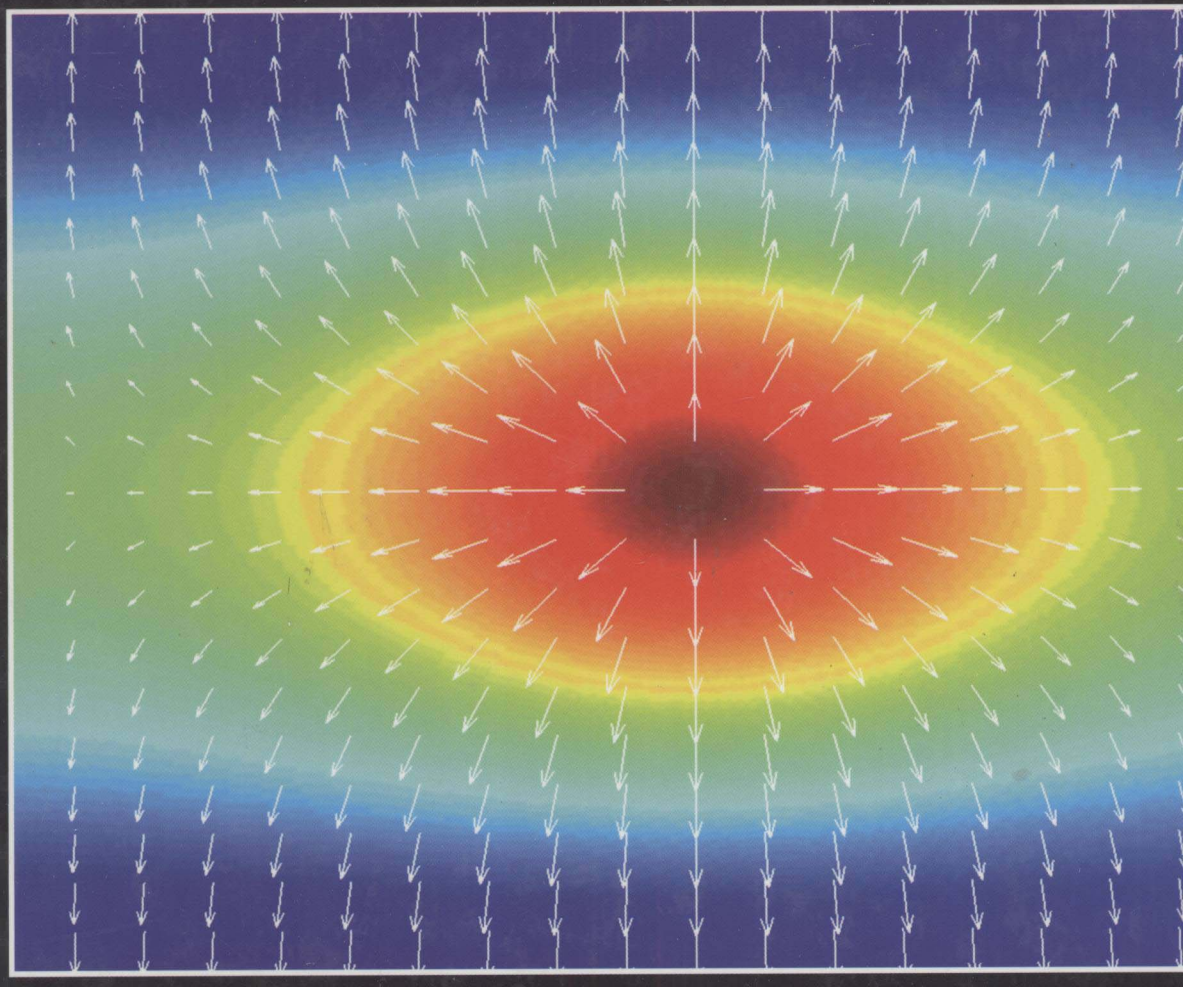
Numerical Methods for Chemical Engineering

