经典原版书库

数值分析

（英文版·第3版）

Numerical Analysis:

Mathematics of Scientific Computing

Third Edition

David Kincaid
Ward Cheney

THE BROOKS/COLE SERIES IN ADVANCED MATHEMATICS
Paul J. Sally, Jr., EDITOR

（美）David Kincaid Ward Cheney 著
得克萨斯大学，奥斯汀

机械工业出版社
China Machine Press

经 典 原 版 书 库

数值分析

(英文版·第3版)

Numerical Analysis

Mathematics of Scientific Computing

(Third Edition)

(美) David Kincaid Ward Cheney 著
得克萨斯大学，奥斯汀

机械工业出版社
China Machine Press

David Kincaid and Ward Cheney: Numerical Analysis: Mathematics of Scientific Computing, Third Edition(ISBN: 0-534-38905-8).

本书版权登记号：图字：01-2003-1015

图书在版编目（CIP）数据

数值分析（英文版·第3版）/（美）金凯德（Kincaid, D.），（美）切尼（Cheney, W.）著. –北京：机械工业出版社，2003.4

（经典原版书库）

书名原文：Numerical Analysis: Mathematics of Scientific Computing, Third Edition

ISBN 7-111-11913-4

Ⅰ. 数… Ⅱ. ①金… ②切… Ⅲ. 数值计算－高等学校－教材－英文 Ⅳ. O241

中国版本图书馆CIP数据核字（2003）第023795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：杨海玲

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2003年4月第1版第1次印刷

787mm × 1092mm 1/16 · 50.25印张

印数：0 001-2 000册

定价：75.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

Preface

Dedicated to our parents Sarah and Robert B. Kincaid,
and (in memoriam) Carleton and Elliott Cheney.

This book has evolved over many years from lecture notes that accompany certain upper-division and graduate courses in mathematics and computer sciences at The University of Texas at Austin. These courses introduce students to the algorithms and methods that are commonly needed in scientific computing. The mathematical underpinnings of these methods are emphasized as much as their algorithmic aspects. The students have been diverse: mathematics, engineering, science, and computer science undergraduates, as well as graduate students from various disciplines. Portions of the book also have been used to lay the groundwork in several graduate courses devoted to special topics in numerical analysis, such as the numerical solution of differential equations, numerical linear algebra, and approximation theory. Our approach has always been to treat the subject from a mathematical point of view, with attention given to its rich offering of theorems, proofs, and interesting ideas. From these arise many computational procedures and intriguing questions of computer science. Of course, our motivation comes from the practical world of scientific computing, which dictates the choice of topics and the manner of treating each. For example, with some topics it is more instructive to discuss the theoretical foundations of the subject and not attempt to analyze algorithms in detail. In other cases, the reverse is true, and the students learn much from programming simple algorithms themselves and experimenting with them—although we offer a blanket admonition to use well-tested software, such as from program libraries, on problems that arise from applications.

There is some overlap between this book and our more elementary text, *Numerical Mathematics and Computing, Fourth Edition* (Brooks/Cole). That book is addressed to students having more modest mathematical preparation (and sometimes less enthusiasm for the theoretical side of the subject). In that text, there is a different menu of topics, and no topic is pursued to any great depth. The present book, on the other hand, is intended for a course that offers a more scholarly treatment of the subject; many topics are dealt with at length. Occasionally we broach topics that heretofore have not found their way into standard textbooks at this level. In this category are the multigrid method, procedures for multivariate interpolation, homotopy (or continuation) methods, delay differential equations, and optimization.

The algorithms in the book are presented in a *pseudocode* that contains additional details beyond the mathematical formulas. The reader can easily write computer routines based on the pseudocode in any standard computer language. We believe that students learn and understand numerical methods best by seeing how algorithms are developed from the mathematical theory and then writing and testing computer implementations of them. Of course, such computer programs do not contain all the complicated procedures and sophisticated checks found in robust routines available in scientific libraries. Examples of general-purpose mathematical libraries are found in the appendix on *An Overview of Mathematical Software*. For most applications, such libraries are strongly preferred to code written oneself.

