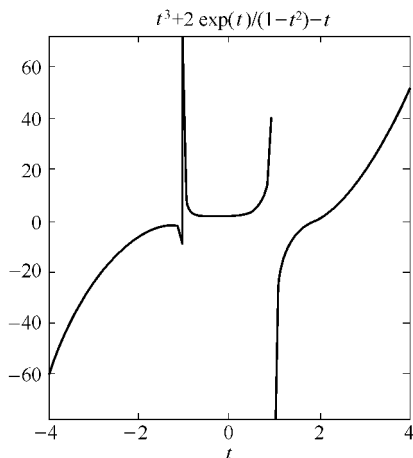


图 2-11 函数 $t^3 + \frac{2e^t}{1-t^2} - t$ 曲线

由例 2-57 可见,用 `ezplot` 绘制函数图非常方便,所以有时也把它称之为简易绘图指令,用它还可以绘制许多较为复杂的函数曲线,如下面的例题。

例 2-58 绘制叶形线 $u^3 + v^3 = 9uv$ 和三叶玫瑰线 $r = \sin(3t)$ (极坐标方程)。

解 (1) 绘叶形线。移项把方程变成 $f(u, v) = 0$, 即隐函数形式 $u^3 + v^3 - 9uv = 0$, 键入:

```
ezplot('u^3+v^3-9*u*v') (或 ezplot u^3+v^3-9*u*v)
```

回车得出图 2-12。

(2) 绘制三叶玫瑰线。把极坐标方程 $r = \sin(3t)$ 通过 $\begin{cases} x = r \cos t \\ y = r \sin t \end{cases}$ 转换成直角坐标方程:

$\begin{cases} x(t) = \sin 3t \cos t \\ y(t) = \sin 3t \sin t \end{cases}$ 键入:

```
ezplot('sin(3*t)*cos(t)', 'sin(3*t)*sin(t)', [0, pi])
```

或

```
ezplot sin(3*t)*cos(t) sin(3*t)*sin(t)
```

回车得出图 2-13。

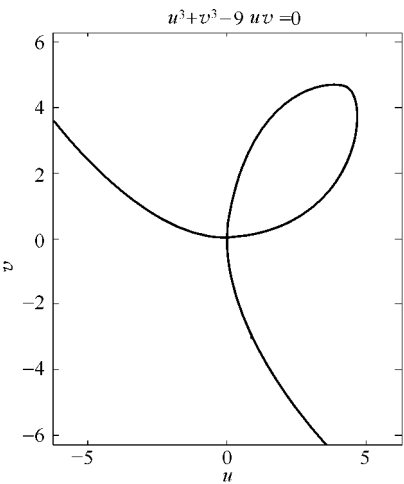


图 2-12 叶形线

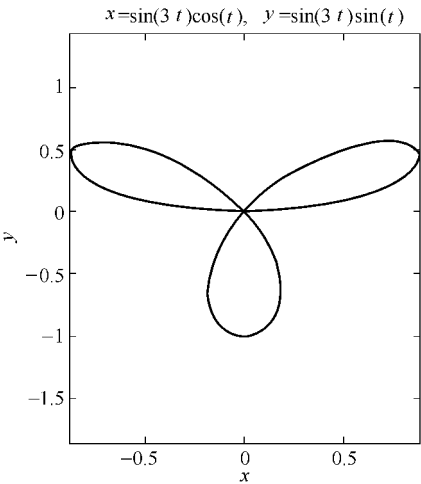


图 2-13 三叶玫瑰线

3. 特殊图形及其他坐标系绘图指令

MATLAB 中设有一些满足特殊需求的平面绘图指令 ,如表 2-10 和表 2-11 所示。还有许多平面绘图指令 ,可用 help 从“ graph2d ”图形库中查询。

表 2-10 绘制特殊图形指令表

指令格式	功 能	指令格式	功 能
stem (x,y)	脉 冲 图	bar (x,y)	条 形 图
stairs (x,y)	阶 梯 图	fill (x,y)	填 色 图
pie (x)	饼 状 图	plotyy (x1,y1,x2,y2)	双 y 坐标图

表 2-11 其他坐标系绘图指令

指令格式	功能或参量意义
polar (θ,ρ)	极坐标绘图 θ—极角 ρ—矢径
loglog(X,Y)	绘制以 log10 为坐标刻度的 X—Y 曲线
semilogx(X,Y)	绘制半对数刻度图 X 轴以 log10 刻度 Y 轴为线性刻度
semilogy(X,Y)	绘制半对数刻度图 X 轴为线性刻度 Y 轴以 log10 刻度

2.3.3 控制图形、画面的一些操作方法

MATLAB 语言中有一些控制画面的指令 ,可以使图形丰富多彩、绘图快捷方便。

1. 保留曲线指令 hold

在指令窗中键入一条画图指令后 ,如果后面键入 hold on (或 hold) 就会使画出的曲线得以保留。此后键入的画图指令 ,还将画在这一张图上 ,直到键入 hold off (或 hold) 。 hold 是一种“ 拉线开关 ”式指令 ,第二次键入 hold 相当于键入 hold off 。

2. 擦去画面上曲线的指令 `clf`

要想擦净图形窗中已有的曲线,可在指令窗中键入指令 `clf`,回车即可。

3. 加画坐标网格指令 `grid`

在指令窗中键入 `grid on`(或 `grid`),就会在图形窗中画上网格线。想去掉画好的网格,可键入 `grid off`(或再键入 `grid`)。 `grid` 也是一个“拉线开关”式指令。

4. 加注图名指令 `title(' ')`

该指令可将括号中英文格式单引号界定的标题、说明等文字加在图上侧。

5. 加注坐标轴名称指令 `xlabel(' '), ylabel(' ')`

该指令可将括号中英文格式单引号内的字符串,加注在相应坐标轴的侧旁。

6. 线型标注指令 `legend(' ', ' ', ..., k)`

同一幅图上绘制多条不同线型的曲线时,可以用该指令注明各条曲线代表的意义。该指令的输入参数用英文格式单引号界定、逗号分隔,单引号内可写入说明曲线意义的中文或英文注释,注释的排序与程序中绘图指令出现的先后顺序必须一致。


输入参数 k 若取 -1 标注写在图外右侧;取 0 则写在图中最佳位置;取 $1, 2, 3$ 和 4 则分别表示写在相应坐标象限的最外角,省略 k 时默认 $k = 1$ 。标注的文字框可以被拖动到任何位置。

如果只有一条曲线,在 `legend` 之后的参数不用写括号、引号。

要想隐去图中已有的线型标注说明,可键入指令 `legend off`。

7. 图形窗中菜单的应用

利用图形窗中菜单【Insert】下属的子菜单,可以在图上加画直线、箭头(`arrow`)、文字框(`textbox`)、矩形(`rectangle`)、椭圆(`ellipse`)等。

单击快捷按钮使其背景色变白,再单击菜单【Edit】,则可利用弹出菜单中的项目功能对画好的图形进行编辑、剪切、复制、粘贴、删除、加上或隐去标注,以及变换图形颜色、线形、线宽等。

8. 复制图形指令

下面介绍两个常用的方法,可以把 MATLAB 中绘制好的图形复制到文章中,即复制到 Word 中。

① 在图形窗中单击主菜单 Edit,再单击弹出的子菜单 Copy Figure,就将图送入了剪贴板,然后打开文字编辑窗口如 Word,粘贴到需要的地方;

② 将图形放大至全屏幕,按一下键盘中部的 `Print Screen` 键,则将图形送入了剪贴板,然后粘贴到需要该图的地方。

2.3.4 三维图绘制初步

MATLAB 绘制三维图的指令非常丰富,这里仅介绍最基本、最常用的 3 个三维绘图指令:三维曲线、三维网线和三维曲面(网面)绘图指令。

1. 三维数据绘图指令

和绘制平面图的指令 `plot` 类似,三维数据绘图指令也是根据大量数据描点绘制的。该指令绘图的基本原理是空间一点由三个坐标(x_k, y_k, z_k)确定。使用格式是:

```
plot3 (X,Y,Z, 'S')
```

```
plot3 (X1,Y1,Z1, 'S1', X2,Y2,Z2, 'S2', ..., Xn,Yn,Zn, 'Sn')
```

① 输入参量 $X_i, Y_i, Z_i (i=1, 2, \dots, n)$ 是同维向量时,由三个向量对应分量构成的数组(x_k, y_k, z_k)确定空间一个点,所有数据连成一条空间曲线;

② 输入参量 $X_i, Y_i, Z_i (i=1, 2, \dots, n)$ 是同阶矩阵时,三个矩阵相同位置对应元素确定空间一个点,三个相应列的数据确定一条空间曲线;

③ 输入参量 $S_i (i=1, 2, \dots, n)$ 为控制曲线特性(点、线、色)标识符,可以省略。

例 2-59 已知空间曲线方程 $\begin{cases} y = \sin x \\ z = \cos x \end{cases} x \in [0, 20]$ 绘制空间曲线,并画坐标网格。

解 键入:

```
x=0:0.05:20; y=sin(x); z=cos(x); subplot(1,2,1), plot3 (x,y,z, 'r. '), grid
```

回车得出图 2-14 的螺旋线。这里 x 分点间隔为 0.05,间隔变大时曲线的折线化程度明显。

例 2-60 绘制两条空间折线,第一条线上节点的空间坐标为(1 3 5), (2 5 1), (5 4 5), 第二条为(5 1 5), (5 4 4), (1 2 1)。

解 每条线上有三个折点,可用给出的数据构成三个 3×2 阶矩阵,再用 `plot3` 画图。

键入:

```
x1=[1 5;2 5;5 1]; y1=[3 1;5 4;4 2]; z1=[5 5;1 4;5 1];
subplot(1,2,2), plot3(x1,y1,z1, 'r'), grid, box
```

回车得出图 2-15。

注: `box` 是个“拉线开关”式指令,功能是在图上加画三维方框箱体,增加立体感。

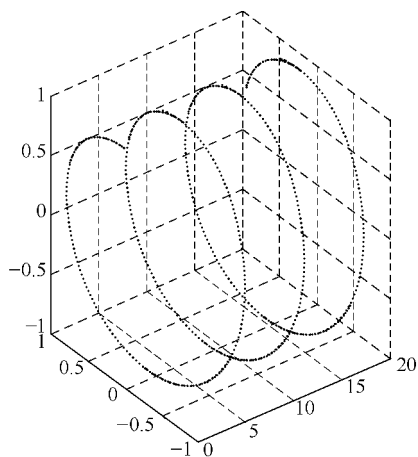


图 2-14 螺旋线图

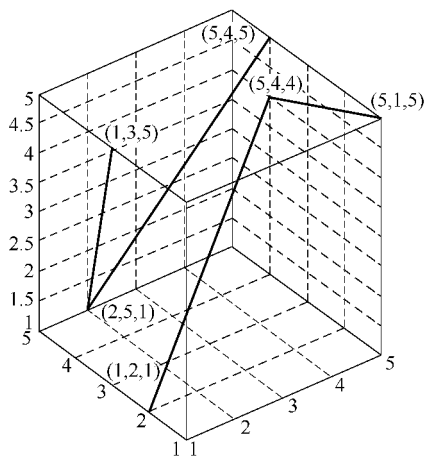


图 2-15 两条空间折线图

2. 绘制三维网格图指令

绘制三维网格图(或网线图)指令 mesh 画图的基本原理是:在 X-Y 平面上确定的绘图范围内,用与 X,Y 坐标轴平行的直线将 X-Y 平面分成方格,算出格点上的函数值 $z=f(x,y)$,把坐标 x,y,z 确定的空间点相连成图。基本调用格式有下列三种。

1) mesh(Z,C)

① 输入参量 Z 为 $m \times n$ 阶矩阵时,则绘出 $m \times n$ 个空间点的连线图。每个空间点的 Z 坐标值等于矩阵 Z 的一个元素值,该元素的列序数为其 X 坐标值,行序数为其 Y 坐标值,三个坐标值确定空间一点;

② C 是控制网线颜色的矩阵,较为复杂,通常被省略,默认为 $C=Z$ 。

2) mesh(x,y,Z,C)

① 输入参量 x,y 是向量,Z 是矩阵,它们满足: $\text{size}(Z) = \text{length}(y) \times \text{length}(x)$,即若 Z 是 $m \times n$ 阶矩阵时, $\text{length}(y) = m$, $\text{length}(x) = n$;

② 输出有 $m \times n$ 个格点的空间网格图,格点的 Z 坐标值为矩阵 Z 的元素值 z_{ij} ,格点的 X 坐标值为向量 x 的第 j (z_{ij} 的列序数)个分量,格点的 Y 坐标值为向量 y 的第 i (z_{ij} 的行序数)个分量;

③ C 是控制网格颜色的矩阵,被省略时默认为 $C=Z$ 。

3) mesh(X,Y,Z,C)

① 输入参量 X,Y,Z 是维数相同的三个数值矩阵;

② 矩阵 X,Y,Z 相当于三组独立的数据,三个矩阵对应元素的取值构成空间点的三个坐标,决定了一个空间格点,绘出连接所有格点的网格图;

③ C 是控制网线颜色的矩阵,较为复杂,被省略时默认 $C=Z$ 。

例 2-61 已知 $x=[1 \ 3]$, $y=[5 \ 2 \ 5]$, $Z=\begin{bmatrix} 4 & 1 \\ 2 & 3 \\ 0 & 5 \end{bmatrix}$,用 mesh(x,y,Z)作空间网格图。

解 键入:

```
x=[1 3]; y=[5 2 5]; Z=[4 1;2 3;0 5];
subplot(1,2,1), mesh(x,y,Z), box
```

回车得出图 2-16。

图 2-16 中标出了三个点的坐标: $(1, 2, 2)$, $(3, 2, 3)$, $(3, 5, 5)$, 仅对第一点作些说明。该点的空间坐标是 $(1, 2, 2)$, 在最靠近我们的竖直平面上。它的 Z 坐标 2 取自矩阵 Z 中元素 z_{21} , z_{21} 在 Z 中的列序号是 1, 所以它的 X 坐标取 x 向量中第一个分量值 $x_1 = 1$ 。 z_{21} 在 Z 中的行序号是 2, 所以它的 Y 坐标取 y 向量中第二个分量值 $y_2 = 2$ 。其他各点依此类推。

例 2-62 设 $X = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$, $Y = \begin{bmatrix} 2 & 1 \\ 3 & 5 \end{bmatrix}$, $Z = \begin{bmatrix} 2 & 3 \\ 5 & 1 \end{bmatrix}$, 用 mesh 指令绘制三维网格图。

解 键入:

```
X=[1 2;3 1]; Y=[2 1;3 5]; Z=[2 3;5 1];
subplot(1,2,2), mesh(X,Y,Z), box
```

回车得出图 2-17。

该网格图是由 4 个空间点 $(1, 2, 2)$, $(2, 1, 3)$, $(3, 3, 5)$ 和 $(1, 5, 1)$ 连线构成的。

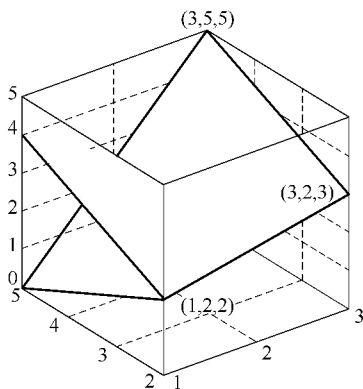


图 2-16 例 2-61 图

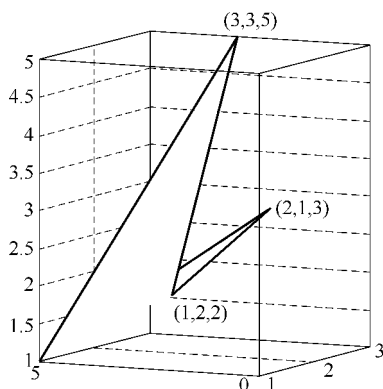


图 2-17 例 2-62 图

4) 用 mesh 指令绘制二元函数空间图

当 mesh 指令的输入参数 $Z = f(X, Y)$ 时, 可绘制出二元函数的空间网格图。这时矩阵 Z 的每个元素 $z_p = f(x_p, y_p)$, p 为序号。画图前, 先在 X - Y 平面上把空间曲面投影区域“分格”取点, 然后得出格点上对应的函数值 $z_{ji} = f(x_j, y_i)$, 再用 mesh 绘图, 绘图步骤如下所述。

第一步 投影区的分割取点

设已知二元函数 $z = f(x, y)$, $x \in [a, b]$, $y \in [c, d]$, 先用平面分割指令 meshgrid 对曲面在 X - Y 平面上的投影区域进行分格, 得出各格点上的坐标值 (x_j, y_i) , 它的调用格式为:

```
[X, Y] = meshgrid(x, y)
```

① 输入参数 x, y 为两个向量, 设 $\text{length}(x) = n$, $\text{length}(y) = m$, m 和 n 取决于分割 x 和 y 的步长。

② 输出参数 X, Y 为两个 $m \times n$ 阶矩阵, X 的各行相同, 每行都由向量 x 的分量组成; Y 的各列相同, 每列都由向量 y 的分量组成。

③ 输入参数中如果缺省 y , 即变成 $[X,Y]=\text{meshgrid}(x)$, 这时输出两个 n 阶方阵 $X=[x;x;\dots;x]_{n \times n}$, 而 $Y=X$ 。

第二步: 用 mesh 画空间网格图

① 据 $z=f(x,y)$ 写出 $Z=f(X,Y)$ 的函数表达式, 式中各变量得用数组算法符号连接;

② 键入 $\text{mesh}(X,Y,Z)$ 或 $\text{mesh}(Z)$ 画出函数网格图。

例 2-63 已知 $z=1-\frac{x^4}{9}-\frac{y^2}{4}$, $x \in [-1.5, 1.5]$ 和 $y \in [-1, 1]$ 。绘制空间曲面在 X - Y 平面上投影的网格图和空间曲面网格图。

解 键入:

```
x = -1.5:0.4:1.5; y = -1.0:0.3:1; [X,Y]=meshgrid(x,y); % 得出分
```

格矩阵

```
mesh(X,Y,zeros(length(y),length(x))),hold
```

回车得出:

```
Current plot held
```

画出图 2-18 中 X - Y 平面上的投影图并予保留。

再键入:

```
Z=1-X.^4/9-Y.^2/4, mesh(Z)
```

回车得出图 2-18 中空间曲面网格图。

例 2-64 已知双曲抛物面(鞍形曲面) $z=\frac{x^2}{2}-\frac{y^2}{3}$, $x \in [-2, 2]$, $y \in [-3, 3]$, 绘制网格图。

解 键入:

```
x = -3:0.2:3; y=x; [x1 y1]=meshgrid(x,y);
```

回车得出投影分格数据, 但不画图。

键入:

```
subplot(1,2,2), z=x1.^2/2-y1.^2/3; mesh(z), box
```

回车得出图 2-19。

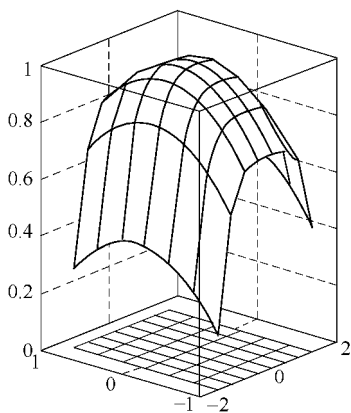


图 2-18 二元函数 $z=1-\frac{x^4}{9}-\frac{y^2}{4}$ 空间图

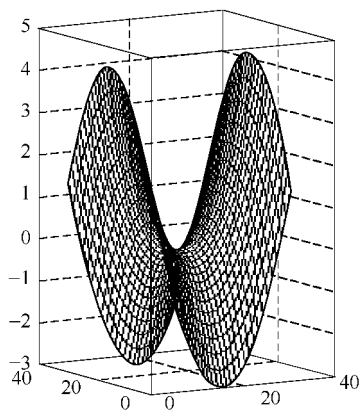


图 2-19 双曲抛物面(鞍形面)图

也可以键入 `mesh(x1,y1,z)` 或直接键入 `mesh(x1.^2/2 - y1.^2/3)`。

比较例 2-63 和例 2-64 的图可以看出,投影面上分格越多、格点越多,网格曲面越光滑。

3. 绘制空间曲面图指令

要想把用 `mesh` 画出的网格图中间的镂空部分涂上颜色使其立体感更强,可以用填色指令,但操作比较麻烦。下面介绍的指令则可在绘制空间网格图的同时,直接在网孔上涂色,它的颜色跟网线相同,构成带色小面元组成的空间曲面图。

该指令根据给定的数值矩阵,绘制出三维曲面(实际由许多网格面构成)彩色图,使用方法跟绘制三维网线图指令 `mesh` 完全一样,它的调用格式如下:

`surf(Z,C)` 矩阵 Z 中每个元素及其所在行、列数决定空间一点坐标

`surf(x,y,Z,C)` 向量 x,y 及矩阵 Z 中对应元素确定空间一点坐标

`surf(X,Y,Z,C)` 三个同阶矩阵 X,Y,Z 中对应元素确定空间一点。

该指令对输入参数的要求跟 `mesh` 指令中的一样,如当输入参量 x,y 为向量而 Z 为矩阵时,要求 `size(Z) = length(y) * length(x)`; X,Y,Z 的维数完全相同。 C 是决定每个网格面元色彩的矩阵,通常予以省略,默认色彩矩阵 $C = Z$ 。

例 2-65 用指令 `surf` 把例 2-62 的矩阵绘制成空间曲面图。

解 键入:

```
x=[1 2;3 1]; y=[2 1;3 5]; z=[2 3;5 1];
```

```
subplot(1,2,1), surf(x,y,z), box
```

回车得出图 2-20。

例 2-66 绘制旋转抛物面 $z = x^2/3 + y^2/4$ 曲面图 $x \in [-3, 3], y \in [-3, 3]$ 。

解 键入:

```
x = -3:0.3:3; y = x; [x1,y1] = meshgrid(x,y); 对投影平面分格。
```

键入:

```
subplot(1,2,2), z = x1.^2/3 + y1.^2/4; surf(z), box
```

回车得出图 2-21。

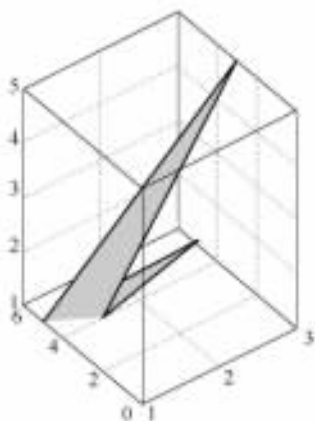


图 2-20 例 2-65 图

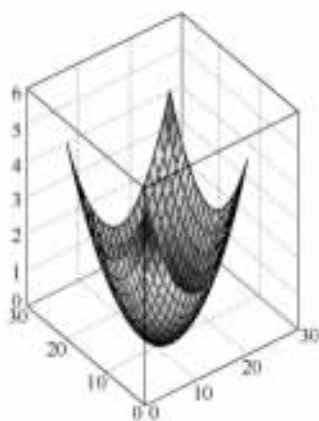


图 2-21 旋转抛物面

2.4 MATLAB 语言编程

MATLAB 是用 C、C++ 语言编写的,由于运算单元为矩阵,所以编程形式、程序结构和语法规则都比较简单。它是一种解释性编程语言,修改和调试也很方便。前边学习和使用的指令都是系统内部已有的 M-文件,如果自己有一些特殊需求,就应该自己动手编写程序,存盘后和软件中已有程序一样可以反复调用。

2.4.1 MATLAB 的编辑调试窗

虽然在指令窗中也能编程,但只能进行行编辑,编辑一些简单程序,无法存盘和反复调用,关机即逝。因此,多数程序得在编辑调试窗中进行,下边先介绍编辑调试窗。

1. 进入和退出编辑调试窗



在指令窗中单击快捷按钮  或依次单击菜单【File】→【New】→【M-File】,屏幕上就出现图 2-23 所示的编辑调试窗,各种程序的编辑、调试都可在其中进行。
















图 2-23 MATLAB 的编辑调试窗

退出编辑调试窗时单击窗口右上角的小图标 .

2. 编辑调试窗中的快捷按钮

在编辑调试窗内可以编辑、调试和修改任何 M-文件。窗内最上一行的主菜单有:【File】(文件)、【Edit】(编辑)、【Text】(正文)、【Debug】(调试)和【Help】(帮助)等,每个主菜单下含有若干子菜单,其使用方法与指令窗中相类似。主菜单行下面设有一排小图标,是一些常用子菜单的快捷按钮,使用它们比逐级调用子菜单方便快捷,可以提高编程效率、节省时间。现在把它们其中的 13 个列在表 2-14 中,同时列出了相应子菜单及其功能。

表 2 - 14 编辑调试窗中的快捷按钮与相应菜单功能对照表

按 钮	相 应 的 菜 单		功 能
	主 菜 单	子 菜 单	
	【File】(文件)	【New】/【M - File】	新建 M - 文件
		【Open】	打开存盘的 M - 文件
		【Save】	把编好的 M - 文件存盘
		【print】	打印
	【Edit】(编辑)	【Cut】	剪切
		【Copy】	复制选定的内容
		【paste】	粘贴剪下或复制的内容
		【Undo】	撤销刚才执行过的操作
		【Redo】	恢复刚撤销的操作
		【Find text】	查找
	Show function		显示函数
	【De bug】(调试)	【Set】/【clear breakpoint】	设置/取消断点
		【Clear all breakpoints】	取消所有断点


2. 4. 2 两类 M - 文件

 凡在编辑调试窗中用 MATLAB 语言编写的程序 , 统称 M - 文件 , 扩展名为“ . m ”。M - 文件分为两类 : M - 指令文件 (script file , 也叫脚本文件) 和 M - 函数文件 (function file) , 它们的修改、调试都应该在编辑调试窗中进行。

1. M - 指令文件

 M - 指令文件就是在编辑调试窗中用 MATLAB 语言编写的一连串 MATLAB 指令集合的总称。把编好的 M - 文件存盘后 , 在指令窗中一旦调用它相当于调用一批指令 , 执行完它的全部指令后才返回指令窗。指令文件的构成具有以下几个特点。

- ① 指令文件的第一部分往往是一些注释文字 (英语或汉语) , 说明文件功能和用法 , 它们的每行都以“ % ”符号开头。对于成段的注释文字不必逐行加写“ % ” , 可以用鼠标左键选中它们 , 然后右击 , 在弹出的选项中单击 comment (uncoment 为去除) 就会在每行文字前都加上“ % ”号。每行程序的末尾 , 也可写上以“ % ”符号开头的注释性文字。
- ② 指令文件的第二部分是程序主体 , 其中的变量都是全局变量 , 即它们被存放在工作空间中 , 可以被指令窗中键入的其他指令调用 , 同时指令文件也可以调用工作空间中已有的其他变量。要想使指令文件不使用工作空间中的变量 , 程序部分的第一行应写上清除变量指令“ clear ” , 含有画图指令的程序第一行还应写上清除图形指令“ cl f ”。

③ 文件编完后 , 单击存盘快捷按钮 , 给文件命名存盘。

下面通过实例 , 说明编制 M - 指令文件的步骤和格式。

例 2 - 67 编写一个 M - 指令文件 , 由键盘输入自变量矩阵 A 时 , 求出 A 中各元素 a 的函


数值 $y = f(a) = a^2 + \sin^3 a - e^{-a}$ 。

解 编写 M-指令文件一般都按下述步骤进行。

① 打开编辑调试窗。

② 在其窗口中编写程序。按照本题要求可键入：

```
%  求出矩阵 A 中各元素 a 的函数值：
%  函数为  $y = f(a) = a^2 + (\sin a)^3 - \exp(-a)$ 
clear      %  如果程序中有画图内容，还得加上 clf 指令擦去原有曲线
A = input('输入数值、向量或矩阵自变量 A = ') %  由键盘输入 A 的内容
f1 = 'A.^2 + sin(A).^3 - exp(-A) = '
disp([A.^2 + sin(A).^3 - exp(-A)])
```

③ 将上述程序存盘。单击编辑窗快捷键  在弹出的菜单【save file as】下面的文件名 (N) 栏内填写“li2_67”(也可另起它名)，单击右下方的【保存(s)】；

④ 退出编辑窗，回到指令窗。

这时在指令窗中如果键入 li2_67，(或在编辑调试窗中依次单击【debug】→【Run】) 回车就得出：

输入自变量 A =

然后 输入：

```
[3 * pi 5 2 ; 7 10 5]
```

回车得出：

```
A =
    9.4248    5.0000    2.0000
    7.0000   10.0000    5.0000

f1 =
A.^2 + sin(A).^3 - exp(-A) =
    88.8264    24.1115    4.6165
    49.2827    99.8389    24.1115
```

注 这里 li2_67 是 M-指令文件名称，A 是键盘输入的数值、向量或矩阵，它是运行程序所要求的，input 是键盘输入指令。存盘的 li2_67 是个永久指令，重新开机后仍可调用。这种方法可用于编辑经常使用的自定义函数。

2. M-函数文件

函数文件是在 MATLAB 编辑调试窗中编写的另一类 M-文件，每个函数文件定义了一个函数，它像通常意义下的函数一样，具有“输入自变量便输出确定的函数值”这种函数功能。MATLAB 中的大部分指令都是函数文件名，如 sin, log 等，函数文件的结构有下述特点。

① M-函数文件的第一部分往往是每行前带有 % 的说明文字，然后是程序部分，程序部分的第一行必须以 function 开头，然后用函数的通用形式 $y = f(x)$ 规定输入变量 x、输出函数 y 和函数名 f，x 和 y 均不限于一个变量，这里确定的函数名 f 就是存盘时的 M-函数文件名。

② 函数文件中的所有变量均为局部变量,即只在该文件内部起作用,不放入工作空间,即使在工作空间中有与它同名的变量,也被视为两个不同的变量。

③ 为了使 M-函数文件中的局部变量成为全局变量,使其在主、子程序(指令窗中键入的程序为主程序,M-文件为子程序)中都能通用,编程时可用指令 `global` 定义它们,格式为 `global a1 a2 a3...`,这样便可使函数文件中的局部变量 `a1, a2, a3...` 成为全局变量,和工作空间中的其他变量一视同仁,同名等价。

调用 M-函数文件时,必须得有输入变量(函数的自变量)。

注 程序运行中经常出现的几个警示语句 `error('message')`——显示出错信息,中止程序的运行 `warning('message')`——显示警告信息,程序继续运行。

下面通过实例,说明编制 M-函数文件的步骤和格式。

例 2-68 编写 M-函数文件,由它求出矩阵 `A` 中各元素 `a` 的函数值 $y = f(a) = a^2 + \sin^3 a - e^{-a}$ 。

解 编写 M-函数文件一般都按下述步骤进行。

① 打开编辑调试窗。

② 在其窗口中编写程序。

```
% 求出矩阵 A 中各元素 a 的函数值,函数为:
% y = f(a) = a^2 + (sin a)^3 - exp(-a)
function f1 = li2_68(A) % 可以有多个输入参量
f1 = A.^2 + sin(A).^3 - exp(-A);
```

③ 将上述程序存盘:与例 2-67 中的步骤③相同,但文件名只能用 `li2_68`。

④ 退回到指令窗:与例 2-67 中的步骤④相同。

在上述程序中的 `f1 = li2_68(A)` 中,`f1` 为输出的函数值,`li2_68` 为 M-函数文件名,`A` 为函数的自变量。

这时若欲求 $A = \begin{bmatrix} 3\pi & 5 & 2 \\ 7 & 10 & 5 \end{bmatrix}$ 中各元素 `a` 为自变量的函数 $y = f(a) = a^2 + \sin^3 a - e^{-a}$ 取值,可在指令窗中键入:

```
y = li2_68([3 * pi 5 2; 7 10 5])
```

回车得出:

```
y =
    88.8264    24.1115    4.6165
    49.2827    99.8389    24.1115
```

2.4.3 关系和逻辑运算

实用中的程序要比例 2-68 复杂得多,这就需要学习专用的编程语言和规律。下面先介绍编程中常用于判断两个变量间关系的关系运算和逻辑运算。

1. 关系运算

两个变量或数值之间进行大小比较,称为进行关系运算。关系运算的结果是二值逻辑

量 ,它只能取 1 或 0 ,其中 1 表示真(T) 0 则表示假(F)。两个同维矩阵间的关系运算规定为它们对应元素间的关系运算 ,运算结果仍是一个矩阵 ,与参与运算矩阵的维数相同 ,是个布尔矩阵(元素值只取 0 或 1 的矩阵)。

MATLAB 中允许“数”跟“矩阵”进行关系运算 ,规定为数与矩阵的每个元素进行关系运算 ,运算结果是一个与矩阵维数相同的布尔矩阵。

关系运算符及其意义列在表 2 - 15 中。

表 2 - 15 关系运算符及其意义

符 号	意 义	符 号	意 义	符 号	意 义
=	相 等	<	小 于	>	大 于
~ =	不 相 等	< =	小于等于	> =	大于等于

例 2 - 69 已知 $a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$, $a2 = 5$ 。求 $a3 = a \geq a2$ (注意 : = 是赋值号)的运算结果。

解 键入 :

```
a1 = [1 2 3 ; 4 5 6 ; 7 8 9 ] ; a2 = 5 ; a3 = a1 >= a2
```

回车得出 :

```
a3 =  
    0    0    0  
    0    1    1  
    1    1    1
```

2. 逻辑运算

两个逻辑量之间可以进行“与”、“或”和“非”三种基本逻辑运算及由它们组合而成的“异或(xor)”逻辑运算。在逻辑运算中 ,非零元素的逻辑量为 1 ,表示“真”(或 T) ;零元素的逻辑量为 0 ,表示“假”(或 F)。逻辑运算符和意义列在表 2 - 16 中。

表 2 - 16 逻辑运算符及其意义

符 号	意 义	符 号	意 义
&	与	~	非
	或	xor	异 或

逻辑运算的结果仍然是逻辑量 0(假)或 1(真)。维数相同的矩阵进行逻辑运算时 ,定义为它们对应元素逻辑量间的逻辑运算结果。

MATLAB 中允许数与矩阵间进行逻辑运算 ,规则与关系运算相同 ,是数与矩阵各个元素间的逻辑运算。

设 A , B 为两个逻辑量 ,对它们进行逻辑“与”、“或”、“非”和“异或”运算时 ,所得结果列

于表 2-17 中。

表 2-17 逻辑运算真值表

逻辑量及其运算	A	B	$\sim A$	$\sim B$	$A \& B$	$A B$	$\text{xor}(A, B)$
真 值 (逻辑量)	1	1	0	0	1	1	0
	1	0	0	1	0	1	1
	0	1	1	0	0	1	1
	0	0	1	1	0	0	0

例 2-70 求数值矩阵 $a = \begin{bmatrix} 1 & 0 & -5 & 0 & 9 \\ 3 & -2 & 0 & 6 & 0 \\ 0 & 0 & 5 & 7 & 8 \end{bmatrix}$ 的“非”， a 和 0 的“异或”。

解 键入：

$a4 = [1 \ 0 \ -5 \ 0 \ 9; 3 \ -2 \ 0 \ 6 \ 0; 0 \ 0 \ 5 \ 7 \ 8]; a5 = \sim a4, a6 = \text{xor}(a4, 0)$

回车得出：

$a5 =$

```
0    1    0    1    0
0    0    1    0    1
1    1    0    0    0
```

$a6 =$

```
1    0    1    0    1
1    1    0    1    0
0    0    1    1    1
```

2.4.4 程序结构

组成计算机程序的一系列指令语句可以分成两类：运算语句和控制顺序语句。MATLAB 编程的主要任务是安排调整好控制顺序语句，使运算语句的运行顺序合理，运算简捷，节省机时。常用的控制顺序语句有三种：顺序结构、循环结构和分支结构。顺序结构语句就是从前到后，依照指令排列的自然顺序逐条执行；循环和分支结构语句则不全按指令语句的自然排序执行，中途可以利用控制顺序语句改变执行次序，它们是学习编程中需要重点关注的内容，下面分别加以介绍。

1. 循环结构语句

运算中经常碰到一些有规律的重复演算，反映在程序中则需要多次反复执行一组语句，即循环使用这同一组指令，直到满足某种条件（循环结束条件）。为此，MATLAB 中设有两种循环结构语句，可根据需要选用。

1) 条件循环语句 while-end

该语句的使用格式、特点见表 2-18，流程如图 2-24。

逻辑关系式表示出执行语句组的条件为：逻辑关系式得 1（真）时，往下执行语句组；一旦

条件不再成立,即逻辑关系式得 0(假),则跳出 while-end 循环,执行后面的程序。

表 2-18 条件循环语句格式及特点

结构格式	while 逻辑关系式 语句组
功能特点	end 规定了执行“语句组”的 条件

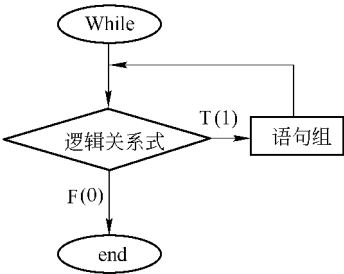


图 2-24 条件循环语句流程图

例 2-71 求满足 $\sum_{n=1}^m n^2 > 10^4$ 的最小整数 m 。

解 由于 $\sum_{n=1}^m n^2 = 1 + 2^2 + 3^2 + \dots + m^2$, 用条件循环语句 while-end, 在编辑调试窗中键入:

```
clear
a=0; n=1;
while a<1e4           % 规定继续执行语句组的条件
    a=a+n^2; n=n+1;    % 语句组
end, m=n-1, a          % 结束循环,显示 m, a 的得数
```

顺序单击菜单【Debug】→【Save and Run】(存盘并运行。若是 M-函数文件,必须将存盘和运行分两步分别操作,因为运行时得输入自变量),在指令窗中得出:

```
m =
    31
a =
   10416
```

2) 次数循环语句 for-end

该语句的使用格式、特点见表 2-19, 流程如图 2-25。

表 2-19 次数循环语句格式及特点

结构形式	for 循环次数表达式 语句组
功能特点	end 规定了执行语句组的次数

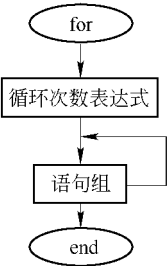


图 2-25 次数循环语句流程图

例 2-72 创建一个范德蒙矩阵 $D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 5 & 2 & 3 \\ 4^2 & 5^2 & 2^2 & 3^2 \\ 4^3 & 5^3 & 2^3 & 3^3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 5 & 2 & 3 \\ 16 & 25 & 4 & 9 \\ 64 & 125 & 8 & 27 \end{bmatrix}$ 。

解 根据矩阵各行元素值之间的特殊关系,可在编辑调试窗中键入下列程序:

```
clear
a = input('输入范德蒙矩阵的基本向量 a = ');
n = length(a);
for k = 0 : n - 1           % 规定执行语句组的次数
    D(k + 1, :) = a.^k;    % 给矩阵 D 的第(k + 1)行赋值
end, D
```

给文件起名“li2_72”并存盘。
在指令窗中键入 li2_72 回车则屏幕显示：
输入范德蒙矩阵的基本向量：

```
a =
```

这时由键盘输入[4 5 2 3] ,回车得出：

```
D =
    1     1     1     1
    4     5     2     3
   16    25     4     9
   64   125     8    27
```

2. 分支结构(条件转移)语句

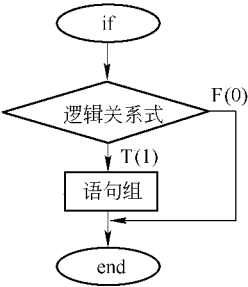
复杂的计算中 ,有时需要根据表达式满足的条件 确定下一步进行哪一项工作。为此 , MATLAB 中设有分支 (条件)结构语句。

1) 单条件分支语句 if-end

该语句的使用格式、特点见表 2-20 ,流程如图 2-26。

表 2-20 单条件分支语句格式及特点

结构形式	if 逻辑关系式
	语句组
	end
功能特点	规定了执行语句组的条件



2-26 单条件分支语句流程图

逻辑关系式结果为 1 时 ,往下执行语句组 ;否则跳过语句组 ,直接以 end 为出口。
例 2-73 编个程序 列表显示 $x=0\ 0.3\ 4$ 、函数 $y = x^2 - 2\sin x$ 小于零的值 并显出表头。
解 利用次数循环语句 for-end 和 if-end 结合 编出如下程序：

```
clear, disp('    x    y = x^2 - 2 sin x')    % 显示函数表的表头
for x = 0 : 0.3 : 4                            % 规定执行语句组的次数
    y = x^2 - 2 * sin(x);                      % 计算函数值语句组
    if y < 0
        disp([x, y])                          % 显示 x 和 y 为负数的
```

数据

end
end

2) 双条件分支语句 if-else-end

使用格式、特点见表 2-21 流程如图 2-27。

表 2-21 双条件分支语句格式及特点

结构形式	if 逻辑关系式 语句组 A else 语句组 B end
功能特点	规定了执行语句组 A 的条件； 若不满足 就执行语句组 B

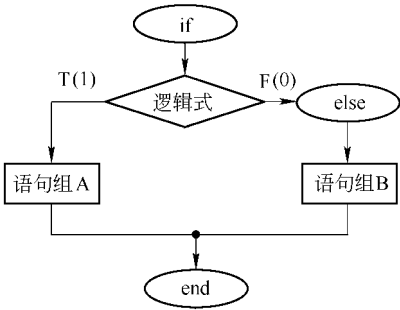


图 2-27 双条件分支语句流程图

逻辑关系式结果为 1 时 执行语句组 A ,为 0 时执行语句组 B。

例 2-74 编程执行下述任务 输入一个数 n ,若使函数 $y = n^2 - n^3 \sin n > 100$,就显示“y > 100 ” ;否则显示“y ≤ 100 ”。

解 在指令窗编程如下：

```
clear , n = input('n = ')  
if n^2 - n^3 * sin(n) <= 100  
  'y < =100 '  
else  
  'y >100 '  
end
```

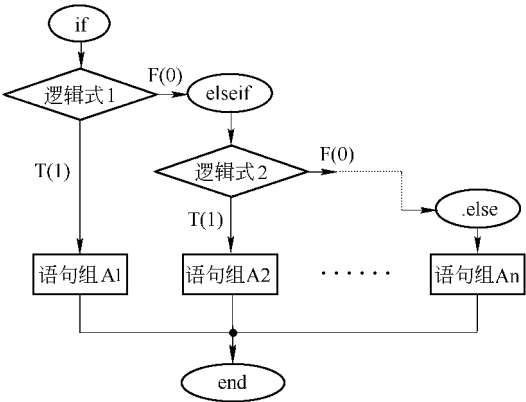
3) 递阶选择分支语句 if-elseif-else-end

有的运算关系较为复杂 ,大条件下还有小条件 ,要求根据分层条件确定运算顺序。这种分层条件语句叫递阶选择分支语句 ,可以实现多路选择 ,其使用格式、特点见表 2-22 流程如图 2-28。

该指令方便于 M-文件的调试修改。

表 2-22 递阶选择分支语句格式及特点

结构形式 (可以接连不断使用好多个 elseif)	if 逻辑式 1 语句组 A1 elseif 逻辑式 2 语句组 A2 elseif ... else 语句组 An end
------------------------------	----------------------------------------------------------------------------------------



例 2-75 由键盘输入任一自变量时,判断函数 $y = x^4 - 19x^3 + 32$ 是负数、偶数,还是其他。

解 先判断函数 y 的正负,再判断其奇偶性,据此在编辑调试窗中键入:

```
clear, x = input('输入自变量 x = ') % 由键盘输入自变量 x
if x^4 - 19 * x^3 + 32 < 0 % 判断函数 y 是否为负数
    a = 'negative'
elseif rem(x^4 - 19 * x^3 + 32, 2) == 0 % 判断 y 是否为偶数
    a = 'even'
else
    a = 'other'
end
```

循环和分支语句可以互相嵌套,形成更为复杂结构的程序语句。例 2-76 就是一个这类嵌套结构的简单例子。

例 2-76 鸡兔同笼,共有 36 个头、100 只脚,问鸡兔各有多少?

