

线性代数实践 及**MATLAB**入门 (第2版)

陈怀琛 龚杰民 编著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

线性代数实践及MATLAB入门 (第2版)

陈怀琛 龚杰民 编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书第1版是根据“用软件工具提高线性代数教学”的指导思想,参照美国1992—1997国家科学基金项目 ATLAST 的思路编写成的线性代数补充教材,其目的是补充我国现有教材忽视应用的缺陷。它分为两篇,第一篇介绍线性代数所用的软件工具 MATLAB 语言,可以作为教材,也可以作为手册使用;第二篇介绍线性代数实践,包括三方面的内容:一是利用 MATLAB 的可视化功能,给线性代数中的概念赋予了几何形象;二是给线性代数中烦琐的计算提供了简明的算法和程序;三是给出了各个工程和经济领域中使用线性代数建模的大量实例。本书第2版在对第1版进行修订的基础上增加了第10章,扩展了在机械和电子专业后续课程中10多个较深的矩阵建模和求解的实例。

本书既可作为大学本科线性代数的配套教材,也可作为广大理工和经管领域的教师、工程师、高年级本科生和研究生深入学习矩阵建模和掌握其计算机解法的参考读物。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

线性代数实践及 MATLAB 入门 / 陈怀琛, 龚杰民编著. 第2版. —北京: 电子工业出版社, 2009.1
ISBN 978-7-121-07223-9

I. 线… II. ①陈… ②龚… III. 线性代数—计算机辅助计算—软件包, MATLAB IV. O151.2-39

中国版本图书馆 CIP 数据核字(2008)第119504号

责任编辑: 顾慧芳

印 刷: 北京智力达印刷有限公司

装 订: 北京中新伟业印刷有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路173信箱 邮编100036

开 本: 787×1092 1/16 印张: 17.25 字数: 408千字

印 次: 2009年1月第1次印刷

印 数: 4000册 定价: 29.80元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010)88254888。

质量投诉请发邮件至 zlt@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010)88258888。

2005 年,作者根据十年来编书中使用矩阵的体会和美国线性代数教材改革的做法,编著了本书第 1 版。那时所写的前言,主要反映国外的经验,希望国内效仿。交稿以后,作者在西安电子科技大学申请了“用软件工具提高线性代数教学水平”的教改基金项目,举办了一个由 40 多位教师参加的培训班,组织几位教师进行了连续三届共 800 多名学生参加的教学改革试点,此项目后来又得到了教育部理工科处及数学教指委数学基础课程分教指委的支持,并在 2008 年 5 月进行了鉴定。这第 2 版的序言就着重介绍近三年来我们教改的经历和体会。

教改的基本指导原则是两条:一是“需求牵引,面向应用”,根据对机械和电子专业后续课大量应用的分析,提出本课程的目标是能解 6 阶以上的线性代数问题;二是“技术推动,引入机算”,借现代化手段之助,做到抽象与形象的结合,笔算与机算的结合,基础课和专业课的结合。我们具体进行了以下几方面的工作。

一、对课程的教学要求进行了全局的论证

四个现代化对教育现代化的要求首先表现在对专业课要求日益扩展和加深,再由专业课反映到基础课,促进整个教学计划的改革创新。要保证高的教学水平,必须经常对这条需求链进行论证,国外大学经常进行的 ABET 论证就包括这个内容。遗憾的是,没有见到国内对线性代数课程做过这样的论证,似乎无人关心课程内容该如何满足专业课的需要。我们在进行这项工作时,以量大面广的机械、电子专业为对象,分析其后续课程在矩阵建模和计算方面的需求,以确定线性代数课程的任务。

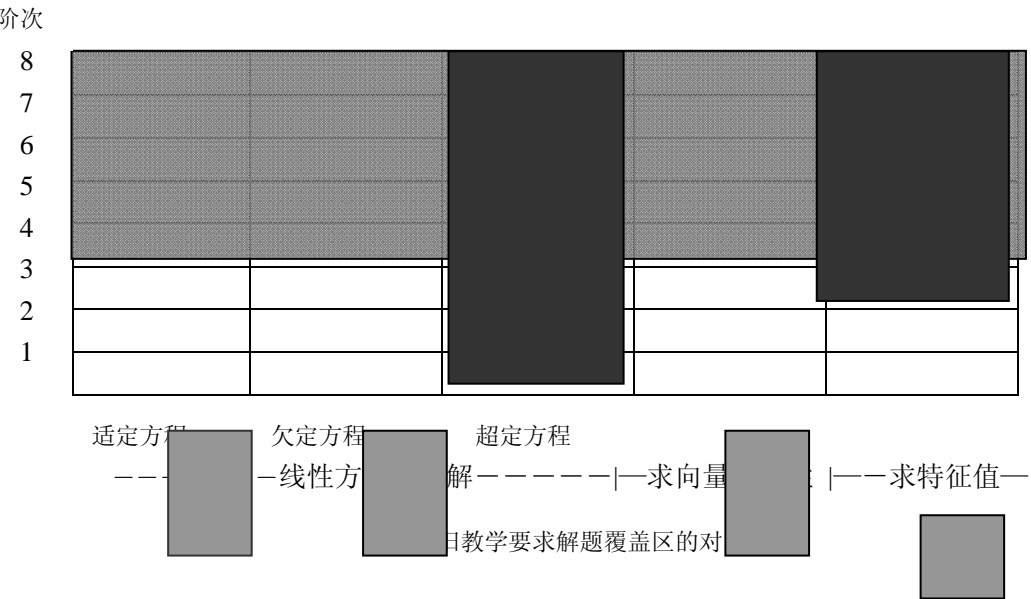
根据十多年来把科学计算用于多门课程的编书实践,我们找到了这两个专业大学三年级前能用到线性方程的十多门课程:化学、高等数学、电路、理论力学、材料力学、计算方法、传热学、物理、计算机图形学、信号与系统、数字信号处理、机械振动、机器人学等(其用矩阵建模和计算的例题都已列入本版书中)。但在实际教材中,这些课程都基本上不用矩阵计算,原因在于目前线性代数所教的内容与后续课的需求脱节。

脱节的第一个表现是阶次 N (指方程数和变量数中的大者),化学方程配平的阶次是反应式前后物质的总数,每边三种物质, N 就是 6。静力学中空间单物体平衡有 6 个方程,多一个物体,方程就加倍。电路图的节点数就对应于方程数,可见大学课程需要的 N 至少是 6 以上,而工程实践中将达到几百和几千。第二个表现是方程组类型,在物理实验和各种测量学中,都使用冗余数据提高精度,遇到的往往是超定方程,而现有课程多数不讲超定方程。第三个表现是数域,交流电路、信号处理遇到的往往是复数代数方程。

根据以上的分析,教改中我们将课程的实践目标定位为:在保持原有理论和实践水平的基础上,使学生学会高效地解 6 阶以上、复数、超定线性代数问题。新旧教学要求的对比可以用图来说明。图中白色部分是现在大纲的覆盖区,它一般只能解 3 阶问题,求特征值只到 2 阶。灰色部分是因计算复杂而难以笔算的,黑色区则是根本没教算法的,此外因为只限于实数问题,白色区又都缩小一半。新提出的目标是学生会解的问题能覆盖全图,

包括实数和复数方程，阶次可以扩展到几十、几百阶，从而可与后续课程实现无缝衔接。

百分之八十以上的后续课程的题目都不会落入白色区域，会做本书题目的线性代数老师大概只是凤毛麟角，所以我国线性代数的教育水平也可见一斑了。



二、关于如何引入计算软件的问题

人们为什么不愿用矩阵解题呢？因为若没有矩阵运算的工具，即使是低阶的题目，其效率也低于代入法和消去法，人们宁可用中学的解题方法。至于 $N > 4$ 的高阶复数问题，更是望洋兴叹，因此若不教工具，后续课程都不愿意用矩阵。许多学生反映：他们学了线性代数在本科就没有用过，只在考研时有用；这就根本无法体现线性代数在教学计划中的基础课地位，至于让学生懂得线性代数在科学计算中的重要地位就更谈不上了。

要达到解高阶复数方程的目标，唯一的方法是用计算机，特别是要用最优秀的软件工具。其实不仅是计算机帮助了线性代数，线性代数对科学计算的帮助也非常大。计算机比计算器的优越性主要不在于单次运算速度快，而在于它能够对海量数据进行连续的运算处理，海量数据的最好组织方法便是矩阵。例如算一个 1024 点的傅里叶变换，就要用信号数组乘一个 1024×1024 的方阵，它包含 100 多万万个数据（见本书例 10.9）。没有矩阵的概念，这么大的数据如何赋值、如何摆放都成问题。所以在学习矩阵之前很难充分发挥计算机的能力，而在学习线性代数的时候完成由计算器向计算机的转变是最合适的选择。

有一些老师对线性代数课使用计算机提出异议，其理由是学生用计算机算题，必然会放松笔算，不动脑筋，影响他们对基本概念的理解。有人甚至以禁止小学生用计算器作为论据，认为在大学一年级禁止用计算机是合理的。

持上述观点的老师，大概没有算过 6 阶以上的应用题，不知道手工做几百次乘法不许出错是什么滋味，更不知道 CAD 在现代化中的重要性。所以，怕学生“偷懒”而不教学生先进的知识，对教育和科学事业是极为有害的。历史上“新技术让人变懒”的论调不知重复了多少次，都不过证明了自己“懒”于跟进时代而已。在开放的信息社会中，我们应

该提倡大学生从网上、从全世界去寻找解决任何问题的最新技术，怎么可以封堵知识呢？从一定意义上说，人如果不想“偷懒”，就没有各种机器的发明和科技的进步。用科学方法来“偷懒”是要提倡的行为。大学如果不大力提倡学最新知识，那怎样能培养“创新人才”呢？在让学生掌握更多知识和计算技能的同时，为了避免他们囫圇吞枣，在出习题和提问的时候，要拐弯，要让他们动脑筋，不能简单抄袭，那是教学的艺术；但绝不能只教笨办法，不教新技术。否则，还有什么“三个面向”？这样培养出的学生与发达国家培养出的同档次人才来比，不是成了无知的傻瓜吗？

在我们的教学实践中，共有 800 多名学生参加了教改试点，试点的学生普遍为刚进大学就能接触到现代化工具而兴高采烈，并因自己既会笔算、又会机算而自豪。不仅他们的实践解题能力大大超越了普通班，而且他们的理论考试成绩也高于普通班。我们倒没有在理论教育上下特别的工夫，其提高的原因可能是：（1）大量的实例和形象化教学，提高了学生学习的积极性；（2）我们在课程中提倡笔算与机算相结合；（3）学生在计算上节省出的时间，有利于他们更多进行概念的思考。实践证明，怕学生偷懒的担心是多余的。

我们觉得，教育部门确实要对学生从小学到大学的科学计算能力的全程培养做出规划。防止学生上网成瘾的最好方法不是封堵，而是引导他们用计算机来学习和算题。“计算机要从娃娃抓起！”从国际上看，大一学生学会用计算机算题，无论如何是不嫌早了。

三、关于如何培养学生抽象思维能力的问题

过去线性代数课程教学中，既缺乏应用实例，用数字运算又太烦琐，因此只好把“抽象思维能力”作为课程主要的培养目标，但我们不赞成这种提法。第一是目标欠妥，工科大一新生的感性知识还很少，三维空间概念还有待建立，工程实际知识几乎空白，根本没有抽象思维的基础；老师们放着图中或书本中大量未解决的实际问题不学不教，却要去教空洞的“抽象思维”，是本末倒置、有害无益的；第二是方法不对，只讲理论，不联系实际，就能培养抽象思维能力吗？根本不行！如果学生对自己熟悉的课程领域都不会用矩阵建模解题，怎么可能指望他们对更深奥的问题进行抽象思考呢？这是人类的思维规律啊！

不管老师还是学生，都要经过大量从感性上升到理性的训练，才能培养抽象思维能力。所以我们在教课时要利用软件工具的优势，使抽象概念形象化；要大量介绍矩阵建模的实例，使学生体会到使用矩阵的优越性。事实上，见多了就会模仿。只有让学生看到线性代数在自己熟悉的各门课程中都能建模，而且解题快捷，他们就能逐渐学会用矩阵建模了。

我在 20 世纪 70 年代开始遇到矩阵，但只用它推理，没法用它算题；1995 年接触了 MATLAB，发现它在解矩阵问题方面的特殊优势，只要写出矩阵表达式，问题都可快速得解。于是我在各门课程中都尝试用矩阵建模和解题，写出了涉及十多门课程的多本教材（见参考文献[10]~[12]），并就逐渐地在前人没做过的领域使用矩阵建模了。其中具有创新价值的基础性工作是把信号流程图抽象为矩阵模型，并且用 MATLAB 求解。

目前国内外所有的信号与系统、信号处理及自动控制教材讲信号流程图时，所用的解法都是 1953 年由梅森提出的以图论为基础的公式，既没有证明、用起来又很烦琐，更无法用计算机编程。掌握了 MATLAB 工具后，我就力求把信号流程图也表现为矩阵，并终于得到了成功（见参考文献[8,9]），使得不管多复杂的连续和离散的信号流程图都可以方便地靠计算

机求解。在本书中，它反映在 8.6.3 节，8.6.4 节和 10.11 节中。这说明抽象思维既需要大量建模实践为基础，又要掌握先进的解题工具，才有动力；而且要靠长期的科研实践，不是靠一门只讲理论、不联系实际的数学课所能培养出来的。

四、理工结合和师资培训问题

工科线性代数属于工程数学，要把数学用于工程，教师必须既有坚实的数学基础，又有必要的工程知识。线性代数的教改加进了数学软件，教师必须要较好地掌握软件的编程；此外，还要有教学经验的积累。线性代数是一个量大面广的课程，每年有 100~200 万学生要上这门课，全国的线性代数教师可能多达 1~2 万人。其中有的是数学专业出身，有的是工程专业出身，要适应笔算和机算兼顾的教学要求，是要下一番工夫才能胜任的。

有人提出，数学课必须由学数学出身的人教才能教好，这是很片面的。另外，有些学校的工程系为了给本系老师争工作量，以为线性代数简单、好教，不经辅导培训就让新上岗的工科教师来应付，这也是不对的。需求和应用是学科发展的导向，工程数学要更好地服务于应用，既需要出身于数学而对工程有兴趣的老师，也需要出身于工程而对数学有兴趣的教师。他们都可以对工科数学的教改作出较大贡献。反之，出身于数学专业而对工程不感兴趣的老师，或出身于工程专业而对数学不感兴趣的老师，肯定教不好工科数学。根据我们的体会，工科线性代数教改中这两类教师的结合非常重要，绝不应该有门户之见，各自要学习对方的长处，克服自己的不足。我们新编的《工程线性代数（MATLAB 版）》教材就是在工程专业和数学专业教师合作下完成的。

国外的经验值得我们借鉴。美国关于线性代数的教改的五条 LACSG Recommendations 就是由数学专家和工程界的权威开会联合提出的。美国大学的工程数学教学工作也有些就由工程系承担，例如斯坦福大学的概率论和数理统计课程。即使由数学系教授任教的工程数学，由于大学分专业晚，而且数学教授从事与工程结合的科研项目也多，他们的工程知识普遍比中国教师强得多，这可从其线性代数教材（见参考文献[1]~[4]）所具有丰富的工程背景看出。为了推动课程改革走向正确的方向，必须加强教师培训和考核，不管出身如何，任课教师应该对工程和数学都有很大的兴趣，愿意把两者更好地结合。要根据坚实的数学基础、必要的工程知识，良好的编程能力和丰富的教学经验四方面的条件对教师进行培训、考核和遴选。

五、需要进一步探索的问题

为了便于检验教改的效果和应对考研，我们的改革是有约束条件的，那就是保持线性代数原有理论内容不变，只增加实践内容和提高解题水平。这就必然要增加课时，增加的一个学分中，MATLAB 占 4 学时，上机 10 机时（合 5 学时），线性代数实践占 6 学时。如果不算 MATLAB 的 9 学时，线性代数多用了 6 学时，其中包括多讲了超定方程解法和与计算有关的额外理论（如计算速度和精度、条件数、奇异值分解等）。

其实，线性代数理论的学时数确有减少的余地。美国的 LACSG Recommendations 建议全课面向非数学专业，突出应用性；不再强调抽象思维，只对数学系另外开课加强抽象性。有些大学干脆把工科线性代数改名为“矩阵应用”，这些措施都是为了减少原来的线性

代数课过于抽象的“数学味”。我认为,如果研究生统考的试题能同步改革,减少抽象性,突出应用性,那么理论部分减少6个学时应该是不困难的。当然这还有待于大家的探索,也需要教指委等教学指导部门和行政、考试部门的正确指导和干预。

六、教指委和本校专家对本项目的鉴定意见

2008年5月,由教育部高等学校数学与统计学教指委数学基础课程分教学指导委员会和西安电子科技大学联合组织了鉴定小组,对这个实施了多年的项目进行了验收鉴定。鉴定意见指出:

“本项目从工程技术应用的视角,审视了线性代数课程教学内容和教学方法,将工程背景、应用实例和现代科学计算软件融入了线性代数教学,符合国内外教学改革的方向和国际潮流,有助于实现“提高教育现代化水平”的目标,在国内线性代数课程教学中属于首创。

课题组编写的《线性代数实践及MATLAB入门》和《工程线性代数(MATLAB版)》两本教材,较好地体现了经典理论与现代计算手段相结合,将抽象概念形象化,使一些复杂的计算问题得以实现,激发了学生学习的兴趣,培养了解决问题的能力,提高了教学质量,为后续相关课程中应用线性代数知识打下了很好的基础。

本课题组的改革思想和取得的改革经验具有示范意义,出版的教材和教学实践在校内外产生了较大的影响,课题组举办的卓有成效的师资培训班以及所提供的程序集、课件和其他教学资料,为本项目的推广应用创造了良好的条件。

专家组高度评价了该项目组两年多来在线性代数课程教学改革中所取得的显著成果,一致认为:该项目改革理念先进,特色鲜明,具有创新性,是一项高水平的教学改革成果,具有很好的推广价值。”

我们将更好地贯彻专家组的意见,为在全国推广这一成果而努力,对本书的修订就是措施之一,也希望全国有更多的大学和老师参与。美国在全国推广“用软件工具提高线性代数教学”的项目用了六年的时间,所以不下大的力气,我国是很难在同样时间内做到这一点的。而线性代数课中是否学会了软件工具,会直接影响几乎所有后续课程现代化的进程,当然也会影响我国教育现代化的进程。

对已学过线性代数理论的人员,包括高年级学生、教师和工程技术人员等,进行在岗培训,补上线性代数实践这一课也是一个很重要的方面,不然他们就没法解决工程实际问题。为了更好地面向这些读者,在本书的第2版中我们增加了第10章,增补了一些机械和电子专业后续课中较深的线性代数应用实例,低年级大学生则可不学这一章。

本书的教学辅助资料有:第一篇有四小时的两张授课光盘可以供售;第二篇的课件bk05ppt.rar(修订前为xianxingdaishu.rar)和全部程序集dsk05n.rar可免费下载,下载网址为:<http://www.broadview.com.cn/manage/0therDownFiles/MATLABDownload/default.htm>。

七、出版说明

由于把计算机与线性代数相结合,本书在印刷排版上出现了一些新问题,需要把两者更好地融合起来。我们作了如下处理:

(1)在叙述文中,全部按原有线性代数书的排版规则,即使遇到MATLAB函数或语

句，矩阵仍用黑斜体，下标则仍用小号字，如 $[p, \textit{lamda}] = \text{eig}(A_3)$ 。

(2) 书中所有人机交互的部分等用白体。即在输入计算机的 MATLAB 完整程序段或程序行中全用正白体，如 $A1 = A3 * A4$ ，因为计算机不接受黑斜体矩阵，下标也不能用小号字。这与我们提供的下载程序集一致。程序运行后计算机显示的结果也全用正白体。如：

输入 $[p, \textit{lamda}] = \text{eig}(A3)$

得到
$$p = \begin{bmatrix} 0.4472 & -0.8944 \\ 0.8944 & 0.4472 \end{bmatrix}, \textit{lamda} = \begin{bmatrix} -7 & 0 \\ 0 & 3 \end{bmatrix}$$

(3) 除了说明矩阵阶数 $A_{m \times n}$ 之外，乘号运算符不用 \times ，统一采用 $*$ ，或完全省略，如 $A*B$ 或 AB 。

读者若有问题和建议，欢迎用电子邮件向作者提出。作者的电子邮址为：
hchchen1934@163.com 或 hchchen@xidian.edu.cn。作者在西安电子科技大学的主页地址为：
<http://see.xidian.edu.cn/faculty/hchchen/>，作者的博客网址为：
<http://www.blogcn.com/user79/chenhuaichen/index.html>

陈怀琛 2008 年 7 月于美国硅谷

陈怀琛与龚杰民两位教授所编写的《线性代数实践与 MATLAB 入门》由科学计算软件 MATLAB 入门与线性代数实践两篇共九章所组成。书末有一个附录，对美国国家科学基金项目——“用软件工具增强线性代数教学 (ATLAST)”进行了简单的介绍。

线性代数是围绕求解线性方程组而发展起来的一门学问，它的基本概念有向量、行列式、矩阵、线性变换、特征值和线性空间等，解析几何是线性方程组的几何背景。随着线性代数的发展，人们发现，使用它的基本概念，许多学科和许多数学分支中的问题有了几何意义，或者几何意义更加丰富凸显，不少深入而复杂的题目可以用简洁的形式来表述；还有，借助于符号的可比性，常常能够启发人们发现有效的求解方法，即算法。在历史上，人们曾经研究过这样一个题目：如果只用直尺，不用圆规，能够解决哪些作图题？今天，在这里，也设想一个问题：如果不准使用线性代数的概念和理论，许多学科将会变得如何的支离破碎，达不到今日的深度。所以我们说不仅理工科专业，甚至大学的几乎所有专业，线性代数是一门必修课，是一门基础课。

线性代数由理论和计算两部分所组成。

20 世纪 50 年代我国在理工科各专业开设线性代数课程时，以介绍理论部分为主。那时，人们已经认识到，线性代数有广泛的应用，但教材中往往限于讲授在二次型中的应用。这是因为当时计算机和编制相关程序的工作离我国的实际情况甚远。虽然已经认识到计算机能够快速高效地求解线性代数中的各种数字题目，但在教材中只能淡淡地指出这个方向而已。

改革开放以来，虽然提倡直接使用国外的教材（也就是说，采用国外的教学大纲），注意计算机的应用，提倡开设使用科学计算软件的数学实验课程，开设某些科学计算软件的师资培训班等，但是除了使用国外教材外，还远没有改变各个课程，线性代数课依然是一片“宁静的沃土”。

现在的科学计算软件已经发展到使用非常方便、功能异常强大，一经使用便令人惊叹不已的地步，科学计算软件已经成为科学工作者的高级计算器。实验室和编写程序的良好环境，加上我国经济迅速发展，计算机广泛普及，让大学各个专业的学生全都学会使用这些软件应该是刻不容缓的事情。

本书介绍了大量的实际应用题目，把科学计算软件和线性代数密切结合，充分利用软件的可视化功能产生的图形和动画补充了现行教材的不足。它明显地接受了美国 ATLAST 计划所产生的先进成果影响，是一本有特色的配套教材；因此，它的出版无疑是非常及时的。值得指出的是，比照美国的实践，我国原有的教材内容和教学水平应该说落后了十几年。

正在或者已经学过线性代数的人员（大学生，研究生，各方工程技术人员），定能从学习本书而加深理解线性代数和软件 MATLAB 这两门学问的知识以及它们之间联系的重要性，并从大量应用实际问题拓宽思路。本书每章末有足够练习题，读者可以从上机做实验中培养技能和兴趣，提高学习线性代数的积极性。此外，本书还可成为使用软件 MATLAB 解决有关线性代数问题的人员的上机参考手册。

我赞成线性代数理论和实践两部分由同一个教师施教，并相信讲授线性代数的教师对于本书中的各个方面的内容，例如令人深思的学术观点，有趣的历史资料，众多有用的应用题，附录中介绍的美国学者的敬业精神、集体主义和工作经验等，都会产生极大的兴趣。

使用本书时可能发生的困难有两点：

一是在增加不多的学时中，如何组织这个实验任务。按本书参考文献[1]，美国实施这门课程总共用 35 学时（他们也喊学时不够），可见理论和实际的结合可能产生事半功倍的效果，这当然有一个探索的过程。

二是少数教师可能对使用软件 MATLAB 进行教学感到困难。

我在过去二十多年的教学生涯中，曾经几次随班听课，甚至随班参加考试过高级算法语言 Pascal、C。虽然多次企盼自己能够编写某些程序，可是事情就那么困难，几个回合败下阵来，再加工作忙碌，无奈放弃，而后畏难情绪迟迟不能消去。近几年，为科研工作所迫，硬着头皮，熬！摸索三个月，算是开始能为我编制程序服务了。

科学计算软件和数学的关系非常密切。有人说大同小异。殊不知，许多时候，所编程序之所以通不过，错误就出在那些小异上。

毕竟是要进入一个崭新的学科，我们当然要认真学习；它既是一门科学，当然一定能够学会，而且那么多的人已经学会了。

今天的科学计算软件和算法语言已经大不一样了。打一个不那么恰当的比喻：改革开放初期，曾经流行过一本英语教材，叫《英语会话 900 句》。它分成若干个部分，包含各个场合所常用的句子，问路、学习、买东西，还有开会等。现在流行的科学计算软件也是这么一种模式，它们都有自己的“900 句”。由若干个函数库所组成，分别为各个任务提供种种函数和命令。当您拿起一个软件，首先按照教材中的例题，边读边在计算机上试算一些最基本的语句，以初步了解该软件的功能。当您学习线性代数时，无需全面熟悉其他各个分支的语句。随着学习的进程，每次学习四、五个语句，就能让计算机开始为您服务。当您掌握若干个语句之后，发现某些规律，学习不仅更加容易，而且延展到别的问题往往也能沿着同一思路得到解决。当您找不到现成的语句解决所提的题目，则需要组合若干基本语句来完成。为了我们的教学工作，也为了今后自身的科研工作，花一定时间来逐步掌握一两个科学计算软件，让它们成为自己的一个终生的学术助手和伙伴，无论如何都是值得的。我也是一名数学老师，即便在“熬”的日子里，也不断地从中得到许多的乐趣，现在，在我写书，算题，科研等工作的过程中，面对屏幕显示的结果，不时自言自语地惊叹说：“太好了！”深深感激科学计算软件给我的帮助。

本书作者陈怀琛教授是计算机科学、机械、电子和控制等学科的专家。具有丰富的教学实践经验和教学管理经验，对我国 21 世纪大学工科专业学生如何培养的问题，有许多很有价值的见解。作者对当前国内外的工科线性代数课程的施教情况十分关心。龚杰民教授是软件专家，二十年前就出版了关于 C 语言的教材。他们不仅亲自执笔编写这本教材，还正面提出了具体改革的见解。听说西安电子科技大学领导已经决定教改立项，将由陈教授亲自负责使用本书书稿，先对该校全体线性代数教师组织培训研讨班，再点面结合地对部分一年级大学生用本教材进行施教，有系统地开展试验，实在是一件大好的事情。

祝这项工作成功！

秦裕瑗
2005 年 中秋节
于武汉科技大学

线性代数的重要性现在比过去任何时候都更加令人刮目相看。在 20 世纪后半期，线性代数的应用继续扩大到了越来越多的新领域。它在数学课程中的角色已经上升到可与微积分相匹敌。线性代数的这种发展首先是由于人们所研究问题的规模愈来愈大，愈来愈复杂，牵涉的变量成百上千，这样复杂的问题，目前只能把变量之间的关系简化为线性才有可能求解。所以大规模的线性代数问题就成为热门的数学工具。除了上述的“需求牵引”之外，线性代数发展的另一个动力是“技术推动”，那就是计算机技术的推动。几十年来计算机硬件的飞速发展给线性代数的研究和教学提供了前所未有的空间和机遇，线性代数课程教学上的许多新面貌、新方法都来自于计算机技术的新发展。

计算机如何推动了线性代数的应用

线性代数是一门应用性很强，但又在理论上进行了高度抽象的数学学科。一方面，中学生就学过了二元一次代数方程的解法，代入法和消去法大概每个人都会记忆一辈子，这就是最简单的线性代数。当把方程的阶次提高到了三元一次以上时，它不但要求较高级的抽象思维能力，而且也要求用十分烦琐的计算步骤才能解决问题。对于数学家，他们重视前者，这无可厚非；但对于大多数工科学生，他们更需要的是能应用它的理论，指导完成实际的计算。事实上，线性代数的那种单调、机械、枯燥的运算，只是由于计算机的出现才赋予了在应用方面的生命力。

举一个典型的例子，Wassily Leontief 教授把美国的经济用 500 个变量的 500 个线性方程来描述。1949 年夏，由于当时大学的计算机（Mark II）能力所限，Leontief 把系统简化为 42 个变量的 42 个线性方程，编程并用穿孔卡输入程序和数据就用了几个月，最后计算机运行了 56 小时才求出了解。当 Leontief 在 1973 年成为诺贝尔经济学奖得主时，这项工作以“第一个有实际意义的利用计算机求解大规模数学模型”列为其得奖的理由之一。他的成就和获奖成为各国科学界用线性代数建立工程和经济模型的巨大动力，推动了这门科学的迅速发展。可以看出，离开了计算机，线性代数在工程中就很难有用武之地。这也反映在美国的大学工科教育中，表现出对这门课的日益重视；课堂上固然着重讲线性代数理论，但同时给学生加上大作业或课程设计等实践环节。大学中的大型计算机很大程度上也支持了这门课的实践环节，使用的软件主要是 FORTRAN 或 COBOL 语言。线性代数的教学不能离开计算机是美国工科教育界的共识。

20 世纪 80 年代，出现了个人计算机并迅速普及。新的硬件也带动了新的软件，出现了新颖的科学计算语言，也称为数学软件，因为它具有高效、可视化和推理能力等特点，故在大学教育和科学研究中，迅速地取代了 FORTRAN 和 BASIC 语言。这类软件中商品化的有 MATLAB、MATHEMATICA、MATHCAD、MAPLE 等，它们的功能大同小异，但

各有所长。目前在美国大学工科中，流行最广的是 MATLAB 语言。

MATLAB 是“矩阵实验室”(Matrix Laboratory)的缩写，它是一种以矩阵运算为基础的交互式程序语言，当然它特别适合于线性代数，并能更广泛地适应科学和工程计算及绘图的需求。与其他计算机语言相比，MATLAB 的特点是简捷和智能化，适应科技专业人员的思维方式和书写习惯，使得编程和调试效率大大提高。它用解释方式工作，键入程序立即得出结果，人机交互性能好，易于调试并为科技人员所乐于接受。特别是它可适应多种平台，并且随计算机硬软件的更新及时升级，因此 MATLAB 语言在国外的大学工学院中，特别是数值计算用得最频繁的电子信息类学科中，已成为每个学生必须掌握的工具。它大大提高了课程教学、解题作业、分析研究的效率。我们学习掌握 MATLAB，不仅可以直接帮助学习线性代数，而且也可以说是在科学计算工具上与国际接轨。

国内外线性代数教学的差距

从美国在线性代数教学中使用计算机的历史可以看出，个人计算机和科学计算软件的普及迅速推动了这门课程的教学方法改善，使得计算机的使用不限于大作业，也可以用于日常课程教学。1990 年，美国成立了线性代数课程研究组(Linear Algebra Curriculum Study Group-LACSG)，然后，在国家科学基金会(NSF)资助下组织了数学和工科专家的一次会议，提出了线性代数课程改革的五点建议，简称为 LACSG Recommendations (见参考文献[3])，其要点是：(1)首先要满足非数学专业面向应用的需要；(2)要以矩阵运算为基础；(3)要从学生的水平和需求出发；(4)要采用最新的软件工具；(5)对想要数学学位的学生应另开相关课程以提高其抽象性。1992 年美国国家科学基金会 (NSF) 资助了一个 ATLAST 计划，ATLAST 是 Augment The Teaching of Linear Algebra through the Use of Software Tools (用软件工具增强线性代数教学)的缩写。该计划在 1992 年到 1997 年六个暑期组织了十八个教师研讨班。共有来自各大学的 425 名教师参加。参加者接受了使用 MATLAB 软件包的训练，详情可参阅附录 B。

在使用 MATLAB 方面，从他们的教材发展来看，在 1995 年算起的头几年，主要反映在采用 MATLAB 的习题并介绍 MATLAB 入门，见参考文献[7]~[9]。到近十年就开始把 MATLAB 掺合到线性代数的各章中去，主要是对有些理论提供计算机的演示和验证，反映在参考文献[1]~[5]中。当然线性代数的整个理论体系，并不受使用计算机而有所改变。

在我国，线性代数课在理工科本科教学的加强开始于改革开放以后，是学习国外先进经验的结晶。当时大学中还没有计算机，虽然利用世行贷款，花了不少钱买了一些大型计算机，但线性代数课并没有用。因为课程内容不作改革，有计算机也用不成，当前的情况就足以为证。如果说以前是出于无奈，那么在个人计算机已经如此普及的情况下，还不用计算机，那就是固步自封了。所以线性代数课中不谈计算机、教线性代数的老师几乎不使用计算机，已经成为我国线性代数教育界与发达国家的明显差距。于是我国的线性代数课程出现了不尽如人意的状况——理论抽象愈来愈深，应用和实际计算很少结合，它成了一门学生感到抽象、冗繁而枯燥的课程。

由于缺乏感性的、实践的基础和应用的推动，后续课程又往往怕烦而避开矩阵方程，教出来的学生当然是理论上害怕矩阵、实践中不会用矩阵算题的。可以做一个测试：在学

生学完线性代数课以后，让他们解一个四元一次代数方程，看他们用什么工具解？要多少时间？做对的有多少比例？按现在的教材和教法，绝大多数学生解这个题用的完全是中学里学的方法：用计算器一个数一个数地算乘法和加法，谁也不会用线性代数去解。而且计算的效率和正确率极低。要知道，许多后续课程都需要用线性代数，比这个四元一次方程要复杂得多，解这么简单的题目还这样的少慢差费，大学工科后续课程怎么能用线性代数呢？又怎么谈得上为工科教育打好数学基础呢？

如果在课程中增加 4~6 学时的实践内容，情况就会完全不同。像上面的测试题，用计算机解，一分钟就可解决问题，正确率 100%。对复杂的问题，提高效率更为明显。通过实践不仅方便了计算，而且对理论和概念的理解也会加深，并节省很多时间。本来，线性代数的理论和实践是应该融合并在一起实施的，因为这门课的特点就应该理论与实际相结合。不过现在在我国实现这个任务似乎还相当艰巨，首先要从上到下达成共识，然后要修改教学计划，接着还要编写新的教材和培养大批合格的师资。

本书的内容安排

在我国，每年学习线性代数课程的大学生大概有 100 万人之多，教这门课的老师应该有上万人。要推动“用计算机提高线性代数教学水平”的事业，不是一两年就能做到的，美国还花了六年时间呢！现在我们的线性代数教学水平比美国已经落后了十多年，所以要奋起直追。我的建议是分两步走。

第一步是单独开设“线性代数实践课”，与线性代数同步实施。其好处是暂时不影响原来教师们的备课和教材，并且让少量的实践领头老师能集中精力，给更多的学生讲课，也培训现有的老师；这些老师也同时承担实践课的辅导任务，这有利于提高他们的计算机使用水平，为以后全面承担这门课程创造条件。第二步是把实践课与理论课合并实施，除了师资外，最主要的是编一本把理论与实践紧密结合的好教材。

在我们的方案中，实践课的计划是一个学分，按 16 学时计算。考虑到我国线性代数课程大都放在大学一年级，此前大一新生未必学过 MATLAB，而且以线性代数作为学习 MATLAB 的切入点有很大的好处，所以把线性代数实践与 MATLAB 入门合成一门课实施比较合适。初步安排讲课约 10~12 学时，其中介绍 MATLAB 语言入门约 4 学时，讲解线性代数实践原理和程序 6~8 学时，上机时间预计 10~12 小时。我们根据这样一个思路编写了这本教材。这本书虽然有实践的部分，但它是从实际应用的角度对线性代数的概念进行了整体的剖析和归纳，并与工程实践有大量的联系，其范围超出了一般的数学实验，故取名为“线性代数实践”。

本教材中 MATLAB 入门部分基本上就是参考文献[4]、[5]两本书中的语言篇，对于线性代数实践而言，主要用到的是第 2 章和第 4 章 4.2 节；虽然书的篇幅多了一些，但可以维持 MATLAB 基本函数的完整性，使这本书兼有 MATLAB 的手册功能，同时也便于利用本书作者的一套四小时讲课光盘，让老师不必花时间为 MATLAB 备课。实验课安排的时间最好在线性代数开课一个月以后，这样衔接比较好。这门课程可促进学生用计算机的经常化，故不要速战速决，以拉开到八周以上为好。

线性代数部分则是参考国外 2000 年后新出版的教材（见参考文献[1]~[4]）和 2003 年

出版的 ATLAST Manual (见参考文献[7]) 等资料编写的, 其中也利用了作者在多本著作中用矩阵建模和解决难题的实例 (见参考文献[10]~[12])。为了尽量加强与线性代数理论部分的衔接, 能帮助学生既避免烦琐刻板的四则运算, 又能真正体会到线性代数中的推理思路, 我们设计了一些简单的 MATLAB 子程序, 来完成高斯消元、行阶梯简化、行交换等任务。为了加强线性代数的几何形象教学, 我们又设计了一些快速简便绘制直线和平面图形的函数; 另外, 还采用了 ATLAST Manual 提供的某些矩阵生成子程序和演示程序。

因为全国各大学的差别很大, 例如专业不同、上课的学期不同 (大一上、大一下、大二上都有), 造成学生的基础不同, 所以书中的实例就不得不取宽一些, 并尽量避开微积分。实例并不需要全讲, 有的可留给后续课程中让学生自学, 书中的小字部分在初学时也可跳过。我们认为, 工科大学生能用计算机和 MATLAB 解线性代数方程的问题, 那么这门实践课的主要目标可以说基本达到了。

互联网联系方式

本书将在电子工业出版社博文视点公司的网址上提供本书课件、子程序、例题程序及 ATLAST Manual 子程序的免费下载。

本书由陈怀琛负责总的策划与编写, 龚杰民担任国外教材资料的翻译及部分习题的选编。由于我们还没见过同类书名的书籍和教材, 写书时很难找到可以直接参考的体系, 这个新生事物, 还缺乏实践经验, 再加上要赶上 2005 级部分新生进行试点, 编写时间紧迫。我们的想法和做法, 肯定有很多不当之处, 欢迎批评指正。更希望各方面的专家和读者通过自己的教学实践向我们提出改进的建议。我们的电子邮件地址为 hchchen@xidian.edu.cn。电话: (029) 88202988。

致谢

本书荣幸地由武汉科技大学秦裕瑗教授审阅, 作为一位在欧美 7 国 14 校进行过讲学、有多部专著的我国数学界前辈, 他不但博学, 而且其严肃认真的治学态度和不断接受新事物的进取精神给我们以很大的激励。在年逾 80 之际, 他仍在孜孜不倦地学习科学计算软件 (Mathematica 和 MATLAB) 并把它用到自己的著作《运筹学简明教程》中, 实在令人肃然起敬。与此相反, 我们看到有些年纪不过四五十岁的中年教师, 已经不想学计算机了。在这里, 我们特别希望广大的线性代数课老师, 能以秦教授为榜样, 把自己用科学计算语言武装起来, 尽快把我国的线性代数课程用计算机武装起来, 创造一个崭新的教学局面。

作 者

2005-8-28 于西安电子科技大学

第一篇 MATLAB 语言入门

第 1 章	MATLAB 语言概述	2
1.1	MATLAB 语言的发展	2
1.2	MATLAB 语言的特点	3
1.3	MATLAB 的工作环境	4
1.3.1	命令窗	4
1.3.2	图形窗	6
1.3.3	文本编辑窗	8
1.4	演示程序	8
第 2 章	基本语法	10
2.1	变量及其赋值	10
2.1.1	标识符与数	10
2.1.2	矩阵及其元素的赋值	11
2.1.3	复数	12
2.1.4	变量检查	13
2.1.5	基本赋值矩阵	15
2.2	矩阵的初等运算	16
2.2.1	矩阵的加减乘法	16
2.2.2	矩阵除法及线性方程组的解	18
2.2.3	矩阵的乘方和幂次函数	20
2.2.4	矩阵结构形式的提取与变换	21
2.3	元素群运算	22
2.3.1	数组及其赋值	22
2.3.2	元素群的四则运算和幂次运算	23
2.3.3	元素群的函数	24
2.4	逻辑判断及流程控制	25
2.4.1	关系运算	25
2.4.2	逻辑运算	27
2.4.3	流程控制语句	28
2.5	基本绘图方法	32
2.5.1	直角坐标中的两维曲线	32

2.5.2	线型、点型和颜色	33
2.5.3	多条曲线的绘制	34
2.5.4	屏幕控制和其他二维绘图	35
2.5.5	三维曲线和曲面	40
2.5.6	特殊图形和动画	42
2.5.7	彩色、光照和图像	44
2.5.8	低层图形屏幕控制功能	46
2.6	M 文件及程序调试	48
2.6.1	主程序文件	49
2.6.2	人机交互命令	50
2.6.3	函数文件	51
2.6.4	文件编辑器及程序调试	53
第 3 章	MATLAB 的开发环境和工具	54
3.1	MATLAB 与其他软件的接口关系	54
3.1.1	与磁盘操作系统的接口关系	54
3.1.2	与文字处理系统 WinWord 的关系	57
3.1.3	图形文件的转储	58
3.1.4	低层输入输出函数库	58
3.1.5	与 C 和 FORTRAN 子程序的动态链接	60
3.2	MATLAB 的文件管理系统	60
3.2.1	安装后的 MATLAB 文件管理系统	60
3.2.2	MATLAB 自身的用户文件格式	61
3.2.3	文件管理和搜索路径	61
3.2.4	与目录和搜索有关的命令	62
3.2.5	搜索顺序	63
3.3	MATLAB 6.x 的开发环境	63
3.3.1	桌面系统的内容	63
3.3.2	桌面命令菜单简介	64
3.3.3	MATLAB 6.x 的用户界面	65
第 4 章	MATLAB 的其他函数库	67
4.1	数据分析函数库 (datafun 函数库)	67
4.1.1	基本的数据分析	67
4.1.2	用于场论的数据分析函数	69
4.1.3	用于随机数据分析的函数	69
4.1.4	用于相关分析和傅里叶分析的函数	70
4.2	矩阵的分解与变换 (matfun 函数库)	72
4.2.1	线性方程组的系数矩阵	72
4.2.2	矩阵的分解	73
4.2.3	矩阵的特征值分析	75

4.2.4	特殊矩阵库 (specmat)	75
4.3	多项式函数库 (polyfun)	76
4.3.1	多项式的四则运算	77
4.3.2	多项式求导、求根和求值	78
4.3.3	多项式拟合	79
4.3.4	多项式插值	80
4.3.5	线性微分方程的解 (residue)	82
4.4	函数功能和数值积分函数库 (funfun)	83
4.4.1	函数功能和数值积分函数库的主要子程序	83
4.4.2	非线性函数的分析	84
4.4.3	任意函数的数值积分	86
4.5	字符串函数库 (strfun)	88
4.5.1	字符串的赋值	89
4.5.2	字符串语句的执行	89
4.5.3	字符串输入输出	90
4.6	稀疏矩阵函数库 (sparfun)	90
4.7	图形界面函数库 (guitools)	92
4.8	数据类型函数库 (datatypes)	93
4.8.1	结构阵列	94
4.8.2	单元阵列	95
4.8.3	类和对象	96
4.9	符号数学 (Symbolic Math) 工具箱简介	99
4.9.1	Symbolic 工具箱的主要功能	99
4.9.2	符号数学式的基本表示方法	99
4.10	习题	101

第二篇 线性代数实践

第 5 章	预备知识	106
5.1	实验在线性代数中的重要性	106
5.2	实验部分的内容组成	108
5.3	直线和平面的快速绘制程序	109
5.4	随机整数矩阵的生成程序	112
5.5	特殊矩阵的生成程序	113
5.6	线性代数建模与应用概述	113
5.7	习题	115
第 6 章	用行阶梯法解线性方程	118
6.1	线性方程组的 MATLAB 表示方法	118
6.2	初等行变换和高斯消元子程序	121

6.3	行阶梯形式的生成	123
6.4	MATLAB 中的行阶梯生成函数	126
6.5	用行阶梯法解欠定方程组	127
6.6	应用实例	130
6.6.1	平板稳态温度的计算	130
6.6.2	化学方程的配平	131
6.6.3	电阻电路的计算	133
6.6.4	交通流量的分析	134
6.7	习题	136
第 7 章	用矩阵运算法解线性方程	138
7.1	矩阵运算规则的 MATLAB 实现	138
7.2	初等变换乘子矩阵的生成	142
7.3	行列式的定义和计算	145
7.4	矩阵的秩和矩阵求逆	148
7.5	用矩阵“除法”解线性方程	150
7.6	应用实例	151
7.6.1	网络的矩阵分割和连接	151
7.6.2	用逆阵进行保密编译码	152
7.6.3	减肥配方的实现	153
7.6.4	弹性梁的柔度矩阵	154
7.6.5	网络和图	156
7.7	习题	158
第 8 章	用向量空间解线性方程组	161
8.1	向量和向量空间	161
8.2	向量空间和基向量	164
8.3	向量的内积和正交性	167
8.4	齐次解空间	171
8.5	超定方程的解——最小二乘问题	173
8.6	应用实例	177
8.6.1	价格平衡模型	177
8.6.2	宏观经济模型	179
8.6.3	信号流程图模型	180
8.6.4	数字滤波器系统函数	182
8.7	习题	184
第 9 章	线性变换及其特征	187
9.1	平面上线性变换的几何意义	187
9.2	二维矩阵特征值的几何意义	189
9.3	三维空间中线性变换的几何意义	193

9.4	基变换与坐标变换	197
9.5	特征值和特征向量的 MATLAB 求法	198
9.6	对称矩阵对角化与二次型主轴	201
9.7	奇异值分解简介	206
9.8	应用实例	209
9.8.1	人口迁徙模型	209
9.8.2	产品成本的计算	211
9.8.3	情报检索模型	212
9.9	习题	214
第 10 章	后续课矩阵建模增补	216
10.1	多项式插值问题	216
10.2	坐标测量仪测定中的拟合问题	217
10.3	天体轨道测量的曲线拟合问题	219
10.4	静力学问题	221
10.5	材料力学的静不定问题	222
10.6	二自由度机械振动的模态分析	224
10.7	交流稳态电路的复数矩阵解	226
10.8	线性系统零输入响应的计算	227
10.9	计算频谱用的 DFT 矩阵	228
10.10	最优化有限冲激响应 (FIR) 数字滤波器设计	230
10.11	信号流图的矩阵建模和计算机求解	232
10.12	自动控制系统的矩阵建模	235
	结束语	238
	附录 A 关于 MATLAB 基本部分函数索引的说明	240
	附录 B 有关美国“用软件工具增强线性代数教学”计划的资料	241
	参考文献	251

MATLAB 语言入门

第一篇

为了使本书能作为一本指南，本书中列出了全部的 MATLAB 基本函数；并采用了多种索引方法。对一些重要的函数指出了它们的应用例题，以便查阅它们的用法；一些比较深入的 MATLAB 函数则用小号字介绍，初学者可以先跳过不看。

作者已按 MATLAB 6.x 和 7.0 为背景，制作了两张在计算机上放映的光盘，可播放四小时，内容覆盖“MATLAB 语言入门”全篇的 90% 以上，可以替代教师讲课。读者可在下载程序集中找到其供货方法的说明。有关本书的改进意见，读者可通过电子邮件或信件向作者提出，作者热忱欢迎。

本篇内容

- MATLAB 语言概述
- 基本语法
- MATLAB 的开发环境和工具
- MATLAB 的其他函数库

MATLAB 语言概述

1.1 MATLAB 语言的发展

MATLAB 是一种科学计算软件，主要适用于矩阵运算及控制和信息处理领域的分析设计。它使用方便，输入简洁，运算高效，内容丰富，并且很容易由用户自行扩展，因此，当前已成为美国和其他发达国家大学教学和科学研究中最常用且必不可少的工具。

MATLAB 是由美国 Mathwork 公司于 1984 年正式推出的，到 1988 年有了 3.1 (DOS) 版本；1992 年推出了 4.1 (Windows) 版本；1997 年推出了 5.1 版本。2001 年推出了 6.1 (R12) 版本。2004 年夏，推出了 MATLAB 7.0 (R14) 的正式版。随着版本的升级，内容不断扩充，功能更加强大。特别是在系统仿真和实时运行等方面，有很多新进展，更扩大了它的应用前景。另一方面，版本的升级对使用环境也提出了更高的要求。

不过 MATLAB 语言的语法从版本 3.0 起已相当成熟，十年来只有很微小的增删。对于学习语法基础的读者来说，各版本的差别不太大。本书的全部程序是在 MATLAB 6.5 (R13) 版本上通过的，但在 5.x 及 4.2c 的环境下也能运行，只有个别程序中的个别语句可能要改一下。根据向上兼容的原则，在 7.0 版本下当然都能运行。

MATLAB 是“矩阵实验室”(Matrix Laboratoy)的缩写，它是一种以矩阵运算为基础的交互式程序语言，专门针对科学、工程计算及绘图的需求。与其他计算机语言相比，其特点是简洁和智能化，适应科技专业人员的思维方式和书写习惯，使得编程和调试效率大大提高。它用解释方式工作，输入程序立即得出结果，人机交互性能好，深得科技人员喜爱。特别是它可适应多种平台，并且，随计算机硬软件的更新及时升级。因此，MATLAB 语言在国外的大学工学院中，特别是数值计算用得最频繁的电子信息技术类学科中，已成为每个学生都掌握的工具了。它大大提高了课程教学、解题作业、分析研究的效率。学习掌握 MATLAB，也可以说是在科学计算工具上与国际接轨。

MATLAB 语言比较好学,因为它只有一种数据类型,一种标准的输入输出语句,不用“指针”,不需编译,比其他语言少了很多内容。听三四个小时课,上机练几个小时,就可入门了。以后自学也十分方便,通过它的演示(demo)和求助(help)命令,人们可以方便地在线学习各种函数的内涵及其用法。

MATLAB 语言的难点是函数较多,仅基本部分就有 700 多个,其中常用的有二三百个,要尽量多记少查,可以提高编程效率,而且这是终身受益的。

1.2 MATLAB 语言的特点

MATLAB 语言有以下特点。

1. 起点高

(1) 每个变量代表一个矩阵,从 MATLAB 名字的来源可知,它以矩阵运算见长,在当前的科学计算中,几乎无处不用矩阵运算,这使它的优势得到了充分的体现。在 MATLAB 中,每个变量代表一个矩阵,它可以有 $n \times m$ 个元素。

(2) 每个元素都看作复数,这个特点在其他语言中也是不多见的。

(3) 所有的运算都对矩阵和复数有效,包括加、减、乘、除、函数运算等。

2. 人机界面适合科技人员

(1) 语言规则与笔算式相似:MATLAB 的程序与科技人员的书写习惯相近,因此,易写易读,易于在科技人员之间交流。

(2) 矩阵行数列数无需定义:要输入一个矩阵,用其他语言时必须先定义矩阵的阶数,而 MATLAB 则不必有阶数定义语句,输入数据的行列数就决定了它的阶数。

(3) 输入算式立即得结果,无需编译:MATLAB 是以解释方式工作的,即它对每条语句解释后立即执行,若有错误也立即作出反应,便于编程者马上改正。这些都大大减轻了编程和调试的工作量。

3. 强大而简易的作图功能

(1) 能根据输入数据自动确定坐标绘图。

(2) 能规定多种坐标系(极坐标,对数坐标等)。

(3) 能绘制三维坐标中的曲线和曲面。

(4) 可设置不同颜色、线型和视角等。

如果数据齐全,通常只需一条命令即可出图。

4. 智能化程度高

(1) 绘图时自动选择最佳坐标。

(2) 做数值积分时,自动按精度选择步长。

(3) 自动检测和显示程序错误的能力强,易于调试。

5. 功能丰富,可扩展性强

MATLAB 软件包括基本部分和专业扩展两大部分。基本部分包括:矩阵的运算和各种

变换、代数和超越方程的求解、数据处理和傅里叶变换、数值积分等，可以满足大学理工科本科的计算需要。本书将主要介绍这部分的内容。专业扩展部分称为工具箱。它实际上是用 MATLAB 的基本语句编成的各种子程序集，用于解决某一方面的专门问题，或实现某一类的新算法。现在已经有控制系统、信号处理、图像处理、系统辨识、模糊集合、神经网络和小波分析等数十个工具箱，并且还在继续发展中。

MATLAB 的核心内容是它的基本部分，所有的工具箱子程序都是用它的基本语句编写而成的。因此，学好基本部分内容是掌握 MATLAB 的关键。

1.3 MATLAB 的工作环境

不同版本的 MATLAB 要安装在不同的操作系统下。MATLAB 4.0 以后的版本都是以 Windows 操作系统为基础的。MATLAB 的工作环境主要由命令窗 (Command Window)，图形窗 (Figure Window)，文本编辑窗 (File Editor) 组成。MATLAB 6.x 和 7.0 又增加了几个辅助视窗，组成其“桌面系统”。考虑到 7.0 版本推出还不久，6.x 版本使用已有五年半，是目前最普及的版本。本书将以 MATLAB 6.x 中的视窗为典型进行讨论。本章着重介绍命令窗，其他视窗将在第 2 章和第 3 章中讨论。

1.3.1 命令窗

在 Windows 桌面上，双击 MATLAB 的图标，就可进入 MATLAB 的工作环境。首先出现 MATLAB 的标志图形，接着出现其默认的桌面系统，如图 1.1 所示。

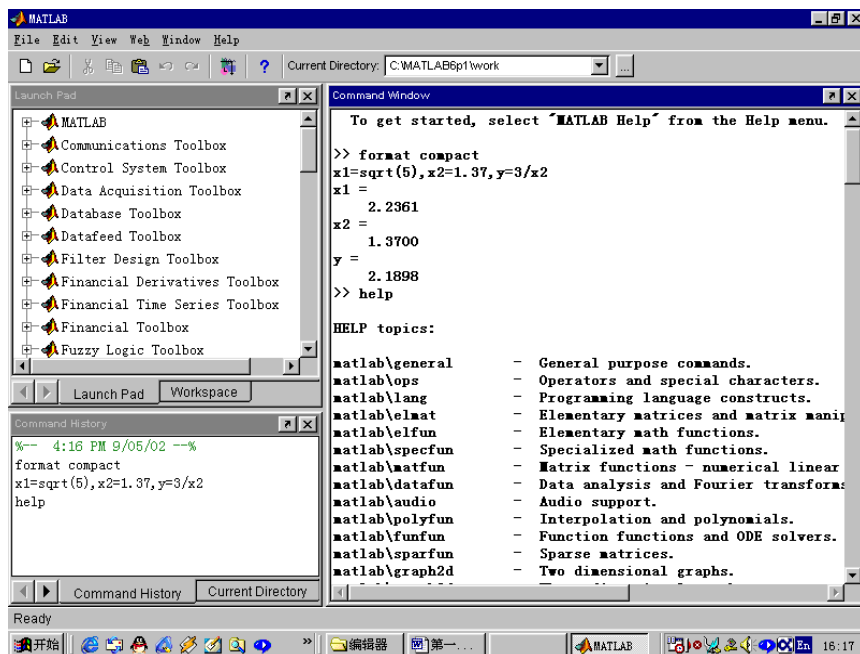


图 1.1 MATLAB 6.x 的桌面系统和命令窗

这是桌面系统的默认画面。其左上视窗为资源目录 (Launch Pad)，可切换为工作空间 (Workspace)；其左下视窗为历史命令 (Command History)，可切换为当前目录 (Current Directory)；右半个视窗则为命令窗 (Command Window)。命令窗是用户与 MATLAB 做人机对话的主要环境。>>是它的提示符，可以在提示符后输入 MATLAB 的各种命令并读出相应的结果。例如输入：

```
x1=sqrt(5), x2=1.37, y=3/x2
```

答案为：x1=2.2361 x2=1.3700 y=2.1898

命令窗主菜单的有些项目与 Word 相仿。这里只对几个主要的命令做一些说明。

- **format 命令：**在这行程序输入前，输入了显示格式命令 **format compact** (紧凑格式)，因为在 MATLAB 默认的 **format loose** (稀疏格式) 下，屏幕上的显示会有许多空行，会多占教材篇幅。**format** 命令还可以控制数字显示的方式。虽然 MATLAB 唯一采用双精度格式进行数据的存储和运算，但数字的显示格式可以有八种，在各种格式控制命令下圆周率 π 的显示结果如表 1.1 所示。

表 1.1 数字显示的八种格式

MATLAB 命令	显示形式	说 明
format long	3.14159265358979	定点 15 位十进制数
format short e	3.1416e+000	浮点 5 位十进制数加指数
format long e	3.141592653589793e+000	浮点 15 位十进制数加指数
format hex	400921fb54442d18	16 位十六进制数
format bank	3.14	两位小数
format +	+	正、负或零
format rat	355 / 113	分数近似
format short (默认)	3.1416	定点 5 位十进制数

显示格式也是 MATLAB 接受输入数据的格式。

- **命令窗编辑功能：**输入和修改程序的方法与通常的文字处理相仿。特殊的功能键有以下三个。
 - (1) **ESC** 恢复命令输入的空白状态。
 - (2) **↓** 调出下一行命令。
 - (3) **↑** 调出上一行 (历史) 命令。

这个功能在程序调试时十分有用。对于已执行过的命令，如要做些修改后重新执行，就可不必重新输入，用 **↑** 键调出原命令做修改即可。

- **主菜单中的编辑 (Edit) 项功能：**用它可以把屏幕上选中了的文字裁剪 (Cut) 或复制 (Copy) 下来，放在剪贴板 (Clip Board) 上，然后粘贴 (Paste) 到任一其他视窗的任何位置上去，这是 MATLAB 与其他软件 (例如 Word) 交换文件、数据和图形的重要方法。
- **主菜单中的视图 (View) 项功能：**用它可以改变屏幕上显示的视窗布局。例如，

我们希望只显示命令窗，使它占整个屏幕，就可如图 1.2 那样，依次引出 View 的下拉菜单，由【View】→【Desktop Layout】→【Command Window Only】，然后可进行如下的操作。

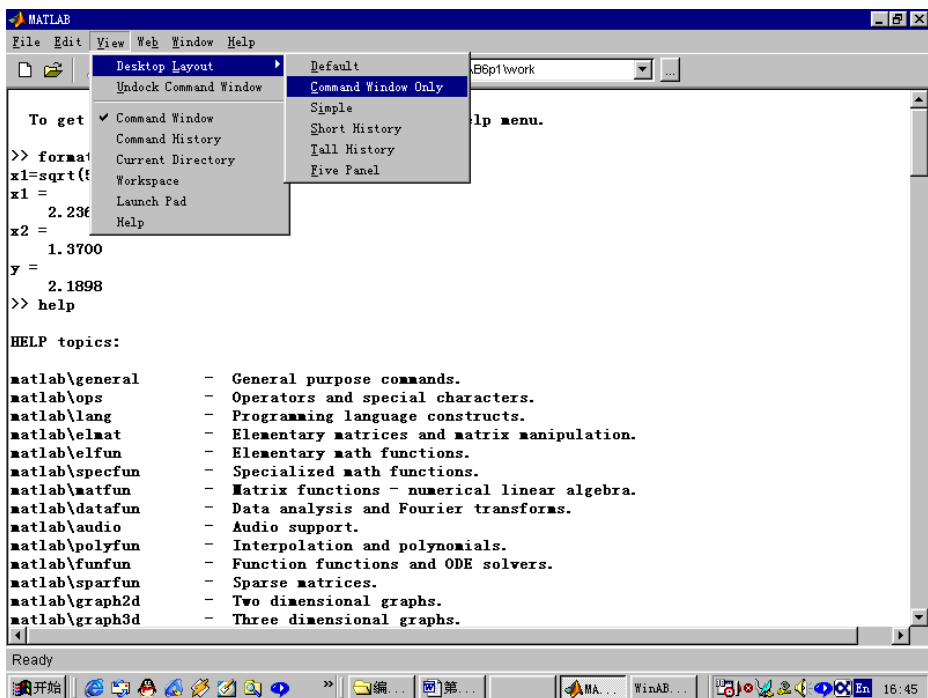



图 1.2 只显示命令窗的屏幕及其生成的菜单

(1) 输入“help”，屏幕上将显示系统中已装入的函数库（即子目录）的名称。如果只装了 MATLAB 的基本部分，则屏幕上显示出表 1.2 中所示的子目录名称。

(2) 输入 help 子目录名，如 help elfun，即得出 elfun 库中各函数名。

(3) 输入 help 函数名，如 help tan2，即得到 tan2 函数的意义及用法。

(4) 退出 MATLAB 有两种方法。一是输入 exit 或 quit，还有一种是用鼠标双击左上角的小方块或单击右上角的  号。但后一种情况属非正常退出，该次进程的所有命令将不记录在“历史命令窗”中，故应尽量避免采用。

1.3.2 图形窗

通常，只要执行了任一种绘图命令，就会自动产生图形窗。以后的绘图都在这一个图形窗中进行。如想再建一个或几个图形窗，则可输入 figure，MATLAB 会新建一个图形窗，并自动给它依次排序，如果要人为规定新图为图 3，则可输入 figure (3)。如要调看已经存在的图形窗 n，也应输入 figure (n)。

在命令窗中，输入 figure，得出空白的图形窗。如输入 logo，即可生成 MATLAB 6.x 的命令窗、图形窗和标志图形，如图 1.3 所示。图形窗上的一排按钮，可以用来对图形进行修改或注释。

表 1.2 MATLAB 基本部分的函数库

库 内 容	库 名	库 序 号	在本书中的章节和表数
数据分析函数库	datafun	(a)	4.1 节表 4.3
动态数据交换库	dde	(g)	3.3 节表 3.4
初等数学函数库	elfun	(c)	2.3 节表 2.7
基本矩阵库	elmat	(d)	2.1 节表 2.1
时间日期函数库	timefun	(w)	3.1 节表 3.2
非线性数值方法库	funfun	(e)	4.4 节表 4.6
通用命令库	general	(f)	3.1 节表 3.1
数据类型库	datatypes	(b)	4.7 节表 4.11
通用图形函数库	graphics	(h)	2.5 节表 2.13
低层输入/出函数库	iofun	(j)	3.3 节表 3.3
语言结构函数库	lang	(k)	2.6 节表 2.17
矩阵线性代数库	matfun	(m)	4.2 节表 4.4
运算符和特殊字符库	ops	(n)	2.4 节表 2.9
二维图形库	graph2d	(p)	2.5 节表 2.14
特殊图形函数库	specgraph	(u)	2.5 节表 2.15
三维图形库	graph3d	(q)	2.5 节表 2.16
多项式和插值函数库	polyfun	(r)	4.3 节表 4.5
稀疏矩阵函数库	sparfun	(s)	4.6 节表 4.9
特殊数学函数库	specfun	(t)	4.4 节表 4.7
字符串函数库	strfun	(v)	4.5 节表 4.8
用户界面工具库	Guitools	(x)	4.7 节表 4.10
MATLAB 演示库	demos	(y)	未列出

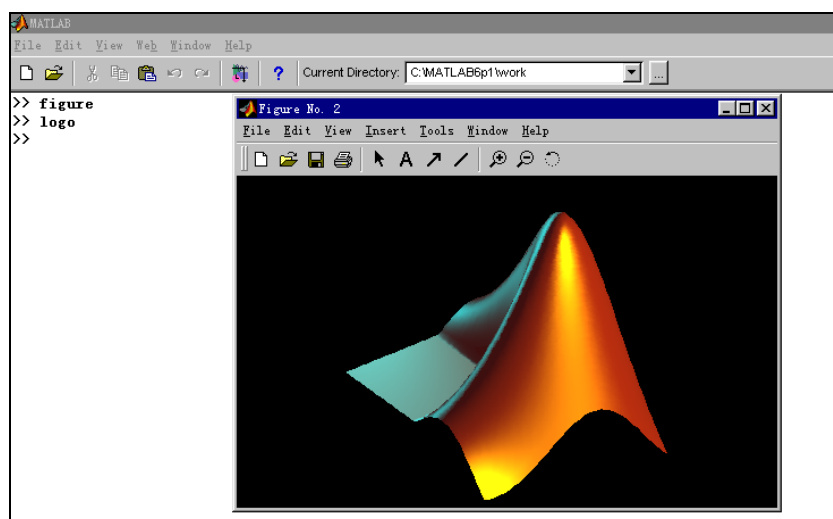


图 1.3 MATLAB 6.x 的命令窗、图形窗和标志图形

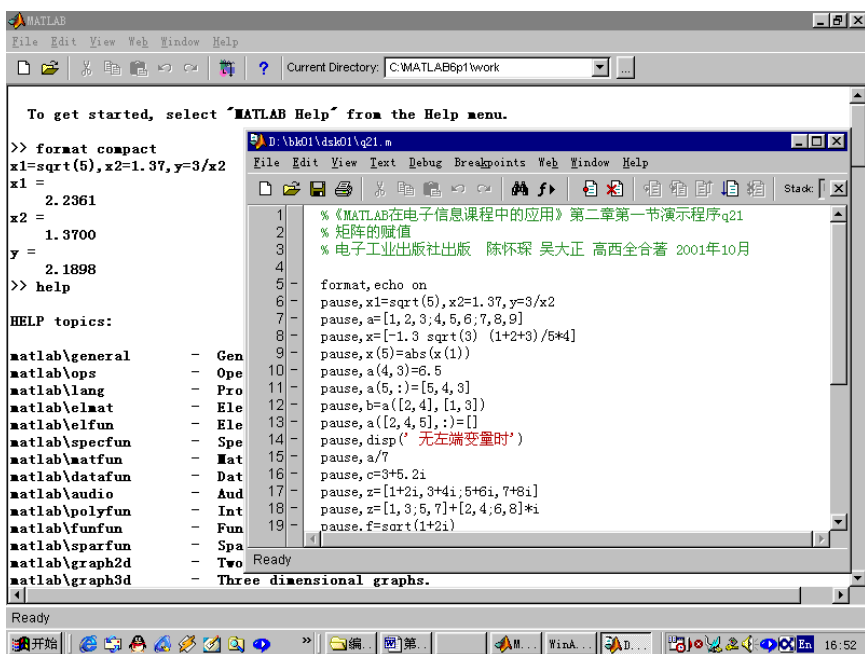


图 1.4 MATLAB 6.x 的命令窗和文本编辑窗

1.3.3 文本编辑窗

MATLAB 有两种程序编制方式：一种称为行命令方式，这就是在命令窗中一行一行地输入程序，计算机每次对一行命令做出反应，像计算器那样。这只能编简单的程序，在入门时可以用这种方式。程序稍复杂一些，就应把程序写成一个有多行语句组成的文件，让 MATLAB 来执行这个文件。编写和修改这种文件程序就要用到文本编辑器。

命令窗上方最左边的按钮是用来打开文本编辑器空白页的，左边第 2 个按钮是用来打开原有程序文件的。按这个按钮并选择文件 ag21.m 后就得到如图 1.4 的画面。

1.4 演示程序

在命令窗中输入 Demo，将出现 MATLAB 的演示视窗，如图 1.5 所示。

演示视窗的左侧是库目录。图上选定的是图形类（Graphics），右方上部是对该演示库的说明，下部则是库中各项目的名称。双击该名称或选中该项目后单击右下角的【Run...】方框，即出现该项目的演示界面。通常，演示画面的右侧是一些功能按钮，左上半部是图形，而左下半部则是相应的 MATLAB 程序语句，还可以在界面上直接修改这些语句并重新执行。因此，演示程序也是一个很好的学习过程。

例如，图 1.5 中选的是 MATLAB 的基本部分【MATLAB】中的绘图库【Graphics】，下选的项目是复数函数图形【Complex Function Plots】。当前选择的例子是复数 z 的三次方（【 z^3 】按钮），如图 1.6 所示。图 1.6 中，底平面表示复数自变量 z 的实部和虚部所张成的

平面，而高则表示 z^3 的绝对值大小。读者可自行判断为什么此图形有三个翼。注意，左下角方框中就是生成此图的 MATLAB 程序，其中前两句是注释，只有后两句是有实效的。这说明 MATLAB 的程序非常简练。修改这两个语句的参数就可改变相应的图形，例如，将最后语句的末位数字由 3 改为 5，再用鼠标单击图形部分，即可生成 z^5 的图形。

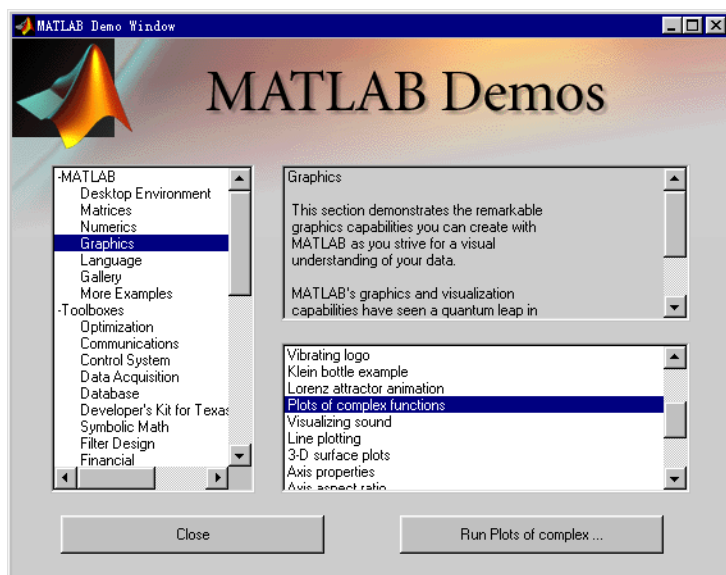


图 1.5 MATLAB 的演示视窗

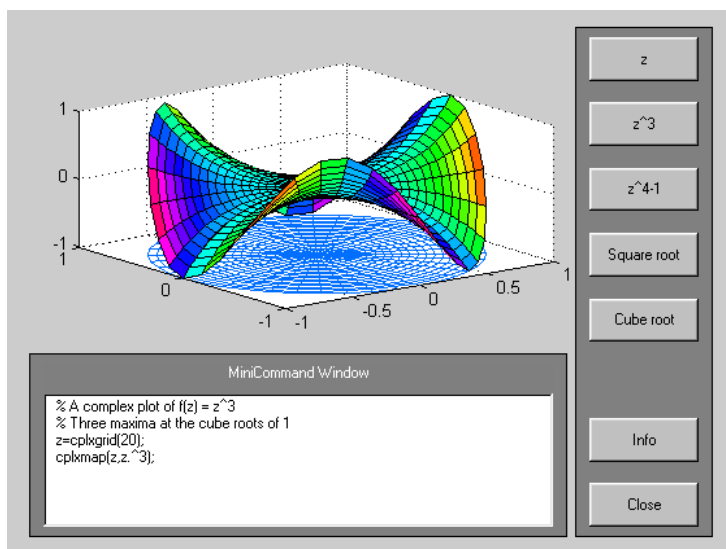


图 1.6 复数图形的演示视窗

第 2 章

基本语法

本章主要介绍 MATLAB 的基本语法。

2.1 变量及其赋值

2.1.1 标识符与数

标识符是标识变量名、常量名、函数名和文件名的字符串的总称。在 MATLAB 中，变量和常量的标识符最长允许 19 个字符；函数和文件名则通常不超过 8 个字符（受文件管理器的限制，但 MATLAB 5.1 的许多函数和文件名就超过了 8 个字符，因此，要求安装在能管理长文件名的操作系统中）。这些字符包括全部的英文字母（大小写共 52 个）、阿拉伯数字和下划线等符号。标识符中第 1 个字符必须是英文字母。MATLAB 对大小写敏感 (Case Sensitive)，即它把 A 和 a 看做两个不同的字符。

MATLAB 内部只有一种数据格式，那就是双精度（即 64 位）二进制数，对应于十进制 16 位有效数和 ± 308 次幂。MATLAB 进行运算和存储时都用双精度格式，这对绝大多数工程计算是足够的，许多情况下甚至过于“浪费”。在一些其他的算法语言中设有多种数据格式，如字符型（8 位）、整数型（16 位）、单精度型（32 位）等，可节省内存和提高速度，但增加了编程的复杂性。MATLAB 把简化编程作为其主要目标，省去了多种数据格式，但在运算速度和内存消耗方面付出了代价。不过，现在计算机的时钟频率和内存容量都以几何级数迅速增长，MATLAB 付出的代价容易得到弥补。虽然它的数据格式只有一种，但为了人机交互的友好方便，数字的显示格式有八种，已在第 1 章中加以说明。

2.1.2 矩阵及其元素的赋值

赋值就是把数赋予代表常量或变量的标识符。**MATLAB** 中的变量或常量都代表矩阵，标量应看做 1×1 阶的矩阵。赋值语句的一般形式为：

变量=表达式（或数）

例如输入语句

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

则显示结果为

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

元素也可以用表达式代替，如输入 `x=[-1.3 sqrt(3) (1+2+3)/5*4]`

结果为 `x=-1.3000 1.7321 4.8000`

可以看出，矩阵的值放在方括号中，同一行中各元素之间以逗号或空格分开，不同的行则以分号隔开，语句的结尾可用回车符或逗号，此时会立即显示运算结果。如果不希望显示结果，就以分号结尾。此时运算仍然执行，只是不显示。

变量的元素用圆括号“`()`”中的数字（也称为下标）来注明，一维矩阵（也称数组或向量）中的元素用一个下标表示，二维的矩阵可有两个下标数，以逗号分开。**MATLAB 5.x** 已扩展了三维和更高维的矩阵，可以有三个或更多下标，用户可以单独给元素赋值，如：`x(2)=1.7321`, `a(2, 3)=6` 等。如果赋值元素的下标超出了原来矩阵的大小，矩阵的行列会自动扩展。如：

```
x(5)=abs(x(1))
```

得 `x=-1.3000 1.7321 4.8000 0 1.3000`

又如输入 `a(4, 3)=6.5`

```
得 a= 1.0000 2.0000 3.0000
      4.0000 5.0000 6.0000
      7.0000 8.0000 9.0000
      0      0      6.5000
```

可见，跳空的元素 `x(4)`, `a(4,1)`, `a(4,2)` 被自动赋值 0。这种自动扩展维数的功能只适用于赋值语句。在其他语句中若出现超维调用的情况，系统将给出出错提示。

给全行赋值，可用冒号。例如，给 `a` 的第 5 行赋值。

输入 `a(5, :)=[5, 4, 3]`

```
得 a = 1.0000 2.0000 3.0000
      4.0000 5.0000 6.0000
      7.0000 8.0000 9.0000
```

0	0	6.5000
5.0000	4.0000	3.0000

把 a 的第 2, 4 行及第 1, 3 列交点上的元素取出, 构成一个新矩阵 b 。

可输入 $b = a([2, 4], [1, 3])$

得 $b = \begin{bmatrix} 4.0000 & 6.0000 \\ 0 & 6.5000 \end{bmatrix}$

要抽去 a 中的第 2、4、5 行, 可利用空矩阵 $[]$ 的概念。

输入 $a([2, 4, 5], :) = []$

得 $a = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$

注意, “空矩阵”是指没有元素的矩阵。对任何一个矩阵赋值 $[]$, 就是使它的元素都消失掉。这完全不同于“零矩阵”, 后者是元素存在, 只是其数值为零而已。可以看出, 空矩阵是使矩阵减缩时不可缺少的概念。

除“变量=表达式(或数)”的标准赋值格式外, 还可以不要等式左端而只剩下“表达式”。这有两种可能: (1) 该表达式并不产生数字解, 例如, 产生图形或改变系统状态; (2) 该表达式产生数字解, 但不需保存它。此时, MATLAB 自动给出一个临时变量 ans , 把右端的结果暂存在 ans 中。例如:

输入 $a/7$

得 $ans = \begin{bmatrix} 0.1429 & 0.2857 & 0.4286 \\ 1.0000 & 1.1429 & 1.2857 \end{bmatrix}$

2.1.3 复数

MATLAB 的每一个元素都可以是复数, 实数是复数的特例。复数的虚数部分用 i 或 j 表示。这是在 MATLAB 启动时就在内部设定的。例如:

输入 $c = 3+5.2i$

得 $c = 3.0000 + 5.2000i$

对复数矩阵有两种赋值方法。

(1) 将其元素逐个赋予复数, 例如:

输入 $z = [1+2i, 3+4i; 5+6i, 7+8i]$

得 $z = \begin{bmatrix} 1.0000 + 2.0000i & 3.0000 + 4.0000i \\ 5.0000 + 6.0000i & 7.0000 + 8.0000i \end{bmatrix}$

(2) 将其实部和虚部矩阵分别赋值, 例如:

$z = [1, 3; 5, 7] + [2, 4; 6, 8]*i$

两种赋值方法得出同样的结果。注意，只有数字和 i 的乘积可省略乘号，在上述矩阵式中若省略乘号“ $*$ ”，就会出错。另外，如果在前面程序中曾经给 i 或 j 赋过其他值，则 i 、 j 已经不是虚数符号，这些虚数赋值语句都不对了。此时应输入：

```
clear i, j
```

即把原赋的 i 、 j 清掉，然后再执行复数赋值语句。

MATLAB 中所有的运算符和函数都对复数有效。例如：

```
输入 f = sqrt(1+2i)
```

```
得 f = 1.2720 + 0.7862i
```

```
检验 f*f
```

```
ans = 1.0000 + 2.0000i
```

因此，复数的表达式同样也能作为赋值语句。再来看复数矩阵 z 的转置、共轭运算，运算符 $'$ 表示把矩阵作共轭转置，即把它的行列互换，同时，把各元素的虚部反号。函数 `conj` 则只把各元素的虚部反号，即只取共轭。所以，若求转置而不要共轭，就把 `conj` 和 $'$ 结合起来完成。例如：

```
输入 w=z' (共轭转置), u=conj(z) (共轭), v=conj(z)' (转置)
```

```
得 w = 1.0000 - 2.0000i      5.0000 - 6.0000i
```

```
      3.0000 - 4.0000i      7.0000 - 8.0000i
```

```
u = 1.0000 - 2.0000i      3.0000 - 4.0000i
```

```
      5.0000 - 6.0000i      7.0000 - 8.0000i
```

```
v = 1.0000 + 2.0000i      5.0000 + 6.0000i
```

```
      3.0000 + 4.0000i      7.0000 + 8.0000i
```

2.1.4 变量检查

在调试程序时，往往需要检查工作空间中的变量及其阶数。可用 `who` 或 `whos` 命令。

```
输入 who
```

```
得 Your variables are:
```

```
a      c      v      x1      z
```

```
ans    f      w      x2
```

```
b      u      x      y
```

这些是前面用过的变量，如果还需要知道它们的详细特征，可输入：

```
whos
```

得	变量名	阶数	元素数	字节数	密度	复数
	a	2 by 3	6	48	Full	No
	ans	2 by 2	4	64	Full	Yes

b	2 by 2	4	32	Full	No
c	1 by 1	1	16	Full	Yes
...
z	2 by 2	4	64	Full	Yes

Grand total is 40 elements using 496 bytes (共 40 个元素, 占 496 字节)

可以看出, 每个实元素占 8 个字节, 复元素则占 16 个字节。读者可自行解释其原因。

在 MATLAB 中实际上还有几个内定的变量, 在变量检查时不显示。把它们列于表 2.1 的特殊变量栏中。这里着重介绍一下 Inf 和 NaN。

表 2.1 基本矩阵和矩阵运算 (elmat) (d)

基本矩阵	zeros	全 0 矩阵 ($m \times n$ 阶)	logspace	对数均分向量 ($1 \times n$ 维数组)
	ones	全 1 矩阵 ($m \times n$ 阶)	Freqspace	频率特性的频率区间
	rand	随机数矩阵 ($m \times n$ 阶)	meshgrid	画三维曲面时的 X, Y 网格
	randn	正态随机数矩阵 ($m \times n$ 阶)	Linspace	均分向量 ($1 \times n$ 维数组)
	Eye (n)	单位矩阵 (方阵)	:	将元素按列取出排成一列
特殊变量和函数	ans	最近的答案	inf	Infinity (无穷大)
	eps	浮点数相对精度	NaN	Not-a-Number (非数)
	realmax	最大浮点实数	flops	浮点运算次数
	realmin	最小浮点实数	computer	计算机类型
	pi	3.14159235358579	inputname	输入变量名
	i, j	虚数单位	size	多维矩阵的各维长度
	length	一维矩阵的长度		
矩阵结构提取和变换	cat *	链接数组	diag	提取或建立对角阵
	fliplr	矩阵左右翻转	ind2sub	把元素序号变为矩阵下标
	flipud	矩阵上下翻转	sub2ind	把矩阵下标变为元素序号
	repmat	复制和排成矩阵	tril	取矩阵的左下三角部分
	reshape	维数重组 (元素总数不变)	triu	取矩阵的右上三角部分
	Rot90	矩阵整体反时针旋转 90°		
特殊矩阵	compan	Companion 矩阵	magic	魔方矩阵
	gallery	Higham 测试矩阵	pascal	Pascal 矩阵
	hadamard	Hadamard 矩阵	rosser	经典的对称特征值测试问题
	hankel	Hankel 矩阵	Toeplitz	Toeplitz 矩阵
	hilb	Hilbert 矩阵	vander	Vandermonde 矩阵
	invhilb	Hilbert 逆矩阵	wilkinson	Wilkinson's 特征值测试矩阵

Inf (还有 -Inf) 是无穷大, 输入 1/0 就可得到它。NaN 是非数字 (Not a Number) 的缩写, 由 0/0, 0*Inf 或 Inf/Inf 而得。在其他语言中遇到上述非法运算时, 系统就停止运算并

退出。而 MATLAB 却不停止运算，仍给结果赋予 Inf 或 NaN，并继续把程序执行完。这有很大的好处，可以避免因为一个数据不好而破坏全局。出现 Inf 或 NaN 后，对它们做任何运算，结果仍为 Inf 或 NaN，这种运算规则称为 IEEE（电工和电子工程师协会）运算规则，它是 IEEE 的一种标准。

2.1.5 基本赋值矩阵

为了方便给大量元素赋值，MATLAB 提供了一些基本矩阵。表 2.1 给出了最常用的一些。其用法可从下面的例子中看到。其中，魔方矩阵 `magic(n)` 的特点是：其元素由 1 到 nn 的自然数组成；每行、每列及两对角线上的元素之和均等于 $(n^3+n)/2$ 。单位矩阵 `eye(n)` 是 $n \times n$ 阶的方阵，其对角线上的元素为 1，其余元素均等于 0。下例列出了表 2.1 中的 4 种矩阵。

输入 `f1=ones(3, 2), f2=zeros(2, 3), f3=magic(3), f4=eye(2)`

得全 1 矩阵 `f1 =`

1	1
1	1
1	1

全 0 矩阵 `f2 =`

0	0	0
0	0	0

魔方矩阵 `f3 =`

8	1	6
3	5	7
4	9	2

单位矩阵 `f4 =`

1	0
0	1

线性分割函数 `linspace(a, b, n)` 在 a 与 b 之间均匀地产生 n 个点值，形成 n 维向量。例如：

输入 `f5 = linspace(0, 1, 5)`

得 `f5 =` 0 0.2500 0.5000 0.7500 1.0000

大矩阵可由若干个小矩阵组成，但必须其行列数正确，恰好填满全部元素。例如：

输入 `fb1= [f1, f3;f4, f2]`

得 `fb1 =`

1	1	8	1	6
1	1	3	5	7
1	1	4	9	2
1	0	0	0	0
0	1	0	0	0

可再由它与 `f5` 组成一个更大的矩阵。

输入 `fb2= [fb1;f5]`

```

得      fb2 =   1.0000   1.0000   8.0000   1.0000   6.0000
              1.0000   1.0000   3.0000   5.0000   7.0000
              1.0000   1.0000   4.0000   9.0000   2.0000
              1.0000   0         0         0         0
              0         1.0000   0         0         0
              0         0.2500   0.5000   0.7500   1.0000

```

可以看出 **fb1** 和 **fb2** 显示的数据不同，**fb1** 的元素都是整数，而 **fb2** 则都是带小数的。其原因是 **MATLAB** 要求一个矩阵中所有元素用同一显示格式。因为 **f5** 中元素含小数，故所有的元素都得用小数格式来显示（0 元素除外，它表示真正的 0 与无法显示的小数值 0.0000 不同）。

为了用同一显示格式，当矩阵中的最大元素小于 0.001 或其最小元素大于 1000 时，**MATLAB** 会把其中小于 0.001 或大于 1000 的公因子提出来，如输入由两个很小的数组成的矩阵。

```

f = [0.000073,    5.33e-6]
得  f = 1.0e-004 *
    [0.73    0.0533]

```

如果矩阵中出现大小差别很多的元素，则显示时将以大元素优先，小元素就只能显示很少的有效位，甚至成为 0.0000，这时不要误以为它是 0。可以用显示单个元素的命令来得到它的准确值，也可改用长格式（**format long**）来显示整个矩阵。

2.2 矩阵的初等运算

2.2.1 矩阵的加减乘法

矩阵算术运算的书写格式与普通算术相同，包括加、减、乘、除，也可用括号来规定运算的优先次序。但它的乘法定义与普通数（标量）不同。相应地，作为乘法逆运算的除法也不同，有左除（\）和右除（/）两种符号。

两矩阵的相加（减）就是其对应元素的相加（减），因此，要求相加的两矩阵的阶数必须相同。检查矩阵阶数的 **MATLAB** 语句是 **size**，例如：

```

输入      [n, m]=size(fb2)
得        n=6      m=5      （6 行 5 列）

```

如果要自己编写矩阵 **A** 和 **B** 相加（减）的程序，就必须先求 n_A , m_A , n_B , m_B ，并检验是否满足 $n_A=n_B$ 和 $m_A=m_B$ 。确认无误后再按对应元素相加（减），得出 $C=A+B$ （或 $A-B$ ）。如果阶数检验不合格，则显示出错。当两个相加矩阵中有一个是标量时，**MATLAB** 承认算式有效，它自动把该标量扩展成同阶等元素矩阵，与另一矩阵相加。例如：

输入 $X = [-1 \ 0 \ 1]$; $Y = X - 1$
 得 $Y = -2 \ -1 \ 0$

如果已经知道 X 是一维矩阵（数组），也可以用

$l = \text{length}(X)$

来求它的长度。注意， size 有 2 个输出量，而 length 只有 1 个输出量。 length 不区分列或行，所以，作加减法阶数检验时只能用 size 。

现在来看矩阵的乘法。 $n \times p$ 阶矩阵 A 与 $p \times m$ 阶矩阵 B 的乘积 C 是一个 $n \times m$ 阶矩阵，它的任何一个元素 $C(I,j)$ 的值为 A 阵的第 i 行和 B 阵的第 j 列对应元素乘积的和。即

$$C(I,j) = A(I,1)B(1,j) + A(I,2)B(2,j) + \cdots + A(I,p)B(p,j) = \sum_k A(I,k)B(k,j)$$

式中的乘号是普通数（标量）的乘号。 p 是 A 阵的列数，也是 B 阵的行数，也称为两个相乘矩阵的内阶数，两矩阵相乘的必要条件是它们的内阶数相等。不难看出，对于标量 A, B ，因为 n, p, m 均为 1，矩阵乘法就退化为普通数的乘法。

如果要自己编写矩阵 A 和 B 相乘的程序，就必须先求 nA, mA, nB, mB ，并检验 nA 是否等于 nB ，确认无误后再按上式把对应元素相乘后累加，得出 $C(I,j)$ 。分别取 i 从 1 到 nA ， j 从 1 到 mB ，得出 $nA \times mB$ 个 C 元素，排成矩阵形式，得到 C 。

实际上，MATLAB 已将上述矩阵加、减、乘的程序编程为内部函数，只要用 $+$ 、 $-$ 、 $*$ 做运算符号就包含了检查阶数和执行运算的全过程，而且，其运算对复数有效。

由此可见，前面定义的 X 和 Y 是不能相乘的，因为它们的内阶数分别为 3 和 1。如果两个乘数之一是标量，则 MATLAB 不检查其内阶数，而用该标量乘以矩阵的每个元素。例如：

