HZ BOOKS

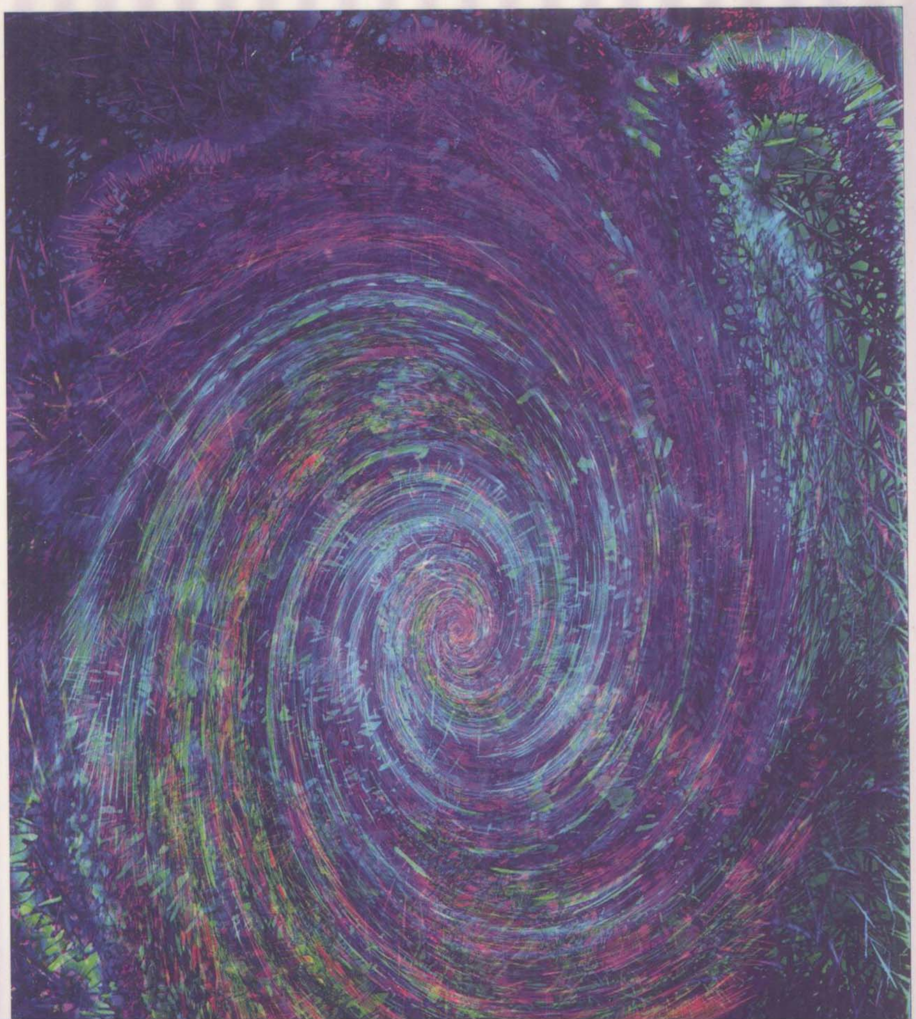
PEARSON
Prentice Hall

经 典 原 版 书 库

科学计算导论

使用MATLAB的矩阵向量方法

(英文版 · 第2版)



经典原版书库

科学计算导论

使用MATLAB的矩阵向量方法

(英文版·第2版)

(美) 查尔斯 F. 范龙 著
康奈尔大学

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

科学计算导论：使用MATLAB的矩阵向量方法（英文版·第2版）/（美）范龙（Van Loan, C. F.）著. -北京：机械工业出版社，2005.4

（经典原版书库）

书名原文：Introduction to Scientific Computing: A Matrix-Vector Approach Using MATLAB, Second Edition

ISBN 7-111-15895-4

Ⅰ.科… Ⅱ.范… Ⅲ.科学计算-英文 Ⅳ.N32

中国版本图书馆CIP数据核字（2004）第141026号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2005年4月第1版第1次印刷

787mm × 1092mm 1/16 · 23.75印张

印数：0 001-3 000册

定价：39.00元

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum’s Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国

家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周立柱	周克定	周傲英	孟小峰	岳丽华
范　明	郑国梁	施伯乐	钟玉琢	唐世渭
袁崇义	高传善	梅　宏	程　旭	程时端
谢希仁	裘宗燕	戴　葵		