An important constituent of the book (and essential to its pedagogic purpose) is the abundance of problems for the student. These are of two types: analytic problems and computer problems. The computer problems are, in turn, of two types: those in which students write their own code, and those in which they employ existing software. We believe that both kinds of programming practice are necessary. Using someone else's software is not always a trivial exercise, even when it is as well documented as with large program libraries or packages. On the other hand, students usually acquire much more insight into an algorithm after coding and testing it themselves, rather than simply using a software package. In most cases the computer problems require access to a computer that has at least a 32-bit word length.

Software, errata, and teaching aids are available via the Internet as discussed in the appendix. Also, the publisher has made available a Solution Manual for instructors who adopt the book for their classes.

The third edition contains new problems, re-ordering of some problems, and corrections to all known errors in the previous edition. Updating of the information about resources on the Internet has been done in the appendix on mathematical software. Also, the bibliography has been updated. Many references to problems and to other parts of the book are now given with page numbers to help the reader easily find them. Also, most theorems are displayed with names or titles to help the reader remember them. The entire book has a new design style and it has been reformatted for improved appearance. Many improvements have been made throughout. For example, in this new edition, we have added a chapter on optimization with subtopics on methods of descent, quadratic fitting algorithms, Nelder-Meade algorithm, simulated annealing, genetic algorithms, Pareto optimization, and convex programming. A standard course of one semester can be based on selected sections from Chapters 1–4 and 6–8. A two-semester course could cover selected sections in Chapters 1–9 plus other topics of interest. Chapters 4 and 5 could be taught independently from the previous chapters as a short course on numerical linear algebra. Because of the ambitious scope of this book, some sections make greater demands on the preparation of the reader. These sections usually occur late in any given chapter so that the reader is not unduly challenged at the start, and they may be skipped at the reader's discretion. Such sections are marked with an asterisk. Page numbers are included with references to problems, computer problems, and items such as theorems and equations outside the section being read. Unless it says otherwise, references to equations, theorems, lemmas, corollaries, etc. are assumed to be in the current section and page numbers are not included.

Acknowledgments

We are glad to express our indebtedness to many persons who have assisted us in the writing of this book.

First Edition

Administrative support was provided by Sheri Brice, Margaret Combs, Jan Duffy, Katherine Mueller, and Jenny Tsao of The University of Texas at Austin. Foremost among these is Margaret Combs of the Mathematics Department, who rendered into TeX innumerable versions of each section, patiently preparing new ones as they were needed for classroom notes in successive years. No technical problem of typesetting was too difficult for her as she mastered the arcane art of dissecting and reassembling TeX macros to serve unusual needs. It is appropriate at this point that we also express our public thanks to Donald Knuth for his magnificent contribution to the scientific community embodied in the TeX typesetting system. We appreciate the suggestions made by the following astute reviewers of preliminary versions of the manuscript: Thomas A. Atchison, Frederick J. Carter, Philip Crooke, Jim D'Archangelo, R. S. Falk, J. R. Hubbard, Patrick Lang, Giles Wilson Maloof, A. K. Rigler, F. Schumann, A. J. Worsey, and Charles Votaw. In addition, thanks are due the following persons for technical help and critical reading of the manuscript: Victoria Hunter, Carole Kincaid, Tad Liszka, Rio Hirowati Shariffudin, and Laurette Tuckerman. David Young was always generous with suggestions and advice. A number of advanced students who served as teaching assistants in our classes also helped; in particular, we thank David Bruce, Nai-ching Chen, Ashok Hattangady, Ru Huang, Wayne Joubert, Irina Mukherjee, Bill Nanry, Tom Oppe, Marcos Raydan, Malathi Ramdas, John Respess, Phien Tran, Linda Tweedy, and Baba Vemuri. The editors and technical staff at Brooks/Cole Publishing Company have been most cooperative and supportive during this project. In particular, we are pleased to thank Jeremy Hayhurst and Marlene Thom for their assistance. Stacey Sawyer of Sawyer and Williams was responsible for a careful copyediting of the manuscript, and Ralph Youngen of the American Mathematical Society provided technical assistance and supervision in turning our TeX files into final printed copy.