输入 $\text{pi} * X$
 得 $\text{ans} = -3.1416 \quad 0 \quad 3.1416$

若把 Y 转置，成为 3×1 阶，则内阶数与 X 的相同，就可求

$X * Y'$
 得 $\text{ans} = 2$

不难用心算来检验其正确性。这个式子可读成 X 左乘 Y' 。现在让 X 右乘 Y' ，这时两者的内阶数都是 1，而外阶数（定义中的 nA 和 mB ）都成了 3。于是有

$Y' * X$
 得 $\text{ans} = \begin{matrix} 2 & 0 & -2 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{matrix}$

显然， X 左乘和右乘 Y' 所得的结果是完全不同的。只有单位矩阵例外，单位矩阵乘以任何矩阵 A （其阶数为 $nA \times mA$ ）时，不管是左乘还是右乘，其积仍等于该矩阵。即

$\text{eye}(nA) * A = A$
 $A * \text{eye}(mA) = A$

读者可按单位矩阵的定义自行检验其正确性。

设有三个线性方程组成的联立方程组：

$$x_1 + 2x_2 + 3x_3 = 2$$

$$3x_1 - 5x_2 + 4x_3 = 0$$

$$7x_1 + 8x_2 + 9x_3 = 2$$

令 $A = [1, 2, 3; 3, -5, 4; 7, 8, 9]$; $X = [x_1; x_2; x_3]$; $b = [2; 0; 2]$

则这个联立方程组就可表示为简洁的矩阵形式

$$AX=b$$

给出 A 和 X ，可以用乘法求出 b 。但若给出 A 和 b 求 X ，那就需要逆运算（矩阵除法）了。

2.2.2 矩阵除法及线性方程组的解

在线性代数中，没有除法，只有逆矩阵。矩阵除法是 MATLAB 从逆矩阵的概念引申来的。先介绍逆矩阵的定义，对于任意 $n \times n$ 阶方阵 A ，如果能找到一个同阶的方阵 V ，使

$$AV=I$$

其中， I 为 n 阶的单位矩阵 $\text{eye}(n)$ 。则 V 就是 A 的逆阵。数学符号表示为

$$V=A^{-1}$$

逆阵 V 存在的条件是 A 的行列式 $\det(A)$ 不等于 0， V 的最古典的求法为高斯消去法，可参阅线性代数书。MATLAB 已把它做成了内部函数 `inv`，输入

$$V=\text{inv}(A)$$

就可得到 A 的逆矩阵 V 。如果 $\det(A)$ 等于或很接近于零，MATLAB 会显示出错或警告信息：

“ A 矩阵病态 (ill-conditioned)，结果精度不可靠。”

现在来看方程 $D \cdot X = B$ ，设 X 为未知矩阵，在等式两端同时左乘以 $\text{inv}(D)$ ，即

$$\text{inv}(D) \cdot D \cdot X = \text{inv}(D) \cdot B$$

等式左端 $\text{inv}(D) \cdot D = I$ ，而 $I \cdot X = X$ ，因此，上式成为

$$X = \text{inv}(D) \cdot B = D \backslash B$$

把 D 的逆阵左乘以 B ，MATLAB 就记作 $D \backslash$ ，称之为“左除”。从 $D \cdot X = B$ 的阶数检验可知， B 与 D 的行数相等，因此，左除时的阶数检验条件是：两矩阵的行数必须相等。

如果原始方程的未知矩阵在左而系数矩阵在右，即

$$X \cdot D = B$$

则按上述同样的方法可以写出

$$X = B \cdot \text{inv}(D) = B / D$$

把 D 的逆阵右乘以 B ，记作 $/D$ ，称之为“右除”。同理，右除时的阶数检验条件是：两矩阵的列数必须相等。

下面来看矩阵左右乘除的一些示例。设 $A=[1, 2, 3; 4, 5, 6]$, $B=[2, 4, 0; 1, 3, 5]$, $D=[1, 4, 7; 8, 5, 2; 3, 6, 0]$
其乘除的结果列于表 2.2 中。

表 2.2 矩阵乘除法的示例

算 式	答 案
$A*B$??? Error using ==> * Inner matrix dimensions must agree. (内阶数必须相等)
$A'*B$	6 16 20 9 23 25 12 30 30
$A*B'$	10 22 28 49
$D\backslash A$??? Error using ==> \ Matrix dimensions must agree. (行数不等)
$D\backslash A'$	-0.0370 0 0.5185 1.0000 -0.1481 0.0000
A/D	0.4074 0.0741 0.0000 0.7407 0.4074 0.0000

矩阵除法可以用来方便地解线性方程组。例如要求下列方程组的解 $x=[x_1; x_2; x_3]$ 。

$$\begin{aligned} 6x_1 + 3x_2 + 4x_3 &= 3 \\ -2x_1 + 5x_2 + 7x_3 &= -4 \\ 8x_1 - 4x_2 - 3x_3 &= -7 \end{aligned}$$

此式可写成矩阵形式 $Ax=B$ ，求解的 MATLAB 程序为

```
A = [6, 3, 4; -2, 5, 7; 8, -4, -3]; B = [3; -4; -7]; x = A\B
得      x = 0.6000
          7.0000
        -5.4000
```

MATLAB 中的除法还可以用来解方程数不等于未知数个数的情况。比如，再加上一个方程

$$x_1+5x_2-7x_3=9$$

这时系数矩阵 $A1$ 的阶数为 4×3 ，不难看出 $A1$ 的行数 $nA1$ 是方程数，其列数 $mA1$ 是未知数的个数， $nA1>mA1$ ，说明方程组是超定的，方程无解。照样列出 MATLAB 程序

```
A1 = [6, 3, 4; -2, 5, 7; 8, -4, -3; 1, 5, -7]; B1 = [3; -4; -7; 9]; x1 = A1\B1
答案为  x1 = -0.1564
          1.0095
```

-0.6952

它并未显示出错信息，却给出了解，这怎么可能呢？实际上，这时 MATLAB 给出的是最小二乘解。把这个 x_1 代入方程组，肯定任何一个方程都不满足，都可得出一个误差，把这 4 个误差的平方相加开方，称为均方差。解 x_1 保证比其他任何解所得的均方差都小。

MATLAB 中的除法还可以用来解方程数少于未知数个数的情况， A_1 矩阵的 $n_{A1} < m_{A1}$ ，说明方程组是不定的，它有无穷个解。此时，仍然可用除法符号来求出解。这个解是满足方程的，但它不是唯一解。它是令 x_1 中某个或某些元素为 0 的一个特殊解。

2.2.3 矩阵的乘方和幂次函数

MATLAB 的运算符 $*$ 、 \wedge 、 \backslash 和 \wedge ，指数函数 \expm 、对数函数 \logm 和开方函数 \sqrtm 是对矩阵进行的，即把矩阵作为一个整体来运算。除此之外，其他 MATLAB 函数都是对矩阵中的元素分别进行，英文直译为数组运算 (Array Operations)，较准确的意义应为“元素群运算”，将在 2.3 节进行讨论。

在幂次运算时矩阵可以作为底数，指数是标量。这是矩阵乘法的扩展，为了保证做乘法时内阶数相同，作为底数的矩阵必须是方阵。矩阵也可以作为指数，底数是标量，此时矩阵仍然应是方阵。底数和指数不能同时为矩阵，如果这样输入，将显示出错信息。表 2.3 给出了一些语句及其结果。注意 \sqrtm 与 \sqrt 、 \expm 与 \exp 、 \logm 与 \log 的不同，也就是矩阵整体运算和矩阵元素群运算的不同。

表 2.3 矩阵整体运算的例及其结果

输入语句	输出结果	说 明
D^2	54 66 15 54 69 66 51 42 33	按矩阵运算
$2.^D$	2 16 128 256 32 4 8 64 1	按元素群运算
D^s	??? Error using ==> ^ Matrix dimensions must agree	非法运算
$u1=\sqrtm(s)$	$u1 = 0.5537 + 0.4644i \quad 0.8070 - 0.2124i$ $1.2104 - 0.3186i \quad 1.7641 + 0.1458i$	按矩阵运算 可用 $u1*u1=s$ 检验
$u2=\sqrt(s)$	$u2 = 1.0000 \quad 1.4142$ $1.7321 \quad 2.0000$	按元素群运算， $u2*u2 \neq s$ $u2.*u2=s$ (见 2.3 节)
$v1=\expm(s)$	$v1 = 51.9690 \quad 74.7366$ $112.1048 \quad 164.0738$	按矩阵运算 可用 $\logm(v1)=s$ 检验
$v2=\exp(s)$	$v2 = 2.7183 \quad 7.3891$ $20.0855 \quad 54.5982$	按元素群运算 可用 $\log(v1)=s$ 检验

续表

输入语句	输出结果	说 明
logm(D)	1.2447 -0.9170 2.8255 1.6044 2.5760 -1.9132 -0.7539 1.1372 1.6724	按矩阵运算
log(D)	0 1.3863 1.9459 2.0794 1.6094 0.6931 1.0986 1.7918 -Inf	按元素群运算

注：此表中 $s = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $D = \begin{bmatrix} 1 & 4 & 7 \\ 8 & 5 & 2 \\ 3 & 6 & 0 \end{bmatrix}$ 。

2.2.4 矩阵结构形式的提取与变换

在做矩阵运算时，往往需要提取其中某些特殊结构的元素来组成新的矩阵，有时则要改变矩阵的排列。除了在 2.1 节讲过的提取行、列和将行列转置的语句之外，MATLAB 还提供了一些改变矩阵结构的函数。这些函数列于表 2.4 中，同时给出了相应的例子。

表 2.4 矩阵结构形式提取和变换语句

函 数 名	功能说明	语 句	结 果
fliplr	矩阵左右翻转	$B = \text{fliplr}(A)$	$B = \begin{bmatrix} 0 & 6 & 1 & 8 \\ 1 & 7 & 5 & 3 \\ 2 & 2 & 9 & 4 \end{bmatrix}$
flipud	矩阵上下翻转	$B = \text{flipud}(A)$	$B = \begin{bmatrix} 4 & 9 & 2 & 2 \\ 3 & 5 & 7 & 1 \\ 8 & 1 & 6 & 0 \end{bmatrix}$
reshape	阶数重组（元素总数不变）	$B = \text{reshape}(A, 2, 6)$	$B = \begin{bmatrix} 8 & 4 & 5 & 6 & 2 & 1 \\ 3 & 1 & 9 & 7 & 0 & 2 \end{bmatrix}$
rot90	矩阵整体逆时针旋转 90°	$B = \text{rot90}(A)$	$B = \begin{bmatrix} 0 & 1 & 2 \\ 6 & 7 & 2 \\ 1 & 5 & 9 \\ 8 & 3 & 4 \end{bmatrix}$
diag	提取或建立对角阵	$B = \text{diag}(A)$	$B = [8, 5, 2]$
tril	取矩阵的左下三角部分	$B = \text{tril}(A)$	$B = \begin{bmatrix} 8 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 4 & 9 & 2 & 0 \end{bmatrix}$

续表

函数名	功能说明	语 句	结 果
triu	取矩阵的右上三角部分	$B=\text{triu}(A)$	$B = \begin{bmatrix} 8 & 1 & 6 & 0 \\ 0 & 5 & 7 & 1 \\ 0 & 0 & 2 & 2 \end{bmatrix}$
:	将元素按列取出排成一列	$B=A(:)'$	$B=8\ 3\ 4\ 1\ 5\ 9\ 6\ 7\ 2\ 0\ 1\ 2$

注：表中 $A = \begin{bmatrix} 8 & 1 & 6 & 0 \\ 3 & 5 & 7 & 1 \\ 4 & 9 & 2 & 2 \end{bmatrix}$ 。

2.3 元素群运算

元素群运算能大大简化编程，提高运算的效率，是 MATLAB 优于其他许多语言的一个特色。

2.3.1 数组及其赋值

数组通常是指单行或单列的矩阵，一个 N 阶数组就是 $1 \times N$ 阶或 $N \times 1$ 阶矩阵。一个 N 阶数组可以表述一个 N 维向量。因为一个三维空间的向量可用它在三个坐标轴上的投影来表示，即 $v=[v_x, v_y, v_z]$ ，也即表示为三阶的数组。这个概念也可扩展到 N 维空间的向量。

在求某些函数值或曲线时，常常要设定自变量的一系列值，例如，设时间 t 从 0 到 1 之间，每隔 0.02 秒取一个点，共 51 个点，是 1×51 阶的数组。如果逐点给它赋值，将非常麻烦。MATLAB 提供了两种为等间隔数组赋值的简易方法。

(1) 用两个冒号组成等增量语句，其格式为： $t=[\text{初值}:\text{增量}:\text{终值}]$ 。例如：

输入 $t=[0:0.02:1]$

得 $t = 0 \quad 0.020 \quad 0.040 \quad \cdots \quad \cdots \quad 0.960 \quad 0.980 \quad 1.000$

此语句中的增量也可设为负值，此时初值要比终值大。例如：

输入 $z=10:-3:-5$

得 $z = 10 \quad 7 \quad 4 \quad 1 \quad -2 \quad -5$

当增量为 1 时，这个增量值可以略去，因而该语句只有一个冒号。例如：

输入 $k=1:6$

得 $k = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6$

(2) 用 linspace 函数。其格式为：

$\text{linspace}(\text{初值}, \text{终值}, \text{点数})$

输入 $\text{theta} = \text{linspace}(0, 2*\text{pi}, 9)$

得 `theta = 0 0.7854 1.5708 2.3562 3.1416 3.9270 4.7124 5.4978 6.2832`

在圆周上 0 和 2π 实际上是同一个点, 所以, 这个命令是把圆周分为 8 份。

有时要求自变量按等比级数赋值。在设频率轴时往往如此, 这时可用 `logspace` 函数。

例如:

输入 `w = logspace(0, 1, 11)`

得 `w = 1.0000 1.2589 1.5849 ... 6.3096 7.9433 10.0000`

它的意义是从 10 的 0 次幂到 1 次幂之间按幂等分 (即数是等比的) 为 11 个点。

2.3.2 元素群的四则运算和幂次运算

元素群运算也就是矩阵中所有元素按单个元素进行运算。为了与矩阵作为整体的运算符号相区别, 要在运算符 `*`、`/`、`\`、`^` 前加一个点符号 `.`, 以表示在做元素群运算。参与元素群运算的两个矩阵必须是同阶的 (只有标量除外, 它会自动扩展为同阶矩阵参与运算), 表 2.5 表示两个 1×3 阶的元素群运算的结果。因为数取得很简单, 可用心算加以检验。

表 2.5 简单的元素群运算

运 算 式	输出结果	思 考
<code>Z=X.*Y</code>	<code>Z = 4 10 18</code>	思考问题: <code>X*Y</code> 能成立吗
<code>Z=X.\Y</code>	<code>Z = 4.0000 2.5000 2.0000</code>	元素群有没有左除右除之分
<code>Z=X.^Y</code>	<code>Z = 1 32 729</code>	思考问题: <code>X^Y</code> 能成立吗
<code>Z=X.^2</code>	<code>Z = 1 4 9</code>	思考问题: <code>X^2</code> 能成立吗
<code>Z=2.^[X Y]</code>	<code>Z = 2 4 8 16 32 64</code>	思考问题: <code>2^[X Y]</code> 能成立吗

注: 表中 `X=[1, 2, 3]`; `Y=[4, 5, 6]`。

元素群的幂次运算也就是各个元素自行进行幂次运算, 对每个元素而言, 这种运算和对标量运算一样, 所以很容易判定它的正确性。

表 2.6 中参与元素群运算的是一个 3×3 阶的方阵, 这是为了便于比较矩阵中的元素运算和矩阵整体运算的不同 (因为非方阵是不能按整体做矩阵乘幂运算的)。从中也可以看出, 不能将元素群运算称为数组运算, 因为这里参加运算的是一个 3×3 阶矩阵而不是数组。

表 2.6 元素群和矩阵幂次运算的比较

输入算式	D	D^3	$D.^3$	$3.^D$
输出结果	1 4 7	627 636 510	1 64 343	3 81 2187
	8 5 2	804 957 516	512 125 8	6561 243 9
	3 6 0	486 612 441	27 216 0	27 729 1

注: 表中 $D = \begin{bmatrix} 1 & 4 & 7 \\ 8 & 5 & 2 \\ 3 & 6 & 0 \end{bmatrix}$ 。

特别注意 $D.^3$ 和 D^3 结果的不同。 $3.^D$ 与 3^D 也完全不同， 3^D 是合法运算，读者可自行实践并对其结果做出解释，不过这是比较难的题目，不作要求。

2.3.3 元素群的函数

前面已指出，大部分的 MATLAB 函数都适用于做元素群运算，只有专门说明的几个除外。那就是 $*$ 、 $/$ 、 \backslash 、 $^$ 运算符和 `sqrtm`、`expm`、`logm` 三个函数。表 2.7 基本函数库中的常用函数都可用于元素群运算，也即其自变量都可以是任意阶的矩阵。

表 2.7 基本函数库 (elfun) (未标注输入变元的为单输入单输出函数) (c)

三角 函数	<code>sin</code>	正弦	<code>cos</code>	余弦
	<code>tan</code>	正切	<code>asin</code>	反正弦
	<code>acos</code>	反余弦	<code>atan</code>	反正切
	<code>atan2(x, y)</code>	4 象限反正切	<code>Sinh</code>	双曲正弦
	<code>cosh</code>	双曲余弦	<code>tanh</code>	双曲正切
	<code>acosh</code>	反双曲余弦	<code>atanh</code>	双曲正切
	<code>asinh</code>	反双曲正弦	<code>sec</code>	正割
	<code>csc</code>	余割	<code>cot</code>	余切
	<code>asec</code>	反正割	<code>acsc</code>	余割
	<code>acot</code>	反余切	<code>sech</code>	双曲正割
	<code>csch</code>	双曲余割	<code>coth</code>	双曲正切
	<code>asech</code>	反双曲正割	<code>acsch</code>	反双曲余割
	<code>acoth</code>	反双曲正切		
指数 函数	<code>exp</code>	以 e 为底的指数	<code>log</code>	自然对数
	<code>log2</code>	以 2 为底的指数	<code>log10</code>	以 10 为底的对数
	<code>pow2</code>	2 的幂	<code>sqrt</code>	方根
	<code>nextpow2</code>	比输入数大而最近的 2 的幂		
复数	<code>abs</code>	绝对值和复数模值	<code>angle</code>	相角
	<code>real</code>	实部	<code>imag</code>	虚部
	<code>conj</code>	共轭复数	<code>isreal</code>	是实数时为真
	<code>unwrap</code>	去掉相角突变	<code>cplxpair</code>	按复数共轭对排序元素群
取整 函数	<code>round</code>	四舍五入为整数	<code>fix</code>	向 0 舍入为整数
	<code>floor</code>	向 $-\infty$ 舍入为整数	<code>ceil</code>	向 ∞ 舍入为整数
	<code>sign</code>	符号函数	<code>rem(a, b)</code>	a 整除 b, 求余数
	<code>mod(x, m)</code>	x 整除 m 取正余数		

下面的例子可以说明利用元素群运算的优越性。例如，要求列出一个三角函数表。这在 MATLAB 中只要两个语句：

输入 `x=[0:0.1:pi/4]';[x, sin(x), cos(x), tan(x)]`

第一条语句把数组 `x` 赋值，经转置后成为一个列向量。因为 `sin`、`cos`、`tan` 函数都对元素群有效，得出的都是同阶的列向量。第二条语句把 4 个列向量组成一个矩阵，并进行显示。

```
得  0          0          1.0000    0
    0.1000    0.0998    0.9950    0.1003
    0.2000    0.1987    0.9801    0.2027
    0.3000    0.2955    0.9553    0.3093
    0.4000    0.3894    0.9211    0.4228
    0.5000    0.4794    0.8776    0.5463
    0.6000    0.5646    0.8253    0.6841
    0.7000    0.6442    0.7648    0.8423
```

第一列是 `x`，以下各列依次是 `sin(x)`，`cos(x)`，`tan(x)`。如果要加一个表头，第二条语句可改成两条如下的显示语句：

```
disp('      x      sin(x)   cos(x)   tan(x) ')
disp([x, sin(x), cos(x), tan(x)])
```

`disp` 后括号内引号中的内容是直接显示的，放入空格就显示空格，放入汉字就显示汉字。后一句括号中没有引号，是变量名组成的矩阵，它就显示该矩阵中各变量的值。

2.4 逻辑判断及流程控制

2.4.1 关系运算

所谓关系运算是指两个元素之间数值的比较，一共有表 2.8 所示的 6 种可能。

表 2.8 关系运算

<	<=	>	>=	==	~=
小于	小于等于	大于	大于等于	等于	不等于

关系运算的结果只有两种可能，即 0 或 1。0 表示该关系式为“假”，即它不成立；1 表示该关系式为“真”，即该关系式是正确的。例如，输入关系式：

```
a = 2+2= =4
```

得 `a = 1`

注意，前面的单个等号表示赋值，后面的双等号则表示关系运算。式中 `2+2==4` 是关系运算，它的优先级高，要先算，算出的结果给 `a` 赋值，为了改善可读性，最好加上括号，写成 `a=(2+2==4)`，表明把括号内的关系式的结果给 `a` 赋值。

MATLAB 中的关系运算都适用于矩阵，它是对矩阵的各个元素进行元素群运算，因此两个相比较的矩阵必须有相同的阶数，输出的结果也是同阶矩阵。例如

输入 `A=magic(6)`

得 `A =`

35	1	6	26	19	24
3	32	7	21	23	25
31	9	2	22	27	20
8	28	33	17	10	15
30	5	34	12	14	16
4	36	29	13	18	1

要找到此矩阵中所有被 3 整除的元素，并在其位置上标以 1。可以用表 2.7 中的 `rem` 函数，`rem(A, 3)` 表示把 A 除以 3 的余数，余数为零就是整除。例如：

输入 `p=(rem(A, 3)==0)`

得 `p =`

0	0	1	0	0	1
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1
1	0	0	1	0	0
0	1	0	0	1	0

关系运算中还包括某些条件判断，例如判断矩阵元素中是否有 NaN、Inf 值，矩阵是否实数阵、稀疏阵或空阵等，它们不能直接用上述 6 种关系符简单地表述，MATLAB 把它们编成了专用的函数以备直接调用，如表 2.9 所示。

表 2.9 运算符和特殊字符库 (ops) (n)

数学及逻辑运算符	符 号	意 义	符 号	意 义	符 号	意 义
	+	加	-	减	*	矩阵乘
	\	矩阵左除	/	矩阵右除	^	矩阵乘幂
	.*	矩阵元素乘	./	矩阵元素除	.^	矩阵元素乘幂
	()	优先，下标	[]	矩阵，向量	:	整行（列）
	{}	输入参量		输出变量	:	等增量赋值
	.	小数点	..	母目录	...	行命令延续符
	,	语句分割符，显示	;	语句分割符，不显示	=	赋值符
	'	转置，引用	!	操作系统命令	%	注释符
	==	关系相等符	<>	关系大小符	~=	关系不等符
	&	逻辑与		逻辑或	~	逻辑非
	xor	异或	kron	Kronecker 积		

续表

逻辑 字符 检查	exist	检查变量或函数是否有定义	any	检查向量中是否有非零元素
	all	检查向量中元素是否全为非零	find	找到非零元素的序号
	isnan	元素为 NaN 时得 1	isinf	元素为 Inf 时得 1
	isfinite	元素为有限值时得 1	isempty	矩阵为空阵时得 1
	isreal	矩阵为实数阵时得 1	issparse	矩阵为稀疏阵时得 1
	isstr	为文本字符串时得 1	isglobal	变量为全局变量时得 1
位 运 算	bitand	按位求“与”	bitcmp	按位求“非”(补)
	bitor	按位求“或”	bitmax	最大浮点整数
	bitxor	按位求“异或”	bitset	设置位
	bitget	获取位	bitshift	按位移动
集合 运算	union	集合“合”	unique	去除集合中的重复元素
	intersect	集合“交”	setdiff	集合“差”
	setxor	集合“异或”	ismember	是集合中的元素时为真

在 MATLAB 的关系运算中还包括某些逻辑字符检查, 例如 $[j,k]=\text{find}(\mathbf{p})$ 给出 \mathbf{p} 矩阵中不为零的元素的两个下标, 左端没有或只有一个变量, 即 $\text{find}(\mathbf{p})$ 或 $\text{lp}=\text{find}(\mathbf{p})$ 给出 \mathbf{p} 矩阵中不为零的元素的序号。矩阵元素是按列排序号的, 先第 1 列, 再接第 2 列……依次排完后, 再确定它们的顺序号。一个 6×6 阶矩阵的 36 个元素的序号排列如表 2.10 所示。因此, 一个 $n\times m$ 阵中下标为 (j, k) 的元素, 其序号为 $l=(k-1)*n+j$ 。

输入 `lp=find(p)'`

得 `lp = 2 5 9 12 13 16 20 23 27 30 31 34`

表 2.10 矩阵元素的序号排法

1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35
6	12	18	24	30	36

可以看出, 这些序号确实对应于 \mathbf{p} 中的 1 元素。矩阵的序号 (index) 与下标 (subscript) 是一一对应的, 其变换关系可由表 2.1 中的 `ind2sub` (读作 index to subscript) 和 `sub2ind` 函数求得。

2.4.2 逻辑运算

逻辑量只能取 0 (假) 和 1 (真) 两个值。逻辑量的基本运算为与 (&)、或 (|) 和非 (~) 三种。有时也包括异或 (xor), 不过异或可以由三种基本运算组合而成。两个逻辑量

经此逻辑运算后的输出仍然是逻辑量，表示逻辑量的输入输出关系的表称为真值表，如表 2.11 所示。

表 2.11 基本逻辑运算的真值表

运 算	A = 0		A = 1	
	B = 0	B = 1	B = 0	B = 1
A & B	0	0	0	1
A B	0	1	1	1
~A	1	1	0	0
xor(A, B)	0	1	1	0

所有的算法语言中都有逻辑运算。MATLAB 的特点是将逻辑运算用于元素群，得出同阶的 0-1 矩阵。为了按列、按行判断一群元素的逻辑值，它又增加了两种对元素群的逻辑运算函数，即 all（全为真）和 any（不全为假）。

现在来看逻辑式 $\mathbf{u}=\mathbf{p}|\sim\mathbf{p}$ ，这是把 \mathbf{p} 和“非” \mathbf{p} 求“或”。 $\sim\mathbf{p}$ 就是把 \mathbf{p} 中的 0 元素换成 1，1 元素换成 0。在每个元素位置上，必有一个是 1，把 \mathbf{p} 和 $\sim\mathbf{p}$ “或”起来，一定是全 1 矩阵。

得 $\mathbf{u} =$

1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1

all 和 any 后的输入变量应为矩阵，它是按列运算的。从它们的定义可知

all(\mathbf{p}) = 0 0 0 0 0 0 （列中有一个元素为 0 即得 0）

all(\mathbf{u}) = 1 1 1 1 1 1 （列中元素为全 1 才得 1）

any(\mathbf{p}) = 1 1 1 1 1 1 （列中有一个元素为 1 即得 1）

2.4.3 流程控制语句

计算机程序通常都是从前到后逐条执行的，但有时也会根据实际情况，中途改变执行的次序，称为流程控制。MATLAB 共有四种流程控制语句。

1. If 语句

根据复杂程度，If 语句有三种形式：

- if（表达式）语句组 A，end

其流程如图 2.1（a）所示。执行到此语句时，计算机先检验 if 后的逻辑表达式，如为 1，它就执行语句组 A；如为 0，就跳过语句组 A，直接执行 end 后的后续语句。注意，这个 end 是绝不可少的，没有它，在表达式为 0 时，就找不到继续执行的程序入口。

- if（表达式 1）语句组 A，else 语句组 B，end

其流程如图 2.1 (b) 所示。执行到此语句时，计算机先检验 if 后的（逻辑）表达式，如为 1，它就执行语句组 A；如为 0，就执行语句组 B。**else** 用来标志语句组 B 的执行条件，同时也标志语句组 A 的结束（免去了 **end**）。同样，最后的 **end** 是不可少的；没有它，执行完语句组 A 后，会找不到进入后续程序的入口。

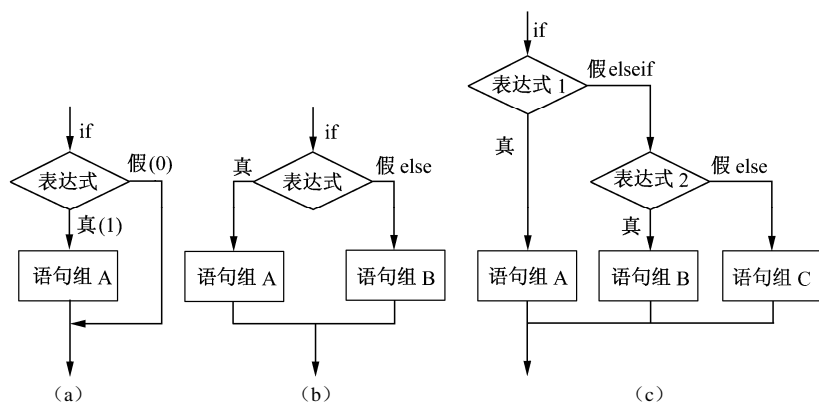


图 2.1 if 语句的 3 种程序结构形式

- **if** (表达式 1) 语句组 A, **elseif** (表达式 2) 语句组 B, **else** 语句组 C, **end**

其流程如图 2.1 (c) 所示。前两种形式的 if 语句都是两分支的程序结构，要实现两个以上分支的结构就得采用含 **elseif** 的结构。这里表示的是 3 分支的情况。在中间可加入多个 **elseif** 以形成多个分支。只是程序结构会显得冗长，MATLAB5.x 中的 **Switch** 语句可以用较简洁对称的形式实现多分支结构。

【例 2.1】 输入数 n ，判断其奇偶性。

程序如下：

```
n=input('n='), if rem(n, 2) ==0 A='even', else A='odd', end
```

运行此程序时，程序要求用户输入一个数，然后它判断该数是奇数还是偶数。所以它共有两个出口。实际上这个程序并不全面，如果用户根本未输入任何数就回车，程序会判断为 **odd**。请读者考虑其原因。为了使程序在用户无输入时自动中止，可以把程序改为：

```
if isempty(n) ==1 A='empty', elseif rem(n, 2) ==0 A='even', else A='odd',
end
```

实际上，这个程序仍不全面，它不能用于负数，请读者分析其原因。

2. while 语句

while 语句的结构形式为：

while (表达式) 语句组 A, **end**

其流程如图 2.2 所示。执行到此语句时，计算机先检验 **while** 后的逻辑表达式，如为 1，它就执行语句组 A；到 **end** 处后，它跳回到 **while** 的入口，再检验表达式；如还是 1，再执行语句组 A；周而复始，直到表达式不成立（结果为零）为止。此时就跳过语句组 A，直接执行 **end** 后的后续语句。与 if 语句的不同之处是它在分支中是循环地执行某个语句组，

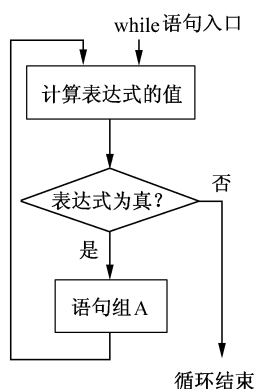


图 2.2 while 语句流程图

故称为循环语句。

【例 2.2】 求 MATLAB 中的最大实数。

解：设定一个数 x ，让它不断增大，直到 MATLAB 无法表示它的值，只能表示为 inf 。于是，可列出下列程序：

```
x=1; while x~=inf, x1=x; x=2*x; end, x1
```

其中，先设 $x=1$ ，进入 while 循环，只要 x 不等于 inf ，就把 x 加倍，直到 $x=\text{inf}$ 。如果把此时的 x 显示出来，它将是无穷大，不是题中要找的数。要找的是变为无穷大之前的最大数。因此，在对 x 加倍之前，把它存在 $x1$ 中，显示的 $x1$ 就是要求的最大数。运行这行程序得

```
x1 = 8.9885e+307
```

系统的最大浮点实数为 $(2-\epsilon)*2^{1023}$ （见表 2.1），其十进制形式为

```
realmax = 1.7977e+308
```

两者数量级接近，但还是相差将近一倍，这是因为每次都把 x 翻一番，故求得的数可能比最大数小不到一半。如果把程序中的 $x=2*x$ 改为 $x=1.1*x$ ，结果就会准确一些，得到

```
x1 = 1.783718732622142e+308
```

【例 2.3】 求 MATLAB 相对精度。

解：解的思路是让 y 不断减小，直至 MATLAB 分不出 $1+y$ 与 1 的差别为止。其程序为

```
y=1;while 1+y>1, y1=y;y=y/2;end, y1
```

结果为

```
y1 = 2.220446049250313e-016
```

与 MATLAB 内部给出的浮点数相对精度 2^{-52} （见表 2.1）的十进制数相同。

3. for 语句

for 语句的结构形式为

for k= 初值: 增量: 终值 语句组 A, end

即它把语句组 A 反复执行 N 次。在每次执行时程序中的 k 值不同。

$N = 1 + (\text{终值} - \text{初值}) / \text{增量}$

【例 2.4】 用 for 语句求三角函数表。

程序如下：

```
for x=0:0.1:pi/4 disp([x, sin(x), cos(x), tan(x)]), end
```

所得的结果将和前面的答案相同。这也可以看出，MATLAB 的元素群运算功能与一个 for 循环相当。由于它不需要每次检验表达式，运算速度比 for 语句快得多。但是，不能认

为它可全部取代 for 语句，由下例可以看出。

【例 2.5】 列出构成 Hilbert 矩阵的程序。

完成这个程序需要两重循环：

```
n=input('n='), format rat
for i=1:n, for j=1:n, h(i, j) =1/(i+j-1);end, end, h
```

执行时，先按提示输入 n，比如输入 5。

结果为：

```
h =    1          1/2          1/3          1/4          1/5
      1/2          1/3          1/4          1/5          1/6
      1/3          1/4          1/5          1/6          1/7
      1/4          1/5          1/6          1/7          1/8
      1/5          1/6          1/7          1/8          1/9
```

为了改善可读性，对于流程控制语句，最好用缩进的方法写程序。本例中应写成：

```
format rat, n=input('n='),
for i=1:n
    for j=1:n
        h(i, j) =1/(i+j-1);
    end
end
h
```

由于现在是在 MATLAB 命令窗中直接输入程序，因此，不得不把它写在一行中。此时要注意，在 if, for, while 与表达式之间应留空格，在表达式与语句组之间必须用空格或逗号分隔，而在语句组的后面，必须要用逗号或分号来与 end 或 else 相分隔。否则，MATLAB 会显示出错信息并中止运行。

break 是中止循环的命令，在循环语句中，可用它在一定条件下跳出循环，它是常常用到的。在多重循环中，break 只能使程序跳出包含它的最内部的那个循环。

4. Switch 语句

Switch-case-otherwise 语句是一种均衡实现的多分支语句，其基本语言结构可表示为：

```
switch 表达式（标量或字符串）
case 值1
    语句组 A
case 值2
    语句组 B
.....
otherwise
```

```
语句组 N  
end
```

当表达式的值（或字符串）与某 case 语句中的值（或字符串）相同时，它就执行该 case 语句后的语句组，然后直接跳到终点的 end。case 语句可以有 N-1 个，如果没有任何一个 case 值能与表达式值相符，则将执行 otherwise 后面的语句组 N。

例如，判断输入数 n 的奇、偶、空的程序可用 Switch 语句写成：

```
switch mod(n, 2), case 1, A='奇', case 0, A='偶', otherwise, A='空', end
```

注意，把它写成单行命令时的标点格式，其中有些逗号可以用分号代替，但不得省略。另外，为了包含负数中的奇数，将前面例中的 rem 改为 mod，读者可从 rem(-3, 2) 和 mod(-3, 2) 的差别得知这样做的原因。在正式写程序时，case 语句必须写在行首，以增强程序的可读性。

2.5 基本绘图方法

MATLAB 可以根据给出的数据，用绘图命令在屏幕上画出其图形，通过图形对科学计算进行描述。这是 MATLAB 独有的优于其他语言的特色。它可选择多种类型的绘图坐标，可以对图形加标号、加标题或画上网状标线。这些命令属于 graph2d 函数库，另外，还有一些命令可用于屏幕控制，坐标比例选取以及在打印机上进行硬拷贝，等等。这些命令放在 graphics 函数库中。三维及颜色绘图命令放在 graph3d 函数库中。还有一些特殊绘图命令放在 specgraph 函数库中。本书不可能介绍所有的命令，但本书会涉及大部分命令，下面分别进行讨论。

2.5.1 直角坐标中的两维曲线

plot 命令用来绘制 X-Y 坐标中的曲线。它是一个功能很强的命令，输入变量不同，可以产生很多不同的结果。

1. Plot (y) —— 输入一个数组的情况

如果 y 是一个数组，函数 plot (y) 给出线性直角坐标的二维图，以 y 中元素的下标作为 X 坐标，y 中元素的值作为 Y 坐标，一一对应画在 X-Y 坐标平面图上，而且将各点以直线相连。例如，要画出 10 个随机数的曲线。可列出：

```
y=5*(rand(1, 10) - .5)  
y = -1.4052 -2.2648 0.8943 0.8965 2.1735 -0.5825 0.0971 1.6548 -2.3271  
-2.2327
```

由 Rand 函数产生的随机数的最大值为 1，最小数为 0，平均值为 0.5。所以 y 的最大值为 2.5，最小值为-2.5，平均值为 0。输入 plot(y)，MATLAB 会产生一个图形窗，自动规

定最合适的坐标比例绘图。X 方向是横坐标，从 1 到 10，Y 方向范围则是 -4 到 4，并自动标出刻度。可以用 `title` 命令给图加上标题，用 `xlabel`, `ylabel` 命令给坐标轴加上说明，用 `text` 或 `gtext` 命令可在图上任何位置加标注，也可用 `grid` 命令在图上打上坐标网格线。

```
输入 title('my first plot')
xlabel('x'), ylabel('Y')
grid
```

这时形成如图 2.3 所示的图。

2. Plot (x, y) —— 输入两个数组的情况

如果数组 `x` 和 `y` 具有相同长度，命令 `plot(x, y)` 将绘出以 `x` 元素为横坐标，`y` 元素为纵坐标的曲线。例如，设 `t` 为时间数组 `t=0: 0.5: 4*pi`，`y` 是一个随 `t` 做衰减振荡的变量，`y=exp(-0.1*t).*sin(t)`，则 `plot(t, y)` 就以 `t` 为横坐标，`y` 为纵坐标画曲线，如图 2.4 中的实线曲线。若设 `y1=exp(-0.1*t).*sin(t+1)`，则由 `plot(t, y1, ':')` 画出的曲线，其正弦波的相位超前了 1 弧度。因此，其波形如图 2.4 中的虚线曲线所示。实际上，在绘制第二条曲线时，如不加别的命令，第一条曲线就自动消失了。不会有两根曲线同在一张图中出现。为了在一张图中绘制多条曲线，要用后面所说的办法。

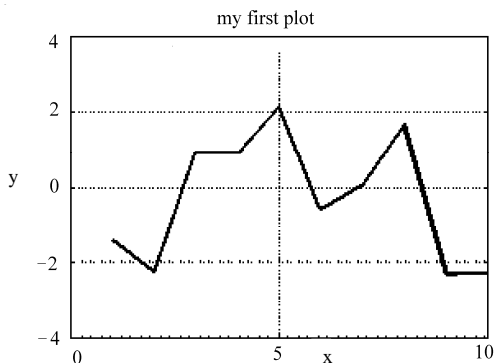


图 2.3 第一张简单的随机数图

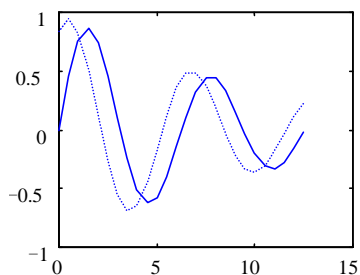


图 2.4 两根曲线画在同一图上

2.5.2 线型、点型和颜色

MATLAB 会自动设定所画曲线的颜色和线型。如果用户对线型的默认值不满意，可以用命令控制线型。也可以根据需要选取不同的数据点的标记。为了设定线型，在输入变量组的后面，加一个引号，在引号内部放入线型和颜色的标识符，例如：

```
plot(x, y, '*b')
```

这样绘出的图线，其数据点处均用 * 做蓝色标记，而各点之间不再连以直线。

```
plot(x1, y1, ':y'), plot(x2, y2, '+r')
```

绘出的第一条曲线是黄色的点线，第二条曲线的数据点标记为红色的“+”号。其他线型、点型和颜色见表 2.12。

表 2.12 线型、点型和颜色

标识符	颜色	标识符	线型和点型
y	黄	.	点
m	品红	o	圆圈
c	青	x	x 号
r	红	+	+号
g	绿	-	实线
b	蓝	*	星号
w	白	:	虚线
k	黑	-.	点画线
		--	长画线

2.5.3 多条曲线的绘制

在一张图上画多根曲线有 4 种方法，其中第 4 种方法是 MATLAB 5.x 中新增加的。

1. 用 `plot(t, [y1, y2, ...])` 命令

该语句中 `t` 是向量，`y=[y1, y2, ...]` 是矩阵，若 `t` 是列（行）向量，则 `y` 的列（行）长与 `t` 长度相同。`y` 的行（列）数就是曲线的根数。例如：

输入 `plot(t, [y; y1])`

就得出图 2.4 中的曲线。它会自动给曲线以不同的颜色。这种方法要求所有的输出量有同样的长度和同样的自变量向量。另外，它不利于用户自行设定线型和颜色。

2. 用 `hold` 命令

在画完前一张图后用 `hold` 命令保持住，再画下一条曲线。例如：

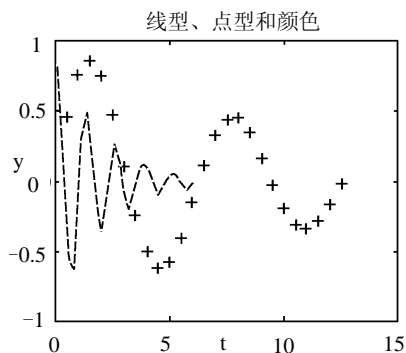


图 2.5 两组长度不同的 `[t, y]` 数据画在同一图上

输入 `plot(t, y), hold on, plot(t, y1, 'g')`

执行此命令时，图形窗产生第一幅图形，同时，命令屏幕显示 `Current plot held`，图形处于保持状态。再执行 `plot(t, y1, 'g')`，就把第二幅图以绿色的曲线叠合在同一张图上。

用这种方法时两张图的变量长度可以各不相同。只要每张图自己的自变量和因变量同长即可。例如，再给一组数据 `[t2, y2]`，其点数比 `[t, y]` 多，但占的时间却短。

输入

```
t2=0:.2:2*pi;y2=exp(-0.5*t2).*sin(5*t2+1); plot(t2, y2)
```

得出的图形为图 2.5 中较短的那条曲线（但线型不同）。用这种方法时，需要注意两点：（1）注意第一张图的坐标要适当，以保证能看清第二张图。因为用第一种方法时，坐标系是系统自动按多根曲线的数据综合选取的，不会有选择不当的问题。（2）注意及时解除保持状态，即输入 `hold off`；否则，以后的图都会叠加在此图上，造成混乱。

3. 在 plot 后使用多输入变量

在 `plot` 后使用多输入变量所用的语句为：

```
plot(x1, y1, x2, y2, ..., xn, yn)
```

其中，`x1`, `y1`, `x2`, `y2` 等分别为数组对。每一对 `X-Y` 数组可以绘出一条图线，这样就可以在一张图上画出多条图线，每一组数组对的长度可以不同，在其后面都可加线型标志符。例如：

输入

```
plot(t, y, '+g', t2, y2, ': r')
title('线型, 点型和颜色')
xlabel('时间'), ylabel('Y')
```

执行这些语句就得到图 2.5。一根图线在数据点处用绿色的虚线做标记，另一根图线用红色的+号做标记。注意，这里用的是汉字标注，MATLAB 也照样把汉字标在图上。因为在引号中的内容，MATLAB 只作为一种代码来传递。

4. 用 plotyy 命令

`plotyy` 设有两个纵坐标，以便绘制两个 `y` 尺度不同的变量，但 `x` 仍只用同一个比例尺，例如：

输入 `y3=5*y2; plotyy(t, y', t2, y3)`

就得到图 2.6。其中，左纵坐标是对 `y` 的，而右纵坐标是对 `y3` 的，纵坐标和曲线的标注可用 `gtext` 命令：

```
grid, gtext('t, t2')
gtext('y'), gtext('y3')
```

`gtext` 命令用鼠标拖动来确定标注文字的位置，用起来比较方便。

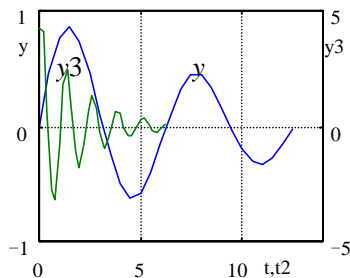


图 2.6 双纵坐标绘图

2.5.4 屏幕控制和其他二维绘图

1. 图形屏幕控制命令（参看表 2.13）

图形屏幕可以开或关，可以开几处图形窗，也可以在一个图形窗内画出几幅分图，几

幅分图也可用不同的坐标。以下几种命令可以实现图形窗口间的转换和清除。

- **figure**: 打开图形窗口。MATLAB 中的第 1 幅图随 **plot** 命令自动打开, 以后的 **plot** 命令都画在同一张图上。如要画在另一张新图上, 就要用 **figure** 命令打开新的图形窗口。有了顺序为 1, 2, 3, ... 的几个图形窗后, 再用 **plot** 语句, 就要指明画在哪张图上, 即输入 **figure_i**; 表示打开第 i 幅图。否则, 所有的图都会画在最后显示的那幅图上。
- **clf**: 清除当前图形窗的内容 (也可用 **clg**, 但以后将被淘汰)。
- **hold**: 保持当前图形窗的内容, 再输入 **hold**, 就解除保持状态。这种拉线开关式的控制有时会造成混乱, 可以用 **hold on** 和 **hold off** 命令以得到确定的状态。
- **close**: 关闭当前图形窗。
- **close all**: 关闭所有图形窗。
- **subplot (n, m, p)** 命令: 将图形窗口分为 $n \times m$ 个子图, 在第 p 个子图处绘制图形。

表 2.13 通用图形函数 (graphics) (h)

图形窗的控制	figure	创建图形窗	shg	显示图形
	gcf	获取当前图形窗的句柄	refresh	刷新图形
	clf	清除当前图形窗	close	关闭图形窗
轴系的控制	axes	在任意位置创建坐标系	ishold	保持当前图形状态为真
	gca	获取当前坐标系的句柄	box*	形成轴系方向
	cla	清除当前坐标系		
图形对象	line	创建直线	surface	创建曲面
	patch	创建图形填充块	light	创建照明
	image	创建图像		
图形句柄操作	set	设置对象特性	gcbo	获得回叫对象的句柄
	get	获得对象特性	gcbf	获得回叫图形的句柄
	reset	复位对象特性	drawnow	直接等待图形事件
	delete	删除对象	findobj	寻找具有特定值的对象
	gco	获得当前对象的句柄	copyobj	为图形对象及其子项作硬拷贝
工具	closereq	请求关闭图形窗	ishandle	是图形句柄时为真
	newplot	说明 NextPlot 的 M 文件		
杂项	ginput	从鼠标作图形输入	uiputfile	给出存储文件的对话框
	graymon	设定图形窗灰度监视器	uigetfile	给出询问文件名的对话框
	rbbox	涂抹块	whitebg	设定图形窗背景色
	rotate	围绕指定方向旋转对象	zoom	二维图形的放大和缩小
	terminal	设定图形终端类型	warndlg	警告对话框

2. 其他二维绘图命令 (参看表 2.14)

在线性直角坐标系中绘出形式图的命令有 **stem** (绘脉冲图)、**stairs** (绘阶梯图)、**bar**

(绘条形图)、`errorbar` (绘误差条形图)、`hist` (绘直方图) 等。这些函数用法与 `plot` 相仿，但没有多输入变量形式。`fill (t, y, '颜色标注符')` 在曲线和坐标轴之间的封闭区填以指定的颜色。

下列程序把画面分成 4 个。

```
subplot(2, 2, 1), stem(t, y)
title('stem(t, y)'), pause
subplot(2, 2, 2), stairs(t, y)
title('stairs(t, y)'), pause,
subplot(2, 2, 3), bar(t, y)
title('bar(t, y)'), pause
subplot(2, 2, 4), fill(t, y, 'r')
title('fill(t, y, ' 'r' ' ')')
```

其运行的结果如图 2.7 所示。读者不难从中弄清这几条绘图命令的意义。程序中最后一行，在 `r` 前后的引号写成两个引号，这是因为它是处在 `title` 后的引号内。`MATLAB` 规定，这种引号必须写成两个，以免混淆。

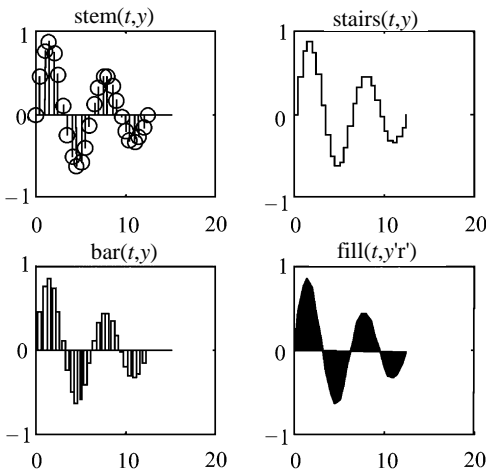


图 2.7 同一函数的几种不同的绘制形式

再输入 `subplot (1, 1, 1)` 命令可取消子图，转回全屏幕绘图。

在对数直角坐标系中的绘图命令有 `loglog`, `semilogx`, `semilogy` 等，在极坐标系中的绘图命令有 `polar`。它们的用法与 `plot` 的基本用法相同。只是数据将画在不同类的坐标系上，可参看表 2.14。

表 2.14 二维图形函数库 (graph2d) (p)

基本	<code>plot</code>	线性 X-Y 坐标绘图	<code>polar</code>	极坐标绘图
X-Y	<code>loglog</code>	双对数 X-Y 坐标绘图	<code>plotyy</code>	用左、右两种 Y 坐标画图
图形	<code>semilogx</code>	半对数 X 坐标绘图	<code>semilogy</code>	半对数 Y 坐标绘图
坐标	<code>axis</code>	控制坐标轴比例和外观	<code>subplot</code>	在平铺位置建立图形轴系
控制	<code>hold</code>	保持当前图形		
图形	<code>title</code>	标出图名 (适用于三维图形)	<code>gtext</code>	用鼠标定位文字
注释	<code>xlabel</code>	X 轴标注 (适用于三维图形)	<code>legend</code>	标注图例
	<code>ylabel</code>	Y 轴标注 (适用于三维图形)	<code>grid</code>	图上加坐标网格 (适用于三维)
	<code>text</code>	在图上标文字 (适用于三维)		
打印	<code>print</code>	打印图形或把图存为 M 文件	<code>orient</code>	设定打印纸方向
	<code>printopt</code>	打印机默认选项		

- `polar (theta, rho)` 为极坐标绘图，以角度 `theta` 为一个坐标，单位是弧度，另外一个为矢径 `rho`。在其后使用 `grid` 命令，可以绘出网状极坐标线。这个函数没有多输入变量形式。

- `loglog` 绘出以 \log_{10} – \log_{10} 为坐标刻度的对数图。
- `semilogx` 使用半对数刻度绘图, X 轴为 \log_{10} 刻度, Y 轴为线性刻度。
- `semilogy` 使用半对数刻度绘图, Y 轴为 \log_{10} 刻度, X 轴为线性刻度。

3. 虚数的绘图

当 `plot(z)` 中的 z 为复数单变量时 (即含有非零的虚部), MATLAB 把复数的实部作为 X 坐标, 虚部作为 Y 坐标进行绘图, 即相当于 `plot(real(z), imag(z))`。如果是双变量, 如 `plot(t, z)`, 则 z 中的虚数部分将被丢弃。要在复平面内绘出多条图线, 必须用 `hold` 命令, 或把多根曲线的实部和虚部明确地写出, 作为 `plot` 函数的输入变元, 即:

```
plot(real(z1), imag(z1), real(z2), imag(z2)).
```

例如, 要绘制 $z = \exp((-1+i)*t)$ 的复数图形, 列出下面的程序:

```
figure(2)
z=exp((-0.1+i)*t);
subplot(2,2,1)
plot(z), pause
title('复数绘图 plot(z)')
subplot(2,2,2)
plot(t, z), pause
title('复数绘图 plot(t, z)')
subplot(2,2,3), polar(angle(z), abs(z))
title('polar(angle(z), abs(z))')
subplot(2,2,4), semilogx(t, z)
title('semilogx(t, z)')
```

运行程序的结果如图 2.8 所示。其中, 第一个子图画出了复数图形; 第二个子图只画了 z 的实部随 t 的变化规律, 第三个子图是用极坐标绘制的复数曲线; 第四个子图说明了半对数坐标绘图的结果。

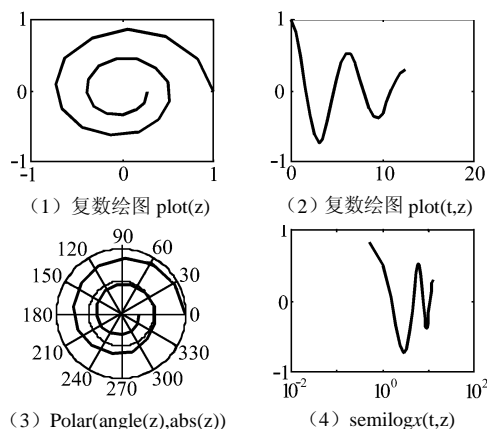


图 2.8 复数绘图及其他坐标轴绘图

4. 坐标比例和尺寸的设定——axis 命令

MATLAB 有根据输入数据自动设定坐标比例的功能。但在有些情况下，用户需要自行设定坐标比例并选择图形边界范围，这时可用 axis 命令。它有多种用法，由输入变量的不同而定。

$V=axis$ ，返回值为当前图形边界的 4 元行向量，即 $V=[xmin, xmax, ymin, ymax]$ ，如果当前图形是三维的，则返回值将是三维坐标边界的 6 元行向量。

$axis(V)$ （其中 V 是一个 4 元向量），将坐标轴设定在 V 规定的范围内。

axis 的另外一个功能是控制图形的纵横比。Axis('square') 或 axis('equal') 使屏幕上 x 与 y 的比例尺相同，在这种方式下，斜率 1 的直线的倾角为 45° ，对于程序：

```
z=0:0.1:2*pi;x=sin(z);y=cos(z);  
subplot(1, 2, 1), plot(x, y), subplot(1, 2, 2), plot(x, y), axis('equal')
```

虽然数据得出的是圆，但由于屏幕本身长宽不等，第一个子图得出的是椭圆。第二个子图由于函数 axis('equal') 的作用，就成为圆（如图 2.9 所示）。axis('normal') 将恢复正常的纵横尺寸比。

输入 $v=axis$
可得 $v = \quad -1 \quad 1 \quad -1 \quad 1$

axis 命令的功能非常丰富，这里只介绍了一部分。要知道详情，可使用命令 help axis。

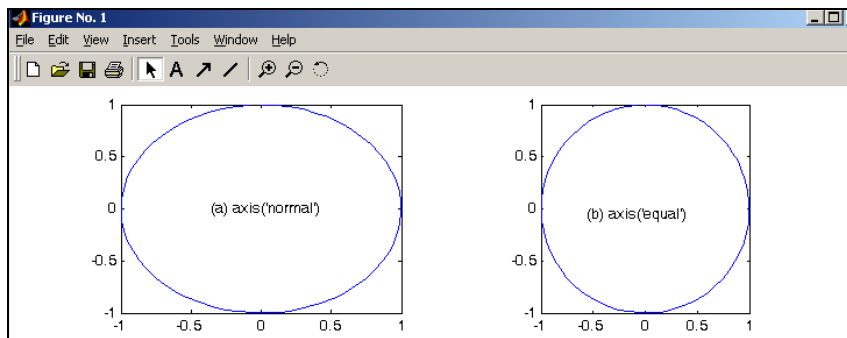


图 2.9 axis('equal') 命令的作用

5. 图形窗中的直接编辑

除了用行命令来给图形窗设定坐标，添加文字之外，可以直接用图形窗编辑功能。图 2.9 中除图形外，还把图形窗顶部的菜单和按钮都显示出来了。着重看按钮，从左至右，前四个按钮不必说了，第五个是“选择”按钮，激活它（变白）后鼠标就可在图中选择特定的对象，例如选整个子图，图中的一根曲线或一组文字。然后可以移动它的位置，也可以单击菜单上的 edit 项对此对象进行编辑修改；第六个是“文字”按钮，激活它可以输入文字，包括英文、汉字和拉丁字母；第七个是“箭头”按钮，激活它可以产生标注的箭头；第八个是“直线”按钮，激活它可以在图上产生直线；再靠右，一个是放大，一个是缩小；

最右边的按钮是旋转。读者可以自行摸索试验以掌握它们的功能和用法，深入一些问题需参阅手册“Using MATLAB Graphics”。

2.5.5 三维曲线和曲面

1. 空间曲线绘制——plot3

其用法大体与 plot 相仿。格式为：

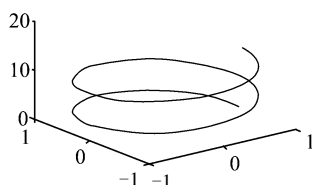


图 2.10 空间螺旋曲线

```
plot3(x,y,z,'s')。
```

其中 s 为线型颜色符。例如输入

```
z=0:0.1:4*pi;x=cos(z);y=sin(z);plot3(x,y,z)
```

得出的将是一条空间螺旋线，如图 2.10 所示。

2. 空间曲面的绘制

函数 mesh 和 surf 用来绘制三维曲面。三维曲面方程应有 x, y 两个自变量，因此，先在 X - Y 平面上建立网格坐标，每一个网格点上的数据 Z 坐标就定义了曲面上的点。通过直线（mesh）或小平面（surf）连接相邻的点就构成了三维曲面。

mesh 函数可以把一个大矩阵形象化地表示出来。例如，函数 $\text{sinc}(r) = \sin(r)/r$ （其中 r 是 X - Y 平面上的向径）的立体图形是很生动的。可用下面的方法来绘制，程序如下：

```
x=-8:0.5:8;y=x'; % 生成一维的自变量数组
X=ones(size(y))*x;Y=y*ones(size(x)); % 生成二维的自变量平面
R=sqrt(X.*X+Y.*Y); z=sin(R)./R; % 生成因变量
mesh(z), pause % 画三维曲面
```

第 1 行命令定义了函数计算的 x, y 取值范围，每一个方向有 33 个样本点；第 2 行命令建立了共有 $33 \times 33 = 1089$ 个网格点的坐标矩阵 X 和 Y ，形成了 33×33 网格的矩阵；第 3 行程序表示数据点到原点的距离，并求得 sinc 函数值，最后用 mesh 函数绘出图形。

图 2.11 画出各点的 X 坐标，即竖线；画出各点的 Y 坐标，即横线。在所关心的 z 的定义域内列出 X, Y 后，即可进行函数的计算和绘图。MATLAB 也提供了生成网格点坐标的专用函数 meshgrid。用这个函数时，上述的两行程序可简化为一条语句：

```
[X,Y] = meshgrid(-8:0.5:8, -8:0.5:8);
```

执行了第 3 行程序后将得出以下的警告：

```
Warning: Divide by zero
```

即出现了被零除的运算。实际上，这发生在 $R=0$ （即原点）处，该处 $\sin(R)$ 也为零，所以得到 NaN。产生的三维曲线如图 2.12 所示，在 $R=0$ 处缺掉一个点，因为 NaN 是无法画出的。从这里可以看出 IEEE 算法的优越性。如果照过去的方法，出现零做除数这种非

法运算，就要退出系统，取消全部结果，那就得不到这个漂亮的图形，其实非法运算只是1089个点中的一个点。不能“攻其一点，不及其余”。解决这问题的方法也很简单，只要把样本点移离原点即可。为此把程序改为：

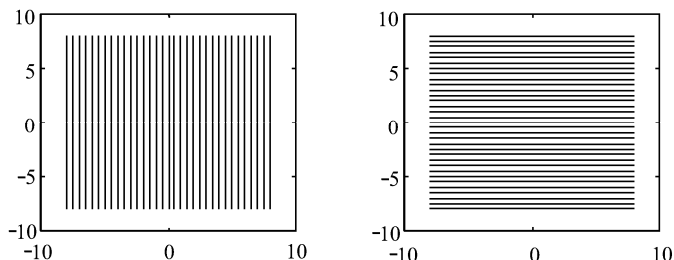


图 2.11 由 meshgrid 函数求出的 X , Y 矩阵对应的 X , Y 坐标

```
R=sqrt(X.*X+Y.*Y) +eps; z=sin(R)./R; figure(1), mesh(z)
```

即把原来的 R 值移动一个极小的数值 eps ，运行就没有问题，而图上不再有缺掉的点了。把上式中的 R 改成 $\text{abs}(x)+\text{abs}(y)$ （称为一范数， $\text{sqrt}(x^2+y^2)$ 称为二范数）。

输入

```
R=abs(X) +abs(Y) +eps; z1=sin(R)./R;
figure(2), surf(z1)
```

得出的三维曲面如图 2.13 所示。

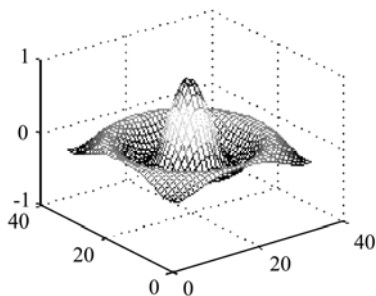


图 2.12 $z=\sin(R)/R$ 的三维曲面图

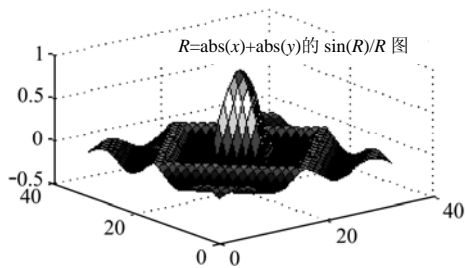


图 2.13 当 R 取一范数时 $z=\sin(R)/R$ 的三维曲面图

3. 其他三维图形绘制命令

三维图形绘制命令中有一组命令属于外观变换，具体有以下几种。

- **view**（方位角，俯仰角）可以变换立体图的视角，例如：

输入 `view(20, 0)`

就得到另一种三维图形，如图 2.14 中的右下子图（方位角，俯仰角的默认值为 $[37^\circ, 30^\circ]$ ）所示。

- **shading flat** 或 **shading interp** 可把表面上的小格平滑掉，使曲面成为光滑表面。**Shading faceted** 是默认状态，它使表面上有小格。
- **rotate3d** 命令，执行此命令后，用户可以用鼠标拖动立体图形做空间连续转动。

另外还有一组等高线绘制命令——`contour` 命令，把曲面的等高线投影在 X - Y 平面上，也就是普通地图中的画法。`contour3` 在三维立体图中画出等高线，这些等高线就像云那样浮在相应的高度上。下列程序使用了三维绘图库中的一些命令，其结果可从图 2.14 中看出。有关彩色的命令在后面讨论。

```
subplot(2, 2, 1), R=sqrt(X.^2+Y.*Y); z=sin(R)./R; meshc(z), pause
title('meshc(z), shading flat'), shading flat
subplot(2, 2, 2), R=sqrt(X.^2+Y.*Y) +eps; z=sin(R)./R; meshz(z), pause
title('meshz(z), shading interp'), shading interp
subplot(2, 2, 3), R=abs(X) +abs(Y) +eps; z1=sin(R)./R; surfc(z1), pause
title('surfc(z1), shading flat'), shading flat, %colormap(gray)
subplot(2, 2, 4); surf1(z1), view(20, 0)
title('surf1(z1), view(20, 0)')
```

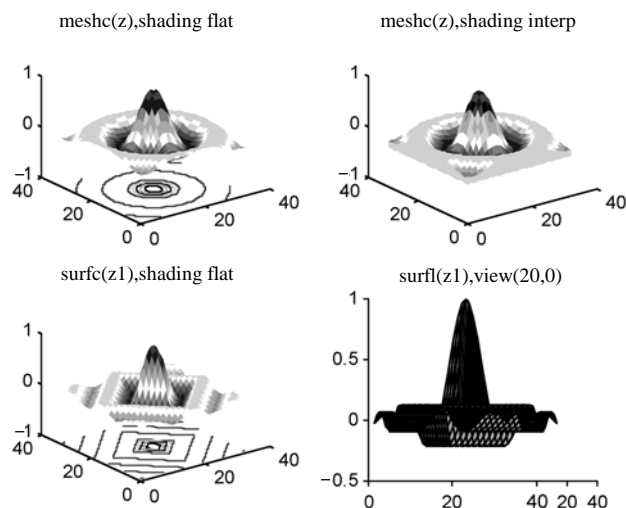


图 2.14 mesh 和 surf 命令的其他形式

2.5.6 特殊图形和动画

表 2.15 列出了 MATLAB 中的一些特殊图形函数，其中一部分是各种不同学科和领域中用到的特殊二维和三维图形。例如，前面提到过的 `stem`、`stairs` 等，`pie` 和 `bar` 是在管理科学中常用的饼图和条形图，`compass` 是电路中常用的相量图，在应用篇中还会介绍，读者可以自行试用。另一部分是前面已介绍过的等高线图形，此处不多占篇幅了。

在这里需要着重介绍的是 MATLAB 的动画命令。它总共只有三条命令：`moviein`，`getframe` 和 `movie`。用 `getframe` 把 MATLAB 产生的图形存储下来，每个图形成一个很大的列向量，再用 N 行这样的列保存 N 幅图，成为一个大矩阵。最后用 `movie` 命令把它们连起来重放，就可以产生动画效果。`Moviein` 用来预留存储空间以加快运行的速度（不用也可以）。下面是 MATLAB 手册上提供的一个漂亮的动画程序。

表 2.15 特殊图形库 (specgraph) (u)

特殊 的二 维图 形	area	填满绘图区域	feather	羽状图
	bar	条形图	fill	填满二维多边形
	barh	水平条形图	pareto	Pareto 图
	bar3	三维条形图	pie	饼图
	bar3h	三维水平条形图	plotmatrix	矩阵散布图
	compass	极坐标向量图	ribbon	画成三维中的色带
	comet	彗星轨迹图	stem	离散序列绘图
	errorbar	误差条图	stairs	阶梯图
等高 线图 形	contour	等高线图	pcolor	伪彩色图.
	contourf	填充的等高线图	quiver	箭头图
	contour3	三维等高线图	voronoi	Voronoi 图
	clabel	等高线图标出字符		
特 殊 的 三 维 图 形	comet3	三维彗星轨迹图	slice	实体切片图
	meshc	三维曲面与等高线组合图	surf	三维曲面与等高线组合图
	meshz	带帘的三维曲面	trisurf	三角表面图
	pie3	三维饼图	trimesh	三角网状表面图
	stem3	三维 stem 图	waterfall	瀑布图
	quiver3	三维 quiver 图		
图像 显示	image	显示图像	imread	从图形文件读出图像
	imagesc	缩放数据并作为图像显示	imwrite	把图像写入图形文件
	colormap	颜色查找表	imfinfo	关于图形文件的信息
电影 和动 画	capture	从屏幕抓取图形文件	rotate	绕给定方向旋转对象
	moviein	初始化电影帧存储器	frame2im	把电影帧转换为索引图像
	getframe	获取电影帧	im2frame	把索引图像转换为电影帧
	movie	重放录下的电影帧		
实体	cylinder	生成圆柱体	sphere	生成球体

```

axis equal,           % 因为产生的图形是圆形，故把坐标设成相等比例
M = moviein(16);      % 为变量 M 预留 16 幅图的存储空间
for j=1:16            % 做 16 次循环
    plot(fft(eye(j+16))); % 画图
    M(:, j) = getframe;   % 依次存入 M 中
end

```

运行完这几句程序后，16 幅画面（每幅用 16858*8=134864 个字节）就放在矩阵 M 中；再输入

```
movie(M, 30)
```

MATLAB 就把 M 中图形播放 30 遍，读者可自行在计算机上实践。

2.5.7 彩色、光照和图像

为了更好地显示图形，特别是空间图形，MATLAB 使用了彩色和光照。颜色提供了三维图形中的第四维坐标，扩展了图形表达的能力，光照又进一步改善了视觉效果，这也是 MATLAB 的一个重要特色。但因为彩色印刷的成本太高，在本书中不便展开，建议读者自己在计算机上实践。

在三维曲面绘图命令中，加入第四维变元，例如，`mesh(x, y, z, w)`，则 w 的大小就用颜色表示，即在坐标值为 (x, y, z) 的点上，赋予对应于 w 值的颜色。颜色与 w 值的一一对应关系，用彩色条 (`colorbar`) 来表示，用命令 `colorbar` 可得到这个关系，它将在已有的图形右侧，加上一条垂直的彩色条，并标以对应的 w 值。

如果在三维曲面绘图命令中，没有第四维变元，则颜色轴将与 Z 轴一致，有一一对应的关系。例如：

输入

```
[x, y] = meshgrid([-2:2:2]);
z = x.*exp(-x.^2-y.^2);
surf(x, y, z), colorbar
```

所得图形及彩色条如图 2.15 (a) 所示，可见 z 最大处为深红色， z 最小处为深蓝色。如果把末行改成

```
surf(x, y, z, gradient(z)), colorbar
```

则彩色轴将表示曲面的梯度，也就是产生第四维的坐标，如图 2.15 (b) 所示。可以看出，在峰点谷点之间的斜率最大处有最深的颜色，负斜率处为深蓝色。

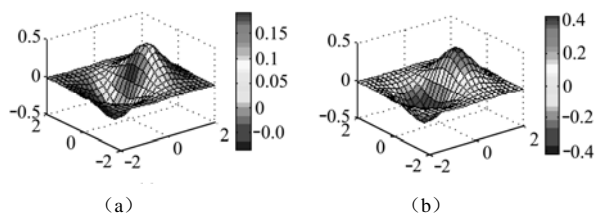


图 2.15 颜色坐标分别表示第三维 (z) 和第四维 (梯度)

彩色条分为 64 个等级，其颜色排序的默认值为 `hsv`，但可由用户自行修改设定为 `hot`, `cool`, `gray`, `copper`, `pink`, `jet`, `prism` 等，其语句为 `colormap (cool)`。大家知道，任何颜色均可用红绿蓝三色的适当组合来产生。因此，有左端变量的语句 `m=colormap` 将返回一个 64×3 阶的矩阵，其各列依次表示红绿蓝三种基色的分量。`Rgbplot(m)` 则用红绿蓝三种曲线表示三基色分量的大小。表示颜色的另一种方法是色调—饱和度—亮度 (`hsv`) 方式，彩色电视机的调色就用这种方式。MATLAB 也提供了两者之间的转换函数。例如，对默认的彩色图 m 求 `h=rgb2hsv(m)`，则 h 也是一个 64×3 阶的矩阵，只不过它用的是 `hsv` 颜色表

示方式，它的第二列、第三列全为 1，说明其饱和度及亮值均为 1，仅仅色调从 0 到 1 单调地变化，所以它才得到了 HSV 彩色图的名称。这个名称不太好，很容易与 hsv 的颜色表示方式相混淆。

把彩色条离散化形成伪彩色板，它默认地分为 64 份，输入 `pcolor ([1:65;1:65]')` (65 个分割点产生 64 根彩条)，可得出伪彩色板图，如图 2.16 所示。输入 `hot (8)` 把伪彩色板设置为 hot 色并将其改为 8 份，显示这个伪彩色板应当用命令 `pcolor ([1:9;1:9]')`。在自动坐标设定的情况下，彩色条应恰好覆盖图中的最小到最大的坐标值（例如，在代表 *z* 值时为 `[zmin zmax]`）并略有余量。人工设定的命令为 `caxis ([cmin cmax])`，其作用是把彩色条中的最小值、最大值分别与 `cmin`、`cmax` 相对应，以达到用户特定的显示需要。

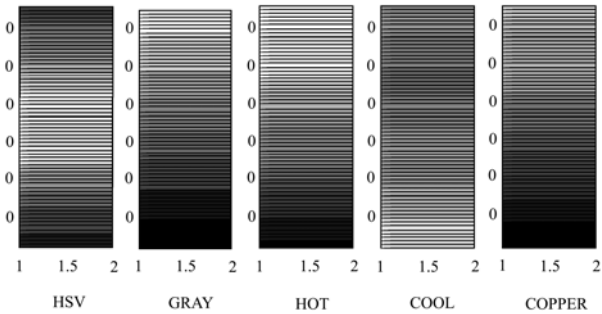


图 2.16 不同彩色设置下的彩色图

在 MATLAB 4.x 中专门设有颜色和照明函数库，在 MATLAB 5.x 中，它被并入三维图形库中，如表 2.16 所示。其中光照模型诸函数的输入变元中都要规定光源的方向或坐标，例如 `surf1 (peaks, [30, 0])` 可得出光源在方位角 30°、俯仰角 0° 方向照射时的 peak 曲面。读者可根据应用的需要自行试验其用法，从 help 文本中可以获得相应的应用信息。

表 2.16 三维绘图和光照函数库（graph3d）(q)

绘制三维 曲线命令	plot3	在三维空间中画线和点	mesh	三维网格图
	fill3	在三维空间中绘制填充多边形	surf	三维曲面图
颜色控制	colormap	彩色查询表	caxis	伪彩色坐标轴定标
	shading	彩色阴影方式	hidden	消隐或显示被遮挡的线条
	brighten	改变彩色图的亮度		
彩色图	hsv	色调—饱和度—亮值彩色图	gray	线性灰度彩色图
	hot	黑—红—黄—白彩色图	cool	蓝绿和洋红阴影彩色图
	bone	蓝色色调的灰度彩色图	copper	铜色调的线性彩色图
	pink	线性粉红色阴影彩色图	prism	光谱彩色图
	jet	HSV 彩色图的变型	flag	红、白、蓝、黑交互的彩色图
	spring	品红和黄阴影彩色图	summer	绿和黄阴影彩色图
	autumn	红和黄阴影彩色图	winter	蓝和绿阴影彩色图
	white	全白彩色图	lines	带颜色线的彩色图
	colorcube	增强的彩色立方体彩色图	colstyle	从字符串分解出颜色和字体

续表

彩色图有关的函数	colorbar	显示彩色条	hsv2rgb	由 hsv 向红绿蓝 (rgb) 转换
	rgb2hsv	红绿蓝向 hsv 转换	contrast	变灰度图为对比增强方式
	rgbplot	用 rgb 绘彩色图	spinmap	旋转彩色图
视点控制	view	规定三维图的视点	viewmtx	视点变换矩阵
	rotate3d	用鼠标拖动图形作三维旋转		
照明模型	surfl	带照明的三维曲面图	specular	镜面反射
	lighting	光照模式	material	材料反射模式
	diffuse	漫反射	surfnorm	曲面法线
轴系控制	见二维图形函数库 (表 2.14), 增加了 xlabel 等			
图形标注	见二维图形函数库			
打印输出	见二维图形函数库			

图像和图形的不同之处在于：图形只有黑白两色，而图像则有深浅之别，称为灰度，彩色图像还有色度。通常黑白图像有 256 个灰度等级，因此，图上的每个点要用 8 位二进制数表示。如果 A 是一个 $M \times N$ 阶的矩阵，其每个元素都是 8 位二进制的整数 (0~255)，则 `image(A)` 就创建了一幅 $M \times N$ 元素组成的图像，其每个元素按其值在彩色图 (colormap) 中取色，输入 `image(A)`，`colormap(gray)` 将得出黑白图像。

2.5.8 低层图形屏幕控制功能

直到现在，所学的都是高层绘图命令。绘图本来是一件很复杂烦琐的工作，MATLAB 怎么把它简化的呢，主要是它代用户做了大量的事务性工作。有些是有根据的，例如，坐标范围根据输入数据的最大最小值来选择；有许多则没有一定的道理，只是开发人员任意给出的默认值。例如，图形的背景色、绘图线条的线宽、线型、颜色、坐标网格的密度及其标注尺寸，等等。在入门的时候，这些内容无须关心，最好由软件开发帮助者帮助设好，免得分散注意力。但一旦遇到默认值不满足实际应用中的特殊需要时，想修改其中某些参数，就会感到高层命令太自动化智能化了，不便于人的干预，低层图形屏幕控制功能就是为补充这个不足的。

MATLAB 中的图形对象共有 11 类，其分类和层次关系如图 2.17 所示。

`root` (根对象) 在最上层，其子类只有一种 `figure` (图形对象)，但可有多，下面有三个子类，即 `uicontrols`，`uimenu`s，`axes`；其中前两类是生成专用人机界面的，这里不涉及，只考虑 `axes` (坐标系对象)，一个图也可有多个坐标系 (如 `subplot`)，在它之下，有六个子类，即 `text` (图中文字)，`line` (线)，`patches` (块)，`surface` (曲面)，`image` (图像) 和 `light` (光照)。任何图形都是由这九类图形对象组成的。`figure` 是 `axes` 的父类，而 `axes` 又是其他六种对象的父类。执行 `figure`，`plot`，`mesh`，`surf`，`fill`，`text` 等高层命令时，MATLAB 将生成某些图形对象，同时，也生成了一个称为“图形句柄”的数组，其中包含了产生各个对象所需的各种默认参数。修改其中的参数就可改变当前对象中的图形特性。

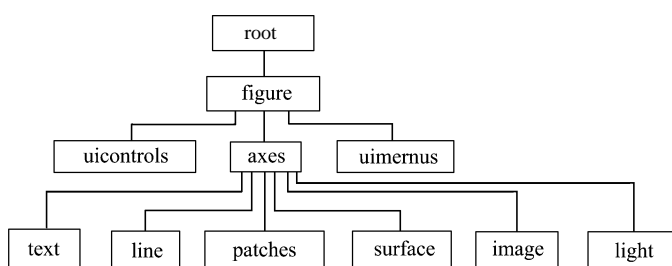


图 2.17 MATLAB 中图形对象的分类和层次关系

1. 取得图形句柄的方法

高层图形命令 `figure`, `plot`, `mesh`, `surf` 等都是无左端变量的。如果在其左端放一个变量，则执行该绘图语句后，该变量就成为此图形的“句柄”名。用 `get` 命令可以得到它的内容。例如：

```

输入  surh=surfc(peaks);
其结果为  surh = 88.0001
           93.0001
           ...
           100.0001
  
```

这表明 `surfc` 命令产生了 9 个对象。因为 `surfc` 除产生三维曲面外还产生了 8 根等高线。其编号分别为 88.0001, 93.0001, 94.0001, 95.0001, 96.0001, 97.0001, 98.0001, 99.0001, 100.0001，这些编号只有内部的意义，对用户来说，主要记住句柄名，把它们命名为 `surh(1)`, ..., `surh(9)` 即可。每一个对象都有自己的句柄。例如：

```

输入
    get(surh(1))
得  CData = [ (49 by 49) ]      注：颜色数据
    EdgeColor = flat            边缘颜色
    EraseMode = normal          可擦模式
    FaceColor = [0 0 0]         前景颜色
    LineStyle = -               线型
    LineWidth = [0.5]           线宽
    MarkerSize = [6]            标志点尺寸
    MeshStyle = both            网格形式
    XData = [ (1 by 49) ]       X 数据
    YData = [ (49 by 1) ]       Y 数据
    ZData = [ (49 by 49) ]      Z 数据
    .....
    Children = [ ]              子类对象的句柄
    Parent = [56.0001]          父类句柄编号
  
```

```
Type = surface
```

对象类型

这里省略了某些无关的句柄内容，在入门课程中，不可能涉及太深。用 `gca` 命令可得到当前 `axes`（即上述对象的父类）句柄，用 `gcf` 命令可得到当前的 `figure`（即上述对象的祖类）句柄，它们的内容更多，此处从略。

2. 修改句柄内容的方法（`set` 命令）

```
set(surh(1), 'linewidth', 2, 'markersize', 3)
```

将三维曲面中的线宽加粗到 2 毫米，标志点直径减小为 3 毫米。但图中的等高线则不受影响。

3. 修改厂设默认参数的方法

前面介绍的方法，只能一张图一张图、一根曲线一根曲线地去修改，如果希望把每张图上的曲线都改粗一些，最好一进 `MATLAB` 环境就把公司出厂设定的默认值改掉。用下述命令可以做到。

```
set(0, 'DefaultLineLineWidth', 1.5)
```

0 代表根对象，把它改了，则它所有的子类和孙类的曲线线宽也都改了。如果只需把当前图中的全部线型改变为点画线，可用

```
set(gcf, 'DefaultLineStyle', '-' )
```

可更改的厂设参数很多，如 `Defaultfigurecolor`, `Defaultsurfaceedgecolor`, `DefaultAxesColor`, `DefaultAxesColorOrder`, `DefaultAxesLineStyleOrder`, `DefaultTextRotation` 等，此处无法穷尽，读者可参阅 `MATLAB` 有关手册。

2.6 M 文件及程序调试

在入门阶段，通常在行命令模式下工作。输入一行命令后，让系统立即执行该命令。用这种方法时，程序可读性很差且难以存储。对于复杂的问题就应编成可存储的程序文本，再让 `MATLAB` 执行该程序文件，这种工作模式称为程序文件模式。

由 `MATLAB` 语句构成的程序文件称为 `M` 文件，它将 `m` 作为文件的扩展名。例如：文件 `expml.m` 用来计算矩阵指数函数的值。因为它是 `ASCII` 码文本文件，所以可以直接阅读并用任何编辑器来建立。

`M` 文件可分为两种：一种是主程序，也称为主程序文件（`script file`），是由用户为解决特定的问题而编制的；另一种是子程序，也称为函数文件（`function file`），它必须由其他 `M` 文件来调用。函数文件往往具有一定的通用性，并且可以进行递归调用（即自己可以调用自己）。`MATLAB` 的基础部分中已有了 700 多个函数文件，它的工具箱中还有千余个函数文件，并在不断扩充积累。`MATLAB` 软件的大部分功能都是来自其建立的函数集，利用这些函数可以使用户方便地解决他们的特定问题。

2.6.1 主程序文件

主程序文件的格式特征如下。

(1) 用 `clear`、`close all` 等语句开始，清除掉工作空间中原有的变量和图形，以避免其他已执行的程序残留数据对本程序的影响。

前几行通常是对此程序用途的说明，特别是在运行时对用户输入数据的要求，更要叙述清楚。不然别人就看不懂也用不成，连自己日后也会遗忘。这些注释行必须以%开始，以便计算机执行时不予理会。**MATLAB** 规定，在输入“**help** 文件名”时，屏幕上会将该文件中以%起头的最前面几行的内容显示出来，使用户知道如何使用。这些注释是可以用汉字的。%也可以放在程序行的后面做注释，**MATLAB** 将不执行该字符后的任何内容。

(2) 程序的主体。如果文件中有全局变量，即在子程序中与主程序共用的变量，应在程序的起始部分注明。其语句是

```
global 变量名1 变量名2 ……
```

为了改善可读性，要注意流程控制语句的缩进及与 `end` 的对应关系。另外，程序中必须都用半角英文字母和符号（只有引号括住的和%后的内容可用汉字）。特别要注意英文和汉字的有些标点符号（如句号、冒号、逗号、分号、引号乃至%、=、（）等），看起来很相似，其实代码不同。用错了，不但程序执行不通，而且几乎必定死机。因此输入程序时，最好从头到尾用英文，不要插入汉字。汉字可在程序调试完毕后加入。用 **MATLAB 5.x** 的编辑器比较好，因为它对出现的非法字符会显示鲜明的红色，引起用户的注意。且在它的菜单 **View** 中，选 **Auto Indent Selection** 项可以自动对程序进行缩进排版。

(3) 整个程序应按 **MATLAB** 标识符的要求起文件名，并加上后缀 `m`。用 **MATLAB 4.x** 时，文件名长度不要超过 8 个字符。文件名中不允许用汉字，因为这个文件名，也就是 **MATLAB** 的调用命令，它是不认汉字的。将文件存入自己确定的子目录中，该子目录应置于 **MATLAB** 的搜索路径下。所以还应避免出现汉字路径名。

这样就完成了程序的编制。在 **MATLAB** 的命令窗中输入此程序的文件名后，系统就开始执行文件中的程序，主程序文件中的语句将对工作空间中的所有数据进行运算操作。

【例 2.6】 要求列写一个求 **Fibonacci** 数的程序。它是一个数列，从[1, 1]开始，由数列的最后两个元素之和生成新的元素，依次类推。其程序如下：

```
% 计算 Fibonacci 数的 M 文件
clear, close all
N = input('输入最大数值范围 N=')
f = [1, 1]; i = 1; % 变量的初始化
while f(i) + f(i+1) < N % 循环条件检验
    f(i+2) = f(i+1) + f(i); i=i+1; % 求 Fibonacci 数的算式
end,
f, plot(f) % 显示和绘图
```

将此程序以文件名 `fibon.m` 存入某一 MATLAB 搜索目录下。在 MATLAB 命令窗中输入 `fibon`，系统就开始执行这个程序。它首先会要求用户输入 `N`，然后计算数值小于 `N` 的 Fibonacci 数，并绘出图线。设输入 `N=100`，得出（图线略）：

`f = 1 1 2 3 5 8 13 21 34 55 89`

该文件执行结束，变量 `f` 和 `i` 仍保存在工作空间中。

【例 2.7】 列出求素数的程序。所谓素数就是只能被它自身和 1 除尽的数。程序如下：

```
% 求素数(prime number)的程序
clear, close all
N =input('N= '), x=2:N;           % 列出从 2 到 N 的全部自然数
for u=2:sqrt(N)                   % 依次列出除数（最大到 N 的平方根）
    n=find(rem(x, u) ==0 & x~=u); % 找到能被 u 除尽而 u 不等于 x 的数序号
    x(n) = [];                    % 剔除该数
end, x                            % 循环结束，显示结果
以 prime.m 为名存入系统，就可以执行了。给出 N=40，结果为
x =  2      3      5      7      11      13      17      19      23      29      31      37
```

2.6.2 人机交互命令

在执行主程序文件中，往往还希望在适当的地方对程序的运行进行观察或干预，这时就需要人机交互的命令，调试程序时，人机交互命令更不可少。这些命令可以从 MATLAB 语言结构库（如表 2.17 所示）中调用。下面介绍几条：

表 2.17 语言结构库（lang）（k）

	名 称	功 能	名 称	功 能
估值并执行	<code>eval</code>	执行 MATLAB 语句字符串	<code>feval</code>	执行由字符串命名的函数
	<code>evalin</code>	估值工作空间中的表示式	<code>builtin</code>	从超载方法执行内置函数
	<code>assignin</code>	分配工作空间中的变量	<code>run</code>	运行程序文件
流程控制语句	<code>if</code>	条件执行命令	<code>else</code>	与 <code>if</code> 联用
	<code>elseif</code>	与 <code>if</code> 联用	<code>end</code>	For, while, if 语句的终点
	<code>for</code>	确定次数的重复语句	<code>while</code>	非确定次数的重复语句
	<code>break</code>	终止执行循环	<code>return</code>	返回到调用函数
	<code>switch</code>	在表示式的几种情况中选择	<code>otherwise</code>	switch 语句中的默认值
	<code>case</code>	switch 语句中的情况		
程序、函数和变量	<code>script</code>	MATLAB 程序文件——M 文件	<code>function</code>	加入新函数
	<code>global</code>	定义全局变量	<code>mfilename</code>	当前执行的文件名
	<code>list</code>	以逗号分割的清单	<code>isglobal</code>	是全局变量时为真
	<code>exit</code>	检查变量或函数是否存在		

续表

	名 称	功 能	名 称	功 能
变元管理	nargchk	检验输入变元的数目	nargin	输入变元的数目
	nargout	输出变元的数目	varargin	长度可变的输入变元清单
	varargout	长度可变的输出变元清单	inputname	输入变元的名称
信息显示	error	跳出函数并显示信息	laster	最近的出错信息
	warning	显示警告信息	errortrap	在测试中跳过错误
	disp	显示数组	fprintf	显示格式化信息
	sprintf	把格式化数据写成字符串	echo	显示执行的 MATLAB 语句
人机交互命令	input	提示用户输入	keyboard	调用等待键盘输入
	menu	生成用户输入的选择菜单	pause	暂停，等待用户响应