解 鸡有两只脚,兔有四只脚。若设鸡兔共有 p 个头、 q 只脚,又设有 n 只鸡,则 $(q - 2n)$ 应该能被 4 整除,其商数即为兔的只数。据此原理,在编辑调试窗中编程如下:

```
% 鸡兔同笼,总头数为 p,总脚数为 q,求鸡兔各有多少
clear, p = input('总头数 p = '); q = input('总脚数 q = ');
n = 1; % 设鸡起码有 1 只,可以多设
while 1 % while 后的逻辑值为 1 时将无限循环下去
    if rem(q - n * 2, 4) == 0 & (n + (q - 2 * n) / 4) == p
        break % 遇此指令则跳出包含它的最小循环语句
    end
    n = n + 1;
end,
disp('鸡的只数'), n, disp('兔的只数'), p - n
```

指令窗中调用该文件时输入 $p = 36$ $q = 100$,则得出鸡数 $n = 22$,兔子数 $= 14$ 。

最后的显示语句 `disp` 可以用 `sprintf` 代替,换成“`sprintf('总共有鸡兔 %d 只,它们共有 %d 只脚,则其中有 %d 只鸡, %d 只兔。', [p q n p - n])`”,这样可将全部内容都显示在同一行内。

注 程序的缩进格式有利于检查和修改。多次修改后若不呈现缩进格式,可用鼠标左键选中全部程序,右击后,再单击弹出菜单中的 `smart indent`,则自动调节成缩进格式。

4) 多路选择分支语句 Switch-case-otherwise-end

有些运算是根据满足的条件来选择运算步骤(用语句组实现)的,多路选择分支语句就满足了这种需求。该语句使用格式、特点见表 2-23,流程如图 2-29。

当表达式输出的数值(或字符串)与某个 case 后的数值 i (或字符串)一致时,就执行该 case 后的语句组 A_i ;若它跟任意 case 后的数值 i (或字符串)都不一致,就执行最后一个语句组 A_n 。

“多路选择分支语句”和“递阶选择分支语句”的功能类似。但是,该语句的选择不是逐

级向下进行的,而是由 switch 后的“表达式结果”一次确定的,表达式结果可以是数,也可以是字符串,它跟哪个 case 后面的内容一致,就执行哪个 case 后面的语句组。

表 2-23 多路选择分支语句格式及特点

结 构 形 式	switch 数值或字符串表达式 case 数值或字符串 1 语句组 A1 case 数值或字符串 2 语句组 A2 ... otherwise 语句组 An end
功能特点	根据满足的条件,选择执行相应的语句组

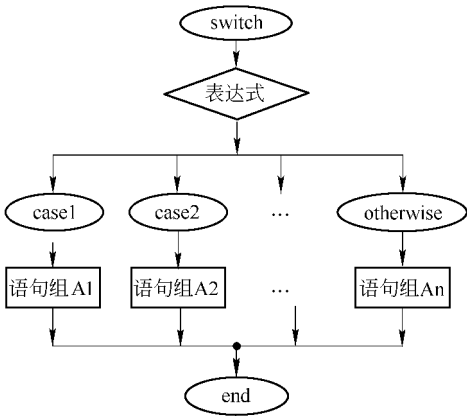


图 2-29 多路选择分支语句流程图

例 2-77 编程判断并显示输入自变量 x 的具体数值时,函数 $y = x^4 - 19x^3 + x + 32$ 是奇数、偶数(不分正负),还是其他数(视 0 为偶数)。

解 可以用多路选择分支语句,在编辑调试窗中编程如下:

```
% 多路选择分支语句
clear,x=input('x='); % 由键盘输入自变量 x
y=x^4-19*x^3+x+32 % 计算函数值
switch mod(y,2) % 得出 y 被 2 除的余数
case 1 % 余数为 1 时,y 是奇数
    disp('是奇数')
case 0 % 余数为 0 时,y 是偶数
    disp('是偶数')
otherwise
    disp('是非奇非偶数')
end
```

2.4.5 两类 M-文件的转换

两类 M-文件可以互相转换,通过例 2-78 可以了解它们间的异同和关系。

例 2-78 编制一个 M-指令文件,列出由键盘输入的正整数 a 到 b 之间的所有素数,并将它改编成 M-函数文件。

解 在编辑调试窗中键入:

% 列出从键盘输入的 a0 到 b0 之间的素数

```
clear,a0=input('a0=');b0=input('b0=');
a=ceil(a0);b=fix(b0);
```

```

k=0;
for m=a:b
    for n=2:m
        if rem(m,n)==0 % 检验 m 可否被 n 整除
            break
        end
    end
    if m==n % 选取只能被自己整除的数
        k=k+1;c(k)=n;
    end
end,c

```

编好后起名(如为“prinum”)存盘即可。

若把这个程序中带底纹的一句改成“function c = prinum(a0,b0)”,并去掉程序最后一行中的 c,就使这个 M-指令文件变成了 M-函数文件,函数文件在调用时必须得有输入参数。

2.4.6 编程中的一些控制指令

编写 M-文件时常常需要设置一些查看、检验、修改和人机对话方面的控制指令,它们对文件的编辑、使用,特别是对文件的调试和修改很有好处,下面介绍一些这方面的指令。

1. 键盘输入指令 input

程序执行到该指令时暂停运行,等待由键盘输入数据,输完数据回车后再继续运行后面的程序。该指令有两种常用格式。

```
a=input('Please input a number :')
```

程序执行到此暂停运行并显示出 Please input a number:,可按提示由键盘输入数值、表达式或字符串,回车后便将输入数据赋给数值变量名 a,然后继续往后运行。

```
a=input('Please input a number :','s')
```

程序执行到此暂停并显示出 Please input a number:,由于加有's',此后由键盘输入的任何数据(数值或字符)均被视为字符串,并赋给字符变量名 a。

2. 暂停运行指令 pause

程序执行中遇到 pause 指令时就暂停运行,等待用户按动键盘任意键后才继续运行。如果加有参数,格式为 pause(n),则暂停 n 秒后继续运行。

3. 人机切换指令 keyboard

程序运行到此便暂时停止,切换到指令窗中的行编辑运行方式,可由键盘输入指令,对程序进行操作、检查、调试或修改。直到键入指令 return,回车后,程序才继续运行。

4. 程序显示指令 echo

该指令方便于 M-文件的调试修改。


① 通常运行 M-文件时,屏幕上并不显示运行程序的内容。若在运行程序前编入“echo M-文件名”则可使其后 M-文件的编程内容在运行的同时显示在屏幕上。

② 在程序中间加有指令 echo on, 可使其后的程序在运行的同时被显示出来;其后若加有 echo off, 则继续运行而不再显示程序。

5. 中止循环指令 break

该指令可使正常的循环到此中止,跳出包含它的最小循环体。

6. 设置/取消程序中的断点

光标置于程序的某行上,单击快捷按钮, 则该行最左边呈现出一个红点,即设置了一个断点。当程序运行到该行时,就暂停运行,并在指令窗屏幕上出现字母 k, 同时调出编辑调试窗,显示出 M-文件的程序,以便调试修改。

练习题 2.4

1. 已知 $\mathbf{a} = \begin{bmatrix} 2 & 8 & 5 & 0 \\ 3 & 4 & 2 & 7 \\ 1 & 6 & 1 & 4 \\ 9 & 1 & 0 & 3 \end{bmatrix}$, $\mathbf{a} = \mathbf{a}^T$, 求 \mathbf{a} “小于等于” \mathbf{a} 的关系运算结果;并求 \mathbf{a} 与

其左旋 180 度后的矩阵“与或”后的结果。

2. 求第 1 题中 \mathbf{a} 与 $(\mathbf{a} - \mathbf{a}^{-1})$ 进行“与”、“或”的结果。

3. 编程序计算: (1) $N_1 = \sum_{n=1}^{20} n$; (2) $N_2 = \sum_{n=1}^{20} n!$; (3) $N_3 = \sum_{n=0}^{63} 2^n$ 。

4. 根据下述程序回答问题:

```
disp('    x    y=x^2 - 2sin x')
for x=1: 5
    y=x^2 - 2* sin(x);
    disp([x,y])
end
```

① 程序中 x 是否为向量? y 的表达式中 x^2 是否写错? 为什么不写成 $x.^2$?

② 如果去掉 y 表达式末尾的分号, 运行结果会有何变化?

③ 把 $\text{disp}([x,y])$ 移至 end 之后, 运行结果会有何变化?

5. 编一个 M-函数文件, 用以创建范德蒙矩阵 $\mathbf{D} = \begin{bmatrix} 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 4^2 & 4^3 \\ 1 & 5 & 5^2 & 5^3 \\ 1 & 2 & 2^2 & 2^3 \\ 1 & 3 & 3^2 & 3^3 \end{bmatrix}$ 。

6. 已知 $f(x) = \begin{cases} x^2, & x < 1 \\ 2x^{2.17} - 1, & 1 \leq x < 10 \\ 3x^2, & x \geq 10 \end{cases}$, 画图并编程求出当 $x = -3.8$ 和 54 时 $f(x)$ 的值。

7. 求出从 86.4 到 157.9 之间的所有素数。

8. 编程计算两个内维数相等的模糊矩阵(元素值在 0 与 1 间的矩阵)之间的 $\max - \min$ 合成:与普通矩阵乘法一样,只是把“矩阵乘法”中对应元素“相乘”改为“取小”,“相加”改为“取大”。

第 3 章 误差和 MATLAB 的计算精度

把客观事物抽象化,建立起数学模型,再对数学模型进行数值计算,这个过程中始终存在着误差问题。本章介绍误差的基本理论,着重介绍数值计算中的误差问题及 MATLAB 在数值计算中的精度选取,最后介绍设计算法时应该注意的事项。

3.1 误差

用数学方法解决实际问题的过程中,数据和客观事物之间总会存在差异,把这种差异称为误差。客观存在的误差只能减少,不能根除。为了尽量减少误差,就要了解误差的性质。

3.1.1 误差的来源

解决科学和工程问题时,常常需要建立描述事物变化规律的数学模型,并对其中的各个物理量进行测量,然后进行数值计算,这些过程中都会产生误差。从产生误差的原因上,可以将误差分成下列几类。

1. 模型误差

把事物变化的规律、特性数学化,即对客观事物进行抽象、提取、简化,表述成数学公式模型的过程中,客观实际和数学模型之间的差异,称为模型误差。

例如,从距地面高 s 处落下的物体,下落时间 t 和下落距离 s 间的关系,在忽略阻力等外界影响时,可以近似地用自由落体规律描述成 $s(t) = gt^2/2$ (g 为重力加速度)。由这个数学模型算出的 $s(t)$ 和实际下落的距离 $\bar{s}(t)$ 的差值 $\bar{s}(t) - s(t)$ 就是模型误差。

2. 观测误差

数学模型中各个变量(物理、化学、机械、电气等)的观测数据与事物本身客观实际间的差异,称为观测误差。

如上述落体的例子中,测得的下落时间 \bar{t} 及距离 \bar{s} ,因受测量仪器、观测方法等条件的限制、影响,记录数据跟客观实际间总会存在差异,这种差异就属于观测误差。

3. 截断误差(方法误差)

数学模型的表述往往是很复杂的,就是一个简单的积分表达式,也未必能够求出它的精确结果,经常是只能用数值计算方法算得它的近似解。精确解析解和近似数值解之间的差异,称为截断误差(方法误差)。

例如,根据某个数学模型,需要计算一个无穷级数

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots$$

实际计算中,我们只能取前面的有限项,比如说取 m 项,这就产生了误差:

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} - \sum_{n=0}^{m-1} \frac{x^n}{n!} = \sum_{n=m}^{\infty} \frac{x^n}{n!}$$

这个误差就属于截断误差。

4. 舍入误差

运算过程中,每个变量都只能取有限位数字参与计算,经常用“四舍五入”或其他方法处理参与运算的数据,多次这样对数据进行“取舍”,必然使计算结果产生误差,这种误差称为舍入误差。

例如,数值计算中经常用到的无理数 e 、 π 等,都有无穷多位小数,运算中却只能取有限位小数,这样引起的误差就属于舍入误差。

模型误差和观测误差产生于建模和测量过程,即从客观实际到数学的映射过程,诸如怎样建立模型、建立什么样的模型及如何进行测量等,这些过程的工作多半属于各学科的专业问题,不属于纯数值计算所研究的范畴。数值计算中,着重研究的是根据数学模型计算时产生的截断误差和舍入误差。

3.1.2 有关误差的一些概念

在数值计算中,要用到许多有关误差的理论和概念,这里仅介绍几个常用的基本概念和基本术语。

1. 绝对误差和绝对误差限

设某个量的准确值为 x^* ,算得的近似值为 x ,则它们之差 $x^* - x$ 叫作近似值的绝对误差 (absolute error,它并不是误差的绝对值),简称误差,记为:

$$ae(x) = x^* - x$$

这样定义误差之后,就可得出这个量的准确值 $x^* = x + ae(x)$ 。由于误差有正有负,而且它的大小是基于一个无法得到的量——准确值 x^* ,因此也就无法得到确切量 $ae(x)$ 。但是,根据计算时的具体情况,可以估计出这个值的大体范围 s ,即:

$$|ae(x)| = |x^* - x| \leq s$$

s 叫近似值 x 的绝对误差限,由它可以知道准确值 x^* 的取值范围 $x - s \leq x^* \leq x + s$ 。于是,准确值 x^* 也可以用近似值 x 和绝对误差限 s 表示为:

$$x^* = x \pm s$$

例如,通常将光速记为 $c = (2.997\,925 \pm 0.000\,001) \times 10^{10}$ cm/s,这表示公认的光速近似值为 $2.997\,925 \times 10^{10}$ cm/s,其误差限是 10^4 cm/s。

2. 相对误差和相对误差限

绝对误差无法反映近似程度的好坏,例如算得一个数据为 1 000 时,其误差限是 4,而算

得另一个数据为 100 时,其误差限是 1,虽然后者的误差限小,但是在数据中所占的比例却比前者大,近似值的精确程度远小于前者。为反映数据的真实精确程度,定义绝对误差与准确值的比为近似值的相对误差(relativistic error):

$$\text{re}(x) = \frac{\text{ae}(x)}{x^*} = \frac{x^* - x}{x^*}$$

实际计算中,准确值 x^* 总是无法知道的,所以在 $\text{ae}(x)$ 较小时把相对误差记为:

$$\text{re}(x) = \frac{\text{ae}(x)}{x} = \frac{x^* - x}{x}$$

近似值的相对误差越小,它的精确程度越高。例如,前面举例中数据 1 000 和 100 的相对误差分别为 $1/250$ 和 $1/100$,显然前一个数据精确。

跟误差一样,相对误差可正可负,同样不可能准确地得出,只能估计它的大小范围。如果确定出一个适当小的正数 se ,使得总满足下述关系:

$$|\text{re}(x)| = \left| \frac{\text{ae}(x)}{x} \right| = \left| \frac{x^* - x}{x} \right| \leq se$$

则称 se 为近似值 x 的相对误差限。

实际上,绝对误差 $\text{ae}(x)$ 常常用绝对误差限 s 代替。当相对误差 $\text{re}(x)$ 较小时,可用下式计算相对误差限:

$$se = \frac{s}{|x|}$$

在数值计算中,常常不说计算误差(无论是相对还是绝对误差),而是说估算(或估计)误差,也就是确定误差限,误差限才具有实际意义。

3. 有效数字

实际计算中,常常从认为是精确数据 x^* 的许多位数字中,用“四舍五入”的方法取前面的有限位 x 为近似值。这时,近似值 x 的绝对误差限,不会超过它末位数的半个单位。如果设精确值 $x^* = e = 2.718\,281\,828\,459\,045\dots$,按照“四舍五入”的方法,则:

若取 $x_1 = 2.718\,28$,则绝对误差 $\text{ae}_1(e) = 0.000\,001\,8\dots$,绝对误差限 $s_1 = 0.000\,005$;

若取 $x_2 = 2.718\,3$,则绝对误差 $\text{ae}_2(e) = 0.000\,021\dots$,绝对误差限 $s_2 = 0.000\,05$ 。

通常,如果近似值 x 的误差限 s 是某一位数上的半个单位,从该位数到 x 的左数第一位非零数共有 n 位,则称 x 有“ n 位有效数字”。上面的 x_1 有 6 位有效数字, x_2 有 5 位有效数字。由于有效数字和绝对误差限有上述关系,所以数值计算中常说一个量的有效数字是几位,就意味着说出了它的绝对误差限。

有效数字和绝对误差限之间的关系,可用下述的一般性关系式表达。设精确数据 x^* 的近似值 x 可以表示为:

$$x = 0.d_1d_2d_3\dots d_n\dots \times 10^p$$

其中 $d_1, d_2, \dots, d_n\dots$ 是 0 到 9 中的一个整数,且 $d_1 \neq 0$ 。

设其绝对误差为:

$$|\text{ae}(x)| = |x^* - x| \leq \frac{1}{2} \times 10^{p-n}$$

则称 x 有 n 位有效数字,其绝对误差限为:

$$s = \frac{1}{2} \times 10^{p-n}$$

如某个数据的近似值 $x = 0.23157 \times 10^{-2}$, 它有 5 位有效数字, 就是说, 它的绝对误差限是

$$|x^* - x| \leq s = \frac{1}{2} \times 10^{-2-5} = \frac{1}{2} \times 10^{-7}.$$

3.2 MATLAB 中的数值计算精度

当今的数值计算都是在计算机上进行的, 首先介绍计算机是如何表示数和实施数值计算的。

3.2.1 浮点数及其运算特点

为了使一个数值的数量级一目了然, 科学计算中常常用 10 的指数决定小数点的位置, 而小数点的位置可以浮动, 如 0.0032416 和 852.176 可以分别表示成:

$$0.32416 \times 10^{-2} \text{ 和 } 0.852176 \times 10^3$$

把这种允许小数点浮动的表示数字方法, 称为浮点表示法, 这样的数称为浮点数, 它是数字计算机中通用的数字表示方法。浮点数的一般表示形式为:

$$x = \pm (0.d_1d_2d_3\dots d_t)\beta^p = \pm (d_1\beta^{-1} + d_2\beta^{-2} + \dots + d_t\beta^{-t})\beta^p$$

其中: ① β 为浮点数的基底, 根据数的进制取值: 数是十进制时 $\beta = 10$; 是二进制时 $\beta = 2$; 是十六进制时 $\beta = 16$;

② p 为浮点表示的阶码, 它有随计算机而异的下限 L 和上限 U , 即 $L \leq p \leq U$ 。

③ $0.d_1d_2d_3\dots d_t$ 称浮点数的尾数。 $d_1, d_2, d_3, \dots, d_t$ 均为正整数, 规定当 $x \neq 0$ 时, $d_1 \neq 0$, 以便保证浮点数表示式的唯一性, 这样的浮点数称为规格化浮点数。

④ t 是用正整数表示的计算机字长, d_t 满足下述关系: $0 \leq d_i \leq \beta - 1, i = 1, 2, \dots, t$ 。例如, 对于十进制中的数 0.0987450 和 2843.54912, 其 $\beta = 10, 0 \leq d_i \leq \beta - 1 = 9, i = 1, 2, \dots, t$, 可分别表示为浮点数 0.98745×10^{-1} 和 0.284354912×10^4 。

计算机中进行数量级(阶码)不同的两个数相加减时, 先将阶码统一为较大者, 然后将尾数相加减。如设 $x = 0.3127 \times 10^{-6}, y = 0.4153 \times 10^{-4}$, 则:

$$x + y \approx 0.0031 \times 10^{-4} + 0.4153 \times 10^{-4} = 0.4184 \times 10^{-4}$$

$$\text{又如 } 0.8961 \times 10^3 + 0.4688 \times 10^{-5} \approx 0.8961 \times 10^3 + 0.0000 \times 10^3 = 0.8961 \times 10^3$$

计算结果表明大数“吃掉”了小数。

计算机中进行浮点数运算时, 实数加法的结合律、乘法对加法的分配律都不成立。

3.2.2 MATLAB 中的数值计算精度

1. MATLAB 中的三种运算精度

MATLAB 中的数值运算, 有三种不同精度可供选择, 它们分别如下所述。

- ① 数值算法——把每个数值都取 16 位有效数字 ,按浮点运算规则进行计算 ,得出结果的近似值最多有 15 位小数 ,是运算速度最快的一种算法。
- ② 符号算法——把每个数据都变换成符号量 ,按有理数计算方法对它们进行运算 ,可得出精确结果 (有理表达式) ,但占空间多、运算速度慢。
- 符号算法得出的精确结果并不实用 ,特别是当它为冗长的有理表达式时。因此 ,往往用转换变量指令 `vpa(a,m)` ,把符号量 `a` 转换成 `m` 位有效数字的近似值 ,省略 `m` 时 ,转换成默认精度 (32 位有效数字) 的近似值 ,该值形式上是数字 ,但它仍是符号量。
- ③ 可控精度算法——该方法介于上述两种算法 ,用控制精度指令 `digits(n)` 可使此后的运算均以 `n` 位有效数字进行 ,直到输入新的控制精度指令。

2. MATLAB 中的数据显示格式


- MATLAB 中数值数据的运算和存储通常都是十进制的 16 位数 (二进制双精度) ,但是显示和打印的输出格式却有 10 种之多 ,见表 3-1。常用控制数据输出格式 (简称数显格式) 的方法有下述两种 :
- ① 在指令窗中依次单击【File】→【Preferences】 ,在弹出的对话框 Text display 内 ,从列表框 Numeric format 的选项区域 10 种数显标识符中选定一种 ,单击  按钮 ,则此后输出的数据都按选中的标识符显示 ;
- ② 在指令窗中键入格式显示指令 `format + 数显标识符` ,此后则按此格式显示数据。数显格式符列在表 3-1 中。

表 3-1 接在 format 之后的数显标识符及其意义

数显标识符	显 示 结 果	举例 (以数值 4π 为例)
short	四位小数定点数	12. 566 4
short e	五位浮点数	1. 256 6e +001
short g	最优化短格式小数 ,最大显五位数	省略小数部最右的零 ,根据数据选用定点或浮点数表示
long	十四位小数定点数 (大于 100 的整数或小于 0. 001 的小数 ,用浮点数表示)	12. 566 370 614 359 17
long e	十六位浮点数	1. 256 637 061 435 917e +001
long g	最优化长格式小数 ,最大显十五位数	12. 566 370 614 359 2
rat	近似有理数 (分数)	1 420 /113
bank	(金融)两位小数	12. 57
hex	十六进制	402 921fb54 442d18
+	表示出大矩阵的正、负或零的符号	显示数据的 +、- 或零
缺省标识符	恢复默认的数字设置 :四位小数定点数	12. 5664

例 3-1 在 MATLAB 中对算式 $a=0.5+9/7$ 分别用数值算法和符号算法进行计算。

解 键入 :

```
format long, a=9/7+0.5
```

回车得出 :

```
a =
    1.78571428571429    (十五位数值量)
```

键入：

```
b1 =sym(0.5); b2 =sym(9/7); b=b1-b2
```

回车得出：

```
b =
    25/14    (符号量)
```

键入：

```
b =vpa(b,10)
```

回车得出同样的结果。

将符号变量 b 分别转换成 6 位和 20 位有效数字的近似值,则可键入：

```
b1=vpa(b,6),b2=vpa(b,20)
```

回车得出：

```
b1 =
    1.78571
b2 =
    1.7857142857142857143
```

也可以键入：

```
vpa(0.5-9/7)
```

回车得出：

```
ans =
    1.7857142857142857142857142857143
```

上述变量中除变量 a 外均为符号变量,包括最后一行,虽然它在形式上是 32 位有效数字。对此,可用 class 指令查验。

3.3 设计算法的若干原则

算法的不同,将直接影响计算结果的误差。为了提高计算精度,必须研究计算方法和运算顺序中的一些科学规律,从中总结出设计算法的原则,以便指导编程。

3.3.1 算法的数值稳定性

实际计算中,由于数据运算顺序及算法的不同,同一个数学模型所得结果的误差也会大相径庭,这主要是由于初始数据的误差及其在计算过程中的传播造成的。计算结果受到计算过程中舍入误差影响小的算法,称为具有较好的数值稳定性。相反的,如果计算结果很容易被计算过程中的舍入误差所左右,就称这种算法是数值不稳定的。下面先看个例题,从中体会数值稳定性概念。

例如,求一元二次方程 $x^2 + 62.1x + 1 = 0$ 的根。

据方程 $x^2 + 2px + q = 0$ 的求根公式 $x_{1,2} = -p \pm \sqrt{p^2 - q}$,可得出 7 位有效数字的近似解：

$$x_1 = -0.016\ 107\ 23 \quad x_2 = -62.083\ 90$$

若取 4 位有效数字近似解, 得出两根为:

$$x_1 = -0.020\ 00 \quad x_2 = -62.10$$

从计算结果看, 由于所取有效数字位数不同, 致使两次算得结果的相对误差大不相同, x_1 为 19.46%, 而 x_2 仅为 0.025 9%, 原因是该题中 $p^2 \gg |q|$ 。这表明都用求根公式计算两个根的数值稳定性不好。

如果改换算法, 取 4 位有效数字求出 $x_2 = -62.10$ 后, 利用根和系数的关系 $x_1 \cdot x_2 = q$, 由 $x_1 = q/x_2$ 算出 $x_1 = 1/(-62.10) = -0.016\ 10$, 则数值稳定性就比较好。可见, 在计算过程中应该选择数值稳定性好的计算方法, 才能使计算结果尽量少受数据误差的影响。

3.3.2 设计算法的若干原则

下面提出算法设计中的一些原则, 以供参考。

1. 避免两个相近数的相减

因为两个相近数之差很小, 运算后的有效数字将比原始数据减少很多, 从而会使运算结果的相对误差变得很大。

例如, 当 $x \gg 1$ 时计算 $\sqrt{x+1} - \sqrt{x}$, 因为两数相差很小则容易产生较大误差。若变换成 $\frac{1}{\sqrt{x+1} + \sqrt{x}}$ 计算, 就能使计算误差减小。

2. 避免两个数量级相差很大的数运算时小数被“吃掉”

两个相差很大的数进行加减运算时, 很小的数将会显得无足轻重, 从而被“吃掉”。这时应改变运算顺序, 避免它们直接发生运算关系。

例如, 在五位浮点计算机上计算 $x = 12\ 652 + 0.5 + 0.3 + 0.4$ 时, 按顺序算法, 结果得出 12 652, 后三个数被吃掉了。若变换计算顺序, 先将三个小数相加, 再与大数加, 得出的结果是 12 653, 计算误差就减小了。

3. 算法中尽量减少运算次数

每运算一步就会产生一次取舍误差, 计算步数越多, 积累误差就越大。因此, 计算中要首先进行公式化简, 运算次数越少越好, 不仅可以减少误差的积累, 而且还能提高运算速度。

例如, 计算多项式 $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ 时, 直接按顺序计算第 k 项需要做 k 次乘法, k 取 0 到 n 一共 $(n+1)$ 项, 共需做 $n(n+1)/2$ 次乘法, n 次加法。若变换成下式(秦九韶算法):

$$P_n(x) = \{ \dots [(a_n x + a_{n-1})x + a_{n-2}]x + \dots + a_1 \}x + a_0$$

则只需计算 n 次乘法, n 次加法。

可见, 化简公式能够减少运算次数, 从而可以减小误差积累, 这对于数值计算是很必要的。因此, 在实际代入数据计算之前, 首先要对公式进行化简。

4. 避免用绝对值过小的数作除数

除数过小会使商数的误差增大,所以遇到这种情况应该改变算法。如当 x 接近零时要计算 $\frac{1 - \cos x}{\sin x}$, 由于分母接近于零使商的误差会非常大。若把计算公式变换成 $\frac{\sin x}{1 + \cos x}$, 分母不会为零相差不多,避免过大的误差。

5. 递推运算中,防止误差的积累增大

递推关系会使运算过程规律化,计算便捷。但若多次递推,误差的积累将会不断增大,甚至导致错误的结果。如果使递推过程中误差的积累不断减小,就会提高计算精度。因此,应该选用误差积累不断减小的递推计算方法,使数值稳定性尽量变好。

例如,计算积分 $E_n = \int_0^1 x^n e^{x-1} dx$, $n = 1, 2, \dots, 9$ 时,利用分部积分法,可得递推公式:

$$E_n = x^n e^{x-1} \Big|_0^1 - n \int_0^1 x^{n-1} e^{x-1} dx = 1 - nE_{n-1} \quad E_1 = 1/e \quad n = 1, 2, \dots, 9$$
由此可以得出两个递推公式 $E_n = 1 - nE_{n-1}$ 和 $E_{n-1} = (1 - E_n)/n$ 。

容易验证,用前者计算时, n 由小向大递推,绝对误差将迅速扩大。如果设 $|ae(E_1)| = 10^{-5}$, 则 $|ae(E_9)| = 9! \times 10^{-5} > 3.6$ 。

用后一个公式计算时, n 由大向小递推,便可使误差逐渐变小,将得到比较精确的结果。因此,往往选用后一递推公式,先算出 E_n , 逐级往前推算。

练习题 3

1. 设 $x = 1.991 \pm 10$, $y = 1.991 \pm 0.0001$, $z = 0.0001991 \pm 0.0000001$, 哪一个的精度高? 为什么?

2. 在 MATLAB 中,以 10 种不同的格式显示 $\sqrt{2}$, 熟悉数显格式间的变换。

3. 已知矩阵 $A = \begin{bmatrix} \sqrt{3} & e^7 \\ \sin 5 & \ln 4 \end{bmatrix}$, 写出 πA 的 6 位有效数字表示式。

4. 如何计算 $\frac{1}{6.251} - \frac{1}{6.252}$, 可使误差减小?

5. 设法改变下列计算关系式,使其计算结果比较精确。

$$(1) \frac{1}{1+3x} - \frac{1-x}{1+x}, |x| \ll 1 \quad (2) \sqrt{x + \frac{1}{x}} - \sqrt{x - \frac{1}{x}}, x \gg 1$$

$$(3) \lg x_1 - \lg x_2, x_1 \approx x_2 \quad (4) \frac{1 - \cos 2x}{x}, x \ll 1, x \neq 0$$

第4章 求解非线性方程 $f(x)=0$

方程求根是初等数学的重要内容之一,也是科学和工程中经常碰到的数值计算问题。它的一般形式是求下列方程的根:

$$f(x)=0 \quad (4-1)$$

实际上,就是寻找使函数 $f(x)$ 等于零的变量 x ,所以求方程(4-1)的根,也叫求函数 $f(x)$ 的零点。如果变量 x 是列阵,则方程(4-1)就代表方程组。

当方程(4-1)中的函数 $f(x)$ 是有限个指数、对数、三角、反三角或幂函数的组合时,则方程(4-1)被称为超越方程,例如 $e^{-x} - \sin(\pi x/2) + \ln x = 0$ 就是超越方程。

当方程(4-1)中的函数 $f(x)$ 是多项式时,即 $f(x) = P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$,则方程(4-1)就成为下面的多项式方程,也称代数方程:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0 \quad (4-2)$$

$P_n(x)$ 的最高次数 n 等于 2、3 时,用代数方法可以求出方程(4-2)的解析解,但是,当 $n \geq 5$ 时,伽罗瓦(Galois)定理已经证明它是没有代数求根方法的。至于超越方程,通常很难求出其解析解。所以,方程(4-1)的求解经常使用作图法或数值法,而计算机的发展和普及又为这些方法提供了广阔的发展前景,使之成为科学和工程中最实用的方法之一。

本章首先介绍求解 $f(x)=0$ 的 MATLAB 符号法指令,然后介绍求方程数值解的基本原理,最后再介绍求解 $f(x)=0$ 的 MATLAB 数值法指令。

4.1 求解 $f(x)=0$ 的 MATLAB 符号法

MATLAB 中设有求出方程 $f(x)=0$ 解析解或精确解的符号法指令 `solve`,由它得出的符号量结果,可以转换成任意位有效数字的数值解。该指令的使用格式为:

```
solve(s1,s2,...,sn,'v1','v2',...,'vn')
```

```
solve(s1,s2,...,sn,'v1,v2,...,vn')
```

```
[z1,z2,...,zn]=solve(s1,s2,...,sn,'v1','v2',...,'vn')
```

① 输入参量 s_1, s_2, \dots, s_n 为待解方程组 $f(x)=0$ 或函数 $f(x)$ 的字符、符号表达式,或者是代表它们的变量名。待解方程可以是任意线性、非线性或超越方程;

② 输入参量 v_1, v_2, \dots, v_n 是与方程对应的未知量,它的数目必须与方程数目相等;若写有输出变量名 z_1, z_2, \dots ,且与方程数相等,则输入参量 v_1, v_2, \dots, v_n 可以缺省;

③ 输出参量 z_1, z_2, \dots, z_n 是指定的输出变量名,方程解的结果分别赋值给它们。但是赋值顺序并不是输入参量 v_1, v_2, \dots, v_n 的排序,而是按未知变量名在字母表中的排序输出。求解方程组时,这些输出参量不可省略,而且必须跟方程数相等,否则只输出解的结构(解的维数);

④ 当方程组不存在解析解或精确解时,该指令输出方程的数字形式符号量解;

⑤ 解析解表达式太冗长或含有不熟悉的特殊函数时,可用 `vpa` 指令转换成数值解。

例4-1 由方程 $ax^2 + bx + 5 = 0$ 求出 x 和 b 来。

解 键入：

```
s1 = 'a*x^2 + b*x + 5'; % 函数 f(x) 的字符串表达式
或 s1 = 'a*x^2 + b*x - 5'; % 方程 f(x) = 0 的字符串表达式
或 s1 = sym('a*x^2 + b*x + 5'); % 函数 f(x) 的符号表达式
或 s1 = sym('a*x^2 + b*x - 5'); % 方程 f(x) = 0 的符号表达式
```

再键入：

```
x = solve(s1)
```

回车得出：

```
x =
[1/2/a*( -b + (b^2 - 20*a)^(1/2))] 即  $x_{1,2} = \frac{1}{2a}(-b \pm \sqrt{b^2 - 20a})$ 
[1/2/a*( -b - (b^2 - 20*a)^(1/2))]
```

若键入：

```
b = solve('a*x^2 + b*x + 5', 'b') (或 b = solve(s1, 'b'))
```

回车得出：

```
b =
-(a*x^2 + 5)/x
```

例4-2 求解方程 $e^{-x} = \sin\left(\frac{\pi x}{2}\right)$ 。

解 键入：

```
x = solve('exp(-x) = sin(pi*x/2)') (或 solve('exp(-x) - sin(pi*x/2)'))
```

回车得出：

```
x =
.44357353410429277965457309417668 (32 位有效数字的符号量)
```

例4-3 求解方程组
$$\begin{cases} x^2 + x\sqrt{5} = -1 \\ x + 3z^2 = 4 \\ yz + 1 = 0 \end{cases}$$

解 键入：

```
s1 = 'x^2 + sqrt(5)*x + 1'; s2 = 'x + 3*z^2 - 4'; s3 = 'y*z + 1';
[u v w] = solve(s1, s2, s3)
```

回车得出：

```
u =
-1/2 - 1/2*5^(1/2)
-1/2 + 1/2*5^(1/2)
1/2 - 1/2*5^(1/2)
1/2 + 1/2*5^(1/2)

v =
```

```

1/76*(54-6*5^(1/2))^(1/2)*(9-5^(1/2))
-1/76*(54-6*5^(1/2))^(1/2)*(9-5^(1/2))
1/44*(42-6*5^(1/2))^(1/2)*(7-5^(1/2))
-1/44*(42-6*5^(1/2))^(1/2)*(7-5^(1/2))

w =
1/6*(54-6*5^(1/2))^(1/2)
-1/6*(54-6*5^(1/2))^(1/2)
1/6*(42-6*5^(1/2))^(1/2)
-1/6*(42-6*5^(1/2))^(1/2)

```

再键入：

```
u=vpas(u,5),v=vpas(v,5),w=vpas(w,5)
```

回车得出相应的小数值：

```

u =
-61800
-61800
4.6180
4.6180

v =
-80599
.80599
-73076
.73076

w =
1.2407
-1.2407
1.3685
-1.3685

```

4.2 求方程 $f(x)=0$ 数值解的基本方法

并非所有的方程 $f(x)=0$ 都能求出精确解或解析解,不存在这种解的方程就需要用数值解法求出近似解,下面介绍几种常见的数值解法基本原理。

4.2.1 求实根的二分法原理

设方程 $f(x)=0$ 中的函数 $f(x)$ 为实函数,且满足：

- ① 函数 $f(x)$ 在 $[a, b]$ 上单调、连续；
- ② 方程 $f(x)=0$ 在 (a, b) 内只有一个实根 x^* 。

则求方程 $f(x)=0$ 的根 就是在 (a, b) 内找出使 $f(x)$ 为零的点 x^* : $f(x^*)=0$,即求函数 $f(x)$

的零点。因为 $f(x)$ 单调连续, 由连续函数的性质可知, 若任意两点 $a_j, b_j \in [a, b]$, 而且满足条件 $f(a_j)f(b_j) < 0$, 则闭区间 $[a_j, b_j]$ 上必然存在方程的根 x^* , 即 $x^* \in [a_j, b_j]$ 。

据此原理提出求实根的二分法如图 4-1 所示,

先用中点 $b_1 = \frac{a+b}{2}$ 将区间 $[a, b]$ 平分为两个子区间

(a, b_1) 和 (b_1, b) , 方程的根必然在子区间两端点上函数值之积小于零的那一半中, 即不在 (a, b_1) 内, 就在 (b_1, b) 内, 除非 $f(b_1) = 0$, 于是寻根的范围缩小了一半。图 4-1 中的根 x^* 在区间中点左侧, 即 $x^* \in (a, b_1)$ 。再将新的含根区间 (a, b_1) 分成两半, 重复上述步骤确定出更新的含根子区间。如此重复 n 次, 设含根区间缩小为 (a_n, b_n) , 则方程的根 $x^* \in (a_n, b_n)$,

这一系列含根的子区间满足:

$$(a, b) \supset (a_1, b_1) \supset (a_2, b_2) \supset \dots \supset (a_n, b_n) \supset \dots$$

由于含根区间范围每次减半, 子区间的宽度为 $b_n - a_n = \frac{b-a}{2^n}$ ($n=1, 2, \dots$), 显然当 $n \rightarrow \infty$ 时, $(b_n - a_n) \rightarrow 0$, 即子区间收敛于一点 x^* , 这点就是方程的根。若 n 为有限整数, 取最后一个子

区间的中点 $x_n = \frac{a_n + b_n}{2}$ 作为方程根的近似值, 它满足 $f(x_n) \approx 0$, 于是有:

$$|x_n - x^*| \leq \frac{1}{2} \frac{b-a}{2^n} = \frac{b-a}{2^{n+1}}$$

这就是近似值 x_n 的绝对误差限。假定预先要求的误差限为 ε , 由 $\varepsilon < \frac{b-a}{2^{n+1}}$ 便可以求出满足误差要求的最小等分次数 n 。

4.2.2 迭代法

迭代法是计算数学中的一种重要方法, 用途很广, 求解线性方程组和矩阵特征值时也要用到它。这里结合非线性方程的迭代法求解, 介绍一下它的基本原理。

1. 迭代法基本原理

迭代法的基本原理就是构造一个迭代公式, 反复用它得出一个逐次逼近方程根的数列, 数列中每个元素都是方程根的近似值, 只是精度不同。

迭代法求解方程 $f(x)=0$ 时, 先把方程等价地变换成形式 $f(x)=x-g(x)=0$ 移项得出:

$$x = g(x) \quad (4-3)$$

若函数 $g(x)$ 连续, 则称(4-3)为迭代函数。用它构造出迭代公式:

$$x_{k+1} = g(x_k) \quad k=0, 1, 2, \dots \quad (4-4)$$

从初始值 x_0 出发, 便可得出迭代序列:

$$\{x_k\} = x_0, x_1, x_2, \dots, x_k, \dots \quad (4-5)$$

如果迭代序列(4-5)收敛, 且收敛于 x^* , 则由式(4-4)有:

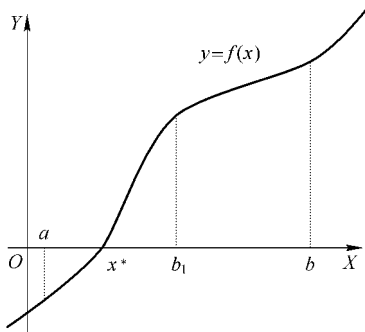


图 4-1 方程求根二分法原理示意图

$$\lim_{k \rightarrow \infty} (g(x_k) - x_{k+1}) = (g(x^*) - x^*) = f(x^*) = 0$$

可见 x^* 便是方程(4-1)的根。

2. 迭代法几何意义

解方程 $f(x) = 0$ 可以等价地变换成求解 $x = g(x)$ 如图 4-2 所示,就等于求曲线 $y = x$ 和 $y = g(x)$ 交点 P^* 的坐标 x^* 。求迭代序列(4-5)就等于从图中 x_0 点出发,由函数 $y = g(x_0)$ 得出 $y = P_0$,代入函数 $y = x$ 中得出 Q_1 ,再把 Q_1 的 x 坐标 x_1 代入方程 $y = g(x)$ 得出 P_1 ,如此继续下去,便可在曲线 $y = g(x)$ 上得到一系列的点 $P_0, P_1, \dots, P_k, \dots$ 这些点的 x 坐标便是迭代数列 $x_1, x_2, \dots, x_k, \dots$ 它趋向于方程(4-1)

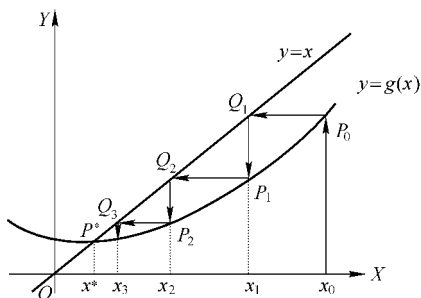


图 4-2 方程求根迭代法原理示意图

的根 x^* , 数列的元素就是方程根的近似值。数列的收敛就等价于曲线 $y = x$ 和 $y = g(x)$ 能够相交于一点。

3. 迭代公式收敛定理

要想用迭代法求出方程根的近似值,迭代序列(4-5)必须收敛。在什么条件下才能保证它收敛,下面的定理回答了这个问题,同时也给出了迭代公式的误差。

收敛定理 方程 $x = g(x)$ 在 (a, b) 内有根 x^* , 如果:

- ① 当 $x \in [a, b]$ 时 $g(x) \in [a, b]$;
- ② $g(x)$ 可导, 且存在正数 $q < 1$, 使得对于任意 $x \in [a, b]$ 都有 $|g'(x)| \leq q < 1$ 则有以下结论。
 - ① 方程 $x = g(x)$ 在 (a, b) 内有唯一的根 x^* 。
 - ② 迭代公式 $x_{k+1} = g(x_k)$ 对 (a, b) 内任意初始近似根 x_0 均收敛于 x^* 。
 - ③ 近似根 x_k 的误差估计公式为:

$$|x^* - x_k| \leq \frac{q^k}{1 - q} |x_1 - x_0| \quad (4-6)$$

证明 ① 设方程 $x = g(x)$ 的根为 x^* , 则 $x^* = g(x^*)$ 。如果 y^* 也是方程的根, 代入方程可得 $y^* = g(y^*)$ 。用中值定理和已知条件可推知:

$$|x^* - y^*| = |g(x^*) - g(y^*)| = |g'(\xi)(x^* - y^*)| \leq q |x^* - y^*| \quad \xi \in [x^*, y^*]$$

由于 q 是小于 1 的正数, 上式矛盾。所以必有 $x^* = y^*$, 即方程有唯一的根 x^* 。

② 任取一初始近似根 $x_0 \in [a, b]$ 若设 $x_k \in [a, b]$ $k = 0, 1, 2, \dots$ 用迭代公式及微分中值定理可得出 $|x^* - x_{k+1}| = |g(x^*) - g(x_k)| = |g'(\xi)(x^* - x_k)| \leq q |x^* - x_k| \leq q^2 |x^* - x_{k-1}| \leq \dots \leq q^{k+1} |x^* - x_0|$, 式中 $\xi \in (x_k, x^*)$ 。

因为 $q < 1$, 当 $k \rightarrow \infty$ 时 $|x^* - x_{k+1}| \rightarrow 0$, 所以 $\lim_{k \rightarrow \infty} x_k = x^*$ 。

③ 对于任意的正整数 p , 由绝对值不等式性质可知下式成立:

$$|x_{k+p} - x_k| \leq |x_{k+p} - x_{k+p-1}| + |x_{k+p-1} - x_{k+p-2}| + \dots + |x_{k+1} - x_k|$$

再由迭代公式及微分中值定理可得出:

$$|x_{k+1} - x_k| = |g(x_k) - g(x_{k-1})| \leq q |x_k - x_{k-1}| \leq \dots \leq q^k |x_1 - x_0|$$

$$\begin{aligned} \text{因为 } |x_{k+p} - x_k| &\leq q^{k+p-1} |x_1 - x_0| + q^{k+p-2} |x_1 - x_0| + q^{k+p-3} |x_1 - x_0| + \dots + q^k |x_1 - x_0| \\ &= q^k (q^{p-1} + q^{p-2} + q^{p-3} + \dots + 1) |x_1 - x_0| = \frac{1 - q^p}{1 - q} q^k |x_1 - x_0| \end{aligned}$$

当 $p \rightarrow \infty$ 时对上式两边取极限, 则得出 $|x^* - x_k| \leq \frac{q^k}{1 - q} |x_1 - x_0|$ 。

由这个定理可知, 将方程 $f(x)=0$ 转化成等价形式 $x=g(x)$ 时, 选择和构造什么样的迭代函数 $g(x)$ 非常重要, 只有当它满足一定的条件时, 迭代序列才收敛于方程的根 x^* 。

4.2.3 切线法

切线法就是从函数曲线上的一点出发, 不断用曲线的切线代替曲线, 求得收敛于根的数列的一种迭代方法。

1. 切线法原理

解非线性方程 $f(x)=0$ 的切线法也称牛顿法, 它是把方程线性化的一种近似方法, 用函数 $f(x)$ 的切线代替曲线产生一个收敛于方程根的迭代序列, 得到方程的近似根。

把函数 $f(x)$ 在某一初始值 x_0 点附近展开成泰勒级数:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + (x - x_0)^2 \frac{f''(x_0)}{2!} + \dots \quad (4-7)$$

取其线性部分, 近似地代替函数 $f(x)$ 可得方程的近似式:

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) = 0$$

设 $f'(x_0) \neq 0$, 解该近似方程可得:

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

把函数 $f(x)$ 在 x_1 点附近展开成泰勒级数, 取其线性部分替代函数 $f(x)$, 设 $f'(x_1) \neq 0$, 得:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

如此继续做下去, 就可以得到牛顿迭代公式:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad k=0, 1, 2, \dots \quad (4-8)$$

由式(4-8)得出的迭代序列 $x_1, x_2, \dots, x_k, \dots$ 在一定的条件下收敛于方程的根 x^* 。

2. 几何意义

切线法的几何意义非常明显。如图 4-3 所示, 选取初值 x_0 后, 过 $(x_0, f(x_0))$ 点作曲线 $y=f(x)$ 的切线, 其方程为 $y - f(x_0) = (x - x_0)f'(x_0)$ 。设切线与 X 轴的

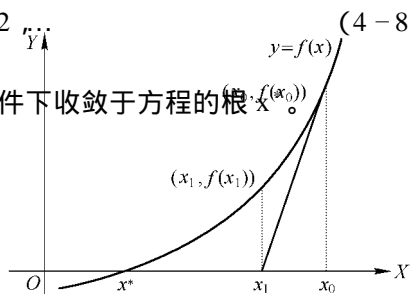


图 4-3 方程求根切线法原理示意图

交点为 x_1 则 $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ 再过 $(x_1, f(x_1))$ 作切线 与 x 轴交点 $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ 如此不断作切线 求与 x 轴的交点 便可得出的一系列的交点 $x_1, x_2, \dots, x_k, \dots$ 它们逐渐逼近方程的根 x^* 。

3. 切线法的收敛性

理论可以证明 在有根区间 $[a, b]$ 上 $f(x) \neq 0$ 、 $f''(x)$ 连续且不变号 则只要选取的初始近似根 x_0 满足 $f(x_0)f'(x_0) > 0$ 切线法必定收敛。它的收敛速度可如下推出。

方程 $f(x) = 0$ 可以等价地写成 $f(x) = (x - x^*)f'(x)$ 若 $f'(x) \neq 0$ 移项可得 $x = x^* - \frac{f(x)}{f'(x)}$ 。

设 $g(x) = x - \frac{f(x)}{f'(x)}$ 两边求导得 $g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$ 代入 $x = x^*$ $f'(x^*) \neq 0$ 则必得 $g'(x^*) = 0$ 。