Second Edition

We would like to acknowledge the reviewers for their work: Dan Boley, University of Minnesota; Min Chen, Pennsylvania State University; John Harper, University of Rochester; Ramon Moore, Ohio State University; Yves Nievergelt, Eastern Washington University; and Elinor Velasquez, University of California-Berkeley. We especially thank Ron Boisvert for clarifying our understanding of the different categories of mathematical software and for the examples given in the appendix. Also, we want to thank those who took the trouble to contact us with suggestions and corrections in the first edition. Some of these are Victor M. Afram, Roger Alexander, A. Awwal, Carl de Boor, T. P. Brown, James Caveny, George J. Davis, Hakan Ekblom, Mariano Gasca, Bill Gearhart, Patrick Goetz, Gary L. Gray, Bob Gregorac, Katherine Hua Guo, Cecilia Jea, Liz Jessup, Grant Keady, Baker Kearfott, Junjiang

Lei, Teck C. Lim, Julio Lopez, C. Lu, Taketomo Mitsui, Irina Mukherjee, Teresa Perez, Robert Piche, Sherman Riemenschneider, Maria Teresa Rodriquez, Ulf Roennow, Larry Schumaker, Wei-Chang Shann, Christopher J. van Wyk, Kang Zhao, and Mark Zhou.

Third Edition

We wish to thank those who have contacted us with suggestions and corrections in the second edition: Eyal Arian, Carl de Boor, Yung-Ming Chang, Antonella Cupillari Paul Eenigenburg, Leopoldo P. Franca, Henry Greenside, R. J. Gregorac, Scott A. King, Robert Piche, N. Shamsundar, Topi Urponen, and Yuan Xu. Also, we are grateful to those who have been helpful in putting out the new edition: Patrick Goetz, Shashank Khandelwal, and especially Durene Ngo.

We are grateful to the Center for Numerical Analysis, the Texas Institute of Computational and Applied Mathematics, and the Computer Sciences Department and the Mathematics Department of The University of Texas at Austin for technical support and for furnishing excellent computing facilities.

Finally, the editors and technical staff at Brooks/Cole–Thomson Learning have been most helpful and supportive during the revision of this book, particularly Bob Pirtle, Janet Hill, and Molly Nance. We also thank Don DeLand, Leslie Galen, Joe Albrecht, and others at Integre Technical Publishing Company for their fine work.

Software supporting this textbook is available online at the Web sites listed in the Appendix (p. 737).

We would appreciate any comments, suggestions, questions, criticisms, or corrections that readers may take the trouble of communicating to us.

David Kincaid
Ward Cheney

Contents

Numerical Analysis: What Is It?

Numerical analysis involves the *study*, *development*, and *analysis* of algorithms for obtaining numerical solutions to various mathematical problems. Frequently, numerical analysis is called the *mathematics of scientific computing*.

The algorithms that we study invariably are destined for use on high-speed computers, and therefore another crucial step intervenes before the solution to a problem can be obtained: a computer *program* or *code* must be written to communicate the algorithm to the computer. This is, of course, a nontrivial matter, but there are so many choices of computers and computer languages that it is a topic best left out of the science of numerical analysis per se.

There are certainly many other purposes to which computers can be put besides the numerical solution of mathematical problems: providing basic communications, keeping large data bases, playing games, "net surfing," writing novels, accounting, and so on. Solving *mathematical* problems numerically on the computer is *scientific computing*. The development of the associated algorithms (procedures) and the study of their behavior are the mathematics of scientific computing.

Often the development of an algorithm is stimulated by a *constructive* proof in mathematics. In classical analysis, nonconstructive methods are frequently used, but generally they do not lead to algorithms. For example, existence and uniqueness theorems might be established by assuming that they are *not* true and then following the trail of a logical argument until arriving at a contradiction. Not every constructive proof will lead to a successful algorithm, however. A difficulty that may arise is that an *analytical* solution to a given problem may be several steps away from a *numerical* solution. Or it might be completely impractical because of slow convergence or the need for lengthy computation.

As an example of the gap between an existence theorem and a numerical solution of a problem, consider the ubiquitous matrix equation $Ax = b$. We know that it has a unique solution whenever A is nonsingular. But this fact may be of little solace when we are faced with a large linear system containing empirical data and we wish to compute an *approximate* numerical solution.