- `echo on (off)` 一般情况下，M 文件中的命令不会显示在屏幕上。而在命令 `echo on` 之后，会在执行每行程序前先显示其内容。
- `pause (n)` 程序执行到此处，暂停 `n` 秒，再继续执行。如果没有括号参数，则等待用户输入任意键后才继续执行。
- `keyboard` 程序执行到此处暂停，在屏幕上显示字符 `K`，并把程序的输入和执行权交给用户（键盘）。用户可以像在普通 MATLAB 命令窗中那样进行任何操作（例如，检查中间结果等）。如果需要系统恢复运行原来的程序，只需输入字符串 `return`。在 M 文件中设置该命令，有利于进行程序调试以及临时修改变量内容。
- `input ('提示符')` 程序执行到此处暂停，在屏幕上显示引号中的字符串。要求用户输入数据。如程序为 `X=input ('X=')`，则屏幕上显示 `X=`，输入的数据将赋值给 `X`。数据输入后，程序继续运行。`input` 命令也可以接受字符，其格式为 `Y=input ('提示符', 's')`，此时 `Y` 将等于输入的字符串。
- `^C` 强行停止程序运行的命令。`^C` 念做（Control-C），即先按下 `Ctrl` 键，不抬起再按 `C` 键。在发现程序运行有错，运行时间太长时，可用此方法中途终止它。
- `menu` 是用来产生人机交互设备选择菜单的命令，参阅 `help` 文件。

2.6.3 函数文件

函数文件是用来定义子程序的。它与程序文件的主要区别有三点：

- 由 `function` 起头，后跟的函数名必须与文件名相同；
- 有输入输出变元（变量），可进行变量传递；
- 除非用 `global` 声明，程序中的变量均为局部变量，不保存在工作空间中。

先看一个简单的函数文件，其文件名 `mean.m`。

输入

`type mean`

得到

```
function y = mean(x)
% MEAN 求平均值。对于向量，mean(x)返回该向量 x 中各元素的平均值
% 对于矩阵，mean(x)是一个包含各列元素平均值的行向量
[m, n] = size(x);
    if m==1 M=n;      end    % 处理单行向量
y=sum(x)/m
```

文件的第一条语句定义了函数名、输入变元以及输出变元。没有这条语句，该文件就成为程序文件，而不再是函数文件。输入变元和输出变元都可以有若干个，但必须在第一条语句中明确地列出。

程序中的前几条带%的字符行为文件提供注解，输入 `help mean` 命令后，系统将显示这几条文字，作为对文件 `mean.m` 的说明，这和主程序文件相同。

变量 m , n 和 y 都是函数 `mean` 的局部变量，当 `mean.m` 文件执行完毕，这些变量值会自动消失，不保存在工作空间中。如果在该文件执行前，工作空间中已经有同名的变量，系统会把两者看做各自无关的变量，不会混淆。这样，调用子程序时就不必考虑其中的变量与程序变量冲突的问题了。如果希望把两者看成同一变量，则必须在主程序和子程序中都加入 `global` 语句，对此共同变量进行声明。

给输入变元 x 赋值时，应把 x 替换成主程序中的已知变量，假如它是一个已知向量或矩阵 Z ，可写成 `mean (Z)`，该变量 Z 通过变元替换传递给 `mean` 函数后，在子程序内，它就变成了局部变量 x 。

下面的例子是多输入变量函数 `logspace`，用于生成等比分割的数组。

```
function y = logspace(d1, d2, n)
% LOGSPACE 对数均分数组
% LOGSPACE(d1, d2)在 10^d1 与 10^d2 之间生成长度为 50 的对数均分数组
% 如果 d2 为 pi, 则这些点在 10^d1 和 pi 之间
% LOGSPACE(d1, d2, n)生成的数组长度为 n, n 的默认值为 50
if nargin == 2    n = 50; end          % 输入变元分析及 n 的默认值设置
if d2 == pi    d2 = log10(pi);end      % d2 为 pi 时的设置
y = (10).^[d1+ (0:n-2)*(d2-d1)/(n-1), d2]; % 将结果返回到输出变元
```

在本例中使用了特定变量 `nargin` 表示输入变元的数目。当只有两个输入变元时，它默认地设定 $n=50$, `nargout` 表示输出变元数目的变量。MATLAB 常常根据 `nargin` 和 `nargout` 的数目不同而调用不同的程序段，从而体现它的智能作用。

再来看 MATLAB 如何定义一个任意非线性函数的。在对微分方程进行数值积分或解任意非线性方程求解时，都需要先列出一个这样的函数文件。一个典型程序（文件名为 `humps.m`）如下：

```
function y = humps(x)
%HUMPS 由 QUADDEMO, ZERODEMO 和 FPLOTDemo 等程序调用的一个函数
```



```
% HUMPS(X) 是一个在 x = 0.3 and x = 0.9 附近有尖锐极大值的函数
% 参看 QUADDEMO, ZERODEMO 和 FPLOTDemo
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

程序中的运算都采用元素群算法，以保证此函数可按元素群调用。MATLAB 中几乎所有的函数都能用元素群运算，所以，本书中自编的子程序，也尽量满足元素群算法的要求。

2.6.4 文件编辑器及程序调试

MATLAB 提供的编辑器较为理想，它把编辑与调试结合在一起了。实际上，MATLAB 的主程序是比较好调试的，因为 MATLAB 的查错能力很强，配上工作空间中变量的保存和显示功能，不需要用专门的调试命令，调试也可以很方便地进行。

需要用调试命令的主要是函数程序。因为在函数程序出错而停机时，其变量不被保存。虽然它也会指出出错的语句，但因为子程序中的变量在程序执行完毕后会消失，其他现场数据都无记录。会给调试带来很大困难。解决这个问题可采用以下的措施。

- 把某些分号改为逗号，使中间结果能显示在屏幕上，作为查错的依据。
- 在子程序中适当部位加 `keyboard` 命令。此处，系统会暂停而等待用户输入命令。这时子程序中的变量还存在于工作空间中，可以对它们进行检查。
- 将函数文件的第一行前加 `%`，使它成为程序文件，进行初步调试。第一行中的输入变元，可改用 `input` 或赋值语句来输入，调好后再改回函数文件。
- 使用 MATLAB 提供的调试命令。其命令共有 11 条，如表 3.1 所示。根据经验，当程序不太长（例如 20 行以下）时，用调试命令反而麻烦。所以，在入门课程中不做介绍。

MATLAB 的开发环境和工具

作为一种优秀的计算软件，MATLAB 之所以能够成功，不仅因为其语法和编程的简洁高效，而且在于它有一个便于使用开发并与其他软件（甚至硬件）进行交互通信的环境。比如，它可以在多种计算机机型和操作系统下运行，其使用的界面和程序又完全相同，使程序设计与平台无关；它能够和一些重要的文字编辑器及图形编辑器进行交互，把计算结果很方便地组织成为图文并茂的文章，等等。从 MATLAB 3.x 开始，其语言已经相当成熟，版本的每一次升级，在语言方面的变化很少，变动主要集中在工具箱的扩充和开发环境的改善，这方面的内容涉及面很广，不可能在一本教科书中叙述清楚。本章将以 MATLAB 6.x 版本为主要对象，对此做一扼要的介绍。

3.1 MATLAB 与其他软件的接口关系

3.1.1 与磁盘操作系统的接口关系

1. 变量的存储和下载

save 命令把工作空间中的全部变量值存入磁盘。其默认的文件名是 matlab.mat。第二次再用 save 命令时，如果仍用默认文件名，则原来文件中的数据就被冲销。如果只要把 *a*，*b*，*c* 三个变量保存在名为 aa.mat 的文件中，则可输入：

```
save aa a b c
```

mat 格式是人读不懂的，如果要存成 ASCII 码格式，则应再加上一个格式说明符：

```
save aa a b c - ascii
```

load 是 save 的逆过程, 它把磁盘上存储的 mat 数据文件取回到 MATLAB 工作空间中。其默认的文件名也是 matlab.mat。在不用默认文件或默认格式时, 其命令格式与 save 命令相仿, 唯一的差别是它不能选择变量。例如 load aa, 它把 aa.mat 文件中的全部数据连同其变量名都下载到工作空间中。

表 3.1 列出了 MATLAB 的通用命令。

2. 工作日志的记录

diary 命令可把 MATLAB 工作过程中的全部屏幕文字和数据以文本方式记录下来, 成为一个工作记录, 默认的文件名为 diary。因为它是文本文件, 并可由任何文字处理器来修改编辑, 所以, 有很大的使用价值, 其用法如下。

当准备作记录时, 在命令窗中输入 diary on 或 diary bbb, 后者用 bbb.txt 为文件名。从此时开始, 所有在 MATLAB 命令窗中出现的文字 (包括输入的命令和数据) 都将记录在 diary.txt 或 bbb.txt 文件中。当需记录的过程结束时, 应输入 diary off。此后的屏幕内容不作记录。如果再次使用 diary on 或 diary 文件名, 则新记录的内容将接在原记录的后面, 不会冲销原记录。diary 文件可以用 notepad 或 Winword 打开阅读。

表 3.1 通用命令库 (general) (f)

函数的管理命令	what	列出 M、MAT 和 MEX 文件	which	找函数和文件所在的子目录
	type	显示 M 文件的全部内容	pcode	建立微码文件 (P 文件)
	edit	编辑 M 文件	inmem	列出内存中的函数
	lookfor	在求助文字中搜索关键词	mex	编译 MEX 函数
通用信息	help	在线帮助文件	whatsnew	未列入说明书的新功能信息
	helpwin	有独立视窗的在线帮助文件	readme	显示 readme 文件
	helpdesk	超文本帮助文件	ver	MATLAB 和各工具箱的版本
	demo	运行演示程序		
工作空间管理	who	列出工作空间变量	save	从工作空间存储变量到磁盘
	whos	列出工作空间变量详情	clear	从内存中清除变量和函数
	load	从磁盘取出变量到工作空间	pack	紧缩工作空间内存
管理搜索路径	path	查找和改变 MATLAB 搜索路径	rmpath	在搜索路径上去除子目录
	addpath	在搜索路径上增加子目录	editpath	修改搜索路径
文件操作系统	cd	更改当前工作目录区	pwd	显示当前工作目录
	dir	列出子目录	web	打开 Web 浏览器
	delete	删除文件	computer	当前的计算机型号
	getenv	获取环境参数	Ctrl C	中断 MATLAB 的运行
命令窗控制	profile	设置 M 文件执行时间	format	设置输出格式
	clc	清除命令窗	diary	保存 MATLAB 运行文字记录
	home	使光标复原到左上角	more	在命令窗中控制分页输出
启动退出	quit	退出 MATLAB	matlabrc	启动的主 M 文件
	startup	启动 MATLAB 时的 M 文件		

续表

公共信息	info	关于 Mathworks 公司的信息	hostid	MATLAB 服务主顾的识别码
	subscribe	订购 MATLAB 须知		
调试命令	dbstop	设置断点	dbclear	清除断点
	dbcont	继续执行	dbdown	改变局部工作空间内容
	dbstack	列出谁调用谁的清单	dbstatus	列出所有断点的清单
	dbstep	执行一行或几行	dbtype	列出带行号的 M 文件
	dbup	改变局部工作空间内容	dbquit	退出调试模式
	dbmex	调试 MEX 文件	dbstop	停止调试

为了避免在日志文件中记录不必要的调试过程和“垃圾内容”，应该在程序调试成功、运行无误后再打开日志文件，让程序正式运行一次。有时还需先输入 `echo on`，使得被执行的语句也在屏幕上显示并被记录到日志中去。记录中如发现有不必要的内容，可用文字处理器予以删改。diary 文件不能记录 MATLAB 运行中生成的图形。

3. 日期和时间命令

MATLAB 中的某些命令与操作系统是有内在联系的。除了前面说过的它可直接应用的操作系统命令 `dir`, `delete`, `cd` 等之外，有关时间和日期方面的命令，都是从操作系统中提取数据的。这些命令如表 3.2 所示。

表 3.2 时间和日期函数库 (timefun) (w)

当前日期	now	当前日期和时间的时点数	clock	当前日期的日期向量
	date	当前日期的字符串		
基本函数	datenum	成序列的日期数	datevec	日期向量
	datestr	日期的字符串格式		
日期函数	calender	日历	eomday	月末日的星期数
	weekday	星期数	datetick	日期的格式设定
定时函数	cputime	以秒计的 CPU 时间	etime	经历时间
	tic,toc	秒表定时器的启动和停止	pause	暂停等待时间

下面介绍如何确定做某种计算所需的时间。例如，想看看做 1 个 100×100 阶矩阵的求逆运算所需的时间，可以用下列三组语句之一。

- `t0=clock;y=inv(rand(100,100));etime(clock,t0)`
- `t=cputime;y=inv(rand(100,100));cputime-t`
- `tic;y=inv(rand(100,100));toc`

这三种方法的差别在于：第一种方法要先后两次提取年月日时分秒数据并将它们相减；第二种方法以开机时间为基准；第三种方法则用 `tic` 把秒表置零，求得的 `toc` 就是经历的时间。

4. 不退出 MATLAB 环境运行其他软件

以“!”开始的命令表示这是一个 DOS 操作系统的命令。可以用这个方法在不退出 MATLAB 环境的条件下，运行以 DOS 操作系统为基础的其他软件。

3.1.2 与文字处理系统 WinWord 的关系

1. 利用剪贴板进行交互

MATLAB 的程序要利用文字处理系统来编辑修改，它的运行结果（包括数据和图形）需要由图文处理系统来整理加工。因此，它与 Word 图文处理系统有非常紧密的关系。它的命令窗中的所有文字数据及图形窗中的所有图形都可用 Windows 的剪贴板（Clip Board）送到 Word 中去，并可以用 Word 对它们进行编辑，形成图文并茂的书面报告。

在图形窗中截取图形时，应先用鼠标拖动边缘的方法将图形窗调到需要的大小，然后，用鼠标单击菜单中的【Edit】项，在【Copy Options】子项中有【Metafile】（矢量模式）和【Bitmap】（点阵模式）。通常应选【Metafile】，因为这种模式便于在 Word 中做进一步的缩放修改。在设定完毕后，再选定【Copy Figure】或【Copy】，图就放到剪贴板上去了。然后，可把这个图贴向 Word 的任何文本文件并在其中做进一步的编辑修改。在 MATLAB 中缩放可以保持图中标注文字的可读性，而在 Word 中缩放图形往往会使文字排列不好。所以，建议在 MATLAB 中先把图形比例取得大体合适，避免到 Word 中做大幅度的缩放调整。

2. 文字编辑器的使用

在 MATLAB 6.x 中，已经把 Word 中的文字编辑功能集成为 MATLAB 的程序编辑和调试器。在图 1.2 显示的命令窗中，按下最左边的按钮，就会激活其程序编辑和调试器，生成图中的视窗。该视窗中的各个按钮的形式和功能与 Word 界面的几乎完全相同，所以不必细说。它的特殊之处在于：

（1）它会用不同颜色显示 MATLAB 规定的保留字（蓝）、非法字符（鲜红）、注释字符（绿）和引用字符（深红）等；

（2）存储文件名的后缀为.m——即生成的是 M 文件；

（3）当被编辑的文件以 function 开头，即被编辑的是一个函数文件时，MATLAB 编辑器会自动将存储文件名定为该程序中的函数名（见 2.6 节中函数文件的命名规定）；

（4）能对程序自动缩进排版，便于阅读和调试。选定需要排版的程序段，单击菜单项【View】下的子项【Auto Indent Selection】，即可完成；

（5）它有程序调试器功能，反映在菜单项【Debug】的各子项中。

3. Notebook 软件工具

Notebook 是 Mathworks 公司开发的软件，它在 Word 和 MATLAB 两个软件系统之间搭起了一座双向接口的桥梁。当这个软件工作时，可在 Word 中输入含有部分 MATLAB 语句的文本文件。以后只要选中这些语句，再按下 Ctrl+Enter 键，该软件就会把这些语句送给 MATLAB 去执行，然后把运行的结果又送回 Word，并用不同的颜色显示输出和输入的不同。利用这个工具，教师可以边写教案，边检验教案中的程序语句。科技工作者也可一边写论文，一边让论文中的程序运行结果直接出现在论文中，就不再需要来回剪贴了。不过要运行这个工具，必须在安装 MATLAB 时，把 Notebook 软件工具装入系统。

3.1.3 图形文件的转储

可以把 MATLAB 的图形文件转储为多种标准图形格式，以便使用各种图形软件进行处理。存储时所用后缀可以是各种标准图形格式的后缀，如 gif, bmp, jpg 等。它们可由图形窗对图形进行转储而得到，并可用菜单操作来实现图形转储。只要单击图形窗的菜单项【File】的子菜单【Export】（导出），就会出现如图 3.1 所示的界面。在【Save as Type】中选定存储格式，给出文件名，再单击【Save】，即可完成图形的存储。这里用【Export】表示 MATLAB 把图形转储为其他软件的格式，是软件之间的接口转换。这样生成的文件，不属于 MATLAB 文件的范畴。

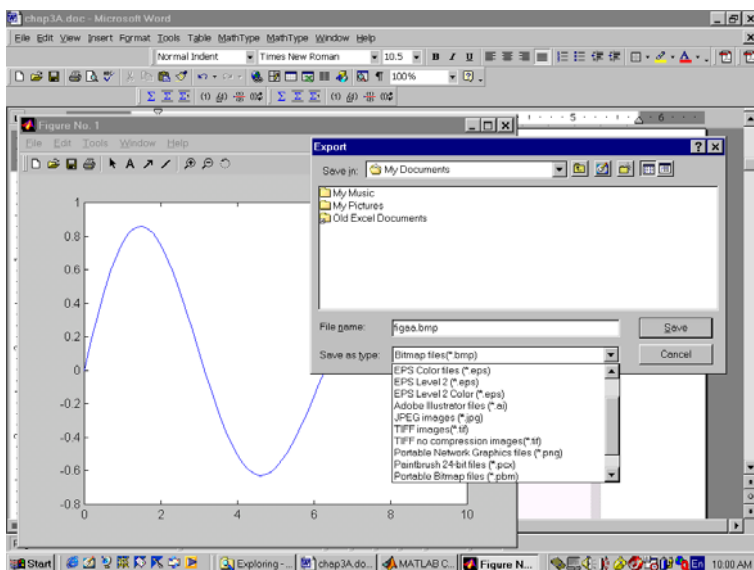


图 3.1 MATLAB 6.x 的图形窗及其转储（导出）界面

图形窗上还有一排图标组成的工具栏，可以完成图形编辑、缩放、旋转和加字符标注等功能，将鼠标移至该图标上时，会出现其功能的说明，此处无须一一介绍，读者可以在试用中学习。

3.1.4 低层输入输出函数库

MATLAB 可以用 load 和 save 命令来保存和提取数据，其数据可以是 mat 或 ASCII 码格式，这已在前面讲过。但这只适合于 MATLAB 环境自身。作为一种科学计算软件，与其他软件系统进行直接的（没有人参与的）数据交换是十分重要的，它可以避免人为差错和运行低效。通过输入输出文件进行数据交换是有效的方法之一。因为几乎任何算法语言都有有限的几种输入输出文件格式（比如二进制格式和 ASCII 码字符格式），MATLAB 可以用这几种格式进行读写，也就保证了它可以在这一级上与其他语言相连接。例如，将其他软件产生的或仪器测量的数据自动读入 MATLAB，再进行分析处理并绘成图形输出等。

读不同格式的文件要用不同的命令，这个库中的命令见表 3.3。

如果要在一个二进制文件 `aaa.bin` 中写入工作空间中的变量 `x`，则其程序为如下两条语句：

```
fid1=fopen('aaa.bin','r+');%打开 aaa.bin, 'r+'表示可读可写。fid1 为文件标识
N=fwrite(fid1,x,'float')    %将 x 以 float（浮点）格式写入 fid1 文件,返回实际写入
```

的元素数 `N`

从数据文件读出变量是一个逆过程，例如，要从 `aaa.bin` 读入二进制数据并将它赋值给 `A`，程序可编写如下：

```
frewind(fid1)
fid1=fopen('aaa.bin','r+'); %打开 aaa.bin, 'r+'表示可读可写。fid1 为文件标识
A=fread(fid1,[5,5],'float')
```

表 3.3 低层文件输入输出库命令（iofun）（j）

文件开闭及 I/O	fopen	打开文件	fscanf	从文件读入格式化数据
	fclose	关闭文件	fprintf	把格式化数据写入文件
	fread	从文件读入二进制数据	fgetl	从文件读入一行，去掉新行字符
	fwrite	把二进制数据写入文件	fgets	从文件读入一行，保留新行字符
文件定位	ferror	询问文件 I/O 的出错状态	ftell	提取文件位置指针
	feof	测试文件结尾	frewind	倒回文件
	fseek	设置文件位置指针		
字符串及文件名处理	sprintf	把格式化数据写入字符串	sscanf	从字符串中读取格式化数据
	MATLABroot	MATLAB 安装的根目录	partialpath	部分路径名
	filesep	本平台的目录分隔符	mexext	本平台的 MEX 文件名后缀
	pathsep	本平台的路径分隔符	fullfile	从各部分构成全文件名
	tempdir	获取当前目录	tempname	获取当前文件
文件输入输出	load	从 MAT 文件下载到工作空间	save	把工作空间变量存入 MAT 文件
	dlmread	从 ASCII 码分隔文件读取矩阵	dlmwrite	把矩阵写入 ASCII 码分隔数据文件
	wk1read	读 WK1 文件	wk1write	在 WK1 格式的文件中写入矩阵
图像声音 I/O	imread	从图形文件读出图像	iminfo	返回图像文件的信息
	imwrite	把图像存入图形文件		
	wavwrite	写入 WAVE（“wav”）声音文件	wavread	读出 WAVE（“wav”）声音文件

注意到这个程序比写入时多了第一行，因为文件的读写犹如磁带，写入以后必须倒带才能重放，要先输入倒带命令 `frewind(fid1)`，而第 3 句表示将 `fid1` 文件中的前 25 个数据以 `float`（浮点）格式读出，列成 5×5 阶矩阵，赋予变量 `A`。如果以后还有从 `fid1` 文件读出的语句，就将从第 26 个数据开始。

输入输出的格式必须相同。**MATLAB** 内部本来只有一种双精度格式，现在要变换为其他语言中的多种数据类型，所以会很不适应。读者应在学了 C 语言或其他语言之后再理

解本节。库中每个命令的具体用法可参看 help 文本，此处不多占用篇幅。

在进行音频信号或图像处理时，需要与声音文件及图像文件接口。MATLAB 也提供了相应的命令。可参看表 3.3。

在 MATLAB 中还有动态数据交换的函数库 (dde)。利用它可以不经过“文件”这个中间环节而直接把运行 MATLAB 的计算机和运行其他软件的计算机之间通过网络进行数据交换。使 MATLAB 与其他软件平台之间的双向调用成为可能，这个函数库中的内容如表 3.4 所示。

表 3.4 客户机函数库 (dde) (g)

动态数据交换	ddeadv	建立链接	ddereq	从应用取得数据
	ddeexec	送出执行字符串	ddeterm	结束 DDE 对话
	ddeinit	DDE 对话初始化	ddeunadv	卸除链接
	ddepoke	把数据送到应用		

3.1.5 与 C 和 FORTRAN 子程序的动态链接

MATLAB 本身是用 C 语言编写的，它丰富的科学计算子程序库中的许多经典部分来自久经考验的 FORTRAN 程序库。它可以直接调用经过一定处理后的 C 和 FORTRAN 可执行文件，因而使执行这些子程序的速度与 C 语言及 FORTRAN 语言相同。这些可执行文件就是后缀为 mex 的文件。除了 MATLAB 中已有的 mex 文件外，用户也可把自己找到的其他可执行文件加入系统。

MATLAB 高级工具箱中还有 C 编译器，可把 MATLAB 语言编写的子程序编译成 C 语言程序，以期提高它的运行速度。MATLAB 6.x 是用 Java 语言扩展的，因此，也为它今后充分利用 Java 的功能创造了有利条件。

3.2 MATLAB 的文件管理系统

3.2.1 安装后的 MATLAB 文件管理系统

如果用光盘来安装 MATLAB 软件，不管版本有何差别，其过程和其他软件相仿，此处从简。安装后的 MATLAB 根目录（通常表为 MATLABroot）下，至少有 bin, extern, help 和 toolbox 这 4 个子目录，其中子目录 bin 包含了 MATLAB 所要用到的二进制文件。启动 MATLAB 的执行文件 matlab.exe 就在这个目录中，双击这个文件就可以启动 MATLAB 软件。子目录 extern 包含了 MATLAB 所要用到的外部文件；子目录 help 包含了 MATLAB 的各种帮助文件，如果有下一级子目录 pdf_doc，则其中将包括 MATLAB 及其工具箱的说明书，那是十分有用的资料。子目录 toolbox 包含了 MATLAB 的各种函数库及已装入的作为下一级子目录的工具箱名称等，它至少应有 local 和 matlab 两项，其中 matlab（注意用的是小写）又有 22 个子目录，分别是本书第 1 章～第 4 章介绍的 MATLAB 中的基本函数集，如 datafun,

elfun 等。通常在 MATLAB 根目录下,还应该建立一个用户自己的子目录,例如 user 或 work,以便把用户自编的应用程序存在这个子目录下,免得与系统中原有的文件系统混淆。

3.2.2 MATLAB 自身的用户文件格式

MATLAB 的用户文件通常包括以下几类。

(1) 程序文件 包括主程序和函数文件,其后缀为.m,即 M 文件。通常它由文本编辑器生成。MATLAB 的各个工具箱中的函数,大部分也是 M 文件。

(2) 数据文件 其后缀为.mat。在 MATLAB 命令窗中,用 save 命令存储的变量,在默认条件下就生成这类文件。

(3) MATLAB 的可执行文件 其后缀为.mex,它们由 MATLAB 的编译器对 M 文件进行编译后生成,其运行速度远高于直接执行 M 文件的速度。

此外,用 Simulink 工具箱建模,会生成模型文件(后缀为.mdl)和仿真文件(后缀为.s),这些是 MATLAB 自身的文件格式。

3.2.3 文件管理和搜索路径

MATLAB 管理的文件范围由它的搜索路径来确定。整个计算机资源管理器文件系统中的任何一个底层文件夹,都可以列入 MATLAB 的搜索路径,在这些文件夹中的文件都可以被执行。反之,如果用户编写的程序未存入 MATLAB 搜索路径的子目录中,则 MATLAB 将找不到它,因而也无法运行这个程序。

要显示或修改搜索路径,可以用 path 命令。

- path 列出 MATLAB 的搜索路径。
- path(path, '新增路径名') 在原搜索路径群中加入一个新路径。

例如,在 C 盘中已有一个子目录 user1,要把它放入 MATLAB 的搜索路径上,可输入: path(path, 'c:\user1')。注意,这个子目录必须先建立好,使用 path 命令才有效,否则, MATLAB 找不到这个子目录,它会显示“无此子目录,命令无效”,并拒绝执行。

用命令行设置路径要输入文件夹的全路径名,很容易出错,用菜单工具较好。MATLAB 6.x 所设置的修改路径的菜单工具如下。

在命令窗中单击【File】→【Set Path】,就会出现如图 3.2 所示的【Set Path】对话框。该对话框左侧是一排按钮。包括【Add Folder】、【Add with Subfolders】、【Move to Top】、【Move Up】、【Remove】、【Move Down】和【Move to Bottom】等。如果要将某文件夹(连它的子文件夹)都列入 MATLAB 搜索路径上去,可单击【Add with Subfolders】,此时将弹出一个系统文件搜索框,即图 3.2 上右下角的小框。在其中找到该文件夹,选中它,再按【确定】按钮,小框即关闭。然后,再在【Set Path】对话框中下面一横排按钮中,按【Save】和【Close】按钮即可。



MATLAB 与目录和搜索有关的命令如下。

- 说明** 这三个命令都是 DOS 操作系统的命令, 在 MATLAB 中同样有效。

- which path

```
c:\matlab\toolbox\matlab\general\path.m
```

- **lookfor** [字符串] 在全部 **help** 文件中搜索包含该字符串的内容。例如，想找到所有与等高线绘制有关的命令，可输入

```
lookfor contour
```

得 CLABEL Add contour labels to a contour plot.
CONTOUR Contour plot.
CONTOUR3 3-D contour plot.
CONTOURC Contour computation.
MESH C Combination MESH/CONTOUR plot.
SURFC Combination SURF/CONTOUR plot.

3.2.5 搜索顺序

在 MATLAB 执行程序时, 当遇到一个字符串时, 如何判别该字符串的意义呢? 它按如下的顺序 (优先级) 与已有的记录相比较: 工作空间的变量名 → 内部固有变量名 → mex 文件名 → M 文件名。如果两个名字相同, 它只认优先级高的名字。比如, 用户在工作空间中给 *i* 赋了值, 那么, 系统就不会取内部固有变量中设定的虚数 *i*; 如果用户在程序中设立了一个与 MATLAB 函数同名的变量, 则每次调用此名字时, 出现的将是用户自定的变量, 调不出 MATLAB 中的函数。所以, 在自设变量名时要防止与 MATLAB 中的函数重名。

MATLAB 中也有函数同名只是后缀不同的情况。因为 mex 后缀是二进制的执行文件, 它的运行速度比 M 文件快得多, 所以会优先执行它。mex 文件通常是对 M 文件做编译后生成的。无法阅读也不好修改。

3.3 MATLAB 6.x 的开发环境

3.3.1 桌面系统的内容

第 1 章中初步介绍了 MATLAB 的几个基本视窗。随着系统的升级, 它们在不断升级, 而且为了开发者的方便, 不断增加着新视窗。到 MATLAB 6.x 则发展到一个新阶段, 它把多种开发工具集成为 MATLAB 桌面系统。该系统由桌面平台以及组件组成。包含如下八个组成部分: 命令窗口 (Command Window)、历史命令窗口 (Command History)、资源目录本 (Launch Pad)、当前路径浏览器 (Current Directory Brower)、帮助浏览器 (Help Browser)、工作空间浏览器 (Workspace Browser)、数组编辑器 (Array Editor) 以及程序编辑调试器 (Editor-Debugger)。它们的功能简述如下。

(1) 命令窗口

第 2 章中的全部工作都是在命令窗中完成的, 所以, 不必更多解释。

(2) 历史命令窗口

历史命令窗口用于记录并显示以前各次工作进程中曾输入的全部行命令。利用它可以方便地修改和输入较长的行命令, 或把多个有用的行命令挑选出来, 组成一个完整的程序文件。因此, 这是一个很有用的工具。

(3) 资源目录本

资源目录本用于把用户在当前系统中安装的所有 MATLAB 产品说明、演示以及帮助

信息的目录集成起来，便于用户迅速调用查阅。

(4) 当前路径浏览器

当前路径浏览器用于随时显示系统当前目录下的 MATLAB 文件信息，包括文件名、文件类型、最后修改时间以及该文件的说明信息等。

(5) 帮助浏览器

所有的帮助信息都可以在该浏览器中显示。而且用户可以对原有的帮助信息编辑取舍，或加入自己的注解，形成自己的帮助文件。

(6) 工作空间浏览器

工作空间浏览器用于显示所有目前保存在内存中的 MATLAB 变量的名称、数学结构、字节数以及类型，并与按下工作空间查看按钮或输入 whos 命令所得的结果相同。只是在工作空间浏览器中，还可以对变量进行编辑或图形操作。

(7) 数组编辑器

用户可以直接在数组编辑器中修改所打开的数据，甚至可以更改该数据的数学结构以及显示方式。

(8) 程序编辑调试器

以上八个组成部分中，许多是 MATLAB 5.x 就有的。但其中有些部分的功能较低，例如，只能显示，不能编辑修改。MATLAB 6.x 把它们集成起来，构成桌面系统。而且各个组成部分都独立地构成视窗，具有自己的菜单和工具条，可以对视窗中的内容进行编辑和存储，这就使它们的功能更强大，使用更方便。对初学者而言，太多的视窗只会造成混乱，因此在本书第 1 章中，我们只介绍最基本的几个视窗，现在才做较详细的讨论。即便如此，如果自己不在应用中去实践，学了也很难记住，所以本书只能作简略介绍，读者仍需自己看说明书并实际应用，才能真正掌握。

3.3.2 桌面命令菜单简介

在第 1 章图 1.3 的第一行给出了 MATLAB 6.x 的桌面命令菜单区，它包括【File】、【Edit】、【View】、【Web】、【Window】和【Help】六项。在第六项的右边，增加了一个显示当前目录的信息区。在主菜单上增加了 Web 项，表明它在联网功能上的加强，它的其他功能扩展主要反映在子菜单中。

在【File】下的子菜单中，增加了【Import Data...】（数据导入）、【Save Workspace As...】（将工作空间保存为文件）、【Set Path...】（搜索路径设定）和【Preference...】（选择）等选项。

在【Edit】下的子菜单各项，与一般文本编辑命令相仿，此处从简。只有【Paste Special】选项有些特别。用它可打开数据输入向导，将剪贴板的数据输入到 MATLAB 工作空间中。

在【View】下的子菜单中，增加了【Desktop Layout】（桌面布局）、【Undock Command Window】（与命令窗分离）、【Command Window】、【Command History】、【Current Directory】、【Workspace】、【Launch Pad】、【Help】、【Current Directory Filter】以及【Workspace View Options】等选项，用以选定观察的视窗。

在子菜单项【Desktop Layout】之下又有下一级子菜单,利用它可以同时显示两个以上的视窗。显示方案列在下一级子菜单中,分别为【Default】(默认方式,同时显示【Command Window】、【Launch Pad】和【Command History】3个视窗)、【Command Window Only】、【Simple】(同时显示【Command Window】和【Command History】或【Workspace】2个视窗)、【Short History】、【Tall History】(各显示【Command Window】、【Current Directory】和【Command History】3个视窗,但形状和排列不同),以及【Five Panel】(同时显示5个视窗)等。

【Web】是 MATLAB 6.x 新增的菜单项,通过该菜单项可以直接得到 MATLAB 的网络资源,当然此时系统必须在联网状态。

【Window】菜单项提供了在已打开的各 MATLAB 视窗之间的切换功能,也可以用它关闭全部视窗。

【Help】下的子菜单项【Full Product Family Help】、【MATLAB Help】、【Using the Desktop】、【Using the Command Windows】和【Demo】分类提供了进入各类帮助信息系统的入口。

3.3.3 MATLAB 6.x 的用户界面

双击 Windows 桌面上的 MATLAB 图标,就可以启动 MATLAB 软件环境。它首先显示出一个标志 MATLAB 6.x 的图形标志,经过几秒钟后,屏幕上将出现图 3.3 所示的 MATLAB 桌面平台,它由命令窗口(Command Window)、历史命令窗口及资源目录本等三个视窗组成。根据用户的需要,可以选定并激活相应的视窗进行操作。例如,在上述界面中关闭后两个视窗,或在【Desktop Layout】中选定【Command Window Only】,就会出现如图 1.3 所示的单一命令窗。

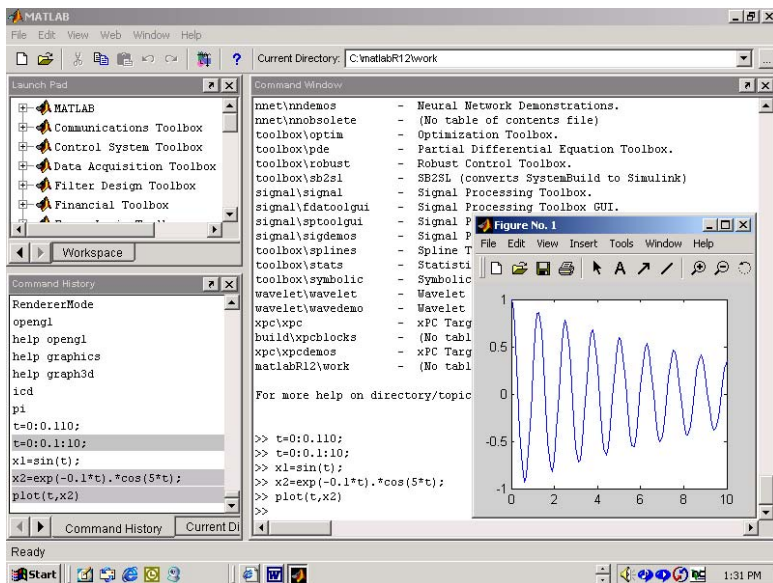


图 3.3 MATLAB 6.x 在默认条件下的桌面平台及历史命令窗的应用

图 3.3 中的图形窗是由命令窗中的 `plot` 语句打开的。画出这个衰减正弦曲线的有效命令有三条，要想把这三条命令组织起来，形成一个程序文件，就可以利用历史命令窗。如图 3.3 所示，在历史命令窗中用 `Ctrl+鼠标单击`，依次选出这三条语句，进行【Copy】，并将它们【Paste】到文本编辑器中去，再【Save】起来即可。也可以把它们【Paste】到命令窗中直接执行。

再看看 MATLAB 6.x 的帮助浏览器，在【View】下选择【Help】项，就会出现图 3.4 所示的帮助浏览器，左边是目录栏，右边是帮助的内容。在图 3.4 上可以看到，目录中有三个 `angle` 命令，它们出现在不同的工具箱中，图 3.4 中显示的是第一个 `angle` 命令的意义和用法。另外，浏览器的目录部分还给出了多种搜索与查找方法。

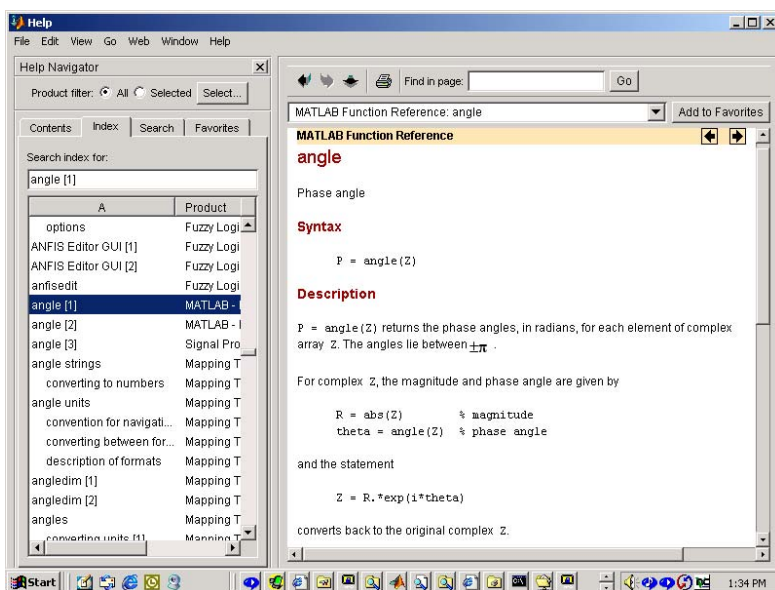


图 3.4 MATLAB 6.x 帮助浏览器

MATLAB 的其他函数库

在前三章中，介绍了最常见的十几个 MATLAB 基本部分的函数库，这些内容实际上已远远超过了 FORTRAN 语言和 C 语言在科学计算上的性能，对于大学低年级的学生来说，已经足以应付各种大学课程计算题的需要。余下的这几个函数库中，有些是涉及深一些的数学和物理概念，对大学低年级读者而言有些超前，不易很快接受，可以留到高年级再深入学习；有些则是用来编写较高级的程序用的，要对 MATLAB 编程比较熟悉之后再学。因此，我们单独把这些内容放在一章中，可以避免对初学者造成“拦路虎”现象。这一章内容具有独立性，其各节也有独立性，读者可以跳过整章或整节，向后阅读。但要读懂本章的各节，必须以前三章为基础，并具备相应的数学和理论知识。

4.1 数据分析函数库（datafun 函数库）

4.1.1 基本的数据分析

MATLAB 的基本数据处理功能是按列向进行的，因此要求待处理的数据矩阵按列向分类，而行向则表示数据的不同样本。例如，10 个学生的身高及三门课程分数列表如下：

data	=	154	49	83	67
		158	99	81	75
		155	100	68	86
		145	63	75	96
		145	63	75	96
		141	55	65	75
		155	56	64	85

```

147    89    87    77
147    96    54   100
145    60    76    67

```

进行简单数据处理的命令见表 4.1。

表 4.1 一些数据处理命令的结果

命 令	功 能	身 高	课程 1	课程 2	课程 3
max(data)	求各列最大值	158	100	87	100
min(data)	求各列最小值	141	49	54	67
mean(data)	求各列平均值	149.2	73.0	72.8	82.4
std(data)	求各列标准差	5.7504	20.4070	10.0421	12.0757
median(data)	求各列中间元素	147	63	75	81
sum(data)	求各列元素和	1492	730	728	824
trapz(data)	梯形法求积分	1342.5	675.5	648.5	757.0

其中大部分命令的意义很明确，不需解释。

- std（标准差）是指列中 N 个元素与该列平均值 `mean(data)` 之差的平方和开方。即

$$\text{std}(\text{data}) = \sqrt{\sum_N (\text{data} - \text{mean}(\text{data}))^2}$$

- trapz（求积分） 可以看成求和，梯形法求积分近似于求元素和，其差别在于梯形法是把相邻两点数据的平均值作为数据点。10 个数据只能产生 9 个数据点，相加以后比元素和少一组数据，把它乘以步长才真正表示了面积。其差额为半个首点和半个末点的数据和，即

$$\text{trapz}(\text{data}) = \text{sum}(\text{data}) - 0.5(\text{data}(1) + \text{data}(N))$$

在 N 很大时两者的误差很小。

有些数据处理命令的结果不是一个标量而是一个列向量，为了节省篇幅，只取数据中的前 3 行，其结果如表 4.2 所示。注意：结果一般与原数据具有同样的行数，只有求差分（`diff`）会减少一行，因为它是求相邻行之间的差。另外，`cumtrapz` 是 MATLAB 5.x 中的新增函数，相当于累计求面积。

表 4.2 产生列结果的数据处理命令

命 令	功 能	身 高	课程 1	课程 2	课程 3
cumsum(data(1:3,:))	列向累加和	154	49	83	67
		312	148	164	142
		467	248	232	228
cumprod(data(1:3,:))	列向累乘积	154	49	83	67
		24332	4851	6723	5025
		3771460	485100	457164	432150

续表

命 令	功 能	身 高	课程 1	课程 2	课程 3
diff(data(1:4,:))	列向差分	4	50	-2	8
		-3	1	-13	11
		-10	-37	7	10
sort(data(1:3,:))	列向重新排序	154	49	68	67
		155	99	81	75
		158	100	83	86
cumtrapz(data(1:4,:))*	列向累加积分(相当于不定积分)	156.0000	74.0000	82.0000	71.0000
		312.5000	173.5000	156.5000	151.5000
		462.5000	255.0000	228.0000	242.5000

4.1.2 用于场论的数据分析函数

MATLAB 用于场论的几个命令如下。

- gradient 用来求二维场和三维场的近似梯度，例如根据电位分布求电场就可用这个函数。
- del2 是二维场和三维场的拉普拉斯算子。
- cross 为两个向量的矢量积。
- dot 为两个向量的数量积。

设 i, j, k 为沿 x, y, z 方向的单位向量，则对于两向量 $a=a_xi+a_yj+a_zk$ 和 $b=b_xi+b_yj+b_zk$ 。向量的矢量积为（叉乘）： $a \times b = (a_yb_z - a_zb_y)i + (a_zb_x - a_xb_z)j + (a_xb_y - a_yb_x)k$
向量的数量积为（点乘）： $a \cdot b = a_xb_x + a_yb_y + a_zb_z$

在 MATLAB 中这两个向量可表为：

```
a=[ax,ay,az]; b=[bx,by,bz];  
cross(a,b) = [ay*bz-az*by, az*bx-ax*bz, ax*by-ay*bx];  
dot(a,b) = a*b';
```

4.1.3 用于随机数据分析的函数

MATLAB 有两个产生随机数的命令。

- rand(m,n) 产生在 0 与 1 之间均匀分布的 m 行 n 列随机数矩阵，其均值为 0.5，标准差（或均方根差）为 0.2887；
- randn(m,n) 产生正态分布的 m 行 n 列随机数矩阵，其均值为 0，标准差为 1；其分布情况可用直方图命令 hist(x,N) 来显示。其中 N 表示直方图横坐标的分割数，其默认值为 10。例如：

```
x=rand(1,1000);hist(x)
```

```
y=randn(1,1000);
```

得出的两组图形如图 4.1 所示。**Hist(x)** 是把 1000 个 x 中, 处于 $0 \sim 0.1$, $0.1 \sim 0.2$, ..., $0.9 \sim 1.0$ 各个区域中的数目分别清点出来, 画成直方图。如果 x 真是均匀分布的, 那这个图应该是水平直线, 实际上, 随机数规律是按统计方法确定的, 所以各区域的数量一般参差不齐, 只有数据量无限增加时, 此图才无限接近于理想情况。**hist(y, 50)** 则把 y 的最小值和最大值之间等分成 50 份进行统计, 得到一个钟形的, 即正态分布的曲线。

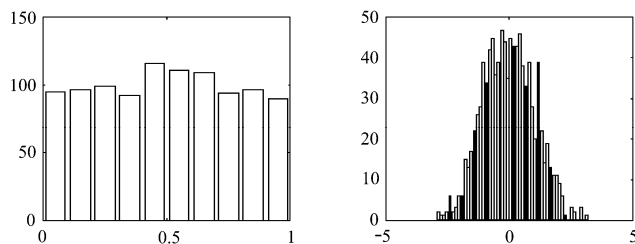


图 4.1 均匀分布与正态分布随机数直方图

4.1.4 用于相关分析和傅里叶分析的函数

相关分析(包括卷积)和傅里叶分析分别用于信号的时域和频域处理。这里给出了十几个函数, 它们是整个信号处理计算的基础。

- **corrcoef** 给出两个同长信号的相关系数, 例如对前面两个随机序列, 输入

```
R=corrcoef(x, y)
```

$$\text{得 } R = \begin{bmatrix} 1 & -0.0508 \\ -0.0508 & 1 \end{bmatrix}$$

对角线上是 x 和 y 的自相关系数, 这说明 x 和 y 的自相关性很强, 而互相关性则很弱。

- **cov(x,y)** 给出 x , y 的协方差矩阵, 对上述 x , y , 有

```
cov(x,y) = [0.0785 -0.0148; -0.0148 1.0782]
```

其主对角线上的值分别为 x 和 y 的均方差, 即标准差的平方(因为是随机数, 它不会严格等于理论值)。

- **conv(x, y)** 给出 x , y 的卷积。如果 x 是输入信号, y 是线性系统的脉冲过渡函数, 则 x , y 的卷积给出系统的输出信号。卷积函数也用于多项式相乘(见 4.3 节)。
- **filter(b, a, x)** 是根据输入信号 x 和线性系统特性求输出信号的函数。其不同在于系统的特性是以传递函数的分子多项式系数向量 b 和分母多项式系数向量 a 给出的, 而不是以脉冲过渡函数的形式给出的。
- **X=fft(x, N)** 求出时域信号 x 的离散傅里叶变换 X 。 N 为规定的点数。 N 的默认值为所给 x 的长度。当 N 取 2 的整数幂时变换的速度最快。通常取大于又最靠近 x 的幂次, 即令 $N=2^{\text{nextpow2}(\text{length}(x))}$ 。例如 x 的长度为 12, $\text{nextpow2}(12)=4$, $N=2^4=16$, 多出的各点补以零。一般情况下, **fft** 求出的函数为复数, 可用 **abs** 及 **angle** 分别求其幅度和相位。在画频谱图时往往最关心其幅频特性。

【例 4.1】 给出一个信号：

```
t=0:.001:3; u=sin(300*t)+2*cos(200*t) ;
```

它的幅频特性可用下列语句求得：

```
U=fft(u);plot(abs(U))
```

得出的频谱曲线如图 4.2（a）所示，它对采样频率呈对称形式。为了把它看得更清楚，把坐标缩小，输入 axis([0,300,0,3000])，得出的频谱曲线如图 4.2（b）图所示。

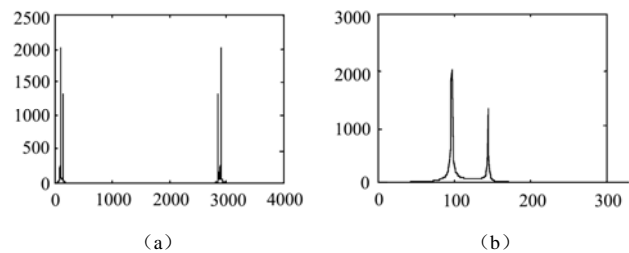


图 4.2 例题中信号的幅频曲线

- x=ifft（X） 傅里叶反变换函数，其用法与 fft 相仿。
- sound（u，s） 会在音箱中产生 u 所对应的声音，s 规定重放的速度，其默认值为 8192（bps-bit per second），如表 4.3 所示。

表 4.3 数据分析和傅里叶变换函数库(datafun)(d)

基本运算	max	最大元素	sum	元素之和
	min	最小元素	prod	元素之积
	mean	平均值	cumsum	元素的累加和
	median	中间值	cumprod	元素的累积
	std	标准差	hist	直方图
	sort	按升序排列	trapz	用梯形法作定积分
	sortrows	按升序排列行	cumtrapz	用梯形法作不定积分
差分	diff	差分函数和近似微分	gradient	近似梯度
	del2	五点离散拉普拉斯算子		
相关运算	corrcoef	相关系数		
	cov	协方差矩阵		
滤波和卷积	filter	一维数字滤波	filter2	二维数字滤波
	conv	卷积和多项式相乘	conv2	二维卷积
	convn	n 维卷积	deconv	反卷积和多项式相除
傅里叶变换	fft	离散傅里叶变换	ifft	离散傅里叶反变换
	fft2	二维离散傅里叶变换	ifft2	二维离散傅里叶反变换
	fftn	n 维离散傅里叶变换	ifftn	n 维离散傅里叶反变换
	fftshift	将零延迟移到频谱中心		
声音函数	sound	把向量放成声音	mu2lin	把 mu 规律编码变为线性信号
	soundsc	自动设比把向量放成声音	lin2mu	把线性信号变为 mu 规律编码

4.2 矩阵的分解与变换 (matfun 函数库)

4.2.1 线性方程组的系数矩阵

在 2.2 节中提到了可以用矩阵除法来解线性方程组，本节将讨论有关解线性方程组的一些深入的问题及其工具函数。这些函数见表 4.4 中的矩阵分析和线性方程部分。

$\text{Det}(a)$ 用以求方阵 a 的行列式。若 $\det(a)$ 不等于零，则 a 的逆阵 $\text{inv}(a)$ 存在。线性方程组的系数矩阵只有满足这个条件，它的解才存在。 $\text{Rank}(a)$ 用以求任意矩阵 a 的秩。也就是它所能划分出的行列式不为零的最大方阵的边长。 $\text{Trace}(a)$ 求出矩阵主对角线上元素的和（即迹）。

如果 $\det(a)$ 虽不等于零，但数值很小，接近于零，则这样的线性方程组称为病态的。其解的精度比较低。为了评价线性方程组系数矩阵的病态程度，用了条件数（condition number）的概念。条件数愈大，方程病态愈重，解的精度愈低。若系数矩阵的条件数很大而又拿它作除数（即解此线性方程组）时，MATLAB 会提出警告：“条件数太大，结果可能不准确”，求条件数的函数为 $\text{cond}(a)$ ， a 可以不是方阵。

在线性方程组 $A*x=B$ 中，系数矩阵 A 的行数 n 表示方程的数目，其列数 m 表示未知数的数目。在正常情况下，方程个数等于未知数个数，即 $n=m$ 。 A 为方阵， $A \setminus B$ 意味着 $\text{inv}(A)*B$ 。实际上，对于方程个数大于未知数个数（ $n>m$ ）的超定方程组，以及方程个数小于未知数个数（ $n<m$ ）的不定方程组，MATLAB 中 $A \setminus B$ 的算式都仍然合法。前者是最小二乘解；而后者则是令 x 中 $m-n$ 个元素为零的一个特殊解。这两种情况下，因为 A 不是方阵，其逆 $\text{inv}(A)$ 不存在，解的 MATLAB 算式均为 $x=\text{inv}(A'*A)*(A'*B)$ 。把 $\text{pinv}(A) = \text{inv}(A'*A)*A'$ 定义为伪逆函数，则 $A \setminus B$ 就等于 $\text{pinv}(A)*B$ 。由于 MATLAB 中引入了伪逆的概念，除矩阵 A 就可以不是方阵。

【例 4.2】 求下列矩阵的行列式、秩、逆阵、迹和条件数。

$$a = \begin{bmatrix} 2 & 9 & 0 & 0 \\ 0 & 4 & 1 & 4 \\ 7 & 5 & 5 & 1 \\ 7 & 8 & 7 & 4 \end{bmatrix}$$

解：■ MATLAB 实现

```
det(a) = -275
rank(a) = 4
inv(a) =
-0.0727    0.4255    0.7855   -0.6218
 0.1273   -0.0945   -0.1745    0.1382
 0.0000   -0.6000   -0.8000    0.8000
-0.1273    0.4945    0.3745   -0.3382
```

```
trace(a) = 15
cond(a) = 33.4763
```

4.2.2 矩阵的分解

矩阵可以分解为几个具有特殊构造性质的矩阵的乘积,这是分析矩阵的一种重要手段,而这种分解在数学计算上通常又是非常烦琐的工作。MATLAB 提供了一些现成的函数可供调用,主要有表 4.4 所列的几种。

表 4.4 矩阵函数和数值线性代数函数库 (matfun) (m)

矩阵分析	norm	矩阵或向量的范数	null	零空间正交基
	normest	矩阵 2 范数的估值	orth	正交化
	rank	矩阵的秩	rref	缩减行梯次格式
	det	行列式(必须是方阵)	subspace	两个子空间之间的夹角
	trace	主对角线上元素的和		
线性方程	\ 和 /	线性方程求解	qr	正交三角分解
	chol	Cholesky 分解	cholinc	不完全 Cholesky 分解
	cond	矩阵条件数	condest	1 范数条件数的估值
	rcond	linpack 逆条件数计算	nnls	非负最小二乘
	lu	高斯消去法系数矩阵	pinv	矩阵伪逆
	inv	矩阵求逆(必须是方阵)	lscov	协方差已知的最小二乘
特征值和奇异值	eig	特征值和特征向量	eigs	若干特征值
	poly	特征多项式(必须是方阵)	condeig	对应于特征值的条件数
	polyeig	多项式特征值问题	schur	Schur 分解
	hess	Hessenberg 形式	balance	均衡(改善条件数)
	qz	广义特征值	svd	奇异值分解
矩阵函数	expm	矩阵指数	expm2	用泰勒级数求矩阵指数
	expm1	用 M 文件求矩阵指数	expm3	用特征值求矩阵指数
	logm	矩阵对数	funm	通用矩阵函数的计算
	sqrtn	矩阵开方		
分解工具	qrdelete	从 QR 分解中删去一列	rsf2csf	实对角阵变为复对角阵
	qrinsert	在 QR 分解中插入一列	cdf2rdf	复对角阵变为实对角阵
	planerot	Given's 平面旋转		

● 三角分解 (lu 分解)

它把一个任意方阵分解为一个准下三角方阵和一个上三角方阵的乘积。由于此函数有两个输出矩阵,其左端应有两个变量 l 和 u 。

输入

```
[l,u]=lu(a)
```

```

得  l = 0.2857  1.0000  0      0
      0      0.5283  0.6838  1.0000
      1.0000  0      0      0
      1.0000  0.3962  1.0000  0
  u = 7.0000  5.0000  5.0000  1.0000
      0      7.5714 -1.4286 -0.2857
      0      0      2.5660  3.1132
      0      0      0      2.0221

```

所谓准下三角阵 l 是因为必须交换两行才能成为真的下三角阵。它的行列式绝对值等于 1 (可正可负), 上三角阵 u 的行列式等于 a 的行列式。 lu 分解只能对方阵进行, 它常用于高斯消去法。

● 正交分解 (qr 分解)

$qr(a)$ 把任意矩阵 $n \times m$ 阶 a 分解为一个正交方阵 q 和一个与 a 有同样阶数的上三角矩阵 r 的乘积。该方阵 q 的边长为矩阵 a 的 n 和 m 中之小者, 且其行列式的值为 1。

【例 4.3】 求下列 3×5 阶矩阵 b 的 qr 分解。

```

b = 0.2190    0.6793    0.5194    0.0535    0.0077
     0.0470    0.9347    0.8310    0.5297    0.3834
     0.6789    0.3835    0.0346    0.6711    0.0668

```

解: ■ MATLAB 实现

输入 `[q,r]=qr(b)`

```

得  q = -0.3063 -0.4667 -0.8297
      -0.0658 -0.8591  0.5076
      -0.9497  0.2101  0.2324
  r = -0.7149 -0.6338 -0.2466 -0.6886 -0.0911
      0      -1.0395 -0.9490 -0.3390 -0.3189
      0      0      -0.0011  0.3805  0.2038

```

● 奇异值分解 (svd 分解)

$svd(a)$ 把任意 $n \times m$ 阶矩阵 a 分解为三个矩阵的乘积, 即 $a=usv$ 。其中 u , v 为 $n \times n$ 阶和 $m \times m$ 阶正交方阵, s 则为 $n \times m$ 阶的对角阵。对角线上的元素就是矩阵 a 的奇异值, 其长度为 n 和 m 中的较小者。

例如对上述 b 作奇异值分解, 输入

```

[u,s,v]=svd(b)
得 u = -0.4623  0.2273  0.8571
      -0.7822  0.3507 -0.5149
      -0.4176 -0.9085  0.0157
  s = 1.7539  0      0      0      0

```

```

0          0.7995  0          0          0
0          0          0.3534  0          0
v = 0.2403 -0.6885  0.4927 -0.4748  0
    0.6872  0.1674  0.3027  0.4193  0.4819
    0.5158  0.4729  0.0506 -0.3723 -0.6076
    0.4102 -0.5151 -0.6122  0.3194 -0.2994
    0.1890  0.0944 -0.5369 -0.5985  0.5558

```

矩阵最大奇异值和最小奇异值之比就是它的条件数。即：

```
cond(b)=max(diag(s))/min(diag(s))
```

4.2.3 矩阵的特征值分析

`eig(a)` 用来求方阵 `a` 的特征根和特征向量，其输出量有两个，即特征向量 `e` 和特征根 `r`。

输入 `[e,r]=eig(a)`

```

得 e = -0.2568 -0.3834 + 0.4681i -0.3834 - 0.4681i  0.6167
        -0.3481 -0.2177 - 0.2869i -0.2177 + 0.2869i -0.1850
        -0.4682  0.5152 + 0.2228i  0.5152 - 0.2228i -0.6624
        -0.7705  0.4217 - 0.1060i  0.4217 + 0.1060i  0.3829
r = 14.2004  0          0          0
      0      0.7495 + 5.2088i  0          0
      0      0          0.7495 - 5.2088i  0
      0      0          0          -0.6993

```

特征根是特征方程的根，矩阵的特征方程系数可用 `poly` 函数求出，再用 `roots` 命令求出其特征根。例如

输入 `p=poly(a)`

```
得 p = 1.0000 -15.0000 38.0000 -359.0000 -275.0000
```

```

而 roots(p) = 14.2004
              0.7495 + 5.2088i
              0.7495 - 5.2088i
              -0.6993

```

结果与 `eig` 函数求出的特征根相同，但 `roots` 函数不能求特征向量。

4.2.4 特殊矩阵库 (specmat)

有一些特殊构造的矩阵在矩阵变换中很有用处，MATLAB 4.x 将它们组成一个专门的

函数库。读者在应用中遇到有关的矩阵，即可在此调用。其调用的参数和调用方法均可从 help 文本中查得，此处不多占篇幅。MATLAB 5.x 已把这个库并入 elmat 库中，见表 2.1 中的“特殊矩阵”栏。

4.3 多项式函数库 (polyfun)

一元高次代数多项式 $a(x) = a_1x^n + a_2x^{n-1} + \cdots + a_nx + a_{n+1}$ ，在 MATLAB 中可以用它的系数向量 $a=[a(1), a(2), \cdots, a(n), a(n+1)]$ 来表示。其幂次已隐含在系数元素离向量右端的元素的间隔中。要注意，如果 x 的某次幂的系数为零，这个零必须列入系数向量中。在微分方程中，通过微分积分运算可把线性系统的特性表示为两个算子 s 的多项式之比。因此，多项式在近代信息和控制理论中有着十分重要的地位。

多项式和插值函数库存如表 4.5 所示。

表 4.5 多项式和插值函数库(polyfun)(r)

多项式	roots	多项式求根	polyfit	用多项式曲线拟合数据
	poly	按根组成多项式	polyder	多项式及求导数
	polyval	多项式求值	conv	多项式相乘，卷积
	polyvalm	矩阵作变元的多项式求值	deconv	多项式相除，反卷积
	residue	部分分式展开（留数）		
数据插值	interp1	一维插值（一维查表）	interpft	用 FFT 方法的一维插值
	interp1q	快速一维线性插值	griddata	网格数据生成
	interp2	二维插值（二维查表）	griddata3	三维网格数据生成
	Interp3	三维插值（二维查表）	griddata4	N 维网格数据生成
	interpnp	N 维插值（二维查表）		
样条函数插值	spline	三次样条函数插值	unmkpp	提供分段多项式的细节
	ppval	分段多项式计算	table1	一维插值表
	mkpp	构成分段多项式	table2	二维插值表
几何分析	delaunay	Delaunay 三角化	voronoi	Voronoi 图
	Delaunay3	三维 Delaunay 三角化	voronoin	N 维 Voronoi 图
	delaunayn	N 维 Delaunay 三角化	polyarea	多边形区域
	Dsearch	最近的 Delaunay 三角化	inpolygon	点在多边形内时为真
	tsearch	最近的三角搜索	rectint	矩形相交区域
	convhull	突壳	convhulln	N 维突壳
工具	xychk	向一维和二维数据变元检查	abcdchk	检查矩阵 A,B,C,D 的一致性
	xyzchk	向三维数据变元检查	ss2tf	状态空间变为传递函数
	xyzvchk	向三维容量数据变元检查	ss2zp	状态空间变为零极增益
	automesh	如果输入是自成网格的为真	tf2ss	传递函数变为状态空间
	mkpp	制造分段多项式	tf2zp	传递函数变为零极增益

续表

工具	unmkpp	提供分段多项式的细节	tfchk	传递函数正确性检查
	resi2	求重极点的留数	zp2ss	零极增益变为状态空间
	tzero	传送零点	zp2tf	零极增益变为传递函数
过时的函数	icubic	一维立方插值	interp6	二维最近邻居插值
	nterp4	二维双线性数据插值	table1	一维表格查找
	interp5	二维双立方数据插值	table2	二维表格查找

4.3.1 多项式的四则运算

【例 4.4】 设有两个多项式 $a(x)=2x^3+4x^2+6x+8$ 及 $b(x)=3x^2+6x+9$ ，要求对此两个多项式作如下运算。

- 多项式相乘(conv) conv 函数本来是卷积（convolution）的意思。但它也符合多项式相乘的运算规则。分析如下：

可以想象把系数向量 a 正常排列，而把 b 反转，先将 a（1）与 b（1）对齐：

a(1)

a(2)

a(3)

a(4)

b(3)

b(2)

b(1)

把上下对应的项相乘，a(1)*b(1)得出多项式乘积 c 的最高次项系数 c(1)；

将 b 右移一位，把上下对应的项相乘并求和，a(1)*b(2)+a(2)*b(1)为次高次项系数 c(2)；

依此类推，可得到乘积 c 的全部系数。这种运算和卷积运算的规则完全相同。

■ MATLAB 实现

输入 a=[2,4,6,8],b=[3,6,9], c=conv(a,b), 得

a =

2

4

6

8

b =

3

6

9

c =

6

24

60

96

102

72

- 多项式相加 MATLAB 规定，只有长度相同的向量才能相加。因此，必须把短的向量前面补以若干个零元素，才能用 MATLAB 的矩阵加法运算符。

■ MATLAB 实现

输入 d=a+[0,b]

得 d = 2 7 12 17

这种手工数两个多项式的长度再补零的方法是不可取的，必须要让计算机自动完成。为此可编一个子程序 polyadd.m，其内容为：

```
function y=polyadd(x1,x2)
n1=length(x1); n2= length(x2);
if n1>n2 x2=[zeros(1,n1-n2),x2];
```

```
elseif n1<n2 x1=zeros(1,n2-n1),x1];
end, y = x1+x2;
```

这样，多项式相加就可写成： $c = \text{polyadd}(a,b)$ ；相减可另编一个子程序，或在 polyadd 的输入变元中加负号来实现。

- 多项式相除 相除是相乘的逆运算，但除法不一定除得尽，会有余子式。

■ MATLAB 实现

```
输入 [q,r]=deconv(c,a)
q =      3      6      9
r =      0      0      0      0      0      0
```

其中 q 是商式， r 是余子式。因为用的是相乘的数据 a 和 c ，恰好除尽。如令 $a1=a+1$ ，则有

```
a1 =      3      5      7      9
[q1,r1]=deconv(c,a1)
q1 =      2.0000      4.6667      7.5556
r1 =      0          0          0      7.5556      7.1111      4.0000
```

可以用商式与除式相乘，再加上余式的方法来检验：

```
c1=conv(q1,a1)+r1
得 c1 =      6      24      60      96      102      72
与 c 相同。
```

4.3.2 多项式求导、求根和求值

- 多项式求导数(polyder)

■ MATLAB 实现

```
输入 e=polyder(c)
得 e =      30      96      180      192      102
```

- 多项式求根(roots 和 poly 函数)

■ MATLAB 实现

```
输入 ra=roots(a);rb=roots(b);rc=roots(c);ra,rb,rc
得 ra =-1.6506
      -0.1747 + 1.5469i
      -0.1747 - 1.5469i
rb =   -1.0000 + 1.4142i
      -1.0000 - 1.4142i
```

rc 是 ra 和 rb 的并，这是完全可以预计到的。

由根求多项式系数是 `roots` 的逆运算，其函数名也是 `poly`。

```
a = poly(ra); b = poly(rb)
```

从 `poly` 函数的用法可以看出 MATLAB 的智能特点，当 `a` 是向量时，`poly` 把它看做根来组成多项式；当 `a` 是方阵时，`poly` 用它组成方阵的特征多项式。

- 多项式求值 (`polyval`) 将多项式 `a` 中的自变量 `x` 赋予值 `xv` 时，该多项式的值可用 `F = polyval(a,xv)` 求得

其中 `xv` 可以是复数，也可以是矩阵或数组，此时 `polyval` 对输入变元作元素群运算，这对于求线性系统的频率特性特别方便。`polyvalm` 则对输入的变元阵（必须是方阵）做矩阵多项式运算。

【例 4.5】 设 `a` 为系统分母系数向量，`b` 为系统分子系数向量，求此系统的频率响应并画出频率特性。

解：■ MATLAB 实现

先令频率数组 `w` 取线性间隔：

```
w=linspace(0,10);           % 在 w=0 到 10 之间按线性间隔取 100 点（默认值）
A=polyval(a,j*w);B=polyval(b,j*w); % 分别求分母分子多项式的值（为复数数组）
subplot(2,1,1);plot(w,abs(B./A)), % 画两者元素群相除所得的幅频特性
subplot(2,1,2);plot(w,angle(B./A)) % 画相频特性
```

频率特性通常在对数坐标中绘制。因此，输入的频率数组取对数等间隔：

```
w1=logspace(-1,1)           % 在 w1 从  $10^{-1}$  到 10 之间，按对数分割为 50 点（默认值）
F=polyval(b,j*w1)./polyval(a,j*w1); % 求出这些点上的频率响应（复数）
subplot(2,1,1),loglog(w1,abs(F))    % 在双对数坐标中画出幅频特性
subplot(2,1,2);semilogx(w1,angle(F)) % 在双对数坐标(x)中画出相频特性
```

所得曲线如图 4.3 所示。

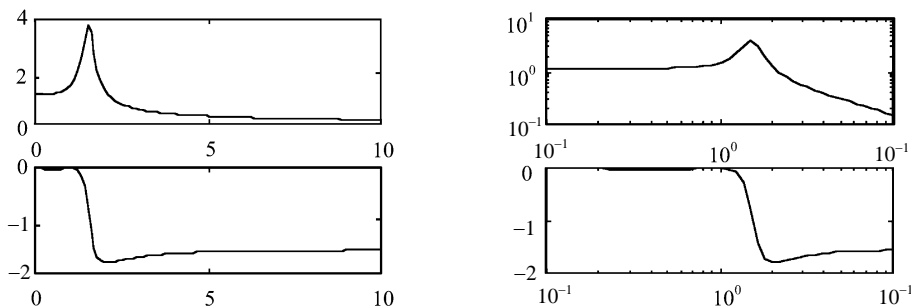


图 4.3 线性坐标和对数坐标中的频率特性

4.3.3 多项式拟合

- `p=polyfit(x,y,n)` 用于多项式曲线拟合。其中 `x`, `y` 是已知的 `N` 个数据点坐标向

量, 当然其长度均为 N 。 n 是用来拟合的多项式次数, p 是求出的多项式的系数, n 次多项式应该有 $n+1$ 个系数, 故 p 的长度为 $n+1$ 。拟合的准则是最小二乘法。

【例 4.6】 设原始数据为 x 时在 11 个点上测得的 y 值:

```
x=0:0.1:1;y=[-0.447,1.978,3.28,6.16,7.08,7.34,7.66,9.56,9.48,9.30,11.2];
```

■ MATLAB 实现

- 线性拟合: `a1=polyfit(x,y,1);`

求出 $a1$ 后, 可求出 `xi=linspace(0,1);` (即 100 个点) 上的 $yi1$ 值并绘图:

```
yi1=polyval(a1,xi); plot(x,y,'o',xi,yi1,'b'),pause
```

其中原始数据用圆圈标出, 而拟合曲线为蓝色。依此类推, 有:

- 二次拟合: `a2=polyfit(x,y,2);yi2=polyval(a2,xi);plot(x,y,'o',xi,yi2,'m')`
- 三次拟合: `a3=polyfit(x,y,3);yi3=polyval(a3,xi);plot(x,y,'o',xi,yi3,'r')`
- 九次拟合: `a9=polyfit(x,y,9);yi9=polyval(a9,xi);plot(x,y,'o',xi,yi9,'c')`
- 十次拟合: `a10=polyfit(x,y,10);yi10=polyval(a10,xi);plot(x,y,'o',xi,yi10,'g')`

所得的曲线如图 4.4 所示。给定 11 点的最大拟合阶次为 10, 此时拟合曲线将通过全部给定点。可以看出, 拟合曲线的阶次太高会造成曲线振荡, 反而看不出函数关系的基本规律, 并不一定好。

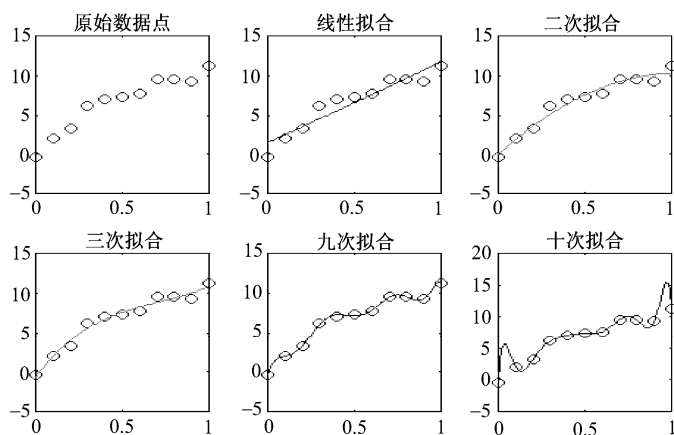


图 4.4 不同的逼近次数产生的不同曲线

4.3.4 多项式插值

插值和拟合的不同处有以下几点。

- 插值函数通常是分段的, 因而人们关心的不是函数的表达式, 而是插值得到的数据点。

插值数据应通过给定的数据点 x, y 。插值函数一般可表示为：

- $yi=interp1(x,y,xi, 'method')$ 其中 xi 为插值范围内的任意点集的 x 坐标, yi 是插值后的对应数据点集的 y 坐标。 $method$ 为插值函数的类型选项, 有 `linear` (线性, 默认项)、`cubic` (三次) 和 `cubic spline` (三次样条) 三种。

1. 一维插值函数 `interp1`

【例 4.7】 仍取例 4.6 中的 x, y, xi , 求其线性和三次插值曲线。

■ MATLAB 实现

- 线性插值: `yi1=interp1(x,y,xi);plot(x,y,'o',xi,yi1)`
- 三次插值: `yi2=interp1(x,y,xi,'spline');plot(x,y,'o',xi,yi2,'g')`

所得曲线如图 4.5 及图 4.6 所示, 三次插值的结果比较光滑。

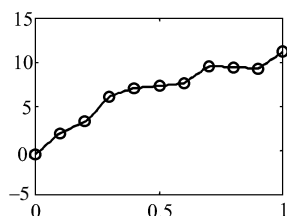


图 4.5 线性插值曲线

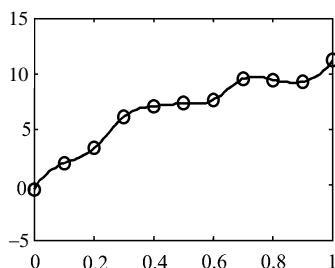


图 4.6 三次插值曲线

2. 二维插值函数 `zi=interp2(x,y,z,xi,yi,'method')`

【例 4.8】 已知某矩形温箱中 3×5 个测试点上的温度, 求全箱的温度分布。

给定: `width=1:5;depth=1:3;temps=[82 81 80 82 84;79 63 61 65 81;84 84 82 85 86]`;

要求计算沿宽度和深度细分网格: `di=1:0.2:3;wi=1:0.2:5`; 交点上的温度。

解: ■ MATLAB 实现

```
tc=interp2(width,depth,temps,wi,di,'cubic'); %求各点温度
mesh(wi,di,tc) %画三维曲面
```

所得温度分布图形如图 4.7 所示。

注意 `interp2` 中所用的 `wi, di` 是宽度和深度方向的细分坐标向量, `di` 必须变换为列向量, 插值函数运算时会自动将它们转变为宽度乘深度平面上的网格, 并计算网格点上的温度。

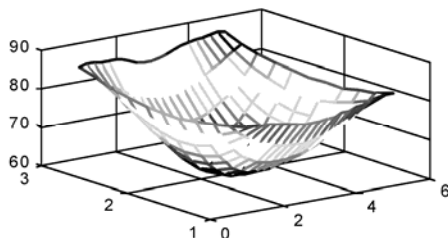


图 4.7 二维插值的曲面

4.3.5 线性微分方程的解 (residue)

线性常微分方程的解可用拉普拉斯算子 s 表示为

$$Y(s) = B(s)/A(s)$$

其中 $B(s)$ 和 $A(s)$ 都是 s 的多项式, 分母多项式的次数 n 通常高于分子多项式的次数 m 。在时间域的解 $y(t)$ 是 $Y(s)$ 的拉普拉斯反变换。求反变换的重要方法之一是部分分式法, 即将上述多项式分解为多个 s 的一次分式之和。留数函数 `residue` 可以完成这一任务。

步骤:

(1) 用 `[r,p,k]=residue(b,a)` 求出 $Y(s)$ 的极点数组 p 和留数数组 r (设分母比分子阶数高, 故 $k=0$), 因而 $Y(s)$ 可表示为

$$Y(s) = \frac{r(1)}{s-p(1)} + \frac{r(2)}{s-p(2)} + \frac{r(3)}{s-p(3)} + \frac{r(4)}{s-p(4)} + \dots$$

(2) 求它的反变换。得

$$y(t) = r(1)*\exp(p(1)*t) + r(2)*\exp(p(2)*t) + r(3)*\exp(p(3)*t) + r(4)*\exp(p(4)*t) + \dots$$

【例 4.9】 求解线性常微分方程

$$y''' + 5y'' + 4y' + 7y = 3u'' + 0.5u' + 4u$$

在输入 $u(t)$ 为单位脉冲及单位阶跃信号时的解析解。

解: 用 Laplace 变换 (脉冲输入 $u(s)=1$, 阶跃输入 $u(s)=1/s$)

$$y(s) = \frac{3s^2 + 0.5s + 4s}{s^3 + 5s^2 + 4s + 7} u(s) = \frac{b(s)}{a(s)}$$

■ 在脉冲输入时的响应

$$a=[1,5,4,7]; b=[3,0.5,4]; [r,p,k]=residue(b,a)$$

得

$$r = \begin{matrix} 3.2288 \\ -0.1144 + 0.0730i \\ -0.1144 - 0.0730i \end{matrix}$$

$$p = -4.4548$$

$$\begin{matrix} -0.2726 + 1.2235i \\ -0.2726 - 1.2235i \end{matrix}$$

$$k = []$$

求时域解, 先设定时间数组 $t=0:0.2:10$; 然后列出:

$$yi=r(1)*\exp(p(1)*t)+r(2)*\exp(p(2)*t)+r(3)*\exp(p(3)*t); plot(t,yi)$$

■ 在阶跃输入时的响应 (此时分母由于乘了个 s , a 将提高一阶, 右端多加一个零);

$$a=[1,5,4,7,0]; b=[3,0.5,4]; [r,p,k]=residue(b,a)$$

$$r = -0.7248$$

```
0.0767 + 0.0764i
0.0767 - 0.0764i
0.5714 + 0.0000i
p = -4.4548
-0.2726 + 1.2235i
-0.2726 - 1.2235i
0
k = [ ]
ys=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t)+r(3)*exp(p(3)*t)+r(4);plot(t,ys)
```

所得曲线如图 4.8 所示。

MATLAB 中的这一类命令有一个共同特点，其输入变元不仅有矩阵变量，而且有函数名。在执行这些命令时，要不断地调用函数作为输入变元。因此，完成这一类命令的功能比较丰富而灵活，掌握了它就能编写出更为清晰的程序。

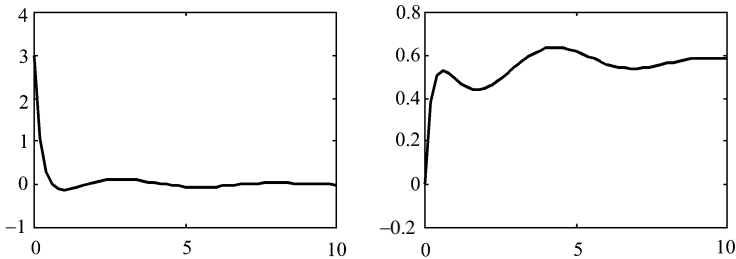


图 4.8 脉冲过渡响应和阶跃过渡响应

4.4 函数功能和数值积分函数库（funfun）

4.4.1 函数功能和数值积分函数库的主要子程序

函数功能和数值积分函数库的主要子程序如表 4.6 所示，将它分为两类来探讨。第一类是对任意非线性函数的分析，包括求极值，过零点等；第二类求任意函数的数值积分，包括定积分和微分方程的数值解等。它们的共同特点是必须会自行定义函数。

表 4.6 函数功能和数值积分函数库（funfun）（e）

	函 数 名	功能说明	需调用的函数
最优化和求根	fmin	单变量函数求极小值 y_{\min}	给出待分析的函数 $y=f(x)$
	fmins	多变量函数求极小值	
	fzero	单变量函数求 $y=0$ 处的 x	
数值定积分	quad	数值积分计算（低阶）	给出被积分的函数 $f(x)dy/dx=f(x)$ ，求 y
	quad8	数值积分计算（高阶）	
	dblquad	双精度数值积分	

续表

	函 数 名	功能说明	需调用的函数
函数绘图	ezplot	简便的函数绘图器	
	fplot	画函数曲线 $y=f(x)$	
内联 (INLINE) 函数对象	inline	构成 INLINE 函数对象	
	argnames	变元名	
	formula	函数公式	
	char	把 INLINE 函数转换为字符数组	
	vectorize	使字符串或 INLINE 函数向量化	
常微分方程数值积分器	ode45	解非刚性微分方程 (中阶方法)	给出导数的函数表达式 $dy/dx=f(y,x)$, 求 y
	ode23	解非刚性微分方程 (低阶方法)	
	ode113	解非刚性微分方程 (变阶方法)	
	ode15s	解刚性微分方程 (变阶方法)	
	ode23s	解刚性微分方程 (低阶方法)	
	odefile	ODE 文件语法	
ODE 输出函数	odeplot	时间序列	
	odephas2	ODE 输出函数的二维相平面	
	odephas3	ODE 输出函数的三维相空间	
	odeprint	打印 ODE 输出函数	

下面取本书 2.6 节中定义过的函数 `humps.m` 来说明这些子程序的用法。它定义了下列非线性函数：

$$y = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x+0.9)^2 + 0.04} - 6$$

4.4.2 非线性函数的分析

■ 绘制函数曲线 `fplot`

其格式为：`fplot('函数名', [初值 x0, 终值 xf])`

例如，要画出 `humps` 函数在 $x=0\sim 2$ 之间的曲线，可输入

```
fplot('humps',[0,2]),grid
```

得出如图 4.9 所示的曲线。

`fplot` 函数对于快速了解一些复杂特殊函数的波形很有用处。例如，求其中第一类 Bessel 函数 (见表 4.7)，可用

```
fplot('besselj(alpha,x) ',[0,10])
```

设 α 为 1, 2, 5 时，得到如图 4.10 所示的曲线。

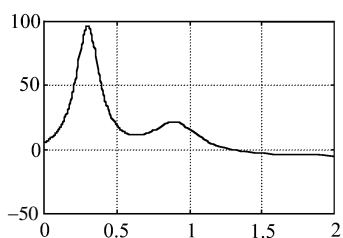


图 4.9 humps 函数的曲线

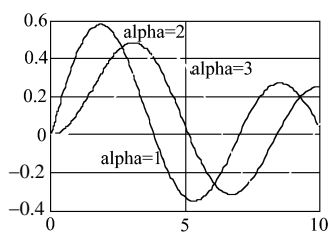


图 4.10 第一类 Bessel 函数的曲线

■ 求函数极值 fmin

其格式为: `fmin('函数名', 初值 x0, 终值 xf)`,

例如, 求 humps 函数在 $x=0\sim 1.5$ 之间的极小值, 则

输入 `m=fmin('humps',0,1.5)`

得 `m = 0.6370`

■ 求函数零点 fzero

其格式为: `fzero('函数名', 初值 x0)`

例如, 求 humps 函数在 $x=1$ 附近的过零点, 则

输入 `z=fzero('humps',1),`

得 `z = 1.2995`

以上给出的是这些函数调用的典型格式, 还有其他选项可作为变元, 例如

```
fplot('tan',[-2*pi 2*pi -2*pi 2*pi],'*'),grid
```

在第 2 项变元中增加了 y 轴的上下限, 第 3 项变元是线型。所得图形如图 4.11 (a) 所示, 读者可从 `help fplot` 中得到进一步的信息。

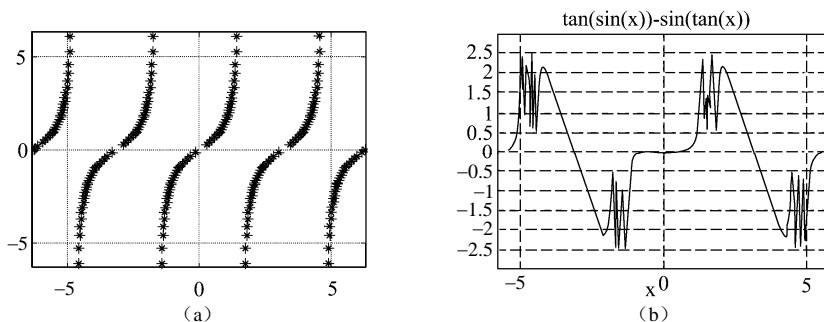


图 4.11 由 fplot 和 ezplot 画出的曲线

还有一个简便画函数图的命令 `ezplot` (读作 easy plot), 它连自变量范围都无须规定, 其默认的自变量范围为 $[-2\pi, 2\pi]$ 。因此只要输入

```
ezplot tan(x),grid
```

也可得到类似于图 4.11 (a) 的曲线, 只是 * 号变为了实线。若输入

```
ezplot tan(sin(x))-sin(tan(x))
```

所得图形见图 4.11 (b)。可以看出, 图上还自动进行了标注。

此外, 还有简便绘制三维曲面的命令 `ezmesh` 和 `ezsurf`, 读者可从 `help` 命令找到它们的用法。

4.4.3 任意函数的数值积分

● 定积分子程序 (`quad` 及 `quad8`) 的格式为:

➤ `quad('函数名', 初值 x0, 终值 xf)`,

例如, 求 `humps` 函数在 $x=1\sim 2$ 之间的定积分。

输入 `s=quad('humps',1,2)`

得 `s = -0.5321`

不难用定积分函数来求不定积分的数值解。只要固定积分下限, 用 `for` 循环, 把积分上限逐步增加即可。

例如要求 `humps` 函数以 $x=0$ 为下限的不定积分, 可编写下列程序:

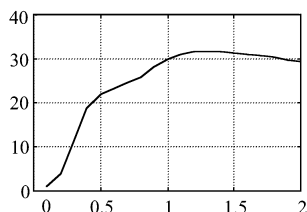


图 4.12 `humps` 函数的积分曲线

```
for i=1:20
    x(i)=0.1*i;
    y(i)=quad('humps',0,x(i));
end, plot(x,y)
```

得出的曲线如图 4.12 所示。可以与图 4.9 对照, 确认它是 `humps` 曲线的积分。

● 微分方程数值解 (`ode23`, `ode45` 等)

如果微分方程可化为一阶微分方程组的形式:

$$dy/dx = f(x,y)$$

其中, x 是标量, y 可以是一个列向量。

$F(x,y)$ 是以 x, y 为变元的函数, 用 MATLAB 函数文件表述, 设文件名为 `yprime.m`, 则求此微分方程的数值解的子程序调用格式为:

➤ `[x,y]=ode23('yprime', 自变量初值 x0, 自变量终值 xf, 因变量初值 y0)`

对于 `humps` 函数, 不能直接用 `ode23` 做数值积分, 其原因在于 `humps` 只有一个输入变元 x 。而微分方程数值解的函数 `ode23`, 要求被调用的函数有两个输入变元。如果把 `humps` 函数文件加一个虚的变元 y , 即把它的第一句换成 `function yp = humps1(x,y)`, 并将此函数另存成一个 `humps1.m` 文件, 则

```
[x,y] = ode23('humps1', 0, 2, 1); plot(x,y)
```

表示在初值 $y_0=1$ 的条件下, 从 $x_0=1$ 到 $x_f=2$ 求微分方程的数值解, 运行后可以得到与图 4.12 相仿的曲线, 只是向上平移了一个单位, 因为这里设了 $y_0=1$ 。在这个例子中, 函

数 `humps1` 中的 `y` 只是一个虚的变元，比较简单。

【例 4.10】 求下列微分方程（范德堡方程）的数值解。

$$y''^4 + r(y'^2 - 1)y' + y = 0$$

解：■ 分析

它可写成导数在左端的两个一阶微分方程构成的方程组。

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= r(1 - y_1^2)y_2 - y_1 \end{aligned}$$

先要建立反映此微分方程组右端的函数文件 `vdpl.m`，存入子目录 `user` 中，其内容为：

```
function yprime = vdpl (x,y)
global r % r 值由主程序通过全局变量传送
yprime = [ y(2); r*(1 - y(1).^2).*y(2) - y(1) ]; % 两行单列向量
```

■ MATLAB 主程序