另一方面 比较迭代公式 $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ 和 $g(x) = x - \frac{f(x)}{f'(x)}$ 可知 $x_{k+1} = g(x_k)$ 。把函数 $g(x)$ 在 x^* 点展成泰勒级数 只取到二阶导数则有：

$$x_{k+1} = g(x_k) = g(x^*) + g'(x^*)(x_k - x^*) + \frac{g''(x^*)}{2}(x_k - x^*)^2$$

由于 $g'(x^*) = 0$ 所以有 $x_{k+1} = g(x_k) = g(x^*) + \frac{g''(x^*)}{2}(x_k - x^*)^2$ 移项并注意 $x^* = g(x^*)$ 得出：

$$x_{k+1} - g(x^*) = x_{k+1} - x^* = \frac{g''(x^*)}{2}(x_k - x^*)^2$$

为了将式中的 $g''(x^*)$ 换成 $f(x)$ 对 $g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$ 两边求导 并代入 $f(x^*) = 0$ 则有：

$$g''(x^*) = \frac{f''(x^*)}{f'(x^*)}$$

将它代入前式得出：

$$x_{k+1} - x^* = \frac{f''(x^*)}{2f'(x^*)}(x_k - x^*)^2 \quad (4-9)$$

$\frac{f''(x^*)}{2f'(x^*)}$ 是个常数 式(4-9)表明用牛顿迭代公式在某次算得的误差 与上次误差的平方成正比 可见牛顿迭代公式的收敛速度很快。

4.2.4 割线法(弦截法)

应用切线法的牛顿迭代公式时 每次都需计算导数 $f'(x_k)$ 若将该导数用差商代替 就成为割线法(有时称快速弦截法)的迭代公式：

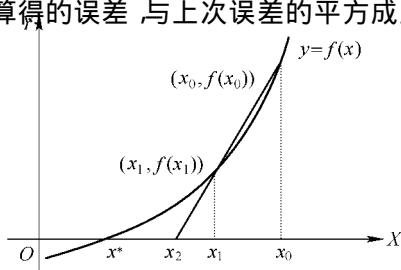


图 4-4 方程求根割线法原理示意图

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k), k=0, 1, 2, \dots \quad (4-10)$$

割线法的几何意义也很明显。如图 4-4 所示,过点 $(x_0, f(x_0))$ 和 $(x_1, f(x_1))$ 作函数 $y=f(x)$ 曲线的割线,交 X 轴于点 x_2 ,再过点 $(x_1, f(x_1))$ 和 $(x_2, f(x_2))$ 作曲线的割线,交 X 轴于点 x_3, \dots 一直做下去,则割线与 X 轴的交点序列将趋于方程的根 x^* 。

非线性方程的数值解法还有许多,这里仅介绍了几种基本方法的原理。二分法简单方便,但收敛速度慢;迭代法虽然收敛速度稍微快点,但需要判断能否收敛;只要初值选取得当,切线法具有恒收敛且收敛速度快的优点,但要求出函数的导数;弦截法不要求导数,特别是前面介绍的快速弦截法,收敛速度很快,但是需要知道两个近似的初始根值才能作出弦,要求的初始条件较多。这些方法各有千秋,需根据具体情况选用。

4.3 方程 $f(x)=0$ 数值解的 MATLAB 实现

MATLAB 中求方程数值解的办法很多,有的是专用指令,有的是根据方程性质而借用其他专用指令求得的,使用中应特别注意应用条件及它们间的差异。

4.3.1 代数方程的求根指令 roots

对于多项式方程(4-2),可用多项式求根指令 `roots` 求解,使用格式为:

`roots(p)`

① 每次只能求一个一元多项式的根,该指令不能用于求方程组的解,必须把多项式方程变成 $P_n(x)=0$ 的形式;

② 参数 p 是多项式 $P_n(x)=a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ 的系数向量 $p=[a_n, a_{n-1}, \dots, a_1, a_0]$,该向量的分量由多项式系数构成,排序是从高次幂系数到低次幂系数,缺少的幂次系数用零填补;

③ 输出多项式方程的所有实数和复数根。

例 4-4 求方程 $x^3 = x^2 + 1$ 的解。

解 键入:

```
roots([1 -1 0 -1])
```

回车得出:

```
ans =
    1.4656
    0.2328 + 0.7926i
    0.2328 - 0.7926i
```

4.3.2 求函数零点指令 fzero

求解方程 $f(x)=0$ 的实数根也就是求函数 $f(x)$ 的零点。MATLAB 中设有求函数 $f(x)$ 零点的指令 `fzero`,可用它来求方程的实数根。该指令的使用格式为:

`fzero(fun,x0,options)`

- ① 输入参数 `fun` 为函数 $f(x)$ 的字符表达式、内联函数名或 M-函数文件名。
- ② 输入参数 x_0 为函数某个零点的大概位置(不要取零)或存在的区间 $[x_i, x_j]$,要求函数 $f(x)$ 在 x_0 点左右变号,即 $f(x_i)f(x_j) < 0$ 。
- ③ 输入参数 `options` 可有多种选择,若用 `optimset('disp','iter')` 代替 `options` 时,将输出寻找零点的中间数据。
- ④ 该指令无论对多项式函数还是超越函数都可以使用,但是每次只能求出函数的一个零点,因此在使用前需摸清函数零点数目和存在的大体范围。为此,常用绘图指令 `plot`, `fplot` 或 `ezplot` 画出函数 $f(x)$ 的曲线,从图上估计出函数零点的位置。

例 4-5 求方程 $x^2 + 4\sin(x) = 25$ 的实数根($-2\pi < x < 2\pi$)。

解 (1) 首先要确定方程实数根存在的大致范围。为此,先将方程变成标准形式 $f(x) = x^2 + 4\sin(x) - 25 = 0$ 。在指令窗中键入:

`clf,ezplot x^2+4*sin(x)-25'`
回车,得出图 4-5。

Current plot held

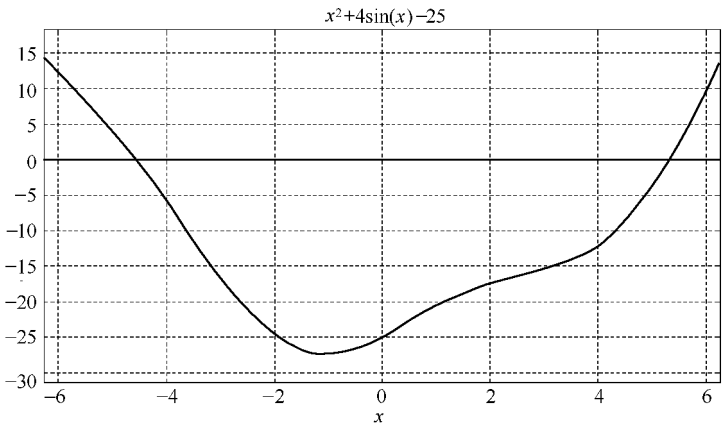


图 4-5 确定方程实数根存在位置的曲线

从曲线上可以看出,函数的零点大约在 $x_1 \approx -4$ 和 $x_2 \approx 5$ 附近。
(2) 直接使用指令 `fzero` 求出方程在 $x_1 \approx -4$ 时的根。在指令窗中键入:

`x1=fzero('x^2+4*sin(x)-25',4)`
回车得出:

`x1 =`
`4.5861`

若键入:

`fzero('x^2+4*sin(x)-25',4,optimset('disp','iter'))`
回车得出:

Search for an interval around 4 containing a sign change:
Func-count a f(a) b f(b) Procedure

1	4	5.97279	4	5.97279	initial	inter-
3	3.88686	7.17961	4.11314	4.77907	val	
5	3.84	7.68241	4.16	4.28931	search	
7	3.77373	8.39553	4.22627	3.60199	search	
9	3.68	9.40652	4.32	2.64161	search	
11	3.54745	10.8364	4.45255	1.30909	search	
13	3.36	12.8437	4.64	0.519124	search	

Search for a zero in the interval [3.36 , 4.64]:

Func-count	x	f(x)	Procedure
13	4.64	0.519124	initial
14	4.59027	0.0408288	interpolation
15	4.58604	9.62876e-005	interpolation
16	4.58605	4.13699e-008	interpolation
17	4.58605	4.61853e-014	interpolation
18	4.58605	0	interpolation

Zero found in the interval [3.36 , 4.64]

ans =

4.5861

中间数据表明,求根过程中不断缩小探测范围,最后得出 - 4 附近满足精度的近似根。

(3) 为了求出 $x_2 \approx 5$ 的根,在指令窗中键入:

```
x2=fzero('x^2-4*sin(x)-25',5)
```

回车得出:

x2 =

5.3186

(4) 也可以用内联函数作为输入参数,如键入:

```
f=inline('x^2-4*sin(x)-25'); x1=fzero(f,4),x2=fzero(f,5)
```

回车得出:

x1=4.5861 x2=5.3186

例4-6 将方程 $x^2 + 4\sin(x) = 25$ 编成 M-函数文件(实用中在函数较为复杂、而又多次重复调用时,才这样做),用 fzero 求解。

解 (1) 在编辑调试窗中键入:

```
function yy=li4_6(x)
yy=x^2-4*sin(x)-25;
```

以 li4_6 为名存盘,退出编辑调试窗,回到指令窗。

(2) 在指令窗中键入作图指令:

```
fplot('li4_6',[ -6 6]),hold,ezplot x x,grid
```

回车。

在图形窗中得出与前面类似的函数曲线,从中可以确定根的大体位置。

(3) 在指令窗中键入下述指令可求出 -4 附近的根：

```
x1 = fzero('li4_6', 4)
```

回车得出：

```
x1 =  
4.5861
```

若键入 `x2 = fzero('li4_6', 5)` , 可以求出 5 附近的另一个根。

4.4 求解非线性方程组数值解的迭代法

对于非线性方程组(以二元方程组为例,其他可以类推)

$$\begin{cases} f_1(x, y) = 0 \\ f_2(x, y) = 0 \end{cases} \quad (4-11)$$

的数值解法跟一元非线性方程的切线法(牛顿法)雷同,也是把非线性函数线性化,近似替代原方程得出数值解,所以也叫作牛顿迭代法。

假设方程组(4-11)的初始估计值为 (x_0, y_0) , 可以把方程组(4-11)中的两个函数 $f_1(x, y)$ 和 $f_2(x, y)$ 在 (x_0, y_0) 处用二元泰勒级数展开,只取线性部分,移项得出：

$$\left. \begin{aligned} \frac{\partial f_1(x_0, y_0)}{\partial x}(x - x_0) + \frac{\partial f_1(x_0, y_0)}{\partial y}(y - y_0) &= -f_1(x_0, y_0) \\ \frac{\partial f_2(x_0, y_0)}{\partial x}(x - x_0) + \frac{\partial f_2(x_0, y_0)}{\partial y}(y - y_0) &= -f_2(x_0, y_0) \end{aligned} \right\} \quad (4-12)$$

若系数矩阵行列式 $J_0 = \begin{vmatrix} \frac{\partial f_1(x_0, y_0)}{\partial x} & \frac{\partial f_1(x_0, y_0)}{\partial y} \\ \frac{\partial f_2(x_0, y_0)}{\partial x} & \frac{\partial f_2(x_0, y_0)}{\partial y} \end{vmatrix} \neq 0$ 则方程组(4-12)的解为：

$$\begin{aligned} x_1 &= x_0 + \frac{1}{J_0} \begin{vmatrix} \frac{\partial f_1(x_0, y_0)}{\partial y} & f_1(x_0, y_0) \\ \frac{\partial f_2(x_0, y_0)}{\partial y} & f_2(x_0, y_0) \end{vmatrix} \\ y_1 &= y_0 + \frac{1}{J_0} \begin{vmatrix} f_1(x_0, y_0) & \frac{\partial f_1(x_0, y_0)}{\partial x} \\ f_2(x_0, y_0) & \frac{\partial f_2(x_0, y_0)}{\partial x} \end{vmatrix} \end{aligned}$$

方程组(4-11)中的两个函数 $f_1(x, y)$ 和 $f_2(x, y)$ 在 (x_1, y_1) 处,再用二元泰勒级数展开,只取线性部分,……如此继续替代下去,直到方程组的根达到所要求的精度,就完成了方程组的求解。

求解方程组(4-11)还有许多其他办法,如“最速下降法”,它是利用方程组(4-11)构成所谓模函数 $\Phi(x) = [f_1(x, y)]^2 + [f_2(x, y)]^2$, 通过求模函数极小值的方法得到方程组的数值解,诸如此类在此不再一一列举。

4.5 求方程组数值解的指令

`fsolve` 是用最小二乘法求解非线性方程组 $F(X)=0$ 的指令,变量 X 可以是向量或矩阵,方程组可以由代数方程或者超越方程构成。它的使用格式为:

`fsolve('fun',X0,OPTIONS)`

① 参数 `fun` 是编辑并存盘的 M-函数文件的名称,可以用@ 代替单引号对它进行标识。M-函数文件主要内容是方程 $F(X)=0$ 中的函数 $F(X)$,即方程左边的函数。

② 参数 `X0` 是向量或矩阵,为探索方程组解的起始点。求解将从 `X0` 出发,逐渐趋向,最终得到满足精度要求、最接近 `X0` 的近似根 X^* : $F(X^*) \approx 0$ 。由于 `X0` 是向量或矩阵,无法用画图方法进行估计,实际问题中常常是根据专业知识、物理意义等进行估计。

③ 该指令输出一个与 `X0` 同维的向量或矩阵,为方程组的近似数值解。

④ 参数 `OPTIONS` 为设置选项,用它可以设置过程显示与否、误差、算法……,具体内容可用 `help` 查阅。通常可以省略该项内容。

此外,还有一些其他调用形式,可用 `help` 查阅,如:

`[X,FVAL,EXITFLAG]=FSOLVE(FUN,X0,...);`

`[X,FVAL,EXITFLAG,OUTPUT]=FSOLVE(FUN,X0,...);`

`[X,FVAL,EXITFLAG,OUTPUT,JACOB]=FSOLVE(FUN,X0,...)。`

例 4-7 求方程组
$$\begin{cases} x^2 + y^2 + z + 7 = 10x \\ xy^2 = 2z \\ x^2 + y^2 + z^2 = 3y \end{cases}$$
 在 $x_0=1$ $y_0=1$ $z_0=1$ 附近的数值解。

解 (1) 在编辑调试窗中编辑 M-函数文件。首先将方程组变换成 $F(X)=0$ 的形式, x , y , z 看成向量 X 的三个分量。打开编辑调试窗,键入:

```
function ms=li4_7(X)
```

```
ms(1)=X(1)^2-10*X(1)+X(2)^2-X(3)-7;
```

```
ms(2)=X(1)*X(2)^2-2*X(3);
```

```
ms(3)=X(1)^2-X(2)^2-3*X(2)-X(3)^2;% 输出参量 ms 也有三个分量
```

用“`li4_7`”为 M-函数文件名存盘,退出编辑调试窗,回到指令窗。

(2) 在指令窗中键入:

```
fsolve('li4_7',[1 1 1])
```

回车得出:

Optimization terminated: first-order optimality is less than options.TolFun.

```
ans =
```

```
1.1042    1.3485    1.0039
```

若键入:

```
x=fsolve(@li4_7,[1 1 1],optimset('Display','iter'))
```

回车得出求解过程:

Norm of First-order Trust-region

Iteration	Func-count	f(x)	step	optimality	radius
0	4	516		204	1
1	8	176.027	1	91.5	1
2	12	9.8406	1.74719	10.2	2.5
3	16	0.376941	0.730715	1.34	4.37
4	20	0.00485028	0.221391	0.183	4.37
5	242	1.0753e-006	0.029966	0.00419	4.37
6	284	2.8134e-013	0.000625014	1.94e-006	4.37
7	32	1.8541e-026	2.78852e-007	4.09e-013	4.37

Optimization terminated: first-order optimality is less than options.TolFun.

x =

1.1042 1.3485 1.0039

该方程也可以用 MATLAB 的符号指令 solve 求解,但结果非常冗长。

例 4-8 求解方程组
$$\begin{cases} 3x = \cos(yz) + 0.5 \\ 2x^2 - 81(y + 0.1)^2 + \sin z + 1.06 = 0, \text{ 在 } x_0 = 0.1, y_0 = 0.1 \text{ 和} \\ e^{-xy} + 20z + \frac{10}{3}\pi = 1 \end{cases}$$

$z_0 = -0.1$ 附近的数值解。

解 首先将方程组变换成 $f_j(x, y, z) = f(X) = 0$ ($j = 1, 2, 3$) 的形式, 设 X 为一个三维向量, 令 $X(1) = x$, $X(2) = y$, $X(3) = z$, 则三维向量 $yy3 = f(X) = f_j(x, y, z)$, 然后编程计算。

(1) 在编辑调试窗中键入: function yy3 = li4_8(X)

yy3(1) = 3*X(1) - cos(X(2)*X(3)) - 0.5;

yy3(2) = 2*X(1)^2 - 81*(X(2) + 0.1)^2 - sin(X(3)) - 1.06;

yy3(3) = exp(-X(1)*X(2)) - 20*X(3) - 40*pi/3 - 1;

以“li4_8”为名存盘, 退出编辑调试窗, 回到指令窗。

(2) 在指令窗中键入:

fsolve('li4_8', [0.1 0.1 -0.1])

回车得到:

ans =

0.5000 0.0144 0.5232

这是方程组的最小二乘解, 用符号指令 solve 无法得出最终结果。

练习题 4

1. 方程 $x^3 - x^2 - 1 = 0$ 在 $x_0 = 1.5$ 附近有根, 把方程写成下列不同的等价形式, 并导出相应的迭代公式, 判断下列每个迭代公式在 $x_0 = 1.5$ 附近的敛散性。

(1) $x = 1 + \frac{1}{x^2}$ 对应的迭代公式 $x_{k+1} = 1 + \frac{1}{x_k^2}$;

(2) $x^3 = x^2 + 1$ 对应的迭代公式 $x_{k+1} = \sqrt[3]{1 + x_k^2}$;

(3) $x^2 = \frac{1}{x-1}$ 对应的迭代公式 $x_{k+1} = \sqrt{\frac{1}{x_k - 1}}$ 。

2. 用“solve”、“roots”和“fzero”求方程 $x^3 - 2x^2 - 4x = 7$ 的根,比较它们的差异。

3. 求方程 $x^3 + 2x^2 + 10x = 20$ 的根。

4. 求方程 $1 - x = \sin x$ 的根。

5. 求方程 $x = \frac{\sin x}{x}$ 的根。

6. 求方程 $\sin x + 1 = \frac{x^2}{2}$ 在区间 $[-4, 2]$ 的根。

7. 求方程 $\cos x = \frac{1}{2} + \sin x$ 的根。

8. 求方程 $4x + e^x = \sin x$ 的根。

9. 已知方程组
$$\begin{cases} x^2 + y\sqrt{2} + 1 = 0 \\ x + 3z = 4 \\ yz + 1 = 0 \end{cases}$$
 求它在 $x=1$ $y=-1$ 和 $z=2$ 附近的数值解。

10. 已知方程组
$$\begin{cases} \sin x + y^2 + \ln z = 7 \\ 3x + 2y - z^3 + 1 = 0 \\ x + y + z = 5 \end{cases}$$
 求它在 $x=0$ $y=2$ 和 $z=5$ 附近的数值解。

第 5 章 求解线性代数方程组的直接法

工程技术和科学研究中的许多科学计算问题,往往归结为求解线性代数方程组,它们的求解方法可以分为直接法和迭代法两类。在没有舍入误差的条件下,直接法可以求得方程组的精确解(如 Cramer 法则),但该法计算烦琐,又受到计算机存储量等因素的限制,只能用于阶数不太高的方程组,实用性不强。但是,鉴于直接法是求解线性代数方程组的重要基础,又和有着重要意义的矩阵分解密不可分,本章专门把直接法、矩阵分解和 MATLAB 联系在一起加以介绍。

迭代法和求矩阵特征值密切相关,又是计算数学中的一种重要方法,所以把它们一并放在第 6 章另做介绍。

5.1 线性代数方程组求解概论

5.1.1 线性代数方程组的矩阵表示

含有 n 个未知量($x_1, x_2, x_3, \dots, x_n$)、 m 个保留方程的线性代数方程组:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{cases}$$

可以表示成矩阵形式:

$$Ax = b \quad (5-1)$$

其中, $A = \{a_{ij}\}_{m \times n}$ 为方程组的系数矩阵,列阵 $b = \{b_k\}_{m \times 1}$ 为方程组的自由项,列向量 $x = \{x_n\}_{n \times 1}$ 为未知量。把 A 和 b 组成的矩阵 $B = (A, b)_{m \times (n+1)}$ 叫作方程组的增广矩阵。

称自由项 $b \neq 0$ 的方程组(5-1)为非齐次线性代数方程组。

若自由项 $b = 0$ 则方程组(5-1)变成:

$$Ax = 0 \quad (5-2)$$

称为齐次线性代数方程组。

5.1.2 线性代数方程组解的性质

如果向量 $S = \{s_n\}_{n \times 1} = [s_1 \ s_2 \ \dots \ s_n]^T$ 代替 x 使方程 $Ax = b$ 成为恒等式,即 $AS = b$,则称 S 为方程(5-1)的一个解向量。

1. 解的判别

① 对于齐次线性代数方程组 $Ax=0$, 根据系数矩阵 A 的秩 $R(A)$ 和待求变量的个数 n (x 的维数) 间的关系, 可以对其解进行判断: 有非零解的充分必要条件是系数矩阵的秩 $R(A) < n$ 。若 $R(A) = n$, 则方程组只有零解。

② 对于非齐次线性代数方程组 $Ax=b$ 根据其系数矩阵 A 的秩 $R(A)$ 、增广矩阵 $B=[A, b]$ 的秩 $R(B)$ 和待求变量个数 n 间的关系, 可以分成三种类型。

- 恰定方程组 当 $R(A) = R(B) = n$ 时, 称方程组为恰定方程组, 这时它有唯一解向量。

- 欠定方程组 当 $R(A) = R(B) < n$ 时, 称方程组为欠定方程组, 这时它有无穷多解向量。

- 超定方程组 当 $R(A) < R(B)$ 时, 称方程组为超定方程组或矛盾方程组, 即保留方程个数大于未知量个数, 一般意义下无解。

2. 解的结构

不同类型的方程组, 其解的情况不同因而解的结构就有很大差异, 可概括如下。

① 齐次线性代数方程组

$$Ax=0$$

若系数矩阵的秩 $r = R(A) < n$, 则全体解向量构成 n 维空间 V_n 的一个子空间 S 。 S 由 $(n-r)$ 个线性无关的解向量 (称为基) 构成, 叫作方程组的解空间。

若 $R(A) = n$, 则解空间为零空间, 方程组没有非零解。

② 非齐次线性代数方程组

$$Ax=b$$

当 $R(A) = R(B) = r \leq n$ (属于恰定或欠定方程组) 时, 若非齐次线性方程组 $Ax=b$ 有一个特解是 β , 而 $\alpha_1, \alpha_2, \dots, \alpha_{n-r}$ 是对应的齐次线性方程组 $Ax=0$ 的 $(n-r)$ 个线性无关解向量, 则 $\beta + k_1\alpha_1 + k_2\alpha_2 + \dots + k_{n-r}\alpha_{n-r}$ (k_1, k_2, \dots, k_{n-r} 为任意常数) 就是非齐次线性方程组 $Ax=b$ 的全部解向量。

当 $R(A) = R(B) = r = n$ 时, 方程组只有唯一解 β 。

当 $R(A) < R(B)$ 时, 属于超定方程组, 这种矛盾方程组的保留方程数多于未知量数, 没有一般意义下的解, 但可以求出其最小二乘解 (在数据拟合部分介绍)。

5.2 恰定线性代数方程组求解

求解线性代数方程组的直接法, 就是经过有限次运算代换, 不断减少方程组中待求变量的个数, 最终得出方程组的解。如果不考虑计算过程中的舍入误差, 这种方法可以得出方程组的精确解, 克莱姆法则和高斯消去法就属此类, 它们是求解线性代数方程组的基础, 下面先作介绍。

5.2.1 克莱姆法则

对于恰定方程组 $AX=b$, 即满足 $R(A)=R(B)=n$ 的方程组求解, 可用克莱姆(Cramer)法则得出唯一解。这时, 系数矩阵 $A=\{a_{ij}\}_{n \times n}$ 是方阵, 如果 A 的 n 阶行列式 $|A|=\det A \neq 0$, 则线性方程组的唯一解是:

$$x_h = \frac{\Delta_h}{\det A}, \quad (h=1, 2, \dots, n) \quad (5-3)$$

式中 Δ_h 也是一个 n 阶行列式, 它是用自由项 b 代换 $\det A$ 中第 h 列元素得到的行列式。式(5-3)就是克莱姆法则公式。

虽然克莱姆法则给出了求解非齐次线性代数方程组的方法, 但是, 用它求解的运算次数太多。由式(5-3)可知, 求出解向量 x 需要计算 $(n+1)$ 个 n 阶行列式的值, 每个 n 阶行列式为 n 项之和, 每项是 n 个数的乘积, 需要进行的乘法运算就有 $N=(n+1)!(n-1)$ 次, 可见 n 较大时用克莱姆法求解是非常烦琐的。因此, 探求运算次数少, 计算过程规律, 需要存储的中间数据不多, 还能满足一定精度要求的其他求解方法是非常必要的。

5.2.2 高斯消去法

高斯消去法是求解线性代数方程组的基本方法之一, 虽然古老, 但是经过改进变形后得出了许多适用于计算机的高效算法, 仍为目前常用方法之一。

高斯消去法就是初等数学中的消元法, 演算过程分两步: 第一步是消元过程, 通过不断代换, 逐步消去待求变量的系数, 最终把原来方程组变换成等价而系数矩阵为三角阵的方程组; 第二步是回代过程, 从三角方程组的尖部开始, 逐个回代来求出各个未知量, 得出方程组的全部解。消元法中最简单、最基本的就是顺序高斯消去法, 以它为基础通过不断改进, 优化算法, 又得出了一些其他形式的消元法。

1. 顺序高斯消去法

1) 运算过程

设线性代数方程组(5-1)为恰定方程组, 系数矩阵是非奇异矩阵, 即 $\det A \neq 0$ 。由克莱姆法则知道, 它有唯一解向量。下边以线性代数方程组的一般形式为例, 介绍顺序消去法的运算过程。设方程组的增广矩阵为:

$$B=[A \ b]=\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right]$$

(1) 消元过程

第一次消元: 设 $a_{11} \neq 0$ (否则进行换行, 把第一列中元素不为零的行调到首行), 为使第一列 a_{11} 以下各元素变成零, 用第 i 行 ($i > 1$) 的各元素减去第一行对应元素的 (a_{i1}/a_{11}) 倍 ($i=2, 3, \dots, n$), 于是可以使原来的增广矩阵变成下述形式:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array} \right]$$

第二次消元: 设 $a_{22}^{(1)} \neq 0$ (否则进行换行), 为使第二列 $a_{22}^{(1)}$ 以下各元素变成零, 用第 i ($i > 2$) 行的各元素减去第二行对应元素的 $(a_{i2}^{(1)}/a_{22}^{(1)})$ 倍 ($i = 3, 4, \dots, n$)。

再进行第三次消元, ……依此方法不断进行消元, 只要消元过程中用作除数的元素不为零, 即 $a_{11}, a_{22}^{(1)}, a_{33}^{(2)}, \dots, a_{(n-1)(n-1)}^{(n-2)} \neq 0$, 经过 $(n-1)$ 次消元, 最终增广矩阵就会变成下述形式:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array} \right]$$

这样就完成了消元过程。

(2) 回代过程

从上述三角方程组最下一个方程逐个向上进行回代, 最终就可求出方程组的解向量。

2) 运算次数

一个 n 阶方程组经过 $(n-1)$ 次消元就可变成 n 阶三角方程组。第 k 次消元, 使 A 中第 k 列 a_{kk} 以下各元素为零时, 需做 $(n-k)$ 次除法、 $(n-k+1)(n-k)$ 次乘法和 $(n-k+1)(n-k)$ 次加减法。这个消元法过程中共需做乘除法次数为:

$$N_1 = \sum_{k=1}^{n-1} (n-k+1)(n-k) + \sum_{k=1}^{n-1} (n-k)$$

共需做加减法的总次数为:

$$N_2 = \sum_{k=1}^{n-1} (n-k+1)(n-k)$$

若令 $h = n - k$, 对以上两式作变量代换, 则:

$$N_1 = \sum_{h=1}^{n-1} h^2 + 2 \sum_{h=1}^{n-1} h, \quad N_2 = \sum_{h=1}^{n-1} h^2 + \sum_{h=1}^{n-1} h$$

把 $\sum_{h=1}^{n-1} h = \frac{1}{2}n(n-1)$ 和 $\sum_{h=1}^{n-1} h^2 = \frac{1}{6}(n-1)n(2n-1)$ 代入上面的 N_1 和 N_2 中, 稍加整理就可算出顺序消元法的总共运算次数为:

$$N = N_1 + N_2 = 2 \sum_{h=1}^{n-1} h^2 + 3 \sum_{h=1}^{n-1} h = \frac{1}{3}(n-1)n(2n-1) + \frac{3}{2}n(n-1) \approx \frac{2}{3}n^3 \quad (5-4)$$

显然, 比克莱姆法则乘除法运算次数 $(n+1)!(n-1)!$ 还要少很多。

2. 选主元高斯消去法

在顺序高斯消元过程中, 第 k 次消元时把第 k 行的第一个不为零元素 $a_{kk}^{(k-1)}$ (k 为正整数, 且 $1 < k < n$) 称为第 k 次消元的主元素。消元过程中用主元素做除数, 因此它不应该为零, 而且它的绝对值不能太小, 否则将会给计算结果带来较大的舍入误差。为此, 对顺序高斯消元

法进行改进,消元中每次都选取绝对值最大的元素做主元素,这可以通过对矩阵行列的调换加以实现,于是产生了选主元高斯消去法。根据选取主元素范围的不同,把选主元高斯消去法分为列主元和全主元消去法两种。

1) 列主元高斯消去法

在第 k 次($k=1, 2, \dots, n-1$)消元时,先从第 k 行第一个不为零元素 $a_{kk}^{(k-1)}$ 所在列中大于 k 行的各元素里选得绝对值最大的元素,通过换行将它调到主元素位置上,然后进行消元。

2) 全主元高斯消去法

在第 k 次($k=1, 2, \dots, n-1$)消元时,从第 k 行第一个不为零元素 $a_{kk}^{(k-1)}$ 右下方的

($n-k+1$)阶子矩阵 $\begin{bmatrix} a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ \vdots & & \vdots \\ a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{bmatrix}$ 里,选出绝对值最大的元素,通过行、列的调换将它

调到主元素的位置上,再进行消元。

5.3 矩阵的三角分解

5.3.1 高斯消去法和三角矩阵

从矩阵变换的角度看,高斯消去法第一步的消元过程,实质上就是用一系列初等方阵 P_1, P_2, \dots, P_{n-1} ,依次乘以方程 $Ax=b$ 两边,对其实行初等变换,即:

$$P_{n-1} P_{n-2} \dots P_1 Ax = P_{n-1} P_{n-2} \dots P_1 b$$

最终使系数矩阵 A 变成一个三角阵 $P_{n-1} P_{n-2} \dots P_1 A$,使方程组(5-1)等价地变换成一个三角形方程组:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{22}^{(1)}x_2 + \dots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \dots \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}$$

第二步的回代过程,就是先求出 $x_n = b_n^{(n-1)} / a_{nn}^{(n-1)}$,然后逐个由下往上进行回代,求得方程组的解。如果令 $P = P_{n-1} P_{n-2} \dots P_1$,这个三角方程组的矩阵形式为 $PAx = Pb$,它的系数矩阵 PA 显然是一个上三角矩阵。可见,高斯消去法的实质就是通过初等变换把待求方程组的系数矩阵 A 变换成三角矩阵,也叫作使矩阵 A 三角化。因此,研究如何使矩阵三角化对于求出线性方程组的解是很有帮助的。根据矩阵性质和需求的不同,产生出多种矩阵三角化的方法,下面仅介绍两种矩阵的三角分解方法。

5.3.2 矩阵的三角分解

1. LU 分解

由矩阵的初等变换理论可以知道,当方阵 A 的所有顺序主子式都不为零时,可以唯一地

分解成两个三角矩阵 L, U 的乘积, 即:

$$A = LU \quad (5-5)$$

其中 $L = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \cdots & \cdots & \cdots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}$ 为单位下三角阵(主对角线上元素均为1的下三角阵);

$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \cdots & \cdots \\ & & & u_{nn} \end{bmatrix}$ 为上三角阵。

通常把一个非奇异方阵 A 分解成两个三角阵的乘积, 叫作方阵 A 的三角分解(LU 分解)。特别地, 像刚才那样 L 为单位下三角阵, U 为上三角阵的分解称为杜利特尔(Doolittle)分解, 它具有唯一性。若 U 为单位上三角阵, L 为下三角阵, 则称为克劳特(Crout)分解。

对一个线性方程组的系数矩阵 A 进行 LU 分解后, 方程(5-1)就可以写成:

$$Ax = LUx = b \quad (5-6)$$

变换可得 $Ux = L^{-1}b$ 若令 $L^{-1}b = y$ 则有:

$$\begin{cases} Ly = b \\ Ux = y \end{cases} \quad (5-7)$$

L 是单位下三角阵时, 从 $Ly = b$ 中很容易求出向量 y , 把它代入 $Ux = y$ 时, 由于 U 是上三角阵便可以容易地得出 x 。可见, 对式(5-1)的系数矩阵 A 进行杜利特尔分解会给求解方程组带来很大方便, 特别是当几个方程组的系数矩阵 A 相同, 只是它们的自由项 b 不同时, 可以共用 L, U , 从而减少了重复计算的工作量。

2. LL^T 分解

如果 A 是 n 阶对称正定方阵, 即对任何 $x \neq 0$ 都有实二次型 $f(x) = xAx > 0$ 的对称矩阵, 则可以唯一地分解为下三角阵 L 和它的转置阵之积:

$$A = LL^T = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \cdots & \cdots & \cdots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & \cdots & l_{n2} \\ & & \cdots & \cdots \\ & & & l_{nn} \end{bmatrix}$$

于是, 方程组(5-1)就变成 $LL^T x = b$, 依次求解下面两个三角方程:

$$\begin{cases} L^T x = y \\ Ly = b \end{cases} \quad (5-8)$$

便可得出方程组的解向量 x 。

将对称正定矩阵 A 分解成 $A = LL^T$, 叫作乔累斯基(Cholesky)分解。

把 A 分解成 LL^T 时, 三角阵 L 的对角元素应该等于 A 的对角元素平方根, 这种分解正好与解方程组的平方根法相对应, 所以它可用于求解 A 为正定对称方阵的线性方程组。

5.4 线性代数方程组数值解和矩阵三角分解的 MATLAB 实现

5.4.1 齐次线性代数方程组求解指令

由线性代数理论可知,为了判断线性代数方程组解的性质,经常要求出矩阵的秩。

1. 求矩阵秩的指令 rank

它的使用格式为:

$$c = \text{rank}(A)$$

输出量 c 是矩阵 A 的秩 $\text{rank}(A)$,即 A 的线性无关最大行数或列数。MATLAB 中该指令是用奇异值分解法算出的。

例 5-1 求矩阵 $a = \begin{bmatrix} 2 & 2 & 1 \\ -3 & 12 & 3 \\ 8 & -2 & 1 \\ 2 & 12 & 4 \end{bmatrix}$ 的秩。

解 键入:

$$a = [2 \ 2 \ 1; -3 \ 12 \ 3; 8 \ -2 \ 1; 2 \ 12 \ 4]; \text{rank}(a)$$

回车得出:

$$\text{ans} = \\ 2$$

2. 求矩阵零空间指令 null

齐次线性代数方程组的矩阵形式为 $Ax=0$,系数矩阵 A 为 $m \times n$ 阶矩阵, x 为 n 维列阵。由线性代数理论可知以下两个结论。

① 当系数矩阵的秩 $R(A) = r = n$ 时,方程组只有零解。

② 当系数矩阵的秩 $r < n$ 时,方程组有无穷多组解,含有 $(n - r)$ 个基础解系。 $r < n$ 时,使 $Ax=0$ 的全体解向量 x 称为 A 的零空间。

MATLAB 中直接调用指令 $x = \text{null}(A)$ 就可以求出满足 $Ax=0$ 的一个解向量 x 。把指令 null 用于求解齐次线性代数方程组时,需注意以下几个方面。

① 输入矩阵 A 必须满足 $R(A) = r < n$, n 为未知量个数。

② 输出参量 x 为 A 的零空间,即 $Ax=0$ 的解向量 x ,它的维数等于 $n - r$,它满足 $x'x = E$ (E 是 $(n - r)$ 阶的单位方阵)。

③ 输入的系数矩阵 A 可以是数值矩阵,也可以是符号矩阵,即:

$$x = \text{null}(A) \quad \text{或} \quad \text{null}(\text{sym}(A))$$

前者给出零空间的近似数值解,后者给出最接近零空间数值解的有理式。

在求解欠定的非齐次方程组通解时,需要先求出其对应齐次方程组的通解,这就需要利用零空间指令 null 。

例 5-2 求齐次方程组
$$\begin{cases} x+2y+2z+w=0 \\ 2x+y-2z-2w=0 \\ x-y-4z-3w=0 \end{cases}$$
 的解。

解 键入方程组系数矩阵：

```
a=[1 2 2 1;2 1 2 2;1 1 4 3]; r=rank(a)
```

回车得出：

```
r =  
2
```

方程组系数矩阵的秩为 2 待求变量数 $n=4$, 所以其基础解系含有 $4-2=2$ 个向量。

键入：

```
c=null(a)
```

回车得出：

```
c =  
0.7177    0.0286  
0.6084    0.2725  
0.0857    0.6241  
0.3277    0.7317
```

这个解向量是由正交基构成的 , 可键入 $x' * c$ 回车得出单位矩阵：

```
ans =  
1.0000    0.0000  
0.0000    1.0000
```

该题也可键入：

```
c1=null(sym(a))
```

回车得出：

```
c1 =  
[2,    5/3 ]  
[ 2,    4/3 ]  
[1,     0 ]  
[0,     1 ]
```

于是解向量为 $[x \ y \ z \ w] = k_1 \xi + k_2 \eta = k_1 [2 \ -2 \ 1 \ 0] + k_2 [5/3 \ -4/3 \ 0 \ 1]$ 即 $x = \frac{5}{3}k_2 + 2k_1$ $y = -2k_1 - \frac{4}{3}k_2$ $z = k_2$, $w = k_1$ 其中 $k_1 \ k_2$ 为任意常数 ξ 和 η 为方程组的一个基础解系。

5.4.2 求解非齐次线性代数方程组的 MATLAB 方法

对于非齐次线性代数方程组 $Ax = b$, 可根据解的结构分下列几种情况求解。

1. 恰定方程组

这时方程组系数满足条件 $R(A) = R(B) = r = n$, A 为 n 阶方阵 若 $\det(A) \neq 0$, 则方程组

有唯一解。从矩阵方程 $Ax = b$ 容易得出 $x = A^{-1}b$, x 有下列几种求法。

① 用求方阵“逆”的指令直接实现: $x = A^{-1}b$ 指令格式为 $x = \text{inv}(A) * b$ 。

② 用矩阵的左除法进行求解, 指令格式为 $x = A \backslash b$ 。

③ 用符号矩阵求解, 指令格式为 $x = \text{sym}(A) \backslash \text{sym}(b)$, 这种格式的输出是最接近精确值的有理数, 只是速度最慢。

矩阵除法运算 $x = A \backslash b$ 和 $x = \text{inv}(A) * b$ 在原理上都是用高斯消元法求解。但 MATLAB 中矩阵除法是通过矩阵分解进行运算的, 所以计算速度比先求逆阵再解方程要快得多。使用中应该注意 b 是非奇次线性方程组的自由项, 不是增广矩阵 B 。

例 5-3 求解方程组

$$\begin{cases} 2x + y - 5z + w = 8 \\ x - 3y - 6w = 9 \\ 2y - z + 2w = -5 \\ x + 4y - 7z + 6w = 0 \end{cases}$$

解 (1) 键入:

```
A=[2 1 5 1;1 3 0 6;0 2 4 2;1 4 7 6];b=[8 9 5 0]';
c=rank(A)=rank([A b])
```

回车得出:

```
c =
1
```

可见, 系数矩阵和增广矩阵的秩相等, 表明方程有解。

检验秩 r 与 $n=4$ 的关系, 键入:

```
rank(A)=4
```

回车得出:

```
ans =
1
```

表明系数矩阵和增广矩阵的秩都等于待求变量数 4, 正是恰定方程组, 有唯一解。

(2) 键入:

```
xx=A\b
```

回车得出:

```
xx =
3.0000
4.0000
4.0000
1.0000
```

于是就得出方程组的解为 $x=3.0000$, $y=-4.0000$, $z=-1.0000$, $w=1.0000$ 。

2. 欠定方程组(不定方程组)

当线性方程组 $Ax = b$ 满足 $R(A) = R(B) = r < n$ 时, 方程组有无穷多组解。它的通解由与其对应的齐次方程 $Ax = 0$ 的通解和 $Ax = b$ 的一个特解构成。求 $Ax = 0$ 的通解用 `null` 指令, 求 $Ax = b$ 的一个特解用矩阵除法或其他方法。用“左除”得出欠定方程组 $Ax = b$ 的一个

特解时,所含非零元素的个数最多等于系数矩阵 A 的秩 r ,是所有解中含零最多的一个。

例 5-4 求非齐次线性方程组
$$\begin{cases} x + y - 3z - w = 1 \\ 3x - y - 3z + 4w = 4 \\ x + 5y - 9z - 8w = 0 \end{cases}$$
 的通解。

解 (1) 解的判断

键入：

```
A=[1 1 3 -1;3 -1 3 4;1 5 -9 -8];b=[1 4 0]';Ar=rank(A),br=rank([A
b])
```

回车得出：

```
Ar =
    2
br =
    2
```

方程组系数的秩和增广矩阵的秩相等,说明方程有解。但它们的秩 $R(A)=R(b)=2$,小于待求变量个数 $n=4$,故为欠定方程组。

(2) 求出与 $Ax=b$ 对应的齐次方程 $Ax=0$ 通解

由于 $n-r=2$,对应的齐次方程组含有两个基向量。在指令窗中键入：

```
null(sym(A))
```

回车得出：

```
ans =
    [3/2, 3/4]
    [3/2, 7/4]
    [ 1,    0]
    [ 0,    1]
```

(3) 求 $Ax=b$ 的一个特解

在指令窗中键入：

```
A\b
```

回车得出：

```
Warning: Rank deficient,rank=2   tol = 8.8373e-015.
ans =
    0
    0
    0.5333
    0.6000
```

若键入：

```
sym(A)\sym(b)
```

回车得出：

```
Warning: System is rank deficient. Solution is not unique.
```

```
ans =
    5/4
    1/4
     0
     0
```

(4) 方程组 $Ax = b$ 的通解

方程组 $Ax = b$ 的通解是由它的一个特解和方程组 $Ax = 0$ 的通解组成, 这些前面已经分别求出, 将它们组合在一起就是非齐次线性方程组的通解:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = k_1 \begin{bmatrix} \frac{3}{2} \\ \frac{3}{2} \\ 1 \\ 0 \end{bmatrix} + k_2 \begin{bmatrix} -\frac{3}{4} \\ \frac{7}{4} \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} \frac{5}{4} \\ -\frac{1}{4} \\ 0 \\ 0 \end{bmatrix} \quad k_1, k_2 \text{ 为任意常数。}$$

3. 超定方程组(矛盾方程组)

如果方程组系数矩阵的秩小于增广矩阵的秩, 即 $R(A) < R(B) = R([A \ b])$, 表明方程组保留方程的个数大于待求变量的个数, 这种矛盾方程组一般意义下是无解的。但是, 在 MATLAB 中仍可用左除 $A \setminus b$ 方法求出它的最小二乘解, 即虽然这个解不满足任何一个方程, 但把它代入每个方程左边的得数与自由项差值的平方和, 比任何其他解的都小。

但应注意, 由于超定方程组没有精确解, 所以不能用符号矩阵除法来求解超定方程组。

例 5-5 求解线性方程组
$$\begin{cases} x - 2y + 3z - w = 1 \\ 3x - y + 5z - 3w = 2 \\ 2x + y + 2z - 2w = 3 \end{cases}$$

解 (1) 判定方程组解的结构

键入:

```
a=[1 2 3 4;3 4 5 3;2 1 2 2];b=[1 2 3]';
ra=rank(a),rb=rank([a b])
```

回车得出:

```
ra =
    2
rb =
    3
```

表明 $R(A) < R(B)$ 是超定方程组。

(2) 求方程组的最小二乘解

键入:

```
a \ b
```

回车得出:

```
Warning: Rank deficient,rank=2   tol= 5.4751e-015.
```

```
ans =
     0
    0.9048
    0.7143
     0
```

如果用符号矩阵求解,键入 `sym(a)\sym(b)`,回车将指出是错误的,用“inf”提示该方程组无精确解。

若键入:

```
sym(a\b)
```

回车得出:

```
0 19/21 5/7 0
```

这是对数值解结果的近似有理化,即表示 $0.9048 \approx 19/21$ $0.7143 \approx 5/7$ 。应该严格区分 `sym(a\b)` 与 `sym(a)\sym(b)` 的差别。

5.4.3 矩阵分解指令

在矩阵的计算和应用中,经常遇到矩阵分解的运算,MATLAB 中提供了许多矩阵分解指令可供选用。下面介绍两个对矩阵进行三角分解的指令。

1. 方阵的三角分解指令 `lu`

求方阵行列式的值、矩阵的逆阵及方阵除法等运算,在 MATLAB 中都是通过对矩阵进行三角分解后进行的。对矩阵 A 进行三角分解的指令 `lu` 的调用格式为:

```
[L,U]=lu(A)
```

```
[L,U,P]=lu(A)
```

① 输入的矩阵 A 必须是方阵;

② 输出参量用 `[L,U]` 格式时 L 为准下三角阵(交换 L 的两行后才能成为真正的下三角阵) U 为上三角矩阵,满足 $A \approx LU$;

③ 输出参量用 `[L,U,P]` 格式时 L 为下三角阵 U 为上三角矩阵 P 为变换方阵元素位置的换位阵,它们满足 $PA \approx LU$ 。

例 5-6 已知 $a = \begin{bmatrix} 1 & 4 & 2 & 3 \\ 5 & 1 & 0 & 2 \\ 2 & 4 & 3 & 0 \\ 0 & 2 & 1 & 6 \end{bmatrix}$ 对它进行三角分解。