In general, in this book, we will begin each topic with a basic mathematical problem that arises frequently in practical applications. Then a certain amount of analysis will be presented in order to arrive at an algorithm for solving the problem. Algorithms are usually given in the form of a pseudocode. Finally, additional analysis of the algorithm may be given to help in understanding its behavior, such as its convergence or its resistance to corruption by roundoff error. Such analysis may take the form of either *forward* or *backward* error analysis.

Behind each basic mathematical problem to be considered there are always physical applications. Let us illustrate all this with a heat-flow problem. The temperature in a solid piece of metal with various boundary conditions is governed by mathematical equations that must be satisfied at every point and at every instant of time. The principal equation here might be the *heat equation*

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}$$

It is a parabolic, linear, second-order, partial differential equation. It models the heat flow inside a rod under certain assumptions on the actual physical problem. The variable x is the space coordinate, and t is the time. The temperature is $u = u(x, t)$. To solve the model problem on the computer, the space-time region is *discretized* by a mesh of grid points, and the numerical solution is sought at each of these points. The partial derivatives in the heat equation can be approximated by finite differences such as

$$\frac{\partial v(x,t)}{\partial t} \approx \frac{1}{k}[v(x,t+k) - v(x,t)]$$

$$\frac{\partial^2 v(x,t)}{\partial x^2} \approx \frac{1}{h^2}[v(x+h,t) - 2v(x,t) + v(x-h,t)]$$

Here, k and h are the mesh spacings in the t-direction and the x-direction, respectively. Also, we have changed to the variable v to emphasize that we are solving an approximation to the model problem rather than the original problem. Replacing the partial derivatives by these approximations and simplifying, we arrive at a linear equation at each grid point (x_i, t_j). Using the abbreviation v_{ij} for $v(x_i, t_j)$, we obtain

$$v_{i,j+1} = s v_{i-1,j} + (1 - 2s)v_{ij} + s v_{i+1,j}$$

where $s = k/h^2$. The numerical solution can be advanced step by step in the t-direction using the preceding equation. This procedure is called an *explicit* method because the new values $v_{i,j+1}$ are explicitly determined one at a time from the previous values $v_{i-1,j}, v_{i,j}, v_{i+1,j}$. This is all very elegant, and one would not anticipate any difficulties. But the *analysis* as well as *numerical experience* indicate that the method is seriously flawed! We turn then to an *implicit* method. In it, all of the new values are determined at the same time by solving a linear system of the special form

$$V_{j+1} = AV_j$$

Here A is a certain tridiagonal matrix and $V_j = [v_{1j}, v_{2j}, \ldots, v_{nj}]^T$. Each of these methods requires a stability analysis to determine the permissible range of values for the mesh sizes h and k and the associated convergence behavior. It is here that the explicit method competes poorly. Complete details can be found in Chapter 9 (p. 615).

Chapter One

Mathematical Preliminaries

1.0 Introduction

This chapter starts with a review of some important topics in calculus that are required in the subsequent chapters. We encourage readers to skip boldly over material that is already familiar to them. In fact, some may wish to begin with Chapter 2, p. 37.

1.1 Basic Concepts and Taylor's Theorem

We begin with a review of some basic concepts from calculus. At this point, one might ask: *Why do we need to discuss such topics if we are primarily interested in scientific computing and numerical algorithms*? A solid background in basic mathematical concepts is essential to understanding the derivation of most numerical algorithms. Taylor's Theorem in various forms is fundamental to many numerical procedures and is an excellent starting point for the study of scientific computing since no advanced mathematical concepts are required.

Limit, Continuity, and Derivative

If f is a real-valued function of a real variable, then the **limit** of the function f at c (if it exists) is defined as follows: The equation

$$\lim_{x \to c} f(x) = L$$

means that to each positive ε there corresponds a positive δ such that the distance between $f(x)$ and L is less than ε whenever the distance between x and c is less than δ; that is,

$$|f(x) - L| < \varepsilon \quad \text{whenever} \quad 0 < |x - c| < \delta$$

If there is no number L with this property, the limit of f at c does not exist.