```
global r, r =input('输入 r, 在 0<r<10 之间选择')
x0=input('x0=' ); xf= input('xf=' ); y0=input('y0=[y10;y20]= ');
[x,y] = ode45('vdpl', x0, xf, y0); plot(x, y)
```

在 `r=2, x0=0, xf=30, y0=[1;2]` 条件下得出的曲线如图 4.13 所示。

`ode45` 是高阶的数值积分函数，其步长可以取得较大，并能保证较高的精度。其调用方法与 `ode23` 相仿。这些函数都有自动选择步长的功能，以保证把误差控制在 0.001 以下。如果要改变容许误差 `tol`，可在输入变元中增加其他选项，详情可从 `help` 命令中获得。求微分方程数值解的函数有 `ode23s`, `ode15s`, `ode113` 等数值积分函数，以及绘制相平面曲线的命令 `odephas2` 和 `odephase3` 等。

从本节可见，要利用函数功能，就要学会定义各种复杂函数，表 4.7 列出了 MATLAB 中已定义的某些复杂的特殊函数。

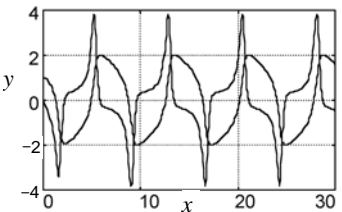


图 4.13 范德堡方程积分的曲线

表 4.7 特殊函数库（specfun）（u）

特殊数学函数	airy	Airy 函数	bessely	第二类 Bessel 函数
	besselj	第一类 Bessel 函数	besselh	第三类 Bessel 函数（Hankel 函数）
	besseli	第一类修正的 Bessel 函数	besselk	第二类修正的 Bessel 函数
	beta	Beta 函数	betainc	不完全的 Beta 函数
	betaln	Beta 函数的对数	ellipj	Jacobi 椭圆函数
	ellipke	完全椭圆积分	erf	误差函数
	erfc	误差补函数	erfcx	标定的误差补函数
	erfinv	逆误差函数	expint	指数整数函数
	gamma	伽马函数	gammainc	不完全的伽马函数
	gammaln	伽马函数的对数	legendre	联合的 Legendre 函数
	cross	向量叉乘		

续表

数论函数	factor	素数分解	primes	产生素数清单
	gcd	最大公约数	lcm	最小公倍数
	rat	有理分式近似	rats	有理分式输出
	isprime	是素数时为真	perms	所有可能的排列数
	nchoosek	N 取 K 的组合数		
坐标变换	cart2sph	从笛卡儿坐标向球坐标变换	cart2pol	从笛卡儿坐标向极坐标变换
	pol2cart	从极坐标向笛卡儿坐标变换	sph2cart	从球坐标向笛卡儿坐标变换

4.5 字符串函数库 (strfun)

MATLAB 的程序和标识符都是用字符串来表示的。每个字符有它对应的 ASCII 码。4.4 节中的函数，都把字符串当做变元来看待，从而使程序更为简化和高效。有时也需要把字符串当做数码来处理。字符串函数库（见表 4.8）中的命令，都是为了增强这一功能。对初学者来说，这部分并不重要，但若编写出人机界面优良，能调用各种函数和文件的高级程序，字符串函数库则必不可少。

表 4.8 字符串函数库 (strfun) (v)

一般函数	char	建立字符数组（字符串）	blanks	空格字符串
	double	把字符串转换为数字	deblank	去除尾部的空格
	cellstr	由字符数组组成字符阵列	eval	执行程序字符串
测试	ischar	是字符串时为真	isletter	是英文字符时为真
	iscellstr	是字符阵列时为真	isspace	是空格字符时为真
字符串比较	strcmp	字符串比较	strncmp	比较前 N 个字符串
	findstr	在字符串中找另一字符串	strjust	调整字符串
	upper	将字符串变为大写	strmatch	找到可能的匹配字符串
	lower	将字符串变为小写	strrep	用一个字符串替代另一个
	strcat	链接字符串	strtok	在字符串中找一个令牌
	strvcat	竖向链接字符串		
字符串与数的转换	num2str	把数转换为字符串	str2mat	由单个字符串形成文本矩阵
	int2str	把整数转换为字符串	sprintf	在格式控制下把数转换为字符串
	str2num	把字符串转换为数	sscanf	在格式控制下把字符串转换为数
	mat2str	把矩阵转换为字符串		
基数的转换	hex2num	把十六进制字符串转换为 IEEE 浮点数	dec2bin	把十进制整数转换为二进制字符串
	hex2dec	把十六进制字符串转换为十进制整数	base2dec	把基数 B 字符串转换为十进制整数
	dec2hex	把十进制整数转换为十六进制字符串	dec2base	把十进制整数转换为基数 B 字符串
	bin2dec	把二进制字符串转换为十进制整数		

4.5.1 字符串的赋值

语句 `s=abyzABYZ0189` 把字符串赋值给 `s`，其结果是

```
s = abyzABYZ0189
而      size(s)= 1 12
```

说明它是以行向量的形式存储的。当然它内部带有字符串的标志，故在屏幕上显示出字符。要找到 `s` 所对应的 ASCII 码，可用 `abs` 命令：

```
abs(s)= 97  98 121 122  65  66  89  90  48  49  56  57
```

从中可以知道英文大小写字母和十进制数字的 ASCII 码值。用 `setstr` 命令可作逆向变换：

```
setstr(abs(s)) = abyzABYZ0189
```

要求出字母和数字的十六进制 ASCII 码值，可用 `dec2hex` 命令：

```
dec2hex(abs(s))= 61 62 79 7A 41 42 59 5A 30 31 38 39
```

MATLAB 显示时并没有各码之间的空格，这里加上空格是为了便于读者阅读。

可以把几个字符串沿行向串接，构成更长的字符串，例如：

```
输入      s1=['welcome ', s]
得        s1 = welcome abyzABYZ0189
```

可以把几个长度相同的字符串沿列向并列，组成一个字符串矩阵，例如：

```
s2 =['a=5      ' ; 'b=2      ' ; 'c=a+b*b' ]
```

这时必须在前两个字符串中增添若干个空格，保证三个字符串长度均为 7，否则赋值无效。

4.5.2 字符串语句的执行

如果字符串的内容是 MATLAB 语句，如上述的 `s2`，则可以用 `eval` 命令来使它执行。例如：

```
输入      for k=1:3 eval(s2(k,:)), end
结果为    a=5, b=2, c=9.
```

在编写 MATLAB 的演示程序时，往往要通过人机交互，让用户输入某种表达式（而不只是数据），然后按此表达式执行，这种程序的格式如下：

```
st = input(' s=表达式 ', ' s '); eval(st)
```

input 语句中的's'表示把输入当做字符串来接受,因此,用户输入的字符串就不必加引号了。下面的例子说明如何将 eval 函数和 load 函数一起使用,读出 10 个具有连续文件名 mydata1, mydata1, ..., mydata10 的数据文件:

```
for i=1:10 fname='mydata'; eval(['load ',fname,int2str(i)]), end
```

特别注意, int2str(i) 把数 1:10 转换为字符 1:10。在显示屏上看不出两者有什么差别,但要记住:字符 0:10 的 ASCII 码是 48:57。数 10 只占一个 MATLAB 双精度存储单元,而字符串 10 却占两个存储单元,并构成一个单行两列的矩阵。在 eval 后的输入变元必须是一个构成 MATLAB 语句的字符串。因此,必须把三个字符串接起来,并且不要忘掉在 load 后面加一个空格。eval 命令是在较高级的程序中常常遇到的,要学会使用。

4.5.3 字符串输入输出

前面一直用 disp 函数来进行字符串和数据的输出。Disp('pi=')将显示引号内的字符串,此处为 pi=。Disp(pi)将显示变量 pi 的值 3.1416 或其他的 8 种显示格式之一,由 format 命令确定。想把字符串 pi=和变量 pi 的值显示在一行上,试用 disp('pi='),结果显示这是非法的。这时应该用 sprintf 函数,它可把数据按要求的格式转换为字符串,再把它与需要显示的字符串组装成一个长字符串,使显示格式非常灵活,人机界面更为友好。

输入 `st=sprintf('圆周率 pi= %8.5f',pi);disp(st)`

结果为: 圆周率 pi= 3.14159

其中%为数据格式符,f 表示十进制浮点数,8.5 表示数字的长度为 8 位,小数点后 5 位。从%到 f 之间的字符都是不显示的,它只规定了显示数据 pi 的格式。

【例 4.11】再举一个用 sprintf 的例子。

```
x = 0:10:90; y = [x; sin(x*pi/180)];disp(sprintf('%10.2f %12.8f\n',y))
0.00    0.00000000
10.00    0.17364818
...
80.00    0.98480775
90.00    1.00000000
```

sprintf 命令是从 C 语言中的同名命令演化来的。sscanf 则是它的逆命令。相仿的还有 fprintf 和 fscanf (如表 3.3 所示)。

4.6 稀疏矩阵函数库 (sparfun)

在许多科学和工程计算中,人们会遇到很大的矩阵,例如几千行几千列,元素则数以百万计。而这些元素中,绝大多数为零。这反映了复杂事物中的诸变量之间,有直接关联的是少数。为了节省内存和提高计算速度,产生了稀疏矩阵的理论和方法。MATLAB 的稀

疏矩阵函数库（如表 4.9 所示）就是为这个目的开发的。

表 4.9 稀疏矩阵函数库（s）

初等稀疏矩阵	speye	稀疏单位矩阵	sprandn	正态分布稀疏随机矩阵
	sprandsym	稀疏对称随机矩阵	spdiags	由对角矩阵生成稀疏矩阵
	sprand	均匀分布稀疏随机矩阵		
全矩阵向稀疏矩阵的转换	sparse	从非零元素创建稀疏矩阵	full	变稀疏矩阵为全矩阵
	find	查找非零元素的下标	spconvert	稀疏矩阵的外部格式转换
用稀疏矩阵的非零元素工作	nnz	非零元素的数目	nonzeros	非零元素
	nzmax	分配给非零元素的内存总额	spones	用 1 替换非零元素
	spalloc	为非零元素分配内存	issparse	矩阵为稀疏时得真
	spfun	对非零元素施加函数	spy	显示稀疏结构
重组算法	colmmd	列的最小度	symmmd	最小对称度
	symrcm	CuthillMcKee 逆排序	colperm	按非零元素数目对列排序
	randperm	随机交换向量	dmperm	DulmageMendelsohn 分解
线性代数	luinc	不完全 lu 分解	svds	一些奇异值
	sprank	结构的秩		
线性方程 (迭代方法)	pcg	预设条件的共轭梯度法	bicg	双共轭梯度法
	bicgstab	双共轭梯度稳定法	cgs	共轭梯度平方法
	gmres	通用最小残差法	qmr	准最小残差法
对树的运算	treelayout	对单树或多树的布局	Treeplot	绘出结构树
	etree	矩阵的消除树	etreeplot	绘出消除树
	gplot	按图论方法画图		
杂项	symbfact	Symbolic 符号因式分解	spparms	为稀疏矩阵设定参数
	spaugment	形成最小二乘增广系统		

稀疏矩阵只存储矩阵的非零元素，其表示形式如下：

```

设      x =      0      0      0      0      0
                0      0      0      0.1302  0
            -0.5936  0      0      0      0
                0      1.4499  0.0388  0      0
            0.5589  0      0      0      0      -0.0954
  
```

● sparse 命令把完全矩阵转换为稀疏矩阵，例如：

```

输入      s=sparse(x)
得      s=(3,1)      -0.5936
          (5,1)      0.5589
          (4,2)      1.4499
          (4,3)      0.0388
  
```

```
(2,4)      0.1302
(5,5)      -0.0954
```

括号内是非零元素的行号和列号，后面则是元素的值。可见 25 个元素中它只须保存 6 个。用命令 `whos` 检验 MATLAB 工作空间中的变量存储情况，结果如下：

Name	Size	Elements	Bytes	Density	Complex
s	5 by 5	6	92	0.2400	No
x	5 by 5	25	200	Full	No

可见 `s` 所占存储单元数为 `x` 所占存储单元数的 0.24 倍。

- `full` 命令把稀疏矩阵转换为完全矩阵。初等矩阵运算的命令，如四则运算，求逆等均可直接用于稀疏矩阵，矩阵分解等命令则要把它化为完全矩阵后才能调用。在大学本科的课程中，解电路矩阵方程也会遇到大量系数为零的情况。但课程习题所遇到的阶次不会高，完全矩阵足以应付。用到稀疏函数库的可能性不大，本书只把函数库列出备查。读者遇到这类应用时，得先参阅有关稀疏矩阵理论的书籍，再用 MATLAB 中的 `help` 文本或参看其他参考书。

4.7 图形界面函数库（`guitools`）

MATLAB 中的图形界面函数库是用来设计像 `demo` 程序中那样的界面。通常，在设计了一个较丰富的 MATLAB 函数集之后，为便于他人使用，应该避免让用户去记忆和输入这些函数的名称。应将其做成这样的图形界面，其中有计算机向用户显示结果或出错信息的各种对话框，有用户对计算机实行控制的各种按钮等。MATLAB 有一本说明书，专门介绍图形界面的设计方法，其搜索路径为：

```
MATLAB\help\pdf_doc\matlab\Buildgui
```

它所用的函数，都已包括在表 4.10 中。读者将来工作中若有需要，可以找此说明书参阅。

表 4.10 图形用户界面工具函数库（`Guitools`）（`x`）

GUI 函数	<code>uicontrol</code>	建立用户控制菜单	<code>dragrect</code>	用鼠标拖引方框
	<code>uimenu</code>	建立用户界面菜单	<code>rbbox</code>	橡皮擦方框
	<code>ginput</code>	用鼠标输入图形	<code>waitfor</code>	执行模块并等待事件发生
	<code>selectmoveresize</code>	交互地选择、移动、变形或复制对象	<code>waitforbuttonpress</code>	等待键或按钮在图上按下
	<code>uiwait</code>	执行模块并等待继续命令	<code>uiresume</code>	继续执行模块 M 文件
GUI 设计 工具	<code>Guide</code>	设计 GUI	<code>Menuedit</code>	编辑菜单
	<code>align</code>	对齐 UI 控制和轴线	<code>propedit</code>	编辑特性
	<code>cbedit</code>	编辑 Callback		

续表

对话框	dialog	建立对话框图形	warndlg	警告对话框
	axlimdlg	对话框轴线范围	uigetfile	标准的打开文件对话框
	errordlg	错误对话框	uiputfile	标准的存储文件对话框
	helpdlg	帮助对话框	uisetcolor	对话框颜色选择
	inputdlg	输入对话框	uisetfont	对话框字体选择
	listdlg	列出待选对话框	pagedlg	对话框页面位置选择
	menu	生成用户输入选择菜单	printdlg	打印对话框
	msgbox	消息框	waitbar	显示等待条
	questdlg	问题对话框		
菜单 设施	makemenu	建立菜单结构	umtoggle	改变用户界面菜单对象的检查状态
	menubar	为不同的计算机设定菜单条默认值	winmenu	为 Window 菜单项建立子菜单
工具条 按钮群 函数	btngroup	建立工具条按钮群	btnstate	工具条按钮群的排队状态
	btnpress	为工具条按钮群按动管理器	btndown	在工具条按钮群中按下按钮
	btnup	在工具条按钮群中抬起按钮		
用户 定义	clruprop	清除用户定义特性	setupprop	设定用户定义特性
	getuprop	获取用户定义特性		
杂项 函数	allchild	获取所有子对象	popupstr	获取弹出菜单选择字符串
	findall	寻找所有对象	remapfig	变换图形对象的位置
	hidegui	隐藏/不隐藏 GUI	setptr	设定图形指针
	edtext	对轴系文本对象作交互的编辑	setstatus	设定图形中的状态文本字符串
	getstatus	获取图形中的状态文本字符串	overobj	获取指针过去后的对象句柄
	getptr	获取图形指针		

4.8 数据类型函数库（datatypes）

由于计算机应用十分广泛，要它处理的数据类型很多，仅以数为例就有整数型（0~65535，16 位）、带符号整数型（-32768~32767，15 位加符号位）、字符型（0~255，8 位）、浮点单精度型（32 位）、浮点双精度型（64 位），等等。其他还有字符类型、指针类型等。在其他语言（例如 C 语言）中，编程时对每一个变量都要作出规定，这就增加了不少语句，而且要好好思考计划，不然会容易出错。

在 MATLAB 中，所有的数都用一种浮点双精度类型来存储和运算。因而省略了定义类型的语句，编程时无须去思考分辨，也减少了错误。当然对于那些本来就只要用一两个字节来表示的变量来说，这种做法既浪费内存，又降低了运算速度。但用牺牲（存储）空间和（运算）时间来换取人机交互友善性的战略被证明是有效的，它形成了科学计算语言的特色，使人们不在编程的细节上花精力，而把注意力集中到科学计算的方法和建模合理

性等大问题上去。

在工程和管理系统中，常常需要分层次地把一些不同类型、不同尺寸和不同分层的数据组织起来，成为一个变量。**MATLAB** 专门为此定义了两种数据类型，一种称为结构阵列，另一种称为单元阵列。这两种数据类型在其他语言中很少见，颇具特色，在此做一简单介绍。

4.8.1 结构阵列

假如要为一个班的学生建立一套管理档案 **student**，记录每个学生的三个项目：姓名（字符串）、出生日期（数字）、四门课（德育，数学，语文，体育）的成绩（数组）。这三项内容的数据类型各不相同，姓名的长度也不同，可以用在 **student** 后加域（field）的方法来建立一个结构阵列。

```
输入值  student.name='John';
        student.birthday='1985.06.15';
        student.score=[85,78,92,68];

再输入  student
得到    ans =   name: 'John'
        birthday: '1985.06.15'
        score: [85.00 78.00 92.00 68.00]
```

也可用 **struct** 命令来建立结构阵列。

```
student(2)=struct('name','Alice','birthday','1986.01.20','score',...
[77,81,65,91]);

再输入  student
得到    student =
        1×2 struct array with fields:
        name
        birthday
        score
```

如果要得到 **student** 各个分量的详细内容，

```
输入    for i=1:2  disp(student(i)), end
```

也可以提取各个域的内容，

```
输入    student.score
得到
ans =      85.00      78.00      92.00      68.00
ans =      77.00      81.00      65.00      91.00
```

结构阵列也可以嵌套。例如若有的分数用优、良、中等级表示，则也必须把课程成绩用结构阵列来表示成为 `student.score.molarity`、`student.score.math`、`student.score.lang`，等等。

4.8.2 单元阵列

单元阵列是用来分层次地组织不同类型数据成为一个变量的另一种方法，与结构阵列的不同在于它不用域名，而是像矩阵那样用下标，与数值或字符串矩阵的区别在于用花括号代替方括号。例如，上述 `student` 也可用单元阵列来存储。

```
输入    name={'John';'Alice'};
        birthday={'1985.06.15';'1986.01.20'};
        score=[85,78,92,68];[77,81,65,91]};
        student={name;birthday;score};
```

```
再输入 student
```

```
得到    student =    {2×1 cell}
           {2×1 cell}
           {2×1 cell}
```

```
输入    student{1}
```

```
得到    ans= 'John'
        'Alice'
```

```
输入    student{1}{2}
```

```
得到    ans='Alice'
```

```
输入    student{3}{2}
```

```
得到    ans =    77    81    65    91
```

```
输入    student{3}{2}(2:3)
```

```
得到    ans =    81    65
```

最后一个是圆括号，因为到这一级是按数组定义的，而前两级都是按单元阵列定义的。

同一个例子，可以用不同数据类型，也可以用不同的分级方式：

(1) `student`-----姓名-----人

---生日

---成绩

(2) `student`-----人-----姓名

---生日

---成绩

其对应的分级构造如图 4.14 所示。

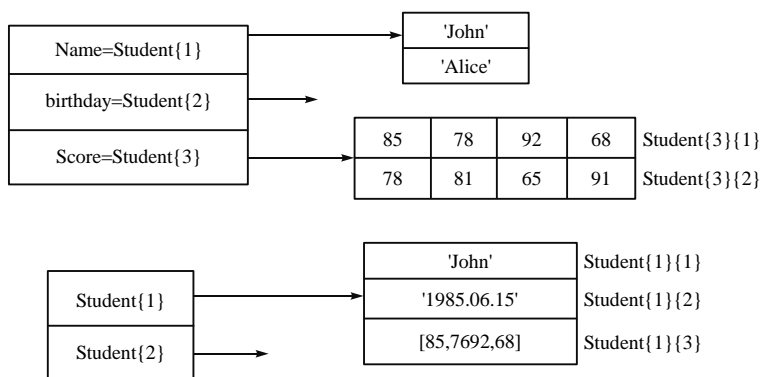


图 4.14 两种不同的数据构造方法

前一种方法在输入新人的时候不大方便。如果要把它改成后一种，即使得：

```
student{1}{1}= 'John'
student{1}{2}= '1985.06.15'
student{1}{3}= 85    78    92    68
```

那就应该按下述语句输入

```
student{1}={'John';'1985.06.15';[85,78,92,68]}
student{2}={'Alice';'1986.01.20';[77,81,65,91]}
```

再输入 `student`

得 `student = {3×1 cell} {3×1 cell}`

即变量 `student` 由两个 3×1 的单元阵列组成，该单元阵列的三列分别为姓名、生日和成绩。

而输入 `student{1}`

得 `ans = 'John'`
`'1985.06.15'`
`[1×4 double]`

由此可见，结构阵列和单元阵列具有相似的功能。其差别在于使用的习惯。结构阵列利用域名来存储和调用各个元素，而单元阵列则是利用下标。前者的好处是具有明确的意义，便于记忆；而后者则比较简洁。

4.8.3 类和对象

所谓“类”就包含了对一种特定的变量结构及其运算方法的定义。所谓“对象”就指一种特定“类”中的某个变量或某个实例。所谓“面向对象的编程”就是描述如何利用“类”和对“对象”进行编程的方法。

MATLAB 的基本部分有五种固有 (built-in) 的类。

- Double——浮点双精度类
- Sparse——稀疏矩阵类
- Char——字符串阵列类 (char, 用双字节存储)
- Struct——结构阵列类
- Cell——单元阵列类

随着应用领域的扩大, 某些 MATLAB 工具箱中也增加了一些其他的数据类型, 例如为图像处理用的 `uint8` 类型 (用单字节存储), 符号推理工具箱提供了 `sym` 类等。结构阵列和单元阵列的应用大大扩展了变量的类型。比如上述的变量 `student` 就可以看做一种新的变量类型。

这些数据类的运算规则和原来为双精度类型设计的 MATLAB 命令和函数不同, 所以它们原则上不能使用 +、-、*、/、\ 等运算符和其他许多函数。MATLAB 采取了扩展运算符的方法, 就是在采用一种特定的数据类型时, 可为该种类型专门定义相应的运算符, 例如用 `plus` 代替 +, `mtimes` 代替 * 等 (参见表 4.10 的最后部分), 这些函数针对不同的数据类型, 因此它们被存储在不同数据类型的子库中。

比如输入 `help plus`

在对 `plus` 函数作出说明以后, 最后显示:

```
Overloaded methods
  help polynom/plus.m
  help sym/plus.m
  help zpk/plus.m
  help tf/plus.m
  help ss/plus.m
```

表示系统中对 `polynom`, `sym`, `zpk`, `tf`, `ss` 这五种数据类型分别由 `plus.m` 文件定义了 `plus` 运算符。其中 `zpk`, `tf`, `ss` 三种数据类型用于控制系统工具箱中, 后面将会介绍。`polynom` 和 `sym` 两种数据类型分别用于多项式和符号运算中。在 4.3 节中已经看到, 多项式的加法不能简单用矩阵加法, 而要单独编一个子程序 `polyadd` 来完成。多项式的乘法也不能简单用矩阵乘法, 而要用 `conv` 来实现, 如果把 `polyadd` 和 `conv` 放在 `polynom` 类的方法库中, 并分别命名为 `plus` 和 `mtimes`, 那么就可以用 + 号和 * 号来对多项式作运算了。MATLAB 在执行程序时, 将先对被运算的变量进行检验, 如果确认其属于 `polynom` 类别, 就会调用方法库中的运算扩展符函数 `polyadd` 和 `conv` 来执行 + 号和 * 号运算符。特定的变量类型加上专门对这种变量类型进行运算的函数, 就构成了一个新的类。有关类和对象的概念, 在入门和基本应用中还不必使用, 本书将在控制系统工具箱中用实例加以说明。

表 4.11 给出了数据类型和结构函数库。

表 4.11 数据类型和结构函数库 (datatypes) (b)

数据类型	double	变换为双精度	sparse	建立稀疏矩阵直接
	char	建立字符 (数组) 串	cell	建立单元阵列
	struct	建立或变换为结构阵列	uint8	变换为无符号 8 位整数
	inline	构成内联 (INLINE) 对象		
多维数组函数	cat	链接数组	ndims	维数
	ndgrid	生成 N 维函数的阵列并插值	permute	重新排列矩阵维数
	ipermute	反向重排矩阵维数	shiftdim	移动维数
	squeeze	去除单项维数 (降维)		
单元阵列函数	celldisp	显示阵列内容	cellplot	显示阵列的图形描述
	num2cell	把数字数组变换为单元阵列	deal	把输入分配给输出
	cell2struct	把单元阵列变换为结构阵列	struct2cell	把结构阵列变换为单元阵列
	iscell	是单元阵列时为真		
结构函数	struct	建立或变换为结构阵列	fieldnames	获取结构域名
	getfield	获取结构域的内容	setfield	设定结构域的内容
	rmfield	去除结构域	isfield	若域在结构阵列中时为真
	isstruct	是结构时为真		
面向对象编程函数	class	建立对象或返回对象类	struct	把对象变换为结构类
	methods	显示类的方法名	isa	若对象是给定类时为真
	isobject	是对象时为真	inferiorto	下级类关系
	superiorto	上级类关系		
可扩展的运算符*	minus	扩展的 a-b	plus	扩展的 a+b
	times	扩展的 a.*b	mtimes	扩展的 a*b
	mldivide	扩展的 a\b	mrdivide	扩展的 a/b
	rddivide	扩展的 a./b	lddivide	扩展的 a.\b
	power	扩展的 a.^b	mpower	扩展的 a^b
	uminus	扩展的 -a	uplus	扩展的 +a
可扩展的运算符*	horzcat	扩展的 [a b]	vertcat	扩展的 [a;b]
	le	扩展的 a<=b	lt	扩展的 a<b
	gt	扩展的 a>b	ge	扩展的 a>=b
	eq	扩展的 a==b	ne	扩展的 a~=b
	not	扩展的 ~a	and	扩展的 a&b
	or	扩展的 a b	subsasgn	扩展的 a(i)=b, a{i}=b 和域=b
	subsref	扩展的 a(i), a{i} 和一个域	colon	扩展的 a:b
	transpose	扩展的 a.'	ctranspose	扩展的 a'
	subsindex	扩展的 x(a)		

4.9 符号数学（Symbolic Math）工具箱简介

顾名思义，符号数学是以符号（如 a, b, c, x, y, z ）为对象的数学，区别于以数字为对象的 MATLAB 基本部分。在大学教育中，符号数学是每门课都用到的，因此，专门以不到 100 美元提供给大学生的版本（Student Edition of MATLAB）中就包括了这个工具箱。国外用 MATLAB 介绍科学计算的教科书中，大概有 15%~20% 的例题和习题会用到这个工具箱，本书在把这个工具箱用到各课程中时，考虑了以下两方面的问题。

（1）在大学教学中，推理是一个基本功。但在大多数的大学课程中，也没有太复杂的推理。一般来说，把数值计算交给计算机做，绝大多数老师还是能接受的，如果把推理也交给计算机，对教学是否有利，可能会有较大争议。作者认为，符号数学工具箱应该教给学生，但大量使用还是放在科研中比较恰当。

（2）Symbolic 工具箱对计算机硬件（包括外存、内存、时钟频率等）的要求比较高，在大部分本科学生的机房内，难以达到。

因为计算机发展很快，第二方面的问题会迅速改变，主要还是考虑第一方面的问题，慎重一些是分两步走，先把大学师生从繁重的数值计算中解放出来，然后再考虑在推理方面做些试验探讨。

4.9.1 Symbolic 工具箱的主要功能

Symbolic 工具箱的主要功能有以下八项：

- （1）用符号定义各种数学运算和函数（syms, symop）等；
- （2）对这些函数式进行代数和三角运算，包括因式分解（factor）、展开（expand）、变量置换（subs）、复合函数（compose）等；
- （3）微分和积分运算（diff, int）等；
- （4）函数的整理和化简（combine, simplify, simple）等；
- （5）可变精度的运算，如可以设置任意多个有效计算位数进行计算（vpa、digits）等；
- （6）解方程，包括单变量的代数方程、多变量非线性的联立代数方程（solve）、单变量微分方程、多变量联立微分方程（dsolve）等；
- （7）线性代数和矩阵运算（determ, linsolve）等；
- （8）变换，包括拉普拉斯变换（laplace）、傅里叶变换（fourier）和 z 变换（ztrans）等。

说明 括号内的英文字符串是 Symbolic 工具箱中的函数名。

4.9.2 符号数学式的基本表示方法

符号数学是对字符串进行运算的，在 MATLAB 中，如果输入

```
f=3*x^2+5*x+2, 或 y=sin(x)
```

系统会指出, 变量 x 无定义, 因为它要求 x 必须是一个数; MATLAB 也可以接受形为 $f=3*x^2+5*x+2$, 或 $y=\sin(x)$ 的语句, 这时 f 和 y 都是一个字符串, 但它没有任何含义。因为它对字符串中的内容不作任何分析。

Symbolic 工具箱必须要能分析字符串的含义, 为此, 首先要对符号变量作出定义, 用语句 “ $x = \text{sym}('x');$ ” 就定义了 x 是一个字符 (串) 变量, 此后输入的算式 $f=3*x^2+5*x+2$, 或 $y=\sin(x)$ 就具有了符号函数的意义, 连表明字符串的引号都可省略, f 和 y 也自然成为字符 (串) 变量。

如果一个数学符号表示式中有多个符号, 如

$$z = a*t^2 + b*t + c$$

可以用简化的多个符号变量定义语句放在此表示式的前面。

`syms a b c t`

注意 只须对算式的右边符号变量 (即自变量) 进行定义, 系统就会自动把因变量定义为符号变量。

在做符号运算时, 比如求这个代数方程的根, 就得知哪个 (些) 是变量, 其余的则是系数, 这时可在语句中指定, 比如 $r = \text{solve}('z = 0', t)$ 表示把 t 作为求解的变量。如果不加说明, 输入 $r = \text{solve}('z = 0')$, 则系统将自动把离 x 最近的那个 (些) 符号当做变量来求解, 在本例中, 自然会选到 t 。

表 4.12 以表格方式列出了一些算例, 读者可对符号数学工具箱的功能有一个大体的印象。

表 4.12 符号推理的语句及其结果的举例

运 算 式	符 号 式	系统响应
符号函数赋值	$r = x^2 + y^2$	$r = x^2 + y^2$
符号函数赋值	$\theta = \text{atan}(y/x)$	$\theta = \text{atan}(y/x)$
符号函数赋值	$e = \exp(i\pi t)$	$e = \exp(i\pi t)$
三角函数式化简	$f = \cos(x)^2 + \sin(x)^2$	$f = \cos(x)^2 + \sin(x)^2$
	$f = \text{simple}(f)$	$f = 1$
微分	$\text{diff}(x^3)$	$\text{ans} = 3*x^2$
积分	$\text{int}(x^3)$	$\text{ans} = 1/4*x^4$
	$\text{int}(\exp(-t^2))$	$\text{ans} = 1/2*\pi^{1/2}*\text{erf}(t)$
矩阵按元素群积分	$[\text{int}(x^a), \text{int}(a^x), \text{int}(x^a, a), \text{int}(a^x, a)]$	$\text{ans} = [x^{a+1}/(a+1), 1/\log(a)*a^x, 1/\log(x)*x^a, a^{x+1}/(x+1)]$
解二次代数方程	$x = \text{solve}('a*x^2 + b*x + c = 0');$ x	$x = [1/2/a*(-b+(b^2-4*a*c)^{1/2})]$ $[1/2/a*(-b-(b^2-4*a*c)^{1/2})]$
解联立代数方程	$[u, v] = \text{solve}('a*u^2 + v^2 = 0', 'u-v=1')$	$u = [1/2/(a+1)*(-2*a+2*(-a)^{1/2})+1]$ $[1/2/(a+1)*(-2*a-2*(-a)^{1/2})+1]$ $v = [1/2/(a+1)*(-2*a+2*(-a)^{1/2})]$ $[1/2/(a+1)*(-2*a-2*(-a)^{1/2})]$

续表

运 算 式	符 号 式	系 统 响 应
以 28 位有效数解联立超越方程	<code>digits(28)</code> <code>[x, y]=solve('sin(x+y)-exp(x)*y = 0', 'x^2-y = 2')</code>	<code>x=-6.017327250059306564109729712</code> <code>y=34.20822723430629650864621443</code>
求一阶微分方程通解	<code>y = dsolve('Dy = -a*y')</code>	<code>y = exp(-a*t)*C1</code> (含任意常数)
给出初始条件求微分方程特解	<code>%y=dsolve('Dy = -a*y', 'y(0) = 1')</code>	<code>y = exp(-a*t)</code>
求二阶微分方程特解， D2 表示二阶导数	<code>% 给出两个初始(边界)条件</code> <code>y= dsolve('D2y = -a^2*y', 'y(0)=1, Dy(pi/a) = 0')</code>	<code>y = cos(a*t)</code>
求二阶微分方程特解	<code>y =dsolve('(Dy)^2 + y^2=1', 'y(0) = 0')</code>	<code>y =[sin(t)]</code> (有两个解) <code>[-sin(t)]</code>
拉普拉斯变换	<code>f=exp(-a*t)*cos(w*t)</code>	<code>f=exp(-a*t)*cos(w*t)</code>
	<code>F=laplace(f)</code>	<code>F=(s+a)/(s+a)^2+w^2)</code>
	<code>pretty(F)</code> 此命令用来改善公式可读性	$\frac{s+a}{(s+a)^2+w^2}$

注：在表中，如符号自变量的定义已经给出，则输入第 2 列的符号式就可得到第 3 列的结果。

表中为了节省篇幅，这里尽量选了一些简单的推导式，实际上可以推导很烦琐的式子。目前 MATLAB 符号数学推理的限度是，表示式的长度不超过 1400 个字符。在一般公式推导意义下使用 MATLAB 是很方便的，只是不给自变量赋以数值，而代之以

```
syms 自变量 1 自变量 2 自变量 3 ...
```

以后的编程和普通 MATLAB 程序完全相同。其执行的结果自然是表达式而不是数值解。如果要做进一步的工作，例如化简、代换、代入数值、解联立方程等，那就需要对这个工具箱有较完整的了解。

不管在课程中是否使用这个工具箱，应该看到计算机科学已经成功地进入了推理领域，已经可以被普通的科技人员在微机上实现。我国数学家吴文俊教授在计算机推理领域做出了国际领先的成果，并因此荣获国家最高科技奖，这也代表了国际科技发展的方向。没有任何理由和方法能够阻止人们去接触、学习、掌握和应用这些推理软件，教师要考虑这一点并探索它对大学教育改革可能产生的影响。教师首先要懂得，要会用。然后应该让学生了解这些软件的功能，便于他们根据自己的需要来选择。

4.10 习题

4.1 求下列联立方程的解

$$3x+4y-7z-12w=4$$
$$5x-7y+4z+2w=-3$$
$$x+8z-5w=9$$
$$-6x+5y-2z+10w=-8$$

$$4.2 \quad \text{设 } A = \begin{bmatrix} 1 & 4 & 8 & 13 \\ -3 & 6 & -5 & -9 \\ 2 & -7 & -12 & -8 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 4 & 3 & -2 \\ 6 & -2 & 3 & -8 \\ -1 & 3 & -9 & 7 \end{bmatrix}$$

求 $C_1 = AB'$, $C_2 = A'B$, $C_3 = A * B$ 并求它们的逆阵。

4.3 a. 列出 2×2 阶的单位矩阵 I , 4×4 阶的魔方矩阵 M 和 4×2 阶的全幺矩阵 A , 全零矩阵 B 。

b. 将这些矩阵拼接为 6×6 阶的矩阵 C :

$$C = \begin{bmatrix} I & A \\ B' & M \end{bmatrix}$$

c. 取出 C 的第 2,4,6 行, 组成 3×6 阶的矩阵 C_1 , 取出第 2,4,6 列, 组成 6×3 阶的矩阵 C_2 。

d. 求 $D = C_1 C_2$ 及 $D_1 = C_2 C_1$ 。

$$4.4 \quad \text{设 } y = \cos x \left[0.5 + \frac{3 \sin x}{(1+x^2)} \right]$$

把 $x=0 \sim 2\pi$ 间分为 101 点, 画出以 x 为横坐标, y 为纵坐标的曲线。

4.5 输入以下程序, 观察得到的矩阵。分析其特点, 并说明如何不用特殊矩阵函数, 而用基本矩阵输入及其组合得到这样的矩阵。

(a) $A = \text{compan}(1:5)$

(b) format rat , $A = \text{hilb}(5)$

(c) $A = \text{hankel}(1:5)$

(d) $A = \text{toeplitz}(1:5)$

(e) $A = \text{vander}(1:5)$

$$4.6 \quad \text{设矩阵 } A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \text{ 问}$$

(a) $B = A^3$ 与 $C = A.^3$ 有何差别?

(b) $B = \exp(A)$ 与 $C = \expm(A)$ 有何差别?

4.7 $v = [1:7]$, $A = v * v'$ 与 $B = v * v'$ 有什么区别? 要将 A 中的奇数行, 奇数列取出来组成新的矩阵 C , 应该用什么语句?

4.8 对于线性方程组

$$x_1 + 2x_2 + 3x_3 = 8$$

$$2x_1 - x_2 + 4x_3 = 7$$

$$3x_1 - x_2 + x_3 = 1$$

(a) 如写成矩阵相乘的形式 $A * x = b$, 则 A, x 和 b 各具何种形式, 如何用矩阵“除法”解出 x ?

(b) 如写成矩阵相乘形式 $x1 * A1 = b1$, 则 $A1, x1$ 和 $b1$ 各具何种形式, 如何用矩阵“除法”解出 $x1$?

4.9 下列语句产生的结果是什么? 它能说明什么问题?

$$\text{syms a b c d; } A = [a, b; c, d], V = \text{inv}(A), \text{pretty}(V)$$

4.10 求代数方程 $3x^5 + 4x^4 + 7x^3 + 2x^2 + 9x + 12 = 0$ 的所有根。

4.11 把 1 开五次方, 并求其全部五个根。(提示: 解 $x^5 - 1 = 0$)

4.12 设方程的根为 $x=[-3,-5,-8,-9]$ ，求它们对应的 x 多项式的系数。

4.13 设微分方程

$$\frac{d^4 y}{dt^4} + 2 \frac{d^3 y}{dt^3} + 5 \frac{d^2 y}{dt^2} + 4 \frac{dy}{dt} + 3 = u$$

求输入 $u(t) = \delta(t)$ 时的输出 $y(t)$ 。

4.14 产生 8×6 阶的正态分布随机数矩阵 R_1 ，求其各列的平均值和均方差。并求全体的平均值和均方差。

4.15 产生 4×6 阶的均匀分布随机数矩阵 R ，要求其元素在 1 到 16 之间取整数值。并求此矩阵前四列组成的方阵的逆阵。

4.16 $x=r \cos t+3t, y=r \sin t+3$ ，分别令 $r=2,3,4$ ，画出参数 $t=0 \sim 10$ 区间生成的 $x \sim y$ 曲线。

4.17 设 $x=\sin t, y=\sin(Nt+\alpha)$ ，

(a) 若 α 为常数，令 $N=1, 2, 3, 4$ ，在 4 个子图中分别画出其曲线。

(b) 若 $N=2$ ，取 $\alpha=0, \pi/3, \pi/2$ 及 π ，在 4 个子图中分别画出其曲线。

4.18 设 $f(x)=x^5-4x^4+3x^2-2x+6$

(a) $x=[-2,8]$ 之间函数的值（取 100 个点），画出曲线，看它有几个过零点。（提示：用 `polyval` 函数）

(b) 用 `roots` 函数求此多项式的根。

4.19 设 $x=z \sin 3z, y=z \cos 3z$ ，要求在 $z=0 \sim 10$ 区间内画出 x, y, z 三维曲线。

4.20 设 $z=x^2 e^{-(x^2+y^2)}$

求定义域 $x=[-2,2], y=[-2,2]$ 内的 z 值（网格取 0.1 见方）。

4.21 设 $z_1=0.05x-0.05y+0.1$ ；画出 z_1 的曲面（平面）图，叠合在上题的图中。

4.22 问题 4.21，求出两曲面的交线，以叉号在图中标出。

4.23 将 4.13 题写成一个函数文件 `f1.m`，用 `fzero` 函数求它的过零点，与 4.13 题的结果做比较讨论：如果在该函数中加一项 $x \sin(x)$ ，过零点怎么求？

4.24 设 $f(x) = \frac{1}{(x-2)^2+0.1} + \frac{1}{(x-3)^4+0.01}$ ，写出一个 MATLAB 函数程序 `f31.m`，使得调用 `f1` 时， x 可用矩阵代入，得出的 $f(x)$ 为同阶矩阵。画出 $x=[0,4]$ 区间内的 `f31` 曲线。

4.25 设 $f(x) = x^3 - 2x^2 \sin x + 5x \cos x + \frac{1}{x}$

(a) 画出它在 $x=[0,4]$ 区间内的曲线。求出它过零点的值。

(b) 求此曲线在 x 轴上方第一块所围的面积的大小。

4.26 已知微分方程： $\frac{dy}{dx} = \frac{x^2}{y} - x \cos y$ ，若 $y(0)=1$ ，求它在 $x=[0,5]$ 区间内的数值积分，并画出曲线。

4.27 用 `eval` 命令执行字符串 `s='y=magic(3)'`。

4.28 如果要用 `for` 循环及 `eval` 语句实现 `yn=magic(n)`, ($n=3,4,5$)，请编出程序。

4.29 用 `sprintf` 命令写出字符串“自然对数底数 $e=2.71828 \cdots$ ”， e 的值应该由 MATLAB 自动生成，其小数点后要显示 20 位。

4.30 用字符串、单元阵列及结构阵列三种方式定义 `student1, student2` 及 `student3` 三个数组，此数组应包括 `Jone, David` 及 `Tom` 三个人名。请比较三者的不同。如果还要包括第二属性——他们的出生地 `birthplace`，分别为 `Shanghai, Nanjing` 和 `Hangzhou`，又有什么差别？

线性代数实践

第二篇

线性代数抽象吗？看了本篇后，你会知道它的概念都基于空间形象。
线性代数繁冗吗？学了本篇后，你会懂得它的计算全可用简明程序。
线性代数枯燥吗？读了本篇后，你会发现它的应用极其广泛而精彩。

本篇内容

- 预备知识
- 用行阶梯法解线性方程
- 用矩阵运算法解线性方程
- 用向量空间解线性方程组
- 线性变换及其特征
- 后续课矩阵建模增补

5.1 实践在线性代数中的重要性

线性代数是一门应用性很强的课程，它表现在几个方面：一是它广泛地应用在科学、工程、经济和管理等各个领域；二是它需要大量使用计算机，离开计算机，任何线性代数的工程实际问题都不可能解决；三是它的概念在三阶以下时具有形象的几何意义，可以用图形来说明，高阶的概念是从低阶引申而来的。《线性代数实践》将在以上三个方面给读者以启示和工具。

如果在做实践之前，读者感觉线性代数很抽象、很难、特别是算题很麻烦的话，学过它的实践课之后，相信读者会有很大的变化。一是知道线性代数在后续课程中有广泛的应用，不抽象；二是知道线性代数方程可以用计算机高效率地解出，甚至比其他课程的解题更容易、更简单；三是线性代数的概念大多数都有几何形象作背景，通过实践可以看出低阶系统的这些形象，便于理解和记忆。总之，如果以前对线性代数有畏惧心理，那么学了实践课之后，读者就不再怕线性代数，会更加乐于使用线性代数模型来解决问题了。

和数学分析不同，线性代数的计算十分烦琐，离开了计算机，解决高于三阶的实际问题的计算量将很难为人们所接受。因此线性代数的教学也必须有在计算机上解题的基本训练。国外在 20 世纪 80 年代以前，一般在线性代数课程末尾要求学生做一个大型作业，在大型机上用 FORTRAN 语言来完成。

从 20 世纪 80 年代开始，个人计算机迅速普及。新的硬件也带动了新的软件，出现了科学计算语言，目前在美国大学的工科教育中，流行最广的是 MATLAB 语言。MATLAB 是 Matrix Laboratory（矩阵实验室）的缩写，可见开发它的初衷就是为了线性代数，这些都可以给线性代数的教学实践带来巨大的推动。使得计算机的使用不限于大作业，也可以融合到每章每节课程教学中去，更好地提高教学的效率和质量。

针对计算工具的这一革命性的变化,美国的线性代数教育指导部门迅速采取了应对措施,1992年美国国家基金会(NSF)资助了一个ATLAST计划,ATLAST是Augment The Teaching of Linear Algebra through the Use of Software Tools(用软件工具增强线性代数教学)的缩写,我们正是根据国际线性代数教学的最新进展和国内个人计算机已相当普及的背景来开设这门实践课的。

利用软件工具进行实践可以提供以下的三个好处:

第一个好处是:对于低阶(三阶及以下)的线性代数问题,MATLAB能提供图形帮助。这对于理解线性代数的理论和概念是很有利的。不要被目前许多线性代数书上 N 阶的概念搞得晕头转向,其实它们几乎都是来自于直观的几何概念,读者从线性代数的许多术语就可以看出这点,如线性空间、向量的张角、正交等。有些概念本来也来自几何概念,如行列式本来来自低阶系统的面积和体积,但为了适应于更普遍的高阶情况,才采用了新的术语。以前有很多教科书就是把线性代数与解析几何放在一起写的,由于线性代数日益重要,而且它的概念已远远超出几何图形,所以现在已很少这样写了。但线性代数的创始者们确实是从几何概念出发,把它引申到代数领域,从而由二、三维引申到高维空间。所以,人们的认识总是从感性到理性、从具体到抽象的。没有平面和空间做基础,是不可能跳跃式地对 N 维空间建立概念的。

我们现在有些线性代数的教材,不谈低阶的概念,而且图很少,这使大学新生很难掌握 N 维空间的特性。因为图形是从几何走向代数的桥梁,国外的线性代数教材,一本书的图都在100幅以上,其中还有不少照片和彩图,值得我们借鉴。但本书篇幅有限,只能补充一部分图。用MATLAB来画图可以省力又准确,不仅可以搭老式的桥,而且还靠新技术提供了动画等演示程序作为新型的“桥”,可以更加深读者的感性认识。所以即使为了课堂演示,也建议线性代数老师们多利用计算机和MATLAB。

第二个好处是:对于高阶的线性代数问题,MATLAB能提供程序帮助,帮助读者快速而准确地进行大量数据的数值计算。线性代数计算是一种刻板、枯燥、但很有规则的低级运算,比如做重复千百次的两数相乘和相加。这种工作不宜由人来做,而最适宜由机器来执行,因为人是很容易出错的。我们培养的高级的人才应该是机器的指挥者,规定了大方向(即程序),让机器按照我们的指挥去完成烦琐的运算,这才符合培养现代化高级人才的需要。有些老师担心用了计算机不利于学生建立概念,其实这是片面的。关键在于正确安排人机分工,把烦杂的工作交给计算机正是为了让人能腾出时间来进行高级的思考,只会有利于深入理解概念。

我国“数学机械化”的带头人,获得首届国家最高科学技术奖的数学家吴文俊院士对“数学机械化”的重要意义作了如下的阐述:“……今天,电子计算机已可以有条件地代替一部分特定的脑力劳动,因而人类面临另一场更宏伟的技术革命,处在又一个新时代的前夕。”他还说:“我国在体力劳动的机械化革命中曾经掉队,以致造成现在的落后状态。在当前新的一场脑力劳动的机械化革命中,我们不能重蹈覆辙。”(见参考文献[13]),数学脑力劳动包括三方面的内容:“数值计算”、“公式推导”与“定理证明”。数值计算的特点是“易、烦、刻板 and 枯燥”,适合计算机实现;因此,它的机械化已经相当成熟,线性代数正好属于这个范畴。此外,本课程中也会用到一些公式推导的概念和功能。

在我国现代化的事业中，面临着激烈的国际竞争。人家是用计算机进行教学、科研和工业生产，如果我们却固守着只把手工计算的方法传给学生，显然不利于人才培养。

第三个好处是：我们在这本实践教材中，较多地放进了线性代数的应用实例，以便低年级的读者了解线性代数在各领域的广泛用途。本来，这部分内容应该在线性代数正式教材中讲的，许多国外教材举出的实例多达数十个，我国的许多教材中实例太少，干巴巴地讲理论，使学生不知道学习目的以及在后续课中有什么用，因而也缺乏主动性。为了弥补这方面的缺陷，在这本实践教材中参照国外的做法适当予以充实。此外，本书第一作者在 50 多年的执教生涯中，曾经先后在机械、自动控制和电子工程系等领域教过十多门课程，而且近十年内写过多本有关 MATLAB 在各门课程中应用的教材和论文，对线性代数的应用有一些特别的心得，愿意尽量把它们介绍给读者，使读者对线性代数的应用有更好的了解，从而激励读者的学习热情，更好地掌握这门课程。

今天，任何一门学科或领域要想得到发展，必须抓住两头，那就是“需求牵引”和“技术推动”，一个在前面拉，一个在后面推，这个学科、这个领域就向前走了，就发展了。如果没有这两头的力量，那必然是停滞的，因为理论是不会自己推动自己的。不过要讲应用部分，涉及的知识面就会广，有些例子会偏深一些，特别是读者的学科领域不同时。但我们还是尽力向读者介绍，偏深的内容不会很多，有些读者可能恰好从较广的知识面得益。有的内容暂时看不懂可以跳过。一本书要有余味可品，内容要略超前一些。总之，我们希望本书在“需求牵引”方面也能做出一点贡献。

5.2 实践部分的内容组成

线性代数要解的基本方程组是

$$Ax=b$$

其完整矩阵形式为：

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

对这三个矩阵 A ， x 和 b 的研究着重点不同，形成了三种方法（视点）：

1. 着重于解 x ，即在 A 和 b 给定的情况下如何求出 x 。这时的注意力放在是研究线性方程的解的存在性和唯一性，探讨在各种 A 时解 x 的基本形式和计算方法。因为是从联立方程的角度来研究的，所以涉及矩阵 A 的处理方法主要是行向，包括行阶梯简化等。同时也从解析几何的角度，考查每个方程的几何意义，把方程组的解看成直线或平面的交点和交线。

2. 着重于系数矩阵 A 的性质和几何意义。此时把矩阵 A 中的各列分别看做向量，把矩阵方程左端看做 n 个向量的线性组合：

$$x_1 \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

即

$$x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \cdots + x_n \mathbf{v}_n = \mathbf{b}$$

系数 $\mathbf{A}=[\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]$ 中各个列向量张成了 r ($r \leq n$) 维空间。 x_1, x_2, \cdots, x_n 被视为各向量的系数。从这里出发研究线性空间、子空间、解空间、正交向量等概念。

3. 将常数数组 \mathbf{b} 看做变量数组 \mathbf{y} , 研究变换 \mathbf{A} 作用于 \mathbf{x} 产生的映射结果 $\mathbf{y}=\mathbf{A}\mathbf{x}$ 。研究线性变换对几何图形产生的效果。探索各种变换的特征以及选择所需变换的方法。虽然变换的几何意义最为形象, 但现在变换的概念已经大大地扩展了。可以把一个向量空间变换为另一个物理意义完全不同的向量空间。比如把工业生产中的物料空间变换为成本空间, 把资料检索的关键词空间转换为图书名空间等。

本书以这三种方法为线索。对第一种方法, 开辟了两章, 分别用行变换和矩阵乘法两种算法进行讨论。对后两种方法, 各用一章, 总计组成了四章。

这是一门实践课, 我们假定读者已经在线性代数理论课程中掌握了系统的理论知识, 在实践课中只需引用现成的结论, 所以在本书中, 一般将不进行定理的证明。实践课的重点, 一是对低阶方程, 进行几何意义的演示和说明; 二是对高阶系统提供计算机的算法和程序, 并通过例题, 提高读者的编程技巧; 三是给读者以线性代数建模和应用的更广的背景。最后这一条是没有边界的, 只能视课程实施的时间状况尽可能提供。在书中给的应用实例范围偏宽偏深一些, 因为各学校在线性代数课程的安排的学期不同 (一年级上学期、一年级下学期、二年级上学期都有), 上本课时微积分的基础差别很大, 而有些例题需要懂微分方程。所以各学校、各专业要选择与自己学科和学生基础相适应的实例。

读者在学习本书的时候, 应该首先着重于对低阶概念的理解, 要在二维和三维空间内体会线性代数的定义和道理, 并且结合相应的 MATLAB 程序, 弄清它的算法和主要函数, 然后再引申到高阶方程或高维空间中去, 进一步搞清在高阶系统中, 算法和程序应有哪些扩展。其实总共大约 10 来个专用函数就足以解决线性代数的基本问题了, 这些专用函数集中在 MATLAB 的矩阵函数子程序库 (matfun) 中。

总之, 这本书是为了给我国现有的线性代数教材补缺的, 通俗地说, 是“打补丁”。这样做, 重点突出, 避免理论阐述上重复。在习题上也避免重复, 本书的习题一般都是适合于计算机算的题目, 特别是应用性比较强的习题, 以便读者更好地了解线性代数在解决工程实际问题中是如何发挥作用的。

5.3 直线和平面的快速绘制程序

为了做线性代数题目的方便, 我们在讲解实践原理之前, 先介绍一些专门为线性代数教学而编写的子程序。这里要介绍的内容一是直线和平面的快速绘制程序, 二是随机矩阵产生程序。

1. 直线的快速绘制程序

MATLAB 中的 `ezplot` 可以绘制很多函数的曲线, 在它的第一输入变元中可以直接输入用 MATLAB 语句写出函数的形式, 不过要用单引号括起来。第二输入变元为自变量的取值范围, 在默认情况下其取值范围为 $[-2\pi, 2\pi]$ 。

引号中的函数可以有一个自变量或两个自变量，只有一个自变量就代表显函数，其典型格式为：

```
ezplot('f(x)', [a,b])
```

系统将在 $a < x < b$ 的范围内画出 $f=f(x)$ 。

输入 `ezplot('sin(x)*cos(x)')`，就绘制出 $y=\sin(x)*\cos(x)$ 在 $-2\pi \leq x \leq 2\pi$ 范围内的曲线。

如果引号中的函数有两个自变量，那就代表隐函数，其典型格式为

```
ezplot('f(x,y)', [a,b])
```

其含义是，在 $a < x < b$ 的范围内画出 $f(x,y)=0$ 。

【例 5.1】 在一张图中绘制两根曲线： $x_1 + 0.2x_2^3 + 1 = 0$ 和 $3x_1 + 2x_2 + 3 = 0$ 。

输入 `ezplot('x1+0.2*x2^3+1')` 就绘制出 $x_1 + 0.2x_2^3 + 1 = 0$ 在 $-2\pi \leq x \leq 2\pi$ 范围内的曲线。如果要在同一张图上画出两根曲线，在画完第一条曲线后，必须输入 `hold on`，再输入第二条曲线的 `ezplot` 命令。例如再要在上图中画直线 $3x_1 + 2x_2 + 3 = 0$ ，就应输入 `ezplot('3*x1+2*x2+3')`。两根曲线如图 5.1 所示，它们有三个交点，这些交点可以用 `solve` 命令求得，其格式为：

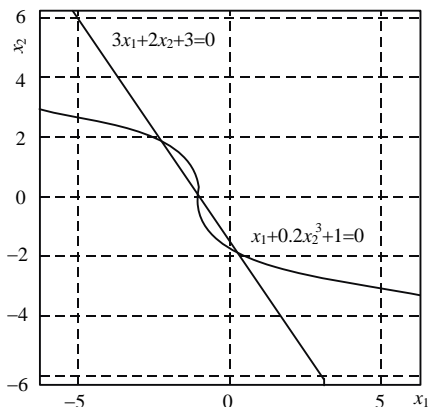


图 5.1 用 `ezplot` 在同一图中画两根曲线

```
[x1,x2]=solve('3*x1+2*x2+3','x1+0.2*x2^3+1')
```

为了避免多次输入函数表达式，也可以按以下的方法输入程序 ag501：

```
s1='x1+0.2*x2^3+1'
s2='3*x1+2*x2+3'
ezplot(s1),hold on
ezplot(s2),grid on
[x1,x2]=solve(s1,s2)
```

解得三个交点的坐标为

$$x1 = \begin{bmatrix} -1. \\ -2.2171612389003691410154884062240 \\ .21716123890036914101548840622401 \end{bmatrix}$$

$$x2 = \begin{bmatrix} 0 \\ 1.8257418583505537115232326093360 \\ -1.8257418583505537115232326093360 \end{bmatrix}$$

得出的曲线如图 5.1 所示。

在线性代数中，遇到的都是一次函数，所以不会出现曲线。我们故意用一个三次曲线来说明 `ezplot` 的用法，是为了使读者知道，这个命令不限于画直线。另外 MATLAB 用 `solve` 命令解题是采用了符号运算工具箱，它的数字精度是 32 位十进制数，而不是一般数值计算时的 16 位十进制数。尽管在本题中有两个有效数字就够了。

2. 平面的快速绘制程序

MATLAB 中的 `ezmesh`（或 `ezsurf`）函数可以绘制函数的曲面。在它的第一输入变元中可以直接输入用 MATLAB 语句写出函数的形式，注意要用单引号括起来。第二输入变元为自变量的取值范围，在默认情况下它在 x, y 两个方向的取值范围都是 $[-2\pi, 2\pi]$ 。

与 `ezplot` 函数不同，引号中的函数只能是显函数 $z=f(x,y)$ 中的 $f(x,y)$ 。它应该有两个自变量，其典型格式为：

```
ezmesh('f(x,y)', [a,b,c,d])
```

系统就会在 $a < x < b$ 和 $c < y < d$ 的范围内画出 $z = f(x,y)$ 。下面我们只介绍线性代数中用到的三维空间中平面的画法。

【例 5.2】 在一张图上画出两个三维空间的方程所表示的平面，两个方程为：

$$z_1 = 3x_1 + 2x_2 + 3$$

和

$$z_2 = x_1 - 2x_2 + 1,$$

输入：`ezmesh('3*x1+2*x2+3')`

系统就绘制出 $z = 3x_1 + 2x_2 + 3$ 在 $-2\pi \leq x_1 \leq 2\pi$,

$-2\pi \leq x_2 \leq 2\pi$ 范围内的平面。

要在同一张图上画出第二个平面，在画完第一个平面后，必须输入 `hold on`，再输入第二个平面的 `ezmesh` 命令。例如

```
hold on
ezmesh('x1-2*x2+1')
```

这时所得到的图形如图 5.2 所示。如果需要，也可以用同样的道理画出第三个平面。

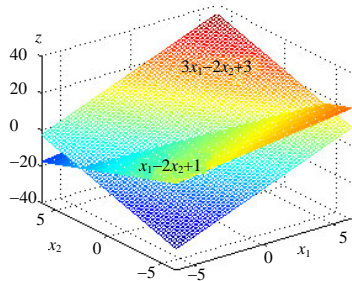


图 5.2 用 `ezmesh` 函数绘制两个平面

3. 用隐函数的平面绘制程序

用 `ezmesh` 画平面必须要解出 $z=f(x,y)$ 的显函数形式, 有时就觉得不太方便, 容易发生移项正负号或系数除法的错误。如果程序能够像 `ezplot` 那样, 接受三维空间的隐函数平面方程, 那就更为方便了。为此, 我们开发了 `ezplot3` 程序。连带也开发了在一张图上画两个平面的 `ezplot2` 程序和画三个平面的 `ezplot3` 程序。

【例 5.3】 绘制由下列三个方程表示的三个平面。

$$x+y-z=5, \quad 2x-3y+z=3 \text{ 和 } -5x+2y-2z=0$$

解: 输入 `ezplot3`, 计算机将要求用户输入所画平面的隐函数方程, 依次输入:

$$x+y-z=5$$

$$2*x-3*y+z=3$$

和

$$-5*x+2*y-2*z=0$$

后, 可得到第 120 页的图 6.3 所示的图形。在此绘图程序中, 还提供了平面交线的绘制语句, 所以我们可以清晰地看出平面的交线。

4. 用符号工具箱解线性方程组

MATLAB 的符号运算工具箱还提供解线性方程组的能力。例如画出上面的图, 要标注交点的坐标, 就可用:

$$[x,y,z]=\text{solve}('x+y-z=5','2*x-3*y+z=3','-5*x+2*y-2*z=0')$$

解得的结果为:

$$x=10/7, \quad y=-13/7, \quad z=-38/7$$

虽然在本课正式习题中不允许使用这种傻瓜式的解题方法, 因为它不能帮助我们理解概念。但仍然应该让读者知道有这种方法。当需要快速地得到结果, 或者要比较不同方法的解答, 不需要钻研概念时, 就可以调用这个函数。

5.4 随机整数矩阵的生成程序

在线性代数的教学中, 往往需要产生具有给定阶数且系数简单随机生成的矩阵, 通常采用随机函数来产生。`rand(n,m)` 函数生成的是 $n \times m$ 矩阵, 其元素的值在 0 与 1 之间按均匀概率密度分布。如果我们希望产生的元素都取个位数, 则可以先把这些元素都乘以 9 再按四舍五入取整。例如输入

$$A=\text{round}(9*\text{rand}(2,4))$$

得到

$$A = \begin{bmatrix} 7 & 3 & 8 & 2 \\ 4 & 3 & 6 & 9 \end{bmatrix}$$

如果要得到系数为在 -9~9 之间的随机整数, 则可输入

$$A=\text{round}(19*(\text{rand}(2,4)-0.5))$$

得到

$$A = \begin{bmatrix} 3 & 3 & -7 & -5 \\ -5 & 3 & -9 & -7 \end{bmatrix}$$

这些结果不是确定的，读者各人实践时会得到不同的结果。

从线性代数的需要来看，在设计系数矩阵时很重要的一点是要求生成的矩阵具有规定的秩，这种矩阵生成的程序是由 ATLAST 项目中编出的，它的程序名是 `randintr.m`。调用的方法是

```
A=randintr(m,n,q,r)
```

其中 m 是矩阵的行数， n 是矩阵的列数， q 是整数系数的最大模，而 r 则是此矩阵的秩。显然输入变元时必须保证 $r \leq \min(n, m)$ ，否则系统将会显示出错警告。例如输入：

```
A=randintr(3,5,9,2)
```

得到

$$A = \begin{bmatrix} -4 & 1 & 3 & 4 & -5 \\ 2 & 1 & -3 & -2 & 1 \\ -3 & 0 & 3 & 3 & -3 \end{bmatrix},$$

可检验其秩， $\text{rank}(A)=2$ 。

`randintr` 子程序的编程的基本原理是：生成一个 $m \times r$ 和一个 $r \times n$ 随机矩阵，再把两者相乘。如果 $r \leq \min(m, n)$ ，而这两个矩阵都是满秩 r ，那么乘积矩阵的秩也是 r 。

5.5 特殊矩阵的生成程序

除了最常用的全零 `zeros(m,n)`、全幺 `ones(m,n)`、单位矩阵 `eye(n)` 等矩阵外，MATLAB 还提供了一些特殊矩阵的生成程序，见本书表 2.1 中的“特殊矩阵”栏，其中列出了 12 个特殊矩阵的生成函数。在本篇中，我们会用到魔方矩阵 `magic(n)` 和范德蒙特矩阵 `vander([0:n])`。另外，在 ATLAST 手册中也提供了 19 个特殊矩阵的生成程序（如本书附录 A 所示）；在本书的子程序集中，也提供了几个初等变换矩阵的生成程序 `E1gen(A,i,j)`、`E2gen(A,i,k)` 和 `E3(A,i,j,k)` 等，读者要注意运用它们及它们的组合，以节省输入矩阵参数的时间。

5.6 线性代数建模与应用概述

在序言中，介绍了 Wassily Leontief 教授把美国的经济用 500 个变量的 500 线性方程来描述的例子，他于 1973 年获得诺贝尔经济奖，从而大大推动了线性代数的发展。本书将在第 8 章中介绍这个模型及其解的基本概念，当然问题是大大简化了的，只用三个方程和三个变量，不过，它的概念照样能够说清楚。科学从来不是以烦琐为特征的，线性代数这门科学也是这样。高明的老师总是能把复杂问题变得简明的。工程问题则是很烦琐的，所以工程师就喜欢要计算机辅助，让计算机去处理大量的简单重复的工作，工程师本人就可以

集中精力处理重要的关键性决策。

以现代飞行器外形设计为例，它决定了整个飞行器的空气动力学特性，因而地位十分重要。现在流体动力学的理论和计算流体动力学软件已经很成熟，问题是怎样用到特定的外形上来。人们采用的方法就是把飞行器的外形分成若干个大的部件，每个部件又沿着其表面用三维的细网格划分出许多立方体，这些立方体包括了机身表面以及此表面内外的空气。对每个立方体列出空气动力学方程，其中包括了与它相邻的立方体的共同边界变量，这些方程通常都已经简化为线性方程。对一个飞行器，小立方体的数目可以多达 400 000 个，而要解的联立方程可能多达 2 000 000 个。即使是现代最大最快的计算机，直接解这样大的方程组仍然是不现实的，所以要简化。主要的简化手段有两个，一个是利用许多不相邻的元素之间没有关联，其交叉系数为零，在这个大联立方程中，绝大部分的系数为零，所以诞生了稀疏矩阵的计算方法；另一个则是把矩阵进行分解，特别是采用第 7 章介绍的 lu 方法分解为三角矩阵，可以大大提高计算的速度。工程问题就是这样向矩阵理论提出需求，从而推动理论发展的。它决不是由哪些超人，从定义出发冥思苦想出新的理论来的。

飞行器的控制更是一个复杂的线性空间的问题。飞行器的运动要用三个转动和三个平移共六个变量来表示（像在第 9 章中介绍的那样）。除此以外，为了控制飞行器的三维转动，需要三个控制面，即方向舵（垂直尾翼）、升降舵（水平尾翼）和副翼，它们的偏转角又构成了三个变量。这些控制面的偏转角与飞行器的运动变量是用刚体运动方程联系起来的，通常它们直接影响的是飞行器的运动变量的（角）加速度，所以它们之间是二阶微分方程关系，飞行器的（角）速度必然出现在方程组中。仅这样粗糙地分析，描述飞行器的运动就需要 12 个变量的组合。而且这已经不单是代数方程，而是微分方程了。不过我们知道，解线性微分方程是以解同阶同型的线性代数方程为基础的。

再以卫星遥感图像处理为例。气象和地球资源卫星约用 90 分钟绕地球一圈，摄出的图像宽度约为 150 公里。地球赤道长 40 000 公里，因此大约每 16 天，它可以扫描地球的每一个角落一遍。摄像可以用多种光谱，例如可用三种可见光和四种红外光，这样，对每一个区域，可以获得七张遥感图像。利用多通道的遥感图像可以获取尽可能多的地面信息，因为各种地貌、作物和气象特征可能对不同波段的光敏感。而在实用上应该寻找每一个地方的主因素，再把它们合成起来，成为一张实用的图像。每一个像素上有七个数据，形成一个多元的变量数组，在其中合成并求取主因素的问题，就与线性代数中要讨论的特征值问题有关。

再来看一个大工程——国家地理信息系统。从概念上说，它就是在全国设立几十万个观察点，把每一点的经度、纬度和高度三个坐标建立起来。由于地壳的变动，测量仪器的现代化和实际需求的增长，这个地理信息系统精细化的工程是永无止境的。取点的密度一次比一次增加，精度要求一次比一次提高。例如现在对于经度纬度的测量精度要求，已经提高到了若干厘米，对于高度的精度要求更高。在一些边远地区，对于一些特征点的测量，要耗费很大的人力物力。例如对珠穆朗玛峰顶高度的测量，要经过多种方法，取得多种数据，并且用最小二乘法进行误差的处理。所以要获得全部数据一般就要好几年，需要解上百万个线性方程，所以只能几十年做一次全面更新。

这里举的都是一些全国性的最先进的大题目，对于一个企业管理，一个具体工程设计等问题，我们将在各章的应用实例中做具体的描述和求解。

5.7 习题

5.1 用 `ezplot` 命令画出以下各组方程所对应的直线，并求其交点。

$$(a) \begin{cases} 4x + y = 7 \\ 3x - 2y = -3 \end{cases}$$

$$(b) \begin{cases} x + y = 7 \\ -8x - 2y = -4 \end{cases}$$

$$(c) \begin{cases} x - 2y = -3 \\ -6x + 4y = 6 \end{cases}$$

$$(d) \begin{cases} x + y = 7 \\ x = 2 \end{cases}$$

$$(e) \begin{cases} x + y = 7 \\ 3x - 2y = -3 \\ x + 3y = -1 \end{cases}$$

$$(f) \begin{cases} x + y = 7 \\ 3x - 2y = -3 \\ x + 3y = 10 \end{cases}$$

5.2 用 `ezplot2` 或 `ezplot3` 命令画出以下各组平面的形状。

$$(a) \begin{cases} x + 2y - z = -3 \\ 2x - y + z = 5 \end{cases}$$

$$(b) \begin{cases} x + y - z = 0 \\ x - y - z = -1 \end{cases}$$

$$(c) \begin{cases} x + y - z = 1 \\ x - y - z = 5 \\ -x - y + z = -9 \end{cases}$$

$$(d) \begin{cases} x + y + z = 1 \\ x - y - z = 1 \\ 5x - y - z = 5 \end{cases}$$

$$(e) \begin{cases} x + y - z = -1 \\ x - y - z = 1 \\ 5x - y - z = 1 \end{cases}$$

$$(f) \begin{cases} 0.1x + 0.2y - z = 0 \\ 0.5x + 0.1y - z = 0 \\ -0.3x + 0.3y - z = -2 \end{cases}$$

5.3 魔方矩阵 $A = \text{magic}(n)$ 由 1 到 n^2 的 n^2 个顺序正整数组成，它的各行元素之和、各列元素之和以及两个主对角线元素之和都相等，其值为 $n(n^2+1)/2$ 。如果我们任意设定其和，而使几个元素待定，例如要

找到 $A = \begin{pmatrix} 8 & 1 & 3 \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$ 中的 $a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}$ ，使其各行、各列和对角的元素和都等于 12，则

可以列出 7 个方程。

(a) 试列写此方程组。

(b) 试用这 7 个方程解 6 个未知数。我们知道，解是存在和唯一的。但是它不符合通常意义下的魔方矩阵，因为它们不大可能是顺序排列的正整数。

- (c) 试填入下列矩阵的元素,使其各行、各列和对角的元素和都等于 12,不要求它们是顺序排列的正整数,只要求它们是整数。

$$\textcircled{1} \begin{bmatrix} 1 & ? & 3 \\ ? & 2 & ? \\ ? & ? & ? \end{bmatrix}, \textcircled{2} \begin{bmatrix} 1 & ? & 3 \\ 2 & ? & ? \\ ? & ? & ? \end{bmatrix}, \textcircled{3} \begin{bmatrix} 1 & ? & ? \\ ? & 3 & 2 \\ ? & ? & ? \end{bmatrix},$$

5.4 随机整数矩阵的生成。

- (a) 产生 5×6 的随机正整数矩阵,使元素的值在 $0 \sim 20$ 之间。
 (b) 产生 3×6 的随机整数矩阵,使元素的值在 $-9 \sim 9$ 之间。
 (c) 产生 4×5 的随机整数矩阵,使元素的值在 $1 \sim 9$ 之间。

5.5 规定“秩”数的随机整数矩阵的生成。用 $A=\text{randintr}(I,j,k,r)$ 函数生成,并要求用 $R=\text{rank}(A)$ 进行检验。

- (a) 产生 5×6 的随机整数矩阵,秩为 4,使元素的值在 $-20 \sim 20$ 之间。
 (b) 产生 3×3 的随机整数矩阵,秩为 2,使元素的值在 $-99 \sim 99$ 之间。
 (c) 产生 4×5 的随机整数矩阵,秩为 3,使元素的值在 $-9 \sim 9$ 之间。
 (d) 产生 5×8 的随机整数矩阵,秩为 4,使元素的值在 $0 \sim 100$ 之间。

5.6 设 $A=\text{magic}(6)$ 。

- (a) 用 $s1=\text{sum}(A)$ 求其各列元素的和。用 $s2=\text{sum}(\text{diag}(A))$ 求其主对角线上元素的和。
 (b) 问如何求其各行元素的和? 如何求其反主对角线上元素的和?
 (c) 用 $A=\text{magic}(7)$,证实各行各列元素的和为 $n(n^2+1)/2$ 。

5.7 (a) 设 $A=\text{vander}([1,3,5])$,观察 A 的特点,予以说明。

- (b) 设 $A=\text{vander}([0:5])$,观察 A 的特点,予以说明。
 (c) 设 $B=A', C=\text{rot90}(A)$,观察 B 和 C 的特点,予以说明。

5.8 分别给出 $n=5$ 和 $n=6$,输入下列命令,以便对 ATLAST 所提供的部分特殊方阵矩阵有所了解,并说明其意义:

- (a) 网格矩阵 $A=\text{gridmat}(n)$
 (b) 最大下标矩阵 $A=\text{maxmat}(n)$
 (c) 符号矩阵 $A=\text{signmat}(n)$
 (d) 反向单位矩阵 $A=\text{backiden}(n)$,它也可用 $\text{fliplr}(\text{eye}(n))$ 生成。
 (e) 顺序整数矩阵 $A=\text{consec}(n)$,它也可用 $\text{reshape}([1:n^2],n,n)'$ 生成。
 (f) 字符 N 矩阵 $A=\text{Nmatrix}(n)$
 (g) 字符 H 矩阵 $A=\text{Hmatrix}(n)$

5.9 本题的目的在于让读者知道,用 randintr 函数生成的随机矩阵的非奇异概率有多大。用 MATLAB 程序来检验是非常方便的。

- (a) 以下取最大数 k 为可变参数,令 $k=1:20$,对每个 k ,生成 100 个 2×2 方阵,测试它们中的奇异矩阵的数目,列写成表。

```
persent=zeros(1,20);
for k=1:20
    for i=1:100
        if det(randintr(2,2,k))==0
```


用行阶梯法解线性方程

6.1 线性方程组的 MATLAB 表示方法

由 n 个变量组成的 m 个联立线性代数方程组：

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots \quad \quad \quad \cdots \quad \quad \quad \cdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \quad (6.1)$$

可以用 MATLAB 语言表示为：

$$\mathbf{A} * \mathbf{x} = \mathbf{b} \quad (6.2)$$

其中：

$$\mathbf{A} = \begin{bmatrix} a(1,1) & a(1,2) & \cdots & a(1,n) \\ a(2,1) & a(2,2) & \cdots & a(2,n) \\ \cdots & \cdots & \cdots & \cdots \\ a(m,1) & a(m,2) & \cdots & a(m,n) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x(1) \\ x(2) \\ \cdots \\ x(n) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b(1) \\ b(2) \\ \vdots \\ b(m) \end{bmatrix} \quad (6.3)$$

\mathbf{A} 为 m 行 n 列的 $m \times n$ 阶系数矩阵， \mathbf{x} 为单列 n 元变量数组（或向量，下同）， \mathbf{b} 为单列 m 元系数数组。 $*$ 为乘法运算符，它同时适用于矩阵和标量。其中，矩阵乘法按照定义应该是左边矩阵的第 i 行的 n 个元素，与右边矩阵的第 j 列的 m 个元素一一对应地相乘，再叠加起来。只有按照这个规则，才能构成下列与原方程等价的向量方程：

$$\begin{bmatrix} a(1,1)*x(1)+a(1,2)*x(2)+\cdots+a(1,n)*x(n) \\ a(2,1)*x(1)+a(2,2)*x(2)+\cdots+a(2,n)*x(n) \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ a(m,1)*x(1)+a(m,2)*x(2)+\cdots+a(m,n)*x(n) \end{bmatrix} = \begin{bmatrix} b(1) \\ b(2) \\ \cdots \\ b(m) \end{bmatrix} \quad (6.4)$$

可以看出, 变量后圆括号中的第一个数是行号, 第二个数是列号, 不加下标的变量 \mathbf{A} 自身代表一个矩阵。除了把下标放入圆括号中之外, MATLAB 的表述与原来的算式是完全一致的, 非常好记。不过在 MATLAB 中变量名和元素名是一致的, 式 (6.3) 中的矩阵 \mathbf{A} 应该是 a 。但是为了使叙述文与一般的矩阵书籍相一致, 我们还是把矩阵用大写字母表示, 数组和元素则用小写字母, 希望读者注意。

如果 n 是未知数的数目, m 是独立方程的数目, 那么当 $n > m$ 时, 未知数比独立方程数目多, 此方程组有无数个解, 称为欠定方程; 当 $n < m$ 时, 未知数比独立方程数目少, 此方程组无解, 称为超定方程; 只有当 $n = m$ 时, 未知数与独立方程数目相等, 因而有唯一的解 x , 称为适定方程。所以, 不能简单地看形式上的 m 和 n , 还必须要剔除其中非独立方程的虚假成分。本章和第 7 章中将讨论的行阶梯形式、行列式和“秩”等概念, 很大程度上就是为了找到独立方程的数目。

【例 6.1】 求解下列四个线性方程组

$$\begin{array}{ll} \text{(a)} \quad \begin{cases} x_1 - 2x_2 = -1 \\ -x_1 + 3x_2 = 3 \end{cases} & \text{(b)} \quad \begin{cases} x_1 - 2x_2 = -1 \\ -x_1 + 2x_2 = 3 \end{cases} \\ \text{(c)} \quad \begin{cases} x_1 - 2x_2 = -1 \\ -x_1 + 2x_2 = 1 \end{cases} & \text{(d)} \quad \begin{cases} x_1 + x_2 = 1 \\ x_1 - x_2 = 3 \\ -x_1 + 2x_2 = -3 \end{cases} \end{array}$$

解: 前三题都是两个二元一次联立方程, 容易用消元法求解, 并可用 `ezplot` 命令画图。

(a) $x_2=2, x_1=3$, 在几何上它是两根直线的交点, 如图 6.1 (a) 所示;

(b) 这两个方程是不相容的, 或矛盾的, 方程组无解。可以在平面上画出代表两个方程的两根直线, 它们保持平行又间隔一定距离, 因此没有交点, 如图 6.1 (b) 所示;

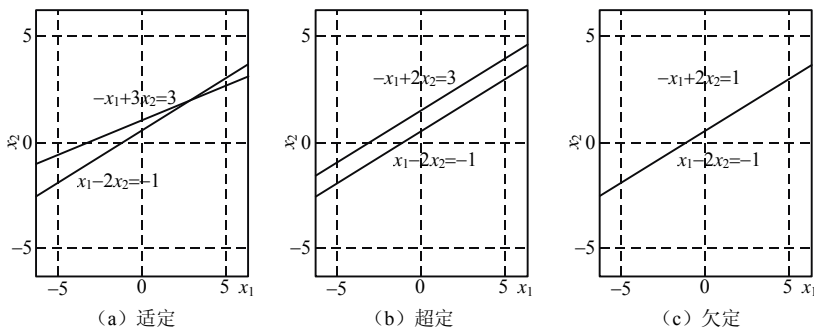


图 6.1 例 6.1 两个二元线性方程解的三种情况

(c) 这两个方程是相依的, 满足第一个方程的解必然也满足第二个方程。两个方程相当于一个方程, 未知数却有两个, 是欠定方程。换言之, 任何满足 $x_1=2x_2-1 (x_2 \in (-\infty, \infty))$ 的 (x_1, x_2) 都是解, 即有无穷组解。令 x_2 为自由变量, 设它为任意常数 α , 则此方程的解可以表为:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \alpha \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

在几何上这两根直线重合, 因此在此直线上, 处处都是交点, 如图 6.1 (c) 所示;

(d) 此题有两个未知数 $n=2$, 三个方程 $m=3$, $n < m$, 属于超定方程, 一般是不相容和无解的, 几何上, 平面上的三根直线很难交于一点, 像这三根直线就不交于一点, 即方程组无解, 如图 6.2 所示。

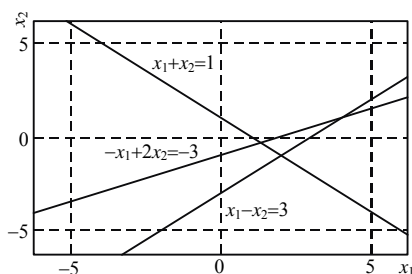


图 6.2 例 6.1 (d) 的解 (二元一次超定方程情况)

上述的基本概念完全可以推广到三元一次方程组, 几何上只是直线换成了平面, 交点变成了交线。

【例 6.2】 求解下列线性方程组

$$\begin{array}{ll} x + y - z = 5 & (1) \\ 2x - 3y + z = 3 & (2) \\ -5x + 2y - 2z = 0 & (3) \end{array} \longrightarrow \begin{array}{l} \text{(由式(1), 式(2)消去 } z) \quad 3x - 2y = 8 \\ \text{(由式(2), 式(3)消去 } z) \quad -x - 4y = 6 \end{array}$$

两式联立解得:

$$x = 1.4286,$$

$$y = -1.8571$$

再代入上述的任一平面方程, 得 $z = -5.4286$

这三个方程的几何意义为空间的三个平面, 两两联立, 意味着求两者的交线, 得到的两个二元一次方程, 对应于求得的两根交线在 xy 平面上的投影。最后由这两根交线得到交点, 该交点就是方程的解 (如图 6.3 所示)。当两个平面平行, 没有交线, 或者两根交线平行, 没有交点时, 方程组就不相容, 因而无解; 同样也可能有无穷个解的情况, 读者可自行思考。

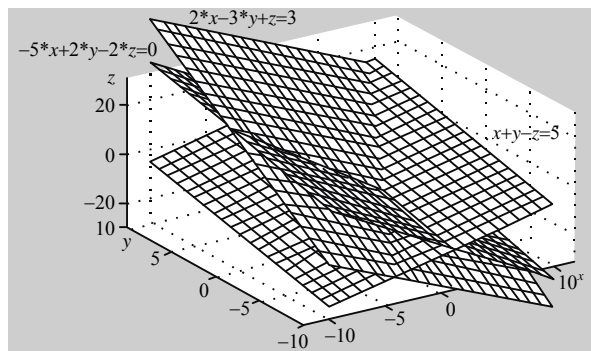


图 6.3 三个三元一次联立方程组的几何图形 (三个平面、交线和交点)

对于更多元的线性代数方程组,也就是在更高维的空间内的方程组,虽然不可能想象出四维空间的图形,但关于欠定、适定和超定方程的基本概念是一脉相承的,关于它们的解的特性也都可以推广到高维空间去。而涉及它们的具体计算,就必须从几何概念过渡到代数概念。这里还要说明一下关于“元、阶和维”的用法。讲到变量数目时,用“元”,讲到方程的数目时,用“阶”,讲到向量空间的大小时用“维”,在我们笼统地说到“多元”、“高阶”或“高维”问题时,它们是一致的。

6.2 初等行变换和高斯消元子程序

将线性方程组的系数矩阵 A 和 B 组成增广矩阵

$$C = [A, B] = \begin{bmatrix} a(1,1) & a(1,2) & \cdots & a(1,n) & b(1) \\ a(2,1) & a(2,2) & \cdots & a(2,n) & b(2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a(m,1) & a(m,2) & \cdots & a(m,n) & b(m) \end{bmatrix} \quad (6.5)$$

增广矩阵的各行各代表一个线性方程左右两端系数。执行以下三种行运算不会影响方程的相等关系,故不会改变方程组的解,我们称它们为矩阵的行初等变换。经行初等变换后的方程组与原方程组有相同的解。同时,我们也把一个矩阵与初等变换后的矩阵称为互为等价的,下面来看它们所对应的 MATLAB 描述:

(1) 行交换: 将增广矩阵的第 i, j 两行位置互换,即相当于把 i, j 两个方程交换位置。显然这不会影响方程的解。这个行运算用 MATLAB 语句可写成:

$$c([i,j],:)=c([j,i],:) \quad (6.6)$$

其中在列下标位置的冒号表示所有各列。要注意,程序中的等式表示的是一个计算机赋值的过程,等式右边的 $c([j,i],:)$ 是赋值前的行向量,等式左边的 $c([i,j],:)$ 则是赋值后的行向量,形式上相同的两项生成的时间不同,因此内容不同,不能看作相等。程序的“赋值”关系不能理解为“相等”的关系。

(2) 行乘数: 将增广矩阵的第 i 行乘以常数 k ,即相当于把第 i 个方程两端乘以常数 k ,这当然也不影响方程的解。用 MATLAB 语句可写成:

$$c(i,:)=k*c(i,:) \quad (6.7)$$

其中同样要注意,等式右端的 $c(i,:)$ 是原来的行向量,乘以 k 后赋值给新的 $c(i,:)$,所以右边的 $c(i,:)$ 不等于左边的 $c(i,:)$ 。

(3) 行相加: 第 i 个方程两端乘以常数 k ,加到第 j 行,替换第 j 行。用 MATLAB 语句可写成:

$$c(j,:) = c(j,:) + k*c(i,:) \quad (6.8)$$

同样要注意这是赋值语句,所以右边的 $c(j,:)$ 不等于左边的 $c(j,:)$ 。

以下,对于任何矩阵,不管它是左端的系数矩阵,还是把右端常数也放进去的增广矩阵,这里统一地都用 A 来表示。消元法是对它进行简化的一种最基本、效率最高的方法,也称为高斯消元法。其实这个方法早在公元前 225 年就由中国发明,但欧洲不知道。到 19

世纪德国数学家 Gauss 重新发现了这个方法，现在就常用他的姓氏命名。消元法的要点是，利用上述初等变换，在保持矩阵等价性和其他各行不变的前提下，改变矩阵的第 j 行，并把其中第 q 列的元素 $a(j,q)$ 变换为零。其做法是：先取一个行向量 $a(i,:)$ 为变换基准，对该行的要求是，元素 $a(i,q)$ 不等于零，然后采用第三种变换，并取 $k=-a(j,q)/a(i,q)$ 。此时第 j 行的数组将成为

$$a(j,:) = a(j,:) + -a(j,q)/a(i,q)*a(i,:) \quad (6.9)$$

而其中的第 q 个元素的取值将成为：

$$a(j,q) = a(j,q) + -a(j,q)/a(i,q)*a(i,q) = 0 \quad (6.10)$$

消元公式 (6.9) 表示的规则看来简单，但其中的冒号：意味着要对全行的所有元素都做一次乘法和一次加法，当 n 较大时，这种重复性工作也是够烦琐的。用计算机解题，就是要把我们掌握的规律变成程序，让计算机去做重复性的工作，而让人腾出时间进行高级的思维。因此要设法编出一个消元子程序 `gauss.m`。编程序之前，一定要弄清，该子程序所需的最低限度的输入变元（在子程序头等式的右端圆括号中）和输出变元（在子程序头等式的左端方括号中）。从式 (6.9) 可看出，等式右端所需的输入变元有 A, i, j, q 四个，等式左端的输出变元只有 A 一个。而这个 A 不等于右端的 A ，在程序头中若把两者赋予同样的变量名会造成内存冲突，必须另外给它取个名字 B 。函数名应该与它的功能相适应，以便于记忆，这里取为 `gauss`。