解 在指令窗中键入:

```
a=[1 4 2 3;5 1 0 2;2 4 3 0;0 2 1 6]; [m u]=lu(a)
```

回车得出:

```
m =
    0.2000    1.0000         0         0
```

```

1.0000      0      0      0
0.4000    0.9474    1.0000      0
      0    0.5263   -0.0476    1.0000

u =
5.0000    1.0000      0    2.0000
      0    3.8000    2.0000    2.6000
      0      0    1.1053    3.2632
      0      0      0    4.4762

```

可以看出 m 是准下三角阵, 将其一、二两行调换后就是下三角阵。

为了验证分解的正确性, 键入:

```
m*u
```

回车得出:

```

ans =
1.0000    4.0000    2.0000    3.0000
5.0000    1.0000      0    2.0000
2.0000    4.0000    3.0000      0
      0    2.0000    1.0000    6.0000

```

这个结果和矩阵 a 完全相等。

若键入:

```
[m1 u1 p]=u(a)
```

回车得出:

```

m1 =
1.0000      0      0      0
0.2000    1.0000      0      0
0.4000    0.9474    1.0000      0
      0    0.5263   -0.0476    1.0000

u1 =
5.0000    1.0000      0    2.0000
      0    3.8000    2.0000    2.6000
      0      0    1.1053    3.2632
      0      0      0    4.4762

p =
      0      1      0      0
      1      0      0      0
      0      0      1      0
      0      0      0      1

```

为了验证结果的正确性, 可以键入:

```
m1*u1
```

回车得出：

```
ans =
    5.0000    1.0000         0    2.0000
    1.0000    4.0000    2.0000    3.0000
    2.0000    4.0000    3.0000         0
         0    2.0000    1.0000    6.0000
```

这是 $p * a$ 的结果。可以看出, 把它的元素位置略加变换就可以成为方阵 a 。

2. 方阵的乔累斯基(Cholesky)分解指令 chol

方阵 A 的乔累斯基分解就是找出一个下三角阵 L , 使方阵 $A = LL^T$ 的分解。

用指令 `chol` 可以实现这种分解, 调用格式为：

```
chol(A)
```

- ① 输入矩阵 A 必须是对称的正定方阵。
- ② 输出为一个与 A 同维的上三角阵 L , 满足 $LL^T = A$ 。
- ③ 可以用该指令检验 A 是否为对称正定方阵, 若不能被分解, 则不是正定方阵。

例 5-7 将矩阵 $a = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 9 \end{bmatrix}$ 进行乔累斯基分解。

解 在指令窗中键入：

```
a1=[1 1 2;1 2 0;2 0 9];c1=chol(a1)
```

回车得出：

```
c1 =
    1    1    2
    0    1    2
    0    0    1
```

可以进行验证, 键入：

```
c1'*c1
```

回车得出：

```
ans =
    1    1    2
    1    2    0
    2    0    9
```

任何一个方阵, 如果能进行乔累斯基分解, 它就是对称的正定阵。仅仅检验矩阵是否对称, 可以用逻辑语句 $a' == a$, 回车得 `ones(size(a))` 则 a 为对称阵。

练习题 5

1. 确定方程组解的结构并求其解：

$$(1) \begin{cases} x_1 + 4x_2 - 2x_3 + 3x_4 = 6 \\ 2x_1 + 2x_2 + 4x_4 = 2 \\ 3x_1 - x_3 + 2x_4 = 1 \\ x_1 + 2x_2 + 2x_3 - 3x_4 = 8 \end{cases}$$

$$(2) \begin{cases} x_1 + 2x_2 + 3x_3 = 2 \\ 2x_1 + 3x_2 + 4x_3 = 3 \\ 3x_1 + 4x_2 + 4x_3 = 3 \end{cases}$$

$$(3) \begin{cases} 10^{-5}x_1 + 10^{-5}x_2 + x_3 = 2 \times 10^{-5} \\ 10^{-5}x_1 - 10^{-5}x_2 + x_3 = -2 \times 10^{-5} \\ x_1 + x_2 + 2x_3 = 1 \end{cases}$$

$$(4) \begin{cases} x_1 - 2x_2 + 4x_3 = 56 \\ 2x_1 + 8x_2 + x_3 = -4 \\ 20x_1 - x_2 + 2x_3 = 74 \end{cases}$$

$$(5) \begin{cases} 0.5x_1 + 1.1x_2 + 3.1x_3 + 1.1x_4 = 9.85 \\ 2.0x_1 + 4.5x_2 + 0.36x_3 + 0.1x_4 = 0.37 \\ 5.0x_1 + 0.96x_2 + 6.5x_3 + 2.5x_4 = 9.71 \\ 0.6x_1 + 2.5x_2 + 0.55x_3 + 0.4x_4 = 3.44 \end{cases}$$

$$(6) \begin{cases} 0.5x_1 + 1.1x_2 + 3.1x_3 = 6.0 \\ 6.5x_1 - 5.78x_2 - 8.7x_3 = -40.8 \\ 5.0x_1 + 0.96x_2 + 6.5x_3 = 0.96 \end{cases}$$

$$(7) \begin{bmatrix} 1 & 0.25 & & & \\ -0.25 & 1 & & & \\ & -0.25 & 1 & & \\ & & -0.25 & 1 & \\ & & & -0.25 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

2. 据各矩阵的条件 对它们进行三角分解和乔累斯基分解：

$$(1) \begin{bmatrix} 5 & 7 & 3 \\ 7 & 11 & 2 \\ 3 & 2 & 6 \end{bmatrix}$$

$$(2) \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

$$(3) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}$$

$$(4) \begin{bmatrix} 10 & 5 & 4 & 3 & 2 & 1 \\ -1 & 10 & 5 & 4 & 3 & 2 \\ -2 & -1 & 10 & 5 & 4 & 3 \\ -3 & -2 & -1 & 10 & 5 & 4 \\ -4 & -3 & -2 & -1 & 10 & 5 \\ -5 & -4 & -3 & -2 & -1 & 10 \end{bmatrix}$$

第 6 章 求解线性代数方程组和计算矩阵特征值的迭代法

在求解线性代数方程组的方法中,迭代法与直接法相比具有程序简单、占用存储单元少等优点,适合于求解高阶,特别是零系数较多的稀疏线性方程组。此外,迭代法是数值计算中非常重要的一种计算方法,在求矩阵特征值和特征向量时也常用到。

本章通过介绍迭代法在求解线性代数方程组和计算矩阵特征值时的应用,着重使我们对迭代法的基本思想有所了解,同时从实用角度考虑介绍了 MATLAB 在求算矩阵特征参数、行列式、正交三角分解等方面的应用。

6.1 求解线性代数方程组的迭代法

大型、稀疏线性方程组的求解,用第 5 章介绍过的直接法是非常烦琐的,而迭代法的优点则给这类问题的计算带来许多方便。同时,迭代法的基本思想在计算方法中占有非常重要的地位,第 4 章已有所接触,本章将结合它的应用再做一些深入的介绍。

6.1.1 迭代法的基本原理

线性代数中已经讲过,如果线性方程组 $Ax=b$ 的系数矩阵 A 非奇异,则方程组有唯一解。把这种方程中的方阵 A 分解成两个矩阵之差:

$$A = C - D$$

若方阵 C 是非奇异的,把 A 代入方程 $Ax=b$ 中,得出 $(C - D)x=b$,两边左乘 C^{-1} ,并令 $M = C^{-1}D$, $g = C^{-1}b$ 移项可将方程 $Ax=b$ 变换成:

$$x = Mx + g \quad (6-1)$$

据此便可构造出迭代公式:

$$x_{k+1} = Mx_k + g, k=0, 1, 2, \dots \quad (6-2)$$

式(6-2)中的 $M = C^{-1}D$ 称为迭代矩阵,知道 x_k 就可以求出 x_{k+1} 。设 x_0 是方程组 $Ax=b$ 的一个初始近似解向量,把它代入公式(6-2)便可得出一个迭代序列:

$$x_0, x_1, x_2, \dots, x_k, \dots \quad (6-3)$$

如果这个序列收敛于某一向量 x^* ,即 $k \rightarrow \infty$ 时, $x_k \rightarrow x^*$,则称迭代公式(6-2)收敛,否则为发散。收敛的迭代公式可以用于求解线性方程组 $Ax=b$,收敛迭代序列的每个元素 x_k 向量就是方程组的一个近似解。据此原理可有多种迭代方法,下面仅介绍两种。

6.1.2 雅可比迭代法

1. 雅可比迭代公式

根据迭代法的基本原理,如果方程组 $Ax=b$ 的系数矩阵 A 非奇异,主对角元素不为零,即 $a_{ii} \neq 0 \quad i=1, 2, \dots, n$, 若可以把 A 分解成:

$$A = D - L - U = D + (-L) + (-U)$$

其中, 对角阵 $D = \text{diag}(a_{11}, a_{22}, \dots, a_{nn})$;

- L 是严格下三角阵, 即主对角线及其右上方元素均为零的三角阵, 其主对角线左下方元素均取自 A 的对应元素;

- U 是严格上三角矩阵, 即主对角线及其左下方元素均为零的三角阵, 其主对角线右上方元素均取自 A 中对应元素。

把 $A = D - L - U$ 代入方程 $Ax=b$, 得到等价方程组 $Dx = (L+U)x + b$, 由于 D 可逆, 此式可以变换成 $x = D^{-1}((L+U)x + b) = D^{-1}(L+U)x + D^{-1}b$, 由此可以构造出迭代公式:

$$x_{k+1} = D^{-1}((L+U)x_k + b) = D^{-1}(L+U)x_k + D^{-1}b \quad (6-4)$$

该式是式(6-2)的一种特殊形式, 称为雅可比迭代公式。

若已知 x_0 是方程组 $Ax=b$ 的一个初始近似解向量, 代入式(6-4)右侧可以得出 x_1 , 把 x_1 代入式(6-4)右侧得出 x_2 , 反复上述步骤便可得到一个迭代向量序列: $x_0, x_1, x_2, \dots, x_k, \dots$ 。把 $M = D^{-1}(L+U)$ 称为雅可比(Jacobi)迭代矩阵, 于是式(6-4)可以写成:

$$x_{k+1} = D^{-1}(L+U)x_k + D^{-1}b = Mx_k + D^{-1}b$$

例 6-1 已知方阵 $A = \begin{bmatrix} 2 & 1 & -5 & 1 \\ 1 & -3 & 0 & -6 \\ 0 & 2 & -1 & 2 \\ 1 & 4 & -7 & 6 \end{bmatrix}$, 试求它的雅可比迭代矩阵。

解 因为 $\det(A) = 27 \neq 0$, 且 $a_{ij} \neq 0$, 所以可将 A 分解: $A = D - L - U$ 其中:

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}, L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ -1 & -4 & 7 & 0 \end{bmatrix}, U = \begin{bmatrix} 0 & -1 & 5 & -1 \\ 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

于是, 可得雅可比迭代矩阵:

$$M = D^{-1}(L+U) = \begin{bmatrix} 0 & -1/2 & 5/2 & -1/2 \\ 1/3 & 0 & 0 & -2 \\ 0 & 2 & 0 & 2 \\ -1/6 & -2/3 & 7/6 & 0 \end{bmatrix}$$

2. 雅可比迭代公式的向量形式

为了将迭代公式(6-4)写成向量的分量形式, 设迭代序列中第 k 个元素是向量 $x_k (k=0, 1, 2, \dots)$, 它的第 j 个分量写成 $(x_k)_j (j=1, 2, \dots, n)$ 则:

$$x_k = [(x_k)_1, (x_k)_2, \dots, (x_k)_n]^T \quad k=0, 1, 2, \dots$$

由于 $D^{-1} = \text{diag}\left(\frac{1}{a_{11}}, \frac{1}{a_{22}}, \dots, \frac{1}{a_{nn}}\right)$, 则雅可比迭代公式的分量形式为:

$$(x_{k+1})_i = \frac{1}{a_{ii}} \left(- \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} (x_k)_j + b_i \right) \quad i = 1, 2, \dots, n \quad (6-5)$$

式中 $(x_k)_i$ 表示迭代序列中第 k 个元素 x_k 向量的第 i 个分量。

6.1.3 赛德尔迭代法

为了提高雅可比迭代向量序列的收敛速度和计算精度,在对雅可比方法进行改进中提出了赛德尔(Seidel)迭代法。依照雅可比迭代法,由式(6-5)可知,要计算 x_{k+1} 的一个分量 $(x_{k+1})_i$ 得用向量 x_k 的全部分量 $(x_k)_j (j=1, 2, \dots, n)$ 。实际上,在计算 x_{k+1} 的第 $i (i>1)$ 个分量 $(x_{k+1})_i$ 时, x_{k+1} 前面的 $(i-1)$ 个分量 $(x_{k+1})_1, (x_{k+1})_2, \dots, (x_{k+1})_{i-1}$ 都已经算出,而且通常它们比 $(x_k)_1, (x_k)_2, \dots, (x_k)_{i-1}$ 更接近精确解 x^* 的前 $(i-1)$ 个分量。因此可以用 $(x_{k+1})_1, (x_{k+1})_2, \dots, (x_{k+1})_{i-1}$ 代替 $(x_k)_1, (x_k)_2, \dots, (x_k)_{i-1}$, 从而可以使计算更加方便,而且计算结果也更加精确。按此思路,把式(6-5)中对 x_k 前 $(i-1)$ 个分量的求和,用 x_{k+1} 前 $(i-1)$ 个分量的求和代替,这样就得出赛德尔迭代公式的分量表达式:

$$(x_{k+1})_i = \frac{1}{a_{ii}} \left(- \sum_{j=1}^{i-1} a_{ij} (x_{k+1})_j - \sum_{j=i+1}^n a_{ij} (x_k)_j + b_i \right) \quad i = 1, 2, \dots, n \quad (6-6)$$

由这个分量表达式可以推出它的矩阵形式。为此将它变换移项,写成:

$$a_{ii} (x_{k+1})_i + \sum_{j=1}^{i-1} a_{ij} (x_{k+1})_j = - \sum_{j=i+1}^n a_{ij} (x_k)_j + b_i \quad i = 1, 2, \dots, n$$

仍然沿用雅可比迭代公式中的矩阵符号 D, L 和 U 则有:

$$(D - L)x_{k+1} = Ux_k + b$$

于是便可得出赛德尔迭代公式的矩阵形式:

$$x_{k+1} = (D - L)^{-1} Ux_k + (D - L)^{-1} b \quad (6-7)$$

把 $M = (D - L)^{-1} U$ 称为赛德尔迭代矩阵。

利用式(6-7)可以很容易地求出例6-1中矩阵 A 的赛德尔迭代矩阵:

$$M = (D - L)^{-1} U = \begin{bmatrix} 0 & -\frac{1}{2} & \frac{5}{2} & -\frac{1}{2} \\ 0 & -\frac{1}{6} & \frac{5}{6} & -\frac{13}{6} \\ 0 & -\frac{1}{3} & \frac{5}{3} & -\frac{7}{3} \\ 0 & -\frac{7}{36} & \frac{35}{36} & -\frac{43}{36} \end{bmatrix}$$

6.1.4 迭代法的敛散性

通过对方程组 $Ax = b$ 系数矩阵的分解变形,可以得出迭代公式(6-2)。但是,只有当这个公式收敛时,才可以构造出一个收敛于方程组解的向量序列,也只有这时序列中的每个元素才能成为方程组的一个不同精度解。而序列收敛与否,或者说,迭代公式收敛与否的关键在于迭代矩阵。下面首先讨论矩阵的一些性质,这些性质跟迭代公式的敛散性密切相关。

1. 方阵的谱半径

在迭代公式的一般形式 $x_{k+1} = Mx_k + g, (k=0, 1, 2, \dots)$ 中, 若取 $M = D^{-1}(L + U)$ 就得出雅可比迭代公式; 若取 $M = (D - L)^{-1}U$ 就得出赛德尔迭代公式。无论哪种形式的迭代公式, 只有形成的迭代序列收敛(称迭代公式收敛), 才可用于求解线性方程组 $Ax = b$ 。由迭代公式的一般形式方程(6-2), 可以推出迭代矩阵序列为:

$$x_1 = Mx_0 + g$$

$$x_2 = Mx_1 + g = M(Mx_0 + g) + g = M^2x_0 + (M + E)g$$

...

$$x_{k+1} = M^{k+1}x_0 + (M^k + M^{k-1} + \dots + M + E)g = M^{k+1}x_0 + g \sum_{n=0}^k M^n$$

...

理论可以证明, 这个矩阵序列收敛与否只跟迭代矩阵 M 有关, 收敛的充要条件是:

$$\lim_{k \rightarrow \infty} M^k = 0 \quad (6-8)$$

由矩阵理论知道, 满足 $\lim_{k \rightarrow \infty} M^k = 0$ 的充要条件是方阵 M 的谱半径 $\rho(M) < 1$, 即

$$\rho(M) = \max_{1 \leq i \leq n} |\lambda_i| < 1 \quad (6-9)$$

方阵 M 的谱半径 $\rho(M)$ 就是 M 最大特征值的绝对值。

由于具体问题中方阵的谱半径不宜求算, 所以, 总是用与它等价而容易计算的数学量代替它去判断迭代公式的收敛性, 并借以估计迭代矩阵序列近似值的误差。比较容易计算的矩阵范数就属于这样的数学量, 为此先介绍范数的概念。

2. 向量范数

设 n 维向量 $x = [x_1, x_2, \dots, x_n]^T$, 按照一定的规则确定一个实数与它对应, 这个实数记为 $\|x\|$, 如果它满足下面的所谓“范数三条件”:

① 非负性—— $\|x\| \geq 0$, 当 $x \neq 0$ 时, $\|x\| > 0$; 当 $x = 0$ 时, $\|x\| = 0$;

② 齐次性——对任意实数 a , 都有 $\|a x\| = |a| \|x\|$;

③ 三角不等式——对于任意维数相等的向量 x, y , 都有 $\|x\| + \|y\| \geq \|x + y\|$ 。

则称该实数 $\|x\|$ 为向量 x 的范数。

n 维向量的范数是三维向量中长度概念的推广。就像实数与角度的对应关系不同可以定义出多种三角函数一样, 由于所选实数与向量的对应关系不同, 可以定义出多种范数, 经常使用的是下列几种称之为基本范数的向量范数。

(1) 1-范数

$$\|x\|_1 = |x_1| + |x_2| + \dots + |x_n| = \sum_{i=1}^n |x_i|$$

(2) 2-范数(Euclidean 范数)

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} = \left(\sum_{i=1}^n x_i^2 \right)^{1/2}$$

(3) 无穷范数

$$\|x\| = \max(|x_1|, |x_2|, \dots, |x_n|)$$

这三种范数可以用如下的一般公式表示：

$$\|x\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{1/p} = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

根据需要选取 $p=1, 2, \dots$, 分别对应于上面的三种不同的向量范数。

3. 矩阵范数

1) 矩阵范数的定义及性质

按照一定法则确定的与矩阵 A 对应的实数 $\|A\|$, 如果满足范数三条件, 同时满足矩阵乘法相容性(次乘性): $\|A\| \|B\| \geq \|AB\|$ (B 为与 A 同维数的矩阵), 则称实数 $\|A\|$ 为矩阵 A 的矩阵范数。

对一个矩阵, 根据其和实数对应关系的不同, 可以定义多个范数。常用的“算子范数”的矩阵范数定义是: 设 A 是 $m \times n$ 阶矩阵, x 是 n 维向量, 则矩阵 A 的范数为:

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\| \quad (6-10)$$

可以证明, 这样定义的范数 $\|A\|$ 完全满足范数三条件和次乘性, 而且与向量范数 $\|x\|$ 具有所谓相容性, 即 $\|A\| \|x\| \geq \|Ax\|$ 。 $\|A\|$ 是从向量范数诱导而成的, 也称诱导范数。

此外, 范数具有等价性, 即对任意的 n 维向量(或 n 阶方阵) x , 任意两种范数 $\|x\|_{p_1}$ 和 $\|x\|_{p_2}$, 总存在与 p_1, p_2 范数有关的正数 M, m 使下述关系成立:

$$M \|x\|_{p_1} \geq \|x\|_{p_2} \geq m \|x\|_{p_1} \quad (6-11)$$

矩阵范数的这种等价性, 使得一个矩阵序列在一种范数下收敛, 则在另一种范数下也是收敛的。于是, 在理论推导或计算中, 可以根据需要和方便程度选用不同的范数, 而所得结果具有普遍性。

2) 常用的几种矩阵范数

根据矩阵范数的定义, 常用的几种矩阵范数有下列几种。

(1) 列和范数(1-范数)

对每列所有元素的绝对值求和, 取其最大值为矩阵范数:

$$\|A\|_1 = \max_{1 \leq j \leq n} \left(\sum_{i=1}^m |a_{ij}| \right)$$

(2) 行和范数(∞ -范数)

对每行所有元素的绝对值求和, 取其最大值为矩阵范数:

$$\|A\|_\infty = \max_{1 \leq i \leq m} \left(\sum_{j=1}^n |a_{ij}| \right)$$

(3) 矩阵的 Frobenius 范数(F-范数)

矩阵所有元素平方求和后再开方(与向量 Euclidean 范数对应, 也叫欧氏范数或 Schur 范数):

$$\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{1/2}$$

和向量范数一样, 以上三种范数可用下面的一般公式表示:

$$\|A\|_p = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^p \right)^{1/p}, (1 \leq p < \infty)$$

(4) 矩阵的谱范数

设 $A^H = (\bar{A})^T$ (A 的共轭转置阵) $\rho(A^H A)$ 为 Hermite 矩阵 $A^H A$ 的谱半径, 矩阵 A 的谱范数定义为:

$$\|A\|_2 = \sqrt{\rho(A^H A)}$$

4. 迭代公式收敛性的判断和误差估计

1) 方阵谱半径和范数间的关系

据定义, 若 λ 是方阵 M 的特征值, x 为与其对应的特征向量, 则有:

$$\lambda x = Mx$$

两边取范数, 并利用范数的有关性质可得: $|\lambda| \|x\| \leq \|M\| \|x\|$, 由于 $\|x\| \neq 0$, 所以 $|\lambda| \leq \|M\|$ 。因为 λ 是方阵 M 的任意一个特征值, 而方阵 M 的谱半径 $\rho(M) = \max_{1 \leq i \leq n} |\lambda_i|$, 故 n 阶方阵 M 的谱半径 $\rho(M)$ 和方阵的范数 $\|M\|$ 存在下述关系:

$$\rho(M) \leq \|M\| \quad (6-12)$$

2) 迭代公式的误差估计

由式(6-12)可知, 判断迭代公式收敛的条件 $\rho(M) = \max_{1 \leq i \leq n} |\lambda_i| < 1$, 可以换成方阵范数的下述表示式: $\|M\| < 1$, 而方阵 M 的范数要比它的谱半径容易计算得多。所以, 根据迭代矩阵的范数 $\|M\| < 1$, 就可以判断迭代公式 $x_{k+1} = Mx_k + g$ 收敛, 于是 $k \rightarrow \infty$ 时, $x_k \rightarrow x^*$ 。据此可由理论推出误差估计式:

$$\|x^* - x_k\| \leq \frac{\|M\|}{1 - \|M\|} \|x_k - x_{k-1}\| \text{ 或 } \|x^* - x_k\| \leq \frac{\|M\|^k}{1 - \|M\|} \|x_1 - x_0\| \quad (6-13)$$

6.2 方阵特征值和特征向量的计算

在研究振动与波及自动控制中的稳定性等问题时, 数学化后的求解常常归结为求一些矩阵的特征值和特征向量。矩阵特征值的计算方法分两类。一类是解多项式方程, 就是由矩阵得出其特征多项式和特征方程, 特征方程的根就是矩阵的特征值。但是, 当矩阵的阶数很高时, 由于高次多项式方程的求根较为繁杂, 而且重根将使计算精度降低, 于是这一方法并不理想。另一类方法是迭代法, 它是构造一个极限为矩阵特征值和特征向量的收敛序列, 序列中的每个元素都是特征值和不同误差特征向量的近似值。这种方法的迭代循环, 给计算机求解带来了方便, 所以广为流行。下面对这两类方法的基本原理做一些简单介绍。

6.2.1 方阵特征方程的求解

线性代数中讲过, 如果 n 阶方阵 A , n 维向量 x 和数 λ 满足关系: $Ax = \lambda x$, 就把数 λ 叫作方阵 A 的特征值, 非零向量 x 是与 λ 对应的方阵 A 的特征向量。移项可得:

$$(A - \lambda E)x = 0$$

式中 E 为 n 阶单位方阵。该方程有非零解的充分必要条件是系数行列式等于零,即:

$$\det(A - \lambda E) = 0$$

该方程式为方阵 A 的特征方程。方程左端的 $\det(A - \lambda E)$ 是特征值 λ 的 n 次多项式,称为方阵 A 的特征多项式。

特征方程是一个以 λ 为未知量的 n 次代数方程,它的根就是方阵 A 的特征值。把特征值代入特征向量满足的方程 $Ax = \lambda x$ 中,就可以求出特征向量 x 。当 n 不太大时,经常用这一方法求方阵 A 的特征值和特征向量,但当 n 较大时,则需要用下述方法来求算。

6.2.2 计算特征值和特征向量的迭代法

迭代法计算方阵特征值和特征向量的方法有许多种,基本原理雷同。在此,仅介绍雅可比法和 QR(正交—三角形分解)算法。

1. 雅可比法

由线性代数知道,对于 n 阶实对称方阵 A ,必然存在正交阵 P ,使得:

$$P^{-1}AP = P^TAP = \Lambda$$

Λ 是对角阵。这个等式表明方阵 A 与对角阵 Λ 相似,由于相似方阵具有相同的特征多项式和相同的特征值,而对角阵 $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$,因此 $\lambda_j (j=1, 2, \dots, n)$ 就是 Λ 和 A 的特征值。同时知道,正交阵 P 的第 j 列就是对应于 λ_j 的特征向量。于是,如果能找到使方阵 A 对角化的正交阵 P ,就容易求得它的特征值和特征向量。雅可比法就是根据这个原理,用矩阵的正交相似变换,将实对称矩阵化为对角阵的一种方法,因为它相当于坐标的旋转变换,所以也叫旋转法。

先以一个二阶方阵的对角化为例引出雅可比法。

设实对称矩阵 $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$,如果设正交阵 $P = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix}$,则 $P^{-1} = P^T =$

$\begin{bmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{bmatrix}$,若取 $\varphi = \frac{1}{2} \arctan \frac{2b}{a-c}$,用 P^{-1} 左右两边乘 A 可得:

$$P^{-1}AP = \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} = \begin{bmatrix} a^{(1)} & 0 \\ 0 & a^{(2)} \end{bmatrix} = A_1$$

这表明 A 经过正交变换后成为对角阵 A_1 ,所以 A 和 A_1 相似。于是 A_1 的对角元素 $a^{(1)}$ 和 $a^{(2)}$ 就是 A 和 A_1 的特征值, $\begin{bmatrix} \cos \varphi \\ \sin \varphi \end{bmatrix}$ 和 $\begin{bmatrix} -\sin \varphi \\ \cos \varphi \end{bmatrix}$ 分别为对应于它们的特征向量。

这个变换方法可以推广至 n 阶实对称矩阵。设 n 阶实对称矩阵 A 的某对非对角线元素 $a_{ij} = a_{ji} \neq 0$,为将它们变换成零,仿照刚才二阶方阵的例子,取如下的正交矩阵 P_{ij} :

$$P_{ij} =$$

i 列

j 列

1

$\cos \varphi$

$\sin \varphi$

1

$-\sin \varphi$

$\cos \varphi$

1

1

1

i 行

j 行

对角线外的省略号均表示零。用 P_{ij} 对方阵 A 进行变换, 有 $P_{ij}^{-1} A P_{ij} = P_{ij}^T A P_{ij} = A_1$, 对它两边进行转置, 可得出:

$$A_1^T = (P_{ij}^T A P_{ij})^T = P_{ij}^T A^T P_{ij} = P_{ij}^T A P_{ij} = A_1$$

可见 A 与 A_1 相似, 且 A 对称时, A_1 也对称。另外, 由二阶方阵的例子可知, 若取 φ 满足关系式 $\varphi = \frac{1}{2} \arctan \frac{2a_{ij}}{a_{ii} - a_{jj}}$ 时, 就能使 A_1 的两个非对角元素 $a_{ij}^{(1)} = a_{ji}^{(1)} = 0$ 。可以不断进行这样的正交变换, 就能使 A 的所有非对角元素都变成零。设第 k 次变换得到的矩阵为 A_k , 如果它的某对非对角元素 $a_{pq}^{(k)} = a_{qp}^{(k)} \neq 0$, 就仿照上述办法构成正交矩阵 P_{pq} , 对 A_k 进行变换:

$$P_{pq}^T A_k P_{pq} = A_{k+1} \quad (6-14)$$

于是, 在变换后的矩阵 A_{k+1} 中非对角元素 $a_{pq}^{(k+1)} = a_{qp}^{(k+1)} = 0$ 。如此不断进行这样的变换, 就可得出一系列实对称矩阵 A_1, A_2, A_3, \dots , 它们将逐渐趋向于对角阵 Λ 。

实际运算并非如此简单, 在迭代过程的每次变换中, 虽然使被变换的非对角元素为零, 但还有可能使一些与被变换元素同行或同列、先前已经变成零的元素又成了非零元素。但是理论已经证明, 正交相似变换中方阵元素的平方和(方阵的 F -范数)保持不变, 而雅可比方法的每次变换, 却总是使对角线元素平方和增大, 非对角元素平方和减小, 所以该方法得出的矩阵序列肯定是收敛的, 即 $k \rightarrow \infty$ 时, 总有 $A_k \rightarrow \Lambda$ 。

雅可比方法的优点是同时可以求出实对称矩阵的特征值和特征向量, 算法稳定, 精度较高, 适用于计算阶数不太高、零元素较少矩阵(稠密矩阵)的特征值和特征向量。

2. QR 算法

由线性代数中的施密特(Schmidt)正交化方法可以推得, 任意 n 阶方阵 A 总可以分解成一个正交矩阵 Q ($Q^T Q = E$) 和一个上三角阵 R 的乘积:

$$A = QR \quad (6-15)$$

这种把矩阵分解成正交阵与上三角阵之积的过程, 叫正交三角分解或 QR 分解。如果 A 是非奇异方阵, 则这种分解是唯一的。QR 算法的理论基础就是矩阵的正交三角分解。

如果 A 为一个方阵, 设 $A_1 = A$, A_1 可以分解为正交阵 Q_1 和上三角阵 R_1 之积:

$$A_1 = Q_1 R_1$$

将 Q_1 、 R_1 顺序颠倒, 令 $A_2 = R_1 Q_1 = Q_1^{-1} A_1 Q_1$, 再对 A_2 重复上述步骤, 得到 A_3, \dots 不断重复这种变换, 则有 QR 算法的计算公式:

$$\begin{cases} A_k = Q_k R_k, Q_k^T Q_k = E, \\ A_{k+1} = R_k Q_k = Q_k^{-1} A_k Q_k \end{cases} \quad k=1, 2, \dots \quad (6-16)$$

E 为 n 阶单位阵。由此可以得出方阵序列 $A_1, A_2, \dots, A_k, \dots$ 。这个序列的每个方阵都与方阵 $A_1 = A$ 相似, 且由于:

$$A_{k+1} = R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^{-1} Q_{k-1}^{-1} A_{k-1} Q_{k-1} Q_k = \dots = Q_k^{-1} Q_{k-1}^{-1} \dots Q_1^{-1} A_1 Q_1 Q_2 \dots Q_k$$

若令 $G_k = Q_1 Q_2 \dots Q_k, k=1, 2, \dots$, 则 $A_{k+1} = G_k^{-1} A_1 G_k$

若令 $F_k = R_k R_{k-1} \dots R_1, k=1, 2, \dots$, 因 $A_k = Q_k R_k$

$$\begin{aligned} G_k A_{k+1} &= Q_1 Q_2 \dots Q_k A_{k+1} = Q_1 Q_2 \dots Q_k R_k Q_k \\ &= Q_1 Q_2 \dots Q_{k-1} A_k Q_k = Q_1 Q_2 \dots Q_{k-1} R_{k-1} Q_{k-1} Q_k = \dots = A_1 G_k, \quad k=1, 2, \dots \end{aligned}$$

所以 $G_k F_k = Q_1 Q_2 \dots Q_k R_k R_{k-1} \dots R_1 = G_{k-1} A_k F_{k-1} = A_1 G_{k-1} F_{k-1} = A_1^2 G_{k-2} F_{k-2} = \dots = A_1^k$
 因为 Q_1, Q_2, \dots, Q_k 是正交阵, 所以 G_k 也是正交阵, 同理 F_k 应是三角阵。由 $A_1^k = G_k F_k$ 可知, $A_1^k = A^k$ 也可进行正交三角分解。

理论证明, 当 $k \rightarrow \infty$ 时, 在一定的条件下方阵序列 $A_1, A_2, \dots, A_k, \dots$ 的主对角线元素趋于方阵 A 的特征值。

QR 算法的收敛速度是线性的, 而且运算量很大。但是它不限定方阵 A 必须对称, 有一定实用价值; 当然, 实际使用中还需进行许多改进, 这里不再介绍。

6.3 矩阵一些特征参数的 MATLAB 求算

以矩阵为运算单元的 MATLAB 中, 有许多求算矩阵特征参数的指令, 使用非常方便。这里只介绍跟求矩阵特征值有关的几个常用指令及 QR 分解指令, 要用其他指令时, 可用 help 调出“矩阵代数(matfun)库”查找。

6.3.1 求方阵特征值的有关指令

求方阵特征值和特征向量有许多方法, 现介绍相关的 MATLAB 指令。

1. 求方阵行列式的指令 det

方阵阶数很高时, 手工演算求行列式值非常麻烦, MATLAB 中设有专用指令 det, 它的调用格式为:

`det(A)`

输入参数 A 必须是数值方阵, 返回 A 的行列式值 $|A|$ 。

2. 求方阵特征多项式的指令 poly

方阵 A 的特征多项式 $\det(A - \lambda E)$ 是关于 λ 的代数多项式, 它的根就是方阵的特征值。求方阵特征多项式的专用指令是 poly, 调用格式为:

`P = poly(A)`

- ① 输入参数 A 必须是方阵;
- ② 输出参数 P 是 A 的特征多项式系数向量;
- ③ 用 `roots(P)` 可求得方阵 A 的特征值。

3. 求方阵特征值和特征向量指令 eig

方阵特征值的求法有多种, 上面介绍的先求特征多项式再求该多项式的根就是其中之一。但是, 在 MATLAB 中还有一个方便而专用的指令 eig, 它同时可以得出特征值和一组特征向量。调用格式为:

`[x r] = eig(A)`

`[x r] = eig(A, "nobalance")`

`eig(A)`

- ① 输入参数 A 必须是方阵；
- ② 输出参数 x 是一个矩阵，它的各列是方阵 A 的特征向量；
- ③ 输出参数 r 是一个对角阵，其元素是方阵 A 的特征值，r 与 x 同列向量相对应；
- ④ 当不写输出格式[x r]时，只输出由 A 的特征值为元素的列阵；
- ⑤ 如果 A 中含有小到跟截断误差相当的元素时，加写输入参数“nobalance”，它可以提高小元素的作用，通常可以省略该参数，以免使结果误差变大。

例 6-2 求方阵 $a = \begin{bmatrix} -2 & 1 & 1 \\ 0 & 2 & 0 \\ -4 & 1 & 3 \end{bmatrix}$ 的行列式值、特征多项式、特征值和特征向量。

解 在指令窗中键入：

```
a3 = [-2 1 1; 0 2 0; -4 1 3]; det(a3)
```

回车得出：

```
ans =  
4
```

a3 的行列式值 $|a3| = 4$ 。

键入：

```
P = poly(a3)
```

回车得出：

```
P =  
1 3 0 4
```

P 为 a3 特征多项式的系数向量，用 poly2str 指令可得出多项式表达式，键入：

```
P1 = poly2str(P, 'y')
```

回车得出：

```
P1 =  
y^3 - 3 y^2 + 4
```

求特征多项式的根，即解方程 $y^3 - 3y^2 + 4 = 0$ ，键入：

```
roots(P)
```

回车得出：

```
ans =  
2.0000  
2.0000  
-1.0000
```

a3 的特征值是 2, 2, -1。

不写输出变量时，eig 指令只输出特征值，如键入：

```
eig(a3)
```

回车得出：

```
ans =  
-1  
2  
2
```

2

这是特征值排成的列阵。如果键入输出格式,一次就可同时得出特征值和特征向量。

键入:

```
[x r]=eig(a3)
```

回车得出:

```
x =
    0.7071    0.2425    0.3015
         0         0    0.9045
    0.7071    0.9701    0.3015

r =
    4     0     0
     0     2     0
     0     0     2
```

也可以先把 a 变成符号矩阵,再用 `eig` 求解,这样可以得出整数特征值。

键入:

```
[x r]=eig(sym(a3))
```

回车得出:

```
x =
    [ 1, 1, 0 ]
    [ 0, 4, 4 ]
    [ 1, 0, 1 ]

r =
    [ 4, 0, 0 ]
    [ 0, 2, 0 ]
    [ 0, 0, 2 ]
```

6.3.2 矩阵的正交三角分解指令 `qr`

用指令 `qr` 可以实现矩阵 a 的正交三角分解,调用格式为:

```
[q r]=qr(a)
```

```
[q r p]=qr(a)
```

① 输入参数矩阵 a 不必是方阵。

② 输出参量用 `[q, r]` 格式(可以用其他字母)时 q 为正交方阵,阶数等于 a 的行数和列数中较小者,满足 $q^T q = E$; r 为与 a 同维的上三角阵,满足 $qr = a$ 。

③ 输出参量用 `[q r p]` 格式时 q 为正交方阵, r 为对角线元素绝对值递减的上三角阵, p 为换位阵,使之满足 $a \times p = q \times r$ 。

如果不写输出格式时,将输出一个变换过的矩阵,在此不予介绍。

例6-3 求矩阵 $A2 = \begin{bmatrix} 1 & 2 & 4 & 1 & 3 \\ 5 & 1 & 2 & 7 & 4 \\ 6 & 2 & 1 & 4 & 8 \end{bmatrix}$ 的正交三角分解。

解 在指令窗中键入：

```
A2=[1 2 4 1 3;5 1 2 7 4;6 2 1 4 8];
```

```
[q r]=qr(A2)
```

回车得出：

```
q =
```

```
0.1270    0.9501    0.2850
0.6350    0.2986    0.7125
0.7620    0.0905    0.6412
```

```
r =
```

```
7.8740    2.4130    2.5400    7.6200    9.0170
0         1.7825    3.2936    0.7782    2.3797
0         0         4.9237    2.7074    1.4249
```

为了验证 q 的正交性,可键入：

```
q*q'
```

回车得出：

```
ans =
```

```
1.0000    0.0000    0.0000
0.0000    1.0000    0
0.0000    0         1.0000
```

如果键入：

```
[q1,r1,p]=qr(A2)
```

回车得出：

```
q1 =
```

```
0.3180    0.2429    0.9165
0.4240    0.9010    0.0916
0.8480    0.3594    0.3895
```

```
r1 =
```

```
9.4340    6.6780    2.9680    2.7560    7.5260
0         4.6265    0.4712    0.3036    2.1056
0         0         3.4596    1.1456    0.9623
```

```
p =
```

```
0    0    0    0    1
0    0    0    1    0
0    0    1    0    0
0    1    0    0    0
1    0    0    0    0
```

若键入：

```
q1*r1
```

回车得出：

```
ans =
```

```
3.0000    1.0000    4.0000    2.0000    1.0000
4.0000    7.0000    2.0000    1.0000    5.0000
8.0000    4.0000    1.0000    2.0000    6.0000
```

再键入：

```
A2*p
```

回车得出：

```
ans =
```

```
3    1    4    2    1
4    7    2    1    5
8    4    1    2    6
```

可见 $A2 * p$ 与 $A2$ 只是元素位置不同而已, p 起到了对 $A2$ 元素位置的调换作用, 所以称换位矩阵。结果满足 $A2 * p = q1 * r1$ 。

矩阵的分解运算是矩阵分析中的重要内容之一, 除了介绍过的 `lu`、`chol` 和 `qr` 外, MATLAB 中还有许多其他矩阵分解的指令, 如对矩阵进行奇异值分解的指令 `svd`, 进行 Schur 分解的指令 `schur`, 进行 Hessenberg 分解的指令 `hess` 等, 需要时可用指令 `help` 查找“矩阵代数(matfun)”函数库的相关内容。

6.3.3 计算范数和矩阵谱半径的指令

1. 计算范数指令 `norm`

向量和矩阵的范数是衡量其分量或元素数量级大小的一种度量。在数值计算中, 为了对计算误差进行估计, 常常用到范数的概念及计算。MATLAB 中, 计算范数的专用指令是 `norm`, 调用格式是：

```
norm(A,ex)
```

输入参数 A 为向量时,

- ① ex 取 1 时, 返回向量 A 的 1-范数;
- ② ex 取 2 或被省略时, 返回向量 A 的 2-范数;
- ③ ex 取 `inf` 时, 返回向量 A 的无穷范数;
- ④ ex 取 `-inf` 时, 返回向量 A 的最小范数, 即各分量绝对值中的最小量。

输入参数 A 为矩阵时,

- ① ex 取 1 时, 返回矩阵 A 的列和范数;
- ② ex 取 2 或被省略时, 返回矩阵 A 的谱范数;
- ③ ex 取“fro”时, 返回 A 的 F-范数;
- ④ ex 取 `inf` 时, 返回矩阵 A 的行和范数。

2. 矩阵谱半径的计算

方阵 M 的谱半径 $\rho(M)$ 就是 M 最大特征值的绝对值, 可以用下述几个结合在一起的指令完成:

```
max(abs(eig(M)))
```

例 6-4 求矩阵 $s = \begin{bmatrix} 2 & 2 & 3 \\ 5 & 4 & -6 \\ 1 & 4 & 2 \\ 3 & 5 & 7 \end{bmatrix}$ 的列和范数、谱范数、F-范数及行和范数。

解 键入:

```
s=[2 2 3;5 4 -6;1 4 2;3 5 7];
```

```
s1=norm(s,1),s2=norm(s,2),s3=norm(s,'fro'),s4=norm(s,inf)
```

回车得出:

```
s1 =
```

```
18
```

```
s2 =
```

```
10.8040
```

```
s3 =
```

```
14.0712
```

```
s4 =
```

```
15
```

例 6-5 已知矩阵 $c = \begin{bmatrix} \frac{1}{2} & -\frac{2}{3} \\ \frac{1}{4} & \frac{1}{5} \end{bmatrix}$, 求它的行和范数及谱半径 $\rho(c)$ 。

解 在指令窗中键入:

```
c=[1/2 2/3;1/4 1/5] norm(c,inf) max(abs(eig(c)))
```

回车得出:

```
ans =
```

```
1.1667
```

```
ans =
```

```
0.5164
```

表明矩阵 c 的行和范数为 1.1667, 谱半径 $\rho(c) = 0.5164$ 。

练习题 6

1. 对下列方程组建立收敛的简单迭代公式:

$$\begin{bmatrix} 1 & 6 & -2 \\ 3 & -2 & 5 \\ 4 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

2. 求下列矩阵的行列式值、范数：

$$(1) \begin{bmatrix} 5 & 7 & 3 \\ 7 & 11 & 2 \\ 3 & 2 & 6 \end{bmatrix}$$

$$(2) \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

$$(3) \begin{bmatrix} 3 & 1 & -1 & 2 \\ -5 & 1 & 3 & -4 \\ 2 & 0 & 1 & -1 \\ 1 & -5 & 3 & -3 \end{bmatrix}$$

$$(4) \begin{bmatrix} 2 & 1 & 4 & 1 & 7 \\ 3 & -1 & 2 & 1 & 1 \\ 1 & 2 & 3 & 2 & 2 \\ 5 & 0 & 6 & 2 & 4 \\ 0 & 3 & 1 & 4 & 0 \end{bmatrix}$$

3. 对下列矩阵做正交三角分解：

$$(1) \begin{bmatrix} 2 & -1 & 0 & 0 \\ 1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

$$(2) \begin{bmatrix} 1 & 4 & -1 & 5 & 6 \\ 2 & 0 & 0 & 0 & -14 \\ -1 & 2 & -4 & 0 & 1 \\ 2 & 6 & -5 & 5 & 1 \end{bmatrix}$$

$$(3) \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \\ 2 & 3 & 3 \end{bmatrix}$$

$$(4) \begin{bmatrix} 1 & 3 & 2 & 1 & 4 \\ 2 & 6 & 1 & 0 & 7 \\ 3 & 9 & 3 & 1 & 11 \end{bmatrix}$$

4. 求下列方阵的特征值和特征向量：

$$(1) \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$(2) \begin{bmatrix} -1 & 1 & 0 \\ -4 & 3 & 0 \\ 1 & 0 & 2 \end{bmatrix}$$

$$(3) \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 3 & 3 & 6 \end{bmatrix}$$

$$(4) \begin{bmatrix} 1 & 1 & & & \\ 1 & 2 & 1 & & \\ & 1 & 3 & 1 & \\ & & 1 & 4 & 1 \\ & & & 1 & 5 \end{bmatrix}$$

第 7 章 插值法和数据拟合

自然现象和工程技术中,同时变化着的几个变量之间往往存在着某种联系,这些联系遵从着一定的规律。如果只考虑两个变量的情况,当变量 x 在其变化范围内取定一个数值时,相关变量 y 按一定的法则总有一个或几个定数和它对应,就叫 y 是 x 的函数,通常用 $y = F(x)$ 的形式来表示。广义地说 F 只表示 y 与 x 之间的一种对应关系,表达对应关系 F 的具体方法一般有下列 3 种。

- ① 解析法:用数学解析式表示 x 和 y 的对应关系,如 $y = f(x) = x^3 + \sin x$ 。
- ② 列表法:用一系列的 x 值和对应的 y 值列成表,如平方表、三角函数表等。
- ③ 图示法:在 X - Y 平面上将对应的 x 和 y 的数值画成一条曲线。