秘　书　组

武卫东　　温莉芳　　刘　江　　杨海玲

Dedicated to

Cleve B. Moler

第2版前言

在这次修订中，我将本书内容升级到MATLAB 5 以上版本，并增添了60个新问题和若干新专题。本书第1章增加了一节讲述结构数组和单元数组的内容，增加了一节讲述如何产生信息更为丰富的图形。对三角内插法的简单讨论现在放到了第2章（对该问题的快速傅里叶变换解法则放到了第5章）。第5章包括了对稀疏数组的简要讨论，这为第6章和第7章中对线性方程稀疏方法和最小二乘方的研究奠定了基础，在这些章节中，单元数组的使用充实了分块矩阵材料。第8章的轨道问题解采用了简化结构，因而简化了表达法。第9章则对ode23进行了较详细的讨论。

我十分感谢Carl de Boor和Mike Overton，他们对如何改进本书第1版提出了许多宝贵的建议。另外，Rob Corliss、Tim Davis以及Dan Druck也提供了许多好的意见。同时，我也要感谢我的助手Cindy Robinson和我的妻子Marian，是她们的支持帮助我完成了本书的修订。

最后，我非常荣幸地再次将此书献给Cleve Moler教授，感谢他在科学计算方面做出的卓越贡献。

第1版前言

MATLAB提倡试验各种有趣的数学概念，从而影响了我们从事科学计算研究的方式。它支持可视化和向量级的思考方式，使我们能够将重点放在高层次的问题上。现在，由于MATLAB在研究与应用领域的广泛使用，它已经成为计算学科中的提升力量。

由于同样的理由，MATLAB也提升了科学计算导论课程的教学水平。学生们需要与数学打交道，因为数学支持着学会的每一个新方法。他们需要用图示来领会收敛和误差；需要有一种矩阵向量程序设计语言来加强对线性代数的理解，并为高级数组级计算做好准备；需要利用一个完整的问题解决环境（这个环境能够使用最新的算法研究成果）以支持科学研究和工程设计。总之，他们需要MATLAB。

本书中讲述了一学期科学计算入门课程中通常会涵盖的各个主题。但是为了使学生们能理解连续数学和计算方法之间的联系，亦将图示法和矩阵向量运算操作融入其中。全书9章中的每一章都会有一个定理。所有分析都辅之以有利于建立感性认识的计算实例。实际上，本书是围绕着200多个M文件中的例子组织的。这些代码对这本教材非常重要。总体而言，它们传达了对关键数学概念的直观认识，有助于学生们体会数值计算的精妙之处。此外，它们还说明了MATLAB的许多功能，这对学生今后的计算生涯可能是很有益处的。

高级计算方法在讨论并行自适应求积法和并行矩阵计算法的章节中作了简要介绍。递归处理方法涉及了均差、自适应近似法、求积、快速傅里叶变换、Strassen矩阵乘法以及Cholesky分解。

数值线性代数没有仅限于在矩阵计算单元中介绍。由于图形表示贯穿于全书，因而从第1章开始的整个内容都涉及数值线性代数。第1章虽然只是MATLAB的入门指导，但该章有许多例子，为以后数值算法的学习奠定了基础。

我要感谢康奈尔大学CS222班级的学生们，他们的鼓励是我编写此书的源动力。同时，我也要感谢我的同事Yuying Li和Steve Vavasis，在校订的过程中他们给予我很大的帮助。

Cindy Robinson从1987年以来一直是我的行政助手。在此期间，多亏了Cindy的关心和支持，我才能够完成5本书的写作。

最后，我要感谢Cleve Moler先生，自从我作为密歇根大学的学生踏进他的办公室以来，他在我的学术生涯中起了关键作用。作为教师、博士生导师和MATLAB的推动者，是他将我引入了数学和计算方法的知识殿堂。我非常荣幸地将此书奉献给他。

软　　件

解决本书所涉及的所有问题的软件可以通过Netlib从如下地址获得:

http://www.netlib.org

读者也可以向这一地址netlib@ornl.gov发送一封内容为“send index”的邮件来获得上述的相关信息。通过访问我的网站http://www.cs.cornell.edu/cv/，读者可以获得本书每章末尾提到的M-file。此外，该网站亦包括本书所涉及的文档（勘误表、精选问题解法等）。

Contents

Chapter 1

Power Tools of the Trade

MATLAB is a matrix-vector-oriented system that supports a wide range of activity that is crucial to the computational scientist. In this chapter we get acquainted with this system through a collection of examples that sets the stage for the proper study of numerical computation. The MATLAB environment is very easy to use and you might start right now by running the overview files `intro` and `demo`. There is an excellent tutorial in *The Student Edition of* MATLAB. Our introduction is similar in spirit but also previews the central themes that occur with regularity in the following chapters.