```
function [B]=gauss(A,i,j,q)
% -----
% function [B]=gauss(A,i,j,q)
% 高斯消元运算子程序
%
% A 为输入矩阵，B 为变换后的输出矩阵
% i 为基准行的行号
% j 为待变换行的行号
% q 为基准元的列号，即 A(i,q) 为基准元，A(j,q) 为待消元
%
x = A(i,:); y=A(j,:); % 取出 A 的第 i,j 两行命名为数组变量 x,y,
z = y - y(q)/x(q)*x; % 实现式 (6.9) 的运算
A(j,:)=z; B=A; % 把运算结果赋值给 A 第 j 行，并将 A 赋值给输出变元
```

紧接在子程序头的下方的带注释符 % 的连续多行提供了帮助说明。当在 MATLAB 命令窗中输入 `help gauss` 时，这些文字将出现在屏幕上，帮助用户正确调用子程序。注意子程序中的所有变量（例如程序中的 x, y ）都是局部变量，它们只在子程序内有效，不会与主程序的变量混淆。子程序与主程序的变量传递是通过程序头中的输入输出变元来完成的。读者还可以仔细思考一下，为什么在公式 (6.9) 中，等式左右可以用同样的变量名，而到子程序中就不允许这么做了？

6.3 行阶梯形式的生成

如果线性方程组的左端系数具有阶梯排列形式,即具有以下三个特点:

1. 所有非零行都处在全零行的上方;
2. 各行首非零元素(以后称之为枢轴元素 pivot element)的列号比其上方所有各行的枢轴元素的列号都要大;
3. 各枢轴元素所处的列中,在枢轴元素下方的所有元素均为零。

这样的矩阵称为行阶梯形式矩阵(或上三角矩阵)。当线性方程组的左端系数具有这种形式时,其解可以通过回代方法,自下而上,依次求得 $x(n), x(n-1), \dots, x(2), x(1)$ 。这样可以避免再联立消元,大大简化了求解的过程。利用第一种和第三种行初等变换,特别是由它们构成的消元子程序,可以很方便地把矩阵变换为行阶梯形式。

如果此矩阵还满足以下两个额外特点,则称之为简化行阶梯形式。

- 各行的枢轴元素是所在列中的唯一非零元素。
- 这些枢轴元素都等于 1。

要实现这种形式,除了利用消元子程序外,还要利用第二种行初等变换。行阶梯形式的回代解法,可见下例。

【例 6.3】 求解下列方程组:

$$\begin{aligned} x_1 - 3x_2 + 5x_3 - 2x_4 &= 0 \\ x_2 + 8x_3 &= -4 \\ 2x_3 - x_4 &= 4 \\ x_4 &= 2 \end{aligned}$$

用逐次回代法,可以依次求得 $x_4=2, x_3=2+x_4/2=3, x_2=-4-8x_3=-28, x_1=3x_2-5x_3+2x_4=-39$ 。

关于计算速度和精度的注记

可以分析一下回代所需的运算次数,在本题中,在各方程中的乘除法次数依次为 3+2+1=6 次。对于 n 元的方程,回代过程中要做的乘法次数为:

$$(n-1) + (n-2) + \dots + 2 + 1 = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

当 $n=100$ 时,就是 4,950 次。加减法的次数也差不多。可见这也不是手工能完成的事。

可见,求出等价的行阶梯形式是求解矩阵的重要步骤。那就要把矩阵的主对角线左下方各元素逐个消为零。现在来探讨如何用消元法把任意矩形的增广矩阵化为等价的行阶梯形式。其基本步骤是:

步骤 1 把主对角线下方第一列元素全变为零。

准备环节: 在矩阵各行中,选择第 1 列元素的绝对值最大的行,通过行交换,把它放到第 1 行(这个环节是为了提高计算精度,如果不考虑精度,只要第 1 行第 1 列元素不为零,也可以跳过不做这个环节)。

实质环节: 以第 1 行为基准行,依次把第 2 行至第 n 行的第 1 列的元素消为零,这可

以连续 $(n-1)$ 次调用 `gauss` 子程序来实现。即用

```
a1=gauss(a,1,2,1)
a1=gauss(a1,1,3,1)
...
a1=gauss(a1,1,n,1)
```

由于不需要保留中间结果,步骤 1 中的中间变量都采用 $a1$, 执行起来只须改变第 3 个变元, 最后得到的 $a1$ 的第一列中, 除第一个元素外, 其余元素均为零。

步骤 2 准备环节: 在 $2 \sim n$ 行中, 选择第 2 列元素的绝对值最大的行, 通过行交换, 把它放到第 2 行 (如果第 2 行第 2 列元素不为零, 这个环节也可以跳过不做)。

实质环节: 以第 2 行为基准行, 依次把第 3 行至第 n 行的第 2 列的元素消为零, 这可以连续 $(n-2)$ 次调用 `gauss` 子程序来实现。即用

```
a2=gauss(a1,2,3,2)
a2=gauss(a2,2,4,2)
...
a2=gauss(a2,2,n,2)
```

由于同样的道理, 步骤 2 中的中间变量都采用 $a2$ (如果不须保留中间结果, 可以一直用 $a1$ 到底), 最后的 $a2$ 的第 2 列中, 除 1, 2 两个元素外, 其余元素也均为零。

步骤 3 ...

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \xrightarrow[\text{步骤}]{\text{步}} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{122} & \cdots & a_{12n} \\ 0 & \vdots & & \vdots \\ 0 & a_{1n2} & \cdots & a_{1nn} \end{pmatrix} \xrightarrow[\text{步骤}]{\text{步}} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{122} & \cdots & a_{12n} \\ 0 & \vdots & & \vdots \\ 0 & 0 & a_{23n} & \cdots & a_{2nn} \end{pmatrix} \rightarrow \cdots$$

这个过程应当进行到第 $(n-1)$ 步, 按 $A \rightarrow A1 \rightarrow A2 \rightarrow \cdots \rightarrow A(n-1)$ 的次序, 直到矩阵左上至右下主对角线的左下方元素全部变成零为止。如果是增广矩阵, 那么其最右边的增广列不应算入主对角线范围。

【例 6.4】 用行阶梯及回代法求下列代数方程组的解

$$x_1 + 4x_2 + 7x_3 = 1$$

$$8x_1 + 5x_2 + 2x_3 = 3$$

$$3x_1 + 6x_2 - 2x_3 = 5$$

解: 先列出此线性方程组的系数增广矩阵, 再进行消元:

$$C = \begin{bmatrix} 1 & 4 & 7 & \vdots & 1 \\ 8 & 5 & 2 & \vdots & 3 \\ 3 & 6 & -2 & \vdots & 5 \end{bmatrix}$$

```
>>C1=gauss(C,1,2,1)
```

$$C1 = \begin{bmatrix} 1 & 4 & 7 & \vdots & 1 \\ 0 & -27 & -54 & \vdots & -5 \\ 3 & 6 & -2 & \vdots & 5 \end{bmatrix}$$

```
>>C1=gauss(C1,1,3,1)
```

$$C1 = \begin{bmatrix} 1 & 4 & 7 & \vdots & 1 \\ 0 & -27 & -54 & \vdots & -5 \\ 0 & -6 & -23 & \vdots & 2 \end{bmatrix}$$

```
>>C1=gauss(C1,2,3,2)
```

$$C1 = \begin{bmatrix} 1.0000 & 4.0000 & 7.0000 & \vdots & 1.0000 \\ 0 & -27.000 & -54.0000 & \vdots & -5.0000 \\ 0 & 0 & -11.0000 & \vdots & 3.1111 \end{bmatrix}$$

这时方程左端系数矩阵（不算增广列）已化为行阶梯形式，下面用回代方法即可得到：

$$x_3 = -3.1111/11 = -0.2828$$

$$x_2 = -(54*x_3 - 5)/27 = 0.7508$$

$$x_1 = 1 - 4*x_2 - 7*x_3 = -0.0236$$

关于计算速度和精度的注记

这里我们也来分析一下把矩阵化简为行阶梯形式所需的运算次数。在步骤1中，需要调用 $(n-1)$ 次消元子程序，每次子程序内要对每个行元素做乘法，故乘法运算次数是 $(n-1)$ ，因此步骤1的乘法次数为 $(n-1)^2$ ，依次类推，步骤2的乘法次数为 $(n-2)^2$ ，...最后总的乘法次数为

$$(n-1)^2 + (n-2)^2 + \cdots + 2^2 + 1 = \sum_{k=1}^{n-1} k^2 = \frac{1}{3}(n-1)^3 + \frac{1}{2}(n-1)^2 + \frac{1}{6}(n-1) < \frac{n^3}{3}$$

n 阶线性方程组求解所需的总运算次数应该是化简为行阶梯形式所需的运算次数与回代运算次数之和，即 $n^3/3 + n^2/2$ 。在 n 很大时，与 $n^3/3$ 相比，回代运算的次数 $n^2/2$ 可以忽略不计，所以用行阶梯法解一个 n 阶线性代数方程所需的乘法次数大约是 $n^3/3$ ，4阶的系统，需要21次乘法。如果 $n=100$ ，那就需要33万次乘法。可见线性代数的任何实际问题，不用计算机是无法解决的。

除了计算速度外，在计算机算题编程中要考虑的另一个问题是计算精度，在化简为行阶梯形式中，基准行和基准元的选择十分重要，通常要保证基准元的值尽量大。如果基准元太小，表现在消元子程序中的分母太小，会造成较大的舍入误差。要减小误差，就要在每个步骤开始时，通过行交换和列交换，把大的数调到基准元的位置，也就是矩阵主对角线上与该步骤相应的位置。在各种软件包中，这些问题都已考虑在内。线性代数的求解程序，已经非常成熟。在速度和精度上都是优化了的，决无必要由工学院毕业生的生手自己去编，这里只是为读者提供一个思路。

简化行阶梯形式的生成方法：矩阵的简化行阶梯形式也可以继续用消元法来生成，并同时得到方程的解，此时要从下而上，从右向左，先以最后一行为基准行，该行中 A 的最右一列元素（不算增广列）为基准元，输入


```
>>C1=gauss(C1,3,2,3)
```

$$C1 = \begin{bmatrix} 1.0000 & 4.0000 & 7.0000 & \vdots & 1.0000 \\ 0 & -27.000 & 0 & \vdots & -20.2727 \\ 0 & 0 & -11.0000 & \vdots & 3.1111 \end{bmatrix}$$

```
>>C1=gauss(C1,3,1,3)
```

$$C1 = \begin{bmatrix} 1.0000 & 4.0000 & 0 & \vdots & 2.9798 \\ 0 & -27.000 & 0 & \vdots & -20.2727 \\ 0 & 0 & -11.0000 & \vdots & 3.1111 \end{bmatrix}$$

```
>>C1=gauss(C1,2,1,2)
```

$$C1 = \begin{bmatrix} 1.0000 & 0 & 0 & \vdots & -0.0236 \\ 0 & -27.000 & 0 & \vdots & -20.2727 \\ 0 & 0 & -11.0000 & \vdots & 3.1111 \end{bmatrix}$$

此时增广矩阵中的 A 阵已化为对角形式，只要把第 2, 3 行两行分别除以对角项的值，就可以得出简化行阶梯形式。输入

```
>>C1(2,:)=1/27* C1(2,:); C1(3,:)=1/11* C1(3,:)
```

$$C1 = \begin{bmatrix} 1.0000 & 0 & 0 & \vdots & -0.0236 \\ 0 & 1.0000 & 0 & \vdots & 0.7508 \\ 0 & 0 & 1.0000 & \vdots & -0.2828 \end{bmatrix}$$

这就是最后的简化行阶梯形式，最右边的一列就是方程的解。本题所用的全部语句编入例题程序 ag604 中。

6.4 MATLAB 中的行阶梯生成函数

上面讨论了用 MATLAB 语句来实现线性方程组求解的过程，其目的是让读者弄清楚解决线性代数问题时大的思路和步骤，既不致被烦琐的数值运算弄昏了头脑，又不会囫圇吞枣地接受现成的结果。因为真正用现成的软件工具时，它已把所有的中间步骤全部集成起来了。读者在实际工作中要用的是最后的结果，但在考试或教别人时，就需要知道中间过程和道理。在本节中，向读者介绍的就是可以用来直接解题的 MATLAB 函数，此处我们假定读者已经具备了必要的矩阵赋值和四则运算的基本语句。

6.3 节所介绍的方法比起一个一个元素地计算，效率是高多了，但仍然很麻烦，头脑中要记各个步骤中的很多细节，而实际上这些步骤仍然是刻板而有规律的，容易机械化。MATLAB 已经把“简化行阶梯形式 (reduced row echelon form)”的计算过程集成为一个子程序 `rref`。它的输入变元可以是线性方程组的系数矩阵，也可以是其增广矩阵，输入

$U0=rref([A,b])$, 输出的结果就是精简行阶梯形式。

【例 6.5】 用 MATLAB 中的 `rref` 函数求解例 6.4 的方程组:

解: 写出 MATLAB 程序:

```
A=[1,4,7;8,5,2,;3,6,-2]; b=[1;3;5]; C=[A,b]; U0=rref(C)
```

运行结果为:

```
U0 =    1.0000    0    0   -0.0236
        0    1.0000    0    0.7508
        0    0    1.0000   -0.2828
```

这和我们上节用许多步骤推导的结果完全相同, 最右的一列就是线性方程组的解。即:

$x_1 = -0.0236, x_2 = 0.7508, x_3 = -0.2828$.

【例 6.6】 设一个 8 阶线性方程组的系数矩阵 A , b 如下, 求解线性代数方程 $A*x=b$:

$$A = \begin{bmatrix} 8 & 6 & 3 & 3 & 7 & 0 & 9 & 9 \\ 5 & 8 & 3 & 5 & 10 & 6 & 8 & 3 \\ 7 & 5 & 5 & 4 & 8 & 4 & 8 & 5 \\ 2 & 3 & 9 & 5 & 4 & 3 & 3 & 4 \\ 2 & 7 & 2 & 5 & 10 & 2 & 5 & 1 \\ 5 & 5 & 7 & 5 & 6 & 1 & 2 & 0 \\ 7 & 1 & 8 & 7 & 1 & 1 & 6 & 7 \\ 8 & 4 & 1 & 6 & 6 & 2 & 0 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ 1 \\ 0 \\ 7 \\ 6 \\ 1 \\ 0 \\ 3 \end{bmatrix}$$

解: 用程序 `ag606` 输入上述系数设定语句后, 输入 $U0=rref([A,b])$, 得到:

$$U0 = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.4414 \\ 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7467 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0.4553 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & -0.1003 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0.5979 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & -0.7748 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & -0.5123 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 1.3790 \end{bmatrix}$$

最右边的一列就是解 x 的各个分量 $[x(1);x(2);...;x(7);x(8)]$ 。为节省篇幅, 不列出。

6.5 用行阶梯法解欠定方程组

上面讨论的是属于方程数 m 与变量数 n 相等的情况。这时, 简化行阶梯形式的各行的

枢轴元素都在主对角线上，每下移一行，枢轴元素也右移一列，最后不会出现全零行，因此方程有唯一的解。

在普遍情况下，若 A 是 $m \times n$ 矩阵，当 $m < n$ 时，方程数小于未知数的数目，属于欠定方程，方程将有无数解，我们必须找出其解的一般形式。即使 $m = n$ ，也有可能是假象，因为有的方程可能是相依的，有效的方程数也许小于 m ，实际上还是欠定方程。要看清它究竟是什么类型，应该看行阶梯形式的结果，主要看它有多少有效行，有效行的数目就是该系数矩阵的“秩”。尽管因变换的次序不同，行阶梯形式有无数种数据，但它的有效行数，即秩是唯一的。

对增广矩阵 $C=[A,b]$ 进行行阶梯变换，得到的行阶梯矩阵 U_c 或简化行阶梯矩阵 U_{oc} 的下方可能有全零行出现，如下面两式所示：

行阶梯形式为：

$$U_c = \begin{bmatrix} \square & * & * & * & * & \vdots * \\ 0 & 0 & \square & * & * & \vdots * \\ 0 & 0 & 0 & \square & * & \vdots * \\ 0 & 0 & 0 & 0 & 0 & \vdots * \\ 0 & 0 & 0 & 0 & 0 & \vdots 0 \end{bmatrix},$$

简化行阶梯形式为

$$U_{oc} = \begin{bmatrix} 1 & * & 0 & 0 & * & \vdots * \\ 0 & 0 & 1 & 0 & * & \vdots * \\ 0 & 0 & 0 & 1 & * & \vdots * \\ 0 & 0 & 0 & 0 & 0 & \vdots * \\ 0 & 0 & 0 & 0 & 0 & \vdots 0 \end{bmatrix} = [U_0, d],$$

矩阵中的竖向虚线表示原方程中等号的位置，它的左边是方程中的左端系数矩阵，它的右边是方程右端的常数项。把简化结果的 U_0 和 d 分开来写：

$$U_0 = \begin{bmatrix} 1 & * & 0 & 0 & * \\ 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad d = \begin{bmatrix} * \\ * \\ * \\ * \\ 0 \end{bmatrix}$$

当 U_0 中某行各变量系数全为零时， d 中的对应行也必须为零，否则就构成了等式左右不相等的矛盾方程，说明原方程是无解的不相容方程组。因此，矩阵 A 和增广矩阵 $[A,b]$ 的行阶梯形式 U 和 U_c （或简化行阶梯形式 U_0 和 U_{oc} ）应当有同样的全零行，也就是两者有同样的秩，这是方程组有解的必要条件。

原来的 m 行中只剩下 r 个非全零行，意味着 m 个方程中只有 r 个有效，也就是它的秩为 r ， $m-r$ 个全零行反映了原方程组中有 $m-r$ 个是相依方程，最后 U_0 中有效的部分是 $r \times n$ 矩阵，就是 r 个方程和 n 个未知数。因为 $r < n$ ，这是一个欠定方程组， n 个未知数（变量）

中有 $n-r$ 个可以任意设定, 我们称这些未知数为自由变量, 也可把它们称为任意常数。

把行阶梯矩阵中 r 个非零行的首非零元素 (即枢轴元素) 以 \square (在简化行阶梯形式中则为 1) 表示, 它所处的行和列分别称为枢轴行 (Pivot rows) 和枢轴列 (Pivot columns)。因为枢轴行对应的是简化后方程的系数。其各列则对应于不同变量的系数。在本例中, 把枢轴元素保留在等式左边, 把其余的各项都移到等式右边, 我们就得到了三个枢轴变量的表示式。在等式右边只有 $n-3$ 个非枢轴变量项和常数项。这 $n-3$ 个变量是可以任意设定的, 最简单的方法是设它们为零, 则三个枢轴变量就可直接求出。枢轴列的列号虽然用观测法可以得到, 但对于几十阶以至几百阶的系统, 还是应该由计算机给出。MATLAB 的行简化函数 `rref` 在运算过程中本来就会产生这些列号, 所以可以向用户提供, 其调用格式为:

```
[U0,ip]=rref(A)
```

其中, `rref` 函数的第二个输出变元 ip 就是枢轴列的列号向量。

【例 6.7】 设线性方程组如下, 试求它的解:

$$\begin{aligned} 3x_1 - 4x_2 + 3x_3 + 2x_4 - x_5 &= 2 \\ -6x_2 - 3x_4 - 3x_5 &= -3 \\ 4x_1 - 3x_2 + 4x_3 + 2x_4 - 2x_5 &= 2 \\ x_1 + x_2 + x_3 - x_5 &= 0 \\ -2x_1 + 6x_2 - 2x_3 + x_4 + 3x_5 &= 1 \end{aligned}$$

解: 先输入其系数矩阵 A, b , 求它的增广矩阵的简化行阶梯形式。输入程序 `ag607`

```
A=[3,-4,3,2,-1;0,-6,0,-3,-3;4,-3,4,2,-2;1,1,1,0,-1;-2,6,-2,1,3];
b=[2;-3;2;0;1];
B=[A,b],[UB,ip]=rref(B);
U0=UB([1:5,1:5]),d=UB(:,6)
```

得到

$$[A,b]=\begin{bmatrix} 3 & -4 & 3 & 2 & -1 & 2 \\ 0 & -6 & 0 & -3 & -3 & -3 \\ 4 & -3 & 4 & 2 & -2 & 2 \\ 1 & 1 & 1 & 0 & -1 & 0 \\ -2 & 6 & -2 & 1 & 3 & 1 \end{bmatrix}$$

及

$$U0=\begin{bmatrix} 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, ip=[1,2,4]$$

可见, 下方有两个全零行, 说明原方程中有两个方程是相依的, 只有三个独立方程, 此时各阶梯的枢轴元不在主对角线上。枢轴列的列号 (也就是枢轴变量的序号) 为 1, 2

和 4。我们应当把 x_1, x_2, x_4 看作待求的变量，而把其余的两个变量 x_3 和 x_5 看作自由变量。把 U_0 系数矩阵恢复成方程形式并移项，使左端只包含待求变量，把 x_3 和 x_5 移到等式右边，其系数应反号。注意最后一列常数项本来就在等式右边，故不需反号，即把 U_0 先“翻译”为下面左端的方程组。移项成为右端的方程组。

$$\left. \begin{array}{l} x_1 + x_3 - x_5 = 0 \\ x_2 = 0 \\ x_4 + x_5 = 1 \\ 0 = 0 \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} x_1 = -x_3 + x_5 \\ x_2 = 0 \\ x_4 = -x_5 + 1 \end{array} \right. \quad x_3 \text{ 与 } x_5 \text{ 为任意常数}$$

如果要给出一个具体解，通常取自由变量 x_3 和 x_5 为零，这样得出的解是 $x_1=x_2=x_3=x_5=0$ ， $x_4=-1$ 。但这样来表示方程的解容易造成误会，人们会误以为方程组只有一个解。所以比较严格一些的表达方法应该是取自由变量 x_3 和 x_5 为 c_1 和 c_2 ：

$$x_3=c_1, x_5=c_2, x_1=-c_1+c_2, x_2=0, x_4=-c_2+1$$

选择不同的 c_1 和 c_2 ，就可以得出不同的解，所以其解有无数组。由于它包含两个自由变量，因此在两个自由度上都可以在 $(-\infty, \infty)$ 范围内变化，相当于“无穷大的平方”，从几何上说，它的解占据了一个两维空间（平面）。

把这个具体例子中的规律总结出来，编成程序 ag607a。令枢轴列号为 i_p ，非枢轴列号为 i_s ，ATLAST MANUAL 提供了计算 i_s 的函数 $i_s=\text{other}(i_p, n)$ ；则下列程序将能自动完成对解的计算，其中假定自由变量的数目不大于 5。

```
syms c1 c2 c3 c4 c5
[m,n]=size(A);
is=other(ip,n),
c=[c1;c2;c3;c4;c5]
x(is,1)=c(1:length(is))
lp=length(ip)
x(ip,1)=-U0(1:lp,ip'+1:n)*x(ip'+1:n)+U0(1:lp,end)
```

不过在学习阶段，我们并不主张太机械化。应该机械化的是那些烦琐的计算，需要由人的智慧进行理解判断的部分还是保留手工作业为好。

6.6 应用实例

6.6.1 平板稳态温度的计算

我们来研究一个薄铁板的热传导问题。设该平板的周边温度已经知道（如图 6.4 所示，数字的单位为“℃”），现在要确定板中间 4 个点 a, b, c, d 处的温度。假定其热传导过程已经达到稳态，因此在均匀的网格点上，各点的温度是其上下左右 4 个点的温度的平均值。于

是对这几个点列出如下方程:

$$x_a = (10 + 20 + x_b + x_c) / 4$$

$$x_b = (20 + 40 + x_a + x_d) / 4$$

$$x_c = (10 + 30 + x_a + x_d) / 4$$

$$x_d = (40 + 30 + x_b + x_c) / 4$$

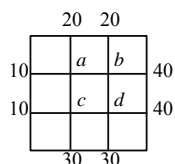


图 6.4 平板的温度分布

移项整理为标准的矩阵方程:

$$\begin{bmatrix} 1 & -0.25 & -0.25 & 0 \\ -0.25 & 1 & 0 & -0.25 \\ -0.25 & 0 & 1 & -0.25 \\ 0 & -0.25 & -0.25 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ x_b \\ x_c \\ x_d \end{bmatrix} = \begin{bmatrix} 7.5 \\ 15 \\ 10 \\ 17.5 \end{bmatrix}$$

即为 $A \cdot x = b$ 的形式。

输入 MATLAB 程序 ag661:

```
A=[1,-0.25,-0.25,0;-0.25,1,0,-0.25;-0.25,0,1,-0.25;0,-0.25,-0.25,1]
b=[7.5;15;10;17.5]
U=rref([A,b])
```

程序运行的结果为:

$$U = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 20.0000 \\ 0 & 1.0000 & 0 & 0 & 27.5000 \\ 0 & 0 & 1.0000 & 0 & 22.5000 \\ 0 & 0 & 0 & 1.0000 & 30.0000 \end{bmatrix}$$

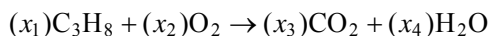
把它“翻译”为方程, 即有

$$x_a = 20^\circ\text{C}, \quad x_b = 27.5^\circ\text{C}, \quad x_c = 22.5^\circ\text{C}, \quad x_d = 30^\circ\text{C}.$$

读者可以用手工对这几点的温度进行验算, 看它们是否等于相邻四点温度的平均值。

6.6.2 化学方程的配平

化学方程描述了被消耗和新生成的物质之间的定量关系。例如, 化学实验的结果表明, 丙烷燃烧时将消耗氧气并确实产生二氧化碳和水, 其化学反应的方程为:



要配平这个方程, 必须找到适当的 x_1, x_2, x_3, x_4 , 使得反应式左右的碳、氢、氧元素相匹配。

配平化学方程的标准方法是建立一个向量方程组, 每个方程分别描述一种原子在反应

前后的数目。在上面的方程中，有碳、氢、氧三种元素需要配平，构成了三个方程。而有四种物质，其数量用四个变量 x_1, x_2, x_3, x_4 来表示。将每种物质分子中的元素原子数按碳、氢、氧的次序排成列，可以写出：

$$\text{C}_3\text{H}_8 : \begin{bmatrix} 3 \\ 8 \\ 0 \end{bmatrix}, \text{O}_2 : \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}, \text{CO}_2 : \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \text{H}_2\text{O} : \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

要使方程配平， x_1, x_2, x_3, x_4 必须满足：

$$x_1 \cdot \begin{bmatrix} 3 \\ 8 \\ 0 \end{bmatrix} + x_2 \cdot \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = x_3 \cdot \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} + x_4 \cdot \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

将所有项移到左端，并写成矩阵相乘的形式，就有：

$$\begin{bmatrix} 3 & 0 & -1 & 0 \\ 8 & 0 & 0 & -2 \\ 0 & 2 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \mathbf{Ax} = \mathbf{0}$$

对矩阵 \mathbf{A} 进行行阶梯变换，输入程序 ag662：

```
A=[3,0,-1,0;8,0,0,-2;0,2,-2,-1]
```

```
U0=rref(A)
```

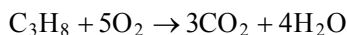
得到

$$\mathbf{U0} = \begin{bmatrix} 1.0000 & 0 & 0 & -0.2500 \\ 0 & 1.0000 & 0 & -1.2500 \\ 0 & 0 & 1.0000 & -0.7500 \end{bmatrix}$$

要注意这四个列对应于四个变量的系数，所以这三行系数对应的方程是：

$$\begin{aligned} x_1 & -0.2500x_4 = 0 \\ x_2 & -1.2500x_4 = 0 \\ x_3 & -0.7500x_4 = 0 \end{aligned}$$

即 x_4 是自由变量。因为化学家们喜欢把方程的系数取为最小可能的整数，此处可取 $x_4=4$ ，则 x_1, x_2, x_3 均有整数解， $x_1=1, x_2=5, x_3=3$ 。因而配平后的化学方程为：



可见要配平比较复杂的、有多种物质参与作用的化学反应，需要解相当复杂的线性代数方程组。对于比较复杂的反应过程，为了便于得到最小整数的解，在解化学配平的线性方程组时，应该在 MATLAB 中先规定取有理分式格式。即先输入 `format rat`，然后输入 `ag662`，结果为：

$$U0 = \begin{bmatrix} 1 & 0 & 0 & -1/4 \\ 0 & 1 & 0 & -5/4 \\ 0 & 0 & 1 & -3/4 \end{bmatrix}$$

这样就很容易看出应令 $x_4=4$ ，几个整数的取值就一目了然了。

6.6.3 电阻电路的计算

如图 6.5 的电路。已知：

$$R_1=2\Omega, R_2=4\Omega, R_3=12\Omega, R_4=4\Omega,$$

$$R_5=12\Omega, R_6=4\Omega, R_7=2\Omega$$

设电压源 $u_s=10V$ 求 i_3 , u_4 , u_7

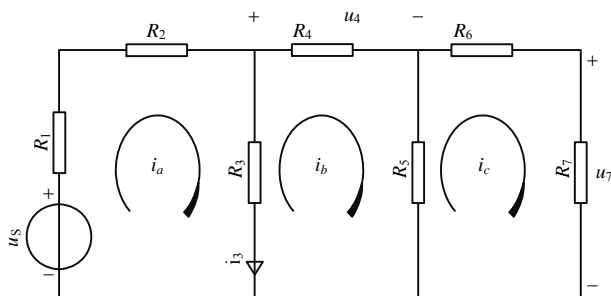


图 6.5 应用实例 6.6.3 的电路图

解：首先进行建模。所选回路如图 6.5 所示，设每个网孔的回路电流分别为 i_a , i_b 和 i_c 。据根基尔霍夫定律，任何回路中诸元件上的电压之和等于零。

按图可列出各回路的电压方程为：

$$\begin{aligned} (R_1+R_2+R_3) i_a - R_3 i_b &= u_s \\ -R_3 i_a + (R_3+R_4+R_5) i_b - R_5 i_c &= 0 \\ -R_5 i_b + (R_5+R_6+R_7) i_c &= 0 \end{aligned}$$

写成矩阵形式为：

$$\begin{bmatrix} R_1+R_2+R_3 & -R_3 & 0 \\ -R_3 & R_3+R_4+R_5 & -R_5 \\ 0 & -R_5 & R_5+R_6+R_7 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} u_s \\ 0 \\ 0 \end{bmatrix}$$

也可把参数代入，直接列出数字方程

$$\begin{bmatrix} 18 & -12 & 0 \\ -12 & 28 & -12 \\ 0 & -12 & 18 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix}$$

简写成

$$A \cdot I = b$$

其中 $I=[i_a; i_b; i_c]$ 。今 u_s 已知为 10，解此矩阵方程，即可得问题的解。

为此，可列出 MATLAB 程序


```
A=[18,-12,0; -12,28,-12; 0,-12,18];
b=[10;0;0];
U0=rref([A,b])
```

程序运行的结果为：

$$U0 = \begin{bmatrix} 1.0000 & 0 & 0 & \vdots & 0.926 \\ 0 & 1.0000 & 0 & \vdots & 0.556 \\ 0 & 0 & 1.0000 & \vdots & 0.370 \end{bmatrix}$$

意味着：

$$\mathbf{I} = \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} 0.926 \\ 0.556 \\ 0.370 \end{bmatrix}$$

任何稳态电路问题，都可以用线性代数方程描述。直流电路构成的是实系数方程，其解为实数；而交流电路构成的是复数系数方程，其解将是复数。所以，即使是二元、三元的线性方程组，解起来也很烦琐，使用矩形方程和计算机软件就更为必要了。

6.6.4 交通流量的分析

某城市有两组单行道，构成了一个包含四个节点 A,B,C,D 的十字路口，如图 6.6 所示。在交通繁忙时段的汽车从外部进出此十字路口的流量（每小时的车流数）标于图上。现要求计算每两个节点之间路段上的交通流量 x_1, x_2, x_3, x_4 。

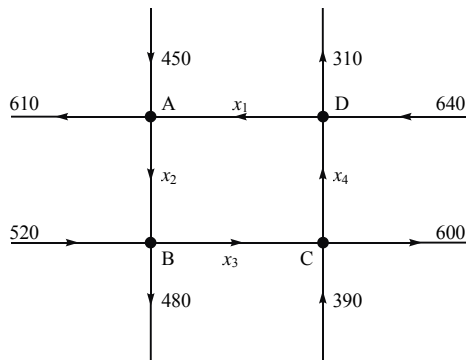


图 6.6 单行线交通流量图

解：在每个节点上，进入和离开车数应该相等，这就决定了四个节点的流通方程：

$$\text{节点 A: } x_1 + 450 = x_2 + 610$$

$$\text{节点 B: } x_2 + 520 = x_3 + 480$$

$$\text{节点 C: } x_3 + 390 = x_4 + 600$$

$$\text{节点 D: } x_4 + 640 = x_1 + 310$$

将这组方程进行整理，写成矩阵形式：

$$\begin{array}{rcl}
 x_1 - x_2 & & = 160 \\
 & x_2 - x_3 & = -40 \\
 & & x_3 - x_4 = 210 \\
 -x_1 & & x_4 = -330
 \end{array}$$

其系数增广矩阵为: $[A, b] = \begin{bmatrix} 1 & -1 & & \vdots & 160 \\ & 1 & -1 & \vdots & -40 \\ & & 1 & -1 & \vdots & 210 \\ -1 & & & 1 & \vdots & -330 \end{bmatrix}$

用消元法求其行阶梯形式, 或者直接调用 $U0 = \text{rref}([A, b])$, 列出程序 ag664:

```

A=[1,-1,0,0;0,1,-1,0;0,0,1,-1;-1,0,0,1]
b=[160;-40;210;-330]
U0=rref([A,b])

```

可以得出其精简行阶梯形式为

$$U0 = \begin{bmatrix} 1 & 0 & 0 & -1 & \vdots & 330 \\ 0 & 1 & 0 & -1 & \vdots & 170 \\ 0 & 0 & 1 & -1 & \vdots & 210 \\ 0 & 0 & 0 & 0 & \vdots & 0 \end{bmatrix}$$

注意这个系数矩阵所代表的意义, 它的左边四列从左至右依次为变量 x_1, x_2, x_3, x_4 的系数, 第五列则是在等式右边的常数项。把第四列移到等式右边, 可以按行列写恢复为方程, 其结果为:

$$\begin{aligned}
 x_1 &= x_4 + 330, \\
 x_2 &= x_4 + 170, \\
 x_3 &= x_4 + 210 \\
 0 &= 0
 \end{aligned}$$

由于最后一行变为全零, 这个精简行阶梯形式只有三行, 也就是说四个方程中有一个是相依的, 实际上只有三个有效方程。方程数比未知数的数目少, 即没有给出足够的信息来唯一地确定 x_1, x_2, x_3 和 x_4 , 其原因也不难从物理上理解。题目给出的只是进入和离开这个十字路口的流量, 如果有些车沿着这四方的单行道绕圈, 那是不会影响总的输入输出流量的, 但可以全面增加四条路上的流量。所以 x_4 被称为自由变量。实际上它的取值也不能完全自由, 因为规定了这些路段都是单行道, x_1, x_2, x_3 和 x_4 都不能取负值。

所以要准确了解这里的交通流量情况, 还应该在 x_1, x_2, x_3 和 x_4 中, 再检测一个变量。

6.7 习题

6.1 用 `gauss` 消元子程序将下列方程的增广矩阵变换为（非简化的）行阶梯形式，列出每一步所用的 MATLAB 消元语句。

$$\begin{array}{ll}
 \text{(a)} \quad \begin{array}{l} x_1 + x_2 = -1 \\ 4x_1 - 3x_2 = 3 \end{array} & \begin{array}{l} x_1 + 3x_2 + x_3 + x_4 = 3 \\ 2x_1 - 2x_2 + x_3 + 2x_4 = 8 \\ 3x_1 + x_2 + 2x_3 - x_4 = -1 \end{array} \\
 \text{(c)} \quad \begin{array}{l} x_1 + x_2 + x_3 = 0 \\ x_1 - x_2 - x_3 = 0 \end{array} & \begin{array}{l} x_1 + x_2 + x_3 + x_4 = 0 \\ 2x_1 + x_2 - x_3 + 3x_4 = 0 \\ x_1 - 2x_2 + x_3 + x_4 = 0 \end{array}
 \end{array}$$

6.2 将下列方程用 `gauss` 消元子程序变换为行阶梯形式所对应的方程，指明方程组是否相容。如果系统是相容的并且没有自由变量，用回代法求出其唯一的解。并将算出的结果与用 `rref` 函数所得的简化行阶梯形式的结果进行比较。

$$\begin{array}{ll}
 \text{(a)} \quad \begin{array}{l} 2x_1 + 3x_2 + x_3 = 1 \\ x_1 + x_2 + x_3 = 3 \\ 3x_1 + 4x_2 + 2x_3 = 4 \end{array} & \begin{array}{l} x_1 + x_2 + x_3 + x_4 = 0 \\ 2x_1 + 3x_2 - x_3 - x_4 = 2 \\ 3x_1 + 2x_2 + x_3 + x_4 = 5 \\ 3x_1 + 6x_2 - x_3 - x_4 = 4 \\ -x_1 + 2x_2 - x_3 = 2 \end{array} \\
 \text{(c)} \quad \begin{array}{l} x_1 + 3x_2 + x_3 + x_4 = 3 \\ 2x_1 - 2x_2 + x_3 + 2x_4 = 8 \\ x_1 - 5x_2 + x_4 = 5 \end{array} & \begin{array}{l} -2x_1 + 2x_2 + x_3 = 4 \\ 3x_1 + 2x_2 + 2x_3 = 5 \\ -3x_1 + 8x_2 + 5x_3 = 17 \end{array}
 \end{array}$$

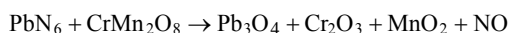
6.3 用 `A=round(10*rand(8))` 生成一个 8×8 的随机矩阵，用 `b=round(10*rand(8,1))` 生成一个 8×1 的随机向量。

- 用 `U=rref([A,b])` 求出其增广矩阵的简化行阶梯形式。说明方程 $A \cdot x = b$ 的解为什么是 `U` 最右边的一列。给出解的数值。
- 用 `U=rref([A,eye(8)])` 求出其增广矩阵的简化行阶梯形式。取出此矩阵的最右边 8 列，组成一个新矩阵 `B`。列出取 `B` 的 MATLAB 语句。
- 说明 `B` 的含义，并用数字证明此含义的正确性。
- 说明如何用 `B` 来求出方程的解 `x`，并与(a)的结果相比较。方法是在 `format long` 条件下，把两个解相减。观察其差。

6.4 一个矿业公司有两个矿井。1 号矿井每天生产 20 公吨铜矿石和 550 公斤银矿石，2 号矿井每天生产 30 公吨铜矿石和 500 公斤银矿石，令 $v_1 = \begin{bmatrix} 20 \\ 550 \end{bmatrix}$ 及 $v_2 = \begin{bmatrix} 30 \\ 500 \end{bmatrix}$ ，于是 v_1 和 v_2 分别表示了 1 号和 2 号矿井每天的产出向量。问：

- 向量 $5v_1$ 具有何种物理意义？
- 设公司让 1 号矿井开工 x_1 天，让 2 号矿井开工 x_2 天，列出使两个矿井生产出 150 公吨铜矿石和 2825 公斤银矿石以天数为变量的方程。
- 解这个方程。

6.5 尽量用有理格式来配平下列化学方程。



6.6 尽量用有理格式来配平下列化学方程。



6.7 某城市有两组单行道，构成了一个包含四个节点 A,B,C,D 的十字路口如图 6.7 所示。在交通繁忙时段的汽车从外部进出此十字路口的流量（每小时的车流数）标于图上，只有从北边流出的流量 x_3 待定。现要求计算 x_3 及每两个节点之间路段上的交通流量 x_1, x_2, x_4, x_5 。

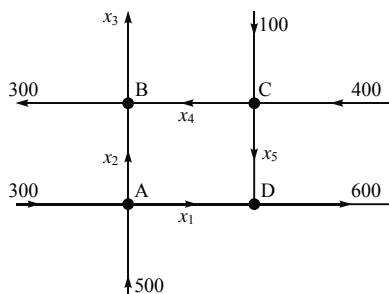


图 6.7 两组单行道交通流量图

6.8 对图 6.8 所示的电路，根据克希荷夫定律，

(a) 列出它的四个回路的电压方程。

(b) 将这四个回路的电压方程写成矩阵形式

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

确定 \mathbf{A} 和 \mathbf{b} 的具体内容。

(c) 用 MATLAB 的 rref 语句解这个方程。

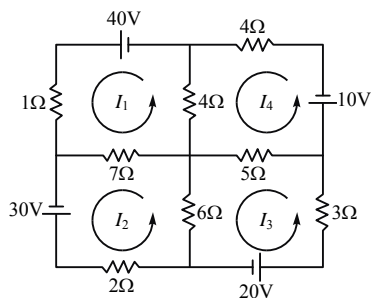


图 6.8 题 6.8 的电路图

6.9 对图 6.4 所示的平板温度分布图，若把边界温度条件向右转过 90° ，即上方为 10°C , 20°C ；右方为 20°C , 30°C ；下方为 40°C , 30°C （从左向右）；左方为 10°C , 40°C （从上向下）；问对平板中间 a, b, c, d 各点的温度有何影响？作出解释。

用矩阵运算法解线性方程

7.1 矩阵运算规则的 MATLAB 实现

这里把矩阵运算的一些主要规则归纳一下，并把与这些运算对应的 MATLAB 运算符也加以说明。

1. 矩阵加（减）法：两矩阵的相加（减）就是其对应元素的相加（减），因此，该两矩阵的阶数必须相同，且它们的和也有相同的阶数。 $C=A+B$ 的 MATLAB 实现为

$$C(i,j)=A(i,j)+B(i,j) \quad (i=1,\dots,m; j=1,\dots,n)$$

因此，两个 $m \times n$ 矩阵的和还是 $m \times n$ 矩阵。

2. 矩阵乘法： $m \times p$ 阶矩阵 A 与 $p \times n$ 阶矩阵 B 的乘积 C 是一个 $m \times n$ 阶矩阵，它的任何一个元素 $C(i,j)$ 的值为 A 矩阵的第 i 行和 B 矩阵的第 j 列对应元素乘积的和。即

$$A * B = \begin{pmatrix} A(1,1) & \cdots & A(1,p) \\ \cdots & & \cdots \\ A(m,1) & \cdots & A(m,p) \end{pmatrix} \begin{pmatrix} B(1,1) & \cdots & B(1,n) \\ \cdots & & \cdots \\ B(p,1) & \cdots & B(p,n) \end{pmatrix} = \begin{pmatrix} C(1,1) & \cdots & C(1,n) \\ \cdots & & \cdots \\ C(m,1) & \cdots & C(m,n) \end{pmatrix} = C$$

$$\text{其中 } C(i, j) = A(i, 1) * B(1, j) + A(i, 2) * B(2, j) + \cdots + A(i, p) * B(p, j) = \sum_{k=1}^p A(i, k) * B(k, j)$$

式中的乘号是普通数（标量）的乘号。 p 是 A 矩阵的列数，也是 B 矩阵的行数， p 被称为两个相乘矩阵的内阶数，两矩阵相乘的必要条件是它们的内阶数相等。

如果要自己编写矩阵 A 和 B 加、减、乘的程序，就必须先求出矩阵 A 和矩阵 B 的行数和列数，按上述规则进行检验，合格后再按定义写出程序。实际上 MATLAB 已将上述矩阵加、减、乘的程序编程为内部函数，我们只要用 +、-、* 作为运算符号，它就包含了

检查阶数和执行运算的全过程，而且其运算对复数有效。

矩阵乘法和普通标量的乘法有一个根本的差别，就是它不满足交换律， $\mathbf{AB} \neq \mathbf{BA}$ ，这是初学者很容易出错的地方。比如：

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

则

$$\mathbf{AB} = 1*4 + 2*5 + 3*6 = 32$$

而

$$\mathbf{BA} = \begin{bmatrix} 4*1 & 4*2 & 4*3 \\ 5*1 & 5*2 & 5*3 \\ 6*1 & 6*2 & 6*3 \end{bmatrix} = \begin{bmatrix} 4 & 8 & 12 \\ 5 & 10 & 15 \\ 6 & 12 & 18 \end{bmatrix}$$

得出的一个是标量，一个是矩阵。两者之间只是左右交换了一下，真是失之毫厘，差之千里。然而矩阵乘法的这种性能是非常有用的，它可以简洁地表达许多复杂的科学计算问题，本书将提供多个实例来说明这一点。

要注意的是，有许多我们习惯的公式，其中隐含地包含了交换律，这些公式在矩阵运算中也不能直接使用。比如：

$$(\mathbf{A} + \mathbf{B})^2 \neq \mathbf{A}^2 + 2\mathbf{AB} + \mathbf{B}^2$$

$$(\mathbf{A} + \mathbf{B})(\mathbf{A} - \mathbf{B}) \neq \mathbf{A}^2 - \mathbf{B}^2$$

等等，读者应该按原有的运算次序（加减法可以交换）进行直接计算。如果一定要用公式化简，则在把这类公式左端展开时，必须保持其原有的左右排列。如：

$$(\mathbf{A} + \mathbf{B})^2 = \mathbf{A}(\mathbf{A} + \mathbf{B}) + \mathbf{B}(\mathbf{A} + \mathbf{B}) = \mathbf{A}^2 + \mathbf{AB} + \mathbf{BA} + \mathbf{B}^2$$

3. 矩阵求逆：矩阵的逆阵 \mathbf{V} 定义为满足 $\mathbf{V}*\mathbf{A}=\mathbf{I}$ (\mathbf{I} 为与 \mathbf{A} 同阶的单位矩阵)。只有方阵才可以求逆。而且此方阵的行列式必须不为零， $\det(\mathbf{A}) \neq 0$ 。MATLAB 中的求逆函数为 $\mathbf{V}=\text{inv}(\mathbf{A})$ 。如果 $\det(\mathbf{A})=0$ ，求逆时就会出现无穷大 (Inf)。此时的矩阵称为奇异的。矩阵求逆是一个特别重要的问题，我们将单独用一节来讨论。

4. 矩阵的转置：矩阵的行列互换后构成转置矩阵。

$$\text{设 } \mathbf{A} = \begin{pmatrix} \mathbf{A}(1,1) & \cdots & \mathbf{A}(1,n) \\ \cdots & & \cdots \\ \mathbf{A}(m,1) & \cdots & \mathbf{A}(m,n) \end{pmatrix} \text{ 则它的转置矩阵 } \mathbf{A}^T = \begin{pmatrix} \mathbf{A}(1,1) & \cdots & \mathbf{A}(1,m) \\ \vdots & & \vdots \\ \mathbf{A}(n,1) & \cdots & \mathbf{A}(n,m) \end{pmatrix}$$

注意矩阵中的下标，如果 \mathbf{A} 是 $m \times n$ 阶的，则 \mathbf{A}^T 是 $n \times m$ 阶的。在 MATLAB 中， \mathbf{A}' 是共轭转置符号， \mathbf{A}^T 是转置符号。对于实数矩阵，共轭不起作用， $\mathbf{A}' = \mathbf{A}^T$ 。例如

$$\mathbf{A} = [1, 2, 3; 6, 7, 8], \quad \mathbf{B} = \mathbf{A}'$$

得到:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 7 & 8 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 6 \\ 2 & 7 \\ 3 & 8 \end{bmatrix}$$

5. 矩阵分块: 矩阵 \mathbf{A} 可以划分为很多小矩阵, 称为矩阵分块。

设要将矩阵按下式分块:

$$\mathbf{A} = \begin{bmatrix} 9 & 7 & 6 & \vdots & 8 & 8 \\ -5 & 5 & -1 & \vdots & 5 & 8 \\ 2 & -1 & 2 & \vdots & -6 & -2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & -9 & 6 & -2 & \vdots & 7 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix}$$

则可用 MATLAB 中的分块语句:

$$\mathbf{A} = [9, 7, 6, 8, 8; -5, 5, -1, 5, 8; 2, -1, 2, -6, -2; 0, 9, 6, -2, 7]$$

$$\mathbf{A}_1 = \mathbf{A}(1:3, 1:3), \quad \mathbf{A}_2 = \mathbf{A}(1:3, 4:5), \quad \mathbf{A}_3 = \mathbf{A}(4, 1:4), \quad \mathbf{A}_4 = \mathbf{A}(4, 5)$$

6. 矩阵分解为向量: 最常见的分块方法是把矩阵沿行向或列向分解为单列或单行向量, 尤其多用的是分解为列向量。方法如下:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_n],$$

其中

$$\mathbf{v}_1 = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix}, \cdots, \mathbf{v}_n = \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

为 n 个列向量, 每个向量的长度为 m 。将它们沿列向并排起来, 就得到矩阵 \mathbf{A} 。也可以把矩阵 \mathbf{A} 分解为行向量, 请读者自行分析。

7. 行向量左乘列向量

行向量与列向量的相乘, 如果行向量在左, 列向量在右, 则得出的是一个标量。此时要求两个向量的长度必须一致。我们就用上述列向量 \mathbf{v}_1 的转置乘 \mathbf{v}_1 , 看看它的结果。

$$\mathbf{v}_1^T \mathbf{v}_1 = [a_1, a_2, \cdots, a_n] \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = a_1^2 + a_2^2 + \cdots + a_n^2 = \sum_{i=1}^n a_i^2$$

得出的是向量各分量的平方和。如果这些分量是相互正交的, 则得出的是向量的长度平方。它的平方根就是向量长度, 也称为向量的范数或 2 范数 ($\text{norm}(\mathbf{v}_1)$)。

行向量左乘列向量得到的是 $R = \sum_{i=1}^n x_i y_i$ 的形式，在工程中具有这样计算形式的问题很多，比如用它计算两个向量 \mathbf{x} 和 \mathbf{y} 之间的相关性。

8. 列向量左乘行向量：

若把列向量放在左边，行向量放在右边相乘，这时内阶数均为 1。如果列向量长度为 n ，行向量长度为 m ，则相乘会得出一个 $n \times m$ 的矩阵，这种方法常常用来生成和计算一些复杂的大矩阵。

例如要绘制一个三维的平面，我们必须给 $\text{mesh}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ 函数提供三个输入变元 \mathbf{X} , \mathbf{Y} , \mathbf{Z} 。此时自变量是在 x, y 平面上一个矩形面积内的许多点构成的阵列。假如 \mathbf{X} 和 \mathbf{Y} 都是在 $[-10, 10]$ 的范围，用间隔为 1 的网格选定这些点阵；于是 x 和 y 方向都取 21 个点，构成了总共 $21 \times 21 = 441$ 个数据点。 \mathbf{X} , \mathbf{Y} 和 \mathbf{Z} 就是这 441 个点的 x , y 和 z 坐标，它们都应当是 21×21 阶的方阵。这些自变量坐标可表示为：

$$(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} (-10,10) & (-9,10) & \cdots & (0,10) & \cdots & (9,10) & (10,10) \\ (-10,9) & (-9,9) & \cdots & (0,9) & \cdots & (9,9) & (10,9) \\ \cdots & \cdots & & \cdots & & \vdots & \\ (-10,-9) & (-9,-9) & \cdots & (0,-9) & \cdots & (9,-9) & (10,-9) \\ (-10,-10) & (-9,-10) & \cdots & (0,-10) & \cdots & (9,-10) & (10,-10) \end{bmatrix},$$

我们还没有学过在一个矩阵元素上输入一对数的表示方法 (MATLAB 中有此方法，但本书不讲)，现在就采取两个矩阵 \mathbf{X} 和 \mathbf{Y} 分别表示其中的 441 个 \mathbf{x} 和 441 个 \mathbf{y} 。不难看出，这两个自变量方阵 \mathbf{X} 和 \mathbf{Y} 的数据如下：

$$\mathbf{X} = \begin{bmatrix} -10 & -9 & \cdots & 0 & \cdots & 9 & 10 \\ -10 & -9 & \cdots & 0 & \cdots & 9 & 10 \\ \cdots & \cdots & & \cdots & & \vdots & \\ -10 & -9 & \cdots & 0 & \cdots & 9 & 10 \\ -10 & -9 & \cdots & 0 & \cdots & 9 & 10 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 10 & 10 & \cdots & 10 & \cdots & 10 & 10 \\ 9 & 9 & \cdots & 9 & \cdots & 9 & 9 \\ \cdots & \cdots & & \cdots & & \vdots & \\ -9 & -9 & \cdots & -9 & \cdots & -9 & -9 \\ -10 & -10 & \cdots & -10 & \cdots & -10 & -10 \end{bmatrix}$$

因变量 \mathbf{Z} 可根据给定的自变量矩阵 \mathbf{X} 和 \mathbf{Y} 由给定的平面方程决定。

这两个矩阵都是 21 行 21 列的，都有 441 个元素，如何快捷地输入呢？这时就可以用到列乘行的乘法运算。可用下面的语句：

```
h=-10:10; lh=length(h)           % 输入一维坐标分割矩阵
X=ones(lh,1)*h,                   % 用全么列乘分割行生成 x 阵
Y=h'*ones(1,lh)                   % 用分割列乘全么行生成 y 阵
```

运行这些语句就可以得出包含 441 个元素的两个自变量矩阵 \mathbf{X} 和 \mathbf{Y} 。如果已给出了平面方程：

$$\mathbf{Z} = \mathbf{a} * \mathbf{X} + \mathbf{b} * \mathbf{Y} + \mathbf{c}$$

由于 MATLAB 具有元素群运算功能，把这两个有 441 个元素的矩阵 \mathbf{X} 和 \mathbf{Y} 代入后，

方程就会给出同阶的包含 441 个元素的 Z 矩阵, 用 `mesh(X,Y,Z)` 就可以画出平面图形了。

这类问题在工程中也经常可以遇到, 要注意利用这里提供的技巧来编程, 可以达到简明、准确、快速的编程效率。

比如计算级数 (项数可变)

$$y = \sin(t) + \sin(3*t)/3 + \sin(5*t)/5 + \sin(7*t)/7 + \sin(9*t)/9 + \dots$$

就可以用这个方法, 读者可以思考如何用最简单的 MATLAB 语句来表达这一算式。

9. 矩阵乘方和幂次:

$$A^2 = A * A, A^n = \underbrace{A * A * \dots * A}_n$$

根据矩阵相乘, 内阶数必须相等的规则, 只有方阵才能乘方和求高次幂。

10. 矩阵指数和级数:

适用于标量的指数函数和某些级数展开式, 也同样适用于矩阵。例如对于方阵 A :

$$(I - A)^{-1} = \frac{I}{I - A} = I + A + A^2 + \dots + A^n + \dots$$

$$e^A = I + A + \frac{A^2}{2!} + \dots + \frac{A^n}{n!} + \dots$$

其中 $n!$ 可用 `prod([1:n])` 表示。不过, 级数的收敛条件将与标量不同, 此处不再深究。

在 MATLAB 中, 一般不需要用户去使用级数, 可以直接调用相关的函数。 $(I - A)^{-1}$ 可以用 `inv(eye(size(A))-A)` 计算, 而 e^A 则用 `expm(A)` 来计算。注意此处不能用 `exp(A)`, 因为 `exp` 函数执行的是元素群运算, 它对矩阵各个元素分别求指数, 只有 `expm(A)` 才把矩阵 A 作为一个整体来求指数。与此类似的函数还有 `sqrtn`, `logm` 等几个。

7.2 初等变换乘子矩阵的生成

现在我们用矩阵乘法来看待行变换。先看下面三个例子:

【例 7.1】 设 $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$, 令 $E_I = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (7.1)

则很容易验算得到

$$E_I \times A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{21} & a_{22} & a_{23} \\ a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (7.2)$$

此项运算的结果是使 A 中的第一行与第二行交换。与 6.2 节中提出的第一种行变换相仿。把左乘改为右乘, 它就成为列交换运算。

$$\mathbf{A} \times \mathbf{E}_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{12} & a_{11} & a_{13} \\ a_{22} & a_{21} & a_{23} \\ a_{32} & a_{31} & a_{33} \end{bmatrix} \quad (7.3)$$

【例 7.2】 令
$$\mathbf{E}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.4)$$

则很容易验算得到

$$\mathbf{E}_2 \times \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 4a_{21} & 4a_{22} & 4a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (7.5)$$

此项运算的结果是使 \mathbf{A} 中第二行的所有元素乘以 4。与 6.2 节中提出的第二种行变换相仿。把左乘改为右乘，它就成为列乘以常数的运算。

$$\mathbf{A} \times \mathbf{E}_2 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & 4a_{12} & a_{13} \\ a_{21} & 4a_{22} & a_{23} \\ a_{31} & 4a_{32} & a_{33} \end{bmatrix} \quad (7.6)$$

【例 7.3】 令
$$\mathbf{E}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \quad (7.7)$$

则很容易验算得到

$$\mathbf{E}_3 \times \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31}+4a_{11} & a_{32}+4a_{12} & a_{33}+4a_{13} \end{bmatrix} \quad (7.8)$$

此项运算的结果是使 \mathbf{A} 中的第三行各元素加上第一行元素乘以 4。这与 6.2 节中提出的第三种行变换相仿。把左乘改为右乘，它就成为第三列乘以 4 与另第一列相加的运算。

$$\mathbf{A} \times \mathbf{E}_3 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11}+4a_{13} & a_{12} & a_{13} \\ a_{21}+4a_{23} & a_{22} & a_{23} \\ a_{31}+4a_{33} & a_{32} & a_{33} \end{bmatrix} \quad (7.9)$$

为了构成任意阶次，任意行和任意常数条件下的这三种初等变换矩阵，就要寻找它的规律，编成子程序。这里提供了以上三个乘子矩阵的生成子程序：

1. 行交换的乘子矩阵 Elgen(n,i,j): 将 i, j 两行交换的 $n \times n$ 乘子矩阵生成子程序，

```
function E=Elgen(n,i,j)
```

```
E=eye(n); E(i,i)=0; E(j,j)=0; E(i,j)=1; E(j,i)=1;
```

2. 乘以 k 的乘子矩阵 $E2gen(n,i,k)$, 将第 i 行乘以 k 的 $n \times n$ 乘子矩阵生成子程序,

```
function E=E2gen(n,i,k)
E=eye(n); E(i,i)=k;
```

3. $E3gen(n,i,j,c)$, 将第 i 行乘以 k 加到第 j 行上的 $n \times n$ 乘子矩阵生成子程序,

```
function E=E3gen(n,i,j,k)
E=eye(n); E(j,i)=k;
```

对这三个子程序, 读者不难自行检验, 并理解其编程的思想, 看看还有更简明的方法吗?

【例 7.4】 要将下列矩阵中的第二行第一个元素消为零, 问应该乘以什么样的矩阵?

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

解: 以第一行第一列元素为枢轴, 在第二行加以第一行乘 $-A(2,1)/A(1,1)=-3$, 就能够消除 $A(2,1)$, 因此所需的乘子矩阵为:

$B = E3gen(A, 1, 2, -3)$, 再输入

$A1 = B * A$, 得到的结果如下

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A1 = \begin{bmatrix} 2 & 6 & 7 & 1 \\ 0 & -16 & -16 & 3 \\ 8 & 4 & 6 & 1 \\ 5 & 8 & 8 & 4 \end{bmatrix}$$

它验证了程序的正确性。初等运算矩阵主要用来说明行运算可以等价为矩阵乘法的概念, 不难看出, $gauss$ 消元子程序相当于用一系列 k 值适当的 E_3 矩阵来乘原矩阵, 过程中有时还需要进行行交换, 那就还要乘以一系列 E_1 矩阵。因此, 整个行阶梯形式的生成过程, 可以看作把原矩阵左乘以一系列的初等变换矩阵 E_1 和 E_3 。把这些初等矩阵的连乘积矩阵写成 E_x

$$U = E_x * A$$

其中 E_x 仍然是 n 阶的方阵。由于 E_1 和 E_3 的行列式为 1, (E_2 的行列式为 k) 所以 E_x 的行列式也是 1, 说明它有逆阵 L 存在, 能使 $L * E_x = I$, 写成 $E_x = L^{-1}$, 于是上式可写成:

$$L * U = A \quad (7.10)$$

这就是说, A 可以分解为一个准下三角矩阵 L 和一个上三角矩阵 U 的乘积。MATLAB 提供了三角分解的函数 `lu`, 它的调用方法是:

```
[L,U]=lu(A)
```

用 `lu` 函数求出的 U 实际上就是 A 的行阶梯形式（不是简化行阶梯形式）。所以，求简化行阶梯形式则应该用 `rref` 函数，而求行阶梯形式可以用 `lu` 函数。不过，它和我们用消元运算所得 U 的数据不一定相同，尽管得出的阶次和阶梯形状相同。但因为行阶梯形式可以有无数种，用不同步骤算出的结果也不同。只有变成简化行阶梯形式，才能进行比较，看它是不是唯一的。

7.3 行列式的定义和计算

按照行列式的不同定义方法，可以有不同的推理方法。

(1) 第一种定义方法：用 n 个元素的连乘积的和来定义。这 n 个元素的第一下标顺序排列，而求和是对第二下标 $p_i(i=1\sim n)$ 的全排列进行，表成算式为：

$$D = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix} = \sum_j^{p_i \text{全排列}} (-1)^{t_j} a_{1p_1} a_{2p_2} \cdots a_{np_n} \quad (7.11)$$

这种排列共有 $n!$ 个，所以求和的下标 j 的取值为 1 到 $n!$ 。其中 t_j 为第 j 个排列的逆序数，即该排列中各个元素的逆序数 t_{j_i} 的和。

$$t_j = t_{j_1} + t_{j_2} + \cdots + t_{j_n} = \sum_{i=1}^n t_{j_i} \quad (7.12)$$

关于计算速度和精度的注记：

可以分析一下按这个定义所需的运算次数，首先看一下式 (7.11) 所需的求和的项数，我们知道 n 个数的全排列数为 $n!$ ，所以式 (7.11) 是 $n!$ 项之和，其中每一项都是 n 个数的乘积，即要做 $(n-1)$ 次乘法，于是不算符号项的总乘法次数为 $(n-1) \times n!$ ，加法次数略去。如果矩阵是 25×25 阶的，那么光乘法的次数用 MATLAB 语句表示就是 `(25-1)*prod(1:25)`，结果为 $3.7227e+026$ 次。假如用每秒能做 10^{12} （1 万亿）次乘法的计算机来计算这个行列式，需要 1200 万年。这里还没有考虑它的符号项。

看它的符号项：对于 n 个数排列中的第 i 项，它的逆序数需要与它后面的 $(n-i)$ 个数进行 $(n-i)$ 次比较来确定。所以，对每一项的符号项需要的比较次数为 $\sum_{i=1}^n (n-i) = \frac{n(n+1)}{2}$ ，总共需要的比较次数就要再乘一个 $n!$ 。它比上面天文数字的乘法次数还要多约 n 倍。可见，定义式 (7.11) 和式 (7.12) 只可用于理论推导和证明，是不宜用来实际计算的。

(2) 用从低阶到高阶的归纳法定义行列式

二元一次方程

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

的解为

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}}, \quad x_2 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{12}a_{21}} \quad (7.13)$$

此解是否存在取决于分母部分 $a_{11}a_{22} - a_{12}a_{21}$ 是否等于零。它仅仅取决于方程左端的系

数阵 $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, 定义它为此系数矩阵的行列式:

$$\det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21} \quad (7.14)$$

请注意行列式等于零的意义。如果 $\det(\mathbf{A})=0$, 便有 $a_{11}/a_{12} = a_{21}/a_{22}$, 即两个方程的左端两系数成同样的比例, 因此两根直线的斜率相等, 如图 6.1 (b)、(c) 所示, 也就是说, 系数行列式等于零, 就判定了方程系数是相依的, 两直线平行。

$$\begin{aligned} & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{aligned}$$

三元一次方程

的解可以通过将系数矩阵化简为行阶梯形式来计算。它的结果是(用印刷体改写其下标后):

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} (a_{12}a_{23}b_3 - a_{12}b_2a_{33} + a_{13}a_{32}b_2 - a_{13}a_{22}b_3 + b_1a_{22}a_{33} - b_1a_{32}a_{23})/D \\ -(a_{11}a_{23}b_3 - a_{11}b_2a_{33} - a_{21}a_{13}b_3 - a_{23}a_{31}b_1 + b_2a_{31}a_{13} + a_{21}b_1a_{33})/D \\ (a_{32}a_{21}b_1 - a_{11}a_{32}b_2 + a_{11}a_{22}b_3 - a_{22}a_{31}b_1 - a_{21}a_{12}b_3 + a_{31}a_{12}b_2)/D \end{bmatrix}$$

其中

$$\begin{aligned} D &= a_{11}a_{22}a_{33} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33} + a_{32}a_{21}a_{13} - a_{22}a_{31}a_{13} + a_{31}a_{12}a_{23} \\ &= a_{11}(a_{22}a_{33} - a_{32}a_{23}) - a_{12}(a_{21}a_{33} - a_{31}a_{23}) + a_{13}(a_{32}a_{21} - a_{22}a_{31}) \end{aligned} \quad (7.15)$$

若 D 等于零, 则方程的解将不存在, 或无穷多(出现于 $x=0/0$ 时), 定义它为系数矩阵的行列式 $D = \det(\mathbf{A})$ 。与二阶的情况相仿, $\det(\mathbf{A})=0$ 将意味着三个方程之间相依, 至少有一个方程的左端系数可以从其他两个方程导出, 因而必有两个平面相互平行。不过现在系数多了, 不是直观所能看出的。三阶行列式的几何意义在第 8 章还要讨论。当然阶次愈高, 人们愈应该从几何概念过渡到代数推理, 不应该停留在几何直观的结果上。从式 (7.15) 不难看出三阶与二阶行列式之间的关系如下:

$$\begin{aligned} \det(\mathbf{A}) &= a_{11} \cdot \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \cdot \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \cdot \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ &= (-1)^2 a_{11} \cdot \det(\mathbf{M}_{11}) + (-1)^3 a_{12} \cdot \det(\mathbf{M}_{12}) + (-1)^4 a_{13} \cdot \det(\mathbf{M}_{13}) \\ &= \sum_{j=1}^3 (-1)^{1+j} a_{1j} \det(\mathbf{M}_{1j}) \end{aligned} \quad (7.16)$$

其中 \mathbf{M}_{ij} , 为在三阶系数矩阵 \mathbf{A} 中去掉第 1 行、第 j 列所得的二阶系数矩阵, 它也称为矩阵 \mathbf{A} 的余子式。把符号部分与此余子式合在一起, 称为代数余子式 \mathbf{A}_{ij} , 其定义为:

$$\det(\mathbf{A}_{ij}) = (-1)^{i+j} a_{ij} \det(\mathbf{M}_{ij}) \quad (7.17)$$

其中 i 不限于取 1, 以便使定义更有普遍意义, 于是有

$$\det(\mathbf{A}) = a_{i1} \det(\mathbf{A}_{i1}) + a_{i2} \det(\mathbf{A}_{i2}) + a_{i3} \det(\mathbf{A}_{i3}) \quad (7.18)$$

不难证明, 这个式子对 $i=1,2,3$ 均成立。

按这个思路, n 阶的行列式可以用由 $(n-1)$ 阶行列式来定义。

$$\det(\mathbf{A}) = a_{i1} \cdot \det(\mathbf{A}_{i1}) + \cdots + a_{in} \cdot \det(\mathbf{A}_{in}) = \sum_{j=1}^n a_{ij} \cdot \det(\mathbf{A}_{ij}) \quad (7.19)$$

式中: \mathbf{A}_{ij} 是 \mathbf{A} 的代数余子式, 它是 $(n-1) \times (n-1)$ 的行列式, 同样也可以按列展开:

$$\det(\mathbf{A}) = a_{1j} \cdot \det(\mathbf{A}_{1j}) + \cdots + a_{nj} \cdot \det(\mathbf{A}_{nj}) = \sum_{i=1}^n a_{ij} \cdot \det(\mathbf{A}_{ij}) \quad (7.20)$$

此式任取 $j=1, 2, \dots, n$ 均成立。根据式 (7.19) 和式 (7.20), 可以看到, 高阶的行列式可以由其低一阶的行列式来定义, 这种方法称为“递归法”。

用递归法定义行列式, 好处之一是避开了烦琐的“全排列”和“逆序数”定义和计算, 好处之二是很自然地得出了行列式按行(或按列)展开的公式。美国教材都用第二种定义方法, 有一定道理。因为他们的书联系实际, 应用讲得多, 图形又多, 篇幅很大。这样讲节省时间, 工科学生比较容易接受。

第二种定义方法并不能提高计算行列式的速度, 因为若从低阶行列式依次向上递增, 需要的运算次数同样是很多的。计算行列式的最好方法还是行阶梯法, 利用 **lu** 分解

$$[\mathbf{L}, \mathbf{U}] = \text{lu}(\mathbf{A})$$

把 \mathbf{A} 分解为一个准下三角矩阵 \mathbf{L} 和一个上三角矩阵 \mathbf{U} 的乘积。因为 $\det(\mathbf{L})=1$, 所以 \mathbf{U} 和 \mathbf{A} 的行列式相等。而三角矩阵 \mathbf{U} 的行列式很好求, 只要把 \mathbf{U} 的主对角线元素连乘就可得到它的行列式。为什么不能用 **rref** 函数求出的行阶梯形式呢? 因为它得到的是简化行阶梯形式, 在化为简化行阶梯形式的过程中, 已经乘了若干个初等变换矩阵 \mathbf{E}_2 , 把其主对角线上的元素变成了 1, 行列式变为 1, 已经不能代表 \mathbf{A} 的行列式特性了。

MATLAB 提供了直接计算矩阵行列式的函数 **det(A)**

关于计算精度和速度的记注:

因为 $n \times n$ 矩阵化简为行阶梯形式所需的乘法次数不到 $n^3/3$, 用这样的方法, 25×25 的矩阵只需要约 5000 次乘法, 比前面所说的方法减少了差不多 10^{23} 倍。用每秒能作 10^{12} (1 万亿) 次乘法的计算机来计算这个行列式, 只需要 5 毫微秒的时间。

【例 7.5】 用化简为三角矩阵的方法求下列矩阵的行列式

$$\mathbf{A} = \begin{bmatrix} 10 & 8 & 6 & 4 & 1 \\ 2 & 5 & 8 & 9 & 4 \\ 6 & 0 & 9 & 9 & 8 \\ 5 & 8 & 7 & 4 & 0 \\ 9 & 4 & 2 & 9 & 1 \end{bmatrix}$$

解: 列出程序:

```
A=[10,8,6,4,1;2,5,8,9,4;6,0,9,9,8;5,8,7,4,0;9,4,2,9,1];
[l,u]=lu(A),           % 分解为上三角矩阵 u 和准下三角矩阵 l
du = diag(u);           % 取出上三角矩阵 u 的主对角线上的元素向量
D=prod(du)              % 求主对角元素的连乘积
```

程序运行的结果如下：

$$I = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0.2000 & -0.7083 & 1.0000 & 0 & 0 \\ 0.6000 & 1.0000 & 0 & 0 & 0 \\ 0.5000 & -0.8333 & 0.8000 & -0.2953 & 1.0000 \\ 0.9000 & 0.6667 & -0.6588 & 1.0000 & 0 \end{bmatrix}$$

$$u = \begin{bmatrix} 10.0000 & 8.0000 & 6.0000 & 4.0000 & 1.0000 \\ 0 & -4.8000 & 5.4000 & 6.6000 & 7.4000 \\ 0 & 0 & 10.6250 & 12.8750 & 9.0417 \\ 0 & 0 & 0 & 9.4824 & 1.1235 \\ 0 & 0 & 0 & 0 & -1.2349 \end{bmatrix}$$

$$du = 10 \quad -4.8 \quad 10.625 \quad 9.4824 \quad -1.2349$$

$$D = 5.9720e+003 = 5972$$

当然直接用 $D=\det(A)$ 也能得到同样的结果。

7.4 矩阵的秩和矩阵求逆

矩阵的秩是矩阵 A 中行列式不等于零的最高阶子式的阶次。是用以衡量联立方程中有效方程数目的指数。按照定义，要计算矩阵的秩需要把矩阵依次分割为各种可能阶次的子矩阵，并求出其行列式，找到它不为零的最高阶次。这个计算过程可能遇到的问题也是子矩阵的数量很大，每个矩阵的行列式计算又非常麻烦，其计算量也将是不可接受的天文数字。其实计算矩阵的秩的最好方法仍然是行阶梯化简，其化简后的非全为零的行数，就是该矩阵的秩。用 MATLAB 函数 $r=\text{rank}(A)$ 可以检验 A 的秩， rank 函数对 A 是否是方阵没有要求，即可以有 $m \neq n$ 。

对于 $n \times n$ 方阵 A ，当 $r=n$ 时，称 A 是满秩的，若 $r < n$ ，必有 $\det(A)=0$ ，称 A 是欠秩的或奇异的。奇异矩阵不可以求逆。

矩阵求逆的最简单方法也是行阶梯化简，其方法是设定一个由 A 和 I 组成的增广矩阵 $C=[A, I]$ ，求 C 的简化行阶梯形式 $UC=\text{rref}([A, I])$ ，得出 $UC=[I, V]$ 。因为可以把 rref 看作一种矩阵左乘以 Q 的乘法运算。 $UC=Q*C=[Q*A, Q*I]=[I, V]$ ，它的左半部为 $Q*A=I$ ，可见 Q 为 A 的逆，故右半部为 $Q*I=Q=V$ ，所以 V 就显示出这个逆阵的内容。

【例 7.6】 设 $A = \begin{bmatrix} 3 & 0 & 3 & -6 \\ 5 & -1 & 1 & -5 \\ -3 & 1 & 4 & -9 \\ 1 & -3 & 4 & -4 \end{bmatrix}$ ，试求其逆阵 $V=A^{-1}$ 。

解：按上述方法写成 MATLAB 程序 ag706。

```
A=[3,0,3,-6;5,-1,1,-5;-3,1,4,-9;1,-3,4,-4];
C=[A,eye(4)]
U0C=rref(C)
V=U0C(:,5:8)
```

运行结果为

$$C = \begin{bmatrix} 3 & 0 & 3 & -6 & 1 & 0 & 0 & 0 \\ 5 & -1 & 1 & -5 & 0 & 1 & 0 & 0 \\ -3 & 1 & 4 & -9 & 0 & 0 & 1 & 0 \\ 1 & -3 & 4 & -4 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U0C = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0.2323 & -0.0101 & -0.1313 & -0.0404 \\ 0 & 1.0000 & 0 & 0 & 0.5354 & -0.3131 & -0.0707 & -0.2525 \\ 0 & 0 & 1.0000 & 0 & 0.5859 & -0.4747 & -0.1717 & 0.1010 \\ 0 & 0 & 0 & 1.0000 & 0.2424 & -0.2424 & -0.1515 & 0.0303 \end{bmatrix}$$

其右边的四列就是逆阵 V

$$V = \begin{bmatrix} 0.2323 & -0.0101 & -0.1313 & -0.0404 \\ 0.5354 & -0.3131 & -0.0707 & -0.2525 \\ 0.5859 & -0.4747 & -0.1717 & 0.1010 \\ 0.2424 & -0.2424 & -0.1515 & 0.0303 \end{bmatrix}$$

这个结果也可以用 $V = \text{inv}(A)$ 直接得到。用 $A * V = I$ 可以验证 V 的正确性。如果 A 不是满秩的，则逆矩阵将不存在。可以从下面的例子看到。

【例 7.7】 设 $A = \begin{bmatrix} -16 & -4 & -6 \\ 15 & -3 & 9 \\ 18 & 0 & 9 \end{bmatrix}$ ，试求其逆阵 $V = A^{-1}$ 。

解：输入 A 的数据后，输入 $V = \text{inv}(A)$ ，程序 ag707 为：

```
A=[-16,-4,-6;15,-3,9;18,0,9], V=inv(A)
```

运行后得到警告信息：

```
Warning: Matrix is close to singular or badly scaled
Results may be inaccurate. RCOND = 6.042030e-018.
```

$$V = 1.0e+15 \begin{bmatrix} 0.4222 & -0.5629 & 0.8444 \\ -0.4222 & 0.5629 & -0.8444 \\ -0.8444 & 1.1259 & -1.6888 \end{bmatrix}$$

它警告说：“此矩阵接近奇异，数据尺度很差，结果可能不准确。逆条件数=6.042030e-018。”

在用数值方法计算矩阵的逆时，由于计算中不可避免地存在方法误差和截断误差，人们不大可能得到理想的零。在消元中也不大可能出现理想的全零行，所以矩阵是否奇异，并不是那么绝对的，往往没有明确的边界。为了评价矩阵接近“奇异”的程度，采用了“条件数”（Condition Number）作为常用的衡量指标。有关条件数的定义此处不作介绍，它永远大于等于 1。条件数愈大，求逆的误差愈大；其数值愈小，计算精度愈高。MATLAB 中，条件数用 $\text{cond}(\mathbf{A})$ 计算。通常人们只大致看其 10 的幂次项，若 $\text{cond}(\mathbf{A})=10k$ ，求逆的有效数位就下降 k 位。MATLAB 的有效数位是 15 位。像现在， k 达到了 18（条件数是逆条件数 RCOND 的倒数），结果连一个有效数位都没有，是根本不可信的。

实际情况是矩阵 \mathbf{A} 的行列式 $\det(\mathbf{A})=0$ ，它的秩为 2，所以它的逆不存在。计算出来的 \mathbf{V} 是不对的，无效的。

7.5 用矩阵“除法”解线性方程

如果方程数与未知数的数目相等， $m=n$ ，则线性代数方程

$$\mathbf{A}\mathbf{x}=\mathbf{b} \quad (7.21)$$

中的系数矩阵 \mathbf{A} 是方阵，设 $\det(\mathbf{A})\neq 0$ ，则它的逆阵 \mathbf{A}^{-1} 存在。MATLAB 中求逆阵的函数为 $\text{inv}(\mathbf{A})$ 。即有

$$\mathbf{A}^{-1}=\text{inv}(\mathbf{A})$$

将上式左右同时左乘以 \mathbf{A}^{-1} ，由于 $\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$ ，得到

$$\mathbf{x}=\mathbf{A}^{-1}\mathbf{A}\mathbf{x}=\mathbf{A}^{-1}\mathbf{b} \quad (7.22)$$

用 MATLAB 语法表示，应为

$$\mathbf{x}=\text{inv}(\mathbf{A})*\mathbf{b} \quad (7.23)$$

本来矩阵理论中没有除法，MATLAB 还创立了矩阵除法的概念，它是从矩阵求逆引申出来的。因为对 \mathbf{A} 求逆就相当于将 \mathbf{A} 放到分母上去，所以从形式上，可以把上式写成

$$\mathbf{x}=\mathbf{A}\backslash\mathbf{b} \quad (7.24)$$

“ \backslash ”就称为左除，因为此处 $\text{inv}(\mathbf{A})$ 是乘在 \mathbf{b} 的左方。而矩阵运算不符合交换律，除号也必须放在左边。当在运算中出现必须把 $\text{inv}(\mathbf{A})$ 乘到矩阵的右方时，其运算符也应该不同，MATLAB 中用 “ $/\mathbf{A}$ ” 表示，并称为右除。

实际上，左除 “ \backslash ” 的功能远远超过了矩阵求逆函数 inv ， $\text{inv}(\mathbf{A})$ 函数要求 \mathbf{A} 必须是方阵，所以式 (7.23) 只能用来解“适定”方程，而式 (7.24) 并不要求 \mathbf{A} 为方阵，在 \mathbf{A} 是 $m\times n$ 阶且 $m<n$ （欠定）时，它只要求 \mathbf{A} 与 \mathbf{b} 的行数相等且 \mathbf{A} 的秩为 m 。所以式 (7.24) 也可以用来解欠定方程，在下例中可以看出。此外，运算符 “ \backslash ” 还能用来解超定方程，这点我们将在第 8 章讨论。

【例 7.8】 用矩阵算法解例 6.7 所给的线性代数方程。

解：先将该例的原始数据输入

$$\mathbf{A}=[3,-4,3,2,-1;0,-6,0,-3,-3;4,-3,4,2,-2;1,1,1,0,-1;-2,6,-2,1,3];$$

```
b = [2; -3; 2; 0; 1];
```

然后，直接用左除的方法求解：输入 $x=A \backslash b$ ，得到的结果是 Inf，并且出现屏幕警告

```
Warning: Matrix is singular to working precision.
```

即“对于当前工作精度，矩阵是奇异的”。实际上从该例前面的分析中，我们已经知道矩阵 A 的秩为 $r=3$ ，而现在 $m=5$ ，故 $\det(A)=0$ 。左除要求的是系数矩阵的行数与秩相同，也就是要剔除相依方程，取出真正有效的方程组，因此只能先求矩阵的秩 r 及其行阶梯形式，求有效行的 $U0$ 和 d 。用 $U0$ 左除 d 来解。于是程序为：

```
B=[A,b], r=rank(B), [UB,ip]=rref(B);
U0=UB(1:r,1:5); d=UB(1:3,6); x=U0\d
```

得到的结果是：

$$U0 = \begin{bmatrix} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad d = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

及

$$x = \begin{bmatrix} 0 \\ 0 \\ 1.0000 \\ 0 \\ 1.0000 \end{bmatrix}$$

表面上看，这个结果好像与例 8 中的特解 $x_1=x_2=x_3=x_5=0, x_4=-1$ 和通解 $x_3=c_1, x_5=c_2, x_1=-c_1+c_2, x_2=0, x_4=-c_2+1$ 都没有共同之处，实际上，如果在通解中设 $c_1=c_2=1$ ，就会得到本例所得的这一个解。可见它确实是一个特解。

7.6 应用实例

7.6.1 网络的矩阵分割和连接

在电路设计中，经常要把复杂的电路分割为局部电路，每一个电路都用一个网络“黑盒子”来表示。“黑盒子”的输入为 u_1, i_1 ，输出为 u_2, i_2 ，其输入输出关系用矩阵 A 来表示（如图 7.1 所示）：

$$\begin{bmatrix} u_2 \\ i_2 \end{bmatrix} = A \begin{bmatrix} u_1 \\ i_1 \end{bmatrix}$$

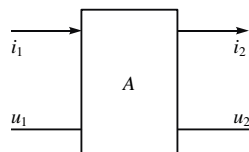


图 7.1 单个子网络模型

A 是 2×2 矩阵, 称为该局部电路的传输矩阵。

把复杂的电路分成许多串接局部电路, 分别求出它们的传输矩阵, 再相乘起来, 得到总的传输矩阵, 可以使分析电路的工作简化。

以图 7.2 为例, 把两个电阻组成的分压电路分成两个串接的子网络。第一个子网络包含电阻 R_1 , 第二个子网络包含电阻 R_2 , 列出第一个子网络的电路方程为:

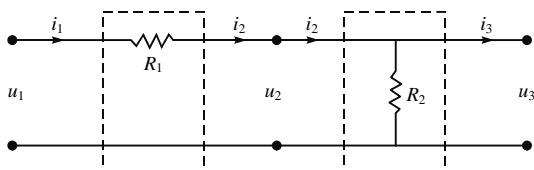


图 7.2 两个子网络串联模型

$$i_2 = i_1, \quad u_2 = u_1 - i_1 R_1$$

写成矩阵方程为:

$$\begin{bmatrix} u_2 \\ i_2 \end{bmatrix} = \begin{bmatrix} 1 & -R_1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ i_1 \end{bmatrix} = A_1 \begin{bmatrix} u_1 \\ i_1 \end{bmatrix}$$

同样可列出第二个子网络的电路方程,

$$i_3 = i_2 - u_2 / R_2, \quad u_3 = u_2$$

写成矩阵方程为:

$$\begin{bmatrix} u_3 \\ i_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1/R_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_2 \\ i_2 \end{bmatrix} = A_2 \begin{bmatrix} u_2 \\ i_2 \end{bmatrix} = A_2 A_1 \begin{bmatrix} u_1 \\ i_1 \end{bmatrix}$$

从上面可分别得到两个子网络的传输矩阵

$$A_1 = \begin{bmatrix} 1 & -R_1 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 0 \\ -1/R_2 & 1 \end{bmatrix}$$

整个电路的传输矩阵为两者的乘积

$$A = A_2 A_1 = \begin{bmatrix} 1 & 0 \\ -1/R_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -R_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -R_1 \\ -1/R_2 & 1 + R_1/R_2 \end{bmatrix}$$

实用中通常对比较复杂的网络进行分段, 对于这样简单的电路是不分段的, 这里只是一个示例。

7.6.2 用逆阵进行保密编译码

在英文中有一种对消息进行保密的措施, 就是把消息中的英文字母用一个整数来表示。然后传送这组整数。例如 “SEND MONEY” 这九个字母就用下面九个数来表示;

[5,8,10,21,7,2,10,8,3]。显然 5 代表 S, 8 代表 E, ……这种方法是很容易被破译的。在一个很长的消息中, 根据数字出现的频率, 往往可以大体估计出它所代表的字母。例如出现频率特别高的数字, 很可能对应于出现频率最高的字母 E。

我们可以用矩阵乘法来对这个消息进一步加密。假如 A 是一个行列式等于 ± 1 的整数矩阵, 则 A^{-1} 的元素也必定是整数。可以用这样的—个矩阵来对消息进行变换。而经过这样变换过的消息就较难破译。为了说明问题, 设

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 3 \\ 2 & 3 & 2 \end{bmatrix}, \quad \text{可得 } A^{-1} = \begin{bmatrix} 1 & -1 & 1 \\ 2 & 0 & -1 \\ -4 & 1 & 1 \end{bmatrix}$$

把编了码的消息也组成一个矩阵

$$B = \begin{bmatrix} 5 & 21 & 10 \\ 8 & 7 & 8 \\ 10 & 2 & 3 \end{bmatrix}$$

$$\text{乘积} \quad AB = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 3 \\ 2 & 3 & 2 \end{bmatrix} \begin{bmatrix} 5 & 21 & 10 \\ 8 & 7 & 8 \\ 10 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 31 & 37 & 29 \\ 80 & 83 & 69 \\ 54 & 67 & 50 \end{bmatrix}$$

所以发出的消息为[31,80,54,37,83,67,29,69,50]。注意原来两个相同的数字 8 和 10, 在变换后成为不同的数字, 所以就难于按其出现的频度来破译了。而接收方只要将这个消息乘以 A^{-1} 就可以恢复原来的消息。

$$\begin{bmatrix} 1 & -1 & 1 \\ 2 & 0 & -1 \\ -4 & 1 & 1 \end{bmatrix} \begin{bmatrix} 31 & 37 & 29 \\ 80 & 83 & 69 \\ 54 & 67 & 50 \end{bmatrix} = \begin{bmatrix} 5 & 21 & 10 \\ 8 & 7 & 8 \\ 10 & 2 & 3 \end{bmatrix}$$

具有整数元素且其行列式等于 ± 1 的编码矩阵的生成方法如下: 它从单位矩阵出发, 反复运用第 1 类和第 3 类初等变换矩阵去乘它, 而其中的乘数必须取整数。这样得出的矩阵将满足

$$\det(A) = \pm \det(I) = \pm 1$$

而 A^{-1} 也将具有整数元素。

7.6.3 减肥配方的实现

设三种食物每 100 克中蛋白质、碳水化合物和脂肪的含量如下表, 表中还给出了 20 世纪 80 年代美国流行的剑桥大学医学院的简洁营养处方。现在的问题是: 如果用这三种食物作为每天的主要食物, 那么它们的用量应各取多少才能全面准确地实现这个营养要求?

营养	每 100g 食物所含营养 (g)			减肥所要求的每日营养量
	脱脂牛奶	大豆面粉	乳清	
蛋白质	36	51	13	33
碳水化合物	52	34	74	45
脂肪	0	7	1.1	3

设脱脂牛奶的用量为 x_1 个单位 (100g), 大豆面粉的用量为 x_2 个单位, 乳清的用量为 x_3 个单位, 表中的三个营养成分列向量为:

$$\mathbf{a}_1 = \begin{bmatrix} 36 \\ 52 \\ 0 \end{bmatrix}, \quad \mathbf{a}_2 = \begin{bmatrix} 51 \\ 34 \\ 7 \end{bmatrix}, \quad \mathbf{a}_3 = \begin{bmatrix} 13 \\ 74 \\ 1.1 \end{bmatrix},$$

则它们的组合所具有的营养为

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + x_3 \mathbf{a}_3 = x_1 \begin{bmatrix} 36 \\ 52 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 51 \\ 34 \\ 7 \end{bmatrix} + x_3 \begin{bmatrix} 13 \\ 74 \\ 1.1 \end{bmatrix}$$

使这个合成的营养与剑桥配方的要求相等, 就可以得到以下的矩阵方程:

$$\begin{bmatrix} 36 & 51 & 13 \\ 52 & 34 & 74 \\ 0 & 7 & 1.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 33 \\ 45 \\ 3 \end{bmatrix} \Rightarrow \mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

用 MATLAB 解这个问题非常方便, 列出程序 ag763 如下:

```
A=[36,51,13;52,34,74;0,7,1.1]
```

```
b=[33;45;3]
```

```
x=A\b
```

程序执行的结果为:

$$\mathbf{x} = \begin{bmatrix} 0.2772 \\ 0.3919 \\ 0.2332 \end{bmatrix}$$

即脱脂牛奶的用量为 27.7g, 大豆面粉的用量为 39.2g, 乳清的用量为 23.3g, 就能保证所需的综合营养量。

7.6.4 弹性梁的柔度矩阵

设简支梁如图 7.3 所示, 在梁的三个位置分别施加力 f_1 , f_2 和 f_3 后, 在该处产生的综

合变形为图示的 y_1 , y_2 和 y_3 , 通常称为挠度。根据胡克定律, 在材料未失去弹性的范围内, 力与它引起的变形呈线性关系, 可以写出:

$$y=Df$$

写出完全的矩阵形式为:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

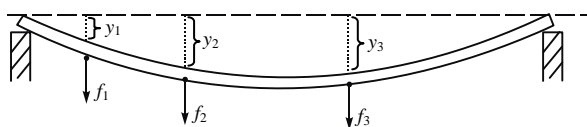


图 7.3 简支梁在三个点的力和变形(挠度)

不难从这个矩阵乘法关系中得知矩阵 D 的物理意义。假如只施加一个力 f_1 , 其余两个力 f_2 和 f_3 为零, 则引起的挠度为: $y_1=d_{11}*f_1$, $y_2=d_{21}*f_1$, $y_3=d_{31}*f_1$, 如果施加的力为一个单位, 则在 1, 2, 3 三个位置引起的挠度分别为 d_{11} , d_{21} 和 d_{31} 。可以用同样的方法来理解其他的 d , 利用这个概念, 可以用实测的方法来得到矩阵 D 中的各个元素。这些挠度元素愈大, 表明这个梁愈柔软, 所以矩阵 D 被称作柔度矩阵。

柔度矩阵的逆就是刚度矩阵 K , $K=D^{-1}$, 在本例中它也是一个 3×3 矩阵。

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

其物理意义为造成单位挠度所需要的力。这些力愈大, 表明这个梁愈刚劲。

在图 7.3 所示的情况下, 柔度矩阵的 9 个元素中, 没有一个可能为零。这说明它们之间相互耦合。在位置 1 施加力不但会引起该处的变形, 同时也必然引起位置 2 和 3 处的变形; 在其他位置施加力亦然。如果矩阵变为对角矩阵, 那也就意味着互相独立。位置 1 施加的力只引起位置 1 处的挠度, 常识告诉我们, 在图 7.3 所示的简支梁上是不可能出现这种情况的。

现在举一个数字的例子, 设柔度矩阵 $D = \begin{bmatrix} 0.005 & 0.002 & 0.001 \\ 0.002 & 0.004 & 0.003 \\ 0.001 & 0.003 & 0.006 \end{bmatrix}$, 单位为【公分/牛顿】。

(1) 设在位置 1, 2, 3 处施加的力为 30, 50 和 20 【牛顿】, 试求出其挠度。

(2) 设要在位置 3 处产生 0.4 【公分】的挠度, 其他两处挠度为零, 试求应施加的力。

解: 列出解此题的 MATLAB 程序 ag764

```
D=0.001*[5,2,1;2,4,3;1,3,6]    % 输入柔度矩阵
f=[30;50;20]                    % 给定力, 求挠度
y=D*f                            % 解题 (1)
y1=[0;0;0.4]                    % 给定挠度, 求力
```

```
K=inv(D) % 求刚度矩阵
f1=K*y1 % 解题 (2)
```

程序运行的结果如下:

$$\text{题 (1) 的三个挠度为 } \mathbf{y} = \begin{bmatrix} 0.2700 \\ 0.3200 \\ 0.3000 \end{bmatrix}$$

$$\text{刚度矩阵为 } \mathbf{K}=\text{inv}(\mathbf{D}) = \begin{bmatrix} 254.2373 & -152.5424 & 33.8983 \\ -152.5424 & 491.5254 & -220.3390 \\ 33.8983 & -220.3390 & 271.1864 \end{bmatrix}$$

$$\text{题 (2) 的三个力为 } \mathbf{f}_1 = \begin{bmatrix} 13.5593 \\ -88.1356 \\ 108.4746 \end{bmatrix}$$

最后的力中有负数是合理的, 因为题目要求有两个位置上的挠度为零, 如果三个力都同向, 造成的变形也同向, 那就没有一处的挠度会等于零, 必须有反向施加的力才行。

可以看到, 在刚度矩阵中也出现了负值元素, 这是什么原因呢? 可从其物理意义去想。对照柔度矩阵的物理意义, 刚度矩阵中的第一列元素应该是: 使挠度 y_1 达到一个单位, 而 y_2 和 y_3 都等于零时, 在三个位置上应施加的力。即同时在位置 1 施加 k_{11} 的力, 在位置 2 施加 k_{21} 的力, 而在位置 3 施加 k_{31} 的力。正如上面的分析, 要使两个位置上的挠度为零, 这三个力必须有正有负。所以表现为刚度矩阵中每列元素中必定有负值元素。只在一个位置加力 (意味着其他两个力为零), 在三个位置引起挠度是非常直观, 符合常识的, 所以读者容易接受柔度矩阵的物理概念; 只在一个位置加“挠度” (意味着其他两个挠度为零), 在三个位置需要加多大力却需要想象, 在常识中大概很少有人经历这种状况, 所以刚度矩阵的物理概念就要难理解一些。

7.6.5 网络和图

图论是应用数学的一个重要分支, 它非常广泛地用于几乎所有的应用科学中, 可以形象地帮助建模。在通信网络中特别有用。

设图上有 n 个顶点 V_1, V_2, \dots, V_n , 在每两个顶点之间以线段相连接, 这些线段可以有

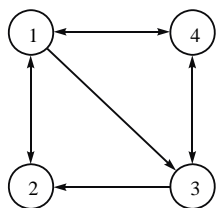


图 7.4 四个城市间航线图

向的, 用箭头表示; 也可以是双向的 (即无向的) 因而可画出双向箭头, 如果图上线段都是双向的, 也可不画箭头。这样的图, 可以用 $n \times n$ 矩阵表示, 行和列分别表示这些顶点的编号。行号表示出发节点, 列号表示到达节点, 任何一根有向线段, 在矩阵中用一个数值为 1 的元素表示。如果是双向的线段, 在矩阵中就得用两个位置转置、数值为 1 的元素表示。

例如, 图 7.4 为 1, 2, 3, 4 四个城市之间的空运航线, 用

有向图表示。则该图可以用下列航路矩阵表示：

$$A_I = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

其中第一行为由第一个城市出发的航班，分别可以到达第三、第四两城市，因此在第三、第四两列处的元素为 1，其余为零。依此类推可以写出其他各行的元素，构成矩阵 A ，称为邻接矩阵。注意图中 1, 4 两城市间有双向航线，因此在矩阵中的 $A(1, 4)$ 和 $A(4, 1)$ 两个转置位置的元素均为 1。对有向线段，其 1 元素位置就没有转置的关系。将此矩阵用 MATLAB 语句输入：

$$A1 = [0,0,1,1; 1,0,0,0; 0,1,0,0; 1,0,1,0]$$

如果我们要分析经过一次转机（也就是坐两次航班）能到达的城市，可以由邻接矩阵的平方 $A_2 = A_I * A_I$ 来求得。实际意义就是把第一次航班的到站再作为起点，求下一个航班的终点。

$$A2 = A1 * A1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

经过两次以内转机能够到达的航路矩阵应为：

$$A = A1 + A2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 \end{bmatrix}$$

在航路矩阵中出现两个数值为 2 的元素，意味着有两条不同的航路可以从城市 1 到达城市 3，以及从城市 4 到达城市 3。

求两次转机（三个航班）的可达矩阵可计算如下：

$$A3 = A1^3,$$

$$A = A1 + A1^2 + A1^3$$

结果为

$$A = \begin{bmatrix} 2 & 2 & 3 & 2 \\ 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 3 & 1 \end{bmatrix}$$

矩阵元素中数字 2, 3 的意义不言自明。依次类推，可以求多次转机时的航路矩阵。

7.7 习题

7.1 构建一个 4×4 的随机正整数矩阵 A ，取三次不同的 A ，检验下式是否满足：

$$(A+I)(A-I)=A^2-I$$

最好的方法是让左端减去右端而检验其差为零矩阵。然后检验下式是否满足：

$$(A+B)(A-B)=A^2-B^2$$

其中 B 也是一个 4×4 的随机正整数矩阵。报告所得结果。

7.2 设 $A = \begin{bmatrix} 1/6 & 1/2 & 1/3 \\ 1/2 & 1/4 & 1/4 \\ 1/3 & 1/4 & 5/12 \end{bmatrix}$ ，计算 $B_1=A^{10}$ ， $B_2=A^{20}$ ， $B_3=A^{30}$ ，分别计算这三个矩阵的秩，对结

果进行说明。（提示，用长格式（format long）显示矩阵 B_1, B_2, B_3 ）。

7.3 设 $B=[-1,-1;1,1]$ ， $A=[\text{zeros}(2), \text{eye}(2); \text{eye}(2), B]$ ，注意 $B^2=O$ 。

(a) 用 MATLAB 计算 A^2, A^4, A^6, A^8 。猜想 A^{2k} 会具备何种形式（以子矩阵 I, O 和 B 表示），用数学归纳法来证明对所有的正整数 k ，这个猜想是正确的。

(b) 用 MATLAB 计算 A^3, A^5, A^7, A^9 。猜想 A^{2k-1} 会具备何种形式（以子矩阵 I, O 和 B 表示），证明这个猜想。

7.4 (a) 下列 MATLAB 命令：

```
A=round(10*rand(6)), B=A'+A
```

将生成一个具有整数元素的对称矩阵，解释其原理并用实验验证之。然后用 MATLAB 语句将 B 分块为四个 3×3 矩阵 $B_{11}, B_{12}, B_{21}, B_{22}$ 。

(b) 设 $C=\text{inv}(B_{11})$ ，此时必有 $C^T=C$ 及 $B_{21}^T=B_{12}$ ，说明其原因。用 MATLAB 验证这个结论。然后设

```
E=B21*C, F=B22-B21*C*B21'
```

用 MATLAB 函数构成： $L = \begin{bmatrix} I & O \\ E & I \end{bmatrix}$ ，及 $D = \begin{bmatrix} B_{11} & O \\ O & F \end{bmatrix}$ ，

计算 $H=L*D*L'$ ，通过计算 $H-B$ 把 H 与 B 进行比较。证明如果用精确的算法，必有 $LDL^T=B$ 。

7.5 用下列语句

```
A=round(10*rand(5)), B=round(20*rand(5))
```

生成两个 5×5 的随机矩阵，用 MATLAB 计算下列数对，说明哪些情况下两者相等。

- | | |
|-----------------------------------|--|
| (a) $\det(A)$ 和 $\det(A^T)$ | (b) $\det(A+B)$ 和 $\det(A)+\det(B)$ |
| (c) $\det(AB)$ 和 $\det(A)\det(B)$ | (d) $\det(A^T B^T)$ 和 $\det(A^T)\det(B^T)$ |
| (e) $\det(A^{-1})$ 和 $1/\det(A)$ | (f) $\det(AB^{-1})$ 和 $\det(A)/\det(B)$ |

7.6 用 MATLAB 证明，任意 $n \times n$ 矩阵中若有两行元素相同，该矩阵的行列式等于零。（提示：对于 $n=3, 4, 5, 6$ ，用下列语句设定有两行相同的 $n \times n$ 矩阵。

```
A=randintr(n),
A(2,:)=A(1,:)
```

然后求其行列式验证，并作出解释。

- 7.7 (a) 求描述如图 7.5 所示的交通流量图的方程组并求其解。

(b) 如果 x_4 的路段被封闭，求此方程组的解。

(c) x_1 的最大值是多少？

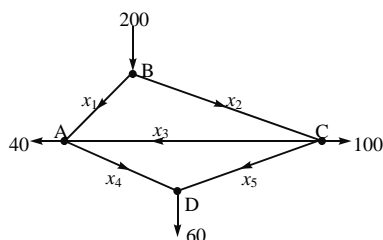


图 7.5 交通流量图

- 7.8 满足 $\mathbf{A}=\mathbf{A}^T$ 的矩阵称为对称矩阵，显然它必定是方阵。可以用 MATLAB 构建随机对称矩阵 \mathbf{A} 如下：

```
B=randintr(n); A=B+B'
```

(a) 证明上述方法产生的确实是对称矩阵。

(b) 生成若干个 4×4 对称矩阵 \mathbf{A} ，观察它的列与行之间有什么关系，以便快速地判定它的对称性。

(c) 用实际计算数据检验下列说法的“真”或“伪”：

(i) 若 \mathbf{A} 和 \mathbf{B} 是对称的，则 $\mathbf{A}+\mathbf{B}$ 也是对称的；

(ii) 若 \mathbf{A} 和 \mathbf{B} 是对称的，则 $\mathbf{A}*\mathbf{B}$ 也是对称的；

(iii) 若 \mathbf{A} 是对称且可求逆的，则 \mathbf{A}^{-1} 也是对称的。

- 7.9 一幢大型公寓楼可以有三种安排各层建筑结构类型。类型甲可以在一层上安排 18 个单元：3 个三室一厅、7 个两室一厅和 8 个一室一厅；类型乙可以在一层上安排 16 个单元：4 个三室一厅、4 个两室一厅和 8 个一室一厅；类型丙可以在一层上安排 17 个单元：5 个三室一厅、3 个两室一厅和 9 个一室一厅；现在要求整个公寓楼恰好有 66 个三室一厅、74 个两室一厅和 136 个一室一厅，问应该怎样选择各类型的层数？

- 7.10 (a) 计算图 7.6 所示三级网络的传输函数；

(b) 设 $\mathbf{A}=\begin{bmatrix} 4/3 & -12 \\ -1/4 & 3 \end{bmatrix}$ ，设计一个三级梯形网络，使它的合成传输函数等于 \mathbf{A} 。

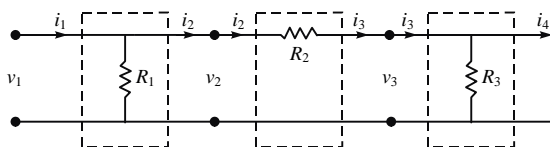


图 7.6 三级网络的串接

- 7.11 考虑图 7.7 的无向图

(a) 求出此图的邻接矩阵 \mathbf{A} ，并将其输入 MATLAB。

(b) 计算 \mathbf{A}^2 并确定用两步从 (i) V_1 到 V_7 ，(ii) V_4 到 V_8 ，(iii) V_5 到 V_6 ，(iv) V_8 到 V_3 的路线数。

(c) 计算 \mathbf{A}^4 ， \mathbf{A}^6 ， \mathbf{A}^8 并确定用 4，6，8 步从 (b) 中所列出的起点到终点的路线数。

(d) 如果在图 7.7 上增加 $\{V_3, V_6\}$ ， $\{V_5, V_8\}$ 两条边，则新的邻接矩阵 \mathbf{B} 可以生成如下：

```
B=A; B(3,6)=1; B(6,3)=1;
```

```
B(5,8)=1; B(8,5)=1
```

用 \mathbf{B} 计算 (b) 中的结果。

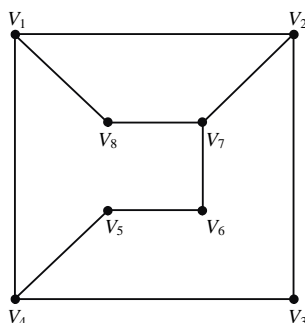


图 7.7 无向图

7.12 铁板中稳态热传导的简单模型已在前面分析过，如果铁板的长度增加，中间取的点数也要增加，图 7.8 表示取 8 个点的情况。按同样的方法列写方程 $Ax=b$ ，可得到下面的 8×8 系数矩阵 A 和

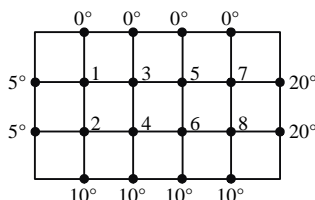


图 7.8 铁板热传导模型图

$$A = \begin{bmatrix} 4 & -1 & -1 & & & \\ -1 & 4 & 0 & -1 & & \\ -1 & 0 & 4 & -1 & -1 & \\ & -1 & -1 & 4 & 0 & -1 \\ & & -1 & 0 & 4 & -1 & -1 \\ & & & -1 & -1 & 4 & 0 & -1 \\ & & & & -1 & 0 & 4 & -1 \\ & & & & & -1 & -1 & 4 \end{bmatrix}$$

$$b = [5; 15; 0; 10; 0; 10; 20; 30]$$

A 中未标注的元素等于零。

(a) 把矩阵 A 进行 lu 分解，用 $L*U=A$ 检验结果的正确性。

(b) 求 A^{-1} ，并用它解此方程。

7.13 按图 7.7 所示，设此梁在四个点受力，在这四个点的柔度矩阵为：

$$D = 0.001 * \begin{bmatrix} 4 & 3 & 1 & 0.5 \\ 3 & 5 & 3 & 1 \\ 1 & 3 & 5 & 3 \\ 0.5 & 1 & 3 & 4 \end{bmatrix}$$

其单位为【公分/牛顿】。

(a) 今测得四个点的挠度分别为， $y_1=0.08$ ， $y_2=0.12$ ， $y_3=0.16$ ， $y_4=0.12$ 【公分】，求加四个力。

(b) 如果要使的第二点的挠度增加 0.24【公分】，其他点挠度不变，问应该在各点额外加多少力【牛顿】？

用向量空间解线性方程组

8.1 向量和向量空间

线性系统的许多重要特性可以用向量的概念来描述,这是分析线性系统的另一个视点。如果说用联立线性方程组的概念来讨论方程

$$Ax = b$$

的解 x 时,其重点是放在已知系数矩阵 A 和 b 讨论解 x 的特点和计算方法的话,那么用向量方法对这个方程的研究着重点就放在系数矩阵 A 的结构,特别是其列向量的结构特点和要求方面。这种方法的好处还在于有鲜明的几何意义,从中可进一步引申出向量空间的概念。

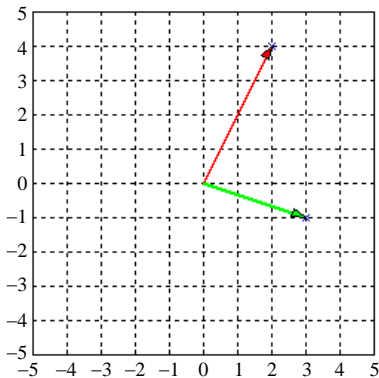


图 8.1 二维空间（平面）中的向量

二维和三维空间中的向量有鲜明的几何意义,掌握它们的基本特性就不难去抽象推想高维的向量。向量有行向量和列向量之分,在研究向量空间时,通常取列向量。

1. 二维空间 R^2 中的向量用两个沿列向顺序排列的元素表示

【例 8.1】 设
$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \quad (8.1)$$

画出此两个向量的图形。

解: \mathbf{u} 和 \mathbf{v} 都是二维空间的列向量。可以用平面坐标系中的两个点, 或从坐标原点引向这两点的箭头来表示。用 MATLAB 可表示为程序 ag801:

```
u=[2;4]; v=[3;-1];
plot([2,3],[4,-1],'x');hold on      % 用 x 号画出两个点
% 若装有 ATLAST 中的子程序 drawvec, 可画向量如下
drawvec(u);hold on                  % 画出向量 u
drawvec(v,'g');hold off             % 画出向量 v
grid on
```

产生的图形见图 8.1。设平面上的任意向量 $\mathbf{w}=1.5\mathbf{u}+2\mathbf{v}$, 则可求得

$$\mathbf{w} = 1.5 \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix} + 2 \cdot \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 9 \\ 4 \end{bmatrix} \quad (8.2)$$

可见, \mathbf{u} 和 \mathbf{v} 经过乘法和加法运算的合成向量 \mathbf{w} 仍然在原来的二维空间之内。这个向量的加法在几何上等效于平行四边形加法, 请读者自行验证。向量经过加法和乘法仍在原 R^2 空间内的特性称为对加法和乘法是封闭的, 合成向量 \mathbf{w} 的集合也称为 \mathbf{u} 和 \mathbf{v} 张成的线性空间。

可以提出一个反问题, 平面上的任何一点 $[w_1; w_2]$ 是不是一定能用 \mathbf{u} 和 \mathbf{v} 的线性组合来实现? 即是不是一定能找到一组常数 $[c_1, c_2]$, 使得

$$c_1 \begin{bmatrix} 2 \\ 4 \end{bmatrix} + c_2 \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad (8.3)$$

要回答这个问题, 只要解这个二元一次联立方程就行了。在本例的 \mathbf{u} 和 \mathbf{v} 下, c_1 和 c_2 是肯定可以求出的, 但并非任何 \mathbf{u} 和 \mathbf{v} 都能达到这个要求。

比如我们将 \mathbf{v} 改为 $\mathbf{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, 此时 \mathbf{u} 和 \mathbf{v} 两个向量的各元素成简单的比例关系, 在几

何上这两个向量是共线的, 那不管如何把它们乘以实系数并相加, 合成的向量只能在一根直线上, 不可能覆盖(张成)整个二维平面。这种情况下, 称这两个向量 \mathbf{u} 和 \mathbf{v} 是线性相关的。在研究向量空间的构成时, 正确地选择基本向量, 防止它们出现线性相关的现象至关重要, 所以要讨论它的一般规律。

2. 三维空间 R^3 中的向量用三个顺序排列的元素表示

例如
$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 5 \\ -4 \\ -7 \end{bmatrix}, \mathbf{v}_3 = \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix}, \quad (8.4)$$

$\mathbf{v}_1, \mathbf{v}_2$ 和 \mathbf{v}_3 都是三维空间的列向量。可以用空间坐标中的三个点, 或从坐标原点引向这三点的箭头来表示。用 MATLAB 则可表示为 $\mathbf{v}_1=[1;-1;-2]$; $\mathbf{v}_2=[5;-4;-7]$; $\mathbf{v}_3=[-3;1;0]$; 也可以用 plot3 命令来绘制三维空间坐标中这些向量端点的位置。

空间向量的线性组合与平面向量相仿, 它们同样服从平行四边形法则, 同时也符合矩阵相加和乘实数的规则。如果三个基本向量之间线性无关, 那么它们的线性组合可以覆盖(张成)整个三维空间。如果它们线性相关, 那么它们的线性组合将只能构成一个平面, 甚至一根直线, 这出现在三个基本向量共面或共线时, 这时, 它们张成的只是三维空间 \mathbf{R}^3 的一个子空间。当然判断三个向量的线性相关性, 就不像二维向量那么简单地用观察法就能做到的了, 判断的方法是利用行列式。把这些向量并排, 构成矩阵形式:

$$\mathbf{A} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] = \begin{bmatrix} 1 & 5 & -3 \\ -1 & -4 & 1 \\ -2 & -7 & 0 \end{bmatrix}, \quad (8.5)$$

如果这三个向量线性相关, 则这个矩阵的行列式就等于零。

其实行列式本身就具有鲜明的几何意义。两个二维向量可构成一个平行四边形, 这个平行四边形的面积就是它的行列式。我们不作一般情况下的证明, 只考虑向量 \mathbf{u} 与横轴平行的情况

$$\mathbf{u} = \begin{bmatrix} u_1 \\ 0 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \quad (8.6)$$

它们的行列式

$$D = \det \begin{pmatrix} u_1 & v_1 \\ 0 & v_2 \end{pmatrix} = u_1 v_2 \quad (8.7)$$

不难看出, u_1 是底, v_2 是高, 它们的乘积的绝对值确实是该平行四边形的面积。当 $D=0$ 时, 面积为零, 说明两个向量是共线的, 也就是线性相关的。这个判据的几何意义如此鲜明, 根本用不着死记硬背。

同样可以证明, 三个三维向量可以组成一个平行六面体, 这三个向量的行列式就等于这个平行六面体的体积, 如果这三个向量共面, 甚至共线, 那么这个六面体就退化为平面或直线, 没有体积可言, 所以其行列式等于零。因此用行列式来判别向量组是否相关, 当然是顺理成章的方法。

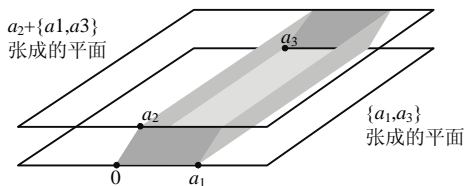


图 8.3 三向量组成体积

注: “张成”是从英文 Span 翻译过来的。它指的是若干个向量 $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ 按线性组合构成的合成向量 \mathbf{w} , $\mathbf{w} = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_n \mathbf{v}_n$ 在所有可能的 c_1, c_2, \dots, c_n 下 \mathbf{w} 的集合。因此几何空间中的两个向量(如果不共线)就将张成一个平面。“组成”则指几个确定的向量按平行四边形法合成一个确定形状。

3. m 维空间 R^m 中的 n 个向量 v_1, v_2, \dots, v_n , 每个向量可用 m 个顺序排列的元素表示

$$v_1 = \begin{bmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{m1} \end{bmatrix}, v_2 = \begin{bmatrix} v_{12} \\ v_{22} \\ \vdots \\ v_{m2} \end{bmatrix}, \dots, v_n = \begin{bmatrix} v_{1n} \\ v_{2n} \\ \vdots \\ v_{mn} \end{bmatrix} \quad (8.8)$$

v_1, v_2, \dots, v_n 都是 m 维空间的列向量, 这时, 就不可能用空间几何的概念来想象了, 需要更多的抽象的代数思维方法。可以把这 m 维空间的 n 个列向量并排, 表示为 $m \times n$ 矩阵 A

$$A = [v_1, v_2, \dots, v_n] = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}, \quad (8.9)$$

可想而知, 这 n 个列向量是否线性相关, 要用它的行列式来判断。不过, 在进入三维以上的空间时, 已经没有可与面积、体积直接相当的概念可用了, 所以采用了秩的概念。当 $m=n$, 且 A 的行列式为零, 也就是它的秩 r 小于 n 时, 说明这 n 个向量是线性相关的。或者换句话说, 当 A 的行列式不为零, 即它的秩 r 等于 n 时, 说明这 n 个向量是线性无关的。秩的概念也概括了面积存在 ($r=2$) 和体积存在 ($r=3$) 的意义, 因此, 它是更高度的抽象。

判断向量相关性的概念和研究线性方程的相依性在原理和方法上都非常相似, 都只需要判别矩阵的秩。差别仅在于方程的相依性是从行向来讨论, 而向量相关性则从列向来探讨。

设矩阵 (8.9) 的秩为 r , 当然有 $r \leq \min(m, n)$ 。如果 $r < n$, 则此 n 个向量线性相关。其中只有 r 个向量是线性无关的, 因此向量组 v_1, v_2, \dots, v_n 只能构成 r 维的子空间。根据同样的道理, 如果 $m < n$, 即向量的数目比空间 R^m 的维数大, 那就无须计算, 可以直接判定, 这 n 个向量是线性相关的, 此问题留给读者思考, 思考时, 要从二维三维起步。

8.2 向量空间和基向量

如果 n 个 m 维向量组的秩为 r , 就是说, 其中只有 r 个向量是线性无关的, 则这 r 个向量的线性组合的全体 V 就构成了 r 维空间。如果 V 不是空集, 并且对加法和数乘两种运算封闭, 则 V 称为向量空间。我们也称 V 为这 r 个向量所张成 (span) 的集或张成的空间 R^r 。它的维数为 r , 是原 m 维空间 R^m 的子空间。生成子空间 R^r 的 r 个线性无关的向量 v 称为基向量或简称基 (Basis), 显然基向量有无数种组合。

当 $r=n$ 时, 给定的 n 个向量就是一组基。如果 $r < n$, 那就要在 n 个向量中选出 r 个线性无关的向量。怎样选? 简单地用秩的概念还无法判定, 因为它不能告诉我们究竟哪些向量是线性无关的, 应该选出来。或哪些向量是线性相关的, 应该去掉。这时就要借助于把矩阵简化为阶梯形式的方法。选择其中的枢轴元素所对应的列, 必定是线性无关的, 因而可以作为子空间 R^r 的基。

【例 8.2】 设由四个五维向量组成的矩阵 A 如下，试求出其所张成的子空间维数，并求出其基向量。

$$A = \begin{bmatrix} 4 & -5 & -4 & -1 \\ 0 & -3 & 0 & 1 \\ -2 & 1 & 2 & 0 \\ -5 & 4 & 5 & 3 \\ -1 & 4 & 1 & -1 \end{bmatrix}$$

解：将 A 输入 MATLAB 工作空间， $A=[4,-5,-4,-1;0,-3,0,1;-2,1,2,0;-5,4,5,3;-1,4,1,-1]$ ，并输入 $U0=rref(A)$ ，可以得到：

$$U0 = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

可见， A 的秩是 3，这四个向量实际上只能在五维空间张成一个三维子空间。可以作为子空间基的是 A 的第 1, 2, 4 三列，因为它们是线性无关的枢轴列，也就是本题所要求的基向量。其实，函数 `rref` 可以直接提供枢轴列的列标，只要再加一个输出变元 i_p 。输入

```
[U0, ip]=rref(A)
```

就可以同时得到上述 U_0 和 $i_p=1,2,4$ 。

在基向量 $v_i (i=1,\dots,r)$ 已知的情况下，常常遇到的一个问题是：任意给出一个 m 维的向量 w ，问它是否处于基向量 v_i 张成的子空间内。对于最简单的平面情况，人们可以用观察法，一眼就看出 w 是否与 v_1, v_2 共面，并且观察如何用平行四边形法则由两个基向量来合成 w 。ATLAST 手册提供了一个游戏子程序 `cogame.m` 就是要测试读者对两个基向量所要乘的系数的猜测能力。三维空间中，为了观察三个向量的空间关系，ATLAST 手册还提供了一个演示程序 `viewsubspaces(u,v,w)`，它用蓝色直线显示向量 u ，同时用红色显示 v 和 w 所张成的平行四边形平面，画在同一个立体图上，以帮助读者建立空间概念。

我们将这个程序演示一下，输入三个三维向量，再输入 `viewsubspaces` 命令：

```
u=[-1;1;8];v=[5;-4;7];w=[-3;1;-5];
viewsubspaces(u,v,w)
```

程序运行的结果产生如图 8.4 所示的图形，三个向量的起点都是 $x=y=z=0$ 的原点。图中的平行四边形是由 v, w 两个向量为边所组成的平行四边形平面， u 向量则是单独以蓝色画出的。可见，在立体图上看出来向量以及它们组成的子空间已经不太容易了，需要有较好的空间概念和观察能力。

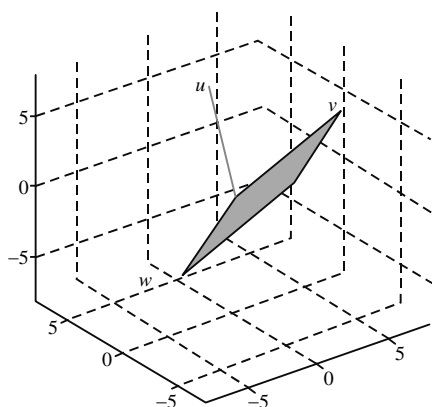


图 8.4 三个向量组成的子空间

上面提出的 w 是否能由 v_1, v_2, v_3 的线性组合构成的问题, 实际上是线性方程组解的存在性问题, 也是 v_1, v_2, v_3 与 w 是否相关的问题。对三维以上的空间, 几何概念就行不通了, 要通过行列式和秩的概念来解决。从下面例题可以说明它的解法。

【例 8.3】 设由三个在 R^4 空间的四维基向量 v_1, v_2, v_3 张成的子空间, 问 w_1 和 w_2 是否在此子空间内。其中:

$$v_1 = \begin{bmatrix} 7 \\ -4 \\ -2 \\ 9 \end{bmatrix}, v_2 = \begin{bmatrix} -4 \\ 5 \\ -1 \\ -7 \end{bmatrix}, v_3 = \begin{bmatrix} 9 \\ 4 \\ 4 \\ -7 \end{bmatrix}, w_1 = \begin{bmatrix} -9 \\ 7 \\ 1 \\ -4 \end{bmatrix}, w_2 = \begin{bmatrix} 10 \\ -2 \\ 8 \\ -2 \end{bmatrix}$$

解: 本题的要点在于研究 w 是否能由 v_1, v_2, v_3 能以线性组合的方式组成, 即是否找到三个常数 c_1, c_2, c_3 , 以便得到三个基向量的线性组合:

$$c_1 v_1 + c_2 v_2 + c_3 v_3 = w$$

把三个列向量并排成矩阵 $v = [v_1, v_2, v_3]$, c_1, c_2, c_3 排列成列向量 c , 则上述方程可以写成矩阵与向量相乘的形式 $v * c = w$ 。若 w 与 v 线性相关, 其组合矩阵 $[v, w]$ 的秩应该与 v 的秩相同, 反之, 其秩应该加 1。可见重要的是秩的增量, 故列出程序 ag803:

```
v1=[7;-4;-2;9]; v2=[-4;5;-1;-7]; % 输入参数
v3=[9;4;4;-7]; w1=[-9;7;1;-4]; w2=[10;-2;8;-2];
v=[v1,v2,v3]; % 将三个基向量组成矩阵
dr1=rank([v,w1])-rank(v) % v 与 w1 组合后矩阵秩的增量
dr2=rank([v,w2])-rank(v) % v 与 w2 组合后矩阵秩的增量
```

运行这个程序, 得到

```
dr1=1, dr2=0
```

这说明 w_1 不是 v_1, v_2, v_3 的线性组合, 而 w_2 则是 v_1, v_2, v_3 的线性组合, w_2 将位于 v_1, v_2, v_3 所张成的 R^3 子空间内。不过, 这个 R^3 空间不是我们习惯的欧几里得几何空间。

由 $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ 组成向量 \mathbf{w}_2 的常数乘子 c_1, c_2, c_3 可以通过语句 $\mathbf{c} = \mathbf{v} \backslash \mathbf{w}_2$ 求得, 结果为:

$$c_1 = -1, \quad c_2 = -2, \quad c_3 = 1$$

8.3 向量的内积和正交性

以上讨论的对各基向量的要求只是线性无关, 实际工程中往往还要求它们之间互相正交并且长度为 1。因此引出了内积和单位向量的概念。在三维空间中, \mathbf{x} 和 \mathbf{y} 两个向量的内积 $[\mathbf{x}, \mathbf{y}] = x_1 y_1 + x_2 y_2 + x_3 y_3$ 。 m 维情况可以写成

$$[\mathbf{x}, \mathbf{y}] = \mathbf{x}^T \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_m y_m = \sum_{i=1}^m x_i y_i \quad (8.10)$$

注意这是一个标量。向量 \mathbf{x} 与自己求内积:

$$\mathbf{x}^T \mathbf{x} = x_1 x_1 + x_2 x_2 + \cdots + x_m x_m = \sum_{i=1}^m x_i^2 \quad (8.11)$$

得到的是其各分量的平方和, 如果各分量正交, 它们的平方根就等于向量的长度 (或模、或范数 norm)。

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}} = \text{norm}(\mathbf{x}) \quad (8.12)$$

因此, 把向量归一化为单位向量 \mathbf{x}_0 的计算公式就是:

$$\mathbf{x}_0 = \frac{\mathbf{x}}{\sqrt{\mathbf{x}^T \mathbf{x}}} = \frac{\mathbf{x}}{\text{norm}(\mathbf{x})} \quad (8.13)$$

在平面情况, 两向量的内积除以两个向量的长度是两个向量夹角的余弦, 可以利用图 8.5 证明如下。

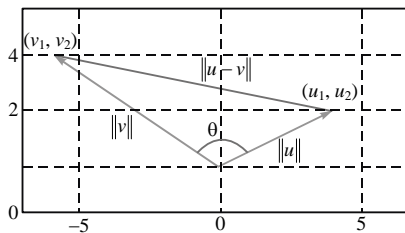


图 8.5 平面上两向量夹角的几何关系

根据三角中的余弦定律,

$$\|\mathbf{u} - \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - 2\|\mathbf{u}\| \cdot \|\mathbf{v}\| \cos \theta$$

移项后, 得到

$$\begin{aligned} \|\mathbf{u}\| \cdot \|\mathbf{v}\| \cos \theta &= \frac{1}{2} [\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - \|\mathbf{u} - \mathbf{v}\|^2] \\ &= \frac{1}{2} [u_1^2 + u_2^2 + v_1^2 + v_2^2 - (u_1 - v_1)^2 - (u_2 - v_2)^2] = u_1 v_1 + u_2 v_2 = [\mathbf{u}, \mathbf{v}] \end{aligned}$$

于是可得出

$$\cos \theta = \frac{\mathbf{u}^T \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} = \frac{[\mathbf{u}, \mathbf{v}]}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} \quad (8.14)$$

这个结果也可以推广到高维空间，只是 θ 被抽象化了：

$$\cos \theta = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\mathbf{x}^T \mathbf{y}}{\sqrt{\mathbf{x}^T \mathbf{x}} \cdot \sqrt{\mathbf{y}^T \mathbf{y}}} = \frac{\mathbf{x}^T \mathbf{y}}{\text{norm}(\mathbf{x}) \cdot \text{norm}(\mathbf{y})} \quad (8.15)$$

$$\theta = \arccos \left(\frac{\mathbf{x}^T \mathbf{y}}{\sqrt{\mathbf{x}^T \mathbf{x}} \cdot \sqrt{\mathbf{y}^T \mathbf{y}}} \right) \quad (8.16)$$

根据这个式子就可以求出任何两个同维向量之间的夹角。ATLAST 手册上还给出了一个子程序 `plotangle(x,y)`，给出任意两个同维向量 \mathbf{x} 和 \mathbf{y} ，该子程序将在 \mathbf{x} 和 \mathbf{y} 所处的平面上画出它们的夹角并给出其数值。

在线性代数实验中，我们一般要求只调用 MATLAB 中的基本函数，而不要用 ATLAST 工具箱函数。所以可以分别用下列语句求单位基向量 \mathbf{x}_0 和两向量 \mathbf{x} 和 \mathbf{y} 之间的夹角：

```
x0=x/sqrt(x'*x)      % 或 x0=x/norm(x)
theta=acos((x'*y)/sqrt(x'*x*y'*y))
```

【例 8.4】 求例 8.3 中的单位基向量 $\mathbf{v}_{10}, \mathbf{v}_{20}, \mathbf{v}_{30}$ ，并分别求它们之间的夹角。

解：根据以上的公式，在 ag804 中补列程序如下：

```
v10=v1/norm(v1),
v20=v2/norm(v2),
v30=v3/norm(v1),
theta12=acos((v1'*v2)/(norm(v1)*norm(v2)))
theta13=acos((v1'*v3)/(norm(v1)*norm(v3)))
theta23=acos((v3'*v2)/(norm(v3)*norm(v2)))
```

程序运行的结果为：

$$\mathbf{v}_{10} = \begin{bmatrix} 0.5715 \\ -0.3266 \\ -0.1633 \\ 0.7348 \end{bmatrix}, \quad \mathbf{v}_{20} = \begin{bmatrix} -0.4193 \\ 0.5241 \\ -0.1048 \\ -0.7338 \end{bmatrix}, \quad \mathbf{v}_{30} = \begin{bmatrix} 0.7071 \\ 0.3143 \\ 0.3143 \\ -0.5500 \end{bmatrix},$$

theta12= 2.7733, theta13 = 1.7254, theta23 = 1.3296 【弧度】

要检验 $\mathbf{v}_{10}, \mathbf{v}_{20}, \mathbf{v}_{30}$ 是否为单位向量，可以检验 $\mathbf{v}_{10}' * \mathbf{v}_{10} \cdots$ ，是否等于 1。

下面讨论向量的正交性，向量正交即它们的夹角 θ 为 $\pi/2$ ，或 $\cos \theta = 0$ ，所以两向量 \mathbf{x}, \mathbf{y} 正交的条件是它们的内积应为零。

$$[x, y] = x^T y = 0 \quad (8.17)$$

当给定 R^m 空间任意 k 个线性无关的基向量组 (所以 $k \leq m$) v_1, v_2, \dots, v_k , 要求将它们变换为同维的规范化正交基向量 e_1, e_2, \dots, e_k 时, 书上通常介绍的是施密特算法, 它的公式如下:

$$\begin{aligned} q_1 &= v_1 \\ q_2 &= v_2 - \frac{[v_2, q_1]}{[q_1, q_1]} q_1 \\ q_3 &= v_3 - \frac{[v_3, q_1]}{[q_1, q_1]} q_1 - \frac{[v_3, q_2]}{[q_2, q_2]} q_2 \\ &\dots \quad \dots \quad \dots \\ q_k &= v_k - \frac{[v_k, q_1]}{[q_1, q_1]} q_1 - \frac{[v_k, q_2]}{[q_2, q_2]} q_2 - \dots - \frac{[v_k, q_{k-1}]}{[q_{k-1}, q_{k-1}]} q_{k-1} \end{aligned} \quad (8.18)$$

得到 $q_i (i=1, 2, \dots, k)$ 后, 再把它们除以 $\text{norm}(q_i)$, 就可归一化为单位向量 e_k 。

按照这个公式, ATLAST 手册中给出了相应的程序 `gschmidt`。公式中的递归计算过程, 在程序中用一个 `for` 循环来实现, 使得程序相当紧凑。但编写和阅读这样的程序会有一些难度, 如果想更好地理解编程原理, 可参考这里列写出来的程序。显然这个程序要求 $n \leq m$ 。

```
function [Q,R]=gschmidt(V)
[m,n]=size(V); R=zeros(n);
R(1,1)=norm(V(:,1));
Q(:,1)=V(:,1)/R(1,1);
for k=2:n
    R(1:k-1,k)=Q(:,1:k-1)'*V(:,k);
    Q(:,k)=V(:,k)-Q(:,1:k-1)*R(1:k-1,k);
    R(k,k)=norm(Q(:,k));
    Q(:,k)=Q(:,k)/R(k,k);
end
```

一般来说, 读书时可以花时间消化一下编程思想, 以后工作时只要能够调用就行。调用时输入 `[Q,R]=gschmidt(v)`, 得出的 Q 就是单位正交基向量 e 。

MATLAB 中不采用施密特算法, 它用更好的算法编成了正交分解子程序 `qr.m`, 将 v 分解为 Q 和 R 两个矩阵的乘积。调用方法为:

```
[Q,R]=qr(v)
```

它满足 $Q^*R=v$

当 v 是 $m \times n$ 矩阵时, 输出变元 R 是 $m \times n$ 的上三角矩阵。而 Q 则是 $m \times m$ 单位正交矩阵 (这和 `gschmidt` 子程序不同)。这个子程序并不限定 $n \leq m$, 在 $n < m$ 时, 在 Q 中取前 n 列, 就是要求的规范化正交基向量 e_1, e_2, \dots, e_n , 即

```
e=Q(:, [1:n])
```

学习时可以用教学子程序来对照理论公式, 所以我们提供了 `gschmidt` 函数。工程应用中一般不要调用教学子程序, 而应当调用 MATLAB 中给出的商品化子程序, 因为这些程序通过了比较严格的测试, 可以适用于各种情况, 并且提供了比较完备的出错提示的功能。

【例 8.5】 对于例 8.3 的数据, 求其规范化正交基向量 e_1, e_2, \dots, e_n 。

解: 已知

$$V = \begin{bmatrix} 7 & -4 & 9 \\ -4 & 5 & 4 \\ -2 & -1 & 4 \\ 9 & -7 & -7 \end{bmatrix}$$

输入

```
V=[7,-4,9;-4,5,4;-2,-1,4;9,-7,-7]
[Q,R]=qr(V)
e=Q(:, [1:3])
e'*e           % 检验 e 是否为规范化正交基向量
```

得到:

$$Q = \begin{bmatrix} -0.5715 & -0.3164 & -0.7473 & -0.1217 \\ 0.3266 & -0.6096 & -0.1080 & 0.7142 \\ 0.1633 & 0.7144 & -0.5022 & 0.4591 \\ -0.7348 & 0.1339 & 0.4216 & 0.5141 \end{bmatrix},$$

$$R = \begin{bmatrix} -12.2474 & 8.8998 & 1.9596 \\ 0 & -3.4341 & -3.3662 \\ 0 & 0 & -12.1173 \\ 0 & 0 & 0 \end{bmatrix}$$

所以规范化正交基向量是 Q 的前三列:

$$e = [e_1 \ e_2 \ e_3] = \begin{bmatrix} -0.5715 & -0.3164 & -0.7473 \\ 0.3266 & -0.6096 & -0.1080 \\ 0.1633 & 0.7144 & -0.5022 \\ -0.7348 & 0.1339 & 0.4216 \end{bmatrix},$$

$$e^T * e = \begin{bmatrix} 1.0000 & 0.0000 & 0 \\ 0.0000 & 1.0000 & 0.0000 \\ 0 & 0.0000 & 1.0000 \end{bmatrix}$$

证明 e 确实为规范化正交基向量。

8.4 齐次解空间

从第6章已经知道, 方程 $\mathbf{Ax}=\mathbf{b}$ 有解的条件是系数矩阵 \mathbf{A} 的秩与增广矩阵 $[\mathbf{A}, \mathbf{b}]$ 的秩相等。设有 m 个方程和 n 个变量, \mathbf{A} 的秩是 r , 则经过行简化后得到的行阶梯矩阵 \mathbf{U} 的有 r 个枢轴元素, 非枢轴元素有 $n-r$ 个。因此该方程的全解将等于 $\mathbf{Ax}=\mathbf{b}$ 的一个特解加上其齐次方程 $\mathbf{Ax}=\mathbf{0}$ 的通解。 $\mathbf{Ax}=\mathbf{b}$ 的特解可以通过 $\mathbf{A} \backslash \mathbf{b}$ 求得, 当 $\det(\mathbf{A})=0$ 时, 可以先变换为行阶梯形式, 用其枢轴行来求。这已在 6.5 节中详细讨论过。现在着重讨论齐次方程 $\mathbf{Ax}=\mathbf{0}$ 的通解。它包含 $n-r$ 个自由变量 (或任意常数), 本节将从向量空间的视点来讨论它的解, 研究这个解所张成的向量空间。

方程 $\mathbf{Ax}=\mathbf{0}$ 有非零解的条件是 $\det(\mathbf{A})=0$ 。如果 $\det(\mathbf{A}) \neq 0$, 即 \mathbf{A} 是满秩的, 则 $\mathbf{Ax}=\mathbf{b}$ 有唯一解, 因而 $\mathbf{Ax}=\mathbf{0}$ 就只有零解。可见解空间的问题只是在 \mathbf{A} “欠秩”, 因而方程在有无数解的情况下才有实际意义。因为 $\mathbf{Ax}=\mathbf{0}$ 意味着这些解 x 的集合经过矩阵 \mathbf{A} 变换后都映射到像空间的零点, 所以英文把此解所张成的空间称为 Null Space, 直译为“零空间”, 我国的通用译名为“解空间”, 这个译名没有指明是什么解的空间, 所以两种译法都不尽人意, 我们觉得用“齐次解空间”较为准确, 在本书中就用了这个术语。这里所以要特别强调原文, 是因为 MATLAB 中几个与此相关子程序的名称都是来源于原文 null。

下面来看一个例子:

【例 8.6】 试求下列系数矩阵的齐次解空间:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -1 & 1 & -1 \\ 5 & -5 & -7 & -3 & -7 \\ 3 & -3 & 0 & -6 & 0 \end{bmatrix} \quad (8.19)$$

解: 输入 \mathbf{A} , 并求出它的简化行阶梯形式, 输入 $[\mathbf{U}, \mathbf{ip}] = \text{rref}(\mathbf{A})$, 得到

$$\mathbf{U} = \begin{bmatrix} 1 & -1 & 0 & -2 & 0 \\ 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{ip} = [1, 3] \quad (8.20)$$

它表示方程组已化简为:

$$\begin{aligned} x_1 - x_2 - 2x_4 &= 0 \\ x_3 - x_4 + x_5 &= 0 \\ 0 &= 0 \end{aligned} \quad (8.21)$$

其枢轴项为 x_1 和 x_3 两项, 说明 $r=2$ 。我们把枢轴项之外的 $n-r=3$ 个变量 x_2, x_4, x_5 作为自由变量, 它们的方程式都是 $0=0$, 可列成为恒等式 $x_2=x_2, x_4=x_4, x_5=x_5$ 。把 \mathbf{U} 方程组前两式中的 x_2, x_4, x_5 移到等式右端, 于是可以列出解的表示式:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_2 + 2x_4 \\ x_2 \\ x_4 - x_5 \\ x_4 \\ x_5 \end{bmatrix} = x_2 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + x_5 \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad (8.22)$$

把 x_2, x_4, x_5 看作任意常数 c_1, c_2, c_3 , 它们所乘的三个列向量看作基向量, 这个式子就表示了一个三维的向量空间, 在这个空间中所有的向量都能使 $Ax=0$ 。所以它被称为齐次解空间或零空间。

从计算机编程的需要出发, 我们尝试从一个实际例题中找到解的结构的一般规律。把这三个列向量并排起来, 得出齐次解空间的系数矩阵

$$N = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8.23)$$

不难看出, N 的行数等于 U_0 的列数。其 1, 3 两个枢轴行的系数 $[1, 2, 0]$ 和 $[0, 1, -1]$ 是 U_0 矩阵中 2, 4, 5 各列并排反号后的系数 (由于移项到右边, 所以要反个号), N 的 2, 4, 5 各行的系数恰好构成了一个 3×3 的单位矩阵 $\text{eye}(3)$ 。把这些行号、列号的具体数字用一般的规则来填入, 枢轴列号是 i_p , 除去枢轴列号以外的自由变量下标号 i_s 可以用 `other` 函数或两条语句写成:

```
is=1:n; is(ip)=[]
```

其中第一句设定了全部列号, 第二句把其中的枢轴列号去掉, 剩下的当然就是其余的列号了。注意 U_0 的列号就是 N 的行号, 这样齐次解空间的 $m \times is$ 系数矩阵 N 可以用下面的程序来自动完成:

```
function N=nulspace(A)
[m,n]=size(A);
[U0,ip]=rref(A)
is=1:n; is(ip)=[];
N(ip,:)=-U0(1:rank(A),is);
N(is,:)=eye(n-rank(A))
```

【例 8.7】 设线性方程的系数矩阵如下, 试求 $Ax=0$ 的通解。

$$A = \begin{bmatrix} -4 & 1 & 6 & -2 & 2 & -2 \\ 1 & -2 & 0 & 1 & -2 & -1 \\ -4 & 1 & 5 & 0 & 3 & -1 \\ 1 & -3 & 1 & 1 & -3 & -2 \end{bmatrix} \quad (8.24)$$

解：输入系数矩阵后，输入 $v=\text{nulbasis}(A)$ ，得到

$$v = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{即 } v_1 = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, v_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (8.25)$$

即 R^6 空间中三个列向量 $[v_1, v_2, v_3]$ 张成的三维子空间包含了能使 $Ax=0$ 的全部向量。

要检验此子程序的正确性，可以显示中间结果 U_0, i_p 如下，看它是否符合前面总结的规律。

$$U_0 = \begin{bmatrix} 1 & 0 & 0 & -3 & -2 & -1 \\ 0 & 1 & 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, i_p = [1, 2, 3] \quad (8.26)$$

当然，更可靠的办法是调用 MATLAB 中求齐次解空间的子程序 `null`，看其结果是否相同。因为齐次解空间基向量其实有无限多种解，如要进行比较，调用 `null` 子程序时应在后面加第二输入变元 '`r`'，表示求其有理数表示式。否则它给出的是规范化正交基，无法与我们的结果对比。输入

```
v=null(A, 'r')
```

得到的结果也为 (8.25)，和自编程序完全相同。尽管学习阶段用自编程序可以帮助我们理解原理，但今后工作时应尽量调用商品软件的子程序。

8.5 超定方程的解——最小二乘问题

有时用向量空间的方法可以更为简捷地推导公式，超定方程的解可以作为一个例子。

既然我们已讨论了“适定”方程组和“不定”方程组的求解方法，自然会提出如何解“超定”方程组的问题。当然也可以用“没有解”三个字把问题挡回去。实际上科学的发展是无止境的，工程上的大量问题不应该那么绝对，比如线性方程组的左右两端一定那么相等吗？考虑到建模时人们忽略的许多次要因素，考虑到系数测量或提取的误差，它们实际上不会真正相等。所以，不能用这种绝对化的思维来处理工程问题，工程问题可以允许方程有误差，往往把一组解代入方程后，每个方程都有误差；但工程上要分析这样的误差是否允许，并希望这些误差在一定意义下的总和为最小。就以例 6.1 来说，虽然无法使三根直线交于一点，但我们可以在交出的三角形中找到一点，使它与三条直线的距离为最小，把它当作解。在这样的思路引导下，于是超定方程也有解的方法了。通常采用的是误差平方和为最小的准则，其解也称为最小二乘解。我们将在本节中讨论这个问题。

设 A 为 $m \times n$ 矩阵, 若 m 是独立方程个数, 而 $m > n$. 表明在 $Ax=b$ 的矩阵方程中, 独立方程数大于未知数的数目, 这时方程组是超定的, 即不可能找到一个 x , 满足 $Ax-b=0$. 前面已指出, 如果我们不寻求理想的数学解, 而是从工程意义上找到尽量接近理想的解, 那就应该引入误差向量 e .

令

$$e = Ax - b \quad (8.27)$$

写出其完全的矩阵形式如下

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = Ax - b \quad (8.28)$$

注意: 误差向量 e 和 b 是 m 维的, 而 x 则是 n 维的, $m > n$. 现在的问题是, 找到解 x , 使误差向量的长度或它的范数为最小.

从向量空间重新研究一下例 6.1 的方程组 (d), 它是一个超定方程

$$\begin{aligned} x_1 + x_2 &= 1 \\ x_1 - x_2 &= 3 \\ -x_1 + 2x_2 &= -3 \end{aligned} \quad (8.29)$$

改写成

$$x_1 \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + x_2 \cdot \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix} \quad (8.30)$$

简写为

$$x_1 \cdot v_1 + x_2 \cdot v_2 = b \quad (8.31)$$

现在是在三维空间讨论问题, 故可以有鲜明的几何意义. 令 $A=[v_1, v_2]$, 其中 v_1 和 v_2

是两个三维空间的向量, 它们在三维空间中将张成一个二维子空间 (即平面), 选择不同的 x_1 和 x_2 将使等式左端有不同的合成向量 $Ax=x_1v_1+x_2v_2=q$, q 必定处于 v_1 和 v_2 张成的平面之内. b 是三维空间中的一个点, 如果这个 b 点碰巧恰好在这个平面内, 那么 x_1 和 x_2 就可以有解. 但一般来说, b 不会在这个平面内, 这时最近似的解 \hat{x} 就应该是该平面上与 b 点最近的点 $A\hat{x}$ 所对应的坐标. 它应该是 b 点向 v_1 和 v_2 张成的平面的投影. 所以 $A\hat{x}$ 和 b 的连线应该和 v_1 和 v_2 张成的平面垂直, 也就是说, 必须分别与 v_1 和 v_2 正交. 如图 8.6 所示.

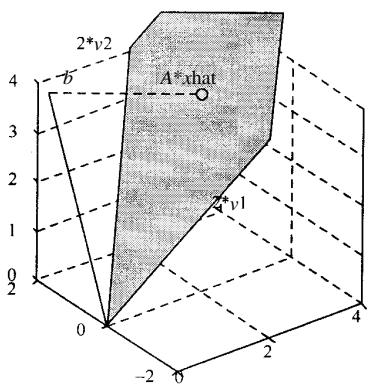


图 8.6 三维空间内最小二乘解的几何意义

$A\hat{x}$ 和 b 的连线向量应该是这两个向量之差, 即 $A\hat{x} - b$, 它与 v_1 和 v_2 正交的要求可以分别表示为:

$$v_1^T (A\hat{x} - b) = 0 \quad (8.32)$$

和

$$\mathbf{v}_2^T(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}) = 0 \quad (8.33)$$

综合在一起可以写成:

$$\mathbf{A}^T(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}) = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix}(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}) = \mathbf{0} \quad (8.34)$$

所以 $\mathbf{A}\hat{\mathbf{x}}$ 点所对应的 x 坐标很容易由上式移项解得为:

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{b}, \quad (8.35)$$

最后解得:

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \quad (8.36)$$

误差向量为

$$\mathbf{e} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{b}$$

这就是在三维空间内导出的最小二乘解的公式。在当前的问題中, \mathbf{A} 是 3×2 矩阵, \mathbf{A}^T 是 2×3 矩阵, 因此 $\mathbf{A}^T\mathbf{A}$ 是 2×2 矩阵, 同理 $\mathbf{A}^T\mathbf{b}$ 是 2×1 向量, 求出的 $\hat{\mathbf{x}}$ 也是一个二维的列向量。从维数上符合命题的要求。

因为在三维的条件下, 我们已经把几何关系上升到了代数方程, 中间的任何一步都没有施加维数的限制, 因此这个公式在高维的线性代数方程中同样适用。

【例 8.8】 求方程组 (8.29) 的最小二乘解。

解: 用 MATLAB 求解, 程序如下:

```
A=[1,1;1,-1; -1,2],
b=[1;3;3]
xhat=inv(A'*A)*A'*b
e=A*xhat-b, norm(e)
```

运行此程序, 得到

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix},$$

$$\hat{\mathbf{x}} = \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 1.0000 \\ -3.0000 \\ -2.0000 \end{bmatrix},$$

$$\text{norm}(\mathbf{e}) = 3.7417$$

图 8.6 就是我们用本例参数和 ATLAST 提供的程序 viewsubspaces 画出的空间图形。其中向量所组成的平面用 $2*\mathbf{v}_1$ 和 $2*\mathbf{v}_2$ 画出, 系数向量 \mathbf{b} 单独画在图中, 它不在 \mathbf{v}_1 和 \mathbf{v}_2 所张成的平面中, 所以这个方程无解, 只能求出向量 \mathbf{b} 在该平面上的垂足 $\mathbf{A}*\mathbf{xhat}$, \mathbf{xhat} 就是差

方程的最小二乘解。 \mathbf{b} 与垂足 $\mathbf{A}*\mathbf{x}_{hat}$ 之间的连线在图 8.6 上用虚线表示。它的长度就等于 $\text{norm}(\mathbf{e})=3.7417$ 。这个程序命名为 ag808。主要提供教师参考。

在 MATLAB 中, 把运算 $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ 单独编成一个子程序, 称为 pinv 函数, pinv 是 Psuedoinverse (伪逆或广义逆) 的缩称。这样求最小二乘解的公式可以写成:

$$\mathbf{x}=\text{pinv}(\mathbf{A})*\mathbf{b}$$

与“适定方程”的解 $\mathbf{x}=\text{inv}(\mathbf{A})*\mathbf{b}$ 非常相似, 只是 pinv 函数并不要求 \mathbf{A} 是方阵。

最小二乘解也可以用“\”运算符表示, 这就把“欠定方程”、“适定方程”和“超定方程”都用统一的运算格式:

$$\mathbf{x}=\mathbf{A}\backslash\mathbf{b}$$

解决, 非常有利于记忆。MATLAB 将自动根据系数矩阵 \mathbf{A} 的行数 m 和列数 n , 来判断采用哪个方法和程序。不过对于欠定方程, 这个式子只给出了一个特解。还要加上齐次解才是通解。

【例 8.9】参考文献 [10] 设在某一实验中, 给某元件加[1,2,3,4,5]V 电压, 测得的电流为 [0.2339, 0.3812, 0.5759, 0.8153, 0.9742]ma。求此元件的电阻。

解: 设直线的方程为 $y=c(1)x+c(2)$, 待定的系数是 $c(1)$, $c(2)$ 。将上述数据分别代入 x , y , 把这五个方程联立, 用矩阵表述:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} c(1) + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} c(2) = \begin{bmatrix} 0.2339 \\ 0.3812 \\ 0.5759 \\ 0.8153 \\ 0.9742 \end{bmatrix}$$

设其系数矩阵为 \mathbf{datax} 和 \mathbf{datay} , 得

$$\mathbf{datax} * c(1) + \text{ones}(N,1) * c(2) = \mathbf{datay}$$

其中 \mathbf{datax} , \mathbf{datay} 都是 5 行数据列向量, 这是 5 个一次代数方程, 含两个未知数, 方程数超过未知数的数目, 是一个超定方程, 写成 $\mathbf{A}*\mathbf{c} = \mathbf{B}$, 其最小二乘解可以直接使用 MATLAB 的左除运算符 $\mathbf{c}=\mathbf{A}\backslash\mathbf{B}$ 来求得。因此程序如下:

```
A = [datax , ones(N,1)]; B = datay; c = A \ B
```

$c(2)$ 的存在说明此直线不通过原点, 若要在过原点的曲线族中拟合。就要在原始方程中规定 $c(2)=0$ 。把 \mathbf{A} 中的第二列去掉, 即令 $\mathbf{A}=\mathbf{datax}$, $\mathbf{c}_0=\mathbf{A}\backslash\mathbf{B}$,

MATLAB 程序 ag809

```
clear,
datax = [1:5]';
datay=[ 0.2339, 0.3812, 0.5759, 0.8153, 0.9742 ]'; % 原始数据
A = [datax , ones(5,1)]; B = datay; c = A\B, r=1/c(1) % 线性拟合
plot(datax,datay,'o'),hold on % 绘出原始数据点图
xi=0:0.1:5;
```

```

yi=c(1)*xi+c(2);    % 设置 51 个取值点
A1 = datax;
c0 = A1\B, r0=1/c0 % 通过原点的线性拟合
plot(xi,yi,xi,c0*xi,':') % 绘图
hold off

```

运行结果

```

c = 0.1905
    0.0247
r =    5.2228
c0 =    0.1972
r0 =    5.0659

```

绘出的两种拟合曲线如图 8.7 所示。

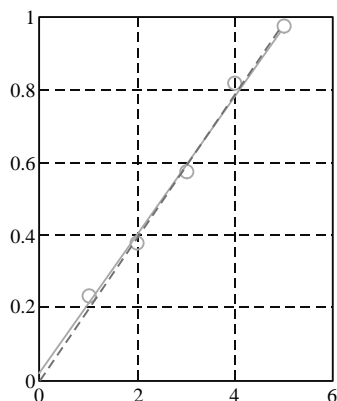


图 8.7 两种拟合结果

8.6 应用实例

8.6.1 价格平衡模型

在 Leontiff 成为诺贝尔奖金获得者的历史中，线性代数曾起过重要的作用，我们来看看他的基本思路。假定一个国家或区域的经济可以分解为 n 个部门，这些部门都有生产产品或服务的独立功能。设单列 n 元向量 \mathbf{x} 是这 n 个部门的产出，组成在 R^n 空间的产出向量。先假定该社会是自给自足的经济，这是一个最简单的情况。因此各经济部门生产出的产品，完全被自己部门和其他部门所消费。Leontiff 提出的问题是，各生产部门的实际产出的价格 \mathbf{p} 应该是多少，才能使各部门的收入和消耗相等，以维持持续的生产。

Leontiff 的输入输出模型中规定：对于每个部门，存在着一个在 R^m 空间单位消耗列向量 \mathbf{v}_i ，它表示第 i 个部门每产生一个单位（比如 100 万美金）产品中，本部门和其他各个部门消耗的百分比。在自给自足的经济中，这些列向量中所有元素的总和应该为 1。把这 n 个 \mathbf{v}_i ，并列起来，它可以构成一个 $n \times n$ 的系数矩阵，可称为内部需求矩阵 \mathbf{V} 。

举一个最简单的例子，假如一个自给自足的经济体由三个部门组成，它们是煤炭业、电力业和钢铁业。它们的单位消耗列向量和销售收入列向量 \mathbf{p} 如下表：

由下列部门购买	每单位输出的消耗分配			销售价格 \mathbf{p} (收入)
	煤炭业	电力业	钢铁业	
煤炭业	0.	0.4	0.6	p_c
电力业	0.6	0.1	0.2	p_e
钢铁业	0.4	0.5	0.2	p_s

这就是说，电力业产出了 100 个单位的产品，有 40 个单位会被煤炭业消耗，10 个单位被自己消耗，而被钢铁业消耗的是 50 个单位，各行业付出的费用为：

$$p_e \cdot v_2 = p_e \cdot \begin{bmatrix} 0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$$

这就是内部消耗的计算方法，把几个部门都算上，可以写出

$$\text{消耗成本} = p_c v_c + p_e v_e + p_s v_s = [v_c, v_e, v_s] \begin{bmatrix} p_c \\ p_e \\ p_s \end{bmatrix} = \text{销售收入} = \begin{bmatrix} p_c \\ p_e \\ p_s \end{bmatrix}$$

其中

$$V = [v_c, v_e, v_s] = \begin{bmatrix} 0. & 0.4 & 0.6 \\ 0.6 & 0.1 & 0.2 \\ 0.4 & 0.5 & 0.2 \end{bmatrix}$$

于是总的价格平衡方程可以写成为：

$$p - Vp = 0$$

$$(I - V)p = 0$$

此等式右端常数项为零，是一个齐次方程。它有非零解的条件是系数行列式等于零，可以用行阶梯简化来求解。

用 MATLAB 语句写出其解的表示式：

```
v=[0.,0.4,0.6;0.6,0.1,0.2;0.4,0.5,0.2],
U0 = rref([(eye(3)-V),zeros(3,1)])
```

程序运行的结果为

$$U0 = \begin{bmatrix} 1.0000 & 0 & -0.9394 & 0 \\ 0 & 1.0000 & -0.8485 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

这个结果是合理的，简化行阶梯形式只有两行，说明 $[I - V]$ 的秩是 2，所以它的行列式必定为零。由于现在有三个变量，只有两个方程，必定有一个变量可以作为自由变量。记住 $U0$ 矩阵中各列的意义，它们分别是原方程中 p_c , p_e , p_s 的系数，所以简化行阶梯矩阵 $U0$ 表示的是下列方程：

$$\begin{cases} p_c - 0.9394p_s = 0 \\ p_e - 0.8485p_s = 0 \end{cases} \Rightarrow \begin{cases} p_c = 0.9394p_s \\ p_e = 0.8485p_s \end{cases}$$

这里取钢铁业价格 p_s 为自由变量，所以煤炭业和电力业的价格应该分别为钢铁业价格的 0.94 和 0.85 倍。如果钢铁业产品价格总计为 100 万元，则煤炭业的产品价格总计为 94 万元，电力业的价格总计为 85 万元。

8.6.2 宏观经济模型

在上一个应用实例中。假定一个国家或区域的经济可以分解为 n 个部门，这些部门都有生产产品或服务的独立功能。设单列 n 元向量 \mathbf{x} 是这些 n 个部门的产出，组成在 R^n 空间的产出向量。当时假定那是一个自给自足的经济体，现在则考虑它有外部的需求，要向外提供产品。设 \mathbf{d} 为最终（即外部）需求向量，它也是 R^n 空间的单列向量，表示这 n 个部门以外的非经济部门对该国经济各部门的需求，比如政府消费、出口和战略储备等。

在各经济部门进行满足外部需求的生产时，它们也必须增加内部的相互需求，这种各部门的内部交叉需求非常复杂。诺贝尔经济学奖得主，美国 Wassily Leontiff 教授提出的问题是，为了满足外部的最终需求向量 \mathbf{d} ，各生产部门的实际产出 \mathbf{x} 应该是多少，这对于经济计划的制订当然很有价值。因为

$$\mathbf{x} = \{\text{内部需求}\} + \{\text{外部需求 } \mathbf{d}\}$$

Leontiff 输入输出模型的一个基本假定是：对于每个部门，存在着一个在 R^n 空间单位消耗列向量 \mathbf{v}_i ，它表示第 i 个部门每产出一个单位（比如 100 万美金）产品，需要消耗其他部门产出的数量。把这 n 个 \mathbf{v}_i 并列起来，可以构成一个 $n \times n$ 的系数矩阵，可称为内部需求矩阵 \mathbf{V} 。由于要向外部提供产品，故内部需求矩阵各列向量元素的和必然小于 1（上例中要求它等于 1）。

举一个最简单的例子，假如国民经济由三个部门组成，它们是制造业、农业和服务业。它们的单位消耗列向量如下表。

向下列部门购买	每单位输出的输入消耗		
	制造业	农业	服务业
制造业	0.5	0.4	0.2
农业	0.2	0.35	0.15
服务业	0.15	0.1	0.3

如果制造业产出了 100 个单位的产品，有 50 个单位会被自己消耗，20 个单位被农业消耗，而被服务业消耗的是 15 个单位，用算式表示为

$$100\mathbf{v}_1 = 100 \cdot \begin{bmatrix} 0.5 \\ 0.2 \\ 0.15 \end{bmatrix} = \begin{bmatrix} 50 \\ 20 \\ 15 \end{bmatrix}$$

这就是内部消耗的计算方法，把几个部门都算上，可以写出

$$\{\text{内部需求}\} = x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + x_3\mathbf{v}_3 = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{V}\mathbf{x}$$

其中

$$\mathbf{V} = \begin{bmatrix} 0.5 & 0.4 & 0.2 \\ 0.2 & 0.35 & 0.15 \\ 0.15 & 0.1 & 0.3 \end{bmatrix}$$

于是总的需求方程可以写成为:

$$\mathbf{x} - \mathbf{V}\mathbf{x} = \mathbf{d}$$

$$(\mathbf{I} - \mathbf{V})\mathbf{x} = \mathbf{d}$$

从而可用 MATLAB 语句写出其解的表示式

$$\mathbf{x} = \text{inv}(\mathbf{I} - \mathbf{V}) * \mathbf{d}$$

用一个数字来试算一下, 设外部需求为 $\mathbf{d} = \begin{bmatrix} 30 \\ 20 \\ 10 \end{bmatrix}$, 则可以用以下程序 ag862 解出 \mathbf{x} 。

$$\mathbf{V} = [0.5, 0.4, 0.2; 0.2, 0.35, 0.15; 0.15, 0.1, 0.3],$$

$$\mathbf{d} = [30; 20; 10]$$

$$\mathbf{x} = \text{inv}(\text{eye}(3) - \mathbf{V}) * \mathbf{d}$$

程序运行的结果为

$$\mathbf{x} = \begin{bmatrix} 160.4563 \\ 94.4867 \\ 62.1673 \end{bmatrix}$$

这个结果是合理的, 因为实际产出应该比外部需求大得多, 以应付内部的消耗。我们常常说, 某某外部需求可以拉动国民经济增长多少个百分点, 就是从这样的模型中得出的。

内部需求矩阵 \mathbf{V} 要满足一些基本要求, 一般各列的列向元素总和必须小于 1, 否则这个部门就将入不敷出而亏损, 但我们仍可能求出上述方程的解。当所有的列向量都出现列向元素总和大于 1 的情况时, 解 \mathbf{x} 中会出现负值, 因而是庸解。请读者分析其实际意义。

8.6.3 信号流图模型

有关信号流图模型的详细情况可参见参考文献[11], 本节仅摘录其部分进行讲解。

信号流图是用来表示和分析复杂系统内的信号变换关系的工具。它和交通流图或其他物流图不同, 其基本概念如下:

(1) 系统中每个信号用图上的一个节点表示。如图中的 u, x_1, x_2 。(一般物流图中是把物流标在箭杆上的。)

(2) 系统部件对信号实施的变换关系用有向线段表示, 箭尾为输入信号, 箭头为输出信号, 箭身标注对此信号进行变换的乘子。如图上的 G_1, G_2 。如果乘子为 1, 可以不必标注。

(3) 每个节点信号的值等于所有指向此节点的箭头信号之和, 每个节点信号可以向外输出给多个部件, 其值不变。(物流图的节点上要求流入与流出相等, 与此不同)

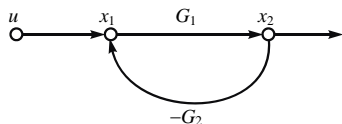


图 8.8 带反馈的简单信号流图

根据这几个概念, 可以列出图 8.8 的方程如下。

$$x_1 = u - G_2 x_2, \quad x_2 = G_1 x_1$$

写成矩阵方程

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & -G_2 \\ G_1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

或

$$\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{P}u$$

移项整理, 可以得到求所有未知信号向量 \mathbf{x} 的公式

$$\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & -G_2 \\ G_1 & 0 \end{bmatrix} \right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & G_2 \\ -G_1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$(\mathbf{I} - \mathbf{Q})\mathbf{x} = \mathbf{P}u$$

$$\mathbf{x} = \text{inv}(\mathbf{I} - \mathbf{Q}) * \mathbf{P}u$$

定义系统的传递函数 \mathbf{W} 为输出信号与输入信号之比 \mathbf{x}/u , 则 \mathbf{W} 可按下式求得:

$$\mathbf{W} = \mathbf{x}/u = \text{inv}(\mathbf{I} - \mathbf{Q}) * \mathbf{P}$$

因为

$$\mathbf{I} - \mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & -G_2 \\ G_1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & G_2 \\ -G_1 & 1 \end{bmatrix}$$

求这个二阶矩阵的逆可以直接用下面的公式:

$$\text{若 } \mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \text{ 则 } \mathbf{A}^{-1} = \frac{1}{ad - bc} \cdot \begin{bmatrix} d & -b \\ -c & a \end{bmatrix},$$

所以

$$(\mathbf{I} - \mathbf{Q})^{-1} = \frac{1}{1 + G_1 G_2} \begin{bmatrix} 1 & -G_2 \\ G_1 & 1 \end{bmatrix}$$

$$\mathbf{x}/u = \begin{bmatrix} x_1/u \\ x_2/u \end{bmatrix} = (\mathbf{I} - \mathbf{Q})^{-1} \mathbf{P} = \frac{1}{1 + G_1 G_2} \begin{bmatrix} 1 & -G_2 \\ G_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{1 + G_1 G_2} \begin{bmatrix} 1 \\ G_1 \end{bmatrix}$$

即对 x_1 的传递函数为 $\frac{1}{1 + G_1 G_2}$, 对 x_2 的传递函数为 $\frac{G_1}{1 + G_1 G_2}$ 。

对于阶次高的情况, 求逆就必须用软件工具了。如果信号流图中有 G_1 那样的符号变量, 那么它的求解要用符号运算工具箱, 对于本题, 其 MATLAB 程序为:

```
syms G1 G2
Q=[0,-G2;G1,0],P=[1;0]
W=inv(eye(2)-Q)*P
```

程序运行的结果是 $\mathbf{W} = \begin{bmatrix} 1/(1+G_2*G_1) \\ G_1/(1+G_2*G_1) \end{bmatrix}$

与前面的结果相同。

这虽然是一个简单的问题, 用一些其他的数学方法也能得到同样的结果。很出名的“梅森公式”就是用图形拓朴的方法得到信号流图的公式, 但非常烦琐, 而且无法自动化。到

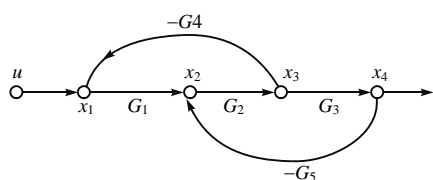


图 8.9 带双重反馈的信号流图
照上述方法列出它的方程如下：

$$\begin{aligned}x_1 &= -G_4x_3 + u \\x_2 &= G_1x_1 - G_5x_4 \\x_3 &= G_2x_2 \\x_4 &= G_3x_3\end{aligned}$$

列为矩阵方程，得到

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -G_4 & 0 \\ G_1 & 0 & 0 & -G_5 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

公式 $\mathbf{W}=\mathbf{x}/u=\text{inv}(\mathbf{I}-\mathbf{Q})*\mathbf{P}$ 同样是正确的，不过这里的 \mathbf{Q} 和 \mathbf{P} 分别为 4×4 和 4×1 矩阵，用手工求逆是很麻烦的。我们可采用类似的 MATLAB 程序 ag863：

```
syms G1 G2 G3 G4 G5
Q=[0,0,-G4,0;G1,0,0,-G5;0,G2,0,0;0,0,G3,0],
P=[1;0;0;0]
W=inv(eye(4)-Q)*P
Pretty(W(4))
```

程序运行的结果为：

$$\mathbf{W} = \begin{bmatrix} [(1+G2*G5*G3)/(1+G2*G5*G3+G1*G2*G4)] \\ [G1/(1+G2*G5*G3+G1*G2*G4)] \\ [G1*G2/(1+G2*G5*G3+G1*G2*G4)] \\ [G1*G2*G3/(1+G2*G5*G3+G1*G2*G4)] \end{bmatrix}$$

我们关心的输出通常是 x_4 ，也就是最后那个传递函数 $\mathbf{W}(4)=\mathbf{x}(4)/u$ ，其结果为：

$$\mathbf{W}(4) = \frac{x(4)}{u} = \frac{G1*G2*G3}{1+G2*G5*G3+G1*G2*G4}$$

8.6.4 数字滤波器系统函数

有关数字滤波器系统函数的详细情况可参见参考文献[12]，本节仅摘录其部分进行讲解。数字滤波器的网络结构图实际上也是一种信号流图。它唯一的特殊要求是在于所有的

相加节点都限定为双输入相加器。如果有三个信号 x_1 , x_2 和 x_3 要相加, 那就要分成两个节点, x_1 和 x_2 在前一个节点相加, 合成信号再在下一个节点与 x_3 相加。这样做的好处是与实际微处理器或信号处理器的硬软件结构相符合, 有助于信号处理实用系统的开发。另外, 数字滤波器器件有一个迟延一个节拍的处理, 它也是一个线性算子, 它的标注符号为 z^{-1} 。根据这样的结构图, 也可以用类似于 8.6.3 节那样的方法, 用线性代数求它的输入输出之间的传递函数, 在数字信号处理中称为系统函数。

图 8.10 表示了某个数字滤波器的结构图, 现在要求出它的系统函数, 即输出 y 与输入 u 之比。先在它的三个中间节点上标注信号的名称 x_1, x_2, x_3 , 以便对每个节点列写方程。由于迟延算子 z^{-1} 不是数, 要用符号代替, 所以取 $q = z^{-1}$, 按照图 8.10 所示的情况, 可以写出:

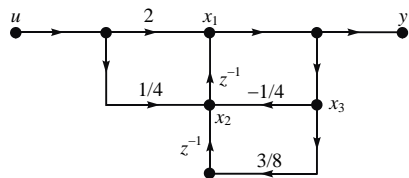


图 8.10 某数字滤波器结构图

$$x_1 = qx_2 + 2u$$

$$x_2 = \left(\frac{3}{8}q - \frac{1}{4} \right) x_3 + \frac{1}{4}u$$

$$x_3 = x_1$$

写成矩阵形式为

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & q & 0 \\ 0 & 0 & \left(\frac{3}{8}q - \frac{1}{4} \right) \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 2 \\ \frac{1}{4} \\ 0 \end{bmatrix} u \Rightarrow \mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{P}u$$

经过移项后, 系统函数 \mathbf{W} 可以写成:

$$\mathbf{W} = \mathbf{x}/u = \text{inv}(\mathbf{I} - \mathbf{Q}) * \mathbf{P}$$

现在可以列写计算系统函数的 MATLAB 程序 ag864,

```
syms q % 规定符号变量
Q(1,2)=q; Q(2,3)=3/8*q-1/4; Q(3,1)=1; % 给非零元素赋值
Q(3,3)=0; % 给右下角元素 Q(3,3) 赋值后, 矩阵中未赋值元素都自动置零
P=[2;1/4;0] % 给 P 赋值
W=inv(eye(3)-Q)*P % 用信号流程图求传递函数的公式
```

程序运行的结果为

$$\mathbf{W} = \begin{bmatrix} -16/(-8+3*q^2-2*q)-2*q/(-8+3*q^2-2*q) \\ -2*(3*q-2)/(-8+3*q^2-2*q)-2/(-8+3*q^2-2*q) \\ -16/(-8+3*q^2-2*q)-2*q/(-8+3*q^2-2*q) \end{bmatrix}$$

我们关心的是 $y=x_3$ 作为输出的系统函数, 故再输入

```
pretty(W(3))
```

得到
$$W(3) = \frac{y}{u} = \frac{-16-2q}{-8+3q^2-2q} = \frac{q+8}{-1.5q^2+q+4} = \frac{z^{-1}+8}{-1.5z^{-2}+z^{-1}+4}$$

用线性代数方法的好处是适用于任何复杂系统，并能用计算机解决问题。

8.7 习题

8.1 设
$$w = \begin{bmatrix} -9 \\ 7 \\ 4 \\ 8 \end{bmatrix}, v_1 = \begin{bmatrix} 7 \\ -4 \\ -2 \\ 9 \end{bmatrix}, v_2 = \begin{bmatrix} -4 \\ 5 \\ -1 \\ -7 \end{bmatrix}, v_3 = \begin{bmatrix} -9 \\ 4 \\ 4 \\ -7 \end{bmatrix},$$

证明 w 是在 R^4 中由 v_1, v_2, v_3 所张成的子空间内。

8.2 设
$$y = \begin{bmatrix} 6 \\ 7 \\ 1 \\ -4 \end{bmatrix}, A = \begin{bmatrix} 5 & -5 & -9 \\ 8 & 8 & -6 \\ -5 & -9 & 3 \\ 3 & -2 & -7 \end{bmatrix},$$

问 y 是否在 A 的列向量张成的子空间中。

8.3 设系数矩阵 A 为：

$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ -1 & -1 & 1 & -2 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

(a) 试求其行阶梯形式 B (简化的非简化的都行)。

(b) 试证明其枢轴元素所在的列是线性无关的，因而枢轴元素的数目就是矩阵的秩。

(c) 说明非枢轴元素的数目就是自由变量数，它决定了齐次解空间的维数。

(d) 说明矩阵的秩与齐次解空间的维数之和等于 A 的列数。

8.4 设
$$w = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -3 \end{bmatrix}, A = \begin{bmatrix} 7 & 6 & -4 & 1 \\ -5 & -1 & 0 & -2 \\ 9 & -11 & 7 & -3 \\ 19 & -9 & 7 & 1 \end{bmatrix},$$

问 w 是在 A 的列向量张成的子空间中还是在 A 的齐次解空间中？

8.5 设
$$w = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix}, A = \begin{bmatrix} -8 & 5 & -2 & 0 \\ -5 & 2 & 1 & -2 \\ 10 & -8 & 6 & -3 \\ 3 & -2 & 1 & 0 \end{bmatrix},$$

问 w 是在 A 的列向量张成的子空间中还是超出了这个空间？

8.6 设 a_1, a_2, \dots, a_5 为矩阵 A 的各列向量，其中

$$A = \begin{bmatrix} 5 & 1 & 2 & 2 & 0 \\ 3 & 3 & 2 & -1 & -12 \\ 8 & 4 & 4 & -5 & 12 \\ 2 & 1 & 1 & 0 & -2 \end{bmatrix}, \quad B = [a_1, a_2, a_4],$$

(a) 解释为什么 a_3, a_5 位于 B 的列空间中。

(b) 找到张成 A 的齐次解空间的一组向量集。

(c) 设 T 是一个 $\mathbf{R}^5 \rightarrow \mathbf{R}^4$ 的映射 $T(x) = Ax$, 说明 T 不是一对一的映射。

8.7 设 H 为由 $[v_1, v_2]$ 张成的集, K 为由 $[v_3, v_4]$ 张成的集。

$$\text{其中} \quad v_1 = \begin{bmatrix} 5 \\ 3 \\ 8 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}, \quad v_3 = \begin{bmatrix} 2 \\ -1 \\ 5 \end{bmatrix}, \quad v_4 = \begin{bmatrix} 0 \\ -12 \\ -18 \end{bmatrix},$$

因此 H 和 K 都是 \mathbf{R}^3 的子空间。实际上 H 和 K 都是通过原点的平面, 其交线为通过原点的一个平面。现要求寻找一个能生成该直线的非零向量 w 。(提示: w 既能写成 $c_1v_1 + c_2v_2$, 又能写成 $c_3v_3 + c_4v_4$, 解方程 $c_1v_1 + c_2v_2 = c_3v_3 + c_4v_4$, 即可求得这些系数 c_j 。

8.8 设

$$V = \begin{bmatrix} 3 & 4 & -6 & 4 \\ -4 & 0 & 0 & 8 \\ -3 & -2 & 3 & 1 \\ 2 & 6 & -9 & 11 \end{bmatrix},$$

(a) 计算 V 的简化行阶梯形式, 并用它来决定 V 的列空间的一组基。

(b) 从列向量 $\{v_1, v_2, v_3, v_4\}$ 中找出能张成 V 的所有可能的基向量子集。

8.9 有多种方法可以找到矩阵 A 的齐次解空间的基, 不同方法得到的结果也不同。

$$(a) \text{ 设} \quad V = \begin{bmatrix} 3 & 4 & -6 & 4 \\ -4 & 0 & 0 & 8 \\ -3 & -2 & 3 & 1 \\ 2 & 6 & -9 & 11 \end{bmatrix},$$

用两种方法来求其齐次解空间的基。(i) 用 MATLAB 中的 null 命令; (ii) 用 ATLAST 中的 nulbasis 命令;

(b) 用前面学过的方法检验 (a) 中的两个基向量组是否确实张成了同样的子空间。

8.10 求下列诸向量的范数, 再用 plotangle 命令画出下列诸向量对之间的夹角。

$$(a) \quad x = \begin{bmatrix} 3 \\ 1 \\ 3 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 2 \end{bmatrix}; \quad (b) \quad x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 2 \\ 1 \end{bmatrix}; \quad (c) \quad x = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 2 \end{bmatrix};$$

8.11 对以下各组 \mathbf{R}^4 空间中的向量, 计算它们的 $\|x\|^2, \|y\|^2, \|x+y\|^2$, 并用 plotangle 命令画出此两个向量之间的夹角。

$$(i) \quad x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad (ii) \quad x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ -2 \\ -3 \\ 4 \end{bmatrix}, \quad (iii) \quad x = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 1 \end{bmatrix}, \quad (iv) \quad x = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ -1 \end{bmatrix},$$

(a) 分别用定义计算和用 MATLAB 命令计算, 进行比较。

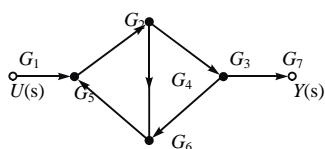
(b) 做猜想, 在什么情况下 $\|x+y\|^2 = \|x\|^2 + \|y\|^2$, 再给出代数的证明。(提示:

$$\|x+y\|^2 = (x+y)^T(x+y)。$$

- 8.12 设某经济体有三个部门: 化工, 动力和机械制造。化工部门把它产出的 30% 卖给动力部门, 50% 卖给机械部门, 其余自己留用。动力部门把它产出的 80% 卖给化工部门, 10% 卖给机械部门, 其余自己留用。机械部门把它产出的 40% 卖给动力部门, 40% 卖给化工部门, 其余自己留用。

(a) 列出此经济体的交换表。

(b) 求出此经济体的平衡价格。



题 8.13 图 某线性系统的信号流图

- 8.13 设某线性系统的信号流图如题 8.13 图所示, 输入信号为 u , 输出信号为 y , 请自行在四个中间节点上标注信号 x_1, x_2, x_3, x_4 , 然后

(a) 列出此系统的线性方程组。

(b) 将此线性方程组写成 $Ax=b$ 的标准矩阵形式。

(c) 用 $x=AB$ 解此方程, 求出输出 y 与输入 u 之比, 即

系统传递函数 (提示: 要把 $G_1, G_2, G_3, G_4, G_5, G_6, G_7$ 设为符号变量)。

- 8.14 设直线方程的形式为: $y=c_1+c_2x+c_3x^2$, 用它以最小二乘法逼近以下数据点, 请列写方程并求出 c_1, c_2 和 c_3 的值, 画出这些点和该逼近直线。

(a) (0, 1), (1, 2), (2, 4), (3, 6) 四点

(b) (1, 0), (2, 1), (4, 2), (5, 3) 四点

(c) (-1, 0), (0, 1), (1, 2), (2, 4) 四点

(d) (2, 3), (3, 2), (5, 1), (6, 0), (7, 0) 五点

- 8.15 已知健康孩子的心脏收缩血压 p 【毫米汞柱】与他的体重 w 【公斤】之间应该有下列近似关系: $\beta_0 + \beta_1 \ln w = p$ 。现有的统计结果为:

w	20	30	40	50	60
ln w	3.00	3.40	3.69	3.91	4.09
p	91	99	105	110	112

(a) 根据以上统计数据来确定 β_0, β_1 的值。

(b) 对于体重为 45【公斤】的孩子, 其收缩压的标准值应为多少?

线性变换及其特征

9.1 平面上线性变换的几何意义

把方程: $Ax=y$ (9.1)

中的 x 看成输入变量, y 看作输出变量, 则这个矩阵方程就代表了一种线性变换。不同的矩阵 A 可以产生不同的变换结果, 这在工程中有广泛的用途。

【例 9.1】 设 x 为二维平面上第一象限中的一个单位方块, 其四个顶点为 $(0, 0)$, $(1, 0)$, $(1, 1)$, $(0, 1)$ 。写成

$$x = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

把不同的 A 矩阵作用于此组数据, 可以得到多种多样的结果。假定 A 是 2×2 矩阵:

$$(1) \text{ 设 } A_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad y_1 = \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad (9.2)$$

$$(2) \text{ 设 } A_2 = \begin{bmatrix} 1.5 & 0 \\ 0 & 1 \end{bmatrix}, \quad y_2 = \begin{bmatrix} 0 & 1.5 & 1.5 & 0 \\ 0 & 0 & 1.0 & 1.0 \end{bmatrix} \quad (9.3)$$

$$(3) \text{ 设 } A_3 = \begin{bmatrix} 1.0 & 0 \\ 0 & 0.2 \end{bmatrix}, \quad y_3 = \begin{bmatrix} 0 & 1.0 & 1.0 & 0 \\ 0 & 0 & 0.2 & 0.2 \end{bmatrix} \quad (9.4)$$

$$(4) \text{ 设 } A_4 = \begin{bmatrix} 1.0 & 0.5 \\ 0 & 1.0 \end{bmatrix}, \quad y_4 = \begin{bmatrix} 0 & 1.0 & 1.5 & 0.5 \\ 0 & 0 & 1.0 & 1.0 \end{bmatrix} \quad (9.5)$$

(5) 设 $t = \pi/6$

$$A_5 = \begin{bmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{bmatrix}, y_5 = \begin{bmatrix} 0 & 0.8660 & 0.3660 & -0.5000 \\ 0 & 0.5000 & 1.3660 & 0.8660 \end{bmatrix} \quad (9.6)$$

可以用 MATLAB 来完成计算和绘图，其核心语句如下。其中把 y_2, y_3, y_4, y_5 的绘图语句中的坐标设置和标题语句都作了省略，读者不难从 y_1 的绘图语句推断，在本书可下载的程序 ag901 中，则完整地包含了全部语句。