工程技术实践中,特别是在实验测试中,往往只能得到两个相关变量的一系列离散值,它们间的函数关系就只得用列表法或图示法表示。但是,有时因为某种原因希望把这种关系转换成解析表达式。另外,有些变量间的关系虽然可以用解析法表示,由于数学解析式过于繁杂不便使用,也希望用一个简单的解析表达式来近似地代替它,凡此种种,利用简单解析式 $\phi(x)$ 近似地代替列表法、图示法或复杂解析式表示的函数 $F(x)$ 一类问题,即寻找一个满足 $\phi(x) \approx F(x)$ 的简单函数 $\phi(x)$ 的问题,都可称之为函数逼近。本章介绍的插值法和数据拟合法就是函数逼近的两种重要方法。

7.1 多项式插值

设函数 $y = F(x)$ 是以表 7-1 给出的,即给出了一系列的 x 和与其对应的 y 值。

表 7-1 函数 $y = F(x)$ 的列表法表示

x	x_0	x_1	x_2	...	x_n
y	y_0	y_1	y_2	...	y_n

其中 $x \in [a, b]$ 。要构造一个简单的解析式 $\phi(x)$,使它满足下述的插值原则:

- ① $\phi(x) \approx F(x), x \in [a, b]$;
- ② $\phi(x_i) = y_i, i = 0, 1, 2, \dots, n$ 。

称 $\phi(x)$ 为 $F(x)$ 的插值函数,点 $x_0, x_1, x_2, \dots, x_n$ 为插值节点(也叫样本点)。由插值原则可知,插值函数在样本点上的取值必须等于已知函数在样本点上的对应值。

插值函数 $\phi(x)$ 可以选用不同的函数形式,如三角多项式、有理函数等。但最常选用的是代数多项式 $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$,因为多项式具有形式简单、计算方便、存在各阶导数等良好性质,下边介绍的就是选用 $\phi(x)$ 为 $p_n(x)$ 的代数多项式插值法。

7.1.1 代数多项式插值的基本原理

1. 构造 n 次插值多项式

假设对于函数 $y = F(x)$ 给出了一组函数值 $y_i = F(x_i) (i = 0, 1, 2, 3, \dots, n)$ 的列表法表示, 如表 7-1 所示。表中 $x_0, x_1, x_2, \dots, x_n \in [a, b]$, 它们是互异的 $(n+1)$ 个插值节点, $y_0, y_1, y_2, \dots, y_n$ 是分别与这些节点对应的函数值。构造一个不超过 n 次的多项式:

$$p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n = \sum_{i=0}^n a_ix^i \quad (7-1)$$

如果它满足插值原则②, 即 $p_n(x_i) = y_i, (i = 0, 1, 2, \dots, n)$ 就被称为 $F(x)$ 的 n 次插值多项式。

2. 确定 n 次多项式系数

求这个多项式就是求出它的 $(n+1)$ 个系数 $a_i (i = 0, 1, 2, \dots, n)$ 。

依次取 $x = x_0, x_1, x_2, \dots, x_n$ 相应地取 $y_0, y_1, y_2, \dots, y_n$, 并使其满足插值原则②: $p_n(x_i) = y_i, (i = 0, 1, 2, \dots, n)$ 则可得出系数 a_i 满足的 $(n+1)$ 个方程构成的方程组:

$$\left. \begin{aligned} a_0 + a_1x_0 + a_2x_0^2 + \dots + a_nx_0^n &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \dots + a_nx_1^n &= y_1 \\ &\dots\dots\dots \\ a_0 + a_1x_n + a_2x_n^2 + \dots + a_nx_n^n &= y_n \end{aligned} \right\} \quad (7-2)$$

式(7-2)也可写成下述形式:

$$p_n(x_j) = \sum_{i=0}^n a_ix_j^i = y_j, (j = 0, 1, 2, \dots, n)$$

这个方程组的系数行列式是范德蒙(Vandermonde)行列式:

$$V(x_0, x_1, x_2, \dots, x_n) = \begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{vmatrix} = \prod_{0 \leq i < j \leq n} (x_i - x_j) \quad (7-3)$$

由范德蒙行列式性质可知, 只要 x_i 互异, 则 $V(x_0, x_1, x_2, \dots, x_n) \neq 0$ 。于是根据克莱姆法则可以知道, 方程组(7-2)存在唯一解。也就是说, 只要 $(n+1)$ 个样本点的值 $x_0, x_1, x_2, \dots, x_n$ 互不相等, 原则上就能由式(7-2)求出 $(n+1)$ 个系数 a_i , 从而就可以得到 n 次插值多项式 $p_n(x)$ 。然而, 当 n 很大时, 通过解方程组(7-2)求系数 a_i 的工作是很烦琐的, 因此, 找出许多求解方程组(7-2)的简捷方法, 下面介绍两种。

7.1.2 两种常见插值法

求解方程组找出插值多项式系数的方法很多, 以拉格朗日(Lagrange)法和牛顿(Newton)法最为常见, 原理简单应用广泛。下面从线性插值、二次插值到 n 次插值, 分别对它们予以

介绍。

1. 线性插值

如果已知函数 $y = F(x)$ 列表法表示(表 7-2), 这时 $n=1$, 用代数插值多项式可写成 $p_1(x) = a_0 + a_1x$, 它满足插值原则②时可有:

表 7-2 函数 $y = F(x)$

x	x_0	x_1
y	y_0	y_1

$$\begin{cases} a_0 + a_1x_0 = y_0 \\ a_0 + a_1x_1 = y_1 \end{cases}$$

解这个方程组可得出系数:

$$a_1 = \frac{y_0 - y_1}{x_0 - x_1} \left(\text{或} \frac{y_1 - y_0}{x_1 - x_0} \right) \quad a_0 = y_0 - a_1x_0 = y_0 - \frac{y_0 - y_1}{x_0 - x_1}x_0$$

对 a_1 和 a_0 进行不同的变换, 可得出下面两种形式的插值函数。

1) Lagrange 线性插值

把 a_0, a_1 代入插值多项式 $p_1(x) = a_0 + a_1x$ 中, 变换成直线的两点式表达式:

$$p_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} = y_0 L_0(x) + y_1 L_1(x) \quad (7-4)$$

其中 $L_0(x) = \frac{x - x_1}{x_0 - x_1}$ 和 $L_1(x) = \frac{x - x_0}{x_1 - x_0}$ 分别称为节点 x_0 和 x_1 的一次插值基函数。插值函数 $p_1(x)$ 是这两个基函数的线性组合, 组合系数就是对应节点上的函数值。这种形式的插值函数叫作 Lagrange 插值多项式。

2) Newton 线性插值

把 a_0, a_1 代入插值多项式 $p_1(x) = a_0 + a_1x$, 变换成直线的点斜式表达式:

$$p_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) = y_0 + (x - x_0)F(x_1, x_0) \quad (7-5)$$

式中的 $F(x_1, x_0) = \frac{y_1 - y_0}{x_1 - x_0}$ 是函数 $y = F(x)$ 在 x_1, x_0 处的一阶均差, 等于自变量增加一个单位时的函数增量。以均差表示的插值多项式函数(7-5)称为 Newton 插值多项式。

2. 二次插值

如果已知函数 $y = F(x)$ 的列表法表示为表 7-3, 则插值多项式最高次数 $n=2$, 代数插值多项式为 $p_2(x) = a_0 + a_1x + a_2x^2$ 。据插值原则②可知, 系数 a_i 应满足方程组(7-2), 但这里 $n=2$, 只有三个方程。当各节点的值 x_0, x_1, x_2 互不相等时, 范德蒙(Vandermonde)行列式 $V(x_0, x_1, x_2) \neq 0$, 方程组存在唯一解。仿照线性插值的推导方法, 有下述两种构成二次插值多项式的方法。

表 7-3 函数 $y = F(x)$

x	x_0	x_1	x_2
y	y_0	y_1	y_2

1) Lagrange 二次插值

可以像推导 Lagrange 一次插值那样, 得出二次插值多项式:

$$p_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) \quad (7-6)$$

$p_2(x)$ 由二次插值的基函数 $L_i(x)$ ($i=0, 1, 2$) 线性组合而成, 基函数应该满足:

$$L_i(x) = \begin{cases} 1, & (x = x_i) \\ 0, & (x \text{ 在节点上, 但 } x \neq x_i) \end{cases}$$

各节点的基函数可据上式算出, 如取 $i=0$, $L_0(x)$ 在 $x=x_1$ 和 $x=x_2$ 处为零, 因此可知 $L_0(x)$ 含有 $(x-x_1)(x-x_2)$ 。又因为 $L_0(x)$ 是一个不多于 2 次的多项式, 不妨设 $L_0(x) = A(x-x_1)(x-x_2)$, 代入 $x=x_0$ 时满足的条件 $L_0(x_0) = 1$, 可得 $A = 1/[(x_0-x_1)(x_0-x_2)]$, 于是可以得出 $L_0(x)$ 的表达式。依此方法也可得出 $L_1(x)$ 和 $L_2(x)$, 把它们一并列在式(7-7)中:

$$\left. \begin{aligned} L_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \\ L_1(x) &= \frac{(x-x_2)(x-x_0)}{(x_1-x_2)(x_1-x_0)} \\ L_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \end{aligned} \right\} \quad (7-7)$$

于是, 把式(7-7)代入式(7-6)就构成了 Lagrange 二次插值多项式函数 $p_2(x)$ 。

2) Newton 二次插值

仿照构成 Newton 一次插值多项式的方法, 设二次插值多项式的形式为:

$$p_2(x) = A + B(x-x_0) + C(x-x_0)(x-x_1),$$

满足插值原则②时应有 $p_2(x_0) = y_0$, 由此可得 $A = p_2(x_0) = y_0$ 。

再利用 $p_2(x_1) = y_1$ 得:

$$B = \frac{y_1 - y_0}{x_1 - x_0} = F(x_1, x_0)$$

称 $F(x_1, x_0)$ 为一阶均差。

利用 $p_2(x_2) = y_2$ 得:

$$C = \frac{y_2 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{(y_1 - y_0)}{(x_1 - x_0)(x_2 - x_1)} = \frac{F(x_2, x_0) - F(x_1, x_0)}{(x_2 - x_1)}$$

这个结果是一阶均差的均差, 称为二阶均差。二阶均差的一般形式写成:

$$F(x_i, x_j, x_k) = \frac{F(x_i, x_j) - F(x_j, x_k)}{(x_i - x_k)} \quad (7-8)$$

于是得到 Newton 二次插值多项式:

$$p_2(x) = y_0 + (x-x_0)F(x_0, x_1) + (x-x_0)(x-x_1)F(x_0, x_1, x_2) \quad (7-9)$$

3. n次插值

1) Lagrange 式的 n 次插值

若知道函数 $y = F(x)$ 在 $(n+1)$ 个节点上的值: (x_0, y_0) , (x_1, y_1) , (x_2, y_2) , ..., (x_n, y_n) 利用这些数据可以构造出代数插值多项式(7-1)。为此, 仿照构造二次 Lagrange 插值多项式的

方法 求出相应的 n 次插值多项式函数：

$$p_n(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + \dots + y_n L_n(x) = \sum_{i=0}^n y_i L_i(x) \quad (7-10)$$

按照式(7-7) 在节点 x_i 上的 Lagrange 插值基函数可写成：

$$L_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \quad (7-11)$$

若令：

$$\omega(x) = (x - x_0)(x - x_1) \dots (x - x_n), \quad \bar{\omega}(x_i) = (x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)$$

则式(7-11)中的 $L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} = \frac{\omega(x)}{(x - x_i)\bar{\omega}(x_i)}$, 于是 n 次 Lagrange 插值多项

式(7-10)可以写成：

$$p_n(x) = \sum_{i=0}^n \frac{\omega(x)}{(x - x_i)\bar{\omega}(x_i)} y_i \quad (7-12)$$

2) Newton 式的 n 次插值法

仿照前边求出 Newton 二次插值多项式的办法 , 首先构造出 n 阶均差的递推公式：

$$F(x_0, x_1, x_2, \dots, x_n) = \frac{F(x_0, x_1, \dots, x_{n-1}) - F(x_1, x_2, \dots, x_n)}{x_0 - x_n} \quad (7-13)$$

由式(7-13)可以得出 $F(x)$ 在 x_i, x_j 处的一阶均差为：

$$F(x_i, x_j) = \frac{y_i - y_j}{x_i - x_j}$$

$F(x)$ 在 x_i, x_j, x_k 处的二阶均差为：

$$F(x_i, x_j, x_k) = \frac{F(x_i, x_j) - F(x_j, x_k)}{x_i - x_k}$$

$F(x)$ 在 x_i, x_j, x_k, x_h 处的三阶均差为：

$$F(x_i, x_j, x_k, x_h) = \frac{F(x_i, x_j, x_k) - F(x_j, x_k, x_h)}{x_i - x_h}$$

以此类推 , 可以求出 F 的各阶均差 , 从而得到 n 次牛顿插值多项式：

$$p_n(x) = y_0 + (x - x_0)F(x_0, x_1) + (x - x_0)(x - x_1)F(x_0, x_1, x_2) + \dots + (x - x_0) \dots (x - x_{n-1})F(x_0, x_1, \dots, x_n) \quad (7-14)$$

7.1.3 插值多项式的误差估计

用 n 次插值多项式 $p_n(x)$ 近似替代函数 $F(x)$ 时 , 设产生的截断误差为 $R_n(x)$ 则：

$$R_n(x) = F(x) - p_n(x) \quad (7-15)$$

由 $p_n(x) = F(x) + R_n(x)$ 可知 $R_n(x)$ 为 n 次插值多项式 $p_n(x)$ 的余项。为了估计插值多项式

的误差,下面先介绍余项定理。

1. 余项定理

当函数 $F(x)$ 足够光滑,即满足以下条件。

① $F(x)$ 在区间 $[a, b]$ 上连续。

② $F(x)$ 具有直至 $(n+1)$ 阶导数,则对一切 $x \in [a, b]$,总存在相应的点 $\xi \in [a, b]$,使得:

$$R_n(x) = F(x) - p_n(x) = \frac{F^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x) \quad (7-16)$$

其中, $\omega_{n+1}(x) = (x - x_0)(x - x_1)\dots(x - x_n) = \prod_{i=0}^n (x - x_i)$

证明:当 $x = x_i (i=0, 1, 2, \dots, n)$ 为插值节点时 $\omega_{n+1}(x_i) = 0$, $R_n(x_i) = 0$, 定理成立。

当 $x \in [a, b]$ 而不是插值基点时,作辅助函数:

$$G(t) = F(t) - p_n(t) - \frac{R_n(x)}{\omega_{n+1}(x)} \omega_{n+1}(t)$$

在 $[a, b]$ 间任选一个非插值基点 x , 容易验证 $G(x) = 0$; 在插值基点上,当 $t = x_i$ 时, $G(x_i) = 0$ 。这样,在 $[a, b]$ 间至少有 $(n+2)$ 个互异的点使 $G(t) = 0$ 。

据罗尔定理,函数的两个零点之间至少有一点使 $G(t)$ 的一阶导数 $G'(t) = 0$ 。于是,在 (a, b) 内至少有 $(n+1)$ 个互异点使 $G'(t) = 0$ 。再对函数 $G(t)$ 应用罗尔定理,以此类推,可知在 (a, b) 内至少有一个点 $t = \xi$ 使 $G^{(n+1)}(\xi) = 0$, 即:

$$G^{(n+1)}(\xi) = F^{(n+1)}(\xi) - \frac{R_n(x)}{\omega_{n+1}(x)} (n+1)! = 0$$

移项整理便得出式(7-16),定理得证。

2. 误差估计

在式(7-16)中,虽然 $F^{(n+1)}(\xi)$ 存在,但 ξ 的值是难以确定的。如果能估算出在区间 $[a, b]$ 上 $|F^{(n+1)}(\xi)|$ 的上界 M , 即 $|F^{(n+1)}(\xi)| \leq M$, 则可以得出截断误差的范围:

$$|R_n(x)| \leq \frac{M}{(n+1)!} |\omega_{n+1}(x)| \quad (7-17)$$

根据式(7-17),只要找出正数 M , 就可以估算出用 $p_n(x)$ 替代函数 $F(x)$ 时产生的截断误差 $|R_n(x)|$, 实用中往往是根据函数 $F(x)$ 的性质估算 M 大小的。

7.2 分段三次插值和三次样条插值

仅从截断误差公式 $R_n(x)$ 来看,用插值多项式 $p_n(x)$ 近似替代函数 $F(x)$ 时,似乎 $[a, b]$ 之间的分点数 n 越多,插值多项式的次数越高,产生的截断误差就越小,实际上并非如此。Runge 和 Faber 的工作证明,高次插值多项式并不一定都能收敛到被插值的函数上,而且还增

加了许多工作量。例如,将函数 $F(x) = \frac{1}{1+30x^2}$ 用 4 次和 10 次插值多项式函数 $P_4(x)$ 和 $P_{10}(x)$ 替代时,高次插值多项式并不理想,这可从图 7-1 看出。图中实线是函数 $F(x)$ 的曲线,两条虚线分别是用插值多项式函数 $P_4(x)$ 和 $P_{10}(x)$ 画的,虽然多项式插值函数都通过了样本点,但是在 $x \approx 0.9$ 时 $P_{10}(x)$ 比 $P_4(x)$ 离开函数 $F(x)$ 曲线更远,逼近效果极差。

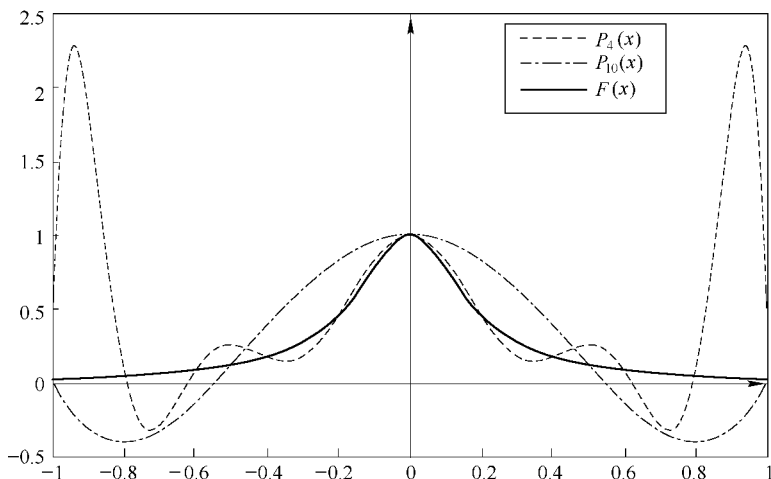


图 7-1 函数 $F(x)$ 及其插值多项式 $P_4(x)$ 和 $P_{10}(x)$ 的曲线比较

7.2.1 分段三次 Hermite 插值

为了利用多项式插值方法而又克服高次插值多项式的缺陷,便引入了分段插值的概念。它的基本思想是把函数在整个区间上分成许多段,每段都选用适当的低次插值多项式代替函数 $F(x)$,整体上按一定的光滑性要求连接起来,构成一个分段的插值函数。

为此,把函数 $F(x)$ 的自变量 x 在区间 $[a, b]$ 上用 $(n-1)$ 个节点分割成 n 段:

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

根据这些节点的取值 x_i , $F(x)$ 在节点上的函数值 $F(x_i) = y_i$ 和导数值 $F'(x_i) = m_i$ ($i = 0, 1, 2, \dots, n$),可以构造一个分段三次插值函数 $H(x)$,它满足下述条件。

① $H(x_i) = y_i$, $H'(x_i) = m_i$ ($i = 0, 1, 2, \dots, n$)。

② 在每个小区间 $[x_i, x_{i+1}]$ ($i = 0, 1, 2, \dots, n-1$) 上 $H(x)$ 都是一个三次多项式:

$$H_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + a_{i3}x^3$$

把这样构成的分段三次函数 $H(x)$ 称为分段三次 Hermite 插值函数,它的各小段均为三次多项式,而整体上具有一阶连续导数。

7.2.2 三次样条插值的基本原理

三次样条插值也是一种分段插值方法,用分段的三次多项式构造一个整体上具有函数、一阶和二阶导函数连续的函数,近似地替代已知函数 $F(x)$ 。“样条”一词源于过去绘图员

使用的一种绘图工具——样条,它是用富于弹性、能弯曲的木条(或塑料)制成的软尺,把它弯折靠近所有的基点用画笔沿样条就可以画出连接基点的光滑曲线。

假设已知函数 $F(x)$ 在区间 $[a, b]$ 上的 $(n+1)$ 个节点 $a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$ 及其对应的函数值 $F(x_i) = y_i, (i=0, 1, 2, \dots, n)$, 即给出 $(n+1)$ 组样本点数据 $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, 可以构造一个定义在 $[a, b]$ 上的函数 $S(x)$, 满足下述条件。

① $S(x_i) = y_i, (i=0, 1, 2, \dots, n)$, 即满足插值原则②。

② $S(x)$ 在每个小区间 $[x_i, x_{i+1}] (i=0, 1, 2, \dots, n-1)$ 上都是一个三次多项式:

$$S_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + a_{i3}x^3 \quad (7-18)$$

③ $S(x), S'(x)$ 和 $S''(x)$ 在 $[a, b]$ 上连续。

可见 $S(x)$ 是一个光滑的分段函数, 这样的函数 $S(x)$ 称为三次样条(Spline)插值函数。

构造的函数 $S(x)$ 是由 n 个小区间上的分段函数组成, 根据条件②, 每个小区间上构造出一个三次多项式, 第 i 个小区间上的三次多项式为 $S_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + a_{i3}x^3$, 共有 n 个多项式, 每个多项式有 4 个待定系数。要确定这 n 个多项式, 就需要确定 $4n$ 个系数 $a_{i0}, a_{i1}, a_{i2}, a_{i3} (i=0, 1, 2, \dots, n-1)$ 。为此, 应该找到包含这些系数的 $4n$ 个独立方程。

根据 $S(x)$ 满足的条件①, 在所有节点上可得出 $(n+1)$ 个条件方程:

$$S(x_i) = y_i, (i=0, 1, 2, \dots, n) \quad (7-19)$$

根据 $S(x)$ 满足的条件③, 除两端点外在所有节点上, 又可得出 $3(n-1)$ 个方程:

$$\left. \begin{aligned} S_i(x_i) &= S_{i+1}(x_i) \\ S'_i(x_i) &= S'_{i+1}(x_i) \\ S''_i(x_i) &= S''_{i+1}(x_i) \end{aligned} \right\} (i=1, 2, \dots, n-1) \quad (7-20)$$

由式(7-19)和式(7-20)可知共有 $(4n-2)$ 个独立方程, 还差两个。通常的办法是在区间 $[a, b]$ 的两个端点上各加一个条件, 即称之为边界条件。常用的三种边界条件是:

① 已知 $S''(x_0)$ 和 $S''(x_n)$, 特别是当取 $S''(x_0) = S''(x_n) = 0$ 时, 称为自然边界条件;

② 已知 $S'(x_0)$ 和 $S'(x_n)$, 即已知两端点处切线的斜率;

③ 已知 $2S''(x_0) = S''(x_1)$ 和 $2S''(x_n) = S''(x_{n-1})$ 。

这样, 在已有的 $(4n-2)$ 个条件方程基础上, 再加上任何一种边界条件, 即可求出这 $4n$ 个系数, 从而就可以求得三次样条插值函数 $S(x)$ 。

7.2.3 三次样条插值函数的一种具体推导方法

作为举例, 这里介绍一种推导三次样条函数的具体方法。设所求三次样条函数 $S(x)$ 在 n 个子区间中的任一子区间 $[x_i, x_{i+1}] (i=0, 1, 2, \dots, n-1)$ 上的三次多项式为 $S_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + a_{i3}x^3$, 则可得出:

$$S'_i(x) = a_{i1} + 2a_{i2}x + 3a_{i3}x^2$$

$$S''_i(x) = 2a_{i2} + 6a_{i3}x$$

最后一个线性方程, 设 $S'_i(x_i) = M_i$, $S'_{i+1}(x_{i+1}) = M_{i+1}$, 则 $S'_i(x)$ 可以用两点式方程表示:

$$S'_i(x) = M_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + M_{i+1} \frac{x - x_i}{x_{i+1} - x_i} = M_i \frac{x_{i+1} - x}{h_i} + M_{i+1} \frac{x - x_i}{h_i}, x \in [x_i, x_{i+1}] \quad (7-21)$$

式中 $h_i = x_{i+1} - x_i$ 。

对式(7-21)两边进行一次积分, 得出:

$$S'_i(x) = M_{i+1} \frac{(x - x_i)^2}{2h_i} - M_i \frac{(x_{i+1} - x)^2}{2h_i} + A_i, \quad x \in [x_i, x_{i+1}] \quad (7-22)$$

再对式(7-22)两边进行一次积分, 得出:

$$S_i(x) = M_{i+1} \frac{(x - x_i)^3}{6h_i} + M_i \frac{(x_{i+1} - x)^3}{6h_i} + A_i(x - x_i) + B_i, \quad x \in [x_i, x_{i+1}] \quad (7-23)$$

式中 A_i, B_i 都是积分常数。式(7-23)中的 $S_i(x)$ 应该满足插值原则②, 即满足式(7-19):

$$S_i(x_i) = M_i \frac{h_i^2}{6} + B_i = y_i, \quad S_i(x_{i+1}) = M_{i+1} \frac{h_i^2}{6} + A_i h_i + B_i = y_{i+1}$$

x_i, y_i 是已知的, 解这两个方程, 得出 A_i (也可先求出 B_i), 代入式(7-22)得出:

$$S'_i(x) = M_{i+1} \frac{(x - x_i)^2}{2h_i} - M_i \frac{(x_{i+1} - x)^2}{2h_i} + \frac{y_{i+1} - y_i}{h_i} - \frac{h_i(M_{i+1} - M_i)}{6}, \quad x \in [x_i, x_{i+1}] \quad (7-24)$$

$S'_i(x)$ 在 $[a, b]$ 上连续, 所以在相邻两个小区间的分界点 x_i (节点) 上取值相等:

$$S'_{i-1}(x_i - 0) = S'_i(x_i + 0), \quad i = 1, 2, \dots, n-1 \quad (7-25)$$

由式(7-24)和式(7-25)便可得出:

$$S'_{i-1}(x_i - 0) = M_i \frac{h_{i-1}}{3} + \frac{y_i - y_{i-1}}{h_{i-1}} + \frac{h_{i-1}}{6} M_{i-1} = S'_i(x_i + 0) = -M_i \frac{h_i}{3} + \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6} M_{i+1}$$

注: 点 $(x_i - 0) \in [x_{i-1}, x_i]$, 点 $(x_i + 0) \in [x_i, x_{i+1}]$, 因此, 在代入式(7-24)时, 区间长度分别用 $h_{i-1} = x_i - x_{i-1}$ 和 $h_i = x_{i+1} - x_i$ 。

移项整理可得:

$$\frac{h_{i-1}}{h_i + h_{i-1}} M_{i-1} + 2M_i + \frac{h_i}{h_i + h_{i-1}} M_{i+1} = \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right) \frac{6}{h_i + h_{i-1}}$$

若令 $b_i = \frac{h_i}{h_i + h_{i-1}}, c_i = \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right) \frac{6}{h_i + h_{i-1}}$, 它们均为常数, 于是上式变成:

$$(1 - b_i) M_{i-1} + 2M_i + b_i M_{i+1} = c_i, \quad (i = 1, 2, \dots, n-1) \quad (7-26)$$

这 $(n-1)$ 个方程中共有 $(n+1)$ 个未知量: M_0, M_1, \dots, M_n , 要求出它们尚缺两个条件方程, 这就需要借助边界条件。如果已知 M_0 和 M_n , 则可由式(7-26)得出所有 $M_i (i = 0, 1, \dots, n)$, 把它们和 A_i, B_i 一并代入式(7-23), 就可求出各子区间上的分段函数 $S_i(x)$ 。

为了再找出两个条件方程, 常用一种由内部外推的方法, 即在区间 $[a, b]$ 起点上用 $S''(x_1)$ 和 $S''(x_2)$ 表示 $S''(x_0) = S''(a)$, 终点上用 $S''(x_{n-2})$ 和 $S''(x_{n-1})$ 表示 $S''(x_n) = S''(b)$ 。

假设 $\frac{S''(x_0) - S''(x_1)}{S''(x_1) - S''(x_2)} = \frac{x_0 - x_1}{x_1 - x_2}$, 则可得出:

$$S''(x_0) = \left(1 + \frac{x_0 - x_1}{x_1 - x_2} \right) S''(x_1) - \frac{x_0 - x_1}{x_1 - x_2} S''(x_2)$$

又假设 $\frac{S''(x_n) - S''(x_{n-1})}{S''(x_{n-1}) - S''(x_{n-2})} = \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}$, 可得到:

$$S''(x_n) = \left(1 + \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}} \right) S''(x_{n-1}) - \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}} S''(x_{n-2})$$

于是,式(7-19)和式(7-20)再加上这两个方程,共有 $4n$ 个条件方程,完全可以确定 $4n$ 个系数,从而可求出三次插值样条函数 $S(x)$ 。这两个外推条件相当于让两端点处的二阶导数是线性的,即让曲线顺势外延。

7.3 插值法在 MATLAB 中的实现

从已知的一些离散数据点及其函数值,即函数的列表法表示,推求出未知点上函数值的所谓插值方法,在科技工作中应用十分广泛,如查对数表、三角函数表中都会遇到这类插值问题。MATLAB 中设有许多插值指令,这里仅介绍最常用的一元函数插值指令,它可以使前面讲过的理论得以用计算机实现。

7.3.1 一元函数的插值(查表)指令 interp1

该指令的调用格式为:

$$y_k = \text{interp1}(x, y, x_k, 'method')$$

- ① 输入参数 x 和 y 为已知的两个同维向量 $x = \{x_i\}_{1 \times n}$ 和 $y = \{y_i\}_{1 \times n}$, 满足函数 $y_i = F(x_i)$ 关系,它们是进行“造表”的根据,把 x_i, y_i 称为样本点(插值节点)。
- ② 输出量 y_k 是与 x_k 对应的函数值。插值点 $x_k \in [x_1, x_n]$, 可以是数值、向量或矩阵, y_k 与 x_k 维数相同,其元素一一对应。
- ③ 用单引号界定的 $method$ 有 4 种参数可供选择:
 - nearest 最近插值——用直角折线连接各样本点。
 - linear 线性插值——用直线依次连接各样本点,形成折线。省略 'method' 时,即默认为此项。
 - pchip (或 cubic) 分段三次插值——用分段三次多项式 Hermite 插值(Piecewise Cubic Hermite Interpolating Polynomial)曲线,依次连接相邻样本点,整体上具有函数及其一阶导数连续性。
 - spline 三次样条插值——用分段三次多项式曲线光滑地连接相邻样本点,整体上具有函数、一阶和二阶导数连续性。插值点 x_k 可以在区间 $[x_1, x_n]$ 外的附近取值,可以是数值、向量或矩阵, y_k 与 x_k 同维。

这个指令并不输出插值多项式函数,只输出插值点上的函数值。这就相当于根据数据对 (x, y) “造表”,然后查出对应于 x_k 的函数值 y_k , 所以又称为查表指令。

7.3.2 三次插值和三次样条插值指令

对于三次插值和三次样条插值, MATLAB 中都设有专用指令。

三次插值指令调用格式为:

$$y_k = \text{pchip}(x, y, x_k)$$

三次样条插值指令调用格式为:

$$y_k = \text{spline}(x, y, x_k)$$

参数 x, y, x_k, y_k 的意义及要求与 `interp1` 中的完全一样。插值效果与 `interp1` 中参数 "method" 分别选用 'pchip' 和 'spline' 等价。它允许 x_k 在区间 $[\min(x), \max(x)]$ 外的附近取值, 因为程序设计中在两个端点处, 都使用了由内部外推的方法。

例 7-1 已知 $y=F(x)$ 的函数关系中, 当 $y = [-0.5 \ 0.3 \ 0 \ 0.8 \ 0.1 \ 0.5 \ -0.3 \ 0.2 \ 0.3 \ -0.7 \ 0]$ 时, 对应的有 $x = 0:\text{length}(y)-1$ 。用“最近插值”和“线性插值”法, 分别求出当 $x_1 = 0:0.1:\text{length}(y)-1$ 时所对应的函数值, 并画出曲线。

解 题设的 x 和 y 是样本点, 所以键入:

```
y=[-0.5 0.3 0 0.8 0.1 0.5 -0.3 0.2 0.3 -0.7 0]; x=0:length(y)-1;
```

% 输入样本点

```
x1=0:0.1:length(y)-1;
```

```
y1=interp1(x,y,x1,'nearest'); y2=interp1(x,y,x1,'linear'); % 求出 x1 的函数值
```

```
plot(x,y,'* ',x1,y1,': ',x1,y2,'r '), legend('样本点','最近插值','线性插值')
```

回车得出图 7-2。

如果去掉“求出 x_1 的函数值”前面一程序末尾的分号, 就会输出相应的函数值。

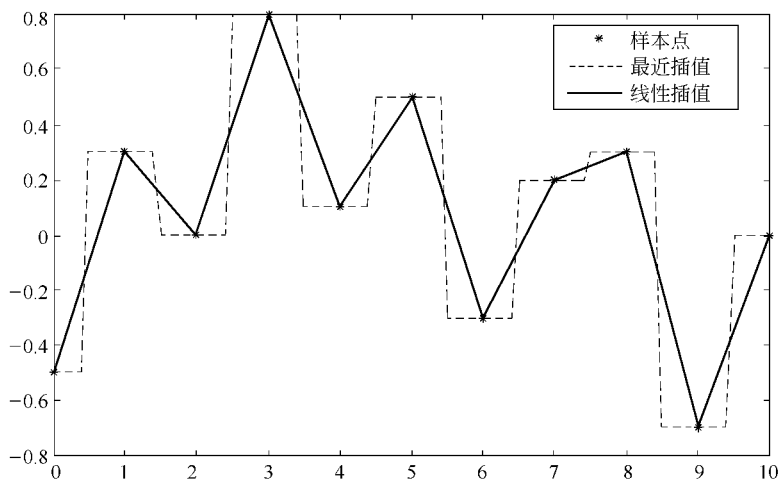


图 7-2 最近插值和线性插值比较图

直接使用“pchip”和“spline”两个绘图指令, 跟用 `interp1` 指令时 method 分别选用 pchip 和 spline 是一样的。下面看一个用三次插值和三次样条作图的例子。

例 7-2 已知 $y=F(x)$ 函数关系: 当 $y = [1 \ 3 \ 0 \ 20 \ 20 \ 4 \ 18]$ 时, 对应的 $x = 0:\text{length}(y)-1$ 。

用“三次插值”和“三次样条插值”法, 分别求出当 $x_1 = -0.5:0.2:5.5$ 时所对应的函数值并画出曲线。

解 题中的向量 x 和 y 是样本点数据, 所以键入:

```

y=[1 3 0 20 20 4 18]; x=0:length(y)-1; x1=0.5:0.2:5.5;
y1=interp1(x,y,x1,'pchip');
y2=spline(x,y,x1); plot(x,y,'*',x1,y1,'r',x1,y2,'-.'),grid...
legend('样本点','三次插值','三次样条插值')

```

回车得出图 7-3。

若去掉 y_1 和 y_2 两句末尾的分号, 就会输出相应的函数值。

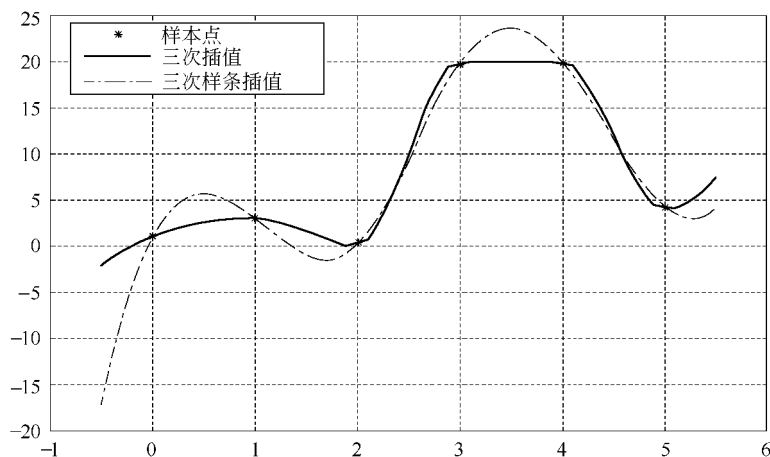


图 7-3 三次插值和三次样条插值比较图

由图 7-3 可见, 用两种方法画出的曲线在样本点之间取值是有差异的, 用三次样条插值时曲线的光滑程度要好一些。另外, 当自变量在限定区间外边界处的曲线上时, 三次样条插值和三次插值法的外延趋势不同, 差异越发明显。

7.4 数据的曲线拟合

科学实验或数据统计中, 在得不到两组相关数据之间的精确解析表达式时, 就希望找到关系 $y=F(x)$ 的近似解析式 $y=F(x) \approx \phi(x)$, $\phi(x)$ 越简单越好。前面讲过寻求简单函数 $\phi(x)$ 的插值法, 要求在样本点 x_i, y_i 上满足插值条件 $y_i = \phi(x_i) = F(x_i)$, 也就是说, 插值函数 $\phi(x)$ 必须通过所有样本点。然而, 许多样本点的取得本身就包含着实验中的测量误差, 这一要求无疑是保留了这些测量误差的影响, 满足这一要求虽然使样本点处“误差”为零, 但会使非样本点处的误差变得过大, 很不合理。为此, 提出了另一种函数逼近方法——数据拟合法, 它不要求构造的近似函数 $\phi(x)$ 全部通过样本点, 而是“很好逼近”它们。这种近似函数 $\phi(x)$ 反映了已知数据组间存在着某种关系的一般趋势, 是“拟合”这些数据得出的函数曲线, 不同于“插值”得出的曲线。寻找“很好逼近”的函数有多种方法, 这里只介绍常用的多项式最小二乘法。

7.4.1 数据曲线拟合的最小二乘法

已知两组相关数据 $x=x_1, x_2, \dots, x_n$ 和 $y=y_1, y_2, \dots, y_n$ 之间存在某种函数关系 $y=F(x)$, 如

果用一个解析式 $\phi(x)$ 近似代替 $F(x)$ 则这个 $\phi(x)$ 取成 m 次代数多项式 $P_m(x)$ 不失为一种最佳选择, 因为任何连续函数, 至少在一个较小的邻域内都可以用代数多项式任意逼近。若取:

$$y = F(x) \approx \phi(x) = P_m(x) = a_0 + a_1x + a_2x^2 + \dots + a_mx^m = \sum_{i=0}^m a_ix^i \quad (7-27)$$

通过已知满足函数 $y = F(x)$ 的数据组 $y_j, x_j (j=1, 2, \dots, n)$, 可以求出逼近函数 $P_m(x)$ 的所谓函数。逼近法有许多种途径, 下面介绍数据拟合中的最小二乘法。

最小二乘法拟合是要求在所有 n 个样本点 $x_j (j=1, 2, \dots, n)$ 处, 多项式取值与函数值 y_j 偏差的平方 $r_j^2 = (P_m(x_j) - y_j)^2$ 之和达到最小, 也就是使 $P_m(x_j)$ “很好地逼近” 函数值 $y_j = F(x_j)$, 为此, 要求下面表达式中的 R 达到最小 (在插值法中要求 $r_j = 0$, 即 $R = 0$, 且一般样本点数 $n = m + 1$):

$$R = \sum_{j=1}^n r_j^2 = \sum_{j=1}^n (P_m(x_j) - y_j)^2 = \sum_{j=1}^n (a_0 + a_1x_j + a_2x_j^2 + \dots + a_mx_j^m - y_j)^2, (m < n) \quad (7-28)$$

这种要求更符合实际需要, 因为“偏差的平方和”尽可能小就保证了偏差绝对值尽可能小, 这正是对实测数据的希望, 反映了给定数据间依存关系的一般趋势。

由式(7-28)可知, R 是待求变量 $a_0, a_1, a_2, \dots, a_m$ 的函数 $R = R(a_0, a_1, a_2, \dots, a_m)$ 。使 R 尽可能地小, 就归结为求多元函数 R 的极小值。这可以用数学分析中求极值的方法, 即让 R 对 $a_i (i=0, 1, 2, \dots, m)$ 的偏导数都等于零, 从而求出 $P_m(x)$ 各项系数 $a_0, a_1, a_2, \dots, a_m$ 满足的方程为:

$$\frac{\partial R}{\partial a_i} = 2 \sum_{j=1}^n (a_0 + a_1x_j + a_2x_j^2 + \dots + a_mx_j^m - y_j)x_j^i = 0 \quad (i=0, 1, 2, \dots, m)$$

移项可得:

$$\sum_{j=1}^n (a_0 + a_1x_j + a_2x_j^2 + \dots + a_mx_j^m)x_j^i = \sum_{j=1}^n y_jx_j^i \quad (i=0, 1, 2, \dots, m < n) \quad (7-29)$$

这是由 m 个方程构成的方程组, 即下述的矩阵方程:

$$\begin{bmatrix} n & \sum x_j & \sum x_j^2 & \dots & \sum x_j^m \\ \sum x_j & \sum x_j^2 & \sum x_j^3 & \dots & \sum x_j^{m+1} \\ \sum x_j^2 & \sum x_j^3 & \sum x_j^4 & \dots & \sum x_j^{m+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_j^m & \sum x_j^{m+1} & \sum x_j^{m+2} & \dots & \sum x_j^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum y_j \\ \sum y_jx_j \\ \sum y_jx_j^2 \\ \vdots \\ \sum y_jx_j^m \end{bmatrix}$$

式中的 \sum 均指 j 从 1 到 n 取和。求解方程组(7-29), 就可得出 $(m+1)$ 个系数 $a_0, a_1, a_2, \dots, a_m$, 也就求得了拟合多项式 $P_m(x) = a_0 + a_1x + a_2x^2 + \dots + a_mx^m = \sum_{i=0}^m a_ix^i$ 。求解方程组(7-28)和超定方程组的近似解(最小二乘解)有一定关系, 下面予以介绍。

7.4.2 超定方程组的最小二乘解

根据线性代数求解方程组的理论, 如果方程组:

$$p_m(x_j) = \sum_{i=0}^m a_i x_j^i = y_j \quad (j = 1, 2, \dots, n) \quad (7-30)$$

其中 $m < n$ (注意, 这里未知量是 a_i), 即未知量个数少于独立方程个数, 方程组系数矩阵的秩小于增广矩阵的秩, 一般意义下它是无解的, 把这样的方程组叫超定方程组。

虽然无法求出式(7-30)中未知量 a_i 的精确解, 但是可以建立起它的所谓正规方程组 (normal equations), 即式(7-29), 求解式(7-29)可以得到式(7-30)中 a_i 的最小二乘解。这些解并不满足式(7-30)的每个方程, 只是代入每个方程时左边得出的值 y_j^* 与右边 y_j 之差的平方和 $\sum_{j=1}^n (y_j^* - y_j)^2$ 最小, 因此 a_i 叫作式(7-30)的最小二乘解。

由上面的分析可知, 根据已知函数 $y = F(x)$ 的数据组 $y_j, x_j (j = 1, 2, \dots, n)$ 进行代数多项式拟合, 即构造多项式 $P_m(x) \approx F(x)$ 时, 需要解方程组(7-28), 这是件非常繁杂的工作, 特别是当样本点数据很多, 即 j 很大时。而在 MATLAB 中, 有专用的拟合指令, 可以方便地给出求算结果。这里先熟悉 MATLAB 中多项式的表示和运算。

7.5 多项式运算在 MATLAB 中的实现

7.5.1 多项式及其系数向量

1. 多项式的向量表示法

在 MATLAB 中, 多项式是用它的系数构成的向量来表示的, 这个向量完全等价于它代表的多项式, 因此常把“多项式系数向量”就称为“多项式”。该向量的分量自左向右, 依次表示多项式高次幂到低次幂项的系数, 缺少的幂次项, 其系数必须用零填补。

如, 在 MATLAB 中多项式 $p(x) = 3x^4 + 5x^3 - 7x + 9$, 被表示成一个向量 $p = [3 \ 5 \ 0 \ -7 \ 9]$ 。

2. 多项式转换指令

将多项式系数向量恢复成多项式的数学形式, 使用的指令及格式为:

```
poly2str(p, 't')
```

- ① 输入的参数 p 为多项式系数向量;
- ② 输入的 t 是输出多项式的变量, 必须用引号界定, 且不得缺省;
- ③ 输出为一个多项式, 其系数是 p 的各分量, 变量为界定的字母 t 。

例如, 键入:

```
p = [3 5 0 -7 9]; % 代表多项式 3x^4 + 5x^3 - 7x + 9
pp = poly2str(p, 'x')
```

回车得出:

```
pp =
3x^4 + 5x^3 - 7x + 9
```

3. 特殊多项式的创建

一般情况下,创建多项式系数向量需要用键盘输入。但有些特殊多项式,可以用指令 `poly` 创建,调用格式为:

```
p1 = poly(p)
```

① 输入参数 `p` 为向量时,输出向量 `p1` 是以 `p` 向量的分量为根的多项式系数向量, `p1` 与 `p` 各分量的排序无关。

例如,若键入 `p=[a b c d]`, `p1=poly(p)` 这里 `a, b, c, d` 都是具体数值时,回车则得出 `p1=(x-a)(x-b)(x-c)(x-d)` 展开成多项式的系数向量。

② 输入参数 `p` 为方阵时,输出向量 `p1` 是方阵 `p` 的特征多项式系数向量。

例 7-3 求一个多项式,使它的根为 3 5, -7 9 0。

解 键入:

```
p=[3 5 -7 9 0]; p1=poly(p)
```

回车得出:

```
p1 =
     1    -10     32    -474     945     0
```

再键入:

```
poly2str(p1, 's')
```

回车得出:

```
ans =
s^5 -10 s^4 +32 s^3 -474 s^2 +945 s
```

这个多项式的数学表示为 $s^5 - 10s^4 - 32s^3 + 474s^2 - 945s = (s-3)(s-5)(s+7)(s-9)s$ 。改变输入参数 `p` 中各分量的排序,输出向量 `p1` 不变。

7.5.2 多项式运算

在 MATLAB 中多项式间的运算完全转换为系数向量间的运算,不同的运算对多项式系数向量有一定的要求,在使用中需加注意。

1. 多项式加减

与矩阵加减一样,维数相等的系数向量方可进行加减运算,这意味着多项式同幂次对齐后才可加减。要对最高幂次不同的多项式进行加减运算,必须用“零”填补低幂次多项式缺少的高幂次项系数,使它们的系数向量维数相等。