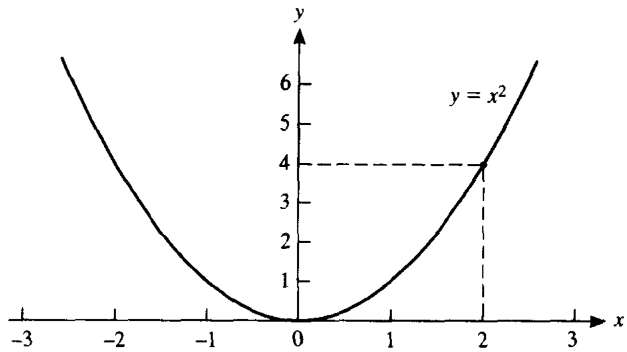

FIGURE 1.1
$y = f(x) = x^2$

For example, we consider the function

$$f(x) = x^2$$

From the graph of $f(x) = x^2$ in Figure 1.1, it appears that as x approaches 2, $f(x)$ approaches 4, so the equation

$$\lim_{x \to 2} x^2 = 4$$

should be true. We now see why this is so by proving that for $\varepsilon > 0$ there is a $\delta > 0$ such that $|x^2 - 4| < \varepsilon$ whenever $0 < |x-2| < \delta$. Let $\varepsilon > 0$ and $\delta = -2 + \sqrt{4+\varepsilon} > 0$ so that $\delta(\delta+4) = \varepsilon$. If $0 < |x-2| < \delta$, then $|x+2| = |x-2+4| \le |x-2| + 4 < \delta + 4$. Thus, we have $|x^2 - 4| = |x+2||x-2| < (\delta+4)\delta = \varepsilon$. Notice that we worked backwards in a sense to discover what values of δ would give us exactly ε. Clearly, other values of δ work as well, such as $\delta = \varepsilon/(5+\varepsilon)$. (See Problem **1.1.1**, p. 12.)

As another example, we consider

$$g(x) = \frac{|x|}{x} = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases}$$

See Figure 1.2 for the graph of the function $g(x) = |x|/x$, from which it is clear why $g(x)$ is undefined at 0. We note that the equation

$$\lim_{x \to 0} \frac{|x|}{x} = L$$

is not true for *any* number L. Indeed, let $\varepsilon = 1$, and suppose that $||x|/x - L| < 1$ whenever $0 < |x| < \delta$. If $x = \frac{1}{2}\delta$, then $0 < |x| < \delta$ and we have $|x|/x = 1$. But if $x = -\frac{1}{2}\delta$, then $0 < |x| < \delta$ and $|x|/x = -1$. In both cases, we must have $||x|/x - L| < 1$. There is no number L satisfying both $|1 - L| < 1$ and $|-1 - L| < 1$, since this would require both $0 < L < 2$ and $-2 < L < 0$. Clearly, this is an impossible situation! So we say that the limit does *not* exist.

If f is defined only on a specified subset X of the real line, the definition of limit is modified so that $|f(x) - L| < \varepsilon$ whenever $x \in X$ and $0 < |x - c| < \delta$.

The function f is said to be **continuous** at c if

$$\lim_{x \to c} f(x) = f(c)$$

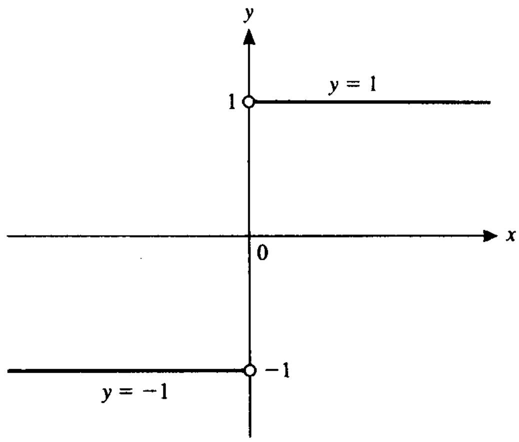

FIGURE 1.2
$y = g(x) = |x|/x$

Thus, the function $f(x) = x^2$ is continuous at the point 2, whereas the function $|x|/x$ is not continuous at 0, no matter how it is defined at 0. These assertions follow from remarks made previously.

An intuitively obvious theorem is given next.

THEOREM 1 **Intermediate-Value Theorem for Continuous Functions**

On an interval $[a, b]$, a continuous function assumes all values between $f(a)$ and $f(b)$.

