Applied Mathematical Sciences
Volume 27

Editors
J.E. Marsden L. Sirovich

Advisors
S. Antman J.K. Hale
P. Holmes T. Kamble J. Keller
B.J. Matkowsky C.S. Peskin

Springer
New York
Berlin
Heidelberg
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Applied Mathematical Sciences

1. *John:* Partial Differential Equations, 4th ed.
2. *Sirovich:* Techniques of Asymptotic Analysis.
3. *Hale:* Theory of Functional Differential Equations, 2nd ed.
4. *Percus:* Combinatorial Methods.
5. *von Mises/Friedrichs:* Fluid Dynamics.
6. *Freiberger/Grenander:* A Short Course in Computational Probability and Statistics.
7. *Pipkin:* Lectures on Viscoelasticity Theory.
8. *Giacaglia:* Perturbation Methods in Non-linear Systems.
9. *Friedrichs:* Spectral Theory of Operators in Hilbert Space.
10. *Stroud:* Numerical Quadrature and Solution of Ordinary Differential Equations.
11. *Wolovich:* Linear Multivariable Systems.
12. *Berkovitz:* Optimal Control Theory.
13. *Bluman/Cole:* Similarity Methods for Differential Equations.
14. *Yoshizawa:* Stability Theory and the Existence of Periodic Solution and Almost Periodic Solutions.
15. *Braun:* Differential Equations and Their Applications, 3rd ed.
16. *Lefschetz:* Applications of Algebraic Topology.
17. *Collatz/Wetterling:* Optimization Problems.
18. *Grenander:* Pattern Synthesis: Lectures in Pattern Theory, Vol. I.
19. *Marsden/McCracken:* Hopf Bifurcation and Its Applications.
20. *Driver:* Ordinary and Delay Differential Equations.
21. *Courant/Friedrichs:* Supersonic Flow and Shock Waves.
22. *Rouche/Habets/Laloy:* Stability Theory by Liapunov's Direct Method.
23. *Lamperti:* Stochastic Processes: A Survey of the Mathematical Theory.
24. *Grenander:* Pattern Analysis: Lectures in Pattern Theory, Vol. II.
25. *Davies:* Integral Transforms and Their Applications, 2nd ed.
26. *Kushner/Clark:* Stochastic Approximation Methods for Constrained and Unconstrained Systems.
27. *de Boor:* A Practical Guide to Splines, Rev. ed.
28. *Keilson:* Markov Chain Models—Rarity and Exponentiality.
29. *de Veubeke:* A Course in Elasticity.
30. *Sniatycki:* Geometric Quantization and Quantum Mechanics.
31. *Reid:* Sturmian Theory for Ordinary Differential Equations.
32. *Meis/Markowitz:* Numerical Solution of Partial Differential Equations.
33. *Grenander:* Regular Structures: Lectures in Pattern Theory, Vol. III.
34. *Kevorkian/Cole:* Perturbation Methods in Applied Mathematics.
35. *Carr:* Applications of Centre Manifold Theory.
36. *Bengtsson/Ghil/Källén:* Dynamic Meteorology: Data Assimilation Methods.
37. *Saperstone:* Semidynamical Systems in Infinite Dimensional Spaces.
38. *Lichtenberg/Lieberman:* Regular and Chaotic Dynamics, 2nd ed.
39. *Piccini/Stampacchia/Vidossich:* Ordinary Differential Equations in $\mathbf{R}^n$.
40. *Naylor/Sell:* Linear Operator Theory in Engineering and Science.
41. *Sparrow:* The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors.
42. *Guckenheimer/Holmes:* Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.
43. *Ockendon/Taylor:* Inviscid Fluid Flows.
44. *Pazy:* Semigroups of Linear Operators and Applications to Partial Differential Equations.
45. *Glashoff/Gustafson:* Linear Operations and Approximation: An Introduction to the Theoretical Analysis and Numerical Treatment of Semi-Infinite Programs.
46. *Wilcox:* Scattering Theory for Diffraction Gratings.
47. *Hale et al:* An Introduction to Infinite Dimensional Dynamical Systems—Geometric Theory.
48. *Murray:* Asymptotic Analysis.
49. *Ladyzhenskaya:* The Boundary-Value Problems of Mathematical Physics.
50. *Wilcox:* Sound Propagation in Stratified Fluids.
51. *Golubitsky/Schaeffer:* Bifurcation and Groups in Bifurcation Theory, Vol. I.
52. *Chipot:* Variational Inequalities and Flow in Porous Media.
53. *Majda:* Compressible Fluid Flow and System of Conservation Laws in Several Space Variables.
54. *Wasow:* Linear Turning Point Theory.
55. *Yosida:* Operational Calculus: A Theory of Hyperfunctions.
56. *Chang/Howes:* Nonlinear Singular Perturbation Phenomena: Theory and Applications.
57. *Reinhardt:* Analysis of Approximation Methods for Differential and Integral Equations.
58. *Dwoyer/Hussaini/Voigt (eds):* Theoretical Approaches to Turbulence.
59. *Sanders/Verhulst:* Averaging Methods in Nonlinear Dynamical Systems.

(continued following index)

Carl de Boor

A Practical Guide to Splines

Revised Edition

With 32 figures

Springer

Carl de Boor
Department of Computer Sciences
University of Wisconsin-Madison
Madison, WI 53706-1685
USA
deboor@cs.wisc.edu

Editors

J.E. Marsden
Control and Dynamical Systems, 107-81
California Institute of Technology
Pasadena, CA 91125
USA

L. Sirovich
Division of Applied Mathematics
Brown University
Providence, RI 02912
USA

Mathematics Subject Classification (2000): 65Dxx, 41A15

Library of Congress Cataloging-in-Publication Data
de Boor, Carl.
A practical guide to splines / Carl de Boor. — Rev. ed.
p. cm. — (Applied mathematical sciences ; 27)
Includes bibliographical references and index.
ISBN 0-387-95366-3 (alk. paper)
1. Spline theory. I. Title. II. Applied mathematical sciences (Springer-Verlag New York Inc.) ; v. 27
QA1 .A647 vol. 27 2001
[QA224]
510s—dc21
[511'.42]
2001049644

Printed on acid-free paper.

First hardcover printing, 2001.

© 2001, 1978 Springer-Verlag New York, Inc.
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use of general descriptive names, trade names, trademarks, etc., in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Production managed by Yong-Soon Hwang; manufacturing supervised by Jacqui Ashri.
Photocomposed copy prepared from the author's files.
Printed and bound by Maple-Vail Book Manufacturing Group, York, PA.
Printed in the United States of America.

9 8 7 6 5 4 3 2 1

ISBN 0-387-95366-3 SPIN 10853120

Springer-Verlag New York Berlin Heidelberg
A member of BertelsmannSpringer Science+Business Media GmbH

Preface

This book is a reflection of my limited experience with calculations involving polynomial splines. It stresses the representation of splines as linear combinations of B-splines, provides proofs for only some of the results stated but offers many Fortran programs, and presents only those parts of spline theory that I found useful in calculations. The particular literature selection offered in the bibliography shows the same bias; it contains only items to which I needed to refer in the text for a specific result or a proof or additional information and is clearly not meant to be representative of the available spline literature. Also, while I have attached names to some of the results used, I have not given a careful discussion of the historical aspects of the field. Readers are urged to consult the books listed in the bibliography (they are marked with an asterisk) if they wish to develop a more complete and balanced picture of spline theory.

The following outline should provide a fair idea of the intent and content of the book.

The first chapter recapitulates material needed later from the ancient theory of polynomial interpolation, in particular, divided differences. Those not familiar with divided differences may find the chapter a bit terse. For comfort and motivation, I can only assure them that every item mentioned will actually be used later. The rudiments of polynomial approximation theory are given in Chapter II for later use, and to motivate the introduction of piecewise polynomial (or, pp) functions.

Readers intent upon looking at the general theory may wish to skip the next four chapters, as these follow somewhat the historical development, with piecewise linear, piecewise cubic, and piecewise parabolic approximation discussed, in that order and mostly in the context of interpolation. Proofs are given for results that, later on in the more general context of splines of arbitrary order, are only stated. The intent is to summarize elementary spline theory in a practically useful yet simple setting.

The general theory is taken up again starting with Chapter VII, which, along with Chapter VIII, is devoted to the computational handling of pp functions of arbitrary order. B-splines are introduced in Chapter IX. It is

only in that chapter that a formal definition of "spline" as a linear combination of B-splines is given. Chapters X and XI are intended to familiarize the reader with B-splines.

The remaining chapters contain various applications, all (with the notable exception of taut spline interpolation in Chapter XVI) involving B-splines. Chapter XII is the pp companion piece to Chapter II; it contains a discussion of how well a function can be approximated by pp functions. Chapter XIII is devoted to various aspects of spline interpolation as a particularly simple, computationally efficient yet powerful scheme for spline approximation in the presence of exact data. For noisy data, the smoothing spline and least–squares spline approximation are offered in Chapter XIV. Just one illustration of the use of splines in solving differential equations is given, in Chapter XV, where an ordinary differential equation is solved by collocation. Chapter XVI contains an assortment of items, all loosely connected to the approximation of a curve. It is only here (and in the problems for Chapter VI) that the beautiful theory of cardinal splines, i.e., splines on a uniform knot sequence, is discussed. The final chapter deals with the simplest generalization of splines to several variables and offers a somewhat more abstract view of the various spline approximation processes discussed in this book.

Each chapter has some problems attached to it, to test the reader's understanding of the material, to bring in additional material and to urge, at times, numerical experimentation with the programs provided. It should be understood, though, that Problem 0 in each chapter that contains programs consists of running those programs with various sample data in order to gain some first-hand practical experience with the methods espoused in the book.

The programs occur throughout the text and are meant to be read, as part of the text.

The book grew out of orientation lectures on splines delivered at Redstone Arsenal in September, 1976, and at White Sands Missile Range in October, 1977. These lectures were based on a 1973 MRC report concerning a Fortran package for calculating with B-splines, a package put together in 1971 at Los Alamos Scientific Laboratories around a routine (now called BSPLVB) that took shape a year earlier during a workshop at Oberlin organized by Jim Daniel. I am grateful for advice received during those years, from Fred Dorr, Cleve Moler, Blair Swartz and others.

During the writing of the book, I had the benefit of detailed and copious advice from John Rice who read various versions of the entire manuscript. It owes its length to his repeated pleas for further elucidation. I owe him thanks also for repeated encouragement. I am also grateful to a group at Stanford, consisting of John Bolstad, Tony Chan, William Coughran, Jr., Alphons Demmler, Gene Golub, Michael Heath, Franklin Luk, and Marcello Pagano that, through the good offices of Eric Grosse, gave me much welcome advice after reading an early version of the manuscript. The

programs in the book would still be totally unreadable but for William Coughran's and Eric Grosse's repeated arguments in favor of comment cards. Dennis Jespersen read the final manuscript with astonishing care and brought a great number of mistakes to my attention. He also raised many questions, many of which found place among the problems at the end of chapters. Walter Gautschi, and Klaus Böhmer and his students, read a major part of the manuscript and uncovered further errors. I am grateful to them all.

Time for writing, and computer time, were provided by the Mathematics Research Center under Contract No. DAAG29–75–C–0024 with the U.S. Army Research Office. Through its visitor program, the Mathematics Research Center also made possible most of the helpful contacts acknowledged earlier. I am deeply appreciative of the mathematically stimulating and free atmosphere provided by the Mathematics Research Center.

Finally, I would like to thank Reinhold de Boor for the patient typing of the various drafts.

Carl de Boor
Madison, Wisconsin
February 1978

The present version differs from the original in the following respects. The book is now typeset (in *plain* TEX; thank you, Don Knuth!), the Fortran programs now make use of FORTRAN 77 features, the figures have been redrawn with the aid of MATLAB (thank you, Cleve Moler and Jack Little!), various errors have been corrected, and many more formal statements have been provided with proofs. Further, all formal statements and equations have been numbered by the *same* numbering system, to make it easier to find any particular item. A major change has occurred in Chapters IX–XI where the B-spline theory is now developed directly from the recurrence relations without recourse to divided differences (except for the derivation of the recurrence relations themselves). This has brought in knot insertion as a powerful tool for providing simple proofs concerning the shape-preserving properties of the B-spline series.

I gratefully acknowledge support from the Army Research Office and from the Division of Mathematical Sciences of the National Science Foundation.

Special thanks are due to Peter de Boor, Kirk Haller, and S. Nam for their substantial help, and to Reinhold de Boor for the protracted final editing of the TEX files and for all the figures.

Carl de Boor
Madison, Wisconsin
October 2000

Contents

III · Piecewise Linear Approximation

IV · Piecewise Cubic Interpolation

V · Best Approximation Properties of Complete Cubic Spline Interpolation and Its Error 51

VI · Parabolic Spline Interpolation 59

VII · A Representation for Piecewise Polynomial Functions

VIII · The Spaces $\Pi_{<k,\xi,\nu}$ and the Truncated Power Basis

IX · The Representation of PP Functions by B-Splines

X · The Stable Evaluation of B-Splines and Splines

XI · The B-Spline Series, Control Points, and Knot Insertion

XII · Local Spline Approximation and the Distance from Splines

XIII · Spline Interpolation

XIV · Smoothing and Least-Squares Approximation

XV · The Numerical Solution of an Ordinary Differential Equation by Collocation

XVI · Taut Splines, Periodic Splines, Cardinal Splines and the Approximation of Curves

XVII · Surface Approximation by Tensor Products

Notation

Here is a detailed list of all the notation used in this book. Readers will have come across some of them, perhaps most of them. Still, better to bore them now than to mystify them later.

$:=$ is the sign indicating "equal by definition". It is asymmetric as such a sign should be (as none of the customary alternatives, such as $\equiv$, or $\stackrel{\text{def}}{=}$, or $\stackrel{\triangle}{=}$, etc., are). Its meaning: "$a := b$" indicates that a is the quantity to be defined or explained, and b provides the definition or explanation, and "$b =: a$" has the same meaning.

$\{ x, y, z, \ldots \} :=$ the set comprising the elements x, y, z, ...

$\{ x \in X : P(x) \} :=$ the set of elements of X having the property $P(x)$.

$(x, y, \ldots) :=$ the sequence whose first term is x, whose second term is y, ...

$\#S :=$ the number of elements (or terms) in the set (or sequence) S.

$\emptyset :=$ the empty set.

$\mathbb{N} := \{1, 2, 3, \ldots\}$.

$\mathbb{Z} := \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

$\mathbb{R} :=$ the set of real numbers.

$\mathbb{C} :=$ the set of complex numbers.

$\overline{z} :=$ the complex conjugate of the complex number z.

$[a \,..\, b] := \{ x \in \mathbb{R} : a \leq x \leq b \}$, a closed interval. This leaves $[a, b]$ free to denote a first divided difference (or, perhaps, the matrix with the two columns a and b).

$(a \,..\, b) := \{ x \in \mathbb{R} : a < x < b \}$, an open interval. This leaves (a, b) free to denote a particular sequence, e.g., a point in the plane (or, perhaps, the inner product of two vectors in some inner product space). Analogously, $[a \,..\, b)$ and $(a \,..\, b]$ denote half-open intervals.

$\text{const}_{\alpha,\ldots,\omega} :=$ a constant that may depend on $\alpha, \ldots, \omega$.