We start with the exercise of plotting. MATLAB has an extensive array of visualization tools. But even the simplest plot requires setting up a vector of function values, and so very quickly we are led to the many vector-level operations that MATLAB supports. Our mission is to build up a linear algebra sense to the extent that vector-level thinking becomes as natural as scalar-level thinking. MATLAB encourages this in many ways, and plotting is the perfect start-up topic. The treatment is spread over two sections.

Building environments that can be used to explore mathematical and algorithmic ideas is the theme of §1.3. A pair of random simulations is used to illustrate how MATLAB can be used in this capacity.

In §1.4 we learn how to think and reason about error. Error is a fact of life in computational science, and our examples are designed to build an appreciation for two very important types of error. Mathematical errors result when we take what is infinite or continuous and make it finite or discrete. Rounding errors arise because floating-point representation and arithmetic is inexact.

§1.5 is devoted to the art of designing effective functions. The user-defined function is a fundamental building block in scientific computation. More complicated data structures are discussed in §1.6, while in the last section we point to various techniques that can be used to enrich the display of visual data.

1.1 Vectors and Plotting

Suppose we want to plot the function $f(x) = \sin(2\pi x)$ across the interval $[0, 1]$. In MATLAB there are three components to this task.

- A vector of x-values that range across the interval must be set up:

$$0 = x_1 < x_2 < \cdots < x_n = 1.$$

- The function must be evaluated at each x-value:

$$y_k = f(x_k), \qquad k = 1, \ldots, n.$$

- A polygonal line that connects the points $(x_1, y_1), \ldots, (x_n, y_n)$ must be displayed.

If we take 21 equally spaced x-values, then the result looks like the plot shown in Figure 1.1. The plot is "crude" because the polygonal effect is noticeable in regions where the function is changing rapidly. But otherwise the graph looks quite good. Our introduction to MATLAB begins with the details of the plotting process and the vector computations that go along with it. The $\sin(2\pi x)$ example is used throughout because it is simple and structured. Exploiting that structure leads naturally to some vector operations that are well supported in the MATLAB environment.

1.1.1 Setting Up Vectors

When you invoke the MATLAB system, you enter the *command window* and are prompted to enter commands with the symbol ">>". For example,

```
>> x = [10.1 20.2 30.3]
```

MATLAB is an interactive environment and it responds with

```
x =

    10.1000    20.2000    30.3000

>>
```

This establishes x as a length-3 row vector. Square brackets delineate the vector and spaces separate the components. On the other hand, the exchange

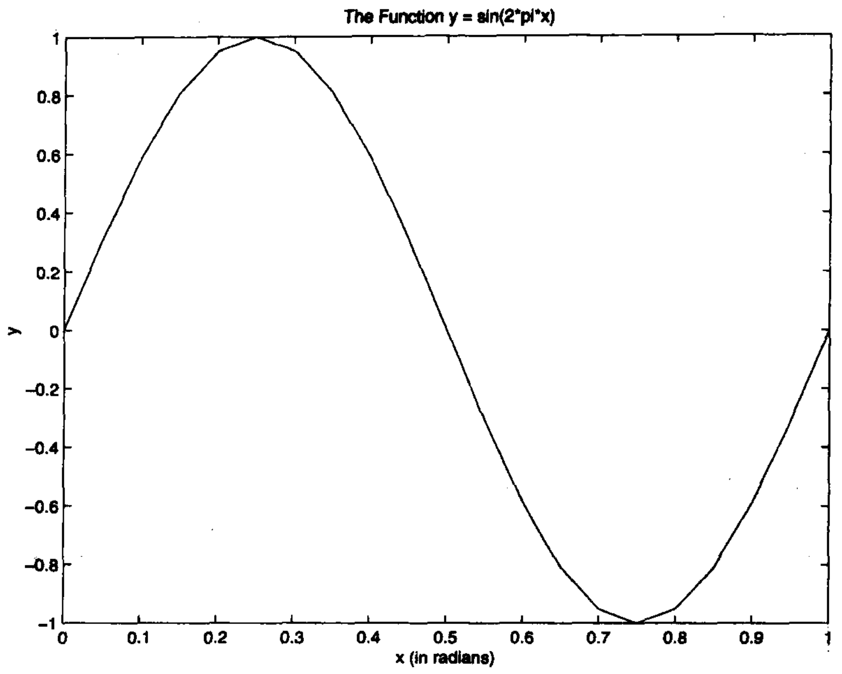


FIGURE 1.1 A crude plot of $\sin(2\pi x)$

```
>> x = [ 10.1; 20.2; 30.3]
x =
    10.1000
    20.2000
    30.3000
```

establishes x as a length-3 column vector. Again, square brackets define the vector being set up. But this time semicolons separate the component entries and a column vector is produced.