Applications in MATLAB®



Kenneth J. Beers

CAMBRIDGE

Numerical Methods for Chemical Engineering

Applications in MATLAB®

KENNETH J. BEERS
Massachusetts Institute of Technology

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521859714

First published 2007

Printed in the United Kingdom at the University Press, Cambridge

A catalog record for this publication is available from the British Library

ISBN-13 978-0-521-85971-4 hardback
ISBN-10 0-521-85971-9 hardback

Numerical Methods for Chemical Engineering

Suitable for a first-year graduate course, this textbook unites the applications of numerical mathematics and scientific computing to the practice of chemical engineering. Written in a pedagogic style, the book describes basic linear and nonlinear algebraic systems all the way through to stochastic methods, Bayesian statistics, and parameter estimation. These subjects are developed at a nominal level of theoretical mathematics suitable for graduate engineers. The implementation of numerical methods in MATLAB® is integrated within each chapter and numerous examples in chemical engineering are provided, together with a library of corresponding MATLAB programs. Although the applications focus on chemical engineering, the treatment of the topics should also be of interest to non-chemical engineers and other applied scientists that work with scientific computing. This book will provide the graduate student with the essential tools required by industry and research alike.

Supplementary material includes solutions to homework problems set in the text, MATLAB programs and tutorial, lecture slides, and complicated derivations for the more advanced reader. These are available online at www.cambridge.org/9780521859714.

KENNETH J BEERS has been Assistant Professor at MIT since 2000. He has taught extensively across the engineering discipline at both the undergraduate and graduate level. This book is a result of the successful course the author devised at MIT for numerical methods applied to chemical engineering.

Preface

This text focuses on the application of quantitative analysis to the field of chemical engineering. Modern engineering practice is becoming increasingly more quantitative, as the use of scientific computing becomes ever more closely integrated into the daily activities of all engineers. It is no longer the domain of a small community of specialist practitioners. Whereas in the past, one had to hand-craft a program to solve a particular problem, carefully husbanding the limited memory and CPU cycles available, now we can very quickly solve far more complex problems using powerful, widely-available software. This has introduced the need for research engineers and scientists to become computationally literate – to know the possibilities that exist for applying computation to their problems, to understand the basic ideas behind the most important algorithms so as to make wise choices when selecting and tuning them, and to have the foundational knowledge necessary to navigate independently through the literature.

This text meets this need, and is written at the level of a first-year graduate student in chemical engineering, a consequence of its development for use at MIT for the course 10.34, "Numerical methods applied to chemical engineering." This course was added in 2001 to the graduate core curriculum to provide all first-year Masters and Ph.D. students with an overview of quantitative methods to augment the existing core courses in transport phenomena, thermodynamics, and chemical reaction engineering. Care has been taken to develop any necessary material specific to chemical engineering, so this text will prove useful to other engineering and scientific fields as well. The reader is assumed to have taken the traditional undergraduate classes in calculus and differential equations, and to have some experience in computer programming, although not necessarily in MATLAB®.

Even a cursory search of the holdings of most university libraries shows there to be a great number of texts with titles that are variations of "Advanced Engineering Mathematics" or "Numerical Methods." *So why add yet another*?

I find that there are two broad classes of texts in this area. The first focuses on introducing numerical methods, applied to science and engineering, at the level of a junior or senior undergraduate elective course. The scope is necessarily limited to rather simple techniques and applications. The second class is targeted to research-level workers, either higher graduate-level applied mathematicians or computationally-focused researchers in science and engineering. These may be either advanced treatments of numerical methods for mathematicians, or detailed discussions of scientific computing as applied to a specific subject such as fluid mechanics.

Neither of these classes of text is appropriate for teaching the fundamentals of scientific computing to beginning chemical engineering graduate students. Examples should be typical of those encountered in graduate-level chemical engineering research, and while the students should gain an understanding of the basis of each method and an appreciation of its limitations, they do not need exhaustive theory-proof treatments of convergence, error analysis, etc. It is a challenge for beginning students to identify how their own problems may be mapped into ones amenable to quantitative analysis; therefore, any appropriate text should have an extensive library of worked examples, with code available to serve later as templates. Finally, the text should address the important topics of model development and parameter estimation. This book has been developed with these needs in mind.