$f: A \to B: a \mapsto f(a)$ describes the function f as being defined on $A =: \operatorname{dom} f$ (called its **domain**) and taking values in the set $B =: \operatorname{tar} f$ (called its **target**), and carrying the typical element $a \in A$ to the element $f(a) \in B$. For example, $F: \mathbb{R} \to \mathbb{R}: x \mapsto \exp(x)$ describes the exponential function. I will use at times $f: a \mapsto f(a)$ if the domain and target of f are understood from the context. Thus, $\mu: f \mapsto \int_0^1 f(x)\,dx$ describes the linear functional μ that takes the number $\int_0^1 f(x)\,dx$ as its value at the function f, presumably defined on $[0\,..\,1]$ and integrable there.

$\operatorname{supp} f := \{\, x \in \operatorname{dom} f : f(x) \neq 0 \,\}$, the support of f. Note that, in Analysis, it is the *closure* of this set that is, by definition, the support of f.

$f_{|I} :=$ $g: I \to B: a \mapsto f(a)$, the restriction of f to I.

$h \dashrightarrow \begin{cases} 0^+ \\ 0^- \end{cases} :=$ h approaches 0 through $\begin{matrix} \text{positive} \\ \text{negative} \end{matrix}$ values.

$f(a^+) :=$ $\lim_{h \to 0^+} f(a+h)$, $f(a^-) := \lim_{h \to 0^-} f(a+h)$.

$\operatorname{jump}_a f :=$ $f(a^+) - f(a^-)$, the jump in f across a.

$g(x) = \mathcal{O}(f(x))$ (in words, "$g(x)$ is of order $f(x)$") as x approaches a $:=$ $\limsup_{x \to a} |\frac{g(x)}{f(x)}| < \infty$. The lim sup itself is called the **order constant** of this order relation.

$g(x) = o(f(x))$ (in words, "$g(x)$ is of higher order than $f(x)$") as x approaches a $:= \lim_{x \to a} \frac{g(x)}{f(x)} = 0$.

$f(\cdot, y) :=$ the function of one variable obtained from the function $f: X \times Y \to Z$ by holding the second variable at a fixed value y. Also, $\int_a^b G(\cdot, y) g(y)\,dy$ describes the function that results when a certain integral operator is applied to the function g.

$(x)_+ :=$ $\max\{x, 0\}$, the **truncation function**.

$\ln x :=$ the natural logarithm of x.

$\lfloor x \rfloor :=$ $\max\{\, n \in \mathbb{Z} : n \leq x \,\}$, the **floor** function.

$\lceil x \rceil :=$ $\min\{\, n \in \mathbb{Z} : n \geq x \,\}$, the **ceiling** function.

$\ell_i :=$ a Lagrange polynomial (p. 2).

$(-)^r :=$ $(-1)^r$.

$\binom{k}{r} :=$ $\frac{k!}{r!(k-r)!}$, a binomial coefficient.

$\mathcal{E}_n :=$ The Euler spline of degree n (p. 65).

Boldface symbols denote sequences or vectors, the ith term or entry is denoted by the same letter in ordinary type and subscripted by i. Thus,

τ, (τ_i), $(\tau_i)_1^n$, $(\tau_i)_{i=1}^n$, $(\tau_i : i = 1, \ldots, n)$, and $(\tau_1, \ldots, \tau_n)$ are various ways of describing the same n-vector.

$\mathbf{m} := (m, \ldots, m)$, for $m \in \mathbb{Z}$.

$X^n := \{ (x_i)_1^n : x_i \in X, \text{ all } i \}$.

$\Delta\tau_i := \tau_{i+1} - \tau_i$, **the forward difference.**

$\nabla\tau_i := \tau_i - \tau_{i-1}$, **the backward difference.**

$S^-\tau :=$ number of strong sign changes in τ (p. 138).

$S^+\tau :=$ number of weak sign changes in τ (p. 232).

$$\sum_{i=\tau}^{s} \tau_i := \begin{cases} \tau_\tau + \tau_{\tau+1} + \cdots + \tau_s, & \text{if } \tau \le s; \\ 0, & \text{if } \tau > s. \end{cases}$$

$\sum_i \tau_i := \sum_{i=\tau}^{s} \tau_i$, with τ and s understood from the context.

$$\prod_{i=\tau}^{s} \tau_i := \begin{cases} \tau_\tau \cdot \tau_{\tau+1} \cdots \tau_s, & \text{if } \tau \le s; \\ 1, & \text{if } \tau > s. \end{cases}$$

For nondecreasing sequences or meshes τ, we use

$|\tau| := \max_i \Delta\tau_i$, **the mesh size.**

$M_\tau := \max_{i,j} \Delta\tau_i/\Delta\tau_j$, **the global mesh ratio.**

$m_\tau := \max_{|i-j|=1} \Delta\tau_i/\Delta\tau_j$, **the local mesh ratio.**

$\tau_{i+1/2} := (\tau_i + \tau_{i+1})/2$.

Matrices are usually denoted by capital letters, their entries by corresponding lower case letters doubly subscripted. A, (a_{ij}), $(a_{i,j})_1^{m,n}$, $(a_{ij})_{i=1;j=1}^{m;n}$, $\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$ are various ways of describing the same matrix.

$A^T := (a_{ji})_{j=1;i=1}^{n;m}$, the transpose of A.

$A^H := (\overline{a}_{ji})_{j=1;i=1}^{n;m}$, the conjugate transpose or Hermitian of A.

$\det A :=$ the determinant of A.

$\delta_{ij} := \begin{cases} 1 & i = j \\ 0 & i \ne j \end{cases}$, the Kronecker Delta.

$\operatorname{span}(\varphi_i) := \{ \sum_i \alpha_i\varphi_i : \alpha_i \in \mathbb{R} \}$, the linear combinations of the sequence $(\varphi_i)_1^n$ of elements of a linear space X. Such a sequence is a basis for its span in case it is linearly independent, that is, in case $\sum_i \alpha_i\varphi_i = 0$ implies that $\boldsymbol{\alpha} = \mathbf{0}$. We note that the linear independence of such a sequence $(\varphi_i)_1^n$ is almost invariably proved by exhibiting a corresponding sequence $(\lambda_i)_1^n$ of linear functionals on X for which the matrix $(\lambda_i\varphi_i)$ is invertible, e.g., $\lambda_i\varphi_j = \delta_{ij}$, for all i, j. In such a case, $\dim \operatorname{span}(\varphi_i)_1^n = n$.

$\Pi_{<k} := \Pi_{k-1}$:= linear space of polynomials of order k (p. 1).

$\Pi_{<k,\xi} := \Pi_{k-1,\xi}$:= linear space of pp functions of order k with break sequence ξ (p. 70).

$D^j f :=$ jth derivative of f; for $f \in \Pi_{<k,\xi}$, see p. 70.

$\Pi_{<k,\xi,\nu} := \Pi_{k-1,\xi,\nu}$:= linear subspace of $\Pi_{<k,\xi}$ consisting of those elements that satisfy continuity conditions specified by ν (p. 82).

$\$_{k,\mathbf{t}} := \text{span}(B_{i,k,\mathbf{t}})$, linear space of splines of order k with knot sequence $\mathbf{t}$ (p. 93).

$B_i := B_{i,k,\mathbf{t}} :=$ ith B-spline of order k with knot sequence $\mathbf{t}$ (p. 87).

$\$_{k,n} := \cup\{ f \in \$_{k,\mathbf{t}} : t_1 = \cdots = t_k = a, t_{n+1} = \cdots = t_{n+k} = b \}$ (pp. 163, 239).

$\$^{\text{nat}}_{k,\mathbf{x}} :=$ "natural" splines of order k for the sites $\mathbf{x}$ (p. 207).

$C[a\,..\,b] := \{ f\colon [a\,..\,b] \to \mathbb{R} : f \text{ continuous} \}$.

$\|f\| := \max\{ |f(x)| : a \le x \le b \}$, the uniform norm of $f \in C[a\,..\,b]$. (We note that $\|f+g\| \le \|f\| + \|g\|$ and $\|\alpha f\| = |\alpha|\|f\|$ for $f, g \in C[a\,..\,b]$ and $\alpha \in \mathbb{R}$).

$\omega(f;h) := \max\{ |f(x) - f(y)| : x, y \in [a\,..\,b], |x-y| \le h \}$, the modulus of continuity for $f \in C[a\,..\,b]$ (p. 25).

$\text{dist}\,(g, S) := \inf\{ \|g - f\| : f \in S \}$, the distance of $g \in C[a\,..\,b]$ from the subset S of $C[a\,..\,b]$.

$C^{(n)}[a\,..\,b] := \{ f\colon [a\,..\,b] \to \mathbb{R} : f$ is n times continuously differentiable $\}$.

$[\tau_i, \ldots, \tau_j]f :=$ divided difference of order $j-i$ of f, at the sites $\tau_i, \ldots, \tau_j$ (p. 3). In particular, $[\tau_i]\colon f \mapsto f(\tau_i)$.

Special spline approximation maps:

$I_k :=$ interpolation by splines of order k (p. 182),

$L_k :=$ Least-squares approximation by splines of order k (p. 220),

$V :=$ Schoenberg's variation diminishing spline approximation (p. 141).

I

Polynomial Interpolation

In this introductory chapter, we state, mostly without proof, those basic facts about polynomial interpolation and divided differences needed in subsequent chapters. The reader who is unfamiliar with some of this material is encouraged to consult textbooks such as Isaacson & Keller [1966] or Conte & de Boor [1980] for a more detailed presentation.

One uses polynomials for approximation because they can be evaluated, differentiated, and integrated easily and in finitely many steps using the basic arithmetic operations of addition, subtraction, and multiplication. A polynomial of **order** n is a function of the form

$$(1) \qquad p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1} = \sum_{j=1}^{n} a_j x^{j-1},$$

i.e., a polynomial of degree $< n$. It turns out to be more convenient to work with the *order* of a polynomial than with its degree since the set of all polynomials of degree n fails to be a linear space, while the set of all polynomials of order n forms a linear space, denoted here by

$$\Pi_{<n} = \Pi_{\leq n-1} = \Pi_{n-1}.$$

Note that a polynomial of order n has exactly n degrees of freedom.

Note also that, in MATLAB, hence in the SPLINE TOOLBOX (de Boor [1990]$_2$), the coefficient sequence `a = [a(1), ... ,a(n)]` of a polynomial of order n starts with the *highest* coefficient. In particular, if `x` is a scalar, then the MATLAB command `polyval(a,x)` returns the number

```
a(1)*x^(n-1) + a(2)*x^(n-2) + ... + a(n-1)*x + a(n)
```

Polynomial interpolation: Lagrange form Let $\tau := (\tau_i)_1^n$ be a sequence of n distinct sites. Then

$$\ell_i(x) := \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-\tau_j}{\tau_i-\tau_j} \tag{2}$$

is the ith **Lagrange polynomial** for τ. It is a polynomial of order n and vanishes at all τ_j's except for τ_i at which it takes the value 1. We write this with the aid of the Kronecker delta as

$$\ell_i(\tau_j) = \delta_{ij} := \begin{cases} 0, & i \neq j; \\ 1, & i = j. \end{cases}$$

Hence, for an arbitrary given function g,

$$p := \sum_{i=1}^{n} g(\tau_i)\ell_i$$

is an element of $\Pi_{<n}$ and satisfies

$$p(\tau_i) = g(\tau_i), \qquad i = 1, \ldots, n.$$

In this way, we obtain, for arbitrary g, a function p in $\Pi_{<n}$ that matches it at the n sites $\tau_1, \ldots, \tau_n$. This shows that the linear system

$$a_1 + a_2\tau_i + \cdots + a_n\tau_i^{n-1} = b_i, \quad i = 1, \ldots, n,$$

has a solution for arbitrary right sides, and, since this linear system is square, this implies that the solution is unique, that is, $p = \sum_{i=1}^{n} g(\tau_i)\ell_i$ is the only interpolant from $\Pi_{<n}$ to g at τ.

(3) Theorem. *If $\tau_1, \ldots, \tau_n$ are distinct sites, and $g(\tau_1), \ldots, g(\tau_n)$ are the given data, then there exists exactly one polynomial $p \in \Pi_{<n}$ for which $p(\tau_i) = g(\tau_i)$, $i = 1, \ldots, n$. This polynomial can be written in* **Lagrange form**

$$p = \sum_{i=1}^{n} g(\tau_i)\ell_i\,, \tag{4}$$

with $\ell_i(x)$ given by (2).

The Lagrange form is certainly quite elegant. But, compared to other ways of writing and evaluating the interpolating polynomials, it is far from the most efficient. To illustrate this point, we consider briefly the computational cost of evaluating the Lagrange form at a site. We denote by A an

addition or subtraction, and by M/D a multiplication or division. Straight evaluation of the Lagrange form takes

$$(2n-2)A + (n-2)M + (n-1)D$$

for each of the n numbers $\ell_i(x)$, and then

$$(n-1)A + nM$$

for forming (4) from the $g(\tau_i)$ and the $\ell_i(x)$. Even if one is clever about it and computes

$$Y_i := g(\tau_i) \Big/ \prod_{j \neq i} (\tau_i - \tau_j), \qquad i = 1, \ldots, n,$$

once and for all, and then computes $p(x)$ by

$$\varphi(x) := \prod_{i=1}^{n} (x - \tau_i),$$

$$p(x) = \varphi(x) \sum_{i=1}^{n} \frac{Y_i}{x - \tau_i},$$

it still takes

$$(2n-1)A + nM + nD$$

per site, compared to

$$(2n-1)A + (n-1)M$$

for the Newton form to be discussed next. And things get much worse if we want to compute derivatives!

Of all the customary forms for the interpolating polynomial, I prefer the Newton form. It strikes me as the best compromise between ease of construction and ease of evaluation. In addition, it leads to a very simple analysis of interpolation error and even allows one to discuss and use osculatory polynomial interpolation with no additional effort.

Polynomial Interpolation: Divided differences and Newton form

There are many ways of defining divided differences. I prefer the following (somewhat nonconstructive)

(5) Definition. The kth divided difference of a function g at the sites $\tau_i, \ldots, \tau_{i+k}$ *is the leading coefficient* (that is, the coefficient of x^k) *of the polynomial* of order $k+1$ *that agrees with g at the sequence* $(\tau_i, \ldots, \tau_{i+k})$ (in the sense of Definition (12) below). *It is denoted by*

$$[\tau_i, \ldots, \tau_{i+k}]g.$$

This definition has the following immediate consequences.

(i) *If $p_i \in \Pi_{<i}$ agrees with g at $\tau_1, \ldots, \tau_i$ for $i = k$ and $k+1$, then*

$$p_{k+1}(x) = p_k(x) + (x - \tau_1) \cdots (x - \tau_k)[\tau_1, \ldots, \tau_{k+1}]g. \tag{6}$$

For, we know that $p_{k+1} - p_k$ is a polynomial of order $k+1$ that vanishes at $\tau_1, \ldots, \tau_k$ and has $[\tau_1, \ldots, \tau_{k+1}]g$ as its leading coefficient, therefore must be of the form

$$p_{k+1}(x) - p_k(x) = C(x - \tau_1) \cdots (x - \tau_k)$$

with $C = [\tau_1, \ldots, \tau_{k+1}]g$.