The **derivative** of f at c (if it exists) is defined by the equation

$$f'(c) = \lim_{x \to c} \frac{f(x) - f(c)}{x - c}$$

Since this limit need not exist for a particular function and a particular c, it is possible for the derivative not to exist for such a function. If f is a function for which $f'(c)$ exists, we say that f is **differentiable** at c. If f is differentiable at c, then f must be continuous at c. We now see why this is so. Consider

$$\lim_{x \to c}[f(x) - f(c)] = \lim_{x \to c} \frac{f(x) - f(c)}{x - c}(x - c)$$
$$= f'(c) \cdot \lim_{x \to c}(x - c) = f'(c) \cdot 0 = 0$$

Clearly, if $f(x)$ is differentiable at c, then $f'(x)$ exists and $\lim_{x \to c} f(x) = f(c)$.

But the converse is not true! For example, if

$$f(x) = |x|$$

then $f'(0)$ does not exist. See the graph of $f(x) = |x|$ in Figure 1.3. The derivative at a point x is the tangent line to the curve at $f(x)$. But at the bottom of the "V" in the curve ($x = 0$) there is no unique tangent and no derivative at $x = 0$.

The set of all functions that are continuous on the entire real line $\mathbb{R}$ is denoted by $C(\mathbb{R})$. The set of functions for which f' is continuous everywhere is denoted by $C^1(\mathbb{R})$. If $f \in C^1(\mathbb{R})$, then f' is continuous at all points in $\mathbb{R}$ and, thereby,

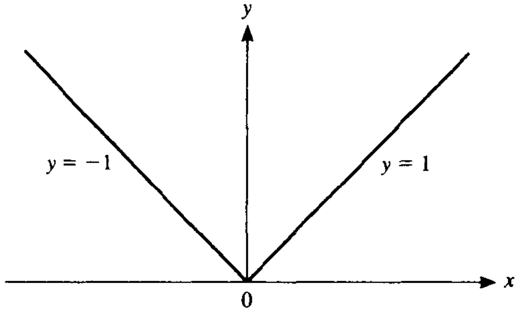

FIGURE 1.3
$y = f(x) = |x|$

differentiable throughout the real line. Since the differentiability of a function at a point implies its continuity at that point, we have $C^1(\mathbb{R}) \subset C(\mathbb{R})$. The set $C^1(\mathbb{R})$ is a *proper* subset of $C(\mathbb{R})$ because there are (many) continuous functions whose derivatives do not exist. The function $f(x) = |x|$ is such an example.

We denote by $C^2(\mathbb{R})$ the set of all functions for which f'' is continuous everywhere. By reasoning similar to that given above,

$$C^2(\mathbb{R}) \subset C^1(\mathbb{R}) \subset C(\mathbb{R})$$

Again, these are proper inclusions since there are functions that are once differentiable but not twice, such as $f(x) = x^2 \sin(1/x)$. (See Problem **1.1.3**, p. 12.)

Similarly, we define $C^n(\mathbb{R})$, for each natural number n, to be the set of all functions for which $f^{(n)}(x)$ is continuous. Finally, $C^\infty(\mathbb{R})$ is the set of functions each of whose derivatives is continuous. We have now

$$C^\infty(\mathbb{R}) \subset \cdots \subset C^2(\mathbb{R}) \subset C^1(\mathbb{R}) \subset C(\mathbb{R})$$

A familiar function in $C^\infty(\mathbb{R})$ is $f(x) = e^x$.

In the same way, we define $C^n[a, b]$ to be the set of functions f for which $f^{(n)}$ exists and is continuous on the closed interval $[a, b]$.

Taylor's Theorem

An important theorem concerning functions in $C^n[a, b]$ is Taylor's Theorem, which arises throughout the study of numerical analysis or in the study of numerical algorithms in scientific computing.

THEOREM 2 Taylor's Theorem with Lagrange Remainder

If $f \in C^n[a, b]$ and if $f^{(n+1)}$ exists on the open interval (a, b), then for any points c and x in the closed interval $[a, b]$,

$$f(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(c)(x - c)^k + E_n(x) \tag{1}$$

where, for some point ξ between c and x, the error term is

$$E_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - c)^{n+1}$$