This text first presents a fundamental discussion of linear algebra, to provide the necessary foundation to read the applied mathematical literature and progress further on one's own. Next, a broad array of simulation techniques is presented to solve problems involving systems of nonlinear algebraic equations, initial value problems of ordinary differential and differential-algebraic (DAE) systems, optimizations, and boundary value problems of ordinary and partial differential equations. A treatment of matrix eigenvalue analysis is included, as it is fundamental to analyzing these simulation techniques.

Next follows a detailed discussion of probability theory, stochastic simulation, statistics, and parameter estimation. As engineering becomes more focused upon the molecular level, stochastic simulation techniques gain in importance. Particular attention is paid to Brownian dynamics, stochastic calculus, and Monte Carlo simulation. Statistics and parameter estimation are addressed from a Bayesian viewpoint, in which Monte Carlo simulation proves a powerful and general tool for making inferences and testing hypotheses from experimental data.

In each of these areas, topically relevant examples are given, along with MATLAB (www.mathworks.com) programs that serve the students as templates when later writing their own code. An accompanying website includes a MATLAB tutorial, code listings of all examples, and a supplemental material section containing further detailed proofs and optional topics. Of course, while significant effort has gone into testing and validating these programs, no guarantee is provided and the reader should use them with caution.

The problems are graded by difficulty and length in each chapter. Those of grade A are simple and can be done easily by hand or with minimal programming. Those of grade B require more programming but are still rather straightforward extensions or implementations of the ideas discussed in the text. Those of grade C either involve significant thinking beyond the content presented in the text or programming effort at a level beyond that typical of the examples and grade B problems.

The subjects covered are broad in scope, leading to the considerable (though hopefully not excessive) length of this text. The focus is upon providing a fundamental understanding of the underlying numerical algorithms without necessarily exhaustively treating all of their details, variations, and complexities of use. Mastery of the material in this text should enable first-year graduate students to perform original work in applying scientific computation to their research, and to read the literature to progress independently to the use of more sophisticated techniques.

Writing a book is a lengthy task, and one for which I have enjoyed much help and support. Professor William Green of MIT, with whom I taught this course for one semester, generously shared his opinions of an early draft. The teaching assistants who have worked on the course have also been a great source of feedback and help in problem-development, as have, of course, the students who have wrestled with intermediate drafts and my evolving approach to teaching the subject. My Ph.D. students Jungmee Kang, Kirill Titievskiy, Erik Allen, and Brian Stephenson have shown amazing forbearance and patience as the text became an additional, and sometimes demanding, member of the group. Above all, I must thank my family, and especially my supportive wife Jen, who have been tracking my progress and eagerly awaiting the completion of the book.

Contents

1 Linear algebra

This chapter discusses the solution of sets of linear algebraic equations and defines basic vector/matrix operations. The focus is upon elimination methods such as Gaussian elimination, and the related LU and Cholesky factorizations. Following a discussion of these methods, the existence and uniqueness of solutions are considered. Example applications include the modeling of a separation system and the solution of a fluid mechanics boundary value problem. The latter example introduces the need for sparse-matrix methods and the computational advantages of banded matrices. Because linear algebraic systems have, under well-defined conditions, a unique solution, they serve as fundamental building blocks in more-complex algorithms. Thus, linear systems are treated here at a high level of detail, as they will be used often throughout the remainder of the text.

Linear systems of algebraic equations

We wish to solve a system of N simultaneous linear algebraic equations for the N unknowns $x_1, x_2, \ldots, x_N$, that are expressed in the general form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N &= b_2 \\ &\vdots \\ a_{N1}x_1 + a_{N2}x_2 + \cdots + a_{NN}x_N &= b_N \end{aligned} \tag{1.1}$$

a_{ij} is the constant coefficient (assumed real) that multiplies the unknown x_j in equation i. b_i is the constant "right-hand-side" coefficient for equation i, also assumed real. As a particular example, consider the system

$$\begin{aligned} x_1 + x_2 + x_3 &= 4 \\ 2x_1 + x_2 + 3x_3 &= 7 \\ 3x_1 + x_2 + 6x_3 &= 2 \end{aligned} \tag{1.2}$$

for which

$$\begin{aligned} a_{11} &= 1 \quad a_{12} = 1 \quad a_{13} = 1 \quad b_1 = 4 \\ a_{21} &= 2 \quad a_{22} = 1 \quad a_{23} = 3 \quad b_2 = 7 \\ a_{31} &= 3 \quad a_{32} = 1 \quad a_{33} = 6 \quad b_3 = 2 \end{aligned} \tag{1.3}$$