This property shows that divided differences can be used to *build up* the interpolating polynomial by adding the data sites τ_i one at a time. In this way, we obtain

$$\begin{aligned} p_n(x) &= p_1(x) + (p_2(x) - p_1(x)) + \cdots + (p_n(x) - p_{n-1}(x)) \\ &= [\tau_1]g + (x - \tau_1)[\tau_1, \tau_2]g + (x - \tau_1)(x - \tau_2)[\tau_1, \tau_2, \tau_3]g + \\ &\qquad \cdots + (x - \tau_1) \cdots (x - \tau_{n-1})[\tau_1, \ldots, \tau_n]g. \end{aligned}$$

This is the **Newton form**,

$$p_n(x) = \sum_{i=1}^{n} (x - \tau_1) \cdots (x - \tau_{i-1})[\tau_1, \ldots, \tau_i]g, \tag{7}$$

for the polynomial p_n of order n that agrees with g at $\tau_1, \ldots, \tau_n$. (Here, $(x - \tau_i) \cdots (x - \tau_j) := 1$ if $i > j$.)

(ii) *$[\tau_i, \ldots, \tau_{i+k}]g$ is a symmetric function of its arguments $\tau_i, \ldots, \tau_{i+k}$*, that is, it depends only on the numbers $\tau_i, \ldots, \tau_{i+k}$ and not on the order in which they occur in the argument list. This is clear from the definition since the interpolating polynomial depends only on the data points and not on the order in which we write down these points.

(iii) *$[\tau_i, \ldots, \tau_{i+k}]g$ is linear in g*, that is, if $f = \alpha g + \beta h$ for some functions g and h and some numbers α and β, then $[\tau_i, \ldots, \tau_{i+k}]f = \alpha[\tau_i, \ldots, \tau_{i+k}]g + \beta[\tau_i, \ldots, \tau_{i+k}]h$, as follows from the uniqueness of the interpolating polynomial.

The next property is essential for the material in Chapter X.

(iv) (**Leibniz' formula**). *If $f = gh$, that is, $f(x) = g(x)h(x)$ for all x, then*

$$[\tau_i, \ldots, \tau_{i+k}]f = \sum_{r=i}^{i+k} ([\tau_i, \ldots, \tau_r]g)([\tau_r, \ldots, \tau_{i+k}]h).$$

For the proof, observe that the function

$$\sum_{r=i}^{i+k} (x - \tau_i) \cdots (x - \tau_{r-1})[\tau_i, \ldots, \tau_r]g \sum_{s=i}^{i+k} (x - \tau_{s+1}) \cdots (x - \tau_{i+k})[\tau_s, \ldots, \tau_{i+k}]h$$

agrees with f at $\tau_i, \ldots, \tau_{i+k}$ since, by (7), the first factor agrees with g and the second factor agrees with h there. Now multiply out and split the resulting double sum into two parts:

$$\sum_{r,s=i}^{i+k} = \sum_{r \le s} + \sum_{r > s}.$$

Note that $\sum_{r>s}$ vanishes at $\tau_i, \ldots, \tau_{i+k}$. Therefore, also $\sum_{r \le s}$ must agree with f at $\tau_i, \ldots, \tau_{i+k}$. But $\sum_{r \le s}$ is a polynomial of order $k+1$. Therefore, its leading coefficient which is

$$\sum_{r=s} ([\tau_i, \ldots, \tau_r]g)\ ([\tau_s, \ldots, \tau_{i+k}]h)$$

must equal $[\tau_i, \ldots, \tau_{i+k}]f$. (I learned this particular argument from W. D. Kammler.)

(v) *If g is a polynomial of order $k+1$, then $[\tau_i, \ldots, \tau_{i+k}]g$ is constant as a function of $\tau_i, \ldots, \tau_{i+k}$. In particular*

$$[\tau_i, \ldots, \tau_{i+k}]g = 0 \qquad \text{for all } g \in \Pi_{<k}.$$

This, too, is quite clear since, in this case, the polynomial of order $k+1$ that agrees with g at $\tau_i, \ldots, \tau_{i+k}$ (or at any $k+1$ sites for that matter) must be g itself, by the uniqueness of the interpolating polynomial. In particular, every $g \in \Pi_{<k}$ has leading coefficient 0 when it is considered as a polynomial of order $k+1$.

As for specific values, we have at once

$$[\tau_1]g = g(\tau_1)$$

Also,

$$[\tau_1, \tau_2]g = \frac{g(\tau_1) - g(\tau_2)}{\tau_1 - \tau_2} \qquad \text{if } \tau_1 \neq \tau_2 \tag{8}$$

from the familiar two-point formula for the secant,

$$p(x) = g(\tau_1) + (x - \tau_1)\frac{g(\tau_1) - g(\tau_2)}{\tau_1 - \tau_2}. \tag{9}$$

But if g is continuously differentiable, then, for $\tau_2 \to \tau_1$,

$$\lim_{\tau_2 \to \tau_1} \frac{g(\tau_1) - g(\tau_2)}{\tau_1 - \tau_2} = g'(\tau_1)$$

and the secant (9) goes over into the tangent

$$p(x) = g(\tau_1) + (x - \tau_1)g'(\tau_1) \tag{10}$$

that agrees with g in the value *and* slope at τ_1. This justifies the statement that the polynomial p given by (10) *agrees with g at the sites τ_1 and τ_1*, that is, *two-fold at τ_1*. With this wording, the definition of the first divided difference implies that

$$[\tau_1, \tau_2]g = g'(\tau_1) \qquad \text{if } \tau_1 = \tau_2. \tag{11}$$

Also, we see that $[\tau_1, \tau_2]g$ is a continuous function of its arguments.

Repeated interpolation at a site is called **osculatory interpolation** since it produces higher than first order contact between the function and its interpolant (*osculari* means "to kiss" in Latin). The precise definition is as follows.

(12) Definition. *Let $\tau := (\tau_i)_1^n$ be a sequence of sites not necessarily distinct. We say that* **the function p agrees with the function g at** *τ provided that, for every site ζ that occurs m times in the sequence $\tau_1, \ldots, \tau_n$, p and g agree m-fold at ζ, that is,*

$$p^{(i-1)}(\zeta) = g^{(i-1)}(\zeta) \qquad \text{for } i = 1, \ldots, m.$$

The definition of the kth divided difference $[\tau_i, \ldots, \tau_{i+k}]g$ of g is to be taken in this sense when some or all of the sites $\tau_i, \ldots, \tau_{i+k}$ coincide.

For the proof of the following additional properties of the kth divided difference, we refer the reader to Issacson & Keller [1966] or to Conte & de Boor [1980] (although these properties could be proven directly from the definition).

(vi) *$[\tau_i, \ldots, \tau_{i+k}]g$ is a continuous function of its $k+1$ arguments in case* $g \in C^{(k)}$, that is, g has k continuous derivatives.

(vii) *If $g \in C^{(k)}$, that is, g has k continuous derivatives, then there exists a site ζ in the smallest interval containing $\tau_i, \ldots, \tau_{i+k}$ so that*

$$[\tau_i, \ldots, \tau_{i+k}]g = g^{(k)}(\zeta)/k!.$$

(viii) For computations, it is important to note that

$$[\tau_i, \ldots, \tau_{i+k}]g = \frac{g^{(k)}(\tau_i)}{k!} \quad \text{if } \tau_i = \cdots = \tau_{i+k}, \quad g \in C^{(k)},$$

while

$$[\tau_i, \ldots, \tau_{i+k}]g =$$

$$\frac{[\tau_i, \ldots, \tau_{r-1}, \tau_{r+1}, \ldots, \tau_{i+k}]g - [\tau_i, \ldots, \tau_{s-1}, \tau_{s+1}, \ldots, \tau_{i+k}]g}{\tau_s - \tau_r} \tag{13}$$

if τ_r, τ_s are any two *distinct* sites in the sequence $\tau_i, \ldots, \tau_{i+k}$. The arbitrariness in the choice of τ_r and τ_s here is to be expected in view of the symmetry property (ii).

(14) Theorem (Osculatory interpolation). *If $g \in C^{(n)}$, that is, g has n continuous derivatives, and $\tau = (\tau_i)_1^n$ is a sequence of n arbitrary sites* (not necessarily distinct), *then, for all x,*

$$g(x) = p_n(x) + (x - \tau_1) \cdots (x - \tau_n)[\tau_1, \ldots, \tau_n, x]g, \tag{15}$$

with

$$p_n(x) := \sum_{i=1}^{n} (x - \tau_1) \cdots (x - \tau_{i-1})[\tau_1, \ldots, \tau_i]g. \tag{16}$$

In particular, p_n is the unique polynomial of order n that agrees with g at τ.

Note how, in (15), the *error* $g(x) - p_n(x)$ of polynomial interpolation appears in the form of the "next" term for the Newton form. One obtains (15) from its simplest (and obvious) case

$$g(x) = g(\tau_1) + (x - \tau_1)[\tau_1, x]g \tag{17}$$

by induction on n, as follows. If we already know that

$$g(x) = p_k(x) + (x - \tau_1) \cdots (x - \tau_k)[\tau_1, \ldots, \tau_k, x]g,$$

then, applying (17) (with τ_1 replaced by τ_{k+1}) to $[\tau_1, \ldots, \tau_k, x]g$ *as a function of* x, we get

$$[\tau_1, \ldots, \tau_k, x]g = [\tau_1, \ldots, \tau_{k+1}]g + (x - \tau_{k+1})[\tau_{k+1}, x]\big([\tau_1, \ldots, \tau_k, \cdot]g\big)$$

which shows that

$$g(x) = p_{k+1}(x) + (x - \tau_1) \cdots (x - \tau_{k+1})[\tau_{k+1}, x]\big([\tau_1, \ldots, \tau_k, \cdot]g\big).$$

This implies that, for any $y \neq \tau_1, \ldots, \tau_{k+1}$, the polynomial

$$p_{k+1}(x) + (x - \tau_1) \cdots (x - \tau_{k+1})[\tau_{k+1}, y]\big([\tau_1, \ldots, \tau_k, \cdot]g\big)$$

of order $k + 2$ agrees with g at $\tau_1, \ldots, \tau_{k+1}$ *and* at y, hence its leading coefficient must equal $[\tau_1, \ldots, \tau_{k+1}, y]g$, that is,

$$[\tau_{k+1}, y]\big([\tau_1, \ldots, \tau_k, \cdot]g\big) = [\tau_1, \ldots, \tau_{k+1}, y]g$$

for all $y \neq \tau_1, \ldots, \tau_{k+1}$. But then, the continuity property (vi) shows that this equality must hold for $y = \tau_1, \ldots, \tau_{k+1}$, too. This advances the inductive hypothesis. It also shows that, for $y \neq \tau_{k+1}$,

$$\begin{aligned}[\tau_1, \ldots, \tau_{k+1}, y]g &= [\tau_{k+1}, y]\big([\tau_1, \ldots, \tau_k, \cdot]g\big) \\ &= \frac{[\tau_1, \ldots, \tau_k, y]g - [\tau_1, \ldots, \tau_k, \tau_{k+1}]g}{y - \tau_{k+1}}\end{aligned}$$

and so provides, incidentally, a proof for (viii) and shows explicitly how the $(k+1)$st divided difference arises as the first divided difference of a kth divided difference, thus explaining the name.

The notion that osculatory interpolation is repeated interpolation at sites is further supported by the following observation: In (15) and (16), let all sites $\tau_1, \ldots, \tau_n$ coalesce,

$$\tau_1 = \cdots = \tau_n = \tau.$$

Then we obtain the formula (using (viii) and (vii))

$$(18) \qquad g(x) = p_n(x) + (x-\tau)^n \frac{g^{(n)}(\zeta_x)}{n!}, \qquad \text{some } \zeta_x \text{ between } \tau \text{ and } x,$$

with

$$(19) \qquad p_n(x) = \sum_{i=1}^{n} (x-\tau)^{i-1} \frac{g^{(i-1)}(\tau)}{(i-1)!},$$

which is the standard **truncated Taylor series** with differential remainder (or error) term. Note that the truncated Taylor series (19) does indeed agree with g n-fold at τ, that is, $p_n^{(i-1)}(\tau) = g^{(i-1)}(\tau)$ for $i = 1, \ldots, n$.

Divided difference table The coefficients

$$[\tau_1]g,\ [\tau_1,\tau_2]g,\ \ldots,\ [\tau_1,\ldots,\tau_n]g$$

for the Newton form (7) are efficiently computed in a divided difference table:

interp. sites	values	first div. diff.	second divided diff.	...	$(n-2)$nd divided diff.	$(n-1)$st divided diff.
τ_1	$g(\tau_1)$					
		$[\tau_1,\tau_2]g$				
τ_2	$g(\tau_2)$		$[\tau_1,\tau_2,\tau_3]g$			
		$[\tau_2,\tau_3]g$				
τ_3	$g(\tau_3)$		$[\tau_2,\tau_3,\tau_4]g$		$[\tau_1,\ldots,\tau_{n-1}]g$	
		$[\tau_3,\tau_4]g$				$[\tau_1,\ldots,\tau_n]g$
τ_4	$g(\tau_4)$				$[\tau_2,\ldots,\tau_n]g$	
$\vdots$	$\vdots$					
τ_{n-1}	$g(\tau_{n-1})$		$[\tau_{n-2},\tau_{n-1},\tau_n]g$			
		$[\tau_{n-1},\tau_n]g$				
τ_n	$g(\tau_n)$					

Assume that the interpolation sites are so ordered that repeated sites occur together, that is, so that $\tau_i = \tau_{i+r}$ can only hold if also $\tau_i = \tau_{i+1} = \cdots = \tau_{i+r-1} = \tau_{i+r}$. Then, on computing $[\tau_i, \ldots, \tau_{i+r}]g$, either $\tau_i = \tau_{i+r}$,

in which case $[\tau_i, \ldots, \tau_{i+r}]g = g^{(r)}(\tau_i)/r!$ must be given, or else $\tau_i \neq \tau_{i+r}$, in which case

$$[\tau_i, \ldots, \tau_{i+r}]g = \frac{[\tau_{i+1}, \ldots, \tau_{i+r}]g - [\tau_i, \ldots, \tau_{i+r-1}]g}{\tau_{i+r} - \tau_i}, \tag{20}$$

hence can be computed from the two entries next to it in the preceding column. This makes it possible to generate the entries of the divided difference table *column by column* from the given data. The top diagonal then contains the desired coefficients of the Newton form (7). The (arithmetic) operation count is

$$n(n-1)A + \frac{n(n-1)}{2}D.$$

(21) Example: Osculatory interpolation to the logarithm For $g(x) = \ln x$, estimate $g(1.5)$ by the number $p_4(1.5)$, with p_4 the cubic polynomial that agrees with g at 1, 1, 2, 2 (that is, in value and slope at 1 and 2). We set up the divided difference table

τ	$g(\tau)$	1. div. diff.	2. div. diff.	3. div. diff.
1	0			
		1.		
1	0		−.306853	
		.693147		.113706
2	.693147		−.193147	
		.5		
2	.693147			

Note that, because of the coincidences in the data sites, the first and third entry in the first divided difference column had to be given data. We read off the top diagonal of the table that

$$p_4(x) = 0 + (x-1)1 + (x-1)^2(-.306853) + (x-1)^2(x-2)(.113706)$$

and then compute, by hook or by crook, that $p_4(1.5) = .409074 \approx .405465 = \ln 1.5$. □

Evaluation of the Newton form Given an arbitrary sequence $\tau_1, \ldots, \tau_{n-1}$ of sites, any polynomial p of order n can be written in exactly one way as

$$p(x) = \sum_{i=1}^{n} (x-\tau_1)\cdots(x-\tau_{i-1})a_i \tag{22}$$

for suitable coefficients $a_1, \ldots, a_n$. Indeed, with τ_n an additional site, we know that the particular choice

$$a_i = [\tau_1, \ldots, \tau_i]p, \quad i = 1, \ldots, n,$$

makes the right side of (22) into the unique polynomial of order n that agrees with p at $\tau = (\tau_1, \ldots, \tau_n)$, hence, since p is itself such a polynomial, it must agree with the right side for that choice of the a_i. We call (22) the **Newton form centered at $\tau_1, \ldots, \tau_{n-1}$ for p.**

The Newton form (22) can be evaluated at a site in

$$(n-1)(2A+M)$$

arithmetic operations since it can be written in the following **nested form**

$$\begin{aligned} p(x) = a_1 + (x-\tau_1)\{a_2 + (x-\tau_2)[a_3 + \\ \cdots + (x-\tau_{n-2})(a_{n-1} + (x-\tau_{n-1})a_n)\cdots]\}. \end{aligned}$$

This suggests the following algorithm.