2. 多项式乘法

两个多项式相乘,即求它们系数向量的卷积,指令及使用格式为:

```
conv(p1, p2)
```

输出一个多项式系数向量,其维数比 `p1, p2` 维数之和少 1。

3. 多项式除法

多项式相除就是多项式系数向量的解卷积(卷积的逆运算)指令及使用格式为:

$[q, r] = \text{deconv}(p, w)$

输出的 q 是 p 被 w 除得到的商多项式系数向量, r 为余数多项式向量。

4. 多项式求导

对多项式求导即对其系数向量求导,其指令及使用格式为:

$k = \text{polyder}(p)$,

输出的 k 也是向量,是对 p 求导得出的多项式系数向量。

5. 多项式求值

计算用“数”或“矩阵”代替多项式中变量得出的数值结果时,用下述指令:

$\text{polyval}(p, x_0)$

① 输入参数 p 为多项式系数向量;

② 输入参数 x_0 为数值时,得出多项式函数在 x_0 处的值;

③ 输入参数 x_0 为矩阵(或向量)时,输出按数组算法得出的矩阵多项式“值”,即与 x_0 同维的数值矩阵(或向量)。

例 7-4 已知 $p_1 = 3x^2 + 2x + 6$, $p_2 = 3x^4 + 5x^3 - 7x + 9$,求两个多项式的和。

解 键入:

$p_1 = [3 \ 2 \ 6]$; $p_2 = [3 \ 5 \ 0 \ -7 \ 9]$;

$p_1 = [0 \ 0 \ p_1]$; $p = p_1 + p_2$

回车得出:

$p =$
3 5 3 -5 15

键入:

$\text{poly2str}(p, 'x')$

回车得出:

$\text{ans} =$
 $3x^4 + 5x^3 - 5x^2 + 3x + 15$

例 7-5 已知 $p_1 = 3x^4 + 2x^3 - 7x + 2$, $p_2 = 3x^3 - x^2 + 6$,求这两个多项式的乘积。

解 在指令窗中键入:

$p_1 = [3 \ 2 \ 0 \ -7 \ 2]$; $p_2 = [3 \ -1 \ 0 \ 6]$; $pp = \text{conv}(p_1, p_2)$

回车得出:

$pp =$
9 3 2 3 25 2 42 12

键入:

$\text{poly2str}(pp, 'x')$

回车得出:

```
ans =
    9x^7 - 8x^6 - 2x^5 - 3x^4 - 25x^3 - 2x^2 - 42x + 12
```

表明：

$$p1 \cdot p2 = (3x^4 + 2x^3 - 7x + 2) \cdot (3x^3 - x^2 + 6) = 9x^7 + 3x^6 - 2x^5 - 3x^4 + 25x^3 - 2x^2 - 42x + 12$$

例7-6 利用例7-5中的向量,计算 $pp/p2$ 和 $pp/p1$ 。

解 键入：

```
[q, r] = deconv(pp, p1)
```

回车得出：

```
q =
    3.0000    -4.0000     0     6.0000
r =
    1.0e-014 *
    0     0     0    0.3553     0     0    -0.7105    0.1776
```

键入：

```
[q, r] = deconv(pp, p2)
```

回车得出：

```
q =
    3.0000    2.0000     0    -7.0000    2.0000
r =
    1.0e-014 *
    0     0     0     0    0.3553    -0.0888     0    0.5329
```

对照例7-5中的数据,可验证其结果,余数数量级很小,仅为 10^{-14} ,可以忽略。

例7-7 已知多项式 $p1 = x^4 + 2x^2 - 7x - 2$,求当 $x = -2, (4, 2, 3)$ 和 $\begin{bmatrix} 2 & 4 \\ 3 & -2 \end{bmatrix}$ 时的取值。

解 键入：

```
p1 = [1 0 2 -7 -2]; polyval(p1, 2)
```

回车得出：

```
ans =
    36
```

键入：

```
polyval(p1, [4 2 3])
```

回车得出：

```
ans =
    258     8    76
```

键入：

```
polyval(p1, [2 4; 3 -2])
```

回车得出：

```
ans =
```


8	258
76	36

7.6 曲线拟合在 MATLAB 中的实现

7.6.1 数据的多项式曲线拟合

用代数多项式拟合数据的指令是 `polyfit` ,使用格式为 :

```
p = polyfit(x,y,m)
```

① 输入参数 x, y 为两个同维向量 (即 $\text{length}(x) = \text{length}(y)$) ,提供了满足函数关系 $y = F(x)$ 的一组对应样本点 y_i, x_i 的数据 ;

② 输入参数 m 为拟合代数多项式的次数 ,原则上 m 小于 x 的维数即可 $m < \text{length}(x) = \text{length}(y)$,但是一般取小于 6 的正整数 ;

③ 输出参数 p 为拟合多项式的系数向量。

例 7-8 已知 $y = f(x) = \cos(x) + 0.5 \times \text{rand}(\text{size}(x))$, $x \in [0, 3\pi]$ 。求出一组对应的 (x, y) 数据 ,并据此把 $f(x)$ 拟合成一个三次多项式 p ,画出 $f(x)$ 和 p 的曲线。

解 设 x 按步长 0.3 取分点值 ,键入 :

```
x = 0:0.3:3*pi; y = cos(x) + 0.5*rand(size(x)); % 得出样本点
p = polyfit(x,y,3) % 得出 3 次拟合多项式系数向量 p
```

回车得出 :

```
p =
    0.0371    0.5276    2.0201    1.9277
```

键入 :

```
poly2str(p,'x') % 求 3 次拟合多项式表达式
```

回车得出 :

```
ans =
    0.0371x^3 + 0.5276x^2 + 2.0201x + 1.9277
```

可知 3 次拟合多项式为 $p(x) = -0.0371x^3 + 0.5276x^2 - 2.0201x + 1.9277$ 。

为了把样本点和拟合曲线画在同一张图上 ,键入 :

```
x1 = 0:0.2:3*pi; y1 = polyval(p,x1); plot(x,y,'*',x1,y1), grid
legend('样本点','拟合曲线 p3(x)')
```

回车得出图 7-4。

由图 7-4 可以看出 ,拟合曲线并未通过所有样本点 ,它冲去了一些随机误差 ,更能真实地反映出两组量间关系变化的趋势。由于 $y = \cos(x) + 0.5 \times \text{rand}(\text{size}(x))$ 中含有随机函数 ,所以得出的样本点 y 数据和拟合曲线具有不可重复性。

7.6.2 多项式数据拟合应用的扩充

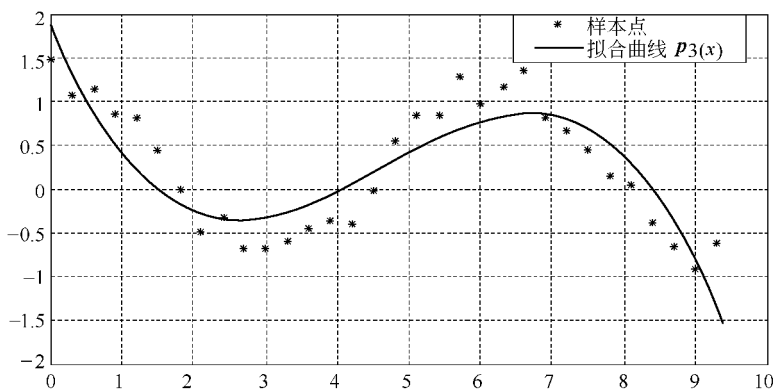


图 7-4 数据点的曲线拟合图

1. 任意函数的多项式拟合

除了经常把一组数据拟合成多项式外,还常把一些运算繁杂的解析函数拟合成多项式,因为代数多项式值的计算比较简单。用代数多项式拟合任意函数的基本步骤如下:

- ① 将自变量 x 离散化为 x_k ($k=1, 2, \dots, n$, 正整数 n 视步长而定), 选取适当的步长;
- ② 用函数求值法算出与自变量 x_k 对应的函数值 $y_k = f(x_k)$, 求值时函数表达式中各变量间用数组算法符号连接;
- ③ 用 `polyfit` 指令根据 x_k 和 y_k 数据拟合出多项式 $p(x)$, 得出该多项式的系数向量。函数偏离线性程度越大, 选取的拟合多项式次数应该越高;
- ④ 依据得出的多项式系数向量, 用 `poly2str` 求出拟合多项式表达式。

例 7-9 用一个二次多项式(即抛物线)拟合悬链线 $y = 5 \cosh \frac{x}{5}$ 并绘出它们的图形。

解 由悬链线方程可知, 用解析式计算给定自变量 x 对应的函数值 y 是较困难的, 若把函数拟合成二次多项式, 计算起来则方便很多。

键入:

```
x = 2 * pi : 0.2 : 2 * pi ; y = 1 / 5 * cosh ( x . / 3 ) ;    % 将自变量和函数离散化
p = polyfit ( x , y , 2 )                                     % 得出拟合多项式系数向量
```

回车得出:

```
p =
    0.0151    0.0003    0.1836
```

键入:

```
pk = poly2str ( p , 'x' )                                     % 得出拟合多项式表达式
pk =
    0.015071x^2 0.00029557x 0.18365
```

键入:

```
ezplot ( 1 / 5 * cosh ( x / 3 ) , hold )                       % 作出悬链线函数曲线并保留
```

`ezplot 0.015071*x^2 0.00029557*x 0.18365 % 作出拟合悬链线函数的多项式函数曲线`
`grid, legend('悬链线函数线', '拟合悬链线的二次多项式曲线')`
回车得出图 7-5。

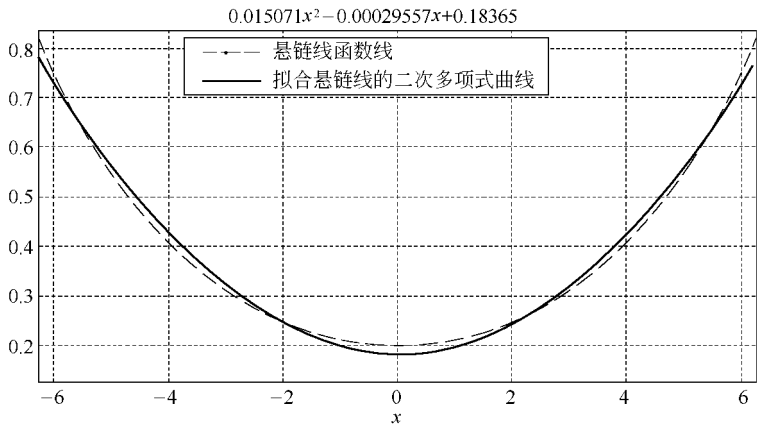


图 7-5 悬链线函数和拟合它的二次多项式函数曲线

2. 数据拟合成指数函数等常用函数

实际问题中有时需要把两组数据变量之间的关系拟合成常用的指数、双曲线、对数等一些函数形式,这时可以先把数据进行适当的变换,然后再用多项式拟合指令进行拟合。

例如,要用一个指数函数 $y = ae^{bx}$ (a, b 均为常数)去拟合一组实验数据 x, y ,首先对指数函数两边取对数得出 $\log y = \log a + bx \log e$,令 $u = \log y, B = b \log e$,指数函数就变换成 $u = \log a + Bx$ 。 u 和 x 是线性关系,只要把实验数据中的 y 通过 $u = \log y$ 变换成 u ,则完全可以用指令 `polyfit` 拟合成一次多项式。

又如要用 $xy = k$ (k 为常数)的双曲线函数去拟合一组实验数据 x, y ,可将双曲线函数两边取对数,得出 $\log x + \log y = \log k$,令 $u = \log y, v = \log x, B = \log k$,双曲线函数就变成 $u = B - v$ 。通过 $u = \log y$ 和 $v = \log x$ 将已知数据 x, y 变换 u 和 v 就完全可以用 `polyfit` 指令拟合成一次多项式。

按照上述思路可以先将常用函数进行求对数、倒数、指数等变换,使它们成为线性形式,然后利用 MATLAB 处理批量数据便捷的优点,对实验数据作相应的变换,再用拟合指令 `polyfit` 对变换后的数据进行拟合,就可得出满足要求的结果。

练习题 7

1. 某海洋的水温 t 随水的深度 h 而变化,现有一组测量数据如表 7-4 所示。用插值指令 `interp1` 在不同的“方法”下求出水深 $h_1 = 500\text{ m}, h_2 = 900\text{ m}, h_3 = 1\,500\text{ m}$ 处水温。

表 7-4 水温 t 和水深 h 函数关系数据表

$h(\text{m})$	466	715	950	1 422	1 635
---------------	-----	-----	-----	-------	-------

t()	7.04	4.28	3.40	2.52	2.13
------	------	------	------	------	------

2. 已知函数 $y = F(x)$ 的部分数值列于表 7-5 中。据此数据 ,用插值指令(线性插值和三次样条插值两种方法)求出 $x = 1.57, 1.68, 1.81$ 和 1.90 处的函数值 ,并画出这两种方法的插值曲线图 ,样本点和曲线用不同的线型并加以标注。

表 7-5 习题 2 函数 $y = F(x)$ 关系数据表

x	1.55	1.62	1.63	1.70	1.92
y	2.587	2.415	2.465	3.031	3.341

3. 求一个多项式 ,使它的根为 $5, 7, 8, 0$ 和 1 。

4. 求出当 $x = 45, -123$ 和 579 时 ,多项式 $3x^5 + 9x^3 + 60x^2 - 90$ 的值。

5. 已知 $p_1(x) = 13x^4 + 55x^3 - 17x + 9$, $p_2(x) = 63x^5 + 26x^3 - 85x^2 + 105$,求 $p_1(x)p_2(x)$ 和 $p_2(x)/p_1(x)$ 。

6. 根据表 7-6 中给出的 $y = F(x)$ 部分数据 ,用多项式拟合该函数 ,写出一、二、三次拟合多项式 ,并做出曲线 ,加以标注。

表 7-6 习题 6 函数 $y = F(x)$ 关系数据表

x	-3	-2	0	3	4
y	18	10	2	2	5

7. 根据表 7-7 中 $y = F(x)$ 的一组数据 ,在同一张图上用不同的线型画出线性插值、三次样条插值及二、四次多项式拟合曲线图 ,并加以标注。

表 7-7 习题 7 函数 $y = F(x)$ 关系数据表

x	1	2	3	4	5	6
y	7.4	4.1	3.0	3.5	4.8	2.3

8. 用一个抛物线函数拟合概率曲线 $y = e^{-5x^2}$ ($-1 < x < 1$) ,并在同一图上作出两个函数的曲线。

第8章 数值积分

解决许多科技问题时,经常需要进行定积分的计算。微积分教材里讲过定积分的计算方法,但主要是讲如何利用牛顿-莱布尼兹(Newton-Leibniz)公式 $\int_a^b f(x)dx = F(b) - F(a)$ 进行计算。应用这个公式时,被积函数 $f(x)$ 在 $[a, b]$ 上必须连续,先求出 $f(x)$ 的一个原函数 $F(x)$,即满足 $\frac{dF(x)}{dx} = f(x)$ 的 $F(x)$,然后再把积分上下限代入 $F(x)$ 。然而在具体应用中求一个函数的原函数并非易事,而且由于种种原因,通常是很难利用这个公式求出定积分的,举例如下。

① 有不少被积函数 $f(x)$ 理论上讲一定存在原函数,但无法用初等函数的有限形式表示出来。例如 $\sin x^2, \frac{\sin x}{x}, \ln^{-1} x, e^{-x^2}, \dots$

② 有些被积函数 $f(x)$ 的原函数需要用很高的技巧方可求出,或者原函数非常冗长,实际上难于应用。例如,下式表明 $F(x)$ 是 $f(x)$ 的原函数:

$$F'(x) = \left(\frac{1}{4}x^2 \sqrt{2x^2 + 3} + \frac{1}{\ln x}x \sqrt{2x^2 + 3} - \frac{9}{16\sqrt{2}} \ln(x\sqrt{2} + \sqrt{2x^2 + 3}) \right)', = x^2 \sqrt{2x^2 + 3} = f(x)$$

但是,若用这样的原函数 $F(x)$ 去计算定积分 $\int_a^b f(x)dx = F(b) - F(a)$ 是非常繁杂的。

③ 有些从工程实际或科学实验中测得的被积函数本身就不是解析表达式,而是表格或曲线,计算它们的定积分,就更是无法找到解析形式的原函数。

可见,应用牛顿-莱布尼兹公式计算定积分虽然非常精确,但是有很大局限性。实际工作中不一定非得算出定积分的绝对精确值,只要达到工作需求的精度就足够了。因此,研究计算定积分的近似方法——数值积分是非常必要的。

本章首先介绍 MATLAB 计算积分解析解和精确解的符号法,然后介绍计算定积分数值解的基本原理及用 MATLAB 实现这些数值解的常用指令。

8.1 计算积分的 MATLAB 符号法

使用 MATLAB 的符号计算功能,可以计算出许多积分的解析解和精确解,只是有些精确解显得冗长烦杂,这时可以用 `vpa` 指令把它转换成位数有限的数字,有效数字的长度可按需选取。符号法计算积分非常方便,常常用它算得的结果跟近似计算结果进行比较,所以先予介绍。

求积分的符号运算指令 `int` (取自 `integrate` 的前三个字母),调用格式为:

$$s = \text{int}(\text{fun}, v, a, b)$$

① 输入参量 `fun` 是被积函数的符号表达式,可以是函数向量或函数矩阵。

② 输入参量 `v` 是积分变量,必须被界定成符号变量。如果被积函数中只有一个变量时

可以缺省。

③ 输入参数 a, b 为定积分的积分限, 缺省时输出被积函数 fun 的一个原函数。

④ 输出参量 s 为积分结果。 s 为有理表达式而过于冗长时, 可在 fun 两端加单引号, 使它自动转换成默认的 32 位有效数字, 或用 vpa 指令把它转换成有限长度的小数。

例 8-1 计算 $\int_0^{\pi/3} \sin x dx$ 。

解 在指令窗中键入：

```
syms x % 把积分变量 x 定义成符号变量
int(sin(x) 0 pi/3)
```

回车得出：

```
ans =
    1/2
```

如果求不定积分 $\int \sin x dx$ 则可键入：

```
syms x c ,int(sin(x)) + c 或 syms x ,int(sin(x)) + 'c'
```

回车得出：

```
ans =
    -cos(x) + c
```

c 是加上去的积分常数, 本来的输出结果只给出一个原函数 $-\cos(x)$ 。

例 8-2 计算定积分 $\int_0^{\pi/3} 3x^y dy$ 。

解 在指令窗中键入：

```
syms x y ,int(3* x^y y 0 pi/5)
```

回车得出：

```
ans =
    3* (x^(1/5* pi) - 1)/log(x)
```

例 8-3 计算定积分 $s = \int_0^{3\pi} y(t) dt$ $y(t) = e^{-0.6t} \sin(t + (\pi/3))$ 。

解 键入：

```
syms x s = int(exp(-0.6* x)* sin(x+pi/3) x 0 3* pi)
```

回车得出：

```
s =
    25/68* exp(-9/5* pi) +15/68* exp(-9/5* pi)* 3^(1/2) +25/68 +15/68
* 3^(1/2)
```

该结果太冗长, 可用下述方法转换成数字。

(1) 在被积函数两端加单引号, 键入：

```
syms t s = int('exp(-0.6* t)* sin(t+pi/3)' ,t 0 3* pi)
```

回车得出：

```
s =
    .75234142914275707307821260870297
```

(2) 用 vpa 对积分结果进行转换,键入:

```
s1 = vpa(s 6)
```

回车得出 6 位有效数字:

```
s1 =
    .752340
```

例 8-4 计算广义积分 $\int_{-\infty}^{+\infty} \frac{dt}{1+t^2}$ 。

解 在指令窗中键入:

```
syms t ,int(1/(1+t^2) ,-inf ,inf)
```

回车得出:

```
ans =
    pi
```

积分结果表明 $\int_{-\infty}^{+\infty} \frac{dt}{1+t^2} = \pi$ 是精确解。 π 属永久变量 pi 表示的无理数,因此无论被积函数 $1/(1+t^2)$ 两端是否加引号,积分结果都一样是 pi。需要用小数时,可用 vpa(pi,n) 转换成 n 位有效数字。

例 8-5 已知 $A(t) = (a(t))_{2 \times 3} = \begin{bmatrix} \cos xt & e^{2xt} & \sin xt \\ -\sin xt & 5x^3 & -\cos xt \end{bmatrix}$, 一次计算 6 个定积分 $s =$

$$\int_{-1}^2 a(t) dt.$$

解 键入:

```
syms x t A=[cos(x*t) exp(2*x*t) sin(x*t);sin(x*t) 5*x^3 -cos(x*t)];
int(A t ,1 2)
```

回车得出:

```
ans =
    [sin(2*x) + sin(x))/x ,1/2*(exp(4*x)-exp(-2*x))/x ,(cos(2*x)-
cos(x))/x ]
    [(cos(2*x)-cos(x))/x ,15*x^3 ,
(x))/x ]
```

例 8-6 求二重积分 $\iint_D t^2 e^{-3tx} dt dx$ 。

解 键入:

```
syms t x c1 c2 y = int(int(t^2*exp(-3*t*x) ,t) +c1 ,x) +c2 % c1 和 c2
是积分常数
```

回车得出:

```
y = -
-1/3*t^2*(-1/3/t/x*exp(-3*t*x)-1/9/t^2/x^2*exp(-3*t*x)) +
c1*x +c2
```

即 $\iint_D t^2 e^{-3tx} dt dx = c2 + c1x + \frac{t}{9x} \left(1 + \frac{1}{3tx}\right) e^{-3tx}$

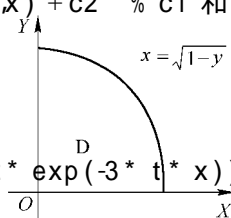


图 8-1 积分区域 D

例 8-7 计算积分 $\iint_D 3x^2y^2 d\sigma$ D 是由 X 轴、 Y 轴和抛物线

$y = 1 - x^2$ 在第一象限内所围成的区域。

解 积分区域 D 如图 8-1 所示 先把二重积分化成二次积分：

$$\iint_D 3x^2y^2 d\sigma = \int_0^1 \left[\int_0^{1-x^2} 3x^2y^2 dy \right] dx = \int_0^1 \left[\int_0^{\sqrt{1-y}} 3x^2y^2 dx \right] dy$$

用先对变量 y 积分的二次积分 在指令窗中键入：

```
syms x y;int(int(3*x^2*y^2,y,0,1-x^2),x,0,1)
```

回车得出：

```
ans =
```

```
16/315
```

8.2 牛顿-柯特斯求积公式

并非所有定积分都能应用牛顿-莱布尼兹公式得出理想结果,大多数实际问题的积分需要用数值积分方法求出近似结果。数值积分原则上可以用于计算各种被积函数的定积分,无论被积函数是解析形式还是数表形式,其基本原理都是用多项式函数近似代替被积函数,用对多项式的积分结果近似代替对被积函数的积分。由于所选多项式的形式的不同,可以有多种数值积分方法,下面仅介绍一种最常用的插值型数值积分方法。

用一个容易积分的函数 $\phi(x)$ 去代替被积函数 $f(x)$,即找出满足 $\int_a^b f(x)dx \approx \int_a^b \phi(x)dx$ 的 $\phi(x)$ 。这样的 $\phi(x)$ 自然以多项式 $P_n(x)$ 为最佳,因为多项式能很好地逼近任何连续函数,而且容易求出其原函数。下边介绍的牛顿-柯特斯求积公式,就是用多项式函数 $P_n(x)$ 近似地代替被积函数 $f(x)$,用对 $P_n(x)$ 的积分代替对 $f(x)$ 的积分推出的一种数值积分方法。

8.2.1 牛顿-柯特斯求积公式推导

1. 牛顿-柯特斯求积公式

用代数多项式 $P_n(x)$ 近似代替被积函数 $f(x)$ 时,有：

$$\int_a^b f(x)dx \approx \int_a^b P_n(x)dx$$

为了用 $P_n(x)$ 近似代替 $f(x)$,可以用前面讲过的插值法,先将被积函数 $f(x)$ 的自变量 x 离散化,即把 $[a, b]$ 区间分成 n 等份,步长为 h ,等分点的坐标则为：

$x_i = a + ih$ 其中 $i=0, 1, 2, \dots, n$ $h = (b - a)/n$ $x_0 = a$ $x_n = b$

再用 $y = f(x)$ 的关系,求出函数在这 $(n+1)$ 个等分点上的取值 $y_i = f(x_i) = f(a + ih)$ $i=0, 1, 2, \dots, n$ 。于是 根据式(7-12)在区间 $[a, b]$ 上构造出一个逼近函数 $f(x)$ 的 n 次 Lagrange 插值

多项式：

$$P_n(x) = \sum_{i=0}^n \frac{\omega(x)}{(x-x_i)\omega'(x_i)} f(x_i) = \sum_{i=0}^n \frac{\omega(x)}{(x-x_i)\omega'(x_i)} y_i$$

式中, $\omega(x) = (x-x_0)(x-x_1)\dots(x-x_n)$ $\omega'(x_i) = (x_i-x_0)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)$

于是有：

$$\int_a^b f(x)dx \approx \int_a^b P_n(x)dx = \int_a^b \left(\sum_{i=0}^n \frac{\omega(x)}{(x-x_i)\omega'(x_i)} f(x_i) \right) dx$$

调换积分符号和累加和符号,并令 $K_i = \int_a^b \frac{\omega(x)}{(x-x_i)\omega'(x_i)} dx$ 则得出：

$$\int_a^b f(x)dx \approx \int_a^b P_n(x)dx = \sum_{i=0}^n \left(\int_a^b \frac{\omega(x)}{(x-x_i)\omega'(x_i)} dx \right) f(x_i) = \sum_{i=0}^n K_i f(x_i)$$

对 K_i 进行变量代换:令 $x = a + th$ $t \in [0, n]$ 则 $dx = hdt$ 。把 $x - x_i = (t - i)h$ 和 $x_i - x_0 = ih$ 代入 $\omega(x)$ 和 $\omega'(x_i)$ 可得出：

$$\omega(x) = \omega(a + th) = h^{n+1} t(t-1)(t-2)\dots(t-n) \text{ 和 } \omega'(x_i) = h^n (-1)^{n-i} (i!)(n-i)!$$

则：

$$\begin{aligned} K_i &= \int_a^b \frac{\omega(x)}{(x-x_i)\omega'(x_i)} dx = \int_0^n \frac{h^{n+1} t(t-1)(t-2)\dots(t-n)}{(-1)^{n-i} h^n (i!)(n-i)! h(t-i)} hdt \\ &= \frac{(-1)^{n-i} h}{i!(n-i)!} \int_0^n \frac{t(t-1)(t-2)\dots(t-n)}{(t-i)} dt \end{aligned}$$

$$\begin{aligned} \text{引入记号} \quad c_i^{(n)} &= \frac{(-1)^{n-i}}{i! h(n-i)!} \int_0^n \frac{t(t-1)(t-2)\dots(t-n)}{(t-i)} dt \\ &= \frac{(-1)^{n-i}}{i! h(n-i)!} \int_0^n t(t-1)\dots(t-i+1)(t-i-1)\dots(t-n) dt \quad (8-1) \end{aligned}$$

注意 $h = (b-a)/n$, 于是 $K_i = nhc_i^{(n)} = (b-a)c_i^{(n)}$, 而 $c_i^{(n)}$ 与被积函数 $f(x)$ 、积分区间 $[a, b]$ 毫无关系, 仅取决于插值多项式的次数 n , 称它为柯特斯求积系数 (正整数 i 满足 $0 \leq i \leq n$, 为系数序号)。这样便得出牛顿-柯特斯求积公式：

$$\begin{aligned} \int_a^b f(x)dx &\approx \int_a^b P_n(x)dx = \sum_{i=0}^n K_i f(x_i) = (b-a) \sum_{i=0}^n c_i^{(n)} f(x_i) \\ &= (b-a) [c_0^{(n)} f(x_0) + c_1^{(n)} f(x_1) + c_2^{(n)} f(x_2) + \dots + c_{n-1}^{(n)} f(x_{n-1}) + c_n^{(n)} f(x_n)] \quad (8-2) \end{aligned}$$

2. 柯特斯求积系数

既然柯特斯系数跟被积函数、积分区间无关, 只与代替被积函数的多项式次数 n 有关, 就可以在求积分前预先算出来。例如, 用 4 次多项式代替被积函数时, 即取 $n=4$, 牛顿-柯特斯求积公式为：

$$\int_a^b f(x)dx \approx \int_a^b P_4(x)dx = \sum_{i=0}^4 K_i f(x_i) = (b-a) \sum_{i=0}^4 c_i^{(4)} f(x_i)$$

$$= (b - a)[c_0^{(4)}f(x_0) + c_1^{(4)}f(x_1) + c_2^{(4)}f(x_2) + c_3^{(4)}f(x_3) + c_4^{(4)}f(x_4)]$$

式中的 $f(x_i)(i=0,1,2,3,4)$ 是被积函数在 $[a,b]$ 区间等分点上的取值,系数 $c_0^{(4)}, c_1^{(4)}, c_2^{(4)}, c_3^{(4)}$ 和 $c_4^{(4)}$ 可以用式(8-1)计算,如计算系数 $c_2^{(4)}$ 时,将 $n=4, i=2$ 代入式(8-1)即可得出:

$$c_2^{(4)} = \frac{(-1)^{4-2}}{4 \times 2! \times (4-2)!} \int_0^4 t(t-1)(t-3)(t-4)dt = \frac{2}{15}$$

利用柯特斯求积系数 $c_i^{(n)}$ 的这个特性,可以预先把它们都计算出来列成表格,使用时只要查表即可。表 8-1 就是部分柯特斯系数的取值表。

表 8-1 部分柯特斯系数取值表

n	$c_i^{(n)}(0 \leq i \leq n)$
0	1
1	$\frac{1}{2}, \frac{1}{2}$
2	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$
3	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$
4	$\frac{7}{90}, \frac{16}{45}, \frac{2}{15}, \frac{16}{45}, \frac{7}{90}$
5	$\frac{19}{288}, \frac{25}{96}, \frac{25}{144}, \frac{25}{144}, \frac{25}{96}, \frac{19}{288}$
6	$\frac{41}{840}, \frac{9}{35}, \frac{9}{280}, \frac{34}{105}, \frac{9}{280}, \frac{9}{35}, \frac{41}{840}$
7	$\frac{751}{17280}, \frac{3577}{17280}, \frac{1323}{17280}, \frac{2989}{17280}, \frac{2989}{17280}, \frac{1323}{17280}, \frac{3577}{17280}, \frac{751}{17280}$
8	$\frac{989}{28350}, \frac{5888}{28350}, \frac{928}{28350}, \frac{10496}{28350}, \frac{4540}{28350}, \frac{10496}{28350}, \frac{928}{28350}, \frac{5888}{28350}, \frac{989}{28350}$

柯特斯系数 $c_i^{(n)}$ 还具有下述特性。

- ① 在 $c_i^{(n)}$ 定义式中作代换 $t = n - i$ 后其值不变,所以它具有对称性,即 $c_i^{(n)} = c_{n-i}^{(n)}$ 。
- ② $\sum_{i=0}^n c_i^{(n)} = 1$ 。将 $f(x) \equiv 1$ 代入牛顿-柯特斯求积公式便可得出该性质,将表 8-1 同一行中的 $c_i^{(n)}$ 取值相加,也可以验证该性质。
- ③ 由于当 $n \geq 8$ 时,柯特斯系数出现负值,根据误差理论此时将导致舍入误差的急剧增大。因此,一般都采用次数较低的插值多项式逼近被积函数,通常 n 的取值不得大于 8。

8.2.2 牛顿-柯特斯求积公式的误差估计

1. 牛顿-柯特斯求积公式的截断误差

牛顿-柯特斯公式是一个插值型数值求积公式,当用插值多项式 $P_n(x)$ 代替 $f(x)$ 进行积分时,其截断误差 $R(f)$,即积分真值 I 和近似值之差可推导如下。

$$R(f) = I - \int_a^b P_n(x)dx = \int_a^b f(x)dx - \int_a^b P_n(x)dx = \int_a^b (f(x) - P_n(x))dx$$

由插值多项式的误差估计可知,用 n 次 Lagrange 多项式 $P_n(x)$ 逼近函数 $f(x)$ 时,产生的误

差为：

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$$

其中 $\xi \in [a, b]$, $\omega_{n+1}(x) = (x - x_0)(x - x_1)\dots(x - x_n) = \prod_{i=0}^n (x - x_i)$ 。对上式两边从 a 到 b 作定积分,便得出它的截断误差：

$$R(f) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b \omega_{n+1}(x) dx \quad (8-3)$$

2. 数值求积公式的代数精度

由式(8-3)可知,截断误差与被积函数 $f(x)$ 、积分限密切相关,如果被积函数 $f(x)$ 是一个 n 次多项式,由于 $f^{(n+1)}(x) = 0$,则 $R(f) = 0$,就会使 $\int_a^b f(x) dx = \int_a^b P_n(x) dx$ 。所以,被积函数 $f(x)$ 为高次多项式时,能使求积公式(8-2)成为精确的等式,便成为衡量数值求积公式精确程度的一个指标。据此,提出数值求积公式代数精度这一概念。

如果被积函数 $f(x)$ 为任意一个次数不高于 n 次的多项式时,数值求积公式一般形式的截断误差 $R(f) = 0$,而当它是 $(n+1)$ 次多项式时, $R(f) \neq 0$,则说明数值求积公式具有 n 次代数精度。一个数值求积公式的代数精度越高,表示用它求数值积分时所需逼近被积函数的多项式次数越高。

3. 牛顿-柯特斯求积公式的代数精度等于 n

如果被积函数 $f(x)$ 是一个不大于 n 次的多项式,则 $f^{(n+1)}(x) = 0$,即 $R(f) = 0$;而当 $f(x)$ 是任意一个 $(n+1)$ 次多项式时, $f^{(n+1)}(x) \neq 0$,故 $R(f) \neq 0$ 。所以,按照代数精度的定义可知,一般情况下,牛顿-柯特斯求积公式的代数精度等于 n ;但当 n 为偶数时,其代数精度为 $n+1$ 。下面对此加以证明。

定理 当 n 为偶数时,牛顿-柯特斯求积公式的代数精度为 $n+1$ 。

证明 当 $f(x)$ 为 n 次多项式时, $f^{(n+1)}(\xi) = 0$ ($\xi \in [a, b]$),牛顿-柯特斯求积公式的代数精确度至少等于 n 。若设 $f(x)$ 是一个 $(n+1)$ 次多项式,这时 $f^{(n+1)}(\xi)$ 为一常数,而：

$$R(f) = \int_a^b (f(x) - P_n(x)) dx = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b (x - x_0)(x - x_1)\dots(x - x_n) dx$$

因此,只要证明在 n 为偶数时, $\int_a^b (x - x_0)(x - x_1)\dots(x - x_n) dx = 0$, $R(f) = 0$,上述定理就得证。为此,设 $x_{i+1} - x_i = h$ ($i=0, 1, 2, \dots, n$),令 $x = a + th$, $t \in [0, n]$, $dx = h dt$ 于是：

$$\int_a^b (x - x_0)(x - x_1)\dots(x - x_n) dx = h^{n+1} \int_0^n t(t-1)(t-2)\dots(t-n) dt$$

由于 n 为偶数,不妨设 $n=2k$, k 为正整数,则 $t \in [0, 2k]$ 。于是：

$$\int_0^n t(t-1)(t-2)\dots(t-n) dt = \int_0^{2k} t(t-1)\dots(t-k)(t-k-1)\dots(t-2k-1)(t-2k) dt$$

再引进变换 $u = t - k$, 则 $t = u + k$, $du = dt$, $u \in [-k, k]$, 代入上式右侧,得出：

$$\begin{aligned}\int_0^n t(t-1)(t-2)\dots(t-n)dt &= \int_{-k}^k (u+k)(u+k-1)\dots(u+1)u(u-1)\dots(u-k+1)(u-k)du \\ &= \int_{-k}^k u(u^2-1)\dots(u^2-(k+1)^2)(u^2-k^2)du\end{aligned}$$

最后的积分中被积函数是奇函数,所以积分结果等于零,定理得证。

8.3 几个低次牛顿-柯特斯求积公式

从上面的讨论可知,用多项式近似代替被积函数进行数值积分时,虽然最高次数可以是8,但是8次多项式的计算是非常繁杂的,一般常用的是下边介绍的低次多项式。

8.3.1 矩形求积公式

在牛顿-柯特斯求积公式中,如果取 $n=0$,用零次多项式(即常数)代替被积函数,即用矩形面积代替曲边梯形的面积,则有:

$$\int_a^b f(x)dx \approx \int_a^b P_0(x)dx = (b-a)c_0^{(0)}f(x_0) = (b-a)f(x_0) \quad (8-4)$$

根据牛顿-柯特斯求积公式的误差理论式(8-3),矩形求积公式的误差估计为:

$$R_0(f) = \frac{f^{(0+1)}(\xi)}{(0+1)!} \int_a^b \omega_{0+1}(x)dx = f'(\xi)(b-a)$$

8.3.2 梯形求积公式

在牛顿-柯特斯求积公式中,如果取 $n=1$,用一次多项式代替被积函数,即用梯形面积代替曲边梯形的面积,则有:

$$\int_a^b f(x)dx \approx \int_a^b P_1(x)dx = (b-a)[c_0^{(1)}f(x_0) + c_1^{(1)}f(x_1)]$$

其中 $f(x_0)=f(a)$, $f(x_1)=f(b)$,查表可得 $c_0^{(1)}=c_1^{(1)}=1/2$,代入上式得出:

$$\int_a^b f(x)dx \approx \int_a^b P_1(x)dx = \frac{b-a}{2}[f(a) + f(b)] \quad (8-5)$$

由于用一次多项式 $P_1(x)$ 近似代替被积函数 $f(x)$,所以它的代数精度是1,也就是说,只有当被积函数是一次多项式时,梯形求积公式才是准确的。

根据牛顿-柯特斯求积公式的误差理论式(8-3),梯形求积公式的误差估计为:

$$R_1(f) = \frac{f''(\xi)}{2!} \int_a^b \omega_2(x)dx = \frac{f''(\xi)}{2!} \int_a^b (x-a)(x-b)dx = -\frac{(b-a)^3}{12}f''(\xi) \quad (8-6)$$

$f''(\xi)$ 是被积函数 $f(x)$ 二阶导数在 $x=\xi$ 点的取值 $\xi \in [a, b]$ 。

8.3.3 抛物线求积公式

1. 抛物线(Simpson)求积公式的推导

在牛顿-柯特斯求积公式中,如果取 $n=2$,用二次多项式代替被积函数,即曲边用抛物线代替,则有:

$$\int_a^b f(x)dx \approx \int_a^b P_2(x)dx = (b-a)[c_0^{(2)}f(x_0) + c_1^{(2)}f(x_1) + c_2^{(2)}f(x_2)]$$

其中 $x_0=a$, $x_1=\frac{a+b}{2}$, $x_2=b$,查表可得 $c_0^{(2)}=c_2^{(2)}=1/6$, $c_1^{(2)}=2/3$,代入上式得出:

$$\begin{aligned}\int_a^b f(x)dx &\approx \int_a^b P_2(x)dx = (b-a)\left[\frac{1}{6}f(a) + \frac{2}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{6}f(b)\right] \\ &= \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \quad (8-7)\end{aligned}$$

这就是抛物线求积公式。它的几何意义是:用过三个点 $(a, f(a))$, $(\frac{a+b}{2}, f(\frac{a+b}{2}))$, $(b, f(b))$ 的抛物线和 $x=a$, $x=b$ 构成的曲边梯形面积,近似地代替了被积函数 $f(x)$ 形成的曲边和 $x=a$, $x=b$ 构成的曲边梯形面积。

2. 抛物线求积公式的误差

下面对抛物线求积公式的误差进行估计。由于抛物线求积公式是用二次多项式逼近被积函数推得的,原则上它的代数精度应该为2。但因多项式次数 $n=2$ 是偶数,据前面的定理知道,它的代数精度为3。

过 $(a, f(a))$, $(\frac{a+b}{2}, f(\frac{a+b}{2}))$ 和 $(b, f(b))$ 三点,构造一个 $f(x)$ 的三次 Lagrange 插值多项式 $P_3(x)$,且使 $P_3'(\frac{a+b}{2}) = f'(\frac{a+b}{2})$ 。根据式(7-16)可以得出:

$$f(x) - P_3(x) = \frac{f^{(4)}(\xi)}{4!} \omega_4(x) = \frac{f^{(4)}(\xi)}{4!} (x-a)\left(x - \frac{a+b}{2}\right)^2 (x-b), \quad \xi \in [a, b].$$

对上式两边从 a 到 b 进行定积分,即可得到:

$$R_2(f) = \int_a^b [f(x) - P_3(x)]dx = \frac{1}{4!} \int_a^b f^{(4)}(\xi)(x-a)\left(x - \frac{a+b}{2}\right)^2 (x-b)dx \quad (8-8)$$

根据定积分中值定理可知,在 $[a, b]$ 上总有一点 η 满足下述关系:

$$\int_a^b f^{(4)}(\xi)(x-a)\left(x - \frac{a+b}{2}\right)^2 (x-b)dx = f^{(4)}(\eta) \int_a^b (x-a)\left(x - \frac{a+b}{2}\right)^2 (x-b)dx$$

通过变量代换 $t=x-a$, $dt=dx$,很容易求得:

$$\int_a^b (x-a)\left(x - \frac{a+b}{2}\right)^2 (x-b)dx = -\frac{(b-a)^5}{120}$$

把这个结果代入式(8-8),便得出抛物线求积公式的误差估计式:

$$R_2(f) = \frac{1}{4!} \left[-\frac{(b-a)^5}{120} f^{(4)}(\eta) \right] = -\frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad \eta \in [a, b] \quad (8-9)$$

8.4 复合求积公式及其 MATLAB 实现

用一个多项式去逼近被积函数时,无论如何都会带来误差。由截断误差公式(8-3)可知,为了减小误差可以采取两种方法:一是增加多项式次数 n ,二是减小积分区间 $[a, b]$ 的宽度。但是增大 n 会带来计算上的麻烦,而且 n 最大不能超过 8,这样就只能在减小积分区间宽度上想办法。于是便产生了复合求积法的思想,即把积分区间 $[a, b]$ 分成若干个子区间,在每个子区间上使用低次牛顿-柯特斯求积公式,各段相加就可得出整个区间上的所谓复合数值求积公式。

8.4.1 复合矩形求积法及其 MATLAB 实现

1. 复合矩形法求定积分的原理

定积分的几何意义是计算曲边梯形的面积,若将积分区间 $[a, b]$ 分为 n 段,每段都是一个小的曲边梯形,用一个个小矩形代替这些小曲边梯形,然后把所有小矩形的面积加起来就近似地等于整个曲边梯形的面积,于是便求出了定积分的近似值,这就是复合矩形法求定积分的基本原理。

设第 i 个小矩形的宽度 $h_i = x_i - x_{i-1}$ ($i = 1, 2, \dots, n$),高度 $y_i = f(x_i)$,根据式(8-4)可知其面积为 $(x_i - x_{i-1})f(x_i) = h_i y_i$,于是可得出复合矩形求积公式:

$$\int_a^b f(x)dx = I(f) \approx y_1 h_1 + y_2 h_2 + \dots + y_n h_n = \sum_{i=1}^n y_i h_i \quad (8-10)$$

如果积分区间 $[a, b]$ 被等分为 n 段: $a = x_0 < x_1 < \dots < x_n = b$, $(x_i - x_{i-1}) = (b - a)/n = h$,则:

$$\int_a^b f(x)dx = I(f) \approx \frac{b-a}{n} \sum_{i=1}^n y_i = h \sum_{i=1}^n y_i \quad (8-11)$$

2. MATLAB 中的求和指令 sum

MATLAB 中的求和指令 `sum` 就可以完成式(8-10)的计算任务,调用格式为:

$$s = \text{sum}(y)$$

① 输入参量 y 是向量 $y = \{y_j\}_{1 \times n}$ 时,则输出参量 s 为向量 y 各分量的累加和: $s =$

$$\text{sum}(y) = \sum_{i=1}^n y_i = y_1 + y_2 + \dots + y_n。$$

② 当输入参量 y 是矩阵 $Y = \{y_{ij}\}_{m \times n}$ 时,输出参量 s 为一个行矩阵 $s = \{s_j\}_{1 \times n}$, $s_j = \sum_{i=1}^m y_{ij}$,

即 s 各元素为矩阵 Y 各对应列的所有元素和: $s = \text{sum}(Y) = (\sum_{i=1}^m y_{i1}, \sum_{i=1}^m y_{i2}, \dots, \sum_{i=1}^m y_{in})。$

例 8-8 计算 $\sum_{n=1}^{31} n^2$ 。

解 键入:

```
n = 1:31; a = sum(n.^2); a
```

回车得出：

```
a =
10416
```

3. 用 sum 指令实现复合矩形法求积计算

① 用复合矩形求积公式计算 $\int_a^b f(x)dx$ 的近似值时, 先将区间 $[a, b]$ 分成 n 段, 第 i 个节点上的函数值 $y_i = f(x_i)$ ($i = 0, 1, 2, \dots, n$), 第 i 段长度 $h_i = x_i - x_{i-1}$ ($i = 1, 2, \dots, n$), 则定积分

$$\int_a^b f(x)dx \approx \sum_{i=1}^n y_i h_i = s = \text{sum}(y * h) \quad (\text{去掉了节点 } y_0 \text{ 的值})$$

- 输入参量 y 和 h 都是向量, 它们之间的乘法运算得用数组乘法符号连接;
- $\text{length}(y) = \text{length}(h) + 1$, 上式中必须去掉行矩阵 y 的起始(或最后)一个分量。

② 通常总是将区间 $[a, b]$ 等分为 n 段, 每段长均为 $h = x_i - x_{i-1}$ ($i = 1, 2, \dots, n$), 这时

$$\int_a^b f(x)dx \approx \frac{b-a}{n} \sum_{i=1}^n y_i = s = h * \text{sum}(y) \quad \text{这里取 } i=1 \text{ 到 } i=n, \text{ 去掉了起始分量 } y_0。$$

例 8-9 用复合矩形法计算 $s = \int_0^{3\pi} y(t)dt$, $y(t) = e^{-0.6t} \sin(t + \pi/3)$, 给出 16 位浮点数。