It is common to write linear systems in *matrix/vector form* as

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1.4}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{1.5}$$

Row i of A contains the values $a_{i1}, a_{i2}, \dots, a_{iN}$ that are the coefficients multiplying each unknown $x_1, x_2, \dots, x_N$ in equation i. Column j contains the coefficients $a_{1j}, a_{2j}, \dots, a_{Nj}$ that multiply x_j in each equation $i = 1, 2, \dots, N$. Thus, we have the following associations,

$$\text{rows} \Leftrightarrow \text{equations} \qquad \text{columns} \Leftrightarrow \begin{array}{c}\text{coefficients multiplying} \\ \text{a specific unknown} \\ \text{in each equation}\end{array}$$

We often write $A\boldsymbol{x} = \boldsymbol{b}$ explicitly as

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{1.6}$$

For the example system (1.2),

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} 4 \\ 7 \\ 2 \end{bmatrix} \tag{1.7}$$

In MATLAB we solve $A\boldsymbol{x} = \boldsymbol{b}$ with the single command, **x = A\b**. For the example (1.2), we compute the solution with the code

```
A = [1 1 1; 2 1 3; 3 1 6];
b = [4; 7; 2];
x = A\b,
x =
    19.0000
    -7.0000
    -8.0000
```

Thus, we are tempted to assume that, as a practical matter, we need to know little about how to solve a linear system, as someone else has figured it out and provided us with this handy linear solver. Actually, we shall need to understand the fundamental properties of linear systems in depth to be able to master methods for solving more complex problems, such as sets of nonlinear algebraic equations, ordinary and partial

differential equations, etc. Also, as we shall see, this solver fails for certain common classes of very large systems of equations, and we need to know enough about linear algebra to diagnose such situations and to propose other methods that do work in such instances. This chapter therefore contains not only an explanation of how the MATLAB solver is implemented, but also a detailed, fundamental discussion of the properties of linear systems.

Our discussion is intended only to provide a foundation in linear algebra for the practice of numerical computing, and is continued in Chapter 3 with a discussion of matrix eigenvalue analysis. For a broader, more detailed, study of linear algebra, consult Strang (2003) or Golub & van Loan (1996).

Review of scalar, vector, and matrix operations

As we use vector notation in our discussion of linear systems, a basic review of the concepts of vectors and matrices is necessary.

Scalars, real and complex

Most often in basic mathematics, we work with *scalars*, i.e., single-valued numbers. These may be real, such as 3, 1.4, 5/7, 3.14159 . . . . , or they may be complex, $1 + 2i$, $1/2\,i$, where $i = \sqrt{-1}$. The *set of all real scalars* is denoted $\Re$. The *set of all complex scalars* we call C. For a complex number $z \in C$, we write $z = a + ib$, where $a, b \in \Re$ and

$$\begin{aligned} a &= Re\{z\} = \text{real part of } z \\ b &= \text{Im}\{z\} = \text{imaginary part of } z \end{aligned} \tag{1.8}$$

The *complex conjugate*, $\bar{z} = z^*$, of $z = a + ib$ is

$$\bar{z} = z^* = a - ib \tag{1.9}$$

Note that the product $\bar{z}z$ is always real and nonnegative,

$$\bar{z}z = (a - ib)(a + ib) = a^2 - iab + iab - i^2 b^2 = a^2 + b^2 \geq 0 \tag{1.10}$$

so that we may define the real-valued, nonnegative *modulus* of z, $|z|$, as

$$|z| = \sqrt{\bar{z}z} = \sqrt{a^2 + b^2} \geq 0 \tag{1.11}$$

Often, we write complex numbers in polar notation,

$$z = a + ib = |z|(\cos\theta + i\sin\theta) \qquad \theta = \tan^{-1}(b/a) \tag{1.12}$$