(23) Algorithm: Nested Multiplication. *We are given the centers $\tau_1, \ldots, \tau_{n-1}$ and associated coefficients $a_1, \ldots, a_n$ in the corresponding Newton form (22) for some $p \in \Pi_{<n}$, and also another site τ_0.*

```
1  b_n := a_n
2  for k = n-1, n-2, ..., 1, do:
2/1  b_k := a_k + (τ_0 - τ_k) b_{k+1}.
```

Then $b_1 = p(\tau_0) = \sum_{i=1}^{n} (\tau_0 - \tau_1)\cdots(\tau_0 - \tau_{i-1})a_i$. Moreover, all the b_k carry useful information about p, for we have the important property that

$$\begin{aligned} p(x) = b_1 + (x-\tau_0)b_2 + (x-\tau_0)(x-\tau_1)b_3 + \cdots \\ \cdots + (x-\tau_0)(x-\tau_1)\cdots(x-\tau_{n-2})b_n. \end{aligned} \tag{24}$$

In words, the algorithm derives from the Newton form for p based on the centers $\tau_1, \ldots, \tau_{n-1}$ the Newton form for the *same* polynomial p but now based on the centers $\tau_0, \tau_1, \ldots, \tau_{n-2}$.

One way to prove (24) is to observe that, from the algorithm,

$$b_{k+1} = (b_k - a_k)/(\tau_0 - \tau_k)$$

so that, since $b_1 = p(\tau_0)$ and $a_k = [\tau_1, \ldots, \tau_k]p$ for all k, we must have $b_k = [\tau_0, \ldots, \tau_{k-1}]p$ for all k by induction on k. Therefore, the right side of (24) is the polynomial of order n that agrees with p at $\tau_0, \ldots, \tau_{n-1}$, hence must be p itself.

This argument brings up the important point that the algorithm may be visualized as a scheme for filling in an additional diagonal on top of the

divided difference table for p, but *working from right to left* as shown here.

interp. sites	values	first div. diff.	second div. diff.	...	$(n-2)$nd div. diff.	$(n-1)$st div. diff.
τ_{-1}	c_1					
		c_2				
τ_0	b_1		c_3			
		b_2				
τ_1	a_1		b_3		c_{n-1}	
		a_2				c_n
τ_2	.		a_3		b_{n-1}	
		.				b_n
τ_3	.		.		a_{n-1}	
		.				a_n
.	.		.		.	

We can repeat the algorithm, starting with the sites $\tau_0, \ldots, \tau_{n-2}$ and the coefficients $b_1, \ldots, b_n$ and an additional site τ_{-1}, obtaining then the coefficients $c_1, \ldots, c_n$ that fit into the diagonal on top of the b_k's, that is, for which

$$p(x) = c_1 + (x - \tau_{-1})c_2 + \cdots + (x - \tau_{-1}) \cdots (x - \tau_{n-3})c_n.$$

We leave it to the reader to invent the corresponding algorithm for filling in additional diagonals on the bottom of the divided difference table.

The algorithm is particularly handy for the efficient calculation of derivatives of p at a site since $p^{(j)}(\tau)/j!$ appears as the $(j+1)$st coefficient in any Newton form for p in which the first $j+1$ centers all equal τ.

(25) Example: Computing the derivatives of a polynomial in Newton form We use the algorithm to fill in three additional diagonals in the divided difference table for p_4 constructed in (21)Example, using *each time* the point 1.5 as the new site. The last computed diagonal (see the full table on the next page) then gives the scaled derivatives of p_4 at 1.5, that is, the Taylor coefficients.

In particular, $p_4(x) = .409074 + .664721(x - 1.5) - .25(x - 1.5)^2 + .113706(x - 1.5)^3$.

We note in passing that this algorithm provides the most efficient (known) way to shift the origin for polynomials in the ordinary power form (1).

τ	$g(\tau)$	1. div. diff.	2. div. diff.	3. div. diff.
1.5	.409 0735			
		.664 7205		
1.5	.409 0735		−.25	
		.664 7205		.113 706
1.5	.409 0735		−.306 853	
		.818 147		.113 706
1.0	**0**		−.363 706	
		1.0		.113 706
1.0	0		**-.306 853**	
		.693 147		**.113 706**
2.0	.693 147		−.193 147	
		.5		
2.0	.693 147			

□

Other polynomial forms and conditions We have mentioned so far only two polynomial forms, the Lagrange form and the Newton form. The power form which we used to define polynomials is a special case of the Newton form. There are several other forms in common use, chiefly expansions in orthogonal polynomials. Such expansions are of the form

$$p = \sum_{i=1}^{n} a_i P_{i-1} \tag{26}$$

with (P_i) a sequence of polynomials satisfying a three-term recurrence,

$$\begin{aligned} P_{-1}(x) &:= 0, \qquad P_0(x) := 1, \\ P_{i+1}(x) &:= A_i(x - B_i)P_i(x) - C_i P_{i-1}(x), \qquad i = 0, 1, 2, \ldots \end{aligned} \tag{27}$$

for certain coefficients $A_i \neq 0$, B_i, C_i. Note that the Newton form (22) (although not based on orthogonal polynomials) fits this pattern, with $A_i = 1$, $B_i = \tau_{i+1}$, $C_i = 0$, all i. A surprisingly simple and very effective such basis (P_i) is provided by the Chebyshev polynomials for which

$$\begin{aligned} A_0 &= 1, \quad B_0 = 0 \\ A_j &= 2, \quad B_j = 0, \quad C_j = 1, \qquad j = 1, 2, 3, \ldots \end{aligned}$$

Evaluation of such a form (26) is efficiently accomplished by a generalization of the Nested Multiplication algorithm, based on the three-term recurrence (27) (see, for example, Conte & de Boor [1980:6.3]).

One is tempted to use such forms in preference to the power form because of considerations of condition. The *condition* of the representation (26) in

terms of the basis $(P_i)_1^n$ for $\Pi_{<n}$ measures the possible relative change in p as a result of a unit change in the coefficient vector $\mathbf{a} = (a_i)$. To be specific, we consider all polynomials as functions on the interval $[a..b]$, and measure the size of a polynomial p by the number

$$\|p\| := \max_{a \le x \le b} |p(x)|.$$

Also, we measure the size of the vector $\mathbf{a} = (a_i)$ by the number

$$\|\mathbf{a}\| := \max_{1 \le i \le n} |a_i|.$$

Then, for all coefficient vectors $\mathbf{a}$,

$$m\|\mathbf{a}\| \le \|\sum_i a_i P_{i-1}\| \le M\|\mathbf{a}\| \tag{28}$$

with

$$m := \min_{\mathbf{a}} \|\sum_i a_i P_{i-1}\|/\|\mathbf{a}\|, \qquad M := \max_{\mathbf{a}} \|\sum_i a_i P_{i-1}\|/\|\mathbf{a}\|. \tag{29}$$

The **condition** (number) of the representation (26) is then defined as

$$\operatorname{cond}(P_i) := \frac{M}{m} \tag{30}$$

and is used as follows.

In constructing that representation (26), we are bound to make rounding errors so that, instead of the vector $\mathbf{a}$, we actually produce the perturbed vector $\mathbf{a} + \delta\mathbf{a}$. Consequently, we will not have the polynomial p, but the perturbed polynomial $p + \delta p = \sum_i (a_i + \delta a_i) P_{i-1}$. Now, by (28),

$$\frac{m\|\delta\mathbf{a}\|}{M\|\mathbf{a}\|} \le \frac{\|\delta p\|}{\|p\|} \le \frac{M\|\delta\mathbf{a}\|}{m\|\mathbf{a}\|}$$

showing that a change in $\mathbf{a}$ of relative size $\|\delta\mathbf{a}\|/\|\mathbf{a}\|$ may result in a relative change $\|\delta p\|/\|p\|$ in p as large as $\operatorname{cond}(P_i)$ times the relative change in $\mathbf{a}$ (and at least $1/\operatorname{cond}(P_i)$ as large). The larger the condition, the greater the possible effect of a "small" relative change of error in $\mathbf{a}$ on the function represented.

If $P_i(x) := (x/b)^{i-1}$, $i = 1, \ldots, n$, and, without loss of generality,

$$|a| \le b,$$

then

$$M = \max_{\|\mathbf{c}\| \le 1} \|\sum_i c_i P_{i-1}\| = n.$$

For the lower bound m, we use the identity

$$p(x) = \sum_{i=1}^{n} \left(\frac{p^{(i-1)}(0) b^{i-1}}{(i-1)!} \right) \left(\frac{x}{b} \right)^{i-1}, \qquad \text{all } p \in \Pi_{<n},$$

to conclude that

$$m^{-1} = \max_{0 \le i \le n} \max_{p \in \Pi_{<n}} \frac{|p^{(i)}(0) b^i|}{i! \|p\|}.$$

If now $0 \le a < b$, then it is known (see, for example, Rivlin [1974:p. 93]) that

$$\max_{p \in \Pi_{<n}} \frac{|p^{(i)}(0)|}{\|p\|} = \mathrm{T}_{n-1}^{(i)}\left(\frac{a+b}{b-a}\right) \left(\frac{2}{b-a} \right)^i,$$

with T_{n-1} the Chebyshev polynomial of degree $n-1$. Further,

$$\sum_{i=0}^{n-1} \mathrm{T}_{n-1}^{(i)}\left(\frac{a+b}{b-a}\right) \frac{\left(\frac{2b}{b-a}\right)^i}{i!} = \mathrm{T}_{n-1}\left(\frac{a+3b}{b-a}\right).$$

We conclude that

$$\mathrm{T}_{n-1}\left(\frac{a+3b}{b-a}\right)/n \le \frac{1}{m} \le \mathrm{T}_{n-1}\left(\frac{a+3b}{b-a}\right).$$

Since $\mathrm{T}_{n-1}(x) \sim (1/2)(x - \sqrt{x^2+1})^{n-1}$ for large $|x|$, it follows that the condition of the scaled power basis $((\cdot/b)^i)_0^{n-1}$ for polynomials on the interval $[a \,..\, b]$ becomes very large, even for fixed n, as the interval length $b-a$ becomes small compared with the right end point b.

At the very least, then, a *local* power basis $\left(\left(\frac{\cdot - a}{b-a}\right)^i\right)_0^{n-1}$ should be used on an interval $[a \,..\, b]$. Its condition is between $\mathrm{T}_{n-1}(3)$ and $n\mathrm{T}_{n-1}(3)$; the first few values of $\mathrm{T}_{n-1}(3)$ are

n	$\mathrm{T}_{n-1}(3)$	n	$\mathrm{T}_{n-1}(3)$
1	1	6	3,363
2	3	7	19,601
3	17	8	114,243
4	99	9	665,857
5	577	10	3,880,899

By contrast, W. Gautschi $[1972]_2$ has shown that the condition of the Chebyshev form is no larger than $n\sqrt{2}$. Finally, the condition of the Lagrange form (4) is easily seen to be the number $\|\lambda_n\|$, with

$$\lambda_n(x) := \sum_{i=1}^{n} |\ell_i(x)|.$$

As is pointed out in the next chapter,

$$\|\lambda_n\| \sim (2/\pi) \ln n + 1$$

if we choose the Chebyshev sites II(9) as the sites τ. Thus, the Lagrange form based on the Chebyshev sites is about the best conditioned polynomial form available.

Nevertheless, we will use the local power form throughout this book because of its simplicity, the easy access to derivatives it affords, and because we do not intend to deal with polynomials of order higher than 15 or 20. Still, the reader should be aware of this potential source of trouble and be prepared to switch to some better conditioned form, for example the Chebyshev form, if need be. For this, see Problems 7 and II.3.

Problems

1. Given the divided difference table for the polynomial $p \in \Pi_{<6}$, based on the sites $\tau_1, \ldots, \tau_6$, locate in it the coefficients for the Newton form for p using the centers (a) $\tau_3, \tau_4, \tau_2, \tau_5, \tau_1, \tau_6$; (b) $\tau_6, \tau_5, \tau_4, \tau_3, \tau_2, \tau_1$; (c) $\tau_4, \tau_5, \tau_6, \tau_3, \tau_2, \tau_1$; (d) $\tau_4, \tau_5, \tau_6, \tau_3, \tau_2, \tau_2$.

2. Assume that $\texttt{TAU}(i) = \tau_i$, $\texttt{D}(i) = g(\tau_i)$, $i = 1, \ldots, n$.

(a) Verify that, after the execution of the following statements,

```
1   for k = 1,...,n-1, do:
    1/1 for i = 1,...,n-k, do:
        1/1/1 D(i) := (D(i+1) - D(i))/(TAU(i+k) - TAU(i))
```

we have $\texttt{D}(i) = [\tau_i, \ldots, \tau_n]g$ for $i = 1, \ldots, n$.

(b) Hence, verify that the subsequent calculation

```
2   value := D(1)
3   for k = 2,...,n, do:
    3/1 value := D(k) + value*(x - TAU(k))
```

produces the value $p_n(x)$ at x of the polynomial p_n of order n that agrees with g at τ.

(c) Finally, verify that the subsequent calculation

```
4   for k = 2,...,n, do:
    4/1 for i = 2,...,n+2-k, do:
        4/1/1 D(i) := D(i) + D(i-1)*(x - TAU(i+k-2))
```

produces the coefficients in the Taylor expansion around x for p_n, that is,

$$\texttt{D}(i) = p_n^{(n-i)}(x)/(n-i)!, \qquad i = 1, \ldots, n.$$

3. Let p_4 be the polynomial of order 4 that agrees with the function ln at 1, 2, 3, 4. Prove that $p_4(x) > \ln(x)$ for $2 < x < 3$.

4. Construct the polynomial p_6 of order 6 that agrees with the function sin at 0, 0, $\pi/4$, $\pi/4$, $\pi/2$, $\pi/2$. Use the error formula (15) along

with the divided difference Property (vii) to estimate the maximum error $\max\{\,|\sin(x) - p_6(x)| : 0 \le x \le \pi/2\,\}$ and compare it with the maximum error found by checking the error at 50 sites, say, in the interval $[0\,..\,\pi/2]$.