```
x=[0,1,1,0;0,0,1,1];
subplot(2,3,1),
fill([x(1,:),0],[x(2,:),0],'r')
axis equal,axis([-1.5,1.5,-1,2]),grid on
A1=[-1,0;0,1]
y1=A1*x
fill([y1(1,:),0],[y1(2,:),0],'g')
A2=[1.5,0;0,1]
y2=A2*x
y3=A3*x
A3=[1,0;0,0.2]
y4=A4*x
A4=[1,0.5;0,1]
y5=A5*x
A5=[cos(t),-sin(t);sin(t),cos(t)]
```

运行这个程序可以得到前面列出的 y_1, y_2, y_3, y_4, y_5 的值，它们与 x 一样都用 2×4 的矩阵表示，同时得到图 9.1 所示的图形。

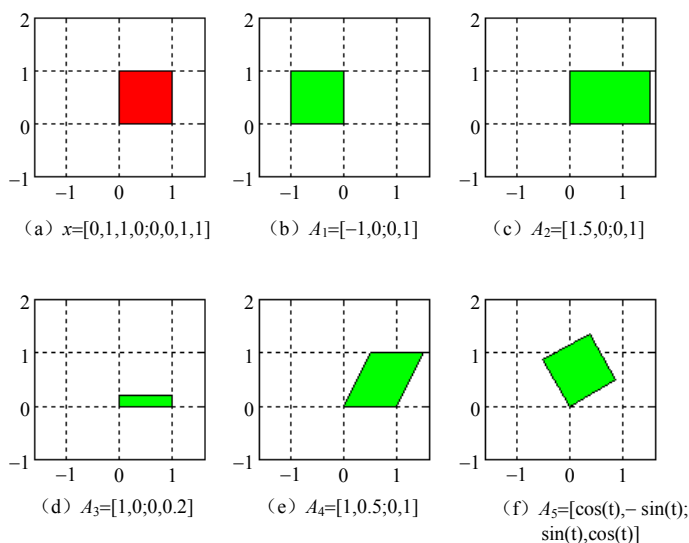


图 9.1 对单元方格进行几种线性变换后生成的图形

可以看出, 矩阵 A_1 使原图对纵轴生成镜像, 矩阵 A_2 使原图在横轴方向膨胀, 矩阵 A_3 使原图在纵轴方向压缩, 矩阵 A_4 使原图向右方剪切变形, 矩阵 A_5 使原图沿反时针方向旋转 $t=\pi/6$ 。为了进一步看出矩阵的特征和它们变换的效果之间的关系, 我们分别计算出这五个矩阵的行列式和特征值。

使用 MATLAB 时, 行列式用 $D_i=\det(A_i)$ 求得, 特征值 λ 和特征向量 p 则用 $[p_i, \text{lamdai}]=\text{eig}(A_i)$ 计算, 算得的结果如下:

$$D1 = \det(A1) = -1, \text{ lamda1} = \begin{bmatrix} -1 & 1 \end{bmatrix}, p1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$D2 = \det(A2) = 1.5, \text{ lamda2} = \begin{bmatrix} 1.0 & 1.5 \end{bmatrix}, p2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$D3 = \det(A3) = 0.2, \text{ lamda3} = \begin{bmatrix} 0.2 & 1.0 \end{bmatrix}, p3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$D4 = \det(A4) = 1, \text{ lamda4} = \begin{bmatrix} 1 & 1 \end{bmatrix}, p4 = \begin{bmatrix} 1.0 & -1.0 \\ 0. & 0. \end{bmatrix}$$

$$D5 = 1, \text{ lamda5} = \begin{bmatrix} 0.866 + 0.5i & 0.866 - 0.5i \end{bmatrix}, p5 = \begin{bmatrix} 0.7071 & 0.7071 \\ 0 - 0.7071i & 0 + 0.7071i \end{bmatrix}$$

对二维空间(平面), 行列式的几何意义实际上是两个向量所构成的平行四边形的面积。一个变换作用于某图形所造成的新图形的面积变化, 就取决于该变换的行列式。可以看出, A_1 , A_4 和 A_5 的行列式绝对值都是 1, 所以它们不会使变换后图形的面积发生改变。而 A_2 和 A_3 的行列式分别为 1.5 和 0.2, 变换后图形面积的增加和减小倍数恰好与这两个值相对应。

9.2 二维矩阵特征值的几何意义

二维矩阵的特征值表示该变换在原图形的特征向量的方向上的放大量。

例如矩阵 A_1 在第一特征向量 $p_1(:,1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ 的方向的特征值为 $\lambda_1(1) = -1$, 即横轴正方向的增益为 -1, 其结果是把原图中横轴正方向的部分变换到新图的负方向去了; A_1 在第二特征向量 $p_1(:,2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 的方向的特征值为 $\lambda_1(2) = 1$, 即纵轴正方向的增益为 1, 因而保持了新图 and 原图在纵轴方向尺度不变。

可以再拿 A_4 来分析一下, 它在第一特征向量 $p_5(:,1) = \begin{bmatrix} 0.7071 \\ 0 - 0.7071i \end{bmatrix}$ 方向的特征值为

$\lambda_5(1) = 0.866 + 0.5i$; 第二特征向量 $p_5(:,2) = \begin{bmatrix} 0.7071 \\ 0 + 0.7071i \end{bmatrix}$ 方向的特征值为 $\lambda_5(2) = 0.866$

$-0.5i$, 这两个特征值是复数, 但它们的绝对值都是 1, 说明原图形和新图形各对应向量的大小是相同的, 其差别只是一个相角, 图 9.1 中反映的也确实是这样情况。

对于比较复杂、带有形状改变和复数特征值等情况，完全凭简单的几何关系去想象是困难的，我们建议读者输入 `eigshow(A4)` 和 `eigshow(A5)`，联系 \mathbf{x} 和 \mathbf{Ax} 的向量图来加以思考。图 9.2 左图就是 `eigshow(A4)` 的结果。

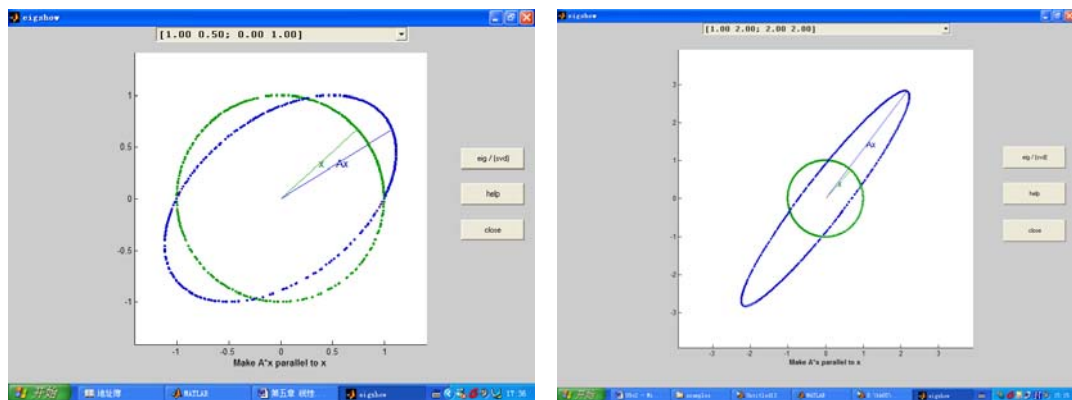


图 9.2 矩阵 \mathbf{A} 的特征向量和特征值演示（左图 $\mathbf{A}=\mathbf{A}_4$ ，右图 $\mathbf{A}=[1,2,2,2]$ ）

绿色的 \mathbf{x} 表示原坐标系中的单位向量，可以用鼠标左键点住 \mathbf{x} 并拖动它围绕原点转动。图中同时出现以蓝色表示的 \mathbf{Ax} 向量，它表示变换后的新向量。 \mathbf{Ax} 与 \mathbf{x} 在长度和相角上的不同就表示了该变换在这个向量方向的幅度增益和相角增量。当两个向量处在同一条直线上时（包括同向和反向），表示两者之间的相位相同，只存在一个（可正可负的）实数乘子 λ ，即

$$\mathbf{Ax} = \lambda \mathbf{x}$$

这时的向量 \mathbf{x} 就称为特征向量，而对应的乘子 λ 就是特征值。在这个图中，当 \mathbf{x} 转到 ± 1 的水平位置时， \mathbf{Ax} 也恰好与 \mathbf{x} 重合，并具有同样的长度，这就是对 \mathbf{p}_4 和 λ_4 的合理性的解释。至于图形的倾斜，只靠特征值和特征向量就无法解释了，必须观察整个 $\mathbf{x} \sim \mathbf{Ax}$ 的向量关系。例如在 \mathbf{x} 位于 45° 处（见图 9.2 左图）， \mathbf{Ax} 变得很长，而且偏右上方，这就说明了原来单位方格的对角线被拉长并偏向右上方，形成了剪切现象。

特别要注意 \mathbf{A} 是对称实矩阵的情况，所谓对称矩阵是满足 $\mathbf{A}^T = \mathbf{A}$ 的矩阵。对 2×2 矩阵，只要求 $\mathbf{A}(1,2) = \mathbf{A}(2,1)$ 。例如令 $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$ ，再输入 `eigshow(A)`，这时的 $\mathbf{Ax} = \lambda \mathbf{x}$ 出现在 \mathbf{Ax} 椭圆轨迹的主轴上，所以两个特征值分别对应于单位圆映射的椭圆轨迹的长轴和短轴（见图 9.2 右图）。这说明，如果把坐标系统改一下，转动一个角度，可以使椭圆的长短轴与新的坐标重合，这时，线性方程组将具有最简单的形式。

【例 9.2】 数据矩阵 $\mathbf{x} = \begin{bmatrix} 0 & 0.50 & 0.50 & 6.00 & 6.00 & 5.50 & 5.50 & 0 \\ 0 & 0 & 6.42 & 0 & 8.00 & 8.00 & 1.58 & 8.00 \end{bmatrix}$ 表示英文大写字母 N 图形的各个节点，要求：

(1) 用 `plot` 语句在图 9.2 左图中正确地画出其形状；

(2) 取 $\mathbf{A} = \begin{bmatrix} 1 & 0.25 \\ 0 & 1 \end{bmatrix}$ 作为变换矩阵对 \mathbf{x} 进行变换，并在图 9.2 右图中画出其图形；

(3) 对结果进行讨论。

解：本题画图的要点是要在给定的数据右方，补上第一点的坐标，使画出的图形封闭。编成的程序 ag902 如下：

```
x0=[0,0.5,0.5,6,6,5.5,5.5,0;0,0,6.42,0,8,8,1.58,8];
x=[x0,x0(:,1)]; % 把首顶点坐标补到末顶点后
A=[1,0.25;0,1]; y=A*x;
subplot(1,2,1),plot(x(1,:),x(2,:))
subplot(1,2,2),plot(y(1,:),y(2,:))
```

生成的图形如图 9.3 所示，这个例子说明，在设计计算机字库中，斜体字库可以不必单独建立，只要对正体字库进行适当的线性变换，就可以实现斜体字，用这个方法可以节约大量的人力，并可节约存储量。

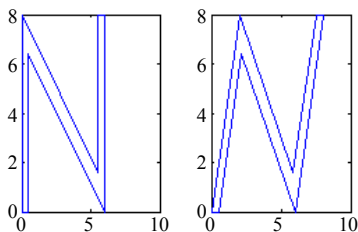


图 9.3 例 9.2 生成的 N 字符图形

以上我们讨论的是由 2×2 的变换矩阵的变换作用，它所实现的是同维空间（平面 \rightarrow 平面）之间的变换，实际上还有不同维空间之间的变换。比如 A 是 1×2 的变换矩阵 $A = \begin{bmatrix} 1 & 0 \end{bmatrix}$ ，用例 9.1 的 x 数据可得 $y = Ax = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$ 。这样原二维向量 x 被变换（映射）为一维的新向量。它的实质是把原来的方块图形投影到横坐标轴上。四个顶点变换（投影）为横坐标上的两个点 0 和 1。相当于 A_3 生成的图形 (d)，但把高度 0.2 缩减为零。

把二维向量变换为三维向量，即把平面上的向量变换成为空间向量，这在某些情况下是很有用的。比如刚体在平面上的运动要用两个平移和一个转动来描述，转动可以从上面的线性变换 A_5 得到，但平移 $y=x+c$ 却不是一个线性变换。因为：

(1) 设 $y_a = x_a + c$ ； $y_b = x_b + c$ ；则它们的和为

$$y = y_a + y_b = x_a + x_b + 2c \neq x + c,$$

可见，它对加法不封闭；

(2) 设 $y_a = x_a + c$ ；将它乘以常数 k ，

$$y = ky_a = k(x_a + c) = kx_a + kc \neq kx_a + c = x + c,$$

可见，它对乘法也不封闭；就是说，这不符合线性变换的规则， x 和 y 不属于同一个向量空间，无法用矩阵乘法来实现平移变换 $y=x+c$ 。

为了把刚体的运动完全用线性变换来描述，人们用增加空间维数的方法，把平面问题映射到高维的空间来建立方程，这就可能把 x 和 y 由扩展了的向量空间来涵盖。把原来通过原点的平面沿垂直方向提高一个单位，与原平面保持平行，于是原来的 x 就用三维向量来表示为：

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \quad (9.7)$$

这样的坐标系称为齐次坐标系。可以把平移矩阵写成：

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & c_1 \\ 0 & 1 & c_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.8)$$

于是

$$\mathbf{y} = \mathbf{M} \cdot \mathbf{x} = \begin{bmatrix} x_1 + c_1 \\ x_2 + c_2 \\ 1 \end{bmatrix} \quad (9.9)$$

可见三维齐次坐标中的前两个分量实现了平移运算的要求。顺便指出，从原来的坐标多加一维变为齐次坐标的操作是不能用线性变换实现的，不然，就说明全部过程都可以用矩阵连乘来实现，那就没必要把 \mathbf{R}^2 空间的问题扩展到 \mathbf{R}^3 中去处理了。

对象若同时有旋转和平移，则可以分别列出旋转矩阵和平移矩阵。不过此时的旋转矩阵也要改为 3×3 维，这可以把上述 \mathbf{A}_5 中增加第三行和第三列，置 $\mathbf{A}(3,3)=1$ ，其余新增元素为零。

$$\mathbf{A}_5 = \begin{bmatrix} \cos t & -\sin t & c_1 \\ \sin t & \cos t & c_2 \\ 0 & 0 & 1 \end{bmatrix}$$

这就是既包括平移，又包括转动的平面齐次坐标系内的变换矩阵。

【例 9.3】 设一个三角形的三个顶点坐标为 $(-1,1),(1,1),(0,2)$ ，今要使它旋转 30° ，右移 2，上移 3，以试设计变换矩阵 \mathbf{A} ，并画出变换前后的图形。

解：先列出数据矩阵，为了画图方便，直接把第一点的数据补到最后，构成 2×4 数据矩阵。根据前面的论述，有平移要求时，必须采用齐次坐标系，因此数据矩阵应该是 3 行的，即最后要补一行 1。平移和转动矩阵按上面的方法确定，于是其程序 ag903 如下：

```
x=[-1,1,0,-1;1,1,2,1;ones(1,4)] % 将平面坐标改为三维齐次坐标
plot(x(1,:),x(2,:)),hold on % 画出原始图形
axis([-2,4,0,6]),pause
M=[1,0,2;0,1,3;0,0,1] % 齐次坐标系中的移位矩阵
t=pi/6;
R=[cos(t),-sin(t),0;sin(t),cos(t),0;0,0,1] % 齐次坐标系中的转动矩阵
y1=R*x,pause % 求出转动后图形参数
fill(y1(1,:),y1(2:),'r') % 画填充红色图
y2=M*R*x,pause % 求出两次变换后图形参数
fill(y2(1,:),y2(2:),'g') % 画填充绿色图
```

程序运行中给出的数字显示为：

$$R = \begin{bmatrix} 0.8660 & -0.5000 & 0 \\ 0.5000 & 0.8660 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$y1 = \begin{bmatrix} -1.3660 & 0.3660 & -1.0000 & -1.366 \\ 0.3660 & 1.3660 & 1.7321 & 0.366 \\ 1.0000 & 1.0000 & 1.0000 & 1.000 \end{bmatrix}$$

$$y2 = \begin{bmatrix} 0.634 & 2.3660 & 1.0000 & 0.6340 \\ 3.366 & 4.3660 & 4.7321 & 3.3660 \\ 1.000 & 1.0000 & 1.0000 & 1.0000 \end{bmatrix}$$

得出的图形如图 9.4 所示。

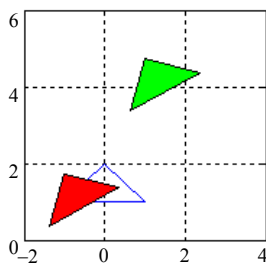


图 9.4 例 9.3 的两次变换图形

矩阵的相乘不符合交换律，线性变换也不遵守交换律。本例取的是先转动后平移，也即先用 R 左乘，再用 M 左乘。如果换了次序，先平移再转动，即把 R 和 M 作用次序交换一下，结果就完全不同了，读者可自行探讨和实验。在计算机图形学中也采用同样的方法来处理图形变换问题。

9.3 三维空间中线性变换的几何意义

三维空间线性变换最直接的几何意义和应用价值可以从飞行器的三维转动坐标中得到解释。飞行器在空中可以围绕三个轴旋转。假如它在向北飞行，机头正对北方，则它围绕铅垂轴的旋转角称为偏航角（Yaw），它描述了飞机左右的偏转，用 u 表示；围绕翼展轴的旋转角称为倾斜角（Pitch），它描述了飞机俯仰姿态，用 v 表示；围绕机身轴的旋转角称为滚角（Roll），用 w 表示； u, v 和 w 三个变量统称为欧拉角，它们完全描述了飞机的姿态。

MATLAB 中有一个演示程序 `quatdemo.m`，专门演示这几个姿态角所造成的飞机状态。输入：

```
>> quatdemo
```

屏幕上将出现如图 9.5 所示的画面。左方为飞行器在三维空间中的模型，其中红色的是飞行器。右上方为三个姿态角 u, v, w 的设定标尺和显示窗，右下方为在地面坐标系中的另外三个姿态角：方位角、俯仰角和倾侧角。左下方还有【静态】和【动态】两个复选钮，我们只介绍【静态】，读者可自行试用【动态】进行演示。

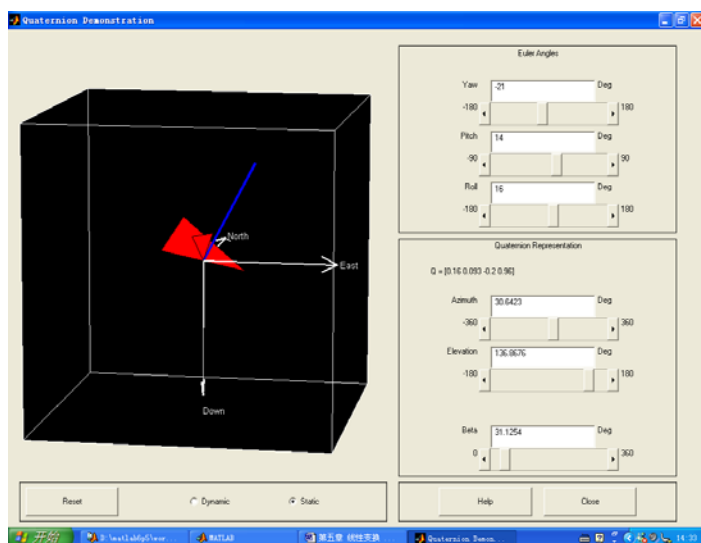


图 9.5 飞行器姿态角演示

用输入参数或移动标尺的方法分别给 u, v, w 赋值并回车后，就可以得出相应的飞行器姿态，同时出现一根蓝色的线表示合成旋转的转轴。

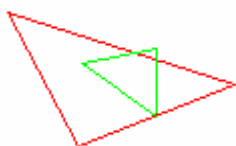


图 9.6 用数据集 G 画出的飞行器

【例 9.4】 用数值来讨论这个程序的实现方法。

先把飞行器的三维图像用 N 个顶点的三维坐标描述，写成一个 $3 \times N$ 的数据矩阵 G 。其顶点次序要适当安排，使得用 `plot3` 命令时按顶点连线能绘制出飞行器的外观。例如以下的程序（ag904 前半部分）即可画出一个最简单的飞行器立体图，如图 9.6 所示。

```
Gw=[-4,-3,0;4,-3,0;0,7,0;-4,-3,0]'; % 主翼的顶点坐标
Gt=[0,-3,0;0,-3,3;0,2,0;0,-3,0]'; % 尾翼的顶点坐标
G=[Gw,Gt] % 整个飞行器外形的数据集
plot3(Gw(1,:),Gw(2,:),Gw(3:,:), 'r'), hold on
plot3(Gt(1,:),Gt(2,:),Gt(3:,:), 'g'),
axis equal
```

运行此程序得出整个飞行器外形的数据集为

$$G = \begin{bmatrix} -4 & 4 & 0 & -4 & 0 & 0 & 0 & 0 \\ -3 & -3 & 7 & -3 & -3 & -3 & 2 & -3 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \end{bmatrix} \quad (9.10)$$

飞行器围绕各个轴的旋转的结果, 表现为各个顶点坐标发生变化, 也就是 \mathbf{G} 的变化。只要把三种姿态的变换矩阵 \mathbf{Y} , \mathbf{P} 和 \mathbf{R} 乘以图形数据矩阵 \mathbf{G} 即可。其中

$$\mathbf{Y} = \begin{bmatrix} \cos(u) & \sin(u) & 0 \\ -\sin(u) & \cos(u) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.11)$$

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(w) & -\sin(w) \\ 0 & \sin(w) & \cos(w) \end{bmatrix} \quad (9.12)$$

$$\mathbf{P} = \begin{bmatrix} \cos(v) & 0 & -\sin(v) \\ 0 & 1 & 0 \\ \sin(v) & 0 & \cos(v) \end{bmatrix} \quad (9.13)$$

单独变化某个姿态角所生成的图形由 $\mathbf{G}_1 = \mathbf{Y} * \mathbf{G}$, $\mathbf{G}_2 = \mathbf{P} * \mathbf{G}$, $\mathbf{G}_3 = \mathbf{R} * \mathbf{G}$ 算出, 如果同时变化三个姿态角, 则最后的图像数据成为 $\mathbf{G}_f = \mathbf{Y} * \mathbf{P} * \mathbf{R} * \mathbf{G} = \mathbf{Q} * \mathbf{G}$ 。这里假定转动的次序为: 先滚动 \mathbf{R} , 再倾斜 \mathbf{P} , 最后偏航 \mathbf{Y} , 由于矩阵乘法不服从交换律, 转动次序不同时结果也不同。最后变换矩阵成为

$$\mathbf{Q} = \begin{bmatrix} \cos(u)*\cos(v) & \sin(u)*\cos(w) - \cos(u)*\sin(v)*\sin(w) & -\sin(u)*\sin(w) - \cos(u)*\sin(v)*\cos(w) \\ -\sin(u)*\cos(v) & \cos(u)*\cos(w) + \sin(u)*\sin(v)*\sin(w) & -\cos(u)*\sin(w) + \sin(u)*\sin(v)*\cos(w) \\ \sin(v) & \cos(v)*\sin(w) & \cos(v)*\cos(w) \end{bmatrix} \quad (9.14)$$

这个过程可以用手工计算, 也可以用 MATLAB 程序来实现。此程序 ag904 后半部分如下:

```
syms u w v
Y=[cos(u),sin(u),0;-sin(u) cos(u),0;0,0,1]
R=[1,0,0;0,cos(w),-sin(w);0,sin(w),cos(w)]
P=[cos(v),0,-sin(v);0,1,0;sin(v),0,cos(v)]
Q=Y*P*R
```

这里采用了符号运算工具箱以得到普遍的公式, 当设定了 u, v, w 的具体数值后, 比如 $u = \pi/4, v = 0, w = \pi/3$, 要求出 \mathbf{Q} 矩阵的数字结果时, 要用 subs (代换) 命令如下:

```
A=subs(Q,{u,v,w},{pi/4,0,pi/3})
```

运行结果为:

$$\mathbf{A} = \begin{bmatrix} 0.7071 & 0.3536 & -0.6124 \\ -0.7071 & 0.3536 & -0.6124 \\ 0 & 0.8660 & 0.5000 \end{bmatrix}$$

我们知道, 二维变换矩阵的行列式代表的是两个向量组成的平行四边形的面积, 三维

变换矩阵的行列式代表的是三个向量组成的平行六面体的体积。不难算出, $\det(\mathbf{Y})=1$, $\det(\mathbf{R})=1$, $\det(\mathbf{P})=1$, $\det(\mathbf{Q})=1$ 。当然也有 $\det(\mathbf{A})=1$ 。说明这几个变换都不会改变图形对象的体积。这是描述刚体运动的一个必须遵守的原则。不仅不允许改变体积, 而且不允许改变形状, 后者的要求反映为要求其变换的特征值必须等于 1。如果特征值是复数, 则它们的绝对值必须为 1。至于对特征向量的要求, 读者可从二维情况中的 \mathbf{A}_5 得到启发和推论。因为求特征值的 MATLAB 函数 `eig` 还不能用于符号函数, 检验上述结论只能用数值矩阵 \mathbf{A} 。输入

```
[p,lamda]=eig(A)
```

```
得到 p = -0.7701          0.1833 - 0.4122i      0.1833 + 0.4122i
          0.3190          0.6702              0.6702
          0.5525        -0.1315 - 0.5745i      -0.1315 + 0.5745i
lamda = 1.0000          0                    0
          0            0.2803 + 0.9599i        0
          0            0                    0.2803 - 0.9599i
```

再输入 `abs(lamda)`, 得到

```
ans = 1.0000          0          0
       0            1.0000      0
       0            0          1.0000
```

说明所有的特征值绝对值均为 1。为了计算各特征向量的长度, 可以输入 $\mathbf{p}'*\mathbf{p}$, 得到

```
ans = 1.0000          0.0000 - 0.0000i      0.0000 + 0.0000i
       0.0000 + 0.0000i      1.0000          -0.0000 - 0.0000i
       0.0000 - 0.0000i     -0.0000 + 0.0000i      1.0000
```

可见其特征向量的长度也均为 1。

飞行器在空中的运动应该由六个自由度, 其中三个是转动, 三个是平移, 要完整地描述这些运动, 三维空间和 3×3 的变换矩阵是肯定不够的。所以需要研究三维以上的线性空间和线性变换问题。就本题来看, 研究的对象不是整个飞行器, 而是飞行器外形上的 N 个顶点。这些顶点用在三维空间中的三个坐标来描述, 也就是 $3 \times N$ 数据集 \mathbf{G}_3 。考虑了平移运动后, 如同二维情况那样, 也必须扩展一维, 成为 $4 \times N$ 数据集 \mathbf{G}_4 , 在四维空间来建立数据组和 4×4 变换矩阵。即数据组成为

$$\mathbf{G}_4 = \begin{bmatrix} \mathbf{G}_3 \\ \text{ones}(1, N) \end{bmatrix}, \quad (9.15)$$

而平移变换矩阵 \mathbf{M} 和旋转变换矩阵 $\mathbf{Y}, \mathbf{R}, \mathbf{P}$ 成为

$$\mathbf{M}_4 = \begin{bmatrix} 1 & 0 & 0 & c_1 \\ 0 & 1 & 0 & c_2 \\ 0 & 0 & 1 & c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (9.16)$$

$$Y_4, (R_4, P_4) = \begin{bmatrix} Y_3, (R_3, P_3) & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9.17)$$

其中 c_1, c_2, c_3 为在三个轴 x_1, x_2, x_3 方向上的平移距离。这种方法在机器人运动学研究中很有用处。

9.4 基变换与坐标变换

因为观察研究问题的角度不同, 在线性空间中常常需要进行坐标变换。例如上面介绍的飞行器, 在设计机上控制系统时, 常用的是机身坐标, 而在研究它相对于地面的运动时, 则要用地面坐标。这种坐标变换, 也就反映为基的变换。由于基的不同, 同一个向量在两个坐标系中的坐标值也不同。

用图 9.7 可以形象地说明这点。按照左图的标准基 $[e_1, e_2]$, x 向量应该表为 $x = \begin{bmatrix} 1 \\ 6 \end{bmatrix}$, 坐标 1 和 6 表明 x 按标准基 $[e_1, e_2]$ 度量的结果, 在 e_1 方向为 1 个单位而在 e_2 方向为 6 个单位。

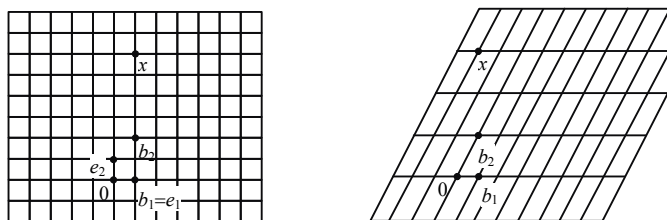


图 9.7 基的改变对坐标的影响。左: 标准坐标; 右: 斜坐标

同样的点换成右图所示的斜坐标纸, 情况就不同了。在斜坐标纸上的 x 点坐标就成为沿 b_1 方向为 -2 个单位而沿 b_2 方向为 3 个单位, 即 $x_s = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$ 了。这反映了不同的基对坐标值的影响。

设线性空间 \mathbf{R}^n 中的两组基向量 $u = [u_1, u_2, \dots, u_n]$, $v = [v_1, v_2, \dots, v_n]$, u_1, u_2, \dots, u_n 和 v_1, v_2, \dots, v_n 都是 n 维列向量, 它们在基准坐标系中的 n 个分量都是已知的, 因此 u 和 v 都可表示为 $n \times n$ 矩阵。如果 \mathbf{R}^n 中的一个向量 w 在以 u 为基的坐标系内的坐标为 w_u ($n \times 1$ 数组), 在以 v 为基的坐标系内的坐标为 w_v ($n \times 1$ 数组), 它们在基准坐标系内的坐标应分别为 $u^* w_u$ 和 $v^* w_v$, 这两者应该相等。

$$u^* w_u = v^* w_v \quad (9.18)$$

所谓基坐标的变换就是已知 w_u , 求出 w_v 。将上面的等式左右均左乘以 $\text{inv}(v)$

$$w_v = \text{inv}(v)^* u^* w_u = v \setminus u^* w_u \quad (9.19)$$

可见, 坐标变换矩阵可由 u 和 v 求得:

$$P(u \rightarrow v) = v \setminus u \quad (9.20)$$

有了 $P(u \rightarrow v)$ 以后, 知道基向量 u 中的坐标 w_u , 便很容易求出在基向量 v 中的坐标 w_v

$$w_v = P(u \rightarrow v)^* w_u \quad (9.21)$$

【例 9.5】 已知 \mathbf{R}^4 空间的两组基向量 \mathbf{u}, \mathbf{v} 如下:

$$\mathbf{u} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & -1 & 2 & -1 \\ -1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 2 & 0 & -2 & 1 \\ 1 & 1 & 1 & 3 \\ 0 & 2 & 1 & 1 \\ 1 & 2 & 2 & 2 \end{bmatrix}$$

试求把基向量 \mathbf{u} 变换为基向量 \mathbf{v} 的坐标变换矩阵。

解: 输入 \mathbf{u} 和 \mathbf{v} 矩阵后, 输入 $\mathbf{P}=\mathbf{v} \backslash \mathbf{u}$, 程序 ag905 成为

$\mathbf{u}=[1,1,-1,-1;2,-1,2,-1;-1,1,1,0;0,1,1,1]$

$\mathbf{v}=[2,0,-2,1;1,1,1,3;0,2,1,1;1,2,2,2]$

$\mathbf{P}=\mathbf{v} \backslash \mathbf{u}$

得到:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & -1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

设

$$\mathbf{w}\mathbf{u} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix},$$

则 $\mathbf{w}\mathbf{v}$ 可计算如下:

$$\mathbf{w}\mathbf{v} = \mathbf{P} * \mathbf{w}\mathbf{u} = \begin{bmatrix} 3 \\ 1 \\ 4 \\ -2 \end{bmatrix}$$

同理可以求得由基向量 \mathbf{v} 变换为基向量 \mathbf{u} 的坐标变换矩阵 $\mathbf{P}(\mathbf{v} \rightarrow \mathbf{u})$ 如下:

$$\mathbf{P}(\mathbf{v} \rightarrow \mathbf{u}) = \mathbf{P}(\mathbf{u} \rightarrow \mathbf{v})^{-1} = \mathbf{u} \backslash \mathbf{v} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

9.5 特征值和特征向量的 MATLAB 求法

MATLAB 提供了计算方阵的特征值和特征向量各步骤的函数。这三个步骤是:

(1) 用 $\mathbf{f}=\text{poly}(\mathbf{A})$ 可以计算方阵 \mathbf{A} 的特征多项式系数向量 \mathbf{f} ;

(2) 用 $\text{lamda}=\text{roots}(f)$ 可以求特征多项式 f 的全部根 lamda (表示为列向量);

(3) 用函数 $p=\text{null}([\text{lamda}*\mathbf{I}-\mathbf{A}])$ 直接给出基础解 p , 将 n 个特征列向量 p 排在一起, 就是 \mathbf{A} 的特征向量矩阵。

【例 9.6】 求矩阵 $\mathbf{A}=\begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix}$ 的特征值和特征向量。

按上面所说的步骤, 解题的程序 ag906 为

```
A=[3,2,4;2,0,2;4,2,3]; % 输入系数矩阵
f=poly(A), % 求特征多项式(步骤1)
r=roots(f),r=real(r) % 求特征根(步骤2),并去掉误差造成的虚部
B1= r(1)*eye(3)-A; % 后面可能需要插入一条语句
% B1=rref(B1,1e-12), % 为保证矩阵 B1 奇异性而设的语句
p1=null(B1,'r') % 把第一个特征值代入方程,求其基础解系
B2=r(2)*eye(3)-A; p2=null(B2,'r') % 求第二个特征值的基础解系
B3=r(3)*eye(3)-A; p3=null(B3,'r') % 求第三个特征值的基础解系
```

程序运行的结果为:

```
f = 1.0000 -6.0000 -15.0000 -8.0000 (特征多项式系数向量)
r = 8.0000 (三个特征根即特征值,后两个是重根)
-1.0000 + 0.0000i (微小虚数可用 r=real(r) 去除)
-1.0000 - 0.0000i
```

对第一个特征值 8 的特征向量为空矩阵, $p_1 = \text{Empty matrix: 3-by-0}$, 即无特征向量。

对第二、三个特征值 (重根) -1 的特征向量为

$$p_2 = \begin{bmatrix} -0.5000 & -1.0000 \\ 1.0000 & 0 \\ 0 & 1.0000 \end{bmatrix}$$

对第一个特征值所以找不到特征向量是由计算误差引起的, 计算误差使得本应为零的 $\det(\mathbf{B}_1)$ 成了一个很小的数 ($-8.7130\text{e-}013$), $\mathbf{B}_1\mathbf{x}=\mathbf{0}$ 就没有非零解了。解决的方法是将 \mathbf{B}_1 化为行阶梯形式, 因为在 rref 函数中可以人为地设定容差, 例如把容差设为 1×10^{-12} , 令 $\mathbf{B}_1=\text{rref}(\mathbf{B}_1,1\text{e-}12)$, 这样变换出的行阶梯形式与原矩阵 \mathbf{B}_1 是等价的, 但它将会把一些小的数看作零, 从而保证系数矩阵的奇异性。在上述程序中求 \mathbf{B}_1 和求 p_1 的两条语句之间, 插入这条语句后执行, 就可求出对应的特征向量为 $p_1=[1,0.5,1]^T$ 。把上述三个特征向量并列, 可以得到特征向量矩阵 (元素取整后) 为:

$$p=[p_1,p_2]=\begin{bmatrix} 2.0000 & -1.0000 & -1.0000 \\ 1.0000 & 2.0000 & 0 \\ 2.0000 & 0 & 1.0000 \end{bmatrix}$$

把特征值也对应地以对角矩阵的形式列出

$$\mathbf{lamda} = \begin{bmatrix} 8.0000 & 0 & 0 \\ 0 & -1.0000 & 0 \\ 0 & 0 & -1.0000 \end{bmatrix}$$

实际上 MATLAB 已经把求特征根和特征向量的步骤集成化，其中也包括了处理计算误差的功能，所以一条命令就解决问题了。这个功能强大的子程序名为 `eig` (特征值英文是 `eigenvalue`，特征向量英文是 `eigenvector`)，调用的形式是：

```
[p, lamda]=eig(A)
```

输出变元中的 \mathbf{lamda} 是特征值， \mathbf{p} 是特征向量。把例 9.6 的系数矩阵 \mathbf{A} 代入，即可得到：

$$\mathbf{p} = \begin{bmatrix} -0.4941 & -0.5580 & 0.6667 \\ -0.4720 & 0.8161 & 0.3333 \\ 0.7301 & 0.1500 & 0.6667 \end{bmatrix}, \quad \mathbf{lamda} = \begin{bmatrix} -1.0000 & 0 & 0 \\ 0 & -1.0000 & 0 \\ 0 & 0 & 8.0000 \end{bmatrix}$$

读者可能奇怪，解出的特征向量怎么与前面计算的不同？其实特征向量本来就不是唯一的，这两个特征向量只是相当于基础解系的一组基，基础解平面可以由无数种基组成。我们只要检查由两种方法算出的两组特征向量是否相关，如果相关，说明解都是正确的。为此，可输入 `rank(p2)` 和 `rank([p2, p(:, [1,2])])`，结果两个秩都是 2。说明 \mathbf{p}_2 中的两个列向量所张成的平面，与 \mathbf{p} 中对应于特征值 -1 两个列向量处在同一平面中，所以两者相互可以用线性组合表达，都是正确的。

【例 9.7】 设矩阵 $\mathbf{A} = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & -1 \end{bmatrix}$ ，

求以下矩阵的特征值；a) \mathbf{A} ；b) $2\mathbf{A}^3 + \mathbf{A} - 5\mathbf{I}$ ；c) $\mathbf{I} + \mathbf{A}^{-1}$ 。

本题只要求出特征方程和特征根，它的计算程序 `ag907` 是比较简单的：

```
A=[1,-1,2;0,2,1;0,0,-1];
B=2*A^3+A-5*eye(3);C=inv(A)+eye(3)
fa=poly(A),ra=roots(fa)
fb=poly(B),rb=roots(fb)
fc=poly(C),rc=roots(fc)
```

程序运行结果为： $\mathbf{A} = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & -1 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} -2 & -15 & 2 \\ 0 & 13 & 7 \\ 0 & 0 & -8 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 2.0 & 0.5 & 2.5 \\ 0 & 1.5 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}$

$\mathbf{fa} = [1 \ -2 \ -1 \ 2]$, $\mathbf{fb} = [1 \ -3 \ -114 \ -208]$, $\mathbf{fc} = [1.00 \ -3.50 \ 3.00 \ 0]$

这意味着三个特征多项式分别为:

$$f_a(\lambda) = \lambda^3 - 2\lambda^2 - \lambda + 2, \quad f_b(\lambda) = \lambda^3 - 3\lambda^2 - 114\lambda - 208, \quad f_c(\lambda) = \lambda^3 - 3.5\lambda^2 + 3\lambda,$$

相应的各组特征根为:

$$\text{ra} = \begin{bmatrix} -1.0000 \\ 2.0000 \\ 1.0000 \end{bmatrix}, \quad \text{rb} = \begin{bmatrix} 13.0000 \\ -8.0000 \\ -2.0000 \end{bmatrix}, \quad \text{rc} = \begin{bmatrix} 0 \\ 2.0000 \\ 1.5000 \end{bmatrix}$$

9.6 对称矩阵对角化与二次型主轴

对称矩阵的特点是元素关于主对角线对称, 即 $\mathbf{A}' = \mathbf{A}$ 。所以对称矩阵一定是方阵。在图 9.2 中, 我们曾要求读者特别注意 \mathbf{A} 是对称矩阵时 \mathbf{x} 与 \mathbf{Ax} 的对应关系, 其特点就是 \mathbf{Ax} 呈椭圆形状, 在椭圆的两个主轴方向, \mathbf{Ax} 与 \mathbf{x} 在一条直线上方向相同或相反, 长度差 λ 倍, 即 $\mathbf{Ax} = \lambda \mathbf{x}$ 。当 \mathbf{Ax} 与 \mathbf{x} 方向相同时, λ 为正数; 当 \mathbf{Ax} 与 \mathbf{x} 方向相反时, λ 为负数; 2×2 实对称矩阵有两个特征值, 在相互正交的两个主轴方向, 各有一个 λ 。

作为 2×2 正交变换的一个应用, 我们来看看它对二次型图形的影响。二次型本身已经不是线性范围。我们要研究的是基坐标做了线性变换后会对二次型图形发生何种影响。

【例 9.8】 设 $\mathbf{A} = \begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix}$, 则令 \mathbf{A} 的二次型等于常数, 则得到的方程为:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 5x_1^2 - 4x_1x_2 + 5x_2^2 = 48$$

是一个椭圆方程, 其图形如图 9.8 (a) 图所示。

如果做一个基坐标的旋转变换, 让坐标轴转过 45° , 这个椭圆的主轴就与新的坐标方向 y_1, y_2 相同, 如图 9.8 (b) 图所示, 其方程将变为标准型椭圆方程。从解析几何知旋转变换关系为:

$$\begin{aligned} y_1 &= x_1 \cos \theta + x_2 \sin \theta \\ y_2 &= -x_1 \sin \theta + x_2 \cos \theta \end{aligned}$$

写成矩阵乘法 $\mathbf{y} = \mathbf{P} \mathbf{x}$

其中

$$\mathbf{P} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

或取其逆变换, 写成 $\mathbf{x} = \mathbf{R} \mathbf{y} = \mathbf{P}^{-1} \mathbf{y}$

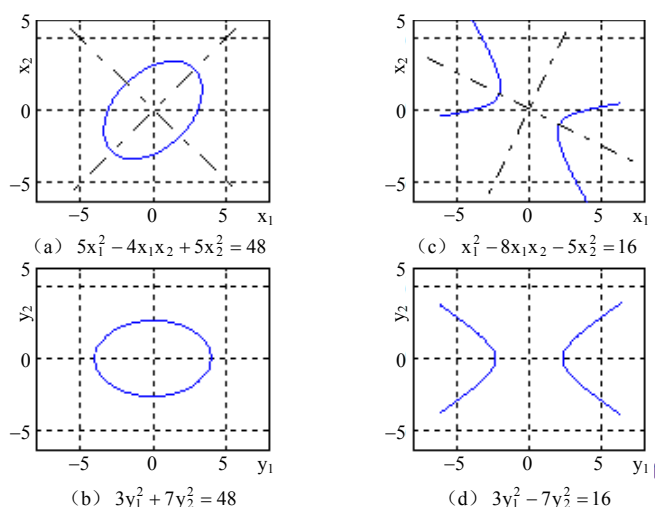


图 9.8 两种二次型经坐标变换到主轴方向

其中

$$\mathbf{R} = \text{inv}(\mathbf{P}) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

用此变换式代入二次型的表达式，由于单位正交矩阵 \mathbf{R} 有 $\mathbf{R}^{-1} = \mathbf{R}^T$ ，故得到：

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{y}^T \mathbf{R}^T \mathbf{A} \mathbf{R} \mathbf{y} = \begin{bmatrix} y_1 & y_2 \end{bmatrix} \mathbf{R}^{-1} \begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix} \mathbf{R} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{y}^T \mathbf{D} \mathbf{y} = 48$$

本题的数据是 $\theta = 45^\circ$ ，得到

$$\mathbf{R} = \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix} \quad \text{及} \quad \mathbf{R}^{-1} = \mathbf{P} = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix}$$

便有

$$\mathbf{R}^T \mathbf{A} \mathbf{R} = \begin{bmatrix} 3 & 0 \\ 0 & 7 \end{bmatrix} = \mathbf{D}$$

及

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \begin{bmatrix} y_1 & y_2 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 3y_1^2 + 7y_2^2 = 48$$

所以从几何图形上寻找二次型主轴的问题，在线性代数中就等价于使矩阵经过正交变换或相似变换 \mathbf{R} （注意这又是一个几何名词，说明被变换的图形的形状和尺寸保持不变），使矩阵 \mathbf{A} 对角化。图 9.8 中的 (c) 和 (d) 表示了对另一种双曲线二次型（它的两个特征值一正一负）的坐标变换，它需要转过的角度是 -30° 左右。靠目测不是科学的方法，要找到计算的规则，那就是把矩阵

$$\mathbf{A} = \begin{bmatrix} 1 & -4 \\ -4 & -5 \end{bmatrix}$$

对角化。要找其主轴的大小和方向，也就是找它的特征值 λ 和特征向量 \mathbf{e} 。输入

```
[e,lambda]=eig(A)
```

得到:

$$\mathbf{e} = \begin{bmatrix} 0.4472 & -0.8944 \\ 0.8944 & 0.4472 \end{bmatrix}, \quad \lambda = \begin{bmatrix} -7 & 0 \\ 0 & 3 \end{bmatrix}$$

对应于特征值-7的特征向量是 \mathbf{e} 中的第一列，它所对应的转角 θ 可按 $\cos \theta = 0.4472$ 或 $\sin \theta = 0.8944$ 求得为 63.4358° 。对应于特征值3的特征向量是 \mathbf{e} 中的第二列，它所对应的转角 $\theta = 153.4358^\circ$ ，即与前一个特征向量正交。这个由两个特征向量列排成的矩阵 \mathbf{e} 其实就是可以把 \mathbf{A} 对角化的正交变换矩阵 \mathbf{R} 。不难检验如下：

$$\mathbf{D} = \text{inv}(\mathbf{e}) * \mathbf{A} * \mathbf{e} = \begin{bmatrix} -7.0000 & 0.0000 \\ -0.0000 & 3.0000 \end{bmatrix}$$

所以按新的坐标系统列出的二次型方程为

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{y}^T \mathbf{R}^T \mathbf{A} \mathbf{R} \mathbf{y} = \begin{bmatrix} y_1 & y_2 \end{bmatrix} \begin{bmatrix} -7 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -7y_1^2 + 3y_2^2 = 48$$

如果我们希望把正的特征值放在前面，则只要把特征向量矩阵中的两列交换一下，使

$$\mathbf{R} = \begin{bmatrix} -0.8944 & 0.4472 \\ 0.4472 & 0.8944 \end{bmatrix}$$

就行了，此时新的二次型方程就成为 $\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{y}^T \mathbf{R}^T \mathbf{A} \mathbf{R} \mathbf{y} = 3y_1^2 - 7y_2^2 = 48$ 。图 9.8 (c) 和 (d) 画出的就是这种情况。

为了更好地理解图 9.8 的意义，画出了二次型：

$$z = f(y_1, y_2)$$

在几种系数下的曲面，如图 9.9 所示。(a) 图是两个特征值均为正的情况；(b) 图是两个特征值一正一负的情况；(c) 图是两个特征值均为负的情况；(d) 图是一个特征值为零的情况；曲面 (a) 与 $z=48$ 水平面的交线就是图 9.8 中的椭圆 (c)；曲面 (b) 与 $z=16$ 水平面的交线就是图 9.8 中的双曲线 (d)。由此我们知道，用正交矩阵把二次型转到对角化，实际上就是把坐标轴转动到二次型曲面的主轴方向。它可以把二次型的方程简化，在许多工程问题中，可以使它的某些物理意义变得清晰，因而得到广泛的应用。

MATLAB 中还提供了一个用以计算正交变换矩阵的函数 $\mathbf{R} = \text{orth}(\mathbf{A})$ 。当 \mathbf{A} 是对称实矩阵时，它的结果和 eig 函数算出的特征向量矩阵是一样的，在此不赘述了。

这是二维的情况，推广到高维空间，这个特性仍然存在，这类矩阵在实际应用中有很重要的价值。

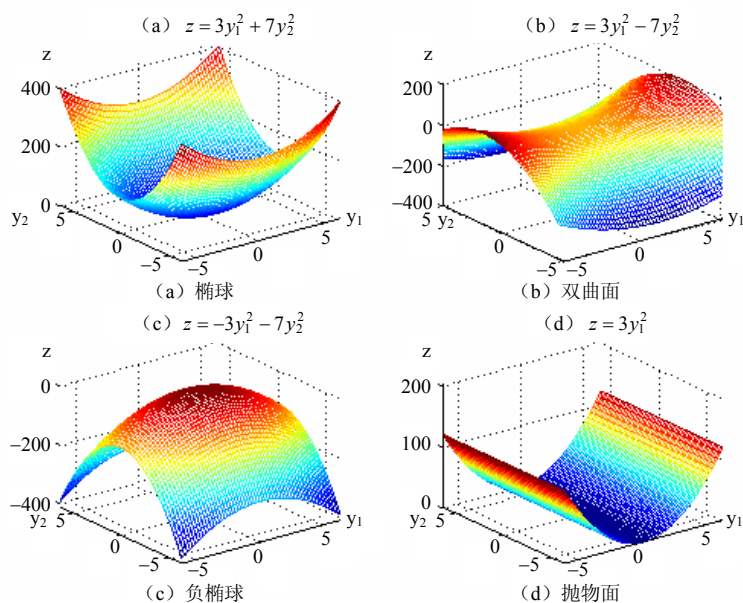


图 9.9 二次型曲面的几种类型

【例 9.9】 化二次型 $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 5x_3^2 + 2x_1x_2 + 6x_1x_3 + 2x_2x_3$ 为标准形。

解：用观察法不难得出其矩阵 A 具有如下形式：

$$A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 5 \end{bmatrix}$$

输入下列程序 ag909:

```
A=[1,1,3;1,2,1;3,1,5]
[R,D]=eig(A)
```

程序运行的结果为：

$$R = \begin{bmatrix} 0.8835 & -0.0253 & 0.4677 \\ -0.1667 & -0.9501 & 0.2636 \\ -0.4377 & 0.3108 & 0.8437 \end{bmatrix}$$

$$D = \begin{bmatrix} -0.6749 & 0 & 0 \\ 0 & 1.6994 & 0 \\ 0 & 0 & 6.9754 \end{bmatrix}$$

得知二次型最后的标准型为

$$f(x_1, x_2, x_3) = -0.6749y_1^2 + 1.6994y_2^2 + 6.9754y_3^2$$

其中

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = R^{-1} * x = R^T * x = \begin{bmatrix} 0.8835 & -0.1667 & -0.4377 \\ -0.0253 & -0.9501 & 0.3108 \\ 0.4677 & 0.2636 & 0.8437 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

【例 9.10】 设有对称实矩阵 $A = \begin{bmatrix} -2 & 4 & 1 \\ 4 & -2 & -1 \\ 1 & -1 & -3 \end{bmatrix}$, 试求

$$y = e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots + \frac{A^n}{n!} + \cdots。$$

解: 建模: 矩阵指数只有在 A 是对角矩阵时才可按对角元素直接计算, 即:

$$e^{\begin{pmatrix} a_1 & 0 \\ 0 & a_n \end{pmatrix}} = \begin{pmatrix} e^{a_1} & 0 \\ 0 & e^{a_n} \end{pmatrix}$$

注意它的非对角元素不符合指数运算规则, $e^0 = 1 \neq 0$ 。

因此首先要把 A 对角化。使 $A = pDp^{-1}$, 将此式代入 y 的表示式中, 可得:

$$y = e^A = pIp^{-1} + pDp^{-1} + \frac{pD^2p^{-1}}{2!} + \frac{pD^3p^{-1}}{3!} + \cdots + \frac{pD^n p^{-1}}{n!} + \cdots = pe^D p^{-1}$$

为此要求出其特征根矩阵 D 和特征向量矩阵 p 。由此可得 MATLAB 程序 ag910 如下:

```
A = [ -2, 4, 1; 4, -2, -1; 1, -1, -3 ],
[p,D] = eig(A)
```

运行此程序, 得

```
p = -0.6572    -0.2610    0.7071
      0.6572     0.2610    0.7071
      0.3690   -0.9294    0.0000
D = -6.5616     0         0
      0   -2.4384     0
      0     0     2.0000
```

由此得到:

$$e^{\begin{bmatrix} -2 & 4 & 1 \\ 4 & -2 & -1 \\ 1 & -1 & -3 \end{bmatrix}} = pe^{\begin{bmatrix} -6.5616 & 0 & 0 \\ 0 & -2.4384 & 0 \\ 0 & 0 & 2.0000 \end{bmatrix}}p^{-1} = p \begin{bmatrix} e^{-6.5616} & 0 & 0 \\ 0 & e^{-2.4384} & 0 \\ 0 & 0 & e^{2.0000} \end{bmatrix} p^{-1}$$

算式中的最后一步可以用下列语句来实现。

$$y=p*[\exp(D(1,1)),0,0;0,\exp(D(2,2)),0;0,0,\exp(D(3,3))]*\text{inv}(p)$$

得到

$$y=\begin{bmatrix} 3.7011 & 3.6880 & 0.0208 \\ 3.6880 & 3.7011 & -0.0208 \\ 0.0208 & -0.0208 & 0.0756 \end{bmatrix}$$

MATLAB 中有直接计算矩阵指数的函数 $\text{expm}(A)$ ，可以用它来检验所得结果。注意 MATLAB 中 expm 函数与 exp 函数的差别。

9.7 奇异值分解简介

奇异值分解(Singular Value Decomposition)对于矩阵的应用具有重要意义。比如矩阵的条件数(对计算精度)、矩阵广义逆(解超定方程)都与矩阵奇异值有关。美国多数教材都有, 而我国传统的教材大多不提, 大纲也不要求。本书折衷, 对这个内容做一扼要介绍。

定义 设矩阵 $A_{m \times n}$, 若存在非负实数 σ 和 n 维非零向量 u , m 维非零向量 v , 使得

$$Au = \sigma v, \quad Av = \sigma u \quad (9.7.1)$$

则称 σ 为 A 的**奇异值**, u 和 v 分别称为 A 对应于奇异值 σ 的**右奇异向量**和**左奇异向量**。

由式 (9.7.1) 可得

$$A^T Au = \sigma A^T v = \sigma^2 u \quad (9.7.2)$$

$$AA^T v = \sigma Au = \sigma^2 v \quad (9.7.3)$$

因此 σ^2 是 $A^T A (n \times n)$ 的特征值, 也是 $AA^T (m \times m)$ 的特征值, 说明 $A^T A$ 与 AA^T 的特征值均为非负实数; 而 u 和 v 分别是 $A^T A$ 和 AA^T 对应于特征值 σ^2 的特征向量。

$A^T A$ 与 AA^T 的非零特征值的个数(重特征值按重数计算)等于 A 的秩。设 $\sigma_1^2, \sigma_2^2, \dots, \sigma_k^2$ 是 $A^T A$ 的全部特征值, $k = \min\{n, m\}$, 称 $\sigma_i (i=1, 2, \dots, k)$ 为矩阵 A 的全部奇异值。

矩阵的奇异值分解定理: 设 A 是 $m \times n$ 矩阵, 设 $\sigma_1, \sigma_2, \dots, \sigma_n$ 是 A 的奇异值, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_n = 0$, 则 $A = USV^T$, 其中 U 是 m 阶正交矩阵, V 是 n

阶正交矩阵, $S = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}$, 而 $\Sigma_r = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$ 。

此式也可以表示为:

$$A = U^T S V = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_n u_n v_n^T \quad (9.7.4)$$

其中, u_i 是矩阵 U 的第 i 列 ($i=1, 2, \dots, m$), v_j 是矩阵 V 的第 j 列 ($j=1, 2, \dots, n$)。

证明不失一般性, 设 $m > n$, $R(A) = r$, v_1, v_2, \dots, v_n 是 $A^T A$ 对应于特征值 $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$ 的标准正交特征向量组, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_n = 0$ 。

令 $u_i = Av_i / \sigma_i$, 其中 $i=1, 2, \dots, r$, 故有

$$u_i^T u_j = \frac{1}{\sigma_i \sigma_j} v_i^T A^T A v_j = \frac{1}{\sigma_i \sigma_j} v_i^T \sigma_j^2 v_j = \frac{1}{\sigma_i \sigma_j} \sigma_j^2 v_i^T v_j = 0$$

$$u_i^T u_i = \frac{1}{\sigma_i^2} v_i^T A^T A v_i = \frac{1}{\sigma_i^2} v_i^T \sigma_i^2 v_i = v_i^T v_i = 1$$

因此, u_1, u_2, \dots, u_r 组成了一个标准正交向量组。如果 $r < m$, 可以把向量组 u_1, u_2, \dots, u_r 扩展为向量组 $u_1, u_2, \dots, u_r, \dots, u_m$, 从而构成 \mathbf{R}^m 空间的一个基。我们将证明 $AV=US$, 其中 S 是主对角元素为奇异值 $\sigma_1, \sigma_2, \dots, \sigma_r, 0, \dots, 0$ 的 $m \times n$ 阶‘对角’矩阵, 而 U 和 V 分别是 $V=[v_1, v_2, \dots, v_n]$ 和 $U=[u_1, u_2, \dots, u_r, \dots, u_m]$, 它们分别是 m 阶和 n 阶的正交矩阵。

$$AV = A[v_1, v_2, \dots, v_n] = [Av_1, Av_2, \dots, Av_n]$$

$$= [\sigma_1 u_1, \dots, \sigma_r u_r, 0, \dots, 0] = US$$

其中

$$S = \left[\begin{array}{c|c} \left. \begin{array}{c} \sigma_1 \\ \vdots \\ \sigma_r \end{array} \right\} r & \begin{array}{c} \mathbf{O} \\ \vdots \\ \mathbf{O} \end{array} \\ \hline \begin{array}{c} \mathbf{O} \\ \vdots \\ \mathbf{O} \end{array} \left\{ \begin{array}{c} r \\ n-r \end{array} \right\} m-r & \begin{array}{c} \mathbf{O} \\ \vdots \\ \mathbf{O} \end{array} \end{array} \right] = \begin{bmatrix} \Sigma_r & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \quad (9.7.5)$$

上式中的矩阵 S 由 m 行 n 列组成, 按假设 $m \geq n$, 所以左上角的 $r \times r$ 矩阵 S_r 决定了 S 的秩为 r 。也就是行列式 $\det(S_r) = \sigma_1 \sigma_2 \cdots \sigma_r \neq 0$, 而 $\det(S_{r+1}) = \sigma_1 \sigma_2 \cdots \sigma_{r+1} = 0$ 。可见 σ_{r+1} 要小到什么程度才能看成零的问题, 会直接影响秩 r 的判别。

因为 $V^T = V^{-1}$, 所以上式可以写成:

$$A = USV^T \quad (9.7.6)$$

命题得证。

如果 $m < n$, 则只需要把(9.7.5)式中 S 的表达形式做相应改动, 但不影响结果。

注解: 式(9.7.4)中的各项都是 $m \times n$ 矩阵。如果 A 由式中的前 r 项的和来近似, 这个和就称为 A 的为 r 阶近似 A_r 。从 S_r 的结构可以看出, A_r 的秩为 r 。

$$A_r = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T \quad (9.7.7)$$

奇异值分解的应用: 数字秩

我们曾把矩阵的秩定义为其最简行阶梯形的非全零行数。对于用浮点计算机算法进行计算的工程实际中的矩阵, 不大可能见到真正的全零行。在进行奇异值分解时, 也不大会出现 $\sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_n = 0$ 的情况, 通常这些 σ_i 都有微小的值。这时把 A 的秩看作多少, 就取决于把多小的 σ_i 值看成零。如果取 A 的秩为 r , 那就意味着 S_r 中只取了 r 个特征值 $\sigma_1, \sigma_2, \dots, \sigma_r$, 而认为 $r+1$ 及以后各个更小的奇异值都等于零, 经过这样的数值处理后得到的秩称为数字秩 (Numerical Rank)。

多小的数该看成零呢? 这取决于所用计算软件的精度。就 MATLAB 而言, 其基本部分采用的是 IEEE 浮点双精度格式标准, 它的精度(即 eps)是 2.24×10^{-16} , 当数 $\varepsilon \leq \text{eps}$ 时,

它与数 1 相加时会被略去, 即 $1+\text{eps}=1$ 。普遍地说, 当 $|b| \leq |a| \cdot \text{eps}$ 时, MATLAB 将无法分辨 $a+b$ 与 a 的差别。把这个概念推广到矩阵, 假设 A 是 $n \times n$ 方阵, 由(9.7.7)式可知:

$$A_{r+1} = A_r + \sigma_{r+1} u_{r+1} v_{r+1}^T = \sigma_1 u_1 v_1^T + \cdots + \sigma_r u_r v_r^T + \sigma_{r+1} u_{r+1} v_{r+1}^T \quad (9.7.8)$$

式中 $u_i v_i^T$ 中所有的 $n \times n$ 个元素都是小于 1 的。决定各项数值大小的主要因素是 σ_i 。如果 σ_{r+1} 比构成 A_r 的主要分量 $\sigma_1, \sigma_2, \dots, \sigma_r$ 的小得多, 以致加上它根本不起作用, 那么可以认为 σ_{r+1} 后的各项都为零。 A 的秩只能取作 r 。用式中的最大特征值 σ_1 与阶数 n 的乘积 $n\sigma_1$ 近似表征 A_r 的大小, 则可以把 $\text{tol} = n \cdot \sigma_1 \cdot \text{eps}$ 设为一个门限。大于这个门限特征值 $\sigma_i (i=1, 2, \dots, r)$ 的数目 r 就是 A 的数字秩, 把小于等于这个门限的特征值 $\sigma_i (i=r+1, \dots, n)$ 都看成零。

把最大与最小奇异值的比 σ_1 / σ_r 定义为矩阵 A_r 的条件数(Condition Number)。矩阵的条件数愈大, 意味着 σ_r 愈小, 它就愈接近奇异; 反之, 条件数愈小, 它就离开奇异门限愈远。

MATLAB 中设有奇异值分解函数, 其调用格式为 $[U, S, V] = \text{svd}(A)$, 其中 U 是 $m \times m$ 归一化正交矩阵, S 是 $m \times n$ 对角矩阵, 它的左上方是 $r \times r$, $r \leq \min(m, n)$ 对角矩阵, 其 r 个特征值已按递减规则, 其余各块元素都是全零。排列的 V 是 $n \times n$ 归一化正交矩阵。根据 S 中大于门限值的特征值数目, 就可以求出矩阵的数字秩, $\text{rank}(A)$ 就是按这个思路编写的。在大于门限值的特征值中, 最大和最小的两个元素之比, 就是矩阵的条件数, 可以调用 $r = \text{cond}(A)$ 算出。

【例 9.11】设 $A = \begin{bmatrix} 2 & 7 & 9 & -5 & 4 \\ -9 & -9 & 5 & 3 & -2 \\ -2 & 5 & -1 & -3 & 5 \\ -4 & 9 & 0 & 9 & -4 \end{bmatrix}$, 求它的各奇异值, 及条件数。

解: 这样阶次的问题, 只能用计算机来解了。程序为:

```
A=[2,7,9,-5,4;-9,-9,5,3,-2;-2,5,-1,-3,5;-4,9,0,9,-4]
[U,S,V]=svd(A), condA= S(1,1)/S(4,4)
```

运行结果为:

$$U = \begin{bmatrix} 0.6084 & -0.1647 & 0.7126 & -0.3081 \\ -0.7146 & -0.0499 & 0.6750 & 0.1767 \\ 0.3385 & -0.0071 & 0.1132 & 0.9341 \\ 0.0679 & 0.9851 & 0.1541 & -0.0358 \end{bmatrix}$$

$$S = \begin{bmatrix} 16.5933 & 0 & 0 & 0 & 0 \\ 0 & 13.9809 & 0 & 0 & 0 \\ 0 & 0 & 11.2638 & 0 & 0 \\ 0 & 0 & 0 & 5.9432 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.4037 & -0.2723 & -0.4876 & -0.6615 & 0.2957 \\ 0.7831 & 0.5813 & 0.0769 & 0.1011 & -0.1810 \\ 0.0943 & -0.1233 & 0.8590 & -0.4750 & 0.1115 \\ -0.3369 & 0.6838 & -0.0436 & -0.1774 & 0.6209 \\ 0.3184 & -0.3243 & 0.1287 & 0.5432 & 0.6941 \end{bmatrix}$$

不难验证： U, V 都是规范正交矩阵，都有 $U^T U = I$, $V^T V = I$,

A 的四个奇异值为： $[16.5933 \ 13.9809 \ 11.2638 \ 5.9432]$,

A 的条件数为： $\text{cond}(A) = 16.5933/5.9432 = 2.7920$ 。

9.8 应用实例

9.8.1 人口迁徙模型

假设在一个大城市中的总人口是固定的。人口的分布则因居民在市区和郊区之间迁徙而变化。每年有 6% 的市区居民搬到郊区去住，而有 2% 的郊区居民搬到市区。假如开始时有 30% 的居民住在市区，70% 的居民住在郊区，问十年后市区和郊区的居民人口比例是多少？30 年、50 年后又如何？

这个问题可以用矩阵乘法来描述。把人口变量用市区和郊区两个分量表示，即

$\mathbf{x}_k = \begin{bmatrix} x_{ck} \\ x_{sk} \end{bmatrix}$ ，其中下标 c 为市区，下标 s 为郊区， k 表示年份的次序。在 $k=0$ 的初始状态：

$$\mathbf{x}_0 = \begin{bmatrix} x_{c0} \\ x_{s0} \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}。$$

一年以后，市区人口为 $x_{c1} = (1-0.06)x_{c0} + 0.02x_{s0}$ ，郊区人口 $x_{s1} = 0.06x_{c0} + (1-0.02)x_{s0}$ ，用矩阵乘法来描述，可写成：

$$\mathbf{x}_1 = \begin{bmatrix} x_{c1} \\ x_{s1} \end{bmatrix} = \begin{bmatrix} 0.94 & 0.02 \\ 0.06 & 0.98 \end{bmatrix} \cdot \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} = A\mathbf{x}_0 = \begin{bmatrix} 0.2960 \\ 0.7040 \end{bmatrix}$$

从初始时间到 k 年，此关系保持不变，因此上述算式可扩展为 $\mathbf{x}_k = A\mathbf{x}_{k-1} = A^2\mathbf{x}_{k-2} = \dots = A^k\mathbf{x}_0$ ，用下列 MATLAB 程序 ag971 进行计算：

```
A=[0.94,0.02;0.06,0.98]
x0=[0.3;0.7]
x1=A*x0,
x10=A^10*x0
x30=A^30*x0
x50=A^50*x0
```

程序运行的结果为：

$$\mathbf{x}1 = \begin{bmatrix} 0.2960 \\ 0.7040 \end{bmatrix}, \mathbf{x}10 = \begin{bmatrix} 0.2717 \\ 0.7283 \end{bmatrix}, \mathbf{x}30 = \begin{bmatrix} 0.2541 \\ 0.7459 \end{bmatrix}, \mathbf{x}50 = \begin{bmatrix} 0.2508 \\ 0.7492 \end{bmatrix},$$

无限增加时间 k ，市区和郊区人口之比将趋向一组常数 0.25/0.75。为了弄清为什么这个过程趋向于一个稳态值，我们改变一下坐标系。在这个坐标系中可以更清楚地看到乘以矩阵 A 的效果，先求 A 的特征值和特征向量，输入 $[e, \text{lamda}] = \text{eig}(A)$ ，得到

$$e = \begin{bmatrix} -0.7071 & -0.3162 \\ 0.7071 & -0.9487 \end{bmatrix}, \text{lamda} = \begin{bmatrix} 0.9200 & 0 \\ 0 & 1.0000 \end{bmatrix}$$

令 $\mathbf{u}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, $\mathbf{u}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ ，它们分别与两个特征向量 $\mathbf{e}_1, \mathbf{e}_2$ 成比例并构成整数。可以看

到，用 A 乘以这两个向量的结果不过是改变向量的长度，不影响其相角（方向），改变的比例分别对应于其特征值 0.92 和 1。

$$A\mathbf{u}_1 = \begin{bmatrix} 0.94 & 0.02 \\ 0.06 & 0.98 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.92 \\ 0.92 \end{bmatrix} = 0.92\mathbf{u}_1$$

$$A\mathbf{u}_2 = \begin{bmatrix} 0.94 & 0.02 \\ 0.06 & 0.98 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \mathbf{u}_2$$

初始向量 \mathbf{x}_0 可以写成这两个基向量 \mathbf{u}_1 和 \mathbf{u}_2 的线性组合：

$$\mathbf{x}_0 = \begin{bmatrix} 0.30 \\ 0.70 \end{bmatrix} = 0.25 \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} - 0.05 \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} = 0.25\mathbf{u}_2 - 0.05\mathbf{u}_1$$

因此
$$\mathbf{x}_k = A^k \mathbf{x}_0 = 0.25\mathbf{u}_2 - 0.05(0.92)^k \mathbf{u}_1$$

式中的第二项会随着 k 的增大趋向于零。如果只取小数点后两位，则只要 $k > 27$ ，这第二项就可以忽略不计而得到

$$\mathbf{x}_k|_{k>27} = A^k \mathbf{x}_0 = 0.25\mathbf{u}_2 = \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix}$$

适当选择基向量可以使矩阵乘法结果等价于一个简单的实数乘子，避免相交项出现，使得问题简单化。这也是方阵求特征值的基本思想。

这个应用问题实际上是所谓马尔可夫过程的一个类型。所得到的向量序列 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ 称为马尔可夫链。马尔可夫过程的特点是 k 时刻的系统状态 \mathbf{x}_k 完全可由其前一个时刻的状态 \mathbf{x}_{k-1} 所决定，与 $k-1$ 时刻之前的系统状态无关。

9.8.2 产品成本的计算

某厂生产三种成品，每件产品的成本及每季度生产件数如表 9.1 及表 9.2 所示。试提供该厂每季度在每种产品上的成本表。

表 9.1 每件产品分类成本

成本（元）	产品 A	产品 B	产品 C
原材料	0.10	0.30	0.15
劳动	0.30	0.40	0.25
企业管理费	0.10	0.20	0.15

表 9.2 每季度产品分类件数

产品	夏	秋	冬	春
A	4000	4500	4500	4000
B	2000	2800	2400	2200
C	5800	6200	6000	6000

解：应当用矩阵来描述此问题，设产品分类成本矩阵为 M ，季度产量矩阵为 P ，便有：

$$M = \begin{bmatrix} 0.10 & 0.30 & 0.15 \\ 0.30 & 0.40 & 0.25 \\ 0.10 & 0.20 & 0.15 \end{bmatrix},$$

$$P = \begin{bmatrix} 4000 & 4500 & 4500 & 4000 \\ 2000 & 2800 & 2400 & 2200 \\ 5800 & 6200 & 6000 & 6000 \end{bmatrix}$$

将 M 和 P 相乘，得到的矩阵设为 Q ， Q 的第一行第一列元素为

$$Q(1,1)=0.1 \times 4000 + 0.3 \times 2000 + 0.15 \times 5800 = 1870$$

不难看出， Q 表示了夏季消耗的原材料总成本。从线性变换的角度来看， M 矩阵把以件数为单位的产品空间映射到了以元为单位的成本空间。

列出 MATLAB 程序 ag982：

```
M=[0.1,0.3,0.15;0.3,0.4,0.25;0.1,0.2,0.15];
P=[4000,4500,4500,4000;2000,2800,2400,2200;5800,6200,6000,6000];
Q=M*P
```

运行的结果显示：

$$Q = \begin{bmatrix} 1870 & 2220 & 2070 & 1960 \\ 3450 & 4020 & 3810 & 3580 \\ 1670 & 1940 & 1830 & 1740 \end{bmatrix}$$

根据各项元素的实际意义, 可将 Q 列成表格, 如表 9.3 所示

表 9.3 每季度产品总成本 Q

成本 (元)	夏	秋	冬	春	全 年
原材料	1870	2220	2070	1960	
劳动	3450	4020	3810	3580	
企业管理费	1670	1940	1830	1740	
总成本 (元)					

将此表格内分别按行或按列相加, 可以得出有综合意义的其他经济指标, 现留给读者自行思考和填写。

9.8.3 情报检索模型

因特网上数字图书馆的发展对情报的存储和检索提出了更高的要求, 现代情报检索技术就构筑在矩阵理论的基础上。通常, 数据库中收集了大量的文件 (书籍), 我们希望能从中搜索那些能与特定关键词相匹配的文件。文件的类型可以是杂志中的研究报告、因特网上的网页、图书馆中的书或胶片库中的电影, 等等。

假如数据库中包括了 n 个文件, 而搜索所用的关键词有 m 个。如果关键词按字母顺序排列, 我们就可以把数据库表示为 $m \times n$ 的矩阵 A 。每个文件用矩阵的列表示。 A 的第 j 列的第一个元素是一个数, 它表示第一个关键词出现的相对频率, 第二个元素表示第二个关键词出现的相对频率, 依次类推。用于搜索的关键词清单用 \mathbf{R}^m 空间的向量 \mathbf{x} 表示。如果关键词清单中第 i 个关键词在搜索行中出现, 则 \mathbf{x} 的第 i 个元素就赋值 1, 否则就赋值 0。为了进行搜索, 只要把 A^T 乘以 \mathbf{x} 。

举例来说, 假如我们的数据库包含有以下书名:

B1——应用线性代数

B2——初等线性代数

B3——初等线性代数及其应用

B4——线性代数及其应用

B5——线性代数及应用

B6——矩阵代数及应用

B7——矩阵理论

而搜索的 6 个关键词组成的集由以下的拼音字母次序排列:

初等, 代数, 矩阵, 理论, 线性, 应用

因为这些关键词在书名中最多出现 1 次, 所以其相对频率数不是 0 就是 1。当第 i 个

关键词出现在第 j 本书名上时, 元素 $A(i, j)$ 就等于 1, 否则就等于 0。这样我们的数据库矩阵就可用下表表示:

关键词	书						
	B1	B2	B3	B4	B5	B6	B7
初等	0	1	1	0	0	0	0
代数	1	1	1	1	1	1	0
矩阵	0	0	0	0	0	1	1
理论	0	0	0	0	0	0	1
线性	1	1	1	1	1	0	0
应用	1	0	1	1	1	1	0

假如读者输入的关键词是“应用, 线性, 代数”, 则数据库矩阵和搜索向量为:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

搜索结果可以表示为两者的乘积 $y = A^T x$, 于是

$$y = A^T x = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 3 \\ 3 \\ 2 \\ 0 \end{bmatrix}$$

y 的各个分量就表示各书与搜索向量匹配的程度。因为 $y_1=y_3=y_4=y_5=3$, 说明四本书 B1,B3,B4,B5 必然包含所有三个关键词。这四本书就被认为具有最高的匹配度, 因而在搜索的结果中把这几本书排在最前面。

本例把线性变换的概念进一步扩展。它不一定是在具体的几何空间内进行变量的变换(或映射), 在本例中, 它是从“关键词”子空间变换为“文献目录”的子空间。在信号处理中, 可以把时域信号变换为频域的频谱, 也属于线性变换。

现代的搜索中往往包括几百万个文件和成千的关键词, 所以数据库矩阵会变得非常大。但是也要注意数据库矩阵和搜索向量中大部分的元素为 0, 这种矩阵和向量被称为稀疏的。利用这个性质, 可以用为稀疏矩阵设计的存储和计算的方法, 节省计算机的存储空间和搜索时间, MATLAB 中就设有一个稀疏矩阵的子程序库。

9.9 习题

9.1 设 \mathbf{R}^2 空间 V 的基向量为 $[\mathbf{e}_1, \mathbf{e}_2]$, 分别为沿 x 和 y 方向的单位向量经过线性变换 $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 变换到像空间 W 。

- 设 $a=1, b=2, c=3, d=4$, 求出 $[\mathbf{e}_1, \mathbf{e}_2]$ 在像空间的坐标 $[\mathbf{q}_1, \mathbf{q}_2]$ 。用手工或 MATLAB 分别画出 $[\mathbf{e}_1, \mathbf{e}_2]$ 在原空间及在像空间的图形, 并用 a, b, c, d 在图上标注。
- 要使 $\mathbf{q}_1, \mathbf{q}_2$ 的长度保持为 1, 即与 $\mathbf{e}_1, \mathbf{e}_2$ 相同, 则 a, b, c, d 之间应符合什么数学关系?
- 要使 $\mathbf{q}_1, \mathbf{q}_2$ 的夹角保持为正交, 即与 $\mathbf{e}_1, \mathbf{e}_2$ 的夹角相同, 则 a, b, c, d 之间应符合什么数学关系?
- 用这两个条件检验图 9.1 中的六种变换, 报告所得的结论。

9.2 令 $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, 调用 ATLAST (“利用软件工具来增强线性代数教学”的缩写词) 中的函数 eigshow(\mathbf{A}),

观察它的图形与你画出的 $[\mathbf{q}_1, \mathbf{q}_2]$ 图形有何联系。

- $\mathbf{A}\mathbf{x}$ 椭圆的两个主轴的方向应该如何计算, 该调用哪个 MATLAB 函数?
- $\mathbf{A}\mathbf{x}$ 椭圆的两个主轴的长度应该如何计算, 该调用哪个 MATLAB 函数?
- 这两个主轴的方向是否正交? 为什么?

(d) 将 \mathbf{A} 改为 $\mathbf{A}_I = \begin{bmatrix} 1 & 3 \\ 3 & 4 \end{bmatrix}$, 做 eigshow(\mathbf{A}_I) 以及 (a), (b), (c) 同样的分析, 看与原来有什么不同, 并解释其理由。

9.3 设 $\mathbf{A} = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}, \mathbf{u} = \begin{bmatrix} 6 \\ -5 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

问 \mathbf{u} 和 \mathbf{v} 是不是 \mathbf{A} 的特征向量? 用什么方法判别?

9.4 按照制图国标, 斜体数字和英文字母的倾斜角度应约为 75° , 问

- \mathbf{A} 应如何选择, 才能保证其水平基线及高度不变, 而垂直基线倾斜成 75° ?
- 用数字 7 的空心字为例, 说明其正体字的数据集如何建立, 又如何用于生成斜体字。用图形加以说明。

9.5 在例 9.3 中, 改变平移和转动的次序, 让三角形先右移 2, 上移 3, 再转动 30° , 要求把平移后的图形用黄色填充, 转动后的图形用蓝色填充。

- 程序应该如何编写。
- 两次的变换矩阵具有何种形式?
- 三角形作为一个刚体, 其形状和大小不变, 只是方向和位置变了, 在变换矩阵中如何反映这一特点?

9.6 在运行 quatdemo 时, 设三个角度为 $u=-10^\circ$, $v=15^\circ$, $w=10^\circ$ 。问综合的变换矩阵 \mathbf{A} 为多少?

9.7 接上题, 按此新参数, 用例题中的飞行器外形数据集, 绘制飞行器图形, 与图 9.6 进行比较, 观察飞行器的状态有何变化。

9.8 设

$$\mathbf{p} = \begin{bmatrix} 1 & 2 & -1 \\ -3 & -5 & 0 \\ 4 & 6 & 1 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} -2 \\ 2 \\ 3 \end{bmatrix}, v_2 = \begin{bmatrix} -8 \\ 5 \\ 2 \end{bmatrix}, v_3 = \begin{bmatrix} -7 \\ 2 \\ 6 \end{bmatrix}$$

(a) 求出 \mathbf{R}^3 中的一组基 $\{u_1, u_2, u_3\}$, 使得 P 是从 $\{u_1, u_2, u_3\}$ 变换到 $\{v_1, v_2, v_3\}$ 的坐标变换矩阵。

(b) 求出 \mathbf{R}^3 中的一组基 $\{w_1, w_2, w_3\}$, 使得 P 是从 $\{v_1, v_2, v_3\}$ 变换到 $\{w_1, w_2, w_3\}$ 的坐标变换矩阵。

9.9 在 \mathbf{R}^4 中取两个基

$$u_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, u_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, u_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, u_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$v_1 = \begin{bmatrix} 2 \\ 1 \\ -1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 3 \\ 1 \\ 0 \end{bmatrix}, v_3 = \begin{bmatrix} 5 \\ 3 \\ 2 \\ 1 \end{bmatrix}, v_4 = \begin{bmatrix} 6 \\ 6 \\ 1 \\ 3 \end{bmatrix},$$

(a) 求从 $\{u_1, u_2, u_3, u_4\}$ 变换到 $\{v_1, v_2, v_3, v_4\}$ 的坐标变换矩阵。

(b) 求在 u 坐标下的向量 $[2, 5, -3, -1]$ 在基 $\{v_1, v_2, v_3, v_4\}$ 下的坐标。

(c) 求在两个基下有相同坐标的向量。

9.10 将下列矩阵对角化, 列出其正交矩阵 P 和对角矩阵 D 。

$$(a) \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}, (b) \begin{bmatrix} 7 & -4 & 4 \\ -4 & 5 & 0 \\ 4 & 0 & 9 \end{bmatrix},$$

9.11 将下列矩阵对角化, 列出其正交矩阵 P 和对角矩阵 D 。

$$(a) \begin{bmatrix} 4 & 1 & 3 & 1 \\ 1 & 4 & 1 & 3 \\ 3 & 1 & 4 & 1 \\ 1 & 3 & 1 & 4 \end{bmatrix}, (b) \begin{bmatrix} 5 & 2 & 9 & -6 \\ 2 & 5 & -6 & 9 \\ 9 & -6 & 5 & 2 \\ -6 & 9 & 2 & 5 \end{bmatrix},$$

9.12 一个热力厂用两种煤, 每烧一吨 A 煤可以产生 27.6 兆 Btu 的热, 3100 克的二氧化硫和 250 克的粉尘, 每烧一吨 B 煤可以产生 30.2 兆 Btu 的热, 6400 克的二氧化硫和 360 克的粉尘。问:

(a) 当烧 x_1 吨 A 和烧 x_2 吨 B 时, 产生的总热量应如何计算?

(b) 如何把此热力厂的产出用包含热量、二氧化硫和粉尘三元的输出向量表示?

(c) 经过了一段时间, 此热力厂共产出了 162 兆 Btu 的热, 23600 克的二氧化硫和 1623 克的粉尘。

试求出该热力厂烧了 A 煤和 B 煤各多少吨?

(d) 这样的变换是不是线性变换? 它是把哪个空间变换到哪个空间?

10.1 多项式插值问题

【例 10.1】 右表给出函数 $f(t)$ 上 4 个点的值，试求 $f(t)$ 的三次插值多项式 $p(t) = a_0 + a_1t + a_2t^2 + a_3t^3$ ，并求 $f(1.5)$ 的近似值。

t_i	0	1	2	3
$f(t_i)$	3	0	-1	6

解：令三次多项式函数 $p(t) = a_0 + a_1t + a_2t^2 + a_3t^3$ 通过表中已知的 4 点，可以得到四元线性方程组：

$$\begin{cases} a_0 &= 3 \\ a_0 + a_1 + a_2 + a_3 &= 0 \\ a_0 + 2a_1 + 4a_2 + 8a_3 &= -1 \\ a_0 + 3a_1 + 9a_2 + 27a_3 &= 6 \end{cases}$$

对于四元方程组，笔算很费事。应该用计算机求解了，输入：

```
A=[1,0,0,0;1,1,1,1;1,2,4,8;1,3,9,27],
b=[3;0;-1;6], a=A\b,
```

得到

$$a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ -2 \\ 1 \end{bmatrix}$$

三次多项函数为

$$p(t) = 3 - 2t - 2t^2 + t^3,$$

其曲线形状可用 MATLAB 语句

```
ezplot('3-2*t-2*t^2+t^3')
hold on, grid on
plot([0:3], [3,0,-1,6], 'x')
line([1.5,1.5],[0,6])
axis([-1,4,-2,8])
t1=1.5; p1=3-2*t1-2*t1^2+t1^3
plot(t1,p1,'o')
```

画出如图 10.1 所示的插值曲线。

故 $f(1.5)$ 近似等于

$$p(1.5) = 3 - 2(1.5) - 2(1.5)^2 + (1.5)^3 = -1.125。$$

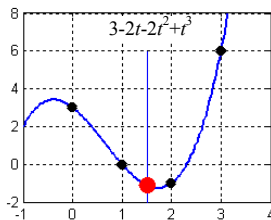


图 10.1 例 10.1 的插值曲线

在一般情况下, 当给出函数 $f(t)$ 在 $n+1$ 个点 $t_i (i=1, 2, \dots, n+1)$ 上的值 $f(t_i)$ 时, 就可以用 n 次多项式 $p(t) = a_0 + a_1t + a_2t^2 + \dots + a_nt^n$ 对 $f(t)$ 进行插值。如果给出的点数 (即方程数) 大于 $n+1$, 方程组成为超定的, 因而没有一个能满足方程组的解, 得出的曲线将不能通过给定的各点, 只是以最小二乘意义下的误差靠近各点, 于是插值就变为拟合。

插值也不一定是自变量的多项式, 比如圆锥曲线方程

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

虽然它有 6 个系数, 若用 a 除此方程两端, 得到的将是有 5 个待定系数的方程 $x^2 + c_1xy + c_2y^2 + c_3x + c_4y + c_5 = 0$ 。如果给出 x - y 平面上的 5 个点, 就可以列出 5 个线性方程来确定这 5 个系数。

10.2 坐标测量仪测定中的拟合问题

精密三坐标测量仪可以测量物体表面上任何一点的三维坐标, 人们要根据这些点的坐标推算出物体的特征尺寸。比如为了测量一个圆锥形截面的半径, 可在 x - y 平面内测量其圆周上 n 个点的坐标 $(x_i, y_i) (i=1, \dots, n)$, 然后拟合出此截面的方程。

对于每一组数据 (x_i, y_i) , 代入圆锥曲线方程, 移项可得:

$$x_i y_i c_1 + y_i^2 c_2 + x_i c_3 + y_i c_4 + c_5 = -x_i^2 \quad (i=1, 2, \dots, n)$$

n 个点就有 n 个方程。其结构相同, 只是数据不同。可以把数据写成列向量, 然后用元素群运算一次列出所有的 n 个方程。若 $n=5$, 是适定方程, $n>5$, 则是超定方程。为了提高测量的精度, 测量点数通常都比待求参数的数目多很多, 所以多采用超定方程计算出系数 c_1, \dots, c_5 。其程序可列写如下, 先把 x, y 设成 $n \times 1$ 列矩阵, 然后给出左端系数矩阵 $\mathbf{A} = [x.*y, y.*y, x, y, \text{ones}(n, 1)]$, 右端系数矩阵 $\mathbf{B} = -x.*x$, 最后用 $\mathbf{c} = \mathbf{A} \backslash \mathbf{B}$ 求出系数 c_1, \dots, c_5 。

【例 10.2】 举一个数字例, 设测量了圆周上 7 个点, 其 x, y 坐标如下:

$x = -3.000 \quad -2.000 \quad -1.000 \quad 0 \quad 1.000 \quad 2.000 \quad 3.000$
 $y = 3.03 \quad 3.90 \quad 4.35 \quad 4.50 \quad 4.40 \quad 4.02 \quad 3.26$

试求出此圆锥截面的方程，并求其最大、最小直径。

解：列出程序如下：