Using the important *Euler formula*, a proof of which is found in the supplemental material found at the website that accompanies this book,

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{1.13}$$

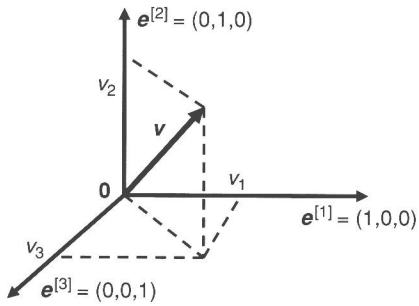


Figure 1.1 Physical interpretation of a 3-D vector.

we can write z as

$$z = |z|e^{i\theta} \tag{1.14}$$

Vector notation and operations

We write a three-dimensional (3-D) vector $\boldsymbol{v}$ (Figure 1.1) as

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{1.15}$$

$\boldsymbol{v}$ is *real* if $v_1, v_2, v_3 \in \Re$; we then say $\boldsymbol{v} \in \Re^3$. We can easily visualize this vector in 3-D space, defining the three coordinate basis vectors in the $1(x)$, $2(y)$, and $3(z)$ directions as

$$\boldsymbol{e}^{[1]} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{e}^{[2]} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \boldsymbol{e}^{[3]} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1.16}$$

to write $\boldsymbol{v} \in \Re^3$ as

$$\boldsymbol{v} = v_1\boldsymbol{e}^{[1]} + v_2\boldsymbol{e}^{[2]} + v_3\boldsymbol{e}^{[3]} \tag{1.17}$$

We extend this notation to define $\Re^N$, the *set of N-dimensional real vectors*,

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \tag{1.18}$$

where $v_j \in \Re$ for $j = 1, 2, \ldots, N$. By writing $\boldsymbol{v}$ in this manner, we define a *column vector*; however, $\boldsymbol{v}$ can also be written as a *row vector*,

$$\boldsymbol{v} = [v_1 \quad v_2 \ldots \quad v_N] \tag{1.19}$$

The difference between column and row vectors only becomes significant when we start combining them in equations with matrices.

We write $\boldsymbol{v} \in \Re^N$ as an expansion in coordinate basis vectors as

$$\boldsymbol{v} = v_1 \boldsymbol{e}^{[1]} + v_2 \boldsymbol{e}^{[2]} + \cdots + v_N \boldsymbol{e}^{[N]} \tag{1.20}$$

where the components of $\boldsymbol{e}^{[j]}$ are *Kroenecker deltas* δ_{jk},

$$\boldsymbol{e}^{[j]} = \begin{bmatrix} e_1^{[j]} \\ e_2^{[j]} \\ \vdots \\ e_N^{[j]} \end{bmatrix} = \begin{bmatrix} \delta_{j1} \\ \delta_{j2} \\ \vdots \\ \delta_{jN} \end{bmatrix} \qquad \delta_{jk} = \begin{cases} 1, & \text{if } j = k \\ 0, & \text{if } j \neq k \end{cases} \tag{1.21}$$

Addition of two real vectors $\boldsymbol{v} \in \Re^N$, $\boldsymbol{w} \in \Re^N$ is straightforward,

$$\boldsymbol{v} + \boldsymbol{w} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \\ \vdots \\ v_N + w_N \end{bmatrix} \tag{1.22}$$

as is multiplication of a vector $\boldsymbol{v} \in \Re^N$ by a real scalar $c \in \Re$,

$$c\boldsymbol{v} = c \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} cv_1 \\ cv_2 \\ \vdots \\ cv_N \end{bmatrix} \tag{1.23}$$