5. Prove that $[\tau_1, \ldots, \tau_n](1/x) = (-1)^{n-1}/(\tau_1 \cdots \tau_n)$. (Hint: Use Leibniz' formula on the product $(1/x)x$, and induction.)

6. Prove: If $p(x) = \sum_1^n a_i \left(\frac{x-a}{b-a}\right)^{i-1}$, and all a_i are of the same sign, then the relative change $\|\delta p\|/\|p\|$ (on $[a\,..\,b]$) caused by a change $\delta \mathbf{a}$ in the coefficient vector $\mathbf{a}$ is no bigger than $n\|\delta \mathbf{a}\|/\|\mathbf{a}\|$.

Conclude that the condition as defined in the text measures only the worst possible case. Also, develop a theory of "condition at a point" that parallels and refines the overall condition discussed in the text.

7. The Chebyshev form for $p \in \Pi_{<n}$ is specified by the end points a and b of some interval and the coefficients $\alpha_1, \ldots, \alpha_n$ for which $p(x) = \sum_1^n \alpha_i \mathrm{T}_{i-1}(y)$, with $y = y(x) := (2x - (b+a))/(b-a)$ and $\mathrm{T}_0(x) := 1$, $\mathrm{T}_1(x) := x$, $\mathrm{T}_{i+1}(x) := 2x\mathrm{T}_i(x) - \mathrm{T}_{i-1}(x)$, $i = 1, 2, 3, \ldots$. Write a `FUNCTION CHBVAL(CCOEF, NTERM, X)` that has `CCOEF` $= (a, b, \alpha_1, \ldots, \alpha_n)$, `NTERMS` $= n + 2$, and argument `X` as input, and returns the value p(`X`), using the three-term recurrence in the process. See, for example, `FUNCTION CHEB`, Conte & de Boor [1980:p. 258].

8. Prove (for example by induction) that for arbitrary nondecreasing $\tau = (\tau_1, \ldots, \tau_n)$, there exist unique constants $d_1, \ldots, d_n$, so that the linear functional $[\tau_1, \ldots, \tau_n]$ can be written

$$[\tau_1, \ldots, \tau_n] = \sum_{i=1}^{n} d_i [\tau_{m(i)}, \ldots, \tau_i],$$

with

$$m(i) := \min\{j \le i : \tau_j = \tau_i\}, \quad \text{all } i.$$

In other words, the divided difference of g at τ is uniquely writeable as a weighted sum of values and certain of the derivatives of g at the τ_i.

9. Prove Micchelli's observation that $[x_0, \ldots, x_j](\cdot - x_0)f = [x_1, \ldots, x_j]f$.

10. Prove Micchelli's handy (see Problem 5) observation that a divided difference identity is valid for all (smooth enough) f if it is known to hold for all f of the form $f(x) = 1/(x-a)$, $a \in \mathbb{R}$.

II

Limitations of Polynomial Approximation

In this chapter, we show that the polynomial interpolant is very sensitive to the choice of interpolation sites. We stress the fact that polynomial interpolation at appropriately chosen sites (for example, the Chebyshev points) produces an approximation that, for all practical purposes, differs very little from the best possible approximant by polynomials of the same order. This allows us to illustrate the essential limitation of polynomial approximation: If the function to be approximated is badly behaved *any*where in the interval of approximation, then the approximation is poor *every*where. This global dependence on local properties can be avoided when using *piecewise* polynomial approximants.

Uniform spacing of data can have bad consequences

We cite the

(1) Runge example: Consider the polynomial p_n of order n that agrees with the function $g(x) := 1/(1+25x^2)$ at the following n uniformly spaced sites in $[-1\,..\,1]$:

$$\tau_i := (i-1)h - 1, \qquad i = 1, \ldots, n, \quad \text{with } h := 2/(n-1).$$

The function g being without apparent defect (after all, g is even analytic in a neighborhood of the interval of approximation $[-1\,..\,1]$), we would expect the maximum error

$$\|e_n\| := \max_{-1 \le x \le 1} |g(x) - p_n(x)|$$

to decrease toward zero as n increases. In the program below, we estimate the maximum error $\|e_n\|$ by the maximum value obtained when evaluating the absolute value of $g - p_n$ at 20 sites in each of the $n-1$ intervals $[\tau_{i-1}\,..\,\tau_i]$, $i = 2, \ldots, n$. In this and later examples, we anticipate that, as a function of n, $\|e_n\|$ decreases to zero like βn^{α} for some constant β and some (negative)

constant α. If $\|e_n\| \sim \beta n^\alpha$, then $\|e_n\|/\|e_m\| \sim (n/m)^\alpha$, and we can estimate the **decay exponent** α from two errors $\|e_n\|$ and $\|e_m\|$ by

$$\alpha \sim \frac{\log(\|e_n\|) - \log(\|e_m\|)}{\log(n/m)}. \tag{2}$$

In the following program, the decay exponent is estimated in this way from successive maximum errors.

```
CHAPTER II. RUNGE EXAMPLE
      INTEGER I,ISTEP,J,K,N,NMK,NM1
      REAL ALOGER,ALGERP,D(20),DECAY,DX,ERRMAX,G,H,PNATX,STEP,TAU(20),X
      DATA STEP, ISTEP /20., 20/
      G(X) = 1./(1.+(5.*X)**2)
      PRINT 600
  600 FORMAT(28H  N    MAX.ERROR    DECAY EXP.//)
      DECAY = 0.
      DO 40 N=2,20,2
C        CHOOSE INTERPOLATION POINTS  TAU(1), ..., TAU(N) , EQUALLY
C        SPACED IN (-1 .. 1), AND SET  D(I) = G(TAU(I)), I=1,...,N.
         NM1 = N-1
         H = 2./FLOAT(NM1)
         DO 10 I=1,N
            TAU(I) = FLOAT(I-1)*H - 1.
            D(I) = G(TAU(I))
   10    CALCULATE THE DIVIDED DIFFERENCES FOR THE NEWTON FORM.
C
C
         DO 20 K=1,NM1
            NMK = N-K
            DO 20 I=1,NMK
               D(I) = (D(I+1)-D(I))/(TAU(I+K)-TAU(I))
   20
C
C        ESTIMATE MAX.INTERPOLATION ERROR ON (-1 .. 1).
         ERRMAX = 0.
         DO 30 I=2,N
            DX = (TAU(I)-TAU(I-1))/STEP
            DO 30 J=1,ISTEP
               X = TAU(I-1) + FLOAT(J)*DX
               EVALUATE INTERP.POL. BY NESTED MULTIPLICATION
C
C
               PNATX = D(1)
               DO 29 K=2,N
                  PNATX = D(K) + (X-TAU(K))*PNATX
   29
C
               ERRMAX = AMAX1(ERRMAX,ABS(G(X)-PNATX))
   30    ALOGER = ALOG(ERRMAX)
         IF (N .GT. 2)  DECAY =
     *      (ALOGER - ALGERP)/ALOG(FLOAT(N)/FLOAT(N-2))
         ALGERP = ALOGER
   40    PRINT 640,N,ERRMAX,DECAY
  640 FORMAT(I3,E12.4,F11.2)
                                                    STOP

      END
```

N	MAX.ERROR	DECAY EXP.
2	0.9615E+00	0.00
4	0.7070E+00	-0.44
6	0.4327E+00	-1.21
8	0.2474E+00	-1.94
10	0.2994E+00	0.86
12	0.5567E+00	3.40
14	0.1069E+01	4.23
16	0.2099E+01	5.05
18	0.4222E+01	5.93
20	0.8573E+01	6.72

In the program listing above, we have marked the divided difference calculation and the evaluation of the Newton form to stress how simple these calculations really are.

From the output, we learn that, contrary to expectations, the interpolation error for this example actually increases with n. In fact, our estimates for the decay exponent become eventually positive and growing so that $\|e_n\|$ grows with n at an ever increasing rate. □

The following discussion provides an explanation of this disturbing situation. We denote by

$$P_n g \tag{3}$$

the polynomial of order n that agrees with a given g at the given n sites of the sequence τ. We assume that the interpolation site sequence lies in some interval $[a\,..\,b]$, and we measure the size of a continuous function f on $[a\,..\,b]$ by

$$\|f\| := \max_{a \le x \le b} |f(x)|. \tag{4}$$

Further, we recall from I(2) the Lagrange polynomials

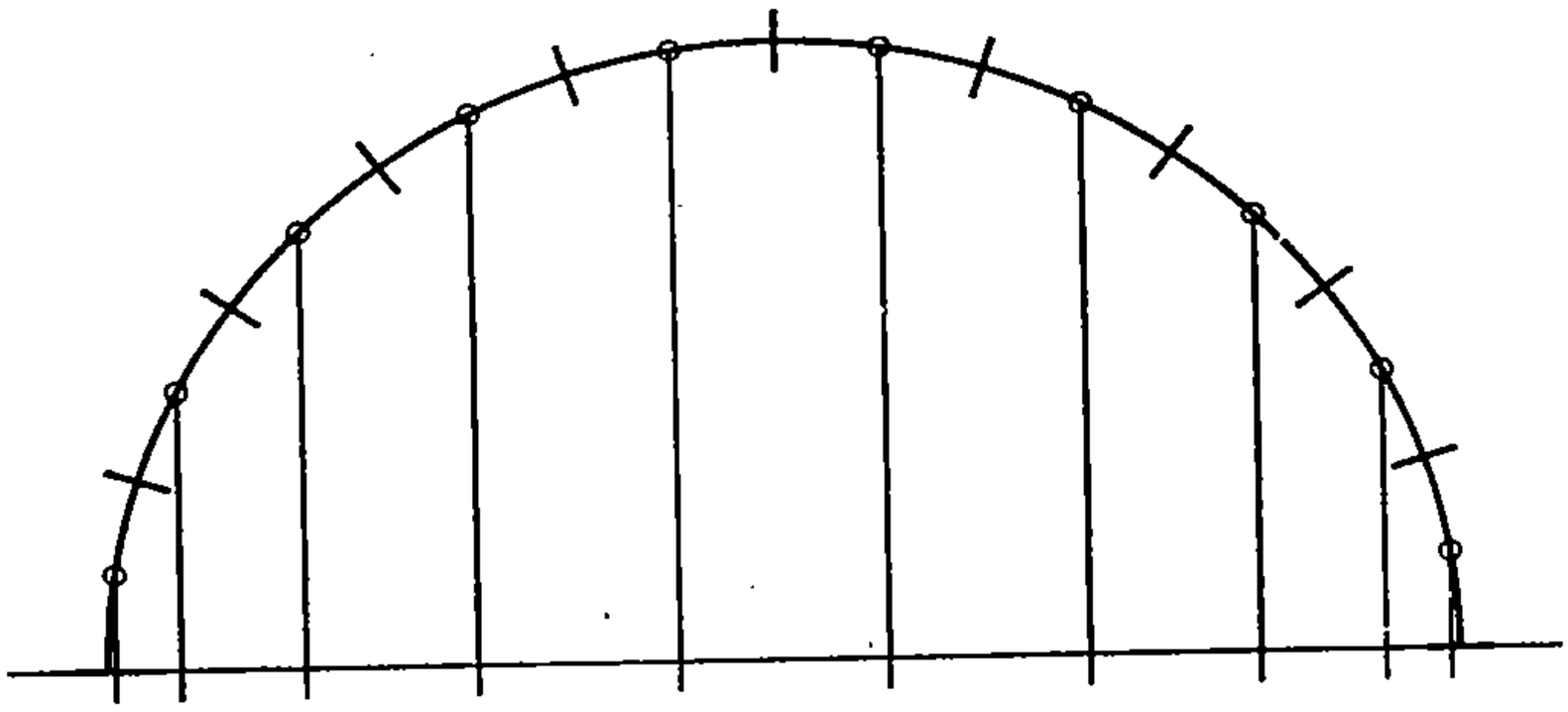
$$\ell_i(x) := \prod_{j \neq i} (x - \tau_j)/(\tau_i - \tau_j),$$

in terms of which

$$|(P_n g)(x)| = \Big|\sum_i g(\tau_i)\ell_i(x)\Big| \le \sum_i |g(\tau_i)||\ell_i(x)| \le (\max_i |g(\tau_i)|) \sum_i |\ell_i(x)|.$$

We introduce the so-called **Lebesgue function**

$$\lambda_n(x) := \sum_{i=1}^{n} |\ell_i(x)|. \tag{5}$$



(6) FIGURE. The Chebyshev sites for the interval $[a\,..\,b]$ are obtained by subdividing the semicircle over it into n equal arcs and then projecting the midpoint of each arc onto the interval.

Since $\max_i |g(\tau_i)| \le \|g\|$, we obtain the estimate

$$\|P_n g\| \le \|\lambda_n\|\,\|g\|, \tag{7}$$

and this estimate is known to be *sharp*, that is, there exist functions g (other than the zero function) for which $\|P_n g\| = \|\lambda_n\|\,\|g\|$ (see Problem 2).

On the other hand, for uniformly spaced interpolation sites,

$$\|\lambda_n\| \sim 2^n/(en\,\ln n)$$

(as first proved by A. N. Turetskii, in 1940; see the survey article L. Brutman [1997]). Hence, as n gets larger, $P_n g$ may fail entirely to approximate g for the simple reason that $\|P_n g\|$ gets larger and larger.

Chebyshev sites are good Fortunately, this situation can be remedied *if* we have the freedom to choose the interpolation sites.

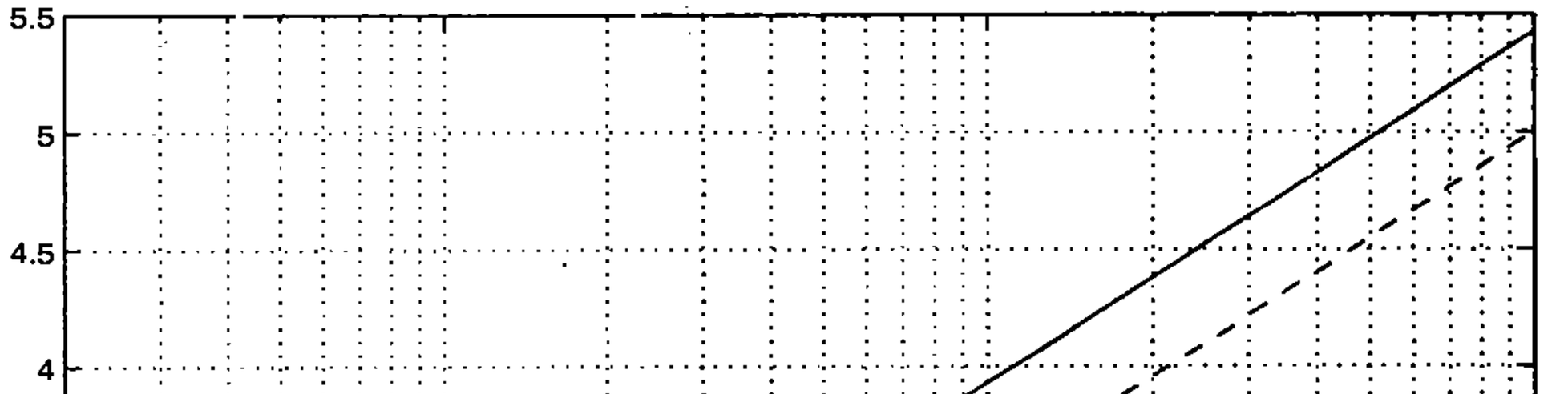
(8) Theorem. *If $\tau_1, \ldots, \tau_n$ are chosen as the zeros of the Chebyshev polynomial of degree n for the interval $[a\,..\,b]$, that is,*

$$\tau_j = \tau_j^c := \left(a + b - (a-b)\cos\left(\frac{(2j-1)\pi}{2n}\right)\right)/2, \qquad j = 1, \ldots, n, \tag{9}$$

then, with λ_n^c the corresponding Lebesgue function, we have

$$\|\lambda_n^c\| < (2/\pi)\ln n + 4.$$

For a proof, see, for example, T. Rivlin [1969:pp. 93–96]. M. J. D. Powell [1967] has computed $\|\lambda_n^c\|$ for the Chebyshev sites for various values of n



(10) FIGURE. The size of the Lebesgue function for the Chebyshev sites (solid) and the expanded Chebyshev sites (dashed).