```

x=[-3:3]'; % 把 x,y 赋值为列向量
y=[3.03,3.90,4.35,4.50,4.40,4.02,3.26]';
A=[x.*y, y.*y, x, y, ones(size(x))] % 列出系数矩阵 A 和 B
B=-x.^2
c=inv(A'*A)*A'*B, % 解超定方程得出 c

```

方程的运行结果为：

$$A = \begin{bmatrix} -9.0900 & 9.1809 & -3.0000 & 3.0300 & 1.0000 \\ -7.8000 & 15.2100 & -2.0000 & 3.9000 & 1.0000 \\ -4.3500 & 18.9225 & -1.0000 & 4.3500 & 1.0000 \\ 0 & 20.2500 & 0 & 4.5000 & 1.0000 \\ 4.4000 & 19.3600 & 1.0000 & 4.4000 & 1.0000 \\ 8.0400 & 16.1604 & 2.0000 & 4.0200 & 1.0000 \\ 9.7800 & 10.6276 & 3.0000 & 3.2600 & 1.0000 \end{bmatrix} \quad B = \begin{bmatrix} -9 \\ -4 \\ -1 \\ 0 \\ -1 \\ -4 \\ -9 \end{bmatrix} \quad c = \begin{bmatrix} 0.0050 \\ 0.9214 \\ -0.2228 \\ -0.4050 \\ -16.8537 \end{bmatrix}$$

即截面的方程为：

$$x^2 + 0.0050xy + 0.9214y^2 - 0.2228x - 0.4050y - 16.8537 = 0$$

用语句

```

ezplot('x^2+0.005*x*y+0.9214*y^2-0.2228*x-0.4050*y-16.8537') %画圆锥曲线
hold on, plot(x,y,'x') % 画测量点
grid on, axis equal % x,y 两方向取同样比例尺

```

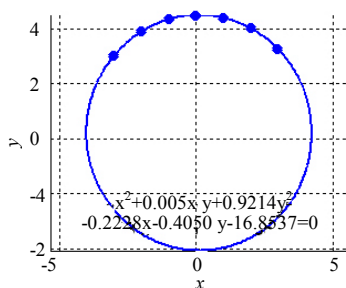


图 10.2 拟合出的圆锥曲面

可画出其拟合的图形和给定的测量点，如图 10.2 所示。从工程角度看，这样的测量点布局对拟合精度不大有利。

要求出此椭圆的长短轴长度，必须求出其主轴的方向和圆心的坐标。这个问题留给读者去做。即使用计算机，其步骤也比较麻烦。

如果我们知道被测工件截面本该是圆形，可以测定它的“拟合圆”的直径，则建模时就不应放进 5 个待定系数，而应该用圆截面的方程。

设圆周方程为：

$$(x - c_1)^2 + (y - c_2)^2 = r^2$$

其中 c_1, c_2 为圆心的坐标， r 为半径。对 c_1, c_2, r 整理上述方程，得到

$$2xc_1 + 2yc_2 + (r^2 - c_1^2 - c_2^2) = 2xc_1 + 2yc_2 + c_3 = x^2 + y^2$$

其中 $c_3 = r^2 - c_1^2 - c_2^2$ ，因而 $r = \sqrt{c_3 + c_1^2 + c_2^2}$ ，求出 c_1, c_2, c_3 就可求出 r 。

用 n 个测量点坐标 (x_i, y_i) 代入，得到

$$\begin{bmatrix} 2x_1 & 2y_1 & 1 \\ 2x_2 & 2y_2 & 1 \\ \dots & \dots & \dots \\ 2x_n & 2y_n & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \dots \\ x_n^2 + y_n^2 \end{bmatrix} \Rightarrow \mathbf{Ac} = \mathbf{B}$$

这是关于三个未知数 c_1, c_2, c_3 的 n 个线性方程，所以是一个超定问题。解出 c_1, c_2, c_3 就可得知这个最小二乘圆的圆心坐标和半径 r 的值。

下面用上题同样的 7 个测量点，求此工件的直径及圆心坐标。

7 点同样应该有 7 个方程，因为方程结构相同，可以用数据列的元素群运算一次完成系数矩阵的赋值。其程序为：

```
x=[-3:3]'; % 把 x,y 赋值为列向量
y=[3.03,3.90,4.35,4.50,4.40,4.02,3.26]';
A=[2*x, 2*y, ones(size(x))] % 按上述矩阵式求出系数矩阵 A,B
B=x.^2+y.^2
c=inv(A'*A)*A'*B, % 求超定方程的解，得出 c
r=sqrt(c(3)+c(1)^2+c(2)^2) % 由 c 求出 r
```

程序运行的最后结果为：

```
c =      0.1018      工件圆心的 x 坐标
      0.4996      工件圆心的 y 坐标
     15.7533
r =      4.0017      工件的半径 r
```

可以看出，采用适当的建模方法，加上把 MATLAB 的元素群运算和矩阵运算相结合，可以把复杂的计算变得相当简明。

10.3 天体轨道测量的曲线拟合问题

【例 10.3】 根据开普勒第一定律，当忽略其他天体的重力吸引时，一个天体应该取椭圆、抛物线或双曲线轨道。在适当的极坐标中，天体的位置 (r, θ) 应满足下列方程：

$$r = \beta + \varepsilon(r \cdot \cos \theta)$$

其中 β 为常数而 ε 是轨道的偏心率，对于椭圆 $0 \leq \varepsilon < 1$ ，对于抛物线 $\varepsilon = 1$ ，而对于双曲线 $\varepsilon > 1$ 。对一个人造卫星的观测得到了表中的数据，试确定其轨道的性质，并预测此天体在 $\theta = 0.46$ 弧度时的位置。

θ	0.88	1.10	1.42	1.77	2.14
r	3.00	2.30	1.65	1.25	1.01

解：对任何一组测量数据 θ_i, r_i ，可列出以下对 β 和 ε 两个变量的线性方程：

$$\beta + (r_i \cdot \cos \theta_i) \varepsilon = r_i$$

五个测量点共有五个方程，轨道只有两个未知数，这是一个超定问题。列出 MATLAB 程序，先用元素群运算一次列出 5 个方程的系数矩阵，再用超定方程解的公式。

```
theta=[0.88,1.1,1.42,1.77,2.14]'; % 测定的 theta 和 r 数组
r =[3, 2.3, 1.65, 1.25, 1.01]';
A=[ ones(5,1), r.*cos(theta)], B=r, % 解超定方程
X=inv(A'*A)*A'*B
rg=X(1)/(1-X(2)*cos(0.46)) %求 theta=0.46 处的 r
```

这意味着方程组具有如下矩阵形式： $AX = B$,

$$\text{其中 } A = \begin{bmatrix} 1.0000 & 1.9115 \\ 1.0000 & 1.0433 \\ 1.0000 & 0.2479 \\ 1.0000 & -0.2474 \\ 1.0000 & -0.5444 \end{bmatrix}, B = \begin{bmatrix} 3.0000 \\ 2.3000 \\ 1.6500 \\ 1.2500 \\ 1.0100 \end{bmatrix}, X = \begin{bmatrix} \beta \\ \varepsilon \end{bmatrix}$$

$$\text{得到: } X = \begin{bmatrix} \beta \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 1.4509 \\ 0.8111 \end{bmatrix}, \text{ rg}=5.3098$$

原方程可写成:

$$\begin{aligned} r &= \beta / (1 - \varepsilon \cdot \cos \theta) \\ &= 1.4509 / (1 - 0.8111 \cdot \cos \theta) \end{aligned}$$

用 MATLAB 语句画图:

```
% 画出极坐标中轨迹
ezpolar('1.4509/(1-0.8111*cos(theta))'),
hold on,polar(theta, r,'x'), %画出测定点
axis([-4,8,-5,5])
```

得到的图形如图 10.3 所示。

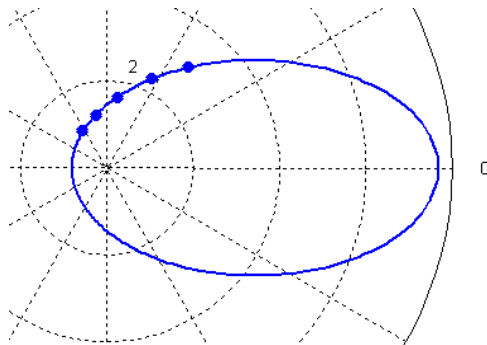


图 10.3 由 5 个测量点拟合的椭圆轨道

10.4 静力学问题

静力学研究物体受力后的平衡方程。一个物体在平面上平衡，需要三个平衡方程。如果是 N 个物体相互作用下的平衡，那么方程的总数就是 $3N$ 个。空间物体的平衡需要 6 个平衡方程。这个方程组通常都是线性的，所以求解就可以归结为矩阵方程。它可以避免解单个方程和单个变量，只要把系数矩阵输入程序，就轻而易举地同时得出所有的解。

【例 10.4】 两杆系统受力图如图 10.4 所示，设已知： $G_1=200$ ； $G_2=100$ ； $L_1=2$ ； $L_2=\sqrt{2}$ ； $\theta_1=30^\circ$ ； $\theta_2=45^\circ$ ；求所示杆系的支撑反力 N_a, N_b, N_c 。

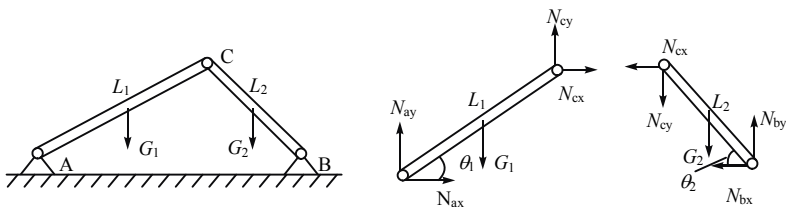


图 10.4 两杆系统的受力图（左）和分立体受力图（右）

解：◆画出杆 1 和杆 2 的分离体图，其中 N_a, N_b, N_c 都用其 x, y 方向的分量 $N_{ax}, N_{ay}, N_{bx}, N_{by}, N_{cx}, N_{cy}$ 表示，于是可列出方程如下：

对杆件 1，

$$\begin{aligned}\sum X=0 \quad N_{ax} + N_{cx} &= 0 \\ \sum Y=0 \quad N_{ay} + N_{cy} - G_1 &= 0; \\ \sum M=0 \quad N_{cy} \cdot L_1 \cos(\theta_1) - N_{cx} \cdot L_1 \sin(\theta_1) - G_1 \cdot L_1 / 2 \cos(\theta_1) &= 0;\end{aligned}$$

对杆件 2，

$$\begin{aligned}\sum X=0 \quad N_{bx} - N_{cx} &= 0; \\ \sum Y=0 \quad N_{by} - N_{cy} - G_2 &= 0; \\ \sum M=0 \quad N_{cy} \cdot L_2 \cos(\theta_2) + N_{cx} \cdot L_2 \sin(\theta_2) + G_2 \cdot L_2 / 2 \cos(\theta_2) &= 0;\end{aligned}$$

这是一组包含六个未知数 $N_{ax}, N_{ay}, N_{bx}, N_{by}, N_{cx}, N_{cy}$ 的六个线性代数方程，用手工解时很麻烦，又易出错，用了 MATLAB 工具，就可直接列出矩阵方程 $\mathbf{AX} = \mathbf{B}$ ，其中

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & L_1 \sin \theta_1 & L_1 \cos \theta_1 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & L_2 \sin \theta_2 & L_2 \cos \theta_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ G_1 \\ G_1 L_1 \cos \theta_1 / 2 \\ 0 \\ G_2 \\ -G_2 L_2 \cos \theta_2 / 2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} N_{ax} \\ N_{ay} \\ N_{bx} \\ N_{by} \\ N_{cx} \\ N_{cy} \end{bmatrix}$$

这个方程组不难用矩阵除法来解。在编写程序时，先输入已知条件，尽量用文字变量，在程序开始处给它们赋值，这样得出的程序具有一定的普遍性，要修改参数时只需修改头几行的数据。

◆ MATLAB 程序 ag1004

```
G1=200; G2=100; L1= 2; L2 = sqrt (2) ;    % 给原始参数赋值
theta1 = 30*pi/180; theta2 = 45*pi/180;    % 将度化为弧度
%按此次序, 系数矩阵 A,B 可写成下式
A=[1,0,0,0,1,0;0,1,0,0,0,1;0,0,0,-sin(theta1), cos(theta1);...
0,0,1,0,-1,0;0,0,0,1,0,-1;0,0,0, sin(theta2),cos(theta2)];
B=[0;G1;G1/2*cos(theta1);0;G2;-G2/2*cos(theta2)];
X = A\B                                % 用左除求解线性方程组
```

◆ 程序运行的结果为:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 0 & 0 & -0.5000 & 0.8660 \\ 0 & 0 & 1 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1.0000 \\ 0 & 0 & 0 & 0 & 0.7071 & 0.7071 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 200.0000 \\ 86.6025 \\ 0 \\ 100.0000 \\ -35.3553 \end{bmatrix}, X = \begin{bmatrix} 95.0962 \\ 154.9038 \\ -95.0962 \\ 145.0962 \\ -95.0962 \\ 45.0962 \end{bmatrix}$$

即:

N_{ax}	N_{ay}	N_{bx}	N_{by}	N_{cx}	N_{cy}
95.0962	154.9038	-95.0962	145.0962	-95.0962	45.0962

这样求解的方法不仅适用于全部静力学题目, 而且可用于材料力学和结构力学中的静力学和静不定问题。因为静不定问题只多了几个形变变量和变形协调方程, 通常也是线性的, 所以只不过是把矩阵方程扩大了几阶, 用计算机时, 解法和难度没有什么差别, 从下面的例子可以说明。

10.5 材料力学的静不定问题

由 n 根杆组成的桁架联结如图 10.5 所示, 受力 P 的作用, 各杆截面面积分别为 A_i , 材料弹性模量为 E , 求各杆的轴力 N_i 及节点 C 的位移。

解: 先列写具有普遍意义的方程, 设各杆均受拉力, A 点因各杆变形而引起的 x 方向位移 Δx , y 方向位移 Δy , 由几何关系, 得变形方程:

$$\Delta L_i = \frac{N_i L_i}{EA_i} = \Delta x \cos \alpha_i + \Delta y \sin \alpha_i \quad (i = 1, \dots, n)$$

$$\text{即: } \frac{N_i}{K_i} - \Delta x \cos \alpha_i - \Delta y \sin \alpha_i = 0$$

其中 $K_i = \frac{EA_i}{L_i}$ 为杆 i 的刚度系数。再加上两个力平衡

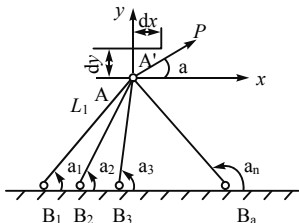


图 10.5 任意静不定杆系受力图 方程:

$$\sum_{i=1}^n N_i \cos \alpha_i = P \cos \alpha$$

$$\sum_{i=1}^n N_i \sin \alpha_i = P \sin \alpha$$

共有 $n+2$ 个方程, 其中包含 n 个未知力和两个待求位移 Δx 和 Δy , 方程组可解。因为这又是一个线性方程组, 可写成 $\mathbf{DX}=\mathbf{B}$, 容易由 MATLAB 的矩阵除法 $\mathbf{X}=\mathbf{D} \backslash \mathbf{B}$ 解出。

【例 10.5】 设三根杆组成的桁架如图 10.6 所示, 挂一重物 $P=3000$, 设 $L=2\text{m}$, 各杆的截面积分别为 $A_1=200\text{e-}6$, $A_2=300\text{e-}6$, $A_3=400\text{e-}6$, 材料的弹性模量 $E=200\text{e-}9$, 求各杆受力的大小。

此时应有两个平衡方程和三个位移协调方程:

$$\begin{aligned} -N_1 \cos(\alpha_1) - N_2 - N_3 \cos(\alpha_3) &= 0; \\ N_1 \sin(\alpha_1) - N_3 \sin(\alpha_3) &= P; \\ N_1/K_1 = d_1 = dx \cos(\alpha_1) + dy \sin(\alpha_1) \\ N_2/K_2 = d_2 = dx \\ N_3/K_3 = d_3 = dx \cos(\alpha_3) - dy \sin(\alpha_3) \end{aligned}$$

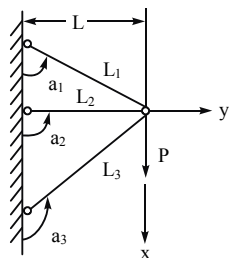


图 10.6 静不定三杆受力图

令 \mathbf{X} 为列向量 $[N_1; N_2; N_3; dx; dy]$, 把上述五个线性方程组列成 $\mathbf{DX}=\mathbf{B}$ 的矩阵形式, 从而就可用 MATLAB 的左除语句 $\mathbf{X}=\mathbf{A} \backslash \mathbf{B}$ 来求解。因此程序 ag1005 如下:

```
P=3000;E=200e9;L=2
A1=200e-6;A2=300e-6; A3=400e-6;
a1=pi/3; a2 = pi/2; a3=3*pi/4;
L1=L/sin(a1); L2=L/sin(a2); L3=L/sin(a3);    % 计算杆长
K1=E*A1/L1; K2=E*A2/L2; K3=E*A3/L3;          % 计算刚度系数
D(1,:) =[cos(a1),cos(a2),cos(a3),0,0];        % 分行给 D 赋值
D(2,:) =[sin(a1),sin(a2),sin(a3),0,0];
D(3,:) =[1/K1,0,0,-cos(a1),-sin(a1)];
D(4,:) =[0,1/K2,0,-cos(a2),-sin(a2)];
D(5,:) =[ 0,0,1/K3,-cos(a3),-sin(a3)];
B = [P;0;0;0;0];                                % 给 B 赋值
format long, X = D \ B                          % 求解线性方程组, 用长格式显示结果
```

执行此程序, 显示的结果为:

```
X = 1763.40607065591(N1)
     591.14251029634(N2)
    -2995.72429657297(N3)
      0.00016949097(dx)
      0.00001970475(dy)
```

若用普通格式显示, $dy=0.0000$, 实际上它不是零, 这可从 N_2 不等于零推想。在 MATLAB 中一个矩阵中包含数值相差很大的元素时, 就得采用 `format long` 来显示, 或者单独显示各个元素 $X(1), \dots, X(5)$ 。读者还可改变几根杆的刚度系数, 看它们如何影响各杆力的分布。

10.6 二自由度机械振动的模态分析

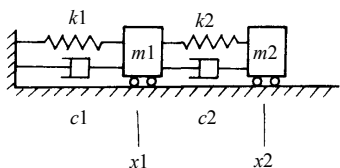


图 10.7 二自由度机械振动模型

图 10.7 表示了一个由两个质量和两个弹簧及阻尼器构成的二自由度振动系统, 今要求在给定两个质量的初始位置和初始速度的情况下求系统的运动。

解: 设 x_1 和 x_2 分别表示两个质量关于它们平衡位置的偏差值, 则此二自由度振动系统的一般方程为:

$$\begin{cases} m_1 \ddot{x}_1 + (c_1 + c_2) \dot{x}_1 - c_2 \dot{x}_2 + (k_1 + k_2) x_1 - k_2 x_2 = 0 \\ m_2 \ddot{x}_2 + c_2 (\dot{x}_2 - \dot{x}_1) + k_2 (x_2 - x_1) = 0 \end{cases} \quad (1)$$

可写成矩阵形式:

$$M\ddot{\mathbf{X}} + C\dot{\mathbf{X}} + K\mathbf{X} = \mathbf{0} \quad (2)$$

其中

$$M = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}, \quad C = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}, \quad K = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (3)$$

这是一个四阶常微分方程组。给出它的初始条件 (初始位置 X_0 和初始速度 X_{d0}):

$$\mathbf{X}_0 = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix}, \quad \mathbf{X}_{d0} = \dot{\mathbf{X}}_0 = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_{d10} \\ x_{d20} \end{bmatrix}, \quad (4)$$

可以求出它的解, 但用解析方法求解非常麻烦。如果 $C=0$, 即无阻尼情况, 则系统可解耦为两种独立的振动模态, 通常的书上只给出解耦简化后的解。

有了 MATLAB 工具, 根本无须设 $C=0$, 也无须解耦, 就可以用很简洁的程序求出其数值解。其基本思路是把原始方程化成典型的四个一阶方程构成的状态方程组:

$$\dot{\mathbf{Y}} = \mathbf{A}\mathbf{Y} \quad (5)$$

这个方程在初始条件 $\mathbf{Y}=\mathbf{Y}_0$ 下的解为 $\mathbf{Y} = \mathbf{Y}_0 e^{\mathbf{A}t}$ 。用 MATLAB 表示为 $\mathbf{Y} = \mathbf{Y}_0 * \text{expm}(\mathbf{A} * t)$ 。其中 expm 表示把 $(\mathbf{A} * t)$ 看成矩阵来求其指数。如果我们只要最后的波形, 可以直接选用 MATLAB 所提供的 expm , expm1 , ……中任何一个函数。所以我们只要把 \mathbf{Y}, \mathbf{Y}_0 和 \mathbf{A} 找到就行。

先把方程 (2) 写成两个一阶矩阵方程:

$$\left. \begin{aligned} \dot{\mathbf{X}} &= \mathbf{X}_d \\ \dot{\mathbf{X}}_d &= \ddot{\mathbf{X}} = -M^{-1}C\dot{\mathbf{X}} - M^{-1}K\mathbf{X} \end{aligned} \right\} \Rightarrow \begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{X}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \mathbf{X}_d \end{bmatrix} \quad (6)$$

$$\text{于是 } \mathbf{Y} = \begin{bmatrix} \mathbf{X} \\ \mathbf{X}_d \end{bmatrix}, \quad \mathbf{Y}_0 = \begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_{d0} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \quad (7)$$

对于本题的二自由度系统, $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 和 $\mathbf{X}_d = \begin{bmatrix} x_{d1} \\ x_{d2} \end{bmatrix}$, 所以 \mathbf{Y} 和 \mathbf{Y}_0 都是 4×1 的单列变量;

由于 \mathbf{A} 中的四个元素都是 2×2 方阵, \mathbf{A} 是 4×4 方阵。对于更多自由度的系统, 公式 (7) 都是正确的。只要改变矩阵 $\mathbf{O}, \mathbf{I}, \mathbf{M}, \mathbf{C}, \mathbf{K}, \mathbf{Y}, \mathbf{Y}_0$ 的阶数即可。

【例 10.6】 下面给出二自由度系统的一个数值例, 设 $m_1=1; m_2=9; k_1=4; k_2=2; c_1$ 和 c_2 可由用户输入。求在初始条件 $x_0 = [1; 0]$ 和 $x_{d0} = [0; -1]$ 下, 系统的输出 x_1, x_2 曲线。

根据上面的模型可以写出程序 ex1006 如下。

```

m1=1; m2=9; k1 = 4; k2=2; % 输入各原始参数
c1=input('c1='); c2=input('c2='); % 输入阻尼系数
x0 = [1; 0]; xd0 = [0; -1]; tf= 50; dt=0.1; % 给出初始条件及时间向量
M = [m1, 0; 0, m2]; K = [k1+k2, -k2; -k2, k2]; % 构成二阶参数矩阵
C = [c1+c2, -c2; -c2, c2];
A=[zeros(2,2), eye(2); -M\K, -M\C]; % 构成四阶参数矩阵
y0=[x0; xd0]; % 四元变量的初始条件
for i=1:round(tf/dt)+1 % 设定计算点, 作循环计算
    t(i)=dt*(i-1);
    y(:,i)=expm(A*t(i))*y0; % 循环计算矩阵指数
end
subplot(2,1,1), plot(t, y(1,:)), grid % 按两个分图绘制 x1, x2 曲线
subplot(2,1,2), plot(t, y(2,:)), grid

```

运行此程序, 输入 $c_1=0.2, c_2=0.5$ 所得的结果如图 10.8 所示。从中可清楚地看到振动的两种模态。特别是 x_1 的运动反映了两种模态的叠合。给出不同的初始条件, 各模态的幅度也会变化。输入 $c_1=0, c_2=0$ 所得的结果如图 10.9 所示, 这时两种振动模态分别出现在两个波形中, 即已经简单地解耦了。

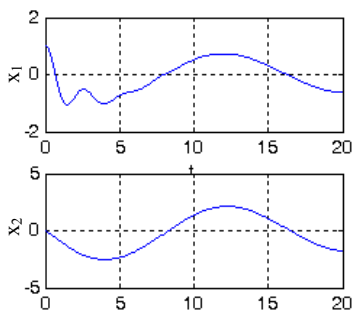


图 10.8 二自由度振动的输出
波形 $c_1=0.2, c_2=0.8$

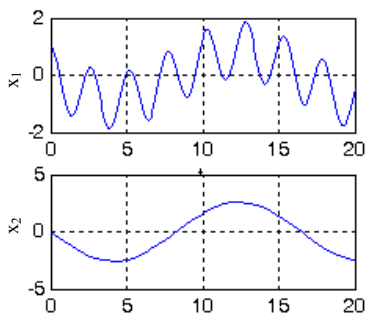


图 10.9 二自由度振动的输出
波形 ($c_1=c_2=0$)

在其他情况下, 解耦就等价于把矩阵 \mathbf{A} 对角化。在输入 $c_1=0.2, c_2=0.5$ 的条件下执行 ag1006 后, 输入

```
[p, lamda]=eig(A)
```

$$\text{得到 } p = \begin{bmatrix} -0.0556 - 0.3717i & -0.0556 + 0.3717i & 0.3021 + 0.0156i & 0.3021 - 0.0156i \\ -0.0024 + 0.0155i & -0.0024 - 0.0155i & 0.8836 & 0.8836 \\ 0.9258 & 0.9258 & -0.0099 + 0.1153i & -0.0099 - 0.1153i \\ -0.0368 - 0.0116i & -0.0368 + 0.0116i & -0.0116 + 0.3380i & -0.0116 - 0.3380i \end{bmatrix}$$

$$\text{lamda} = \begin{bmatrix} -0.3646 + 2.4361i & 0 & 0 & 0 \\ 0 & -0.3646 & -2.4361i & 0 \\ 0 & 0 & -0.0132 + 0.3825i & 0 \\ 0 & 0 & 0 & -0.0132 - 0.3825i \end{bmatrix}$$

特征根 **lamda** 由两组共轭复根组成，它的虚部是振动的角频率，它的实部是它的衰减系数。所以矩阵特征根反映了两种振动模态的特征，使我们更进一步理解了“特征”两字的物理意义。回想在例 9.10 中曾介绍过矩阵指数的求法，它的一个中间步骤也正是使 **A** 对角化，也表明了数学和物理的统一。

从这个例子也可以看出，用计算机解题时，问题和数据的复杂性对解题的过程几乎没有影响。通常我们选择比较简单，且有解析解的数据作为检验程序之用。一旦确定程序正确，它就可用在很复杂的情况。我们曾将这个程序扩展，用于解 5 自由度的振动系统，也得到了很好的结果。

10.7 交流稳态电路的复数矩阵解

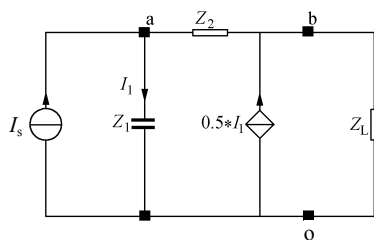


图 10.10 例 10.7 的原电路

交流稳态电路方程是线性的，但其系数是复数，变量是平面上的向量（也是复数），所以得到的是复数线性方程组。虽然线性代数理论只证明到实数，但 MATLAB 中有关线性代数的函数对实数和复数同样有效，所以可以用在复数条件下。

【例 10.7】 如图 10.10 所示电路，设 $Z_1 = -j250 \Omega$ ， $Z_2 = 250 \Omega$ ， $I_s = 2 \angle 0^\circ \text{ A}$ ，负载 $Z_L = 500 + j 500 \Omega$ ，求负载电压。

解： ■ 建模

以节点电压 \dot{U}_a \dot{U}_b 和电流 \dot{I}_1 为未知量（都是复数），用节点电压法，按进出 a, b 点的电流相等，列出方程组如下，其中系数（阻抗值）也都是复数：

$$\left(\frac{1}{Z_1} + \frac{1}{Z_2} \right) \dot{U}_a - \frac{1}{Z_2} \dot{U}_b = \dot{I}_s$$

$$-\frac{1}{Z_2} \dot{U}_a + \frac{1}{Z_2} \dot{U}_b = \frac{\dot{U}_b}{Z_L} + 0.5 \dot{I}_1 \quad \frac{1}{Z_1} \dot{U}_a = \dot{I}_1$$

整理以上方程, 将未知量 \dot{U}_a , \dot{U}_b , \dot{I}_1 均移到等号左端, 得

$$\begin{bmatrix} \frac{1}{Z_1} + \frac{1}{Z_2} & -\frac{1}{Z_2} & 0 \\ -\frac{1}{Z_2} & \frac{1}{Z_2} - \frac{1}{Z_L} & -0.5 \\ \frac{1}{Z_1} & 0 & -1 \end{bmatrix} \begin{bmatrix} \dot{U}_a \\ \dot{U}_b \\ \dot{I}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [\dot{I}_s] \Rightarrow \mathbf{AX} = \mathbf{BI}_s$$

令 $\dot{I}_s = 2\angle 0^\circ = 2 + j0$, 用矩阵运算求得 \dot{U}_a , \dot{U}_b , \dot{I}_1 , (程序中变量都已是复数)

MATLAB 程序 ag1007.m

```
clear, format compact
Z1=-j*250;Z2=250;ki=0.5;Is=2+j*0;
zL=500+j*500; % 设定元件参数
a11=1/Z1+1/Z2;a12=-1/Z2;a13=0; % 设定系数矩阵 A
a21=-1/Z2;a22=1/Z2-1/zL;a23=-ki;
a31=1/Z1;a32=0;a33=-1;
A=[a11,a12,a13;a21,a22,a23;a31,a32,a33];
B=[1;0;0]; % 设定系数矩阵 B
X=A\B*Is; % 求方程解 X=[Ua;Ub;I1]
Ub=X(2) % 负载电压
absUb=abs(Ub) % 负载电压的幅度
angleUb=angle(Ub)*180/pi % 负载电压的相角(度)
compass([Ub]); % 画出复数解的向量图
```

程序运行结果为:

负载电压 $\dot{U}_b = -2.5000\text{e}+002 -7.5000\text{e}+002\text{i}$

其幅度 $\text{absUb} = 790.5694$, 相角 $\text{angleUb} = -108.4349$

\dot{U}_b 的向量如图 10.11 所示。

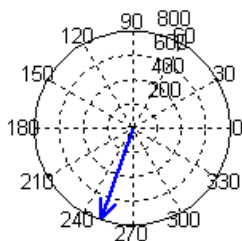


图 10.11 \dot{U}_b 的向量图

10.8 线性系统零输入响应的计算

任意线性时不变连续系统的特性可用常微分方程表示为:

$$a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + y = b_m \frac{d^m u}{dt^m} + \cdots + b_1 \frac{du}{dt} + b_0 u \quad (n \geq m) \text{ 求其零输入响应。}$$

应。

在零输入条件 $u=0$ 时, 等式右端为零。系统响应的通解为:

$$y(t) = C_1 e^{p_1 t} + C_2 e^{p_2 t} + \cdots + C_n e^{p_n t}$$

其中, $\mathbf{p}=[p_1, p_2, \dots, p_n]$ 是特征方程 $a_n \lambda^n + a_{n-1} \lambda^{n-1} + \cdots + a_1 \lambda + 1 = 0$ 的 n 个根组成的向量,

其每个分量的系数 C_i 则应由 y 及其各阶导数的初始条件 $y_0, D y_0, \dots, D(n-1) y_0$ 来确定。

$$\begin{aligned}
 y(0) &= C_1 + C_2 + \cdots + C_n = y_0 \\
 y'(0) &= p_1 C_1 + p_2 C_2 + \cdots + p_n C_n = Dy_0 \\
 y''(0) &= p_1^2 C_1 + p_2^2 C_2 + \cdots + p_n^2 C_n = D^2 y_0 \\
 &\vdots \\
 y^{(n-1)}(0) &= p_1^{(n-1)} C_1 + p_2^{(n-1)} C_2 + \cdots + p_n^{(n-1)} C_n = D^{(n-1)} y_0
 \end{aligned}$$

初始条件数应该和常数数相等, 由此构成一个确定 C_1, \dots, C_n 的线性代数方程组, 写成:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ p_1 & p_2 & \cdots & p_n \\ \vdots & \vdots & \ddots & \vdots \\ p_1^{n-1} & p_2^{n-1} & \cdots & p_n^{n-1} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{bmatrix} = \begin{bmatrix} y_0 \\ Dy_0 \\ \vdots \\ D^{(n-1)} y_0 \end{bmatrix} \Rightarrow \mathbf{V} \cdot \mathbf{C} = \mathbf{Y}_0'$$

从而可用矩阵左除计算出常数 $\mathbf{C} = \mathbf{V} \backslash \mathbf{Y}_0'$ 。

矩阵 \mathbf{V} 由特征根向量 \mathbf{p} 唯一地确定, 这种形式称为范德蒙特矩阵, 在 MATLAB 中, 有生成它的函数 `vander(p)`。试用一下后, 可以知道它产生的矩阵与本题的矩阵排列转了 90° , 应该用 $\mathbf{V} = \text{rot90}(\text{vander}(\mathbf{p}))$, 于是即可按此思路编成 MATLAB 程序 `ag1008`:

```

a=input('输入分母系数向量 a=[a1,a2,...]= ');
n=length(a)-1;
Y0=input('输入初始条件向量 Y0=[y0,Dy0,D2y0,...]= ');
p=roots(a);V=rot90(vander(p));c= V\Y0';
% 以下是计算并画出时间响应
dt=input('dt='); tf=input('tf= ');           % 输入时间数组的参数
t=0:dt:tf; y=zeros(1,length(t));              % 生成 t 向量和 y 初始向量
for k=1:n y= y+c(k)*exp(p(k)*t);end            % 将各分量叠加
plot(t,y),grid                                  % 绘图

```

【例 10.8】 用这一个普遍程序来解一个三阶系统, 设其微分方程为:

$$\frac{d^n y}{dt^n} + 2 \frac{d^{n-1} y}{dt^{n-1}} + 9 \frac{dy}{dt} + 3y = 0$$

初始条件为:

$$y(0)=1, y'(0)=0, y''(0)=0$$

解: 运行 `ag1008`, 按提示输入 \mathbf{a} 为 `[1,2,9,3]`; \mathbf{Y}_0 为 `[1,0,0]`; `dt` 为 `0.1`; `tf` 为 `5`; 即可得到 $t=0 \sim 5$ 秒的零输入响应曲线。

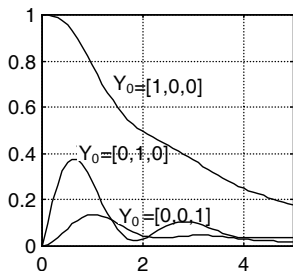


图 10.12 三阶系统零状态分量的解

再分别取 \mathbf{Y}_0 为 `[0,1,0]`; `[0,0,1]`, 用 `hold on` 语句使三次运行生成的图形叠画在一幅图上, 得到图 10.12 中的三根曲线。

10.9 计算频谱用的 DFT 矩阵

有限长序列 $x(n)$ ($0 \leq n \leq N-1$) 有 N 个样本值。它

的傅里叶变换 $X(j\omega)$ 在频率区间 $(0 \leq \omega < 2\pi)$ 的 N 个等间隔分布的点 $\omega_k = 2\pi k/N (0 \leq k \leq N-1)$ 上也有 N 个样本值。这两组有限长的序列之间可以用简单的关系联系起来:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad 0 \leq k \leq N-1$$

其中 $W_N = e^{-j\frac{2\pi}{N}}$, 是一个相角为 $-\frac{2\pi}{N}$ 的单位向量, 也称为旋转因子。 $X(k)$ 就称为 $x(n)$

的离散傅里叶变换(DFT), 也就是 $x(n)$ 的频谱。用矩阵来表示, 可写成:

$$\mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = \mathbf{W} \cdot \mathbf{x}$$

所以, 信号频谱的计算可以简单地用一个矩阵乘法来完成。信号是 $N \times 1$ 维数组, 矩阵 \mathbf{W} 称为 DFT 矩阵, 它是 $N \times N$ 维的复数阵。把矩阵乘法看作一变换, 我们就可以把频谱计算看作信号从时域到频域的线性变换。

这个矩阵运算可用下列几条 MATLAB 语句来实现。

```
n = [0:1:N-1]; k = [0:1:N-1]; % 设定 n 和 k 的行向量
WN = exp(-j*2*pi/N); % 设定 Wn 因子
nk = n'*k; % 产生一个含 nk 值的 N 乘 N 维的整数矩阵
WNnk = WN.^nk; % 求出 w 矩阵
Xk = xn * WNnk; % 求出离散傅里叶级数系数
```

前两条语句无须解释。第三条语句把 \mathbf{n} 转列为列向量 \mathbf{n}' , 再与行向量 \mathbf{k} 做矩阵乘, 它产生的是一个整数矩阵:

$$\mathbf{n}' * \mathbf{k} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ N-1 \end{bmatrix} [0, 1, 2, \cdots, N-1] = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & N-1 \\ 0 & 2 & \cdots & 2 \times (N-1) \\ \vdots & \vdots & & \vdots \\ 0 & N-1 & \cdots & (N-1) \times (N-1) \end{bmatrix}$$

这个整数方阵恰好是算式 $\mathbf{X} = \mathbf{W} \cdot \mathbf{x}$ 中 \mathbf{W} 矩阵的指数部分。第四条语句就产生了 \mathbf{W} 矩阵, 而最后一条语句就完成了 DFT 的全部运算。所以如果写成 MATLAB 子程序, 它可以命名为 DFT。

在这 5 行 DFT 子程序前面, 加上输入语句:

```
xn=input('xn= ');
N=length(xn);
```


在子程序后面，加上绘图语句：

```
subplot(2,1,1),stem(xn,'.');
subplot(2,1,2),stem(abs(Xk),'.');
```

就得到本题的程序 ag1009。

运行 ag1009，并按提示输入 x_n ，即可在 10.13 图的图 (a) 上得到 x_n 序列的波形，并在 10.13 图的图 (b) 上得到此信号的幅频特性。

【例 10.9】 例如输入序列为 $x_n=[\text{ones}(1,64),\text{zeros}(1,192)]$ ，则得到的时域波形及其频谱图如图 10.13 所示。

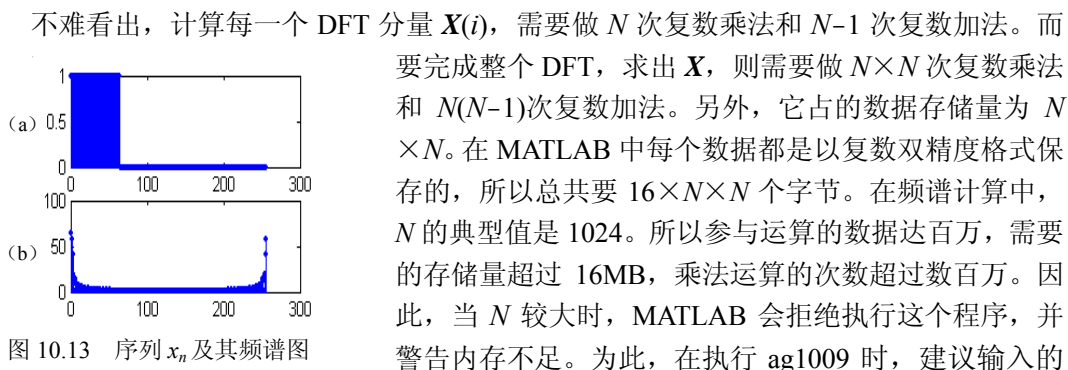


图 10.13 序列 x_n 及其频谱图

不难看出，计算每一个 DFT 分量 $X(i)$ ，需要做 N 次复数乘法和 $N-1$ 次复数加法。而要完成整个 DFT，求出 X ，则需要做 $N \times N$ 次复数乘法和 $N(N-1)$ 次复数加法。另外，它占的数据存储量为 $N \times N$ 。在 MATLAB 中每个数据都是以复数双精度格式保存的，所以总共要 $16 \times N \times N$ 个字节。在频谱计算中， N 的典型值是 1024。所以参与运算的数据达百万，需要的存储量超过 16MB，乘法运算的次数超过数百万。因此，当 N 较大时，MATLAB 会拒绝执行这个程序，并警告内存不足。为此，在执行 ag1009 时，建议输入的 x_n 长度不要超过 256。此外，人们研究了很多种加快计算并节省内存的方法，统称为快速傅里叶变换 (FFT)。

从这里也可以看出，矩阵给我们提供了一个组织海量数据进行运算的最好方法，包括对上百万数据进行赋值、运算和绘图，只用了几行程序。比如其中给 100 多万个元素的 DFT 矩阵赋值，用的是 1024 元单列矩阵乘以单行矩阵的方法，这真是矩阵运算的奇妙点之一。线性代数的重要性之所以在近二十年来可与微积分相媲美，这是一个重要原因。如果学线性代数而不涉及工程计算实践，那就无法真正体会线性代数的精髓所在。

10.10 最优化有限冲激响应 (FIR) 数字滤波器设计

有关最大化有限冲激响应 (FIR) 数字滤波器设计的详细情况可参见参考文献[12]，本节仅摘录部分进行讲解。

设给定滤波器的预期幅频特性 $D(\omega)$ 在 $\omega = \omega_i (i=1, 2, \dots, K)$ 的 K 个频点上的值，若实际滤波器的脉冲响应的长度为 $N=2L+1$ ，则其幅特性与预期特性相拟合的方程为：

$$A(\omega_i) = \sum_{n=0}^L d(n) \cos n\omega_i = D(\omega_i) = D_i \quad (i=1, 2, \dots, K)$$

这 K 个方程联立。由于 $A(\omega_i)$ 是 $L+1$ 个谐波的线性组合，这方程组中的待定参数 $d(n)$ 有 $L+1$ 个。

如果 $K=L+1$ ，即方程数与未知数的数目相等，则这个线性方程组是适定的，恰好可以解出这些系数。

如果 $K > L+1$, 即方程数比未知数的数目多, 形成了所谓超定方程组, 那就不可能找到精确满足这些方程组的系数 $d(n)$, 只能找到最近似地满足这些方程的最小二乘解。

如果 $K < L+1$, 则形成了不定方程组, 那会有无穷个解, 工程上是没有意义的。所以只考虑 $K \geq L+1$ 的情况。

$$e = D - Pd$$

其中: e 为单列 K 元误差向量, D 为预期频率特性在样本点列上的 K 元单列向量, d 为 $L+1$ 元待定系数单列向量, 而 P 则是由 $\cos(n\omega_i)$ 组成的 $K \times (L+1)$ 的系数矩阵。

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_K \end{bmatrix}, D = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_K \end{bmatrix}, P = \begin{bmatrix} 1 & \cos(\omega_1) & \cdots & \cos(L\omega_1) \\ 1 & \cos(\omega_2) & \cdots & \cos(L\omega_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cos(\omega_K) & \cdots & \cos(L\omega_K) \end{bmatrix}, d = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_L \end{bmatrix}$$

前面已经得到它的最小二乘解的公式:

$$d = (P^T P)^{-1} P^T D$$

用 MATLAB 计算此式特别方便, 可以表示为 $d = \text{inv}(P' * P) * P' * D$, 也可以更简便地用左除运算 $d = P \backslash D$ 来完成。

【例 10.10】 举一个数字例, 要求设计长度为 $N=9$ (有 5 个待定系数) 的 FIR 滤波器, 要使它在 $0 \sim \pi$ 之间的八个频点 ω 上逼近预期的低通幅特性 D :

$$\omega = [0, 0.33, 0.67, 1, 1.5, 2, 2.5, 3.14]; D = [1, 1, 1, 1, 0, 0, 0, 0];$$

解: 列出方程组的完整形式:

$$\begin{bmatrix} 1 & \cos \omega_1 & \cos 2\omega_1 & \cos 3\omega_1 & \cos 4\omega_1 \\ 1 & \cos \omega_2 & \cos 2\omega_2 & \cos 3\omega_2 & \cos 4\omega_2 \\ 1 & \cos \omega_3 & \cos 2\omega_3 & \cos 3\omega_3 & \cos 4\omega_3 \\ 1 & \cos \omega_4 & \cos 2\omega_4 & \cos 3\omega_4 & \cos 4\omega_4 \\ 1 & \cos \omega_5 & \cos 2\omega_5 & \cos 3\omega_5 & \cos 4\omega_5 \\ 1 & \cos \omega_6 & \cos 2\omega_6 & \cos 3\omega_6 & \cos 4\omega_6 \\ 1 & \cos \omega_7 & \cos 2\omega_7 & \cos 3\omega_7 & \cos 4\omega_7 \\ 1 & \cos \omega_8 & \cos 2\omega_8 & \cos 3\omega_8 & \cos 4\omega_8 \end{bmatrix} \begin{bmatrix} d(0) \\ d(1) \\ d(2) \\ d(3) \\ d(4) \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ D_6 \\ D_7 \\ D_8 \end{bmatrix} \Rightarrow P \cdot d = D$$

看来系数矩阵 P 的赋值比较麻烦, 其实, 回想求离散傅里叶变换时的做法, 可以利用列矩阵 $w' = [w_1; w_2; \cdots; w_8]$ 乘行矩阵 $[0:4]$ 形成一个 8×5 矩阵, 然后求余弦得到 P 。这可以用 MATLAB 语句 $P = \cos(w' * [0:4])$ 轻而易举地列出。因此用下列几行 MATLAB 程序 ag1010 就可以方便地完成最小二乘最优滤波器的设计。

```
N=9; L=floor((N-1)/2); % 列出待求系数序号
w=[0,0.33,0.67,1,1.5,2,2.5,3.14]; % 列出频率向量
D=[1,1,1,1,0,0,0,0]; % 列出预期幅特性向量
P=cos(w'*[0:L]); % 列出 P 矩阵
d=inv(P'*P)*P'*D'; % 求出待求系数
```

程序运行的结果为

```

d = 0.3981
    0.6039
    0.2137
   -0.1205
   -0.1506

```

知道 $d(n)$ 以后, 就可以计算滤波器的频率特性进行校核。

得到的频率特性见图 10.14。因为 $L=4$, 最高的谐波次数为 4, 在 $0 \sim \pi$ 之间幅特性摆动最多两个周期, 达到峰值的次数应为四次, 这从图 10.14 上也看得很清楚。

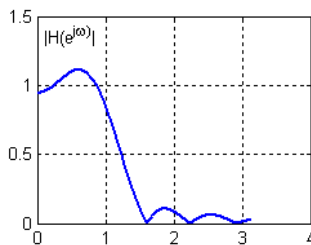


图 10.14 设计出的最优滤波器的频率特性

10.11 信号流图的矩阵建模和计算机求解

任何复杂结构的线性系统, 其内部的信号传输关系, 都可画成信号流图, 在信号与系统、数字信号处理以及自动控制课程中所介绍的唯一解法是梅森公式。那是一个以图论为基础导出的公式, 它没有证明, 只能叫学生死记硬背, 计算起来又非常烦琐, 阶次高一点就极易出错, 更无法用计算机辅助求解。所以在实际中极少有用处。

由于用 MATLAB 在机算矩阵问题上的巨大优势, 激励我们用矩阵来给信号流图建模, 以得出更简明的公式。

设信号流图中有 I 个输入节点, N 个中间和输出节点, 它们分别代表输入信号 u_i ($i=1,2,\dots,I$) 和系统状态 x_j ($j=1,2,\dots,N$)。信号流图代表它们之间的联结关系。用拉普拉斯算子表示后, 任意状态 x_j 可以表为 u_i 和 x_j 的线性组合:

$$x_j = \sum_{k=1}^N q_{jk} x_k + \sum_{i=1}^I p_{ji} u_i \quad (j = 1, 2, \dots, N)$$

将 N 个方程联立, 写成矩阵形式, 可写成:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1N} \\ q_{21} & q_{22} & \cdots & q_{2N} \\ \vdots & \vdots & & \vdots \\ q_{N1} & q_{N2} & \cdots & q_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1I} \\ p_{21} & p_{22} & \cdots & p_{2I} \\ \vdots & \vdots & & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NI} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_I \end{bmatrix}$$

用矩阵符号表示为:

$$\mathbf{X} = \mathbf{Q}\mathbf{X} + \mathbf{P}\mathbf{U}$$

其中: $\mathbf{X} = [x_1, x_2, \dots, x_N]$ 为 K 维状态列向量, $\mathbf{U} = [u_1, u_2, \dots, u_I]$ 为 I 维输入列向量, \mathbf{Q} 为 $N \times N$ 维的联结矩阵, \mathbf{P} 为 $N \times I$ 维的输入矩阵, \mathbf{Q} 和 \mathbf{P} 的元素 q_{ji} 和 p_{jk} 是各信号节点之间的传递系数。移项后, 上式可写为

$$(\mathbf{I} - \mathbf{Q})\mathbf{X} = \mathbf{P}\mathbf{U}$$

由此得到各输入对各信号的传递函数公式:

$$\mathbf{W} = \mathbf{X}/\mathbf{U} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{P}$$

因此, 系统的传递函数矩阵为 $\mathbf{W} = (\mathbf{I} - \mathbf{Q})^{-1} \mathbf{P}$, 这个简明的公式就等价于梅森公式。只要写出 \mathbf{P} 和 \mathbf{Q} , 任何复杂系统的传递函数都可用这个简单的式子求出。

存在的困难在于, 连续系统的传递关系是用常数及拉普拉斯算子 s 组成的, 离散系统的传递关系是用常数及 z 变换算子 z 或 z^{-1} 组成的, 但这个矩阵公式中用到的是普通的矩阵乘法和加法, 无法应用于算子变量。

用 MATLAB 的符号运算(Symbolic)工具箱解决了这个问题。从下面的数字例中可看出其做法。

【例 10.11】 设系统的信号流图如图 10.15 所示, 求以 u 为输入, x_8 为输出的传递函数。

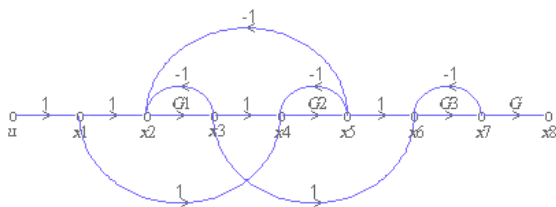


图 10.15 系统的信号流图

解: 先列出信号流图方程组, 把每一个信号写在等式左端, 右面是其他信号的线性组合。信号间的传递关系先笼统地用符号 G_1, G_2, \dots 来表示, 可得以下八个联立方程:

$$\begin{aligned} x_1 &= u \\ x_2 &= x_1 - x_3 - x_5 \\ x_3 &= G_1 * x_2 \\ x_4 &= x_3 + x_1 - x_5 \\ x_5 &= G_2 * x_4 \\ x_6 &= x_3 + x_5 - x_7 \\ x_7 &= G_3 * x_6 \\ x_8 &= K * x_7 \end{aligned}$$

用矩阵表示, 可写成:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & G_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & G_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & G_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & K & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u = \mathbf{QX} + \mathbf{PU}$$

因 x_8 未出现在右端, \mathbf{Q} 的最右应补上一个全零列, 成为 8×8 矩阵。以 u 为输入, \mathbf{X} 为输出的系统传递函数为

$$\mathbf{W} = (\mathbf{I} - \mathbf{Q})^{-1} \mathbf{P}$$

它有八行, 最后一行是 $\mathbf{W}(8) = x_8/u$, 即本题要求出的结果。

建模已经完成,下面就可利用 MATLAB 编程解题。由于使用了符号变量,先要用 `syms` 语句定义符号自变量,矩阵 Q, P 中如果有一个元素是符号变量,就要把矩阵定义为符号矩阵,定义的方法是把符号变量放在 Q, P 的第一个元素赋值语句内。这样我们就不难看懂以下的 MATLAB 程序 `ag1011`。

```
syms G1 G2 G3 K      % 定义字符自变量
Q(3,2)=G1;           % 采用字符矩阵,第一条赋值语句右端必须是字符变量
Q(2,1)=1;Q(2,3)=-1;Q(2,5)=-1;      % 列出联结矩阵
Q(4,3)=1;Q(4,1)=1;Q(4,5)=-1;
Q(5,4)=G2;
Q(6,3)=1;Q(6,5)=1;Q(6,7)=-1;
Q(7,6)=G3; Q(8,7)=K;
Q(:,end+1)=zeros(max(size(Q)),1)    % 加一个全零列,补成方阵
B=[1;0;0;0;0;0;0;0];
I=eye(size(Q));
W=(I-Q)\B            % 求出完整的传递矩阵
W8 = W(8)            % x8 为输出的传递函数为其第八项 W(8)
pretty(W8)           % 给出便于阅读的形式
```

程序运行结果:

$$W8 = G3 * K * (2 * G2 * G1 + G1 + G2) / (2 * G2 * G1 + 2 * G2 * G1 * G3 + G3 + G1 + 1 + G2 + G2 * G3 + G1 * G3)$$

写成美观形式为:

$$W8 = \frac{K G_3 (2 G_2 G_1 + G_1 + G_2)}{(2 G_2 G_1 + 2 G_2 G_1 G_3 + G3 + G1 + 1 + G2 + G2 G_3 + G1 G_3)}$$

读者如果熟悉梅森公式,可以用它算的结果进行比较,应该是完全相同的。同时,可以测定一下所花费的时间,也做一比较。

如果 G_1, G_2, G_3 是自变量 s 的函数,

$$G_1 = \frac{s}{s+1}, G_2 = \frac{3}{s+4}, G_3 = \frac{1}{s^2 + 5s + 6},$$

那自变量中就不该有 G_i 出现,第一条语句应改为 `syms K s`,再把其中的三条赋值语句改为:

```
Q(3,2)=s/(s+1);Q(5,4)=3/(s+4);Q(7,6)=1/(s^2+5*s+6);
```

修改后的程序命名为 `ag1011a`,运行此程序的结果为:

$$W8 = \frac{K(s^2 + 13s + 3)}{2s^4 + 28s^3 + 111s^2 + 161s + 49}$$

读者可以再用梅森公式来作笔算,没有半小时以上,恐怕得不到结果。而且出错的概率极高。这例题还只到四阶,阶次再高,怎么解?用哪种方法好?相信大家从这个例子中必能达到毫无悬念的共识。当然要有一个前提,老师和学生都要会用 MATLAB 编程解题。

这个例子是对连续系统的,用的算子变量是 s ,对离散系统,只要把算子变量换成时

移算子 z ，或 $q=z^{-1}$ ，其他一切都同样处理即可，具体细节可参阅例 8.6.4。

10.12 自动控制系统的矩阵建模

众所周知，系统结构图与信号流图是等价的，所以上述信号流图的矩阵建模也适用于线性自动控制系统。在自控中，各条支路的运算函数一般比较复杂，因为这些支路往往代表一个部件，本身就是一个高阶的子系统。为了表示部件动态特性的方便，很少用信号流图，多用结构图。但这不要紧，矩阵建模的关键不在于用什么图，而在于列写联立方程。用结构图同样可以列出系统的方程组。

若系统比较简单，通常就可以用并联、串联和反馈三种简单情况的公式来求合成的传递函数，即使如此，笔算仍是令人头痛的事。若系统的节点数（也就是信号数）比较多，结构又比较复杂，按现在书上教的手工推导方法，往往就要把节点作移位，使系统变换成上述三种简单情况的叠加，这很容易出错，而且往往会因此改变一些信号的性质，失去其结构图上原有的重要物理意义。

我们发现，用线性方程组的矩阵解法处理这个难题是最好的选择。首先根据系统结构图，自由选择若干个最关心的信号点组成向量，分别列出它们的方程，只要方程的右端出现的信号都在所选的信号向量 \mathbf{X} 之中。这样就可写出方程组的矩阵形式 $\mathbf{X} = \mathbf{Q}\mathbf{X} + \mathbf{P}\mathbf{U}$ ，然后就用公式 $\mathbf{W} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{P}$ 求出对各点的传递函数。这样的解法不但可以保持各信号节点的原来意义，而且只要方程列写正确，推导过程的正确性将由计算机软件来保证。

【例 10.12】 对图 10.16 所示的随动系统，求以驱动信号 u_1 及测量干扰 u_2 为输入，图中 x_1, x_2, x_3, x_4 四个信号节点为输出的传递函数，要求以美观方式显示系统的跟踪误差 x_1 的表达式，说明它如何受 u_1, u_2 影响。

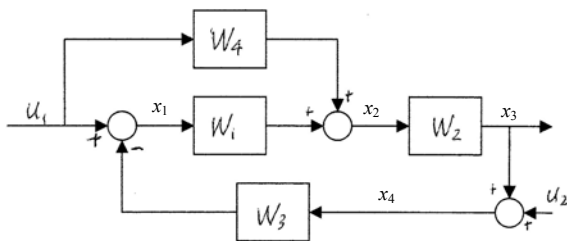


图 10.16 例 10.12 的系统结构图

解：列写方程，有：

$$\begin{aligned} x_1 &= -W_3x_4 + u_1 \\ x_2 &= W_1x_1 + W_4u_1 \\ x_3 &= W_2x_2 \\ x_4 &= x_3 + u_2 \end{aligned}$$

写成矩阵形式为：

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -W_3 \\ W_1 & 0 & 0 & 0 \\ 0 & W_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ W_4 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \Rightarrow X = Q \cdot X + P \cdot U$$

左右对照, 可得知系数矩阵 Q 和 P 的内容, 根据

$$W = X/U = (I - Q)^{-1}P$$

就可以求出以 u_1, u_2 为输入, $x_1 \sim x_4$ 为输出的系统的传递函数。这里的 Q 是 4×4 矩阵, 当然要靠机算求 $(I - Q)$ 的逆阵。 x_1 点的信号可用 $x_1 = W(1,1) \cdot u_1 + W(1,2) \cdot u_2 = W(1,:) \cdot U$ 求出, 因此可列出解本题的程序 ag1012 如下:

```
syms W1 W2 W3 W4 u1 u2
Q(1,4)=-W3; Q(2,1)=W1;           % 给 Q 赋值, 注意第一个赋值的是符号变量
Q(3,2)=W2; Q(4,3)=1;
P(2,1)=W4; P(1,1)=1; P(4,2)=1;   % 给 P 赋值, 注意第一个赋值的是符号变量
W=inv(eye(4)-Q)*P                 % 信号流图方程解
x1=W(1,:)*[u1;u2]                 % 计算各信号点在 u1,u2 作用下的输出 x1
pretty(W(1,1)), pretty(W(1,2)),   % 美观显示对跟踪误差的两个传递函数
pretty(x1)                         % 美观显示输出误差项
```

程序运行结果为:

```
W = [ 1/(1+w1*w2*w3)-w2*w3/(1+w1*w2*w3)*w4,      -w3/(1+w1*w2*w3) ]
     [ w1/(1+w1*w2*w3)+1/(1+w1*w2*w3)*w4,        -w1*w3/(1+w1*w2*w3) ]
     [ w2*w1/(1+w1*w2*w3)+w2/(1+w1*w2*w3)*w4,    -w1*w2*w3/(1+w1*w2*w3) ]
     [ w2*w1/(1+w1*w2*w3)+w2/(1+w1*w2*w3)*w4,    1/(1+w1*w2*w3) ]
x1 = [ (w2*w1/(1+w1*w2*w3)+w2/(1+w1*w2*w3)*w4)* u1-w1*w2*w3/(1+w1*w2*w3)*u2 ]
```

$W(i,j)$ 分别表示第 j 个输入对第 i 个输出的传递函数, W 是 4×2 矩阵, 说明这种方法的计算效率非常高, 一次把 8 个传递函数都求出来了。不过其书写的形式太难读了, `pretty` 是符号运算工具箱中的一个命令, 它可以把表达式中的分子分母和指数分别放在上下不同的行中, 比较美观也易于阅读, 当然远达不到排版水平。显示的结果如下:

$$W(1,1) = \frac{1}{1+W_1W_2W_3} - \frac{W_2W_3W_4}{1+W_1W_2W_3}, \quad W(1,2) = -\frac{W_3}{1+W_1W_2W_3}$$

$$x_1 = \left(\frac{1}{1+W_1W_2W_3} - \frac{W_2W_3W_4}{1+W_1W_2W_3} \right) u_1 - \frac{W_3}{1+W_1W_2W_3} u_2$$

用这样的推导也可以研究环节参数敏感度的问题, 只要把环节的传递函数给出一个增量, 例如令 W_3 为 $W_3 + dW_3$ 就可研究 dW_3 引起的输出。

用结构图矩阵解法还可以对系统的详细特性进行推导。例如本题中, 设 $W_1 = \frac{1}{0.05s+1}$,

$W_2 = \frac{2}{(0.01s^2 + 0.1s + 1)s}$, $W_3 = 1$, $W_4 = \frac{0.5s}{0.001s + 1}$, 此时可设拉普拉斯算子 s 为符号变量,

W_1, W_2, W_3 为 s 的函数, 放在程序 ag1012 的前部, 把程序的前两句换成为下面 5 条语句, 程序取名为 ag1012a:

```
clear, syms s u1 u2
W1=1/(0.05*s+1)
W2=2/(0.01*s^2+0.1*s+1)/s
W3=1;W4=0.5*s/(0.001*s+1)
Q(2,1)=W1;Q(1,4)=-W3; % 因 W3=1, 把符号变量 w1 调到第一个赋值
..... (以下同程序 ag1012)
```

运行此程序, 将得出一个比较冗长的结果, 人工做近似整理后的结果为:

$$W(1,1) = \frac{(s+20)(s+10)s^2}{s^4 + 30s^3 + 300s^2 + 2000s + 4000}$$

$$W(1,2) = \frac{-(s+20)(s^2 + 10s + 100)s}{s^4 + 30s^3 + 300s^2 + 2000s + 4000}$$

$$x_1 = \frac{(s+20)(s^2 + 10s)su_1 - (s+20)(s^2 + 10s + 100)su_2}{s^4 + 30s^3 + 300s^2 + 2000s + 4000}$$

手工推导这个结果恐费时很多, 且出错概率很大, 这是教育现代化 (这里是用矩阵建模并用软件工具解题) 给我们带来的好处, 要充分利用。

结束语

学术性的著作能否搞得活泼些？我们来试一试！对于线性代数这门课，写几句‘顺口溜’，可能有助于读者记忆。这个‘顺口溜’是：

“一、两、三，三、两、一，还要三句作对比！”，解释如下：

1. ‘一、两、三，’

- **一个中心**，是用消元法化矩阵为行阶梯形式。这是贯串整个线性代数的核心方法；不仅要掌握，而且还能从行阶梯形式中看出许多问题，如方程的解、矩阵求逆、矩阵的秩（rank）、行列式（det）、线性无关列、线性相关列、自由变量、解空间等。用到的MATLAB子程序和命令主要有 `gauss`，`rref`，`lu` 等。

- **两个方向**，是从行向和列向两个基本方向来分析系统。

行向分析主要是考虑方程特性，包括方程的相依性和独立方程的数目（也就是行向的秩），它们的解的特性；

列向分析主要是考虑未知数，也就是变量的特点，变量的相关性和枢轴变量的数目，枢轴变量向量张成的空间。自由（非枢轴）变量向量张成的空间等。

- **三个类型**，指方程的三种类型（适定、欠定和超定）及对应的三种解的求法。

MATLAB 把它们统一为一个运算命令 $\mathbf{x}=\mathbf{A}\backslash\mathbf{b}$ ，但必须深入理解，正确使用。

适定方程： \mathbf{A} 的秩等于变量数，此时方程有唯一解， $\mathbf{x}=\text{inv}(\mathbf{A})*\mathbf{b}$ 。

欠定方程： \mathbf{A} 的秩小于变量数，行阶梯式中的枢轴列数是方程秩 r （用 `[U,ip]=rref(A)` 命令），其余 $n-r$ 个非枢轴元素是自由变量。方程的全解可取 $\mathbf{Ax}=\mathbf{b}$ 中对枢轴变量的一个特解加上 $\mathbf{Ax}=0$ 的齐次解集（用 `null` 命令）得到。所以欠定方程是这三个类型中最麻烦的。

超定方程：增广行阶梯形式中的方程数大于变量数，（下方出现等式左端全零而增广项不为零的矛盾方程），此时方程有唯一的最小二乘解。

2. ‘三、两、一，’

- 三个视点，解矩阵方程 $Ax=y$ （或 b ）可从三个视点出发：求线性方程的解 x ，研究系数向量矩阵 A 张成的空间，以及线性变换的像空间 y 。

从求线性方程的解 x 的视点前面已讲得很多了，在此不再赘述。

从系数向量 A 张成的空间，研究列向量空间，线性空间，齐次解空间，向量的内积和正交性、基向量等，构成正交的基向量用 `qr` 命令或 `ATLAST` 的 `gschmidt` 命令。

从线性变换的像空间 y 的特性，可研究变换对图形特征的影响，特征值和特征向量（用 `eig` 命令）、基向量的坐标变换，实对称矩阵的对角化（用 `eig` 或 `orth` 命令）及对二次型的影响等。

- 两个动力，工程应用是‘需求牵引’，软件工具是‘技术推动’。它们是线性代数生命力之源。
- 一个目标，首先要满足本科后续课程的广泛而丰富的需要，然后才是其他。

3. ‘还要三句作对比！’

三句作对比的话是：

线性代数很**抽象**吗？看了本书后，你应该知道它的概念都基于空间**形象**。

线性代数很**冗繁**吗？学了本书后，你应该懂得它的计算全可有**简明**程序。

线性代数很**枯燥**吗？读了本书后，你应该发现它的应用极其广泛而**精彩**。

能体会到这三点对比，就基本达到了这本书的目的。

附录 A

关于 MATLAB 基本部分 函数索引的说明

MATLAB 函数的检索可以用命令：

```
>> which 函数名
```

实现，检索结果中还会给出该函数所在的目录路径，例如在我的系统中输入：

```
>> which ezplot
```

得到：C:\MATLAB6p1\toolbox\matlab\specgraph\ezplot.m

因而用户可以知道从哪个函数库中去寻找。因为现在 MATLAB 的版本很多，在每个版本中函数所在的库又常常不同，用户在自己所用的 MATLAB 系统中去查找是最好最可靠的方法，用固定的纸面索引来检索反而容易误导读者。因此，我们将这个附录予以取消，特此说明。

附录 B

有关美国“用软件工具增强线性代数教学”计划的资料

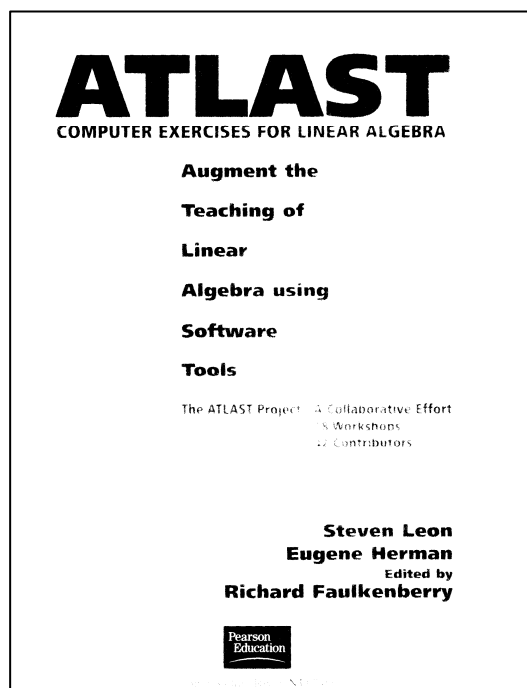
注：ATLAST 是 Augment the Teaching of Linear Algebra using Software Tools(用软件工具增强线性代数教学)的缩写。该项目受美国国家科学基金会六年的资金资助，ATLAST MANUAL 是根据六年中积累的成果编出的手册。这是项目负责人 Steven Leon 为该手册第二版（2003 年出版）写的序言

I. ATLAST MANUAL 序言

协作的成果

这本书提供了在 1992—1997 年间参加 18 个 ATLAST 计划的研讨班的成员们创造性工作的成果。研讨班成员参与了设计计算机上机练习、大作业和修订用于本科生线性代数课程的教学计划。编辑们从 ATLAST 全部资料的数据库中选择了复盖线性代数作为起步课程的纲要性习题。每章都分为两节。第一节由较短小的习题构成；第二节则为较长的大作业。所有的习题与大作业都基于 MATLAB 软件包。

与其他计算机练习手册不同，本集代



表了许多观点的广泛合作。由于存在许多子任务，从而会有大量的重复内容，这是不足为奇的。在这种情况下本书所取用的通常是提交的各种版本的集成。后来的研讨班所研讨的许多课堂教学计划也都被本书采纳为计算机大作业。此外，编辑们遇到不少原先是很短的有洞察力的练习，如把它们开发深一些，就会对学生很有益。于是，一些短的练习题被扩展成了较长的大作业。对所有的成果都进行了重新编辑，以维持本书一致的格式与风格。

第 2 版的新亮点是什么呢

1. 新的大作业

1995—1997 年间，有七个 ATLAST 研讨班致力于实现线性代数可视化方面，以便开发讲课计划，很多此类讲课计划已被本书采纳为学生的计算机大作业。

2. 新练习

本版的练习部分也得到了扩充。ATLAST 研讨班开发的讲课计划中的一些单元被结合为新的习题。

3. 新的 M-文件

书中有 10 个新的 MATLAB 程序（M-文件），可以帮助做习题和大作业。对许多其他的 ATLAST M-文件都进行了修改和改进。所有的 M-文件都已更新成与 MATLAB 6.5 版本兼容。

利用软件来教线性代数

计算机已经改革了我们教学的方式，这对线性代数尤其正确。现代的软件包使大学生们更容易探讨线性代数的应用，并通过实践发现重要的理论成果。也许最重要的在于，计算机对课程中的主要概念。例如线性系统、线性变换、特征值、特征向量和正交性等都可给学生提供直观的可视化手段。

近年来，软件工具的图形功能有了巨大的发展和改进。现在完全可能在课堂上演示复杂的可视化模型，以便说明线性代数的很多主要理论概念。这就使得学生可以通过直观的几何形象进而了解事物的本质。

过去，在尚无个人机和膝上机之前，很难在课堂上计算和解出一些很有趣的应用问题。现在，有了软件的帮助，几乎不花计算时间，就可表现出线性代数的宽广应用价值。从而，学生们能更加体验到线性代数是一个多么强有力的工具，实际上它也是解决所有应用数学问题时有的一个重要工具。

计算机也是数学探索的极好载体。在数学软件包（如 MATLAB）的帮助下，人们可很快生成各种例题。如果这些例子是仔细选择的，那么可以使学生很容易地发觉许多有趣和重要的结果。许多定理如果用这种方式探索，那么它对学生就会变得更有意义了。本书包含多种多样的数值实验，它需要学生从观察数据来猜想其普遍结果，然后学生们将受到激励，去证明或解释其猜想的正确性。

MATLAB 软件包

ATLAST 工程的最初目标就是鼓励把软件用于线性代数教学。市场上有许多出色的数学软件包，我们相信 MATLAB 是用于线性代数教学的最佳选择。MATLAB 中把矩阵视为基本数据变量。的确，MATLAB 是一种几乎完全围绕矩阵建立起来的软件包。而且，MATLAB 有许多子程序来生成随机数矩阵和各种类型的结构化矩阵。这使得在课堂上很容易迅速生成有趣的例子。虽然 ATLAST 数据库中的练习题可采用各种软件包，但本书选择的习题全部适合用 MATLAB，还开发出了 MATLAB 子程序（M-文件）集，来陪伴本书的出版。

ATLAST M-文件

与此书相关的许多 M-文件都设计来形象地解释线性代数中的重要概念，如坐标系、线性变换与特征值等。其余的 M-文件说明可视化的应用，例如用线性变换来做计算机动画或用矩阵分解来处理数字图像。还有些 M-文件可用来生成一定结构的矩阵，然后要求学生们去探讨特定矩阵的性质。

ATLAST 的全部 M-文件集可从 ATLAST 网站下载，网址为：

<http://www.umassd.edu/SpecialPrograms/Atlast/>

这些文件是本书中许多练习题和大作业所需要的。这些文件是定期更新的，因此读者应常常看网页，以获得最新版本。

ATLAST 计划项目的历史

ATLAST 是“利用软件工具来增强线性代数教学”的缩写词。该计划最初来自于国际线性代数学会（ILAS—International Linear Algebra Society）所属教育委员会的活动的结果。该委员会的 Steven Leon 委员收到 ILAS 的一个种子合同，要他写一个基金项目的申请，这个项目是促进软件在线性代数教学中的应用。差不多在同时，在国家科学基金会（NSF—National Science Foundation）支持下成立了另一个独立的线性代数课程研究小组（LACSG—Linear Algebra Curriculum Study Group）。这个小组对线性代数教育改革提出了不少建议。特别是他们推荐，在线性代数的第一门课程中就得采用面向矩阵的观点。他们还推荐在线性代数教学中要用软件。最初的 ATLAST 计划可看成是实施 LACSG 小组推荐意见的最及时的响应与努力。

ATLAST 计划还从本科教育分会获得提高本科教师水平基金（UFE—Undergraduate Faculty Enhancement）的资助。初始的合同支持了 1992 年夏天的 5 个研讨会以及 1993 年夏天增添的 5 个研讨会。每个研讨会限 30 人参加。1992 年就有 250 人申请这 150 个名额。那时，许多资深的申请人不得被拒绝，不过 1993 年那些再次申请者都被接纳了。

1994 年夏天，ATLAST 计划得到了国家科学基金会的补充资助，用于在加州大学圣迭

哥分校召开高级开发者研讨会。25 位 ATLAST 的前参加者被邀请来开四天研讨会，目的是为 ATLAST 书开发附加练习。参加者分为多个小组，对线性代数各个专题进行工作。在研讨会期间对计划进行测试，并在随后的一个月中进行修改与完善。圣迭哥研讨会做出了大量的引人瞩目的练习题，其中相当一部分已收集在本书中。

第二个 UFE 合同使得 ATLAST 计划能在 1996—1997 学年继续工作。这期间举行了五个定期的研讨班和两个开发者的研讨班。这些研讨班的重点是开发课堂授课计划，强调利用 ATLAST 的 M-文件帮助学生探索线性代数的重要概念并使之可视化。这些计划的某些版本已放到 ATLAST 网页上。许多课堂授课计划又以计算机习题或大作业的形式重新编写并收集在本书中。

最初的 ATLAST 申请书提出的目标是，希望这个计划能在线性代数改革运动中起到显著的作用。除了开发计算机习题之外，ATLAST 研讨班还花时间讨论线性代数大纲以及和线性代数教学相关的其他问题。研讨会为线性代数教师提供了很好的机会来交流和共享大家的想法与经验。许多参加者在线性代数教育改革中继续起着积极的作用。

将来的计划

ATLAST 书的版权全部归 Dartmouth UMASS 基金会所有。该基金会建立了一个 ATLAST 账号，它将用于支持今后与线性代数教育相关的活动。

（以下感谢词，ATLAST 研讨班的领导者及参与者的名单等略去。）

Steven Leon, ATLAST 项目负责人
sleon@umassd.edu

注：作者 Steven Leon，现为马萨诸塞大学达特茅斯分校数学系首席教授，ILAS（国际线性代数协会）、MAA（美国数学学会）和 SIAM（美国工业和应用数学学会）成员。主要从事科学计算、线性代数和应用数学领域的研究。

II. ATLAST MANUAL 详细目录

第 1 章 线性系统	1
练习题	
线性系统和矩阵化简	1
矩阵的秩	3
实践中的线性系统	3
大作业	
1. 两个未知数的线性系统的图形分析	6
2. 三个未知数的线性系统的图形分析	6
3. 简化行阶梯形式的唯一性	8
4. 非线性方程的解	8
5. 设计一个滑雪跳跃	10
6. 用三次样条法设计	11
7. 网络中的流	14
第 2 章 矩阵代数	17
练习题	
代数规则	17
实践中的矩阵	18
特殊矩阵	19
线性系统和矩阵代数	20
矩阵的逆	21
特殊矩阵的逆	24
矩阵的秩	25
大作业	
1. 矩阵乘法的观察	26
2. 矩阵乘积的逆阵和转置	27
3. 对称和斜对称矩阵	27
4. 图和飞行航线	29
5. 马尔可夫过程入门	30
6. 各态历经矩阵的一个特性	31
7. 各态历经矩阵的其他特性	32
8. 一个风险马尔可夫过程	32
9. Leslie 人口模型	33
10. 电力网络	35
11. 密码和矩阵乘法	39
12. 交替矩阵	41

13. 基本矩阵和行简化	43
14. 基本矩阵和它们的逆	44
15. 分块矩阵	45
16. 有多少奇异矩阵?	47
17. 几何级数和矩阵求逆	48
第 3 章 行列式	51
练习题	
行列式和行运算	51
行列式和矩阵运算	52
特殊矩阵的行列式	54
字母矩阵	56
大作业	
1. 特殊三角矩阵的行列式	58
2. 插值	59
第 4 章 向量空间概念	63
练习题	
线性无关和线性相关	63
张集、基和维	65
与矩阵相联系的子空间	66
由基构成的坐标系	67
构成基的方法	68
子空间可具有很多的基	69
大作业	
1. 坐标游戏	71
2. 张集的可视化	72
3. 凸组合和图像变形	72
4. 向量空间的维数与系数矩阵	76
5. 一个具体应用	77
6. 行空间和列空间的比较	78
7. 列运算和零空间基	79
8. 单边求逆	81
9. 自身交换	82
10. 图和入射矩阵	83
11. 两个向量空间交集的基	85
第 5 章 线性变换	87
练习题	
由 \mathbf{R}^n 到 \mathbf{R}^m 的线性变换	87
由 \mathbf{R}^n 到 \mathbf{R}^n 的线性变换	88

大作业	
1. 寻找图像和核的基-----	91
2. 计算机图形学中的应用-----	93
3. 变换器设计-----	102
4. 仿射变换和分形图像-----	106
5. 偏航、俯仰和滚动-----	108
第 6 章 正交性-----	115
练习题	
平面中的向量-----	115
\mathbf{R}^n 中的向量-----	117
正交集、正交集和正交向量矩阵-----	119
基的延伸-----	121
最小二乘问题-----	123
大作业	
1. 统计相关性-----	125
2. 计算最小二乘圆-----	129
3. 计算水污染的矩阵模型-----	132
第 7 章 特征值-----	137
练习题	
特征值和特征向量的几何形象-----	137
特殊矩阵-----	139
特征多项式-----	140
大作业	
1. 指数法-----	142
2. 字母的特征值-----	143
3. 哈密尔顿矩阵的特征值-----	145
4. 正定和半正定矩阵的特征值-----	146
5. 马尔可夫链-----	147
6. 航天飞行器的定向-----	150
7. Gerschgorin 圆-----	152
第 8 章 奇异值分解-----	157
练习题	
1. Cholesky 分解和矩阵平方根-----	158
2. 奇异值和数字秩-----	159
3. 欠秩的最小二乘问题-----	160
4. 伪逆和线性系统-----	161
5. 伪逆的代数特性-----	162

大作业

1. 奇异值分解-----	164
2. 奇异值分解的可视化-----	165
3. SVD 和数字图像处理-----	166
4. 空间的圆-----	171
附录 A: MATLAB-----	175
附录 B: ATLAST 特殊矩阵生成程序-----	187
附录 C: ATLAST 子程序-----	193
参考书目	

III. ATLAST 子程序目录

特殊矩阵生成程序

序 号	函 数 名	功 能	说 明
1	gridmat(n)	X 坐标矩阵	$\text{ones}(n,1) * [1:n]$
2	maxmat(n)	最大下标矩阵	$a(i,j) = \max(i,j)$
3	minmat(n)	最小下标矩阵	$a(i,j) = \min(i,j)$
4	checker(n)	1-0 交替矩阵	
5	achecker(n)	0-1 交替矩阵	
6	signmat(n)	正负 1 交替矩阵	
7	backiden(n)	反向单位矩阵	$\text{fliplr}(\text{eye}(n))$
8	jordan0(n)	约当 0 矩阵	$\text{diag}(\text{ones}(n-1,1), 1)$
9	jordan1(n)	约当 1 矩阵	$\text{eye}(n) + \text{jordan0}(n)$
10	cyclic(n)	单位矩阵循环右移一位	
11	consec(n)	顺序数字矩阵	$\text{reshape}([1:n^2], n, n)'$
12	hconsec(n)	Hankel 顺序数矩阵	
13	Hmatrix(n)	H 字符矩阵	n 奇数
14	Lmatrix(n)	L 字符矩阵	
15	Nmatrix(n)	N 字符矩阵	
16	Tmatrix(n)	T 字符矩阵	n 奇数
17	Xmatrix(n)	X 字符矩阵	
18	Ymatrix(n)	Y 字符矩阵	n 奇数
19	Zmatrix(n)	Z 字符矩阵	

ATLAST 子程序目录

序 号	函 数 名	功 能	说 明
1	achecker(n)	0-1 交替矩阵	
2	Alphabet Matrix	H, L, N, T, X, Y, Z 7 个字符矩阵	
3	backiden(n)	反向单位矩阵	$\text{fliplr}(\text{eye}(n))$
4	cogame	猜坐标游戏	
5	colbasis	列向量基生成函数	$[R, jp] = \text{rref}(A); C = A(:, jp)$
6	consec(n)	顺序数字矩阵	$\text{reshape}([1:n^2], n, n)'$
7	convexcombs(x,y,h)	画出 x,y 的凸组合	
8	cyclic(n)	单位矩阵循环右移一位	
9	drawvec(v,color,s)	画向量函数	
10	editmag(B,imag,back)	编辑由 makeimag 生成的图形	

续表

序 号	函 数 名	功 能	说 明
11	eigplot(A)	绘制矩阵 A 的特征值	
12	eigshow(A)	动态显示 x 与 Ax 的变化关系	
13	fractal	分形演示程序	
14	gersch(A)	绘制 gerschgorin 特征值圆	
15	gridmat(n)	X 坐标矩阵	ones(n,1)*[1:n]
16	gschmidt(A,1)	施密特法求正交基	
17	hconsec(n)	Hankel 顺序数矩阵	
18	inciden(E)	入射矩阵	
19	jordan0(n)	约当 0 矩阵	diag(ones(n-1,1),1)
20	makeimag(n,imag,back)	制作图形	
21	morph(im1,im2,cmap,n,j,k)	图形变异	
22	movefig(F,M,n)	图形移动	
23	niceimag	逐步改善图形	
24	nulbasis(A)	构成 A 的基础解	
25	other	在下标 1:n 中去掉 ip 后的余下标	用于提取非枢轴列的下标
26	plotangle	画两个向量的夹角	
27	plotline(a,b,c,s)	画直线	
28	powplot	画矩阵指数	
29	randintr(m,n,k,r)	随机矩阵生成	
30	randstoc(n)	随机历经矩阵	
31	rollit(n)	轮子滚动程序	
32	rowcomb(A,i,j,c)	行乘加运算	
33	rowscale(A,i,c)	行乘数运算	
34	rowswap(A,i,j)	行交换	
35	signmat(n)	正负 1 交替矩阵	
36	solution(A,b)	线性方程组的解	
37	spanview	三维空间张集观看	
38	svdimage	奇异值分解图形	
39	transfor	线性变换图形显示	
40	viewspace(u,v,w,color1,color2)	子空间可视化	
41	walk(steps,t)	人走步动画	

这些子程序放在一个名为 atlast65 的文件夹中，可以从网址：

<http://www.umassd.edu/SpecialPrograms/atlast/>

下载，也可从电子工业出版社博文视点的网址：

<http://www.broadview.com.cn/MATLABDownload/default.aspx>

与本书的程序集同时下载。

参考文献

- [1] Steven J. Leon, Linear Algebra with Applications (6th Edition), 2002, 影印版“线性代数”, 机械工业出版社, 2004, ISBN 7-111-15216-6, pp.545, 机械工业出版社影印
- [2] David C. Lay, Linear Algebra and Its Application (3rd Edition), 2004, ISBN: 0201709708, pp.492+76, 电子工业出版社影印
- [3] S. K. Jain & A.D. Gunawardena, Linear Algebra: An Interactive Approach, Brooks/Cole, 2004, ISBN 0-534-40915-6, pp.412, 机械工业出版社影印
- [4] Lee W. Johnson, R.Dean Riess, J. T. Arnold, Introduction to Linear Algebra (5th Edition) 2004, ISBN: 0-201-65859-3, 机械工业出版社影印
- [5] W. Keith Nicholson, Elementary Linear Algebra, 2004, (1st Edition), ISBN: 0-07-301876-7
- [6] Jimmie Gilbert, Linda Gilbert, Linear Algebra and Matrix Theory, (2nd Edition), Brook/Cole, 2004, ISBN: 0-534-40581-9
- [7] Steven J Leon, Eugene Herman, Richard Faulkenberry, ATLAST Manual, 2/E, Publisher: Prentice Hall, 12/19/2003, pp.270, ISBN: 0-13-101121-9
- [8] Hal G. Moore, Adil Yaqub, A First Course in Linear Algebra with Applications, 3e, 2001, ISBN: 0-12-505760-1
- [9] Erwin Kleinfeld, Margaret Kleinfeld, Understanding Linear Algebra Using MATLAB, Prentice Hall, 2001, ISBN: 0-13-060945-5
- [10] 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
- [11] 陈怀琛, 吴大正, 高西全. MATLAB 及其在电子信息课程中的应用. 北京: 电子工业出版社, 2002
- [12] 陈怀琛. 数字信号处理教程——MATLAB 释义与实现. 北京: 电子工业出版社, 2004
- [13] 吴文俊. 数学机械化. 北京: 科学出版社, 2003
- [14] 陈怀琛, 高淑萍, 杨威. 工程线性代数 (MATLAB 版). 北京: 电子工业出版社, 2007



《线性代数实践及 MATLAB 入门(第2版)》读者交流区

尊敬的读者:

感谢您选择我们出版的图书,您的支持与信任是我们持续上升的动力。为了使您能通过本书更透彻地了解相关领域,更深入的学习相关技术,我们将特别为您提供一系列后续的服务,包括:

1. 提供本书的修订和升级内容、相关配套资料;
2. 本书作者的见面会信息或网络视频的沟通活动;
3. 相关领域的培训优惠等。

请您抽出宝贵的时间将您的个人信息和需求反馈给我们,以便我们及时与您取得联系。

您可以任意选择以下三种方式与我们联系,我们都将记录和保存您的信息,并给您提供不定期的信息反馈。

1. 短信

您只需编写如下短信: B07223+您的需求+您的建议

发送到1066 6666 789 (本服务免费,短信资费按照相应电信运营商正常标准收取,无其他信息收费)

为保证我们对您的服务质量,如果您在发送短信24小时后,尚未收到我们的回复信息,请直接拨打电话(010) 88254369。

2. 电子邮件

您可以发邮件至 jsj@phei.com.cn **或** editor@broadview.com.cn。

3. 信件

您可以写信至如下地址: 北京万寿路173信箱博文视点, 邮编: 100036。

如果您选择第2种或第3种方式,您还可以告诉我们更多有关您个人的情况,及您对本书的意见、评论等,内容可以包括:

- (1) 您的姓名、职业、您关注的领域、您的电话、E-mail地址或通信地址;
- (2) 您了解新书信息的途径、影响您购买图书的因素;
- (3) 您对本书的意见、您读过的同领域的图书、您还希望增加的图书、您希望参加的培训等。

如果您在后期想退出读者俱乐部,停止接收后续资讯,只需发送“B07223+退订”至10666666789即可,或者编写邮件“B07223+退订+手机号码+需退订的邮箱地址”发送至邮箱: market@broadview.com.cn 亦可取消该项服务。

同时,我们非常欢迎您为本书撰写书评,将您的切身感受变成文字与广大书友共享。我们将挑选特别优秀的作品转载在我们的网站(www.broadview.com.cn)上,或推荐至CSDN.NET等专业网站上发表,被发表的书评的作者将获得价值50元的博文视点图书奖励。

我们期待您的消息!

博文视点愿与所有爱书的人一起,共同学习,共同进步!

通信地址: 北京万寿路 173 信箱 博文视点 (100036)

电话: 010-51260888

E-mail: jsj@phei.com.cn, editor@broadview.com.cn

www.phei.com.cn

www.broadview.com.cn

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036