For all $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w} \in \Re^N$ and all $c_1, c_2 \in \Re$,

$$\begin{aligned} \boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) &= (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w} & c(\boldsymbol{v} + \boldsymbol{u}) &= c\boldsymbol{v} + c\boldsymbol{u} \\ \boldsymbol{u} + \boldsymbol{v} &= \boldsymbol{v} + \boldsymbol{u} & (c_1 + c_2)\boldsymbol{v} &= c_1\boldsymbol{v} + c_2\boldsymbol{v} \\ \boldsymbol{v} + \boldsymbol{0} &= \boldsymbol{v} & (c_1 c_2)\boldsymbol{v} &= c_1(c_2\boldsymbol{v}) \\ \boldsymbol{v} + (-\boldsymbol{v}) &= \boldsymbol{0} & 1\boldsymbol{v} &= \boldsymbol{v} \end{aligned} \tag{1.24}$$

where the *null vector* $\boldsymbol{0} \in \Re^N$ is

$$\boldsymbol{0} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1.25}$$

We further add to the list of operations associated with the vectors $\boldsymbol{v}$, $\boldsymbol{w} \in \Re^N$ the *dot (inner, scalar) product*,

$$\boldsymbol{v} \cdot \boldsymbol{w} = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N = \sum_{k=1}^{N} v_k w_k \tag{1.26}$$

For example, for the two vectors

$$\boldsymbol{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \tag{1.27}$$

$$\begin{aligned} \boldsymbol{v} \cdot \boldsymbol{w} = v_1 w_1 + v_2 w_2 + v_3 w_3 &= (1)(4) + (2)(5) + (3)(6) \\ &= 4 + 10 + 18 = 32 \end{aligned} \tag{1.28}$$

For 3-D vectors, the dot product is proportional to the product of the lengths and the cosine of the angle between the two vectors,

$$\boldsymbol{v} \cdot \boldsymbol{w} = |\boldsymbol{v}||\boldsymbol{w}| \cos\theta \tag{1.29}$$

where the *length* of $\boldsymbol{v}$ is

$$|\boldsymbol{v}| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}} \geq 0 \tag{1.30}$$

Therefore, when two vectors are parallel, the magnitude of their dot product is maximal and equals the product of their lengths, and when two vectors are perpendicular, their dot product is zero. These ideas carry completely into N- dimensions. The *length* of a vector $\boldsymbol{v} \in \Re^N$ is

$$|\boldsymbol{v}| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}} = \sqrt{\sum_{k=1}^{N} v_k^2} \geq 0 \tag{1.31}$$

If $\boldsymbol{v} \cdot \boldsymbol{w} = 0$, $\boldsymbol{v}$ and $\boldsymbol{w}$ are said to be *orthogonal*, the extension of the adjective "perpendicular" from $\Re^3$ to $\Re^N$. If $\boldsymbol{v} \cdot \boldsymbol{w} = 0$ and $|\boldsymbol{v}| = |\boldsymbol{w}| = 1$, i.e., both vectors are *normalized* to unit length, $\boldsymbol{v}$ and $\boldsymbol{w}$ are said to be *orthonormal*.

The formula for the length $|\boldsymbol{v}|$ of a vector $\boldsymbol{v} \in \Re^N$ satisfies the more general properties of a *norm* $\|\boldsymbol{v}\|$ of $\boldsymbol{v} \in \Re^N$. A norm $\|\boldsymbol{v}\|$ is a rule that assigns a real scalar, $\|\boldsymbol{v}\| \in \Re$, to each vector $\boldsymbol{v} \in \Re^N$ such that for every $\boldsymbol{v}, \boldsymbol{w} \in \Re^N$, and for every $c \in \Re$, we have

$$\begin{gathered} \|\boldsymbol{v}\| \geq 0 \qquad \|\boldsymbol{0}\| = 0 \\ \|\boldsymbol{v}\| = 0 \qquad \text{if and only if (iff)} \qquad \boldsymbol{v} = \boldsymbol{0} \\ \|c\boldsymbol{v}\| = |c|\|\boldsymbol{v}\| \\ \|\boldsymbol{v} + \boldsymbol{w}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{w}\| \end{gathered} \tag{1.32}$$

Each norm also provides an accompanying *metric*, a measure of how different two vectors are

$$d(\boldsymbol{v}, \boldsymbol{w}) = \|\boldsymbol{v} - \boldsymbol{w}\| \tag{1.33}$$

In addition to the length, many other possible definitions of norm exist. The *p-norm*, $\|\boldsymbol{v}\|_p$, of $\boldsymbol{v} \in \Re^N$ is

$$\|\boldsymbol{v}\|_p = \left[\sum_{k=1}^{N} |v_k|^p \right]^{1/p} \tag{1.34}$$