(his $v(n)$ equals our $1 + \|\lambda_n^c\|$). Figure (10) summarizes his results. It is possible to improve on this slightly by using the **expanded** Chebyshev sites

$$(11) \quad \tau_n^e := \left(a + b - (a-b)\frac{\cos((2j-1)\pi/(2n))}{\cos(\pi/(2n))} \right)/2, \qquad j = 1, \ldots, n,$$

instead. L. Brutman [1978] has shown that

$$(2/\pi)(\ln n) + .5 < \|\lambda_n^e\| < (2/\pi)(\ln n) + .73$$

and that, for all n, $\|\lambda_n^e\|$ is within .201 of the smallest value for $\|\lambda_n\|$ possible by proper choice of τ. Numerical evidence strongly suggests that the difference between $\|\lambda_n^e\|$ and the smallest possible value for $\|\lambda_n\|$ is never more than .02.

Runge example with Chebyshev sites We change our experimental interpolation program to read in the 10–loop

$$\texttt{TAU(I)} = \texttt{COS(FLOAT(2 * I - 1) * PIOV2N)}$$

with the line `H=2./FLOAT(NM1)` before the 10-loop replaced by

$$\texttt{PIOV2N} = \texttt{3.1415926535/FLOAT(2 * N)}$$

The program now produces the output

N	MAX.ERROR	DECAY EXP.
2	0.9259+00	0.00
4	0.7503+00	-0.30
6	0.5559+00	-0.74
8	0.3917+00	-1.22
10	0.2692+00	-1.68
12	0.1828+00	-2.12
14	0.1234+00	-2.55
16	0.8311-01	-2.96
18	0.5591-01	-3.37
20	0.3759-01	-3.77

(with `MAX.ERROR` an estimate on $[\tau_1 \,..\, \tau_n]$ only). □

This is quite satisfactory, so let's take something tougher.

(12) Squareroot example We take $g(x) := \sqrt{1+x}$ on $[-1\,..\,1]$, that is, we modify our experimental interpolation program further by changing line 5, from `G(X) = 1./(1. + (5.*X)**2)` to

$$\texttt{G(X)} = \texttt{SQRT(1.} + \texttt{X)}$$

This produces the output

N	MAX.ERROR	DECAY EXP.
2	0.7925-01	0.00
4	0.3005-01	-1.40
6	0.1905-01	-1.12
8	0.1404-01	-1.06
10	0.1114-01	-1.04
12	0.9242-02	-1.02
14	0.7900-02	-1.02
16	0.6900-02	-1.01
18	0.9425-02	2.65
20	0.1061+00	22.98

which shows the maximum error $\|e_n\|$ to decrease only like n^{-1}. This is due to the singularity of g at $x = -1$. (The increase in the error for $n = 18, 20$ is due to roundoff; see Problem 3.)

If we put the singularity inside of $[-1\,..\,1]$, things get even worse. E.g., for $g(x) = \sqrt{|x|}$ on $[-1\,..\,1]$, we get

N	MAX.ERROR	DECAY EXP.
2	0.8407+00	0.00
4	0.5475+00	-0.62
6	0.4402+00	-0.54
8	0.3791+00	-0.52
10	0.3382+00	-0.51
12	0.3083+00	-0.51
14	0.2853+00	-0.50
16	0.2666+00	-0.51
18	0.2512+00	-0.51
20	0.2383+00	-0.50

Our complaint here is that the error, while decreasing, decreases far too slowly if we are seeking several places of accuracy. As the output indicates, we have for this particular g

$$\|e_n\| \sim \text{const} \cdot n^{-1/2}.$$

Hence, if we wanted an error of 10^{-3}, then we would want

$$10^{-3} = \|e_n\| = (\|e_n\|/\|e_{20}\|)\|e_{20}\| \approx .2384(n/20)^{-1/2}$$

or

$$(n/20) \approx (.2384)^2 10^6,$$

that is, we would need $n \sim 1.14$ million. □

The following discussion might help to explain this situation. Recall from Theorem I(14) that the interpolation error satisfies

$$e_n(x) = g(x) - P_n g(x) = (x - \tau_1) \cdots (x - \tau_n)[\tau_1, \ldots, \tau_n, x]g$$

while, by Property I(vii) of the divided difference,

$$[\tau_1, \ldots, \tau_n, x]g = g^{(n)}(\zeta_x)/n!$$

for some ζ_x in the interior of the smallest interval containing $\tau_1, \ldots, \tau_n$ and x. This provides the estimate

$$\|g - P_n g\| \leq \|(\cdot - \tau_1) \cdots (\cdot - \tau_n)\| \max_{a \leq \zeta \leq b} \frac{|g^{(n)}(\zeta)|}{n!}. \tag{13}$$

Now, it so happens that

$$\min_{a \leq \tau_1 \leq \cdots \leq \tau_n \leq b} \|(\cdot - \tau_1) \cdots (\cdot - \tau_n)\| = 2\frac{(b-a)^n}{4^n} \tag{14}$$

and that this minimum is taken on when the interpolation sites are the Chebyshev sites. This means that we have done already as well as possible as far as the choice of interpolation sites is concerned. The difficulty in controlling the interpolation error then must lie with the second factor in (13),

$$\max_{a \leq \zeta \leq b} \frac{|g^{(n)}(\zeta)|}{n!}.$$

Indeed, in our case, $g^{(n)}$ becomes infinitely large in $[a\,..\,b]$ for all $n \geq 1$.

Interpolation at Chebyshev sites is nearly optimal Now, there might be some hope that we could do even better if we gave up the idea of interpolation and constructed the approximation $p \in \Pi_{<n}$ to g by some other means. Suppose, in fact, that we choose for p the *best possible* $p^* \in \Pi_{<n}$, that is,

$$\|g - p^*\| = \operatorname{dist}(g, \Pi_{<n}) := \min_{p \in \Pi_{<n}} \|g - p\|.$$

Then certainly

$$\|g - p^*\| \leq \|g - P_n g\|.$$

But, this won't gain us very much, as I now show. It is easy to see that the interpolation process is **additive**, that is,

$$P_n(g + h) = P_n g + P_n h,$$

and also $P_n = 1$ on $\Pi_{<n}$, that is,

$$P_n p = p \qquad \text{for all } p \in \Pi_{<n}.$$

This implies that

$$g - P_n g = (g - p) - P_n(g - p)$$

for all $p \in \Pi_{<n}$ and shows that we can estimate the interpolation error by

$$\|g - P_n g\| \leq \|g - p\| + \|P_n(g - p)\| \leq (1 + \|\lambda_n^c\|)\|g - p\|, \qquad \text{for all } p \in \Pi_{<n}.$$

This is valid for $p = p^*$, too, and therefore

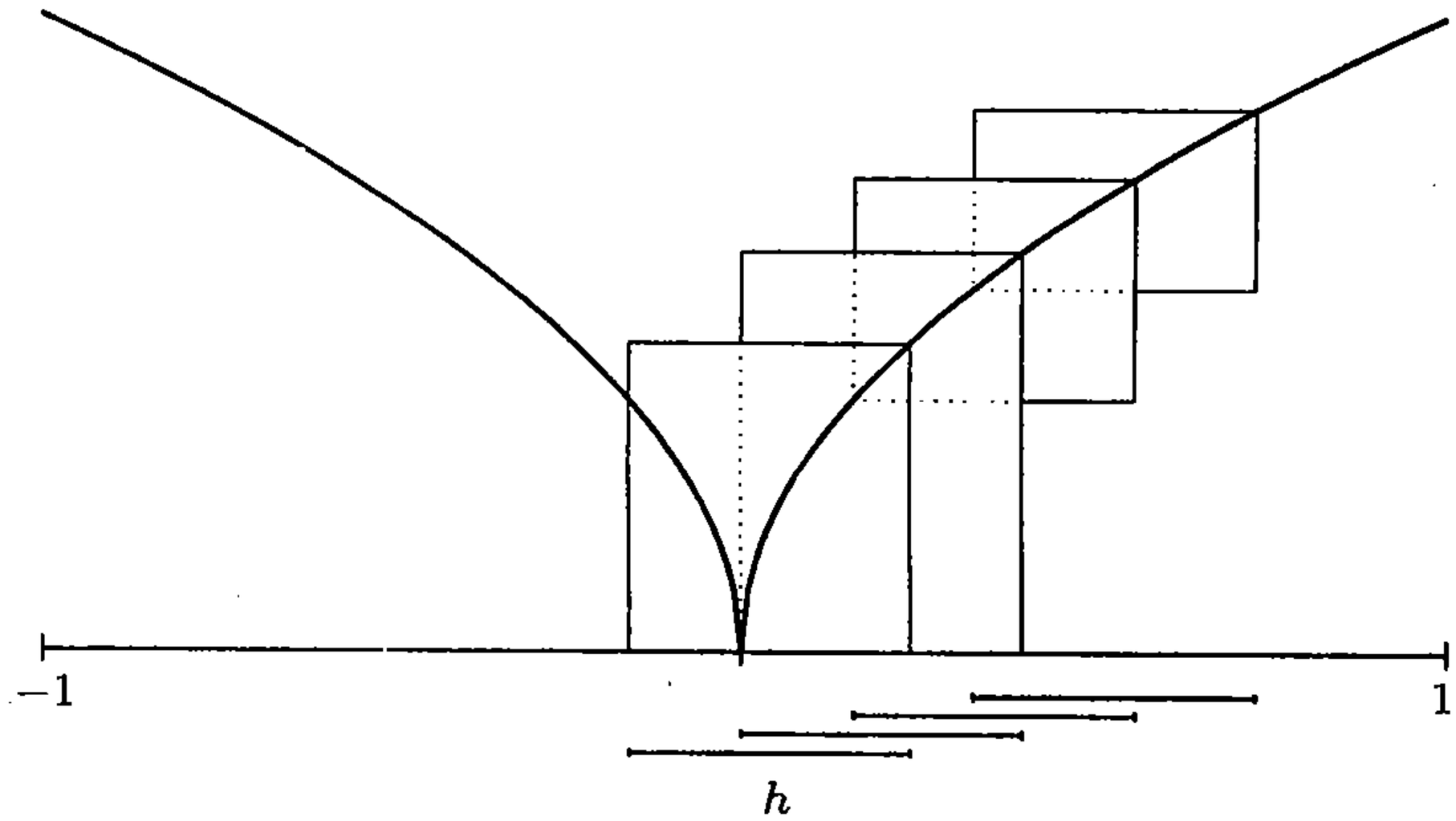
$$\operatorname{dist}(g, \Pi_{<n}) \leq \|g - P_n g\| \leq (1 + \|\lambda_n^c\|) \operatorname{dist}(g, \Pi_{<n}). \tag{15}$$

But, for $n \leq 20$, $(1 + \|\lambda_n^c\|) < 4$, by Figure (10), which says that we can hope, *at best*, to cut the error by a factor of 4 by going to some other approximation scheme.

The distance from polynomials For practical purposes, then, we might as well stick with the comparatively simple scheme of interpolation at Chebyshev sites. But this means, as the earlier example showed, that we won't be able to approximate *some* functions very well by polynomials. Also, from (13), (14), and (15), we then have the bounds

$$\frac{\|g - P_n g\|}{(1 + \|\lambda_n^c\|)} \leq \operatorname{dist}(g, \Pi_{<n}) \leq \|g - P_n g\| \leq 2\left(\frac{b-a}{4}\right)^n \frac{\|g^{(n)}\|}{n!}, \tag{16}$$

which may go to zero very slowly or not at all because $\|g^{(n)}\|$ may grow too fast with n.



(17) FIGURE. For $g(x) = \sqrt{|x|}$ on $[-1 \mathinner{..} 1]$, $\max |g(x) - g(y)|$ in a strip of width h occurs when 0 lies on the boundary of the strip, and then $\max |g(x)-g(y)| = |g(0)-g(h)| = \sqrt{h}$. Therefore, $\omega(g; h) = \sqrt{h}$.

Approximation theory provides rather precise statements about the rate at which $\operatorname{dist}(g, \Pi_{<n})$ goes to zero for various classes of functions g. An appropriate classification turns out to be one based on the number of continuous derivatives a function has and, within the class $C^{(r)}[a \mathinner{..} b]$ of functions having r continuous derivatives, on the modulus of continuity of the rth derivatives. For $g \in C[a \mathinner{..} b]$, that is, for a function g continuous on $[a \mathinner{..} b]$, the **modulus of continuity of g at h** is denoted by $\omega(g; h)$ and is defined by

$$\omega(g; h) := \sup\{\, |g(x) - g(y)| : x, y \in [a \mathinner{..} b], |x - y| \le h \,\}. \tag{18}$$

It is clear that $\omega(g; h)$ is monotone in h and **subadditive** in h (that is, $\omega(g; h + k) \le \omega(g; h) + \omega(g; k)$) and that (for a *finite* interval $[a \mathinner{..} b]$)

$$\lim_{h \to 0^+} \omega(g; h) = 0, \qquad \text{for all } g \in C[a \mathinner{..} b], \tag{19}$$

but the *rate* at which $\omega(g; h)$ approaches 0 as $h \to 0^+$ varies widely among continuous functions. The following are simple, yet practically useful, examples: The fastest possible rate for a function g that isn't identically constant is $\omega(g; h) \le \text{const}h$ (see Problem 5). This rate is achieved by all functions g with continuous first derivative, that is,

$$\text{if } g \in C^{(1)}[a \mathinner{..} b], \quad \text{then } \omega(g; h) \le \|g'\| h. \tag{20}$$

If, more generally, $\omega(g;h) \leq Kh$ for some constant K and all (positive) h, then g is said to be **Lipschitz continuous**. Piecewise continuously differentiable functions in $C[a\,..\,b]$ belong to this class which is therefore much larger than the class $C^{(1)}[a..b]$ of continuously differentiable functions on $[a..b]$. An even larger class consists of those continuous functions on $[a..b]$ that satisfy **a Hölder condition with exponent** α for some $\alpha \in (0\,..\,1)$. This means that

$$\omega(g;h) \leq Kh^{\alpha} \quad \text{for some } K \text{ and all (positive) } h.$$

For instance,

(21) $\omega(g;h) \leq h^{\alpha}$ for the function $g(x) := |x|^{\alpha}$ on $[-1\,..\,1]$.