解 键入：

```
format long; h=pi/1000; t=h:h:3*pi; % 去掉了 t 的第一个分量
y=exp(-0.6*t).*sin(t+pi/3); s=h*sum(y)
```

回车得出：

```
s =
7.509763339880702e-001
```

若将 $t = h:h:3\pi$ 改成 $t = 0:h:3\pi-h$ 则 $s = 7.537065566765496e-001$, 但不得取 $t = 0:h:3\pi$ 。用矩形法计算时, 改变 h 的取值 s 将会发生变化, h 越小 s 越精确。

8.4.2 复合梯形求积法及其 MATLAB 实现

1. 复合梯形法求定积分的原理

将积分区间 $[a, b]$ 分为 n 段, 用直线依次连接各分点, 每段都形成一个小的直边梯形。用这些小直边梯形面积之和代替原来的小曲边梯形面积之和, 就可求得定积分的近似值。设第 i 个小梯形的宽度 $h_i = x_i - x_{i-1}$, 两底高度分别为 $y_{i-1} = f(x_{i-1})$ 和 $y_i = f(x_i)$ ($i = 1, 2, \dots, n$), 则由式(8-5)可知, 定积分的近似值为：

$$\int_a^b f(x)dx = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} f(x)dx \approx \sum_{i=1}^n \frac{1}{2} (y_{i-1} + y_i) h_i$$

$$= \frac{1}{2} [(y_0 + y_1)h_1 + (y_1 + y_2)h_2 + \dots + (y_{n-2} + y_{n-1})h_{n-1} + (y_{n-1} + y_n)h_n]$$

设被积函数 $f(x)$ 在 $[a, b]$ 上连续二阶可导, 把 $[a, b]$ 区间 n 等分, 令 $h = (b - a)/n$, 于是:

$$\int_a^b f(x)dx \approx \frac{h}{2}(y_0 + 2y_1 + 2y_2 + \dots + 2y_{n-2} + 2y_{n-1} + y_n) = \frac{h}{2}(y_0 + y_n + 2 \sum_{j=1}^{n-1} y_j)$$

由于式中 $y_0 = f(a)$, $y_n = f(b)$, $y_j = f(x_j) = f(a + jh)$, 代入上式则得出复合梯形求积公式:

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(a) + f(b) + 2 \sum_{j=1}^{n-1} f(a + jh)) = T_n \quad (8-12)$$

T_n 表示区间分为 n 等份时, 用复合梯形法求出的定积分值。

2. 复合梯形求积公式的误差

复合梯形求积公式的推导中, 据式(8-6)可以知道, 在每个子区间上都会带来一定的误差 $-\frac{h^3}{12}f''(\xi_j)$, 相加后的总误差为:

$$R_1(f, T_n) = \int_a^b f(x)dx - T_n = -\frac{h^3}{12} \sum_{j=1}^n f''(\xi_j)$$

由于 $f''(x)$ 在 $[a, b]$ 上连续, 故若用 $\overline{f''}$ 表示函数 $f(x)$ 在 $[a, b]$ 上各分节点处二阶导数的平均值, 即令 $\overline{f''} = \frac{1}{n} \sum_{j=1}^n f''(\xi_j)$, 代入上式就得到复合梯形求积公式的误差估计式:

$$R_1(f, T_n) = -\frac{nh^3}{12} \overline{f''} = -\frac{(b-a)h^2}{12} \overline{f''} \quad (8-13)$$

3. 用 trapz 实现复合梯形法求积计算

MATLAB 中求梯形面积指令, 可以用于实现复合梯形求积的计算。

1) MATLAB 中计算梯形和的指令 trapz

trapz 是英语梯形 trapeze 的简缩, 它的调用格式为:

$$s = \text{trapz}(x, y)$$

① 当 x, y 是同维向量 $y_i = f(x_i)$ ($i=0, 1, 2, \dots, n$) 时,

$$s = \text{trapz}(x, y) = \sum_{i=1}^n \frac{y_i + y_{i-1}}{2} (x_i - x_{i-1})$$

② 当 x 是 m 行的列向量(列阵), 而 y 是 $m \times n$ 维矩阵时, 输出结果 s 是一个 n 列的行矩阵, 每个元素等于 y 中与之对应的列与 x 进行上式运算的结果。

③ 使用 trapz 指令而缺省参数 x 时, 表示 x 被等分, 每份宽均为 $h_i = x_i - x_{i-1} = 1$ 。

$$s = \text{trapz}(y) = \sum_{i=1}^n \frac{y_i + y_{i-1}}{2}$$

2) 用 trapz 实现复合梯形求积公式的计算

① 积分区间 $[a, b]$ 分成 n 段, 取 $a = x_0$ 时第 i 个节点上的函数值为 $y_i = f(x_i)$ ($i=1, 2, \dots, n$), 第 i 段长为 $h_i = x_i - x_{i-1}$ ($i=1, 2, \dots, n$), 则定积分:

$$\int_a^b f(x)dx \approx T = \sum_{i=1}^n \frac{y_i + y_{i-1}}{2} h_i = s = \text{trapz}(x, y)$$

使用时要求 $\text{length}(x) = \text{length}(y) = n + 1$ 。

② 如果 $[a, b]$ 是被等分为 n 段的, 每段长均为 $h_i = h$, 其使用格式为:

$$s = \text{trapz}(y) * h$$

例 8-10 用复合梯形法计算例 8-9 中的积分。

解 键入:

```
h=pi/1000;t=0:h:3*pi; % 与例 8-9 中 t=h:h:3*pi 不同
y=exp(-0.6*t).*sin(t+pi/3);
s=h*trapz(y)
```

回车得出:

$$s = \\ 0.7523$$

8.5 变步长复合求积及其 MATLAB 实现

8.5.1 复合抛物线求积公式

1. 复合抛物线求积公式推导

复合抛物线法就是把积分区间 $[a, b]$ 分成若干个子区间, 在每个子区间上使用抛物线求积公式, 然后各段相加得出整个积分区间上的数值求积公式。设被积函数 $f(x)$ 在 $[a, b]$ 上连续 4 阶可导, 因为抛物线公式要用区间中点的函数值, 所以构造复合抛物线公式时必须把区间等分为偶数份。若用 $(n-1)$ 个节点把 $[a, b]$ 等分为 n (偶数) 份, 则称 $h = (b-a)/n$ 为步长。取 $n = 2m$ (m 为正整数), 则这 m 个子区间 $[x_{2j-2}, x_{2j}]$ ($j = 1, 2, \dots, m$) 的宽度都是 $2h = x_{2j} - x_{2j-2} = (b-a)/m$, 其间有三个节点 x_{2j-2} , x_{2j-1} 和 x_{2j} 。在每个子区间上用抛物线求积公式 (8-7), 把这 m 个子区间的结果相加, 就可求得定积分的近似值:

$$\int_a^b f(x)dx = \sum_{j=1}^m \int_{x_{2j-2}}^{x_{2j}} f(x)dx \approx \sum_{j=1}^m \frac{h}{3} [f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})] = S_m$$

对上式整理便可得出抛物线求积公式, 即:

$$S_m = \frac{h}{3} [f(a) + f(b) + 4 \sum_{j=1}^m f(x_{2j-1}) + 2 \sum_{j=1}^{m-1} f(x_{2j})] \quad (8-14)$$

2. 复合抛物线求积公式的误差

在每个宽度为 $2h$ 的子区间上, 使用抛物线求积公式时都会带来一定的截断误差, 按照式 (8-9), 每个子区间上的截断误差 $R(f_j) = -\frac{(2h)^5}{2 \cdot 880} f^{(4)}(\eta_j)$, $x_{2j-2} < \eta_j < x_{2j}$, ($j = 1, 2, 3, \dots, m$)。 m 个子区间累加后的总误差为:

$$R(f, S_m) = -\frac{(2h)^5}{2 \cdot 880} \sum_{i=1}^m f_i^{(4)}(\eta_i) = -\frac{h^4}{180} \frac{b-a}{m} \sum_{j=1}^m f_j^{(4)}(\eta_j)$$

令 $\overline{f_m^{(4)}} = \frac{1}{m} \sum_{j=1}^m f_j^{(4)}(\eta_j)$ 表示 $f^{(4)}(x)$ 在 $a < x < b$ 间 m 个子区间上的平均值。代入上式, 可得出复合抛物线求积公式的误差估计式:

$$R(f, S_m) = -\frac{(b-a)h^4}{180} \overline{f_m^{(4)}} \quad (8-15)$$

8.5.2 变步长复合抛物线求积公式

由式(8-15)可知 h 的大小对误差有着直接的影响。实际问题中如果预先提出了对误差的要求, 如何据此来确定 h 的大小呢? 如果直接按照式(8-15)确定 h 的话, 将涉及函数的高阶导数, 一般较为困难。为此, 提出步长减半法, 即每次总是在前一次的基础上, 将区间对分、节点加密一倍, 这样不断减小积分子区间宽度来提高计算精度, 直至满足误差要求, 这样不仅避开了求函数高阶导数, 而且还可以减少计算量。根据误差要求确定步长, 不断改变步长满足误差要求的方法叫自适应法。

当前一次的基础上, 将区间对分、节点加密一倍, 把区间 $[a, b]$ 等分成偶数 n 份, 由 $n = 2m$ 知被分成 m 个子区间时, 据式(8-14)可以得出复合抛物线求积结果是:

$$S_m = \frac{h}{3} [f(a) + f(b) + 4 \sum_{j=1}^m f(x_{2j-1}) + 2 \sum_{j=1}^{m-1} f(x_{2j})]$$

式中 $f(a)$, $f(b)$ 和 $f(x_{2j})$ 都是前一次分点上的函数值, 不必重新计算。只有 $f(x_{2j-1})$ 是新分点上的函数值需要计算。对此, 可以从下面的简单例子看出:

当 $n = 2$ 时, $S_1 = \frac{h}{3} [f(a) + f(b) + 4f(\frac{a+b}{2})]$, 若将区间对分、节点加密一倍, 即让 $n = 4$

时, $S_2 = \frac{h}{6} [f(a) + f(b) + 4f(x_1) + 4f(x_3) + 2f(x_2)]$, 这时的 $x_2 = \frac{a+b}{2}$, $f(x_2)$ 不用重新计算, 只需计算新分点上的函数值 $f(x_1)$ 和 $f(x_3)$ 。

另外, 采用步长减半法还可以由预先要求的误差 ε 确定出步长 h 。令 $I = \int_a^b f(x)dx$ 表示积分的真值, 预先提出的误差限是 ε 。把 $[a, b]$ 分成 m 个子区间, 采用复合抛物线求积公式时, 由式(8-15)知其截断误差估计式为:

$$R(f, S_m) = -\frac{(b-a)h^4}{180} \overline{f_m^{(4)}}$$

若采用步长减半法, 把 $[a, b]$ 分成 $2m$ 个子区间, 采用复合抛物线求积公式时, 其截断误差估计式将成为:

$$R(f, S_{2m}) = -\frac{b-a}{180} \left(\frac{h}{2}\right)^4 \overline{f_m^{(4)}}$$

将上面两个 R 式相减, 可得出:

$$R(f, S_m) - R(f, S_{2m}) = S_{2m} - S_m = -\frac{b-a}{180} \left(\frac{h}{2}\right)^4 (16 \overline{f_m^{(4)}} - \overline{f_{2m}^{(4)}})$$

当 m 很大时 $\overline{f_m^{(4)}} \approx \overline{f_{2m}^{(4)}}$, 所以:

$$(S_{2m} - S_m)/15 \approx -\frac{b-a}{180} \left(\frac{h}{2}\right)^4 \overline{f_{2m}^{(4)}} = R(f, S_{2m}) = I - S_{2m}$$

若要求绝对误差为 ε , 则当 $|S_{2m} - S_m| < \varepsilon$ 时, 必有 $|R(f, S_{2m})| = |I - S_{2m}| < \varepsilon$ 。即采用步长减半法计算时, 当相邻两次运算结果之差小于 ε 就可停止计算, 认为 S_{2m} 就是满足精度要求的近似值。

8.5.3 求数值积分的指令 quad 和 quadl

在 MATLAB 中有一些求数值积分的专用指令, 掌握它们可免去编程之劳, 这里仅介绍两个常用的数值积分指令, 需要了解更多的内容可查阅“功能函数和数值积分 (funfun)”函数库。

1. quad 指令

quad 是低阶法数值积分指令, 在要求的绝对误差范围内, 用自适应递推复合抛物线法计算出数值积分, 即自动变换、选择步长, 以满足精度要求, 实现变步长复合抛物线积分的数值计算。它的调用格式为:

quad (S, a, b, tol)

- ① 输入参量 S 是被积函数, 可用三种形式: 字符表达式、内联函数或 M-函数文件名。
- ② 由于要进行数值积分计算, 函数表达式中的乘、除和幂运算必须用数组算法符号。
- ③ 输入参数 a, b 是积分限。
- ④ 输入参数 tol 是要求的计算结果绝对误差限, 省略时默认值为 $1e-6$ 。

例 8-11 计算 $\int_1^5 x^2 \sqrt{2x^2 + 3} dx$, 取最优化长格式小数。

解 由于被积函数可取三种形式, 所以可用三种方法积分。

(1) 用字符串方法

键入:

```
format long g y1 = 'x.^2.* sqrt(2*x.^2+3)'; s1 = quad(y1, 1, 5)
```

回车得出:

```
s1 =
232.805729993697
```

(2) 用内联函数法

在指令窗中键入:

```
y2 = inline('x.^2.* sqrt(2*x.^2+3)'); s2 = quad(y2, 1, 5)
```

回车得出:

```
s2 =
232.805729993697
```

(3) 用 M-函数文件法

打开编辑调试窗, 编辑名为“li8_11”的函数文件, 键入:

```
function y = li8_11(x)
```

```
y = x.^2.* sqrt(2 * x.^2 + 3);
```

将上述文件以“li8_11”为名存盘 退出编辑调试窗。

在指令窗中键入：

```
tic , s3=quad(@ li8_11 ,1 5) ,toc
```

回车得出：

```
s3 =
    232.805729993697
elapsed_time =
    0.06000000000000023
```

一旦关机 前两种方法建立的被积函数不复存在 而第三种方法不受关机影响 可以长期反复被调用。

注 指令中的 tic ,toc 是秒表计时指令 ,tic 表示秒表计时开始 ,toc 表示秒表计时结束。运行花费时间输出格式为“elapsed_time = ”,计时单位为秒。关于计时的指令还有 4 个 xclock、etime、date 和 cputime 详情查看“timefun(时间和日期)”函数库。

2. quadl 指令

在要求的绝对误差范围内 用自适应递推变步长复合 Lobatto 数值积分法 与它相应的是高阶数值积分指令 其使用方法、要求、输入参数和 quad 相同 调用格式为：

```
quadl(S,a,b,tol)
```

例 8-12 用 quadl 指令求积分 $\int_{e^{-5}}^5 \sin x^2 dx$ 结果取 14 位定点小数。

解 为了能多次调用该函数的积分 可编成 M-文件。

打开编辑调试窗 键入：

```
function y = li8_12(x)
    y = sin(x.^2);
```

将上述文件以“li8_12”为名存盘 退出编辑调试窗。

在指令窗中键入：

```
format long ,s=quadl('li8_12' ,exp(-5) 5)
```

回车得出：

```
s =
    0.52791717919279
```

对于例 8-11 和例 8-12 若用符号积分指令 int 计算 就很难得出这样理想的数值积分结果。

还有一些类似的数值积分指令如 dblquad 等 详情查看“funfun”库。

练习题 8

1. 分析下列程序中每个语句的意义 并在指令窗中输入它们 回车后观察图形窗中曲线的变化。并改变暂停指令 pause 的输入参数 观察对曲线变化的影响。

```

clf
for h=[2 1 0.5 0.1]
    x=0:h:10;y=-x.*x+112;plot(x,y),hold
    x1=0:h:10;n=length(x1);
    y1=-x1.*x1+112;
    s1=sum(y1(1:n-1))*h;
    s2=trapz(y1)*h;
    stairs(x1,y1,'r'),plot(x1,y1,'*')
    [h,s1,s2],pause(3),hold
end

```

2. 用梯形求和指令 `trapz` 实现复合梯形求积法, 计算积分 $\int_0^{\frac{\pi}{2}} \sin x dx$, 并变换分节点数, 观察对积分结果的影响。

3. 试用符号法、复合矩形法、复合梯形法和 `quad` 指令计算下列积分：

$$(1) \int_0^{\frac{\pi}{2}} \sqrt{4 - \sin^2 x} dx$$

$$(2) \int_0^1 e^{-x^2} \sqrt{1+x^2} dx$$

$$(3) \int_{-4}^4 \frac{dx}{1+x^2}$$

$$(4) \int_0^1 (x^3 - 2x^2 - 3) dx$$

$$(5) \int_1^3 \frac{1}{\ln x} dx$$

$$(6) \int_1^3 \left(\frac{\sin x^2}{x} + xe^x - 4 \right) dx$$

4. 根据表 8-2 所列函数 $y = F(x)$ 的数据, 计算积分：

$$(1) \int_{0.5}^{1.1} y^2 dx$$

$$(2) \int_{0.5}^{1.1} x^2 y dx$$

表 8-2 练习题 4 函数数据表

x	0.5	0.6	0.7	0.8	0.9	1.0	1.1
y	0.480 4	0.566 9	0.649 0	0.726 2	0.798 5	0.865 8	0.928 1

5. 计算定积分：

$$(1) \int_{0.01}^1 \frac{\ln(1+x)}{1+x^2} dx$$

$$(2) \int_{\frac{\pi}{4}}^{\frac{\pi}{3}} \frac{x}{\sin^2 x} dx$$

$$(3) \int_{-1}^2 \left[\int_{y^2}^{y+2} xy dx \right] dy$$

$$(4) \int_0^R \left[\int_0^{\sqrt{R^2-x^2}} \sqrt{R^2-x^2} dy \right] dx \quad (\text{取 } R = \text{某定值})$$

第9章 常微分方程初值问题的数值解

自然科学和工程技术的许多领域里,常会碰到常微分方程的定解问题。通常把含有自变量 x 、未知的一元函数 $y(x)$ 及其导数或微分的方程,叫作常微分方程,一般形式是:

$$G(x, y, y', \dots, y^{(m)}) = 0 \quad \text{或} \quad y^{(m)} = f(x, y, y', \dots, y^{(m-1)}) \quad (9-1)$$

方程中出现的函数最高阶导数 $y^{(m)}$ 的阶数 m 称为常微分方程的阶数。

求解微分方程就是寻找一个解函数 $y = f(x)$, 它能满足微分方程(9-1)。 m 阶常微分方程的通解含有 m 个待定常数, 如果给出了 m 阶常微分方程的 m 个初始条件:

$$y(x_0) = y_0, y'(x_0) = y_1, \dots, y^{(m-1)}(x_0) = y_{m-1} \quad (9-2)$$

就可以确定出这 m 个待定常数。式(9-2)中 $y_j = y^{(j)}(x_0) = \left. \frac{d^j y(x)}{dx^j} \right|_{x=x_0} \quad (j=0, 1, 2, \dots, m-1)$,

$y^{(j)}$ 上标(j)中的数字 j 表示导数阶数。

不含待定常数的解函数, 称为常微分方程的特解。

微分方程的解函数 $y = f(x)$ 如果是解析表达式, 称为微分方程的解析解, 如果用表格法或图示法表示出函数关系 $y_i = F(x_i) (i=0, 1, \dots, n)$, 满足或近似满足微分方程(9-1)和初始条件(9-2), 就称它为微分方程初值问题的数值解。

除了少数几种类型的常微分方程可以求出解析解外, 多数情况都只能借助于数值解法得出近似解。有的常微分方程看似非常简单, 例如, 具有初始条件的一阶微分方程:

$$\begin{cases} \frac{dy(x)}{dx} + 2xy(x) = 1 \\ y(0) = 0 \end{cases} \quad (9-3)$$

很容易得到它的解为 $y(x) = e^{-x^2} \int_0^x e^{t^2} dt$ 。但是, 要得出它的最终解, 还得进行数值积分。即便解法最简单的线性常系数微分方程, 特征方程若是高次代数方程, 求根也并非易事, 有时还会把求解搞得更为复杂, 更不必说是非线性微分方程了。因此, 学习和掌握微分方程的数值解法, 是非常必要和很有实用价值的。

本章首先介绍 MATLAB 求解常微分方程的符号法, 它可以求出许多简单常微分方程的解析解。然后, 介绍一阶常微分方程数值解的基本原理, 最后介绍求解常微分方程数值解的 MATLAB 指令及其应用。

9.1 求解常微分方程的 MATLAB 符号法

用 MATLAB 语言求解常微分方程的解析解有专用指令, 用该指令可以求出常微分方程的通解和特解, 非常方便, 只是需要先把微分方程变换成符号法要求的标准形式。

9.1.1 常微分方程的 MATLAB 符号表示法

MATLAB 符号法要求微分方程作如下形式上的变换。

① 用“Dmy”表示函数 $y=f(x)$ 的 m 阶导数 $y^{(m)}=f^{(m)}(x)$ 。例如 :Dy 表示 y 对自变量的一阶导数 $\frac{dy}{dx}$ 或 $\frac{dy}{dt}$;Dmy 表示 y 对自变量的 m 阶导数 $\frac{d^m y}{dx^m}$ 或 $\frac{d^m y}{dt^m}$,式中的 D 必须得大写。据此 ,常微分方程(9-1)可写成 :

$$Dmy = F(x, y, Dy, D2y, \dots, D(m-1)y)$$

② 初始条件的写法和上述规定完全一致。因此式(9-2)可写成 :

$$y(x_0) = y_0, Dy(x_0) = y_1, \dots, D(m-1)y(x_0) = y_{m-1}$$

③ 不特别界定时 ,通常默认小写字母“t”为函数的自变量。

据上规定 ,符号法可以把具有初始条件的一阶微分方程(9-3)表示成 :

$$Dy + 2xy = 1, y(0) = 0$$

9.1.2 求解常微分方程的符号法指令 dsolve

该指令的使用格式为 :

$$[y1, y2, \dots, y12] = dsolve(a1, a2, \dots, a12)$$

① 每个输入参数 $a1, a2, \dots, a12$ 都包含三部分内容 :符号化的微分方程、初始条件和界定的自变量 ,每个部分都用单引号界定 ,两部分之间用逗号分隔 ,第一部分不得缺省。

② 当“初始条件”全部缺省或部分缺省时 ,输出含有待定常数的微分方程通解 ,待定常数的数目等于缺省的初始条件数。待定常数用 C1, C2, ... 表示。

③ 当“界定的自变量”缺省时 ,默认的自变量是小写“t”。

④ 由于每个输入参量 $ai (i=1, 2, \dots, 12)$ 中第一部分内容不限于一个微分方程 ,参量 ai 又可以多达 12 个 ,所以该指令可以用于求解常微分方程组。

⑤ 输出参量只有在求解一个常微分方程时可以缺省 ,求解常微分方程组时不得缺省 ,因为这时要输出多个函数 ,缺省将无法区分。

例 9-1 求二阶常微分方程 $\frac{d^2 y}{dt^2} + y = 1 - \frac{t^2}{\pi}$ 的通解及满足 $y(0) = 0.2, y'(0) = 0.5$ 的特解。

解 (1) 将微分方程符号化 ,写成 $D2y + y = 1 - t^2/\pi$ 。

(2) 求通解 ,键入 :

$$y = dsolve('D2y + y = 1 - t^2/\pi'),$$

回车得出 :

$$y = (-\pi - 2 + t^2)/\pi + C1 * \cos(t) + C2 * \sin(t)$$

这是二阶微分方程的通解 ,即 $y = (\pi + 2 - t^2)/\pi + C1 \cos t + C2 \sin t$,含有两个任意常数 C1

和 C2。由于方程中含有的自变量 t 正好是默认的“ t ”,所以省去了“界定的自变量”。

(3) 求特解,键入:

```
y = dsolve('D2y + y = 1 - t^2/pi', 'y(0) = 0.2, Dy(0) = 0.5')
```

回车得出:

```
y =  
- (- pi - 2 + t^2)/pi - 2/5 * (2 * pi + 5)/pi * cos(t) + 1/2 * sin  
(t)
```

这是二阶微分方程的一个特解:

$$y = \frac{1}{\pi}(\pi + 2 - t^2) + \frac{\sin t}{2} - \frac{4\pi + 10}{5\pi} \cos t$$

利用 MATLAB 画图指令,可以把它画成曲线。为此,在指令窗中键入:

```
ezplot(' - (- pi - 2 + t^2)/pi - 2/5 * (2 * pi + 5)/pi * cos(t) + 1/2 * sin(t)', [-  
3 3]), grid
```

回车,得出图 9-1。

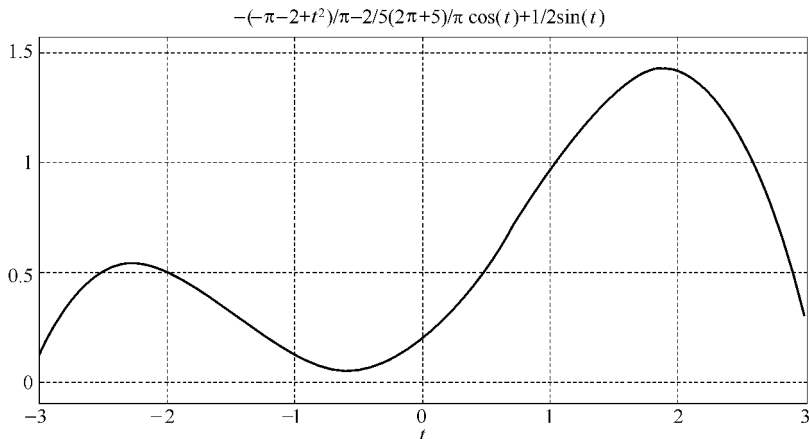


图 9-1 例 9-1 中二阶常微分方程的一条特解曲线

常微分方程的通解是一族函数曲线,而特解是一条函数曲线,这条曲线上的任何一点所对应的坐标值 t 、函数值 $y = f(t)$ 和函数在该坐标点的导数值 $y'(t)$ 满足微分方程。当然,这条曲线也满足初始条件,如图 9-1 中,曲线的起点为 $y(0) = 0.2$, $y'(0) = 0.5$ 。

例 9-2 求 $\begin{cases} \frac{du}{ds} = 3u - 2v \\ \frac{dv}{ds} + v = 2u \end{cases}$ 的通解及满足初始条件 $u(0) = 1$ 和 $v(0) = 0$ 时的特解。

解 求通解时键入:

```
[u, v] = dsolve('Du = 3 * u - 2 * v, Dv = 2 * u - v')
```

回车得出:

```
u =  
- exp(t) * (2 * t * C1 - C2 - 2 * t * C2)  
v =
```


$$- \exp(t) * (-C1 + 2 * t * C1 - 2 * t * C2)$$

这是方程组的通解。未输入界定的自变量 输出结果则自动用了小写“t”。换 t 为 s ,并用 A , B 作积分常数 略加整理 ,可得通解为：

$$u = (A + Bs)e^s, v = \frac{1}{2}(2A - B + 2Bs)e^s$$

求特解时 在输入参数中加进“初始条件” ,键入：

[u,v]=dsolve('Du=3*u-2*v,Dv=2*u-v','u(0)=1,v(0)=0','s')

回车得出：

$$\begin{aligned} u &= \\ &= \exp(s) * (-1 - 2 * s) \\ v &= \\ &= 2 * s * \exp(s) \end{aligned}$$

特解的代数表达式分别为 $u = (1 + 2s)e^s$, $v = 2se^s$ 。由于界定的自变量为“s” ,特解的函数中便以“s”作为自变量。

9.2 常微分方程数值解的基本原理

实际遇到的常微分方程 ,多数是很难找到解析解的。因此 ,必须学会用数值解法求出常微分方程的特解 ,即找出表格或图示表示的解函数 ,近似满足微分方程和初始条件。

9.2.1 求常微分方程数值解的基本原理

1. 高阶微分方程与一阶微分方程组

一般的高阶微分方程总可化为一阶微分方程组来求解。例如 m 阶微分方程初值问题：

$$\begin{cases} y^{(m)} = f(x, y, y', \dots, y^{(m-1)}) \\ y(x_0) = y_0, y'(x_0) = y_1, \dots, y^{(m-1)}(x_0) = y_{m-1} \end{cases}$$

只要引入新变量 $y_1 = y, y_2 = y', \dots, y_m = y^{(m-1)}$ 就可以变换成一阶微分方程组：

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \vdots \\ y_{m-1}' = y_m \\ y_m' = f(x, y_1, y_2, \dots, y_m) \end{cases}$$

初值条件为：

$$y_1(x_0) = y(x_0) = y_0, y_2(x_0) = y'(x_0) = y_1, \dots, y_m(x_0) = y^{(m-1)}(x_0) = y_{m-1}$$

2. 一阶微分方程数值解的基本原理

因为高阶微分方程可以变换成多个一阶微分方程构成的常微分方程组 ,所以只讨论一阶

常微分方程初值问题的数值解。

求一阶常微分方程的数值解 就是找出用表格法或图示法表示的解函数 $y_i = y(x_i)$ 。为此先把自变量离散化成 x_1, x_2, \dots 找出与其对应的函数值 y_1, y_2, \dots 如表 9-1 所示。

表 9-1 函数 $y = f(x)$ 的表格表示

x	$x_0 = a$	x_1	x_2	\dots	x_i	x_{i+1}	\dots	$x_n = b$
$y(x)$	y_0	y_1	y_2	\dots	y_i	y_{i+1}	\dots	y_n

$$\text{使它们满足下述方程: } \begin{cases} \frac{dy(x_i)}{dx} = g(x, y(x_i)), & x_i \in [a, b] \\ y(a) = y_0 \end{cases} \quad (9-4)$$

不妨设自变量 x 的节点 $x_i (i=0, 1, 2, \dots, n)$ 为 $a = x_0 < x_1 < x_2 < \dots < x_i < \dots < x_n = b$ 这些节点的分割是任意的。但为了方便起见,通常选取相邻两个节点间的距离 h 是等长的,即步长 $h = x_{i+1} - x_i$ 为定值,这时 $x_i = x_0 + ih, (i=0, 1, 2, \dots, n)$ 。

然后,从已知的 $x_0 = a, y(a) = y(x_0) = y_0$ 点出发,设法求出 x_1 点的函数近似值 $y_1 \approx y(x_1)$,再由已知的 y_0 和 y_1 求出 $y_2 \approx y(x_2)$,以此类推,由 y_0, y_1, y_2 求出 y_3, \dots 。若能找出这样的递推公式(称为计算公式),就可以求出在自变量各个节点 x_i 上满足或近似满足式(9-4)的函数值 y_i ,也就是求出了式(9-4)的数值解 $y_i \approx y(x_i) (i=0, 1, 2, \dots, n)$ 。

可见,求一阶常微分方程初值问题数值解,就是把连续性方程(9-4)的求解变成求自变量在离散节点 x_i 上的函数近似值 $y_i \approx y(x_i)$ 。解决这个问题有多种方法,常用的有泰勒展开法、数值积分法和数值微分法等。

9.2.2 泰勒展开法

1. 泰勒(Taylor)公式

如果常微分方程(9-4)中的已知函数 $g(x, y)$ 和解函数 $y = f(x)$ 是充分光滑的,可利用 Taylor 公式将欲求的函数 $y = f(x)$ 在自变量节点 x_i 上展开:

$$f(x) = f(x_i) + (x - x_i)f'(x_i) + \frac{1}{2!}(x - x_i)^2 f''(x_i) + \dots + \frac{1}{p!}(x - x_i)^p f^{(p)}(x_i) + \frac{1}{(p+1)!}(x - x_i)^{p+1} f^{(p+1)}(\xi_i)$$

其中 $\xi_i \in (x_i, x)$ 。由上式求出 x_{i+1} 点的函数值,代入 $x = x_{i+1}$ 并令 $x_{i+1} - x_i = h$ 得出:

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{1}{2!}h^2 f''(x_i) + \dots + \frac{1}{p!}h^p f^{(p)}(x_i) + \frac{1}{(p+1)!}h^{p+1} f^{(p+1)}(\xi_i)$$

利用式(9-4)中的已知关系 $\frac{dy}{dx} = f(x) = g(x, y)$, 便可得出:

$$f(x_{i+1}) = f(x_i) + hg(x_i, f(x_i)) + \frac{1}{2!}h^2 g'(x_i, f(x_i)) + \dots + \frac{1}{p!}h^p g^{(p-1)}(x_i, f(x_i)) + \frac{1}{(p+1)!}h^{p+1} g^{(p)}(\xi_i, f(\xi_i)) \quad \xi_i \in (x_i, x_{i+1}) \quad (9-5)$$

略去余项,并用 y_i 和 y_{i+1} 分别代替 $f(x_i)$ 和 $f(x_{i+1})$,则得到差分方程:

$$y_{i+1} = y_i + hg(x_i, y_i) + \frac{1}{2}h^2 g'(x_i, y_i) + \dots + \frac{1}{p!}h^p g^{(p-1)}(x_i, y_i) \quad (9-6)$$

称式(9-6)为求解一阶常微分方程初值问题(9-4)的 Taylor 公式,是一种递推公式。

把 $i=0$ 时的自变量初始值 x_0 、初始条件 $y_0 = f(a) = f(x_0)$ 及 g 的各阶导数在 (x_0, y_0) 处的取值代入式(9-6),求出 y_1 ,再把 x_1, y_1 代入式(9-6),又可以得出 y_2, \dots 。继续做下去,便可以知道在 $x_0, x_1, x_2, \dots, x_n$ 各节点上的函数值 $y_0, y_1, y_2, \dots, y_n$,这样就得到了微分方程(9-4)满足初始条件的数值解。

2. 欧拉(Euler)折线法

作为 Taylor 公式(9-6)的最简单特例,取 $p=1$,即只取式(9-6)的前两项,则得出数值解最简单形式的所谓欧拉(Euler)公式 $y_{i+1} = y_i + hg(x_i, y_i)$, $i=0, 1, 2, \dots, n$ 。将它和初始条件结合在一起,就成为求式(9-4)数值解的欧拉折线法公式:

$$\left. \begin{aligned} y(x_0) &= y_0 \\ y_{i+1} &= y_i + hg(x_i, y_i) \end{aligned} \right\} \quad (9-7)$$

欧拉公式(9-7)也可以用数值微分法由差商代替导数直接得到。它的几何意义如图9-2所示,求解式(9-4)就是找一根过点 (x_0, y_0) 的曲线 $y=f(x)$,使它上面每一点的坐标、切线斜率满足微分方程。而 Euler 法是用 x_1, x_2, \dots 节点将 X 轴等分,过点 (x_0, y_0) 做一段直线,其斜率等于点 (x_0, y_0) 处曲线的斜率,从而得到点 (x_1, y_1) ,再过点 (x_1, y_1) 做一段直线,其斜率等于曲线上 x_1 处的斜率,……如此继续下去,便得到一条靠近所求曲线的折线,用它近似代替了要寻求的函数曲线。

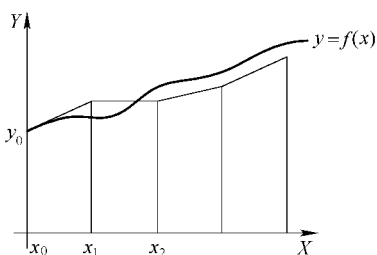


图 9-2 欧拉折线法原理示意图

显然,这种方法求得的结果误差很大。由 Taylor 公式(9-6)可以看出,如果 p 取得越大,式(9-6)中的项数就取得越多,得到的数值积分公式误差就会越小,因此它可以成为比较各种不同数值解法的标准。但是, p 增大时会带来计算 $g(x_i, y_i)$ 高阶导数的麻烦,所以提出了下边的龙格-库塔(Runge-Kutta)方法。

9.2.3 龙格-库塔法

龙格-库塔(Runge-Kutta)方法的基本原理是:用方程(9-4)中函数 $g(x, y)$ 在前一节点 x_i 上取值 $g(x_i, y_i)$ 的线性组合构造一个表示 y_{i+1} 的近似公式,从而避免了求 y_{i+1} 时用 $g(x, y)$ 的高阶导数。该方法有多种推导方法,这里用数值积分法推导。为此,先将微分方程(9-4)略加变形得出 $dy = g(x, y)dx$,对该式两边在相邻两节点 x_i 和 x_{i+1} 之间求积分,移项得出:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} g(x, y)dx \quad (9-8)$$

采用不同的方法计算式(9-8)中定积分,便可得出数值积分的不同近似结果。如果用矩

形求积公式(8-4)计算, $\int_{x_i}^{x_{i+1}} g(x, y) dx \approx (x_{i+1} - x_i)g(x_i, y_i) = hg(x_i, y_i)$, 代入式(9-8)就得出 $y_{i+1} = y_i + hg(x_i, y_i)$, 这个结果和由 Taylor 公式得出的 Euler 公式结果完全一样。

1. 二阶龙格-库塔公式

若用梯形求积公式计算式(9-8)中的积分, 则有:

$$\int_{x_i}^{x_{i+1}} g(x, y) dx \approx \frac{1}{2}h[g(x_i, y_i) + g(x_{i+1}, y_{i+1})]$$

令 $K_1 = g(x_i, y_i)$, 上式中 $g(x_{i+1}, y_{i+1})$ 里的 $x_{i+1} = x_i + h$, y_{i+1} 用 Euler 公式代换, 则可得出 $y_{i+1} = y_i + hg(x_i, y_i) = y_i + hK_1$, 把 K_1 、 x_{i+1} 和 y_{i+1} 代入上式则有:

$$\int_{x_i}^{x_{i+1}} g(x, y) dx \approx \frac{1}{2}h[K_1 + g(x_i + h, y_i + hK_1)]$$

令 $K_2 = g(x_i + h, y_i + hK_1)$, 则得出式(9-8)的一个近似结果:

$$y_{i+1} = y_i + \frac{1}{2}h(K_1 + K_2) \quad (9-9)$$

这就是二阶龙格-库塔公式, 它的局部截断误差为 $O(h^3)$ 。式(9-9)右边的函数是 K_1 和 K_2 的线性组合, 而 K_1 和 K_2 是把 x_i, y_i 和 $x_{i+1} = x + h$ 的值代入函数 $g(x, y)$ 得出的。这样计算 y_{i+1} 时不再像用 Taylor 公式那样求 $g(x, y)$ 的导数。

2. 三阶龙格-库塔公式

若用抛物线(Simpson)求积公式(8-7)计算式(9-8)中的积分, 则有:

$$\int_{x_i}^{x_{i+1}} g(x, y) dx \approx \frac{1}{6}h[g(x_i, y_i) + 4g(x_{i+1/2}, y_{i+1/2}) + g(x_{i+1}, y_{i+1})]$$

式中 $h = x_{i+1} - x_i$, $x_{i+1/2} = x_i + h/2$, $x_{i+1} = x_i + h$, 而 $y_{i+1/2}$ 和 y_{i+1} 都是未知的, 可以用 Euler 格式估算 $y_{i+1/2} = y_i + \frac{1}{2}hg(x_i, y_i)$ 和 $y_{i+1} = y_i + hg(x_i, y_i)$ 。类似于二阶龙格-库塔公式的推导, 并令:

$$K_1 = g(x_i, y_i), K_2 = g\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right), K_3 = g(x_i + h, y_i - hK_1 + 2hK_2)$$

把它们代入式(9-8)则得出三阶龙格-库塔公式, 它的局部截断误差为 $O(h^4)$ 。

$$y_{i+1} = y_i + \frac{h}{6}(K_1 + 4K_2 + K_3) \quad (9-10)$$

3. 四阶龙格-库塔公式

若用三次多项式代替(9-8)式中积分里的 $g(x, y)$ 去计算定积分, 则有:

$$\int_{x_i}^{x_{i+1}} g(x, y) dx \approx \frac{h}{8}[g(x_i, y_i) + 3g(x_{i+1/3}, y_{i+1/3}) + 3g(x_{i+2/3}, y_{i+2/3}) + g(x_{i+1}, y_{i+1})]$$

(参见表 8-1, 取 $n=3$), 式中 $x_{i+1/3} = x_i + \frac{h}{3}$, $x_{i+2/3} = x_i + \frac{2h}{3}$, 未知的 $y_{i+1/3}$ 、 $y_{i+2/3}$ 和 y_{i+1} 用 Euler 公式代换后, 一并代入式(9-8)并令:

$$K_1 = g(x_i, y_i), K_2 = g\left(x_i + \frac{1}{3}h, y_i + \frac{1}{3}K_1\right)$$

$$K_3 = g\left(x_i + \frac{2}{3}h, y_i + \frac{1}{3}K_1 + \frac{1}{3}K_2\right) \quad K_4 = g(x_i + h, y_i + K_1 - K_2 + K_3)$$

则可得出 4 阶龙格-库塔公式：

$$y_{i+1} = y_i + \frac{h}{8}(K_1 + 3K_2 + 3K_3 + K_4) \quad (9-11)$$

这是用 3/8 辛普森 (Simpson) 求积法 (积分区间被分为 3 的倍数份) 得出的结果, 它的局部截断误差为 $O(h^5)$ 。

更多使用的是基于 1/3 辛普森求积法 (积分区间被分为偶数份) 得出的 4 阶龙格-库塔公式, 它的推导方法和截断误差与式 (9-11) 相同：

$$y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \quad (9-12)$$

其中 $K_1 = g(x_i, y_i)$, $K_2 = g\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right)$, $K_3 = g\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2\right)$, $K_4 = g(x_i + h, y_i + hK_3)$ 。

还可以构造出更高阶的龙格-库塔公式, 它的一般形式是：

$$y_{i+1} = y_i + c_1K_1 + c_2K_2 + \dots + c_NK_N = y_i + \sum_{j=1}^N c_jK_j \quad (9-13)$$

其中 $K_1 = g(x_i, y_i)$, $K_j = g\left(x_i + a_jh, y_i + \sum_{m=1}^{j-1} b_{jm}K_m\right)$, $h = x_{i+1} - x_i$, $j = 2, \dots, N$

把式 (9-13) 中的 N 与式 (9-6) 中的 p 取成相同的值, 然后进行拟合, 让它们 h 同幂次项的系数相等, 确定出参数 c_j 和 a_j, b_{jm} , 从而得出 N 阶龙格-库塔公式, 根据 Taylor 级数理论可知, 它的总体截断误差是 $O(h^N)$ 。

9.2.4 阿达姆斯法

泰勒公式和龙格-库塔公式在计算 y_{i+1} 值时, 只用前一点 x_i 的 y_i 和 $g(x_i, y_i)$ 近似值, 在给定初始值后可以逐步往后推算。这种被称为一步法。此方法的优点是只要知道初值就可进行计算。但是, 要提高精度就需要增加中间函数值 (如 $y_{i+1/3}, y_{i+2/3}$ 等值) 的计算, 这就增大了计算量。下面以常用的 4 阶阿达姆斯 (Adams) 法为例介绍多步法, 这种方法在计算 y_{i+1} 的值时, 除了利用 y_i 外还要用到已经算出的 y_{i-p} ($p=1, 2, \dots$) 等, 虽然也增多预知函数值, 但都是已经算出的, 不再新增计算量。

假设已知微分方程 (9-4) 解函数 $y(x)$ 在 $x_i, x_{i-1}, x_{i-2}, x_{i-3}$ 的函数 $y(x)$ 的值 $y_i, y_{i-1}, y_{i-2}, y_{i-3}$ 。由于式 (9-8) 中被积函数 $g(x, y) = g(x, y(x))$ 表明 g 是 x 的函数, 则可用这四点构造出一个三次 Lagrange 多项式 $P_3(x)$ 近似地代替 $g(x, y)$ ：

$$\begin{aligned} P_3(x) = & \frac{(x - x_{i-1})(x - x_{i-2})(x - x_{i-3})}{(x_i - x_{i-1})(x_i - x_{i-2})(x_i - x_{i-3})} g(x_i, y(x_i)) + \\ & \frac{(x - x_i)(x - x_{i-2})(x - x_{i-3})}{(x_{i-1} - x_i)(x_{i-1} - x_{i-2})(x_{i-1} - x_{i-3})} g(x_{i-1}, y(x_{i-1})) + \\ & \frac{(x - x_i)(x - x_{i-1})(x - x_{i-3})}{(x_{i-2} - x_i)(x_{i-2} - x_{i-1})(x_{i-2} - x_{i-3})} g(x_{i-2}, y(x_{i-2})) + \end{aligned}$$

$$\frac{(x - x_i)(x - x_{i-1})(x - x_{i-2})}{(x_{i-3} - x_i)(x_{i-3} - x_{i-1})(x_{i-3} - x_{i-2})} g(x_{i-3}, y(x_{i-3}))$$

于是得出：

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} g(x, y) dx \approx y_i + \int_{x_i}^{x_{i+1}} P_3(x) dx$$

把 $P_3(x)$ 代入上式积分。这里, 仅计算 $P_3(x)$ 第 4 项的积分, 令 $g_{i-3} = g(x_{i-3}, y(x_{i-3}))$, $h = x_{i+1} - x_i$ 则:

$$\frac{(x - x_i)(x - x_{i-1})(x - x_{i-2})}{(x_{i-3} - x_i)(x_{i-3} - x_{i-1})(x_{i-3} - x_{i-2})} g(x_{i-3}, y(x_{i-3})) = \frac{(x - x_i)(x - x_i + h)(x - x_i + 2h)}{-6h^3} g_{i-3}$$

h 和 g_{i-3} 都是常量, $\int_{x_i}^{x_{i+1}} (x - x_i)(x - x_i + h)(x - x_i + 2h) dx = \frac{9}{4} h^4$, 于是得出:

$$\int_{x_i}^{x_{i+1}} \frac{(x - x_i)(x - x_{i-1})(x - x_{i-2})}{(x_{i-3} - x_i)(x_{i-3} - x_{i-1})(x_{i-3} - x_{i-2})} g(x_{i-3}, y(x_{i-3})) dx = -\frac{9}{24} h g_{i-3}$$

同样可以算出其他三项积分值, 代入 $y_{i+1} \approx y_i + \int_{x_i}^{x_{i+1}} P_3(x) dx$ 可得出阿达姆斯公式:

$$y_{i+1} \approx y_i + \frac{h}{24} (55g_i - 59g_{i-1} + 37g_{i-2} - 9g_{i-3}) \quad (9-14)$$