In general, MATLAB displays the consequence of a command unless it is terminated with a semicolon. Thus,

```
>> x = [ 10.1; 20.2; 30.3];
```

sets up the same column 3-vector as in the previous example, but there is no echo that displays the result. However, the dialog

```
x = [10.1; 20.2; 30.3];
x

x =
   10.1000
   20.2000
   30.3000
```

shows that the contents of a vector can be displayed merely by entering the name of the vector. Even if one component in a vector is changed with no terminating semicolon, MATLAB displays the whole vector:

```
x = [10.1; 20.2; 30.3];
x(2) = 21

x =

    10.1000
    21.0000
    30.3000
```

It is clear that when dealing with large vectors, a single forgotten semicolon can result in a deluge of displayed output.

To change the orientation of a vector from row to column or column to row, use an apostrophe. Thus,

```
x = [10.1 20.2 30.3]'
```

establishes `x` as a length-3 column vector. Placing an apostrophe after a vector effectively takes its transpose.

The plot shown in Figure 1.1 involves the equal spacing of $n = 21$ x-values across $[0, 1]$; that is

```
x = [0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 ...
           .55 .60 .65 .70 .75 .80 .85 .90 .95 1.0 ]
```

The ellipsis symbol "..." permits the entry of commands that occupy more than one line.

It is clear that for even modest values of n, we need other mechanisms for setting up vectors. Naturally enough, a `for`-loop can be used:

```
n = 21;
h = 1/(n-1);
for k=1:n
   x(k) = (k-1)*h;
end
```

This is a MATLAB *script*. It assigns the same length-21 vector to `x` as before and it brings up an important point.

> *In* MATLAB, *variables are not declared by the user but are created on a need-to-use basis by a memory manager. Moreover, from* MATLAB*'s point of view, every simple variable is a complex matrix indexed from unity.*

Scalars are 1-by-1 matrices. Vectors are "skinny" matrices with either one row or one column. We have much more to say about "genuine" matrices later. Our initial focus is on real vectors and scalars.

In the preceding script, **n**, **h**, **k**, and **x** are variables. It is instructive to trace how **x** "turns into" a vector during the execution of the **for**-loop. After one pass through the loop, **x** is a length-1 vector (i.e., a scalar). During the second pass, the reference **x(2)** prompts the memory manager to make **x** a 2-vector. During the third pass, the reference **x(3)** prompts the memory manager to make **x** a 3-vector. And so it goes until by the end of the loop, **x** has length 21. It is a convention in MATLAB that this kind of vector construction yields row vectors.

The MATLAB **zeros** function is handy for setting up the shape and size of a vector prior to a loop that assigns it values. Thus,

```
n = 21;
h = 1/(n-1);
x = zeros(1,n);
for k=1:n;\\
   x(k) = (k-1)*h;
end
```

computes **x** as row vector of length 21 and initializes the values to zero. It then proceeds to assign the appropriate value to each of the 21 components. Replacing **x = zeros(1,n)** with the command **x = zeros(n,1)** sets up a length 21 column vector. This style of vector set-up is recommended for two reasons. First, it forces you to think explicitly about the orientation and length of the vectors that you are working with. This reduces the chance for "dimension mismatch" errors when vectors are combined. Second, it is more efficient because the memory manager does not have to "work" so hard with each pass through the loop.

MATLAB supplies a **length** function that can be used to probe the length of any vector. To illustrate its use, the script

```
u = [10 20 30];
n = length(u);
v = [10;20;30;40];
m = length(v);
u = [50 60];
p = length(u);
```

assigns the values of 3, 4, and 2 to **n**, **m**, and **p**, respectively.

This brings up another important feature of MATLAB. It supports a very extensive **help** facility. For example, if we enter

```
help length
```

then MATLAB responds with

```
LENGTH Number of components of a vector.
 LENGTH(X) returns the length of vector X.  It is equivalent
 to MAX(SIZE(X)).
```

So extensive and well structured is the help facility that it obviates the need for us to go into excessive detail when discussing many of MATLAB's capabilities. Get in the habit of playing around with each new MATLAB feature that you learn, exploring the details via the **help** facility. Start right now by trying