A typical example for the use of this classification is

(22) Jackson's Theorem. *If $g \in C^{(r)}[a\,..\,b]$, that is, g has r continuous derivatives on $[a\,..\,b]$, and $n > r+1$, then*

$$\text{(23)} \qquad \text{dist}\,(g, \Pi_{<n}) \leq \text{const}_r \left(\frac{b-a}{n-1}\right)^r \omega(g^{(r)}; \frac{b-a}{2(n-1-r)}).$$

Here, the constant can be chosen as

$$\text{const}_r := 6(3e)^r/(1+r).$$

A proof of this theorem can be found, for example, in T. Rivlin [1969:p. 23].

Our last example shows that this estimate is sharp, in general, as far as the order of convergence, that is, the decay exponent, is concerned. For $g(x) = \sqrt{|x|}$ on $[-1\,..\,1]$, we have, as already noted in (21),

$$\omega(g;h) = |g(h) - g(0)| = h^{1/2}$$

and so, from (23) with $r = 0$,

$$\text{dist}\,(g, \Pi_{<n}) \leq \text{const} \cdot n^{-1/2}$$

which is the rate that we observed numerically in Example (12).

There are, of course, functions that are efficiently approximated by polynomials, namely (well behaved) analytic functions. For an analytic g,

$$\text{dist}\,(g, \Pi_{<n}) = \mathcal{O}\left(e^{-\alpha n}\right)$$

for some positive α. But, if g has only r derivatives, then (23) describes correctly how well we can approximate g by polynomials of order n.

The fact that (23) is sharp in general means that the only way for making

the error small is to make $(b-a)/(n-1)$ small. We accomplish this by letting n increase and/or making $(b-a)$ small. Since we are given $[a\,..\,b]$ in advance, we can achieve the latter only by partitioning $[a\,..\,b]$ appropriately into small intervals, on each of which we then approximate g by a suitable polynomial, that is, by using *piecewise* polynomial approximation.

Note that cutting $[a\,..\,b]$ in this way into k pieces or using polynomials of order kn will have the same effect on the bound (23) and corresponds either way to a k-fold increase in the degrees of freedom. But these degrees of freedom enter the approximation process differently. Evaluation of a polynomial of order kn involves kn coefficients and basis functions whose complexity increases with the order kn, while evaluation of a piecewise polynomial function of order n with k pieces at a site involves only n coefficients and a local basis of fixed complexity no matter how big k might be. This structural difference also strongly influences the construction of approximations, requiring the solution of a full system in the polynomial case and usually only a banded system in the piecewise polynomial case. Also, use of polynomials of order 20 or higher requires special care in the representation of such polynomials (Chebyshev expansions or some other orthogonal expansion must be used, see the discussion of condition in Chapter I), while use of a larger number of low order polynomial pieces is no more delicate than the use of one such polynomial. Finally, interpolation in a table, that is, to equally spaced data, by piecewise polynomials leads to no difficulties, in contrast to what happens when just one polynomial is used, as we saw with the Runge example (1).

For these and other reasons, it is usually much more efficient to make $(b-a)$ small than to increase n. This is the justification for piecewise polynomial approximation.

Problems

1. Every continuous function g on $[a\,..\,b]$ has a (unique) best uniform approximation p^* from $\Pi_{<k}$ on $[a\,..\,b]$, that is, there is one and only one $p^* \in \Pi_{<k}$ so that $\|g-p^*\| = \operatorname{dist}(g, \Pi_{<k})$. This best uniform approximation p^* is characterized by **equioscillation** of the error (see, for example, Rivlin [1969: p. 26]): *Let $p \in \Pi_{<k}$ and $e := g-p$ the error in the approximation p to g. Then $\|e\| = \operatorname{dist}(g, \Pi_{<k})$ if and only if there are sites $a \le x_1 < \cdots < x_{k+1} \le b$ for which*

$$e(x_i)e(x_{i+1}) = -\|e\|^2, \qquad i = 1, \ldots, k.$$

(a) Conclude that a best approximation to g from $\Pi_{<k}$ on $[a\,..\,b]$ must interpolate g at k distinct sites (at least) in that interval.
(b) Conclude that $\operatorname{dist}(g, \Pi_{<k}) = \operatorname{const}_g g^{(k)}(\zeta)$ for some $a < \zeta < b$, in case g has k continuous derivatives, with const_g depending on the sites at which the best approximation interpolates g.

(c) Conclude that $\operatorname{dist}(g, \Pi_{<k}) \geq (2(b-a)^k/4^k) \min_{a \leq x \leq b} |g^{(k)}(x)/k!|$. (Hint: use (14) and I(14).)

2. Prove that the inequality (7) is sharp. (Hint: Pick $\hat{x}$ in $[a \,..\, b]$ for which $\lambda_n(\hat{x}) = \|\lambda\|$; such a site must exist since λ_n is continuous and positive. Then choose $\sigma_i := \operatorname{signum} \ell_i(\hat{x})$, all i, and construct a continuous function g on $[a \,..\, b]$ with $g(\tau_i) = \sigma_i$, all i, and $\|g\| = 1$, for instance by linear interpolation in, and constant extrapolation from, the table (τ_i, σ_i), $i = 1, \ldots, n$. Then show that $\|P_n g\| = \|\lambda_n\| \|g\|$.)

3. The roundoff visible in the Squareroot Example (12) could be fought by a careful ordering of the interpolation points. But, the best remedy is to construct the Chebyshev form of the interpolating polynomial. This construction is greatly facilitated by the fact that *the Chebyshev polynomials* $\mathrm{T}_0, \mathrm{T}_1, \ldots, \mathrm{T}_{n-1}$ *are orthogonal with respect to the discrete inner product*

$$\langle f, g \rangle := \sum_{i=1}^{n} f(\tau_i^*) g(\tau_i^*)$$

based on the Chebyshev sites $\tau_i^* := \cos((2i-1)\pi/(2n))$, $i = 1, \ldots, n$, for the interval $[-1 \,..\, 1]$ (see for example, Rivlin [1974]). This means that $\langle \mathrm{T}_i, \mathrm{T}_j \rangle = 0$ for $i \neq j$. Also, $\langle \mathrm{T}_i, \mathrm{T}_i \rangle = n/2$ for $i = 1, \ldots, n-1$, while $\langle \mathrm{T}_0, \mathrm{T}_0 \rangle = n$. Note that the Chebyshev sites (τ_c) for the interval, as given by (9), are related to the τ_i^* by $y(\tau_i^c) = \tau_i^*$, with

$$y(x) := (2x - (b+a))/(b-a).$$

(a) Prove that the polynomial $P_n g$ of order n that agrees with g at the Chebyshev sites (τ_i^c) can be written

$$(P_n g)(x) = \sum_{i=1}^{n} \alpha_i \mathrm{T}_{i-1}(y(x)),$$

with

$$\alpha_{i+1} := \sum_{j=1}^{n} g(\tau_j^c) \mathrm{T}_i(\tau_j^*) / \langle \mathrm{T}_i, \mathrm{T}_i \rangle, \qquad i = 1, \ldots, n.$$

(b) Develop a subroutine `CHBINT(A, B, G, N, CCOEF, NTERMS)` that returns, in `CCOEF` and `NTERMS`, the Chebyshev form for $P_n g$ (as defined in Problem I.7), given the endpoints `A`, `B` of the interval in question, the number `N` of interpolation points to be used, and a `FUNCTION G(X)` that provides function values of g on demand. Note that, with

$$v_{ji} := g(\tau_j^c) \mathrm{T}_{i-1}(\tau_j^*), \qquad \text{all } i, j,$$

one has $\alpha_1 = \sum_j v_{j,1}/N$, and

$$v_{j,i} = \begin{cases} g(\tau_j^c), & i = 1; \\ v_{j,1}\tau_j^*, & i = 2; \\ 2\tau_j^* v_{j,i-1} - v_{j,i-2}, & i > 2, \end{cases} \quad \text{all } j.$$

Also, one can forget or overwrite $v_{j,i}$ as soon as $v_{j,i+2}$ has been constructed, provided one has by then calculated $\texttt{CCOEF}(i+2) = 2(\sum_j v_{j,i})/\texttt{N}$.

Incidentally, it is also possible (for certain values of N) to make use here of the Fast Fourier Transform.

(c) Repeat the calculations in the Squareroot Example (12) above, but use CHBINT, and CHBVAL of Problem I.7, to uncover the amount of roundoff contaminating the calculations reported there.

Note: A routine like CHBINT is very useful for the construction of the Chebyshev form of a polynomial from some other form or information.

4. The calculation of $\|g\| = \max\{\, |g(x)| : a \le x \le b \,\}$ is a nontrivial task. Prove that, for $p \in \Pi_{<n}$, though, a good and cheap approximation to $\|p\|$ is provided by $\|p\|_c := \max\{\, |p(\tau_i^c)| : i = 1, \ldots, n \,\}$, with (τ_i^c) given by (9). By what number c_n must one multiply $\|p\|_c$ to ensure that $c_n\|p\|_c \ge \|p\|$?

5. Prove that g is a constant in case $\omega(g; h) = o(h)$. (Hint: Show that then g is differentiable everywhere and $g' = 0$.)

6. Prove that $\omega(g; h)$ is a monotone and subadditive function of h, that is, $\omega(g; h) \le \omega(g; h+k) \le \omega(g; h) + \omega(g; k)$ for nonnegative h and k. Conclude that $\omega(g; \alpha h) \le \lceil \alpha \rceil \omega(g; h)$ for $\alpha \ge 0$, with $\lceil \alpha \rceil := \min\{\, n \in \mathbb{Z} : \alpha \le n \,\}$.

III

Piecewise Linear Approximation

Piecewise linear approximation may not have the practical significance of cubic spline, or even higher order, approximation. But it shows most of the essential features of piecewise polynomial approximation in a simple and easily understandable setting.

Broken line interpolation We denote the piecewise linear, or **broken line**, interpolant to g at points $\tau_1, \ldots, \tau_n$ with

$$a = \tau_1 < \cdots < \tau_n = b$$

by $I_2 g$. The subscript 2 refers to the order of the polynomial pieces that make up the interpolant. The interpolant is given by

$$I_2 g(x) := g(\tau_i) + (x - \tau_i)[\tau_i, \tau_{i+1}]g \quad \text{on } \tau_i \le x \le \tau_{i+1}, \quad i = 1, \ldots, n-1. \tag{1}$$

Since

$$g(x) = g(\tau_i) + (x - \tau_i)[\tau_i, \tau_{i+1}]g + (x - \tau_i)(x - \tau_{i+1})[\tau_i, \tau_{i+1}, x]g,$$

we have, for $\tau_i \le x \le \tau_{i+1}$,

$$g(x) - I_2 g(x) = (x - \tau_i)(x - \tau_{i+1})[\tau_i, \tau_{i+1}, x]g.$$

Therefore, with $\Delta\tau_i := (\tau_{i+1} - \tau_i)$, we get the error estimate

$$|g(x) - I_2 g(x)| \le (\Delta\tau_i/2)^2 \max_{\tau_i \le \zeta \le \tau_{i+1}} |g''(\zeta)/2|$$

in case g has two continuous derivatives, hence, with $|\tau| := \max_i \Delta\tau_i$,

$$\|g - I_2 g\| \le \frac{1}{8} |\tau|^2 \|g''\|. \tag{2}$$

Clearly, we can make this error bound as small as we like simply by making $\Delta\tau_i$ small for all i. And, while this increases the number of parameters needed to describe the approximating function $f = I_2 g$, it does not really increase the complexity of f locally since, locally, f is always just a straight line.

Broken line interpolation is nearly optimal Let $\$_2$ denote the collection or linear space of all continuous broken lines on $[\tau_1 \,..\, \tau_n]$ with breaks at $\tau_2, \ldots, \tau_{n-1}$. The notation "$\$_2$" reflects the fact that these are **splines of order 2**, in the terminology to be introduced later. We claim that

$$I_2 f = f \qquad \text{for all } f \in \$_2. \tag{3}$$

Indeed, on each interval $[\tau_i \,..\, \tau_{i+1}]$, $I_2 f$ agrees with the straight line that interpolates f at τ_i and τ_{i+1}. But if $f \in \$_2$, then f itself is a straight line on $[\tau_i \,..\, \tau_{i+1}]$, therefore $I_2 f$ must be f itself on $[\tau_i \,..\, \tau_{i+1}]$, by the uniqueness of polynomial interpolation.

Further, we observe that

$$\|I_2 g\| = \max_i |(I_2 g)(\tau_i)| = \max_i |g(\tau_i)| \le \|g\|$$

therefore

$$\|I_2 g\| \le \|g\| \qquad \text{for all } g \in C[a \,..\, b]. \tag{4}$$

We combine (3) and (4) appropriately to get

$$\|g - I_2 g\| = \|(g - f) - I_2(g - f)\| \le \|g - f\| + \|g - f\| \qquad \text{for all } f \in \$_2.$$

Here, we minimize the right side of the inequality over all $f \in \$_2$ and so prove the second inequality in

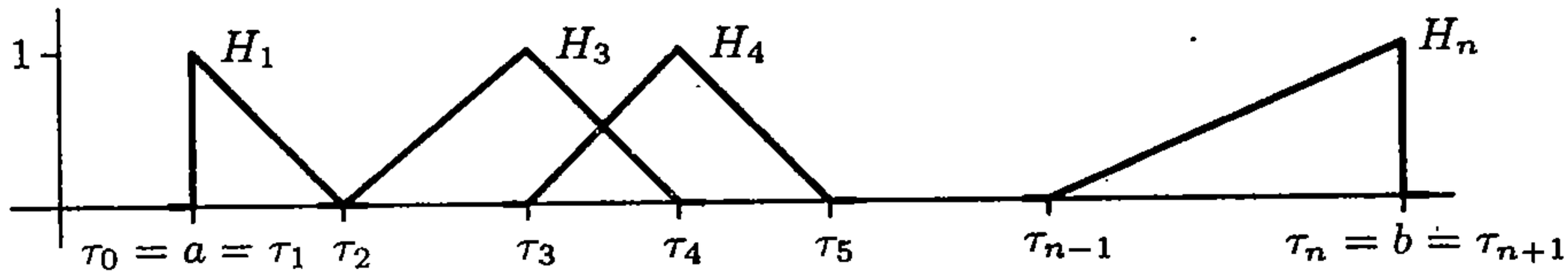
$$\operatorname{dist}(g, \$_2) \le \|g - I_2 g\| \le 2 \operatorname{dist}(g, \$_2). \tag{5}$$

This shows that we could, *at best*, halve the error by going over to a *best possible* approximation to g from $\$_2$.

Least-squares approximation by broken lines In preparation for things to come, such as least-squares approximation by splines, the error in cubic spline interpolation, and B-splines, we now discuss least-squares (or, L_2-) approximation by broken lines. For this, we need a convenient *basis* for $\$_2$.

Let $\tau_0 := \tau_1$, $\tau_{n+1} := \tau_n$, and set

$$H_i(x) := \begin{cases} (x - \tau_{i-1})/(\tau_i - \tau_{i-1}), & \tau_{i-1} < x \le \tau_i, \\ (\tau_{i+1} - x)/(\tau_{i+1} - \tau_i), & \tau_i \le x < \tau_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$



(7) FIGURE. Some of the elements of the standard basis for $\$_2$.