由于式(9-14)所取各节点都在积分区间 $[x_i, x_{i+1}]$ 之外, 故称阿达姆斯外插值公式, 局部截断误差是 $O(h^5)$ 。由于阿达姆斯外插值公式得预先知道多个点的函数取值, 所以这个方法不能独立使用, 或者说无法自启动。

若所取各节点中含有 x_i 点, 类似地, 还有称之为阿达姆斯内插值公式:

$$y_{i+1} = y_i + \frac{h}{24} (9g_{i+1} + 19g_i - 5g_{i-1} + g_{i-2}) \quad (9-15)$$

9.3 常微分方程初值问题数值解的 MATLAB 实现

在 MATLAB 中, 有多个求解常微分方程数值解的指令, 在此仅介绍两个常用的指令, 想了解其他指令时, 可用 help 调出“funfun(功能函数)库”进行查阅。

9.3.1 求常微分方程初值问题数值解的指令

ode23 和 ode45 是求解常微分方程数值解最常用的两个指令, 指令中的“ode”是英文常微分方程“Ordinary Differential Equation”的缩写, 它们都采用龙格-库塔公式进行数值求解, 23 和 45 分别表示使用的是 2/3 阶和 4/5 阶龙格-库塔公式。它们的调用格式基本相同。在此仅以 ode23 为例作如下说明。指令 ode23 的调用格式为:

$$[x, y] = \text{ode23}('Fun', Tspan, y_0, options)$$

① 该指令适用于一阶常微分方程组 $y'_j(x) = g_j(x, y_j)$, ($j=1, 2, \dots$), 如遇到高阶常微分方程, 必须先把它们变换为一阶常微分方程组, 即状态方程, 方可使用。

② 输入参数“Fun”为定义微分方程组 $\frac{dy_j}{dx} = g_j(x, y_j)$ 的 M-函数文件名, 可以在文件名前

回车将显示出自变量 t 和两个待求函数 $Y(:,1) = y(t)$ 和 $Y(:,2) = y'(t)$ 的对应数据。为节省篇幅,省写了中间的许多数据,用“.....”代替了:

```
t =
    - 2.0000
    - 1.9200
    .....
    6.8681
    7.0000
Y =
    - 5.0000    5.0000
    - 4.5852    5.3658
    .....
    - 6.8931    - 4.7083
    - 7.5759    - 5.6421
```

再键入:

```
s1 = size(t), s2 = size(Y)
```

回车得出:

```
s1 =
    42    1
s2 =
    42    2
```

表明自变量被分为 42 个节点,并计算出了对应点上的函数及其一阶导数的取值。

例 9-5 求常微分方程 $x^3 \frac{d^3 y}{dx^3} + x^2 \frac{d^2 y}{dx^2} - 4x \frac{dy}{dx} = 3x^2$ 的解,使其满足初始条件 $y(1) = 0$,

$y'(1) = -1$ 和 $y''(1) = 1$ 的解。

解 (1) 首先将三阶微分方程变换成三个一阶微分方程组。为此,令:

$$y_1 = y(x) \quad y_2 = \frac{dy(x)}{dx} = \frac{dy_1}{dx} \quad y_3 = \frac{d^2 y(x)}{dx^2} = \frac{dy_2}{dx}$$

微分方程可写成:

$$\frac{d^3 y(x)}{dx^3} = \frac{dy_3}{dx} = -y_3/x + 4y_2/x^2 + 3/x$$

于是可得微分方程组:

$$\begin{cases} \frac{dy_1}{dx} = y_2 \\ \frac{dy_2}{dx} = y_3 \\ \frac{dy_3}{dx} = -\frac{y_3}{x} + \frac{4y_2}{x^2} + \frac{3}{x} \end{cases}$$

也可写成矩阵方程：

$$\frac{dY}{dx} = \frac{d}{dx} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{4}{x^2} & -\frac{1}{x} \end{bmatrix} Y + \begin{bmatrix} 0 \\ 0 \\ \frac{3}{x} \end{bmatrix}$$

(2) 打开编辑调试窗,可用下述两种方法之一编辑微分方程组的 M-函数文件。

第一, function dy = li9_5_1(x,y)

dy=zeros(3,1); % 规定变量 dy 的维数,使用微分方程组时,不可缺少

dy(1)=y(2); % dy(1)表示 y 的一阶导数 y(2)表示 y 的第二列,即

$y'(x)$

dy(2)=y(3); % dy(2)表示 y 的二阶导数 y(3)表示 y 的第三列,即

$y''(x)$

dy(3)=-y(3)/x+4*y(2)/x^2+3/x;

以“li9_5_1”为函数文件名存盘;

第二, function dY = li9_5_2(x,Y)

dY=[0 4 0;0 0 1;0 4/x^2 -1/x]*Y+[0 0 3/x];

以“li9_5_2”为函数文件名存盘。退回到指令窗中。

(3) 在指令窗中可根据需要用下述两种方法之一求解微分方程。

第一,如果键入：

ode23('li9_5_1',[1,4],[0;-1;1]),grid

回车得出图 9-5。

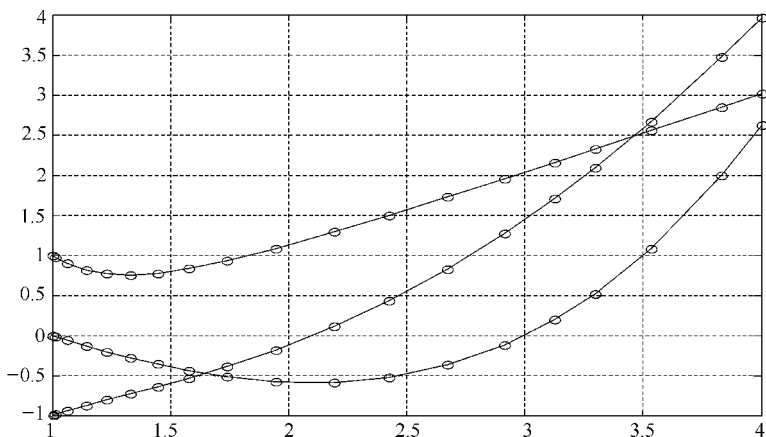


图 9-5 例 9-5 中三阶微分方程的一条特解函数及其一、二阶导函数曲线

由于没写输出变量,只输出曲线,不输出数据。

第二,如果键入：

[x,y]=ode23('li9_5_1',[1,4],[0;-1;1])

回车得出下列数据(中间省略了一些数据,变量 x,y 后括号中的内容是帮助理解加上去的):

```

x = (x_i)
    1.0000
    1.0001
    1.0005
    .....
    3.8321
    4.0000

y = (y_1 = y(x_i), y_2 = \frac{dy(x)}{dx} \Big|_{x=x_i}, y_3 = \frac{d^2y(x)}{dx^2} \Big|_{x=x_i})
    0      -1.0000      1.0000
   -0.0001  -0.9999      0.9998
   -0.0005  -0.9995      0.9990
   .....
   1.9984    3.4746      2.8491
   2.6228    3.9670      3.0148

```

键入：

```
xn = size(x), yn = size(y)
```

回车得出：

```

xn =
    22     1

yn =
    22     3

```

表明 x 有 22 个节点 $y_j(j=3)$ 为函数 $y(x)$ 、一阶导数 $y'(x)$ 和二阶导数 $y''(x)$ 在每个节点上的取值。

作为综合练习,可键入下述指令,并仔细理解每步程序的意义：

```
[x,y] = ode23('li9_5_1',[1,4],[0;-1;1]); % 得出自变量和函数的数值解
```

```
poly2str(polyfit(x,y(:,1),3),'t') % 将上述数据拟合成 3 次多项式
```

回车得出：

```

ans =
    0.14176 t^3 - 0.26304 t^2 - 0.78818 t + 0.90515

```

若用符号法求解三阶微分方程,可键入：

```

dsolve('t^3 * D3y + t^2 * D2y - 4 * t * Dy = 3 * t^2','y(1)=0,Dy(1)=-1,D2y(1)=1','t')

```

回车得出：

```

ans =
   -1/2 * t^2 - 1/6 + 1/2/t + 1/6 * t^3

```

为了绘图,键入 $t = 1:0.2:4$;

```
y10 = 0.14176 * t.^3 - 0.26304 * t.^2 - 0.78818 * t + 0.90515 * ones
(size(t));
```

```
y11 = -1/2 * t.^2 - 1/6 * ones(size(t)) + 1/2 ./ t + 1/6 * t.^3;
```

```
plot(t,y10,'0',t,y11,'r+',x,y(:,1)), legend('拟合点','数值解','符号解',2),
```

回车得出图 9-6。

练习题 9

1. 根据给出的条件求下列微分方程的解析解或数值解,并画出函数曲线:

$$(1) \frac{dy}{dx} = 2x - y, y(1) = 1 \quad x \in [1, 2] \quad (2) \frac{dy}{dx} = y - \frac{x}{y}, y(0) = 1 \quad x \in [0, 1]$$

$$(3) \frac{dy}{dx} = -xy^2, y(0) = 2 \quad x \in [0, 1] \quad (4) \frac{dy}{dx} = \frac{3y}{1+x} - y, y(0) = 1$$

$$(5) x^2 + y + (x - 2y) \frac{dy}{dx} = 0, y(0) = 1, \text{并求出 } y(1) \text{ 值};$$

$$(6) \frac{dy}{dx} = x + y^2, y(0) = 0.$$

2. 求解常微分方程组:

$$(1) \begin{cases} \frac{dx}{dt} = y \\ \frac{dy}{dt} = -x \end{cases}$$

$$(2) \begin{cases} \frac{df}{dx} = 3f + 4g \\ \frac{dg}{dx} = -4f + 3g \end{cases} \quad f(0) = 0, g(0) = 1$$

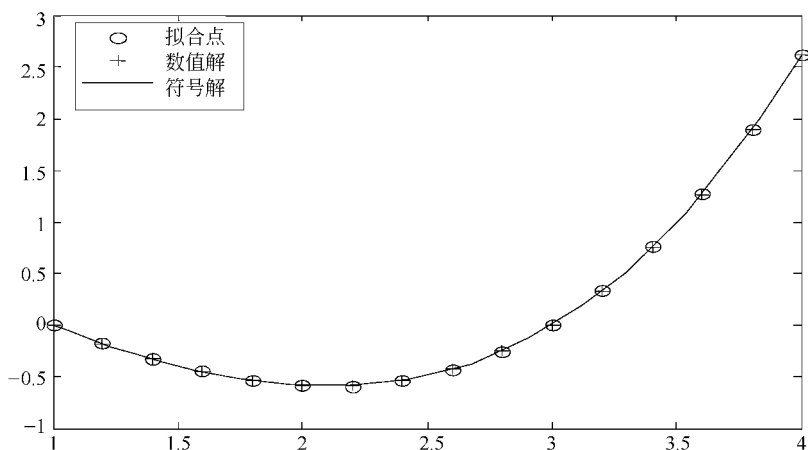


图 9-6 例 9-5 中三阶微分方程的一条特解函数曲线

$$(3) \begin{cases} \frac{d^2 x}{dt^2} + 2 \frac{dy}{dt} - x = 0 \\ \frac{dx}{dt} + y = 0 \end{cases} \quad x(0) = 1 \quad y(0) = 0$$

3. 根据所给初始条件, 选用适当方法求解下列高阶常微分方程:

$$(1) \frac{d^2 y}{dx^2} = y \quad y(0) = 0 \quad y'(0) = 1$$

$$(2) \frac{d^2 y}{dx^2} = 1 + (1 + x^2)y \quad y(0) = 1 \quad y'(0) = 3$$

$$(3) \frac{d^2 y}{dx^2} + (y^2 - 1) \frac{dy}{dx} + y = 0 \quad y(0) = 0 \quad y'(0) = 0.25$$

$$(4) \frac{d^2 y}{dx^2} + y \cos x = 0 \quad y(0) = a \quad y'(0) = 0$$

$$(5) (1 - x^2) \frac{d^2 y}{dx^2} - 2x \frac{dy}{dx} + n(n-1)y = 0 \quad n \text{ 为常数}$$

$$(6) \frac{d^3 y}{dx^3} = -y \quad y(0) = 1, \quad y'(0) = 0 \quad y''(0) = 0$$

$$(7) x^3 \frac{d^3 y}{dx^3} - 2 \frac{d^2 y}{dx^2} - 3 \frac{dy}{dx} = 3e^{2x} \quad y(0) = 1 \quad y'(0) = 10 \quad y''(0) = 30 \quad x \in [0, 1.5]$$

4. 将例 9-5 的矩阵方程编写成 M-函数文件, 用 ode23 指令求解。

5. 用符号法求解例 9-5, 并作出 $x - y(x)$ 的函数图 ($x \in [1, 4]$)。

附录 A MATLAB – 7 内容列表

英 文 名 称	中 文 名 称	版 次
MATLAB	矩阵实验室	7.0.1
Simulink	仿真	6.1
Aerospace Blockset	太空模块	1.6.1
Bioinformatics Toolbox	生物信息工具箱	1.1.1
CDMA Reference Blockset	码分多址参数模块	1.1
Communications Blockset	通信模块	3.0.1
Communications Toolbox	通信工具箱	3.0.1
Control System Toolbox	控制系统工具箱	6.1
Curve Fitting Toolbox	曲线拟合工具箱	1.1.2
Data Acquisition Toolbox	数据获取工具箱	2.5.1
Database Toolbox	数据库工具箱	3.0.1
Data feed Toolbox	数据供给工具箱	1.6
Embedded Target for Infineon C166 Microcontrollers	Infineon C166 微控制器嵌入目标	1.1.1
Embedded Target for Motorola HC12	摩托罗拉 HC12 的嵌入目标	1.1.1
Embedded Target for Motorola MPC555	摩托罗拉 MPC555 的嵌入目标	2.0.1
Embedded Target for OSEK VDX	OSEK VDX 嵌入目标	1.1.1
Embedded Target for TI C2000 DSP(tm)	TI C2000 DSP(tm)嵌入目标	1.1.1
Embedded Target for TI C6000 DSP(tm)	TI C6000 DSP(tm)嵌入目标	2.2.1
Excel Link	优化链接	2.2.1
Extended Symbolic Math	扩展符号数学	3.1.1
Filter Design HDL Coder	滤波器设计 HDL 编码器	1.1
Filter Design Toolbox	滤波器设计工具箱	3.1
Financial Derivatives Toolbox	金融衍生工具箱	3.0.1
Financial Time Series Toolbox	财经时序工具箱	2.1.1
Financial Toolbox	财经工具箱	2.4.2
Fixed-Income Toolbox	固定输入工具箱	1.1
Fixed-Point Toolbox	定点数工具箱	1.1
Fuzzy Logic Toolbox	模糊逻辑工具箱	2.1.1
GARCH Toolbox	GARCH 工具箱	2.0.1
Gauges Blockset	Gauges 模块	2.0
Genetic Algorithm Direct Search Toolbox	遗传算法直接搜索工具箱	1.7
Image Processing Toolbox	图像处理工具箱	3.1
Instrument Control Toolbox	仪表控制工具箱	2.1
Link for Code Composer Studio	编码复合工作室链接	1.3.2
Link for ModelSim	模型仿真链接	1.2
MATLAB Builder for COM	COM 的 MATLAB 生成器	1.1.2

续表

英 文 名 称	中 文 名 称	版 次
MATLAB Builder for Excel	Excel 的 MATLAB 生成器	1.2.2
MATLAB Compiler	MATLAB 编译器	4.1
MATLAB Report Generator	MATLAB 报告发生器	2.1.1
MATLAB Web Server	MATLAB 网络服务器	1.2.3
Mappings Toolbox	地图工具箱	2.0.3
Model Predictive Control Toolbox	模型预测控制工具箱	2.1
Model-Based Calibration Toolbox	模型校正工具箱	2.1.2
Neural Network Toolbox	神经网络工具箱	4.0.4
OPC Toolbox	OPC 工具箱	1.1.1
Optimization Toolbox	优化工具箱	3.0.1
Partial Differential Equation Toolbox	偏微分方程工具箱	1.0.6
RF Blockset	RF 模块	1.0.1
RF Toolbox	RF 工具箱	1.0.1
Real-Time Windows Target	实时 Windows 目标	2.5.1
Real-Time Workshop	实时工作空间	6.1
Real Time Workshop Embedded Coder	实时工作空间内置编码器	4.1
Robust Control Toolbox	鲁棒控制工具箱	3.0
Signal Processing Blokset	信号处理模块	6.0.1
Signal Processing Toolbox	信号处理工具箱	6.2.1
SimDriveline	仿真驱动连接	1.0.1
SimMechanics	仿真机理	2.2.1
SimPowerSystems	仿真动力系统	4.0
Simulink Accelerator	加速器仿真	6.0.1
Simulink Control Design	控制设计仿真	1.1
Simulink Fixed Point	定点控制仿真	5.0.1
Simulink Parameter Estimation	参数估计仿真	1.1
Simulink Report Generator	仿真报告生成器	2.1.1
Simulink Response Optimization	仿真响应优化	2.1
Simulink Verification and Validation	仿真确认和生效	1.0.1
Spline Toolbox	样条工具箱	3.2.1
Stateflow	状态流	6.1
Stateflow Coder	状态流编码器	6.1
Statistics Toolbox	统计工具箱	5.0.1
Symbolic Math Toolbox	符号数学工具箱	3.1.1
System Identification Toolbox	系统辨识工具箱	6.1
Video and Image Processing Blockset	视频和图像处理模块	1.0.1
Virtual Reality Toolbox	虚拟实现工具箱	4.0.1
Wavelet Toolbox	小波工具箱	3.0.1
xPC Target	xPC 目标	2.6.1
xPC Target Embedded Option	xPC 目标嵌入选择	2.6.1

附录 B MATLAB 指令索引

指 令	意 义	页码	指 令	意 义	页码
abs(x)	x 绝对值或模	27	diag	对角阵	13
angle(x)	产生复数 x 的相角	27	diff	求导函数	43
ans	未定义变量名	12	digits	控制运算精度	82
asin(x)	arcsin x	27	dir	显示文件及子目录名	5
atan	arctan x	27	disp	显示字符、变量	31
axis	规定坐标范围	52	doc	显示 M 文件的资料	4
bank	金融数显示	82	dot	计算点积	28
bar	条形图	55	dsolve	常微分方程符号法求解指令	169
blanks	显示空字符	31	echo	显示程序	76
box	画三维箱体	57	eig	求方阵特征值、特征向量指令	123
break	跳出包含它的最小循环	76	eps	浮点数精度(误差限)	12
cd	查询、改变路径	5	eval	转换成数值指令	34
ceil(x)	输出靠向 的整数	27	exp(x)	e^x	27
chol	矩阵 cholesky 分解	113	expand	因式展开	42
class	查验变量性质	37	expm(a)	e^a	23
clc	擦去指令窗显示	8	eye	产生单位阵	13
clear	清除工作空间变量	8	ezplot	隐函数绘图	53
clf	擦去图形窗显示	56	ezmesh	隐函数空间绘图	62
collect	同幂次项合并	42	ezsurf		
colormap	色彩控制	62	factor	因式分解	41
compact	紧凑标识符	3	factori-	阶乘指令	27
conj	求共轭矩阵	20	al		
conv	多项式积	144	figure	图形窗	47
cos(x)	$\cos x$	27	fill	添色图	55
cosh(x)	双曲余弦 $\cosh x$	27	findsym	查询变量	44
cot(x)	$\cot x$	27	fix(x)	输出靠近零的整数	27
cross	计算叉积	8	fliplr	左右翻转矩阵	14
Ctrl + C	强行中止运行指令	4	flipud	上下翻转矩阵	14
Ctrl + K	删除光标到行尾的内容	4	floor(x)	输出靠向 - 的整数	27
deconv	多项式商	144	format	输出格式指令	3
demos	范例与演示	9	fplot	函数绘图	52
det	计算方阵行列式值	123	fsolve	方程组求解	96
			funm	任意矩阵函数	23

续表

指 令	意 义	页码	指 令	意 义	页码
fzero	求函数零点	93	mldivide	矩阵左除指令	22
global	定义全局变量	67	mpower	方阵 a 的 a^n 运算	20
grid	加画网格线	56	mrdivide	矩阵右除指令	22
help	在线帮助	4	mtimes	矩阵乘积指令	20
hex	十六进制显示	82	NaN	不定值	12
hold	保留图线	56	norm	求范数	127
imag(x)	取 x 的虚部	27	null	矩阵零空间	106
INF 或 -Inf	正无穷大	12	ode23 , ode45	常微分方程数值解	176
inline	内联函数	33	ones	产生全 1 矩阵	13
input	键盘输入指令	75	path	设置或显示搜索路径	5
int	符号积分指令	151	pause	暂停程序运行	75
interp1	一元函数插值	139	pchip	分段三次插值指令	139
inv	求逆阵	22	pi	常数 π	12
i ,j	虚数单位	12	pie	画饼状图	55
keyboard	人机对话切换	75	pinv	求矩阵伪逆阵	22
legend	标注线型	56	plot	画图	48
length	查验向量维数	24	plot3	三维绘图	57
limit	求极限	43	plotyy	画双纵坐标图	55
linspace	创建等差数向量	15	polar	极坐标绘图	55
load	显示变量内容	8	poly	方阵特征多项式、根值多项式	143
log 10 (x)	$\log 10x$	27	poly2str	向量转成多项式	143
log 2 (x)	$\log 2x$	27	polyder	多项式求导	144
log (x)	$\ln x$	27	polyfit	多项式拟合	147
loglog	对数坐标图	55	polyval	求多项式值	145
logm(a)	$\ln(a)$ a 为方阵	23	pow2 (x)	2^x	27
logspace	创建等比数向量	16	prod (m : n)	$m(m+1) \dots (n-1)n$	27
long	定点数显示	82	qr	矩阵正交三角分解	125
lookfor	用关键词检索	4	quad	低阶法数值积分	165
loose	稀疏元素标识符	3	quadl	数值积分	166
lu	矩阵三角分解	111	rand	产生随机矩阵	13
magic	产生魔方阵	13	randn	产生随机矩阵	13
max(x)	x 各列元素中最大值	27	rank	矩阵秩	106
mean(x)	x 各列元素之平均值	27	rat	近似有理数(分数)显示	82
median (x)	x 各列元素的中间值	27	real (x)	取 x 的实部	27
mesh	空间网线图	58	rem (a , b)	输出 a 被 b 整除后得到的余数	27
meshgrid	投影平面分格	59	mod(a,b)		27

续表

指 令	意 义	页码	指 令	意 义	页码
reshape	变换矩阵结构	14	surf	空间曲面图	61
roots	代数方程求根	93	syms	定义符号变量	34
rot90	矩阵逆时针转 90°	14	symsum	级数求和	44
round(x)	四舍五入取整	27	sym	定义符号量、表达式	35
save	存储指令窗中变量	8	tan(x)	tan x	27
sec	sec x	27	taylor	泰勒展开	45
semilog	半对数坐标图	55	tic,toc	秒表计时指令	165
short	浮点数显示	82	title	加注图名	56
sign(x)	x 的正负号	27	trapz	梯形和	162
sin	sin x	27	tril	下三角阵	13
sinh(x)	双曲正弦 sh x	27	triu	上三角阵	13
size	查验矩阵维数	19	type	显示文件内容	4
solve	方程组符号解	86	ver	显示 MATLAB 全部内容	5
sphere	画球面指令	46	vpa	符号量转换为数值量	82
spline	三次样条插值指令	139	what	显示目录上 M-文件名	4
sprintf	格式化数据显示指令	32	which	显示文件的路径	5
sqrt(x)	\sqrt{x}	27	who	显示工作空间中的变量	4
sqrtm(a)	矩阵 a 的平方根	23	whos	显示工作空间变量资料	4
stairs	阶梯图	55	xlabel ylabel	加注坐标轴名	56
stem	脉冲图	55			
subplot	绘图窗口分割	47	zeros	创建零矩阵	13
subs	符号矩阵元素替换	39	%	程序中% 后的文字起注释和说明作用 ,不参与运行 % 后的空行起割断作用	7
sum	分量求和	27			

附录 C 部分练习题参考答案

练习题 1

6. save a :hxt6 a t y1 y2

load a :hxt6 a t y1 y2

练习题 2. 1

$$1. \mathbf{a1} * \mathbf{a2} = \begin{bmatrix} 461 & 1212 \\ 416 & 774 \\ 641 & 1662 \end{bmatrix} \quad \mathbf{a1}(:, 1:2) * \mathbf{a2} \text{ Error } \mathbf{a1}^{\wedge}2 = \begin{bmatrix} 604 & 834 & 3596 \\ 616 & 1849 & 835 \\ 762 & 780 & 5476 \end{bmatrix}$$

$$\mathbf{a1}(:). 2 = [25 \ 169 \ 81 \ 144 \ 1681 \ 36 \ 2209 \ 45041]$$

$$3. \mathbf{b1} = \begin{bmatrix} 5 & 12 & 47 \\ 13 & 41 & 2 \\ 9 & 6 & 71 \\ 9 & 15 & 21 \\ 12 & 6 & 7 \end{bmatrix} \quad \mathbf{c1} = 9 \quad \mathbf{c2} = 7 \quad \mathbf{b1}(3:4, :) * \mathbf{a2} = \begin{bmatrix} 641 & 1662 \\ 345 & 747 \end{bmatrix}$$

$$4. \mathbf{a1} + 5\mathbf{a1}' - \mathbf{E} = \begin{bmatrix} 29 & 77 & 92 \\ 73 & 245 & 32 \\ 244 & 16 & 425 \end{bmatrix}$$

$$\mathbf{a1}^3 - \text{rot90}(\mathbf{a1})^2 + 6\mathbf{E} = \begin{bmatrix} 43644 & 61919 & 281384 \\ 33546 & 86434 & 90783 \\ 62798 & 73320 & 425662 \end{bmatrix}$$

$$5. \mathbf{a} = \text{fliplr}(\text{reshape}(1:2:31, 4, 4))$$

$$9. \mathbf{c}^{-4} = \begin{bmatrix} -0.0004 & 0.0006 & 0.0001 & -0.0003 \\ 0.4674 & -0.8176 & 0.2330 & 0.1172 \\ -0.9336 & 1.6335 & -0.4663 & -0.2337 \\ 0.4666 & -0.8165 & 0.2332 & 0.1167 \end{bmatrix} \times 10^{14}$$

$$(\mathbf{c}^3)^{-1} = \begin{bmatrix} 0.0004 & 0.0010 & -0.0031 & 0.0018 \\ -1.1265 & 0.5615 & 2.2565 & -1.6915 \\ 2.2518 & -1.1259 & -4.5036 & 3.3777 \\ -1.1257 & 0.5634 & 2.2502 & -1.6880 \end{bmatrix} \times 10^{15}$$

$$(3\mathbf{c} + 5\mathbf{c}^{-1})/5 = \begin{bmatrix} 0.1504 & -0.0378 & -0.3755 & 0.2630 \\ -1.9501 & 1.9500 & 1.9504 & -1.9503 \\ 3.4491 & -3.7865 & -2.7742 & 3.1116 \\ -1.6493 & 1.8743 & 1.1993 & -1.4243 \end{bmatrix} \times 10^{16}$$

$$10. \begin{bmatrix} 1.0000 & 0+9.0000i & 7.0000 \\ 0+1.0000i & 2.0000-1.0000i & 4.0000 \\ 3.0000 & 8.0000 & 8.0000+1.0000i \end{bmatrix}$$

13. 169

14. $\mathbf{a} \times \mathbf{b} = 88.4400$; $\mathbf{a} \times \mathbf{b} = -47.5996i + 60.8486j - 17.2157k$

练习题 2.2

2. $f_1(0.5) = -120.8234$ $f_1(3) = 145.9100$ $f_1(e) = 51.3375$ $f_1(1.2) = -95.9109$

$$4. \mathbf{a1}^{-1} = \begin{bmatrix} b1 & b3 \\ b2 & b4 \end{bmatrix} \text{ 其中}$$

$$b1 = -(x^2 + 5) / (-\sin(x) * x^2 - 5 * \sin(x) + \cos(x) * x^2 + 5 * \cos(x) + \log(x^2) * \exp(2 * x)) ;$$

$$b2 = \exp(2 * x) / (-\sin(x) * x^2 - 5 * \sin(x) + \cos(x) * x^2 + 5 * \cos(x) + \log(x^2) * \exp(2 * x)) ;$$

$$b3 = \log(x^2) / (-\sin(x) * x^2 - 5 * \sin(x) + \cos(x) * x^2 + 5 * \cos(x) + \log(x^2) * \exp(2 * x)) ;$$

$$b4 = (-\sin(x) + \cos(x)) / (-\sin(x) * x^2 - 5 * \sin(x) + \cos(x) * x^2 + 5 * \cos(x) + \log(x^2) * \exp(2 * x)) .$$
 $\mathbf{a1} + \mathbf{a2} =$

$$\begin{bmatrix} \sin(x), & 1 \\ \exp(2 * x) + \exp(-x), & 5 \end{bmatrix}$$

 $\mathbf{a1} \mathbf{a2} =$

$$\begin{bmatrix} (\sin(x) - \cos(x)) * \cos(x) + \log(x^2) * \exp(-x), & (\sin(x) - \cos(x)) * (1 - \log(x^2)) - \log(x^2) * x^2 \\ \exp(2 * x) * \cos(x) + (x^2 + 5) * \exp(-x), & \exp(2 * x) * (1 - \log(x^2)) - (x^2 + 5) * x^2 \end{bmatrix}$$

$$\mathbf{a1} / \mathbf{a2} = \begin{bmatrix} c1 & c3 \\ c2 & c4 \end{bmatrix} \text{ 其中}$$

$$c1 = -1 / (\cos(x) * x^2 + \exp(-x) - \log(x^2) * \exp(-x)) * (-\log(x^2) * \exp(-x) -$$

$$\sin(x) * x^2 + \cos(x) * x^2)$$

$$c2 = (\exp(-x) * x^2 + 5 * \exp(-x) + \exp(2 * x) * x^2) / (\cos(x) * x^2 + \exp(-x) - \log(x^2) * \exp(-x))$$

$$c3 = -(-\sin(x) + \cos(x) + \log(x^2) * \sin(x)) / (\cos(x) * x^2 + \exp(-x) - \log(x^2) * \exp(-x))$$

$$c4 = -(-\exp(2^* x) + \log(x^2)^* \exp(2^* x) + \cos(x)^* x^2 + 5^* \cos(x)) / (\cos(x)^* x^2 + \exp(-x) -$$

$$\log(x^2)^* \exp(-x))$$

$$6. (x+1)^3 + (x-1)^3 - 2x^3 = 6x$$

$$8. (1) dz = \cos y \cos(x \cos y) dx - x \sin y \cos(x \sin y) dy$$

$$(2) du = \frac{-xz}{(x^2+y^2)\sqrt{x^2+y^2-z^2}} dx + \frac{-yz}{(x^2+y^2)\sqrt{x^2+y^2-z^2}} dy + \frac{1}{\sqrt{x^2+y^2-z^2}} dz$$

$$9. 1968329a/127008 \quad 1/6^* (\text{signum}(a)^* \inf - a)^* \pi^2$$

$$10. 31581/20 \quad 113245/4$$

练习题 2.3

1. 该题可用 plot 指令做 注意 plot 指令的用法。

$$3. x1 = -3:0.1:0; y1 = 4.^* \cos(x1 - \pi/4).^2;$$

$$x2 = 0:0.1:5; y2 = x2./5 + 4;$$

$$y3 = 2:0.1:4; x3 = \text{zeros}(\text{size}(y3));$$

$$x4 = 5:0.1:13; y4 = x4.^* \exp(5 - x4);$$

$$\text{plot}(x1, y1, x2, y2, x3, y3, x4, y4)$$

$$4. (1) t = -10:0.1:10;$$

$$x = 3.^* \cos(t); y = 3.^* \sin(t); z = 3.^* t;$$

$$\text{plot3}(x, y, z)$$

$$5. (1) x = -4:0.1:4; y = x; [x1 \ y1] = \text{meshgrid}(x, y);$$

$$z = \sqrt{9 - x1.^2./8 - y1.^2./6};$$

$$\text{mesh}(z), \text{box}(\text{网线图})$$

$$x = -4:0.1:4; y = x; [x1 \ y1] = \text{meshgrid}(x, y);$$

$$z = \sqrt{9 - x1.^2./8 - y1.^2./6};$$

$$\text{surf}(z), \text{box}(\text{表面图})$$

练习题 2.4

$$1. a1 \leq a2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \text{ xor}(a1, \text{rot90}(a1, 2)) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$2. a1 \& (a1 - \text{inv}(a1)) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} a1 | (a1 - \text{inv}(a1)) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$3. (1) 210$$

$$(2) 2.5613 \times 10^{18}$$

$$(3) 1.8447 \times 10^{19}$$

$$5. \text{将例 2-72 循环内的语句变为 } x(:, k+1) = (a').^k;$$

6. 在文件编辑器中输入：

```
clear,
a = input('请输入 x 的值:');
if a < 1 & a > -5 f = a^2;
elseif a >= 1 & a < 10 f = 2 * a^2.17 - 1;
elseif a >= 10 f = 3 * a^2;
end; f
```

然后以名字 ti246.m 存盘,在命令窗中输入:ti246 回车,即可运行此程序。

x = -3, f = 9; x = 8, f = 181.2784; x = 54, f = 8748.

7. 89 97 101 103 107 109 113 127 131 137 139 149 151 157

8. clear;

```
a = input('输入模糊矩阵 a =');
```

```
b = input('输入模糊矩阵 b =');
```

```
if size(a,2) == size(b,1)
```

```
    for i = 1:size(a,1)
```

```
        for j = 1:size(b,2)
```

```
            c(i,j) = min(a(i,1),b(1,j));
```

```
            for k = 1:size(b,1)
```

```
                c(i,j) = max(min(a(i,k),b(k,j)),c(i,j));
```

```
        end
```

```
    end
```

```
end
```

```
else printf('a 和 b 不能相乘'),
```

```
end,c
```

练习题 3

1. 比较 $\text{re}(x) = 10/1\ 991$ $\text{re}(y) = 0.1/1\ 991$ $\text{re}(z) = 1/1\ 991$,可知 $\text{re}(y)$ 最小 y 精度最高。

$$3. \begin{bmatrix} 1.732\ 05 & 1\ 096.63 \\ -0.958\ 924 & 1.386\ 29 \end{bmatrix}$$

$$4. \frac{1}{6\ 251} - \frac{1}{6\ 252} = \frac{1}{6\ 251 \times 6\ 252}$$

$$5. (1) \frac{1}{1+3x} - \frac{1-x}{1+x} = \frac{x(3x-1)}{(1+3x)(1+x)}$$

$$(2) \sqrt{x + \frac{1}{x}} - \sqrt{x - \frac{1}{x}} = \frac{\frac{2}{x}}{\sqrt{x - \frac{1}{x}} + \sqrt{x + \frac{1}{x}}}$$

$$(3) \lg x_1 - \lg x_2 = \lg \frac{x_1}{x_2}$$

$$(4) \frac{1 - \cos 2x}{x} = \frac{2\sin^2 x}{x}$$

练习题 4

2. 用 solve 可键入 $s1 = 'x^3 - 2 * x^2 - 4 * x = 7'$, solve(s1)

用 roots 求得 $x = 3.6320$, $-0.8160 + 1.1232i$ 和 $-0.8160 - 1.1232i$

用 fzero 得 $x = 3.6320$

3. $-1.6844 + 3.4313i$, $-1.6844 - 3.4313i$, 1.3688

4. $0.51097342938856910952001397114508$

5. 0.8767

6. -0.7750 , 1.9619

7. $[\text{atan}((-1/4 + 1/4 * \sqrt{7}) / (1/4 + 1/4 * \sqrt{7}))]$

$[\text{atan}((-1/4 - 1/4 * \sqrt{7}) / (1/4 - 1/4 * \sqrt{7})) - \pi]$

8. $-0.25688136414541019066006204008599$

9. $x = 0.4366$ $y = -0.8419$ $z = 1.1878$

10. $x = 0.5991$ $y = 2.3959$ $z = 2.0050$

练习题 5

1. (1) $x1 = 1.0000$ $x2 = 2.0000$ $x3 = 0.0000$ $x4 = -1.0000$

(2) $x1 = 1$ $x2 = -1$ $x3 = 1$

(3) $x1 = -1$ $x2 = 2$ $x3 = 1 \times 10^5$

(4) $x1 = 2.359$ $x2 = -2.6033$ $x3 = 12.109$

(5) $x1 = -2.6$ $x2 = 1$ $x3 = 2$ $x4 = 3.5$

(6) $x1 = -9.629 \times 10^{13}$ $x2 = -2.826 \times 10^{14}$ $x3 = 1581 \times 10^{14}$

Results may be inaccurate. RCOND = 1.136180×10^{17} .

(7) $x1 = 0.76471$ $x2 = 0.94118$ $x3 = 1$ $x4 = 0.94118$ $x5 = 1.2353$

练习题 6

2.

题 号	行列式值	列和范数	谱范数	行和范数
(1)	1	20	16.662	20
(2)	-4	4	3.4142	4
(3)	40	11	8.4728	13
(4)	4	16	13.073	17

4. (1) $x = [-0.70711 \quad -0.70711]$, $[-0.70711 \quad 0.70711]$ $r = 2, 4$

(2) $x = [0 \quad 0 \quad 1]$, $[0.40825 \quad 0.8165 \quad -0.40825]$,

$[0.40825 \quad 0.8165 \quad -0.40825]$ $r = 2, 1, 1$

(3) $x = [0.70711 \quad -0.70711 \quad 0]$, $[0.57735 \quad 0.57735 \quad -0.57735]$,

$[0.40825 \quad 0.40825 \quad 0.8165]$ $r = -1, -1.1102 \times 10^{-15}, 9$


```
(4) x=[0.777 05   -0.579 8   0.235 37   -0.066 575   0.014 027 ],
      [0.542 49   0.429 8    -0.631 78   0.333 22    -0.103 88 ],
      [0.301 51   0.603 02   0.301 51    -0.603 02   0.301 51 ],
      [-0.103 88   -0.333 22   -0.631 78   -0.429 8    0.542 49 ],
      [0.014 027   0.066 575   0.235 37   0.579 8    0.777 05 ]
```

```
r=0.253 84 ,1.792 3 ,3 ,4.207 7 ,5.746 2
```

练习题 7

```
2. yk=interp1(x,y,xk,'nearest')=[7.04      3.4      2.52 ]
   yk=interp1(x,y,xk,'linear')=[6.663 1    3.587 2    2.377 2 ]
   yk=interp1(x,y,xk,'pchip')=[6.545 3     3.536     2.376 5 ]
   yk=interp1(x,y,xk,'spline')=[6.490 3     3.522 6     2.384 5 ]
3. yk1=interp1(x,y,xk,'linear')=[2.537 9    2.920 7    3.186    3.312
```

```
8 ]
```

```
yk2=interp1(x,y,xk,'spline')=[1.827 3    3.105 7    1.807 4    2.600
```

```
9 ]
```

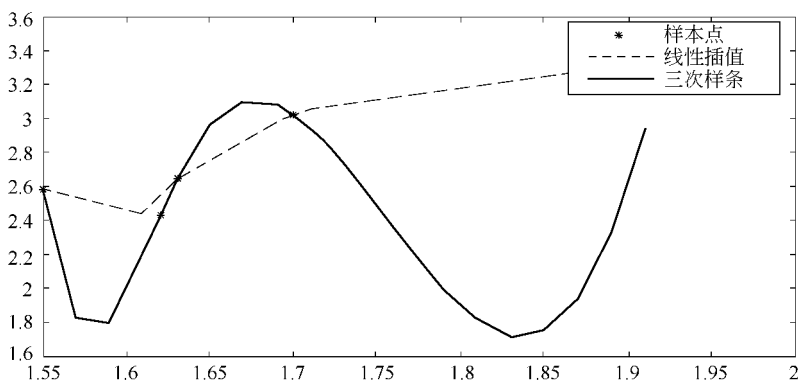
```
x1=1.55:0.02:1.92;y1=interp1(x,y,x1,'linear');
```

```
y2=interp1(x,y,x1,'spline');
```

```
plot(x,y,'* ',x1,y1,'- ',x1,y2,'r ');legend('样本点','线性插值','
```

```
三次样条')
```

得出图：



```
4.  $x^5 - 21x^4 + 151x^3 - 411x^2 + 280x$ 
```

```
5.  $5.545 3e+008 - 8.447 5e+010 - 1.952 2e+014$ 
```

```
6.  $p_1(x)p_2(x) = 819x^9 + 3465x^8 + 338x^7 - 746x^6 - 4108x^5 + 923x^4 + 7454x^3 - 765x^2 - 1785x + 945$ 
```

```
 $p_2(x)/p_1(x)$  商  $4.846 2x - 20.503$ 
```

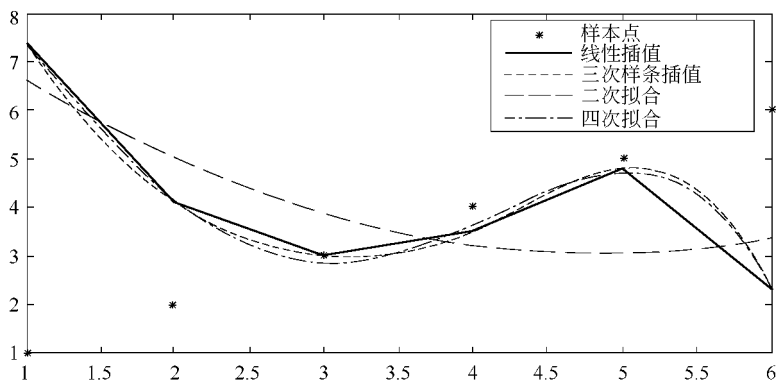
```
余  $-7.105 4e-015x^5 + 5.684 3e-014x^4 + 1153.7x^3 - 2.615 4x^2 - 392.17x + 289.53$ 
```

```
7.  $p1 = -1.688 2x + 8.075 3$ 
```

$$p2 = 0.87283x^2 - 2.6455x + 1.8247$$

$$p3 = -0.038995x^3 + 0.91746x^2 - 2.263x + 1.7692$$

8.



练习题 8

3. (1) 2.9349 ; (2) 0.8319 ; (3) 2.6516 ; (4) -3.4167 ; (5) 3.8394 ; (6) 32.5306

4. 0.0037 0.0031

5. (1) 0.2721 (2) 0.3835 (3) 45/8 (4) $R=5 \ 250/3$

练习题 9

1. (1) $2x - 2 + e^{-x}/e^{-1}$ (2) $(2 + 4x + 2e^{2x})^{1/2}/2$ (3) $2/(x^2 + 1)$ (4) $e^{-x}x^3 + 3e^{-x}x^2 + 3e^{-x}x + e^{-x}$ (5) $[x/2 - (9x^2 + 36 + 12x^3)^{1/2}/6]$ $[x/2 + (9x^2 + 36 + 12x^3)^{1/2}/6]$

(6) 在编辑器中输入：

function y1 = fun916(x,y)

y1 = y^2;

以 'fun916' 为名存盘。执行 `ode23(@fun916, [-5 5], [0])`，即可。2. (1) $u = C1\cos t + C2\sin t, v = -C1\sin t + C2\cos t$ (2) $u = e^{3t}\sin 4t, v = e^{3t}\cos 4t$ (3) $u = \cos t, v = \sin t$ 3. (1) $\sinh(x) + \cosh(x)$

(2) 在编辑器中输入：

function dy = fun932(x,y)

dy = zeros(2,1);

dy(1) = y(2);

dy(2) = -(1 - y^2) * y(1);

以 'fun932' 为名存盘。

```
ode23(@fun932,[0 5],[1 3]),grid
```

(3)(4)(5)(6)(7)均可仿照(2)方法求解。

4. 编辑器中输入：

```
function dY =fun94(x,Y)
dY =[0 1 0;0 0 1;0 4/x^2 -1/x]* Y + [0;0;1]* 3/x;
```

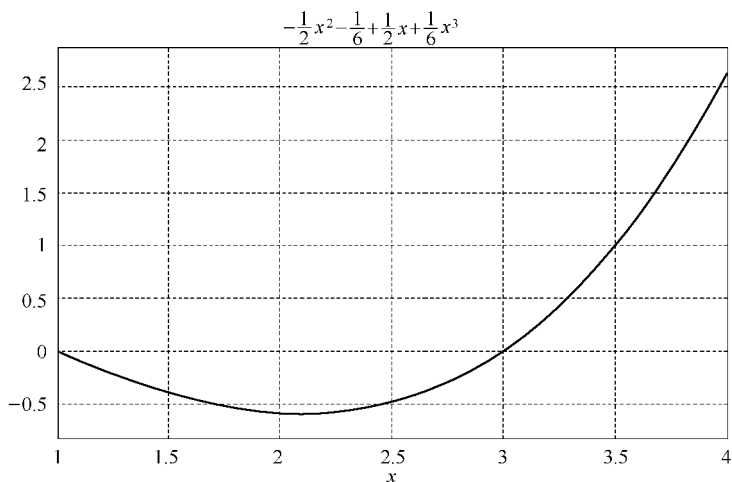
以‘fun94’为名存盘。

```
ode23(@fun94,[1 4],[0;-1;1]),grid
```

5. dsolve('x^3* D3y + x^2* D2y - 4* x* Dy = 3* x^2','y(1) = 0,Dy(1) = -1, D2y(1) = 1','x')

```
= -1/2* x^2 -1/6 +1/2/x +1/6* x^3
```

```
ezplot(' -1/2* x^2 -1/6 +1/2/x +1/6* x^3',[1 4]),grid
```



参 考 文 献

1. 陈怀琛. MATLAB 及其在理工课程中的应用指南. 西安 :西安电子科技大学出版社 2000
2. 张志涌 ,刘瑞桢 ,杨祖樱. 掌握和精通 MATLAB. 北京 :北京航空航天大学出版社 ,1997
3. 徐翠薇 ,孙绳武. 计算方法引论. 第 2 版. 北京 :高等教育出版社 2002
4. Mathews J H Fink K D. 数值方法(MATLAB 版). 陈渝 ,周璐 ,钱方译. 第 3 版. 北京 :电子工业出版社 2002
5. Nakamura S. 科学计算引论 :基于 MATLAB 的数值分析. 梁恒 ,刘晓艳译. 第 2 版. 北京 :电子工业出版社 , 2002
6. 石钟慈. 第三种科学方法 :计算机时代的科学计算. 北京 :科学出版社 2001
7. 白峰杉. 数值计算引论. 北京 :高等教育出版社 2004