These functions have been called **hat functions** or, more learnedly, **chapeau functions**. Clearly, $H_i \in \$_2$, all i, and, as is obvious from Figure (7),

$$H_i(\tau_j) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}, \qquad \text{for all } i, j.$$

This shows that $\sum_i^n g(\tau_i) H_i$ is an element of $\$_2$ that agrees with g at $\tau_1, \ldots, \tau_n$, while, by (3), $I_2 g$ is the *only* element of $\$_2$ with that property. This establishes the "Lagrange form"

$$I_2 g = \sum_{i=1}^{n} g(\tau_i) H_i \tag{8}$$

for the broken line interpolant. It also implies that $(H_i)_1^n$ is a basis for $\$_2$, that is, every broken line on $[\tau_1 \,..\, \tau_n]$ with breaks at $\tau_2, \ldots, \tau_{n-1}$ can be written in exactly one way as a linear combination of the H_i's. The *coordinates* of a given $f \in \$_2$ with respect to the basis $(H_i)_1^n$ consist simply of its values $(f(\tau_1), \ldots, f(\tau_n))$ at the breaks, that is,

$$f = \sum_{i=1}^{n} f(\tau_i) H_i \qquad \text{for all } \; f \in \$_2. \tag{9}$$

Let $L_2 g$ be the *least-squares approximation to g from* $\$_2$, that is, $L_2 g \in \$_2$ and

$$\int |g(x) - L_2 g(x)|^2 \, \mathrm{d}x = \min_{f \in \$_2} \int |g(x) - f(x)|^2 \, \mathrm{d}x \, .$$

We determine $L_2 g$ with the aid of the normal equations; that is, we find a minimum of

$$\int |g(x) - \sum_{j=1}^{n} \alpha_j H_j(x)|^2 \, \mathrm{d}x$$

by setting its first partial derivatives with respect to $\alpha_1, \ldots, \alpha_n$ to zero. This gives the linear system

$$\sum_{j=1}^{n} \left[\int H_i(x) H_j(x) \, \mathrm{d}x \right] \alpha_j = \int H_i(x) g(x) \, \mathrm{d}x, \qquad i = 1, \ldots, n, \tag{10}$$

for the coefficients $(\alpha_i)_1^n$ of the broken-line $\mathbf{L}_2$-approximation to g,

$$L_2 g =: \sum_{j=1}^{n} \alpha_j H_j.$$

The integrals involved are easily evaluated. We get, more explicitly, the linear system

$$\begin{aligned}(\tfrac{\Delta\tau_{i-1}}{6})\alpha_{i-1} + (\tfrac{\tau_{i+1}-\tau_{i-1}}{3})\alpha_i + (\tfrac{\Delta\tau_i}{6})\alpha_{i+1} &= \\ = \beta_i := \int H_i(x)g(x)\,\mathrm{d}x \qquad & i = 1,\ldots,n,\end{aligned} \tag{11}$$

(with α_0, α_{n+1} of no interest since $\Delta\tau_0 = \Delta\tau_n = 0$). The coefficient matrix is tridiagonal, and strictly diagonally dominant. The system has therefore exactly one solution and this solution can be obtained efficiently and stably by Gauss elimination *without* pivoting (see, for example, Forsythe & Moler [1967]).

(12) Theorem. *The* $\mathbf{L}_2$*-approximation* $L_2 g$ *to* $g \in C[a\,..\,b]$, *that is, to a continuous function* g *on* $[a\,..\,b]$, *by elements of* $\$_2$ *satisfies*

$$\|L_2 g\| \le 3\|g\|. \tag{13}$$

Hence, since L_2 *is additive and* $L_2 f = f$ *for all* $f \in \$_2$, *we have*

$$\|g - L_2 g\| \le 4\,\mathrm{dist}\,(g, \$_2). \tag{14}$$

PROOF. To bound $\|L_2 g\| = \max_i |(L_2 g)(\tau_i)| = \max_i |\alpha_i|$ in terms of $\|g\|$, we multiply, for each i, both sides of the ith equation in (11) by $6/(\tau_{i+1} - \tau_{i-1})$ to obtain the equivalent linear system

$$\frac{\Delta\tau_{i-1}}{\tau_{i+1}-\tau_{i-1}}\alpha_{i-1} + 2\alpha_i + \frac{\Delta\tau_i}{\tau_{i+1}-\tau_{i-1}}\alpha_{i+1} = 3\hat{\beta}_i, \quad i = 1,\ldots,n, \tag{15}$$

for the coefficients (α_i) of $L_2 g$. This system is strictly diagonally dominant uniformly in τ (since $\tau_{i+1} - \tau_{i-1} = \Delta\tau_{i-1} + \Delta\tau_i$), and this we can exploit as follows: If j is such that

$$|\alpha_j| = \|\alpha\| := \max_i |\alpha_i|,$$

then, from the jth equation in (15),

$$|2\alpha_j| = |3\hat{\beta}_j - (\alpha_{j-1}\Delta\tau_{j-1} + \alpha_{j+1}\Delta\tau_j)/(\Delta\tau_{j-1} + \Delta\tau_j)| \le 3|\hat{\beta}_j| + |\alpha_j|$$

or,

$$\|\alpha\| \le 3|\hat{\beta}_j| \le 3\|\hat{\beta}\|.$$

On the other hand,

$$\hat{\beta}_i = \int \hat{H}_i(x) g(x)\,\mathrm{d}x$$

with

$$\hat{H}_i(x) := (2/(\tau_{i+1} - \tau_{i-1}))H_i(x) \tag{16}$$

positive for $\tau_{i-1} < x < \tau_{i+1}$ and zero otherwise, and $\int \hat{H}_i(x)\,\mathrm{d}x = 1$. Thus $|\hat{\beta}_i| = |\int \hat{H}_i(x)g(x)\,\mathrm{d}x| \leq \int \hat{H}_i(x)\,\mathrm{d}x \cdot \max\{\,|g(x)| : \tau_{i-1} \leq x \leq \tau_{i+1}\,\} \leq \|g\|$.

This proves that $\|L_2 g\| \leq 3\|g\|$. But (14) follows from this and from the fact that $L_2 f = f$, for all $f \in \$_2$, and $L_2(f+g) = L_2 f + L_2 g$, much as (5) follows from (3) and (4) (see Problem 2). □

The use of diagonal dominance to bound the solution of a linear system in terms of the right side has been standard in Numerical Analysis for some time. But the first spline paper to employ the technique is Sharma & Meir [1966] in which Theorem (12) is proved this way (modulo the identification made in Corollary V(9)).

Good meshes

We observed earlier that, with $|\tau| := \max_i |\Delta\tau_i|$,

$$\|g - I_2 g\| \leq \frac{1}{8}|\tau|^2 \|g''\| \tag{2}$$

in case $g \in C^{(2)}$ and concluded that we could make the interpolation error arbitrarily small by making $|\tau|$ small. In particular, for a *uniform* mesh, we get

$$\|g - I_2 g\| \leq \frac{1}{8}\left(\frac{b-a}{n-1}\right)^2 \|g''\| = \mathcal{O}\left(n^{-2}\right), \tag{17}$$

with n, the number of interpolation points, equal to the degrees of freedom used. Here, we have used the symbol "$\mathcal{O}\left(n^{-2}\right)$" (read "big oh of n^{-2}") to express the fact that the decay exponent for the bound in (17) is at least as small as -2. See p. xvi for a full explanation.

Even if g is only continuous, for example g'' might be infinite somewhere, or g' is not defined somewhere, we can still make the error small by making $|\tau| = \max_i \Delta\tau_i$ small. On $\tau_i \leq x \leq \tau_{i+1}$, we have

$$(I_2 g)(x) = \big((\tau_{i+1} - x)g(\tau_i) + (x - \tau_i)g(\tau_{i+1})\big)/\Delta\tau_i$$

and

$$\big((\tau_{i+1} - x) + (x - \tau_i)\big)/\Delta\tau_i = \big(|\tau_{i+1} - x| + |x - \tau_i|\big)/\Delta\tau_i = 1,$$

therefore

$$\begin{aligned}|g(x) - I_2 g(x)| &\le \big((\tau_{i+1} - x)|g(x) - g(\tau_i)| + (x - \tau_i)|g(x) - g(\tau_{i+1})|\big)/\Delta\tau_i \\ &\le \max_{\tau_i \le x \le \tau_{i+1}} \max\{\, |g(x) - g(\tau_i)|, |g(x) - g(\tau_{i+1})| \,\} \\ &\le \omega(g; \Delta\tau_i).\end{aligned}$$

Hence, by II(19),

$$\|g - I_2 g\| \le \omega(g; |\tau|) \xrightarrow[|\tau|\to 0]{} 0. \tag{18}$$

For the Squareroot Example II(12),

$$g(x) = \sqrt{|x|} \text{ on } [a\,..\,b] = [-1\,..\,1],$$

and uniformly spaced interpolation sites, this gives

$$\|g - I_2 g\| = \mathcal{O}\left(n^{-1/2}\right) \tag{19}$$

which is *no gain* in the convergence rate over polynomials (except that polynomials of order n are more complex objects than are broken lines with n pieces). But, because we have the location of the breaks at our disposal, we can do much better.

(20) Theorem. *If $g \in C^{(2)}(a\,..\,b)$, that is, if g has two continuous derivatives inside the interval $(a\,..\,b)$, and $|g''|$ is monotone near a and b, and $\int_a^b |g''(x)|^{1/2}\,dx < \infty$, then, with $\tau_2 < \cdots < \tau_{n-1}$ chosen so that*

$$\int_a^{\tau_i} |g''(x)|^{1/2}\,dx = \frac{i-1}{n-1}\int_a^b |g''(x)|^{1/2}\,dx, \qquad \text{all } i, \tag{21}$$

we have

$$\|g - I_2 g\| = \mathcal{O}\left(n^{-2}\right).$$

A simple proof of this can be found in de Boor [1973]. We will discuss such theorems for arbitrary order in Chapter XII. Note that this theorem not only claims the possibility of good convergence to functions with singularities at the ends of the interval, but it also indicates *how the breaks should be chosen* to achieve the optimal convergence rate $\mathcal{O}\left(n^{-2}\right)$.

For the Squareroot Example II(12), we consider only [0..1], for simplicity; the interval $[-1\,..\,0]$ can be handled by symmetry. For $x > 0$, $g''(x) = -(1/4)x^{-3/2}$, therefore

$$\int_0^t |g''(x)|^{1/2}\,dx = \frac{1}{2}\int_0^t x^{-3/4}\,dx = \frac{1}{2}\,4t^{1/4} < \infty.$$

We therefore want

$$2(\tau_i)^{1/4} = \frac{i-1}{n-1}2, \quad \text{or} \quad \tau_i = \left(\frac{i-1}{n-1}\right)^4, \qquad i = 2, \ldots, n.$$

For this choice, as one verifies directly (see Problem 6),

$$\|g - I_2 g\| = \max_{\tau_{n-1} \le x \le \tau_n} |(g - I_2 g)(x)| \le (n-1)^{-2}/2$$

or

$$\|g - I_2 g\| = \mathcal{O}\left(n^{-2}\right). \tag{22}$$

This shows that, for this choice of breaks, the decay exponent is -2. This is to be compared with the decay exponent of only $-1/2$ for the uniformly spaced breaks demonstrated earlier, and with the same decay exponent $-1/2$ obtained in Example II(12) when approximating g by polynomials with the same number of free parameters.

Problems

1. Verify that broken line interpolation preserves monotonicity and convexity.

2. For $g \in C[a..b]$, let Pg be an approximation to g in the linear subspace S of $C[a..b]$ so that (i) $Pg = g$ whenever $g \in S$, (ii) $P(g+h) = Pg + Ph$ for every $g, h \in C[a..b]$, and (iii) $\|P\| := \sup\{\, \|Pg\|/\|g\| : g \in C[a..b] \,\}$ is finite. Prove that then

$$\|g - Pg\| \le (1 + \|P\|)\, \mathrm{dist}\,(g, S).$$

3. Let $\$_1 := \{\, f' : f \in \$_2 \,\}$, that is, $\$_1$ is the linear space of piecewise constant functions on $[a..b]$ with breaks $\tau_2, \ldots, \tau_{n-1}$.
(a) Show that $\mathrm{dist}\,(g, \$_1) \le \omega(g; |\tau|/2)$ for $g \in C[a..b]$. (Note: We take here $\|f\| := \sup\{\, |f(x)| : x \in [a..b] \setminus \{\tau_2, \ldots, \tau_{n-1}\} \,\}$ in order to avoid discussing what the value of an $f \in \$_1$ at a break might be.)
(b) Show that $\|g' - (I_2 g)'\| \le 2\,\mathrm{dist}\,(g', \$_1)$ for any continuously differentiable g, and even $\|g' - (I_2 g)'\| = \mathcal{O}(|\tau|)$ if g' is Lipschitz continuous.
(c) Show that, for any three times continuously differentiable g, and with $\tau_{i+1/2} := (\tau_i + \tau_{i+1})/2$,

$$\max_{1 \le i < n} |(g - I_2 g)'(\tau_{i+1/2})| = \mathcal{O}\left(|\tau|^2\right).$$

4. Use the estimate $\|\hat{\alpha}\| \le 3\|\hat{\beta}\|$ obtained in the proof of Theorem (12)

to conclude that (11) has one and only one solution for given g.

5. Let $0 \le a < b$, and let f be the straight line that interpolates $g(x) := \sqrt{x}$ at a and b.
(a) Show that $\max\{\, |g(x)-f(x)| : a \le x \le b \,\} = \frac{1}{4}(\sqrt{b}-\sqrt{a})^2/(\sqrt{b}+\sqrt{a})$.
(b) Use part (a) of this problem and (5) above to prove that the distance of $g(x) := \sqrt{x}$ on $[-1\,..\,1]$ from broken lines with uniformly spaced breaks goes to zero faster than $n^{-1/2}$.

6. Use Problem 5(a) to verify (22). For this, show that for the interpolation sites $\tau_i = ((i-1)/(n-1))^4$, $i = 1, \ldots, n$, broken line interpolation to $\sqrt{x}$ on $[0\,..\,1]$ has maximum absolute error in $[\tau_i\,..\,\tau_{i+1}]$ equal to

$$\left(2 - \frac{1}{i^2 + (i-1)^2}\right) / (4(n-1)^2).$$

7. Verify numerically that $\|g - I_2 g\| = \mathcal{O}\left(n^{-2}\right)$ in case $g(x) = \sin\sqrt{x}$ and $\tau_i = ((i-1)/(n-1))^4$, $i = 1, \ldots, n$.

IV

Piecewise Cubic Interpolation; CUBSPL

Broken lines are neither very smooth nor very efficient approximators. Both for a smoother approximation and for a more efficient approximation, one has to go to piecewise polynomial approximation with higher order pieces. The most popular choice continues to be a cubic approximating function. In this chapter, we describe various schemes for piecewise cubic interpolation and give a program for cubic spline interpolation with various end conditions.

Given the data $g(\tau_1), \ldots, g(\tau_n)$ with $a = \tau_1 < \cdots < \tau_n = b$, we construct a piecewise cubic interpolant f to g as follows. On each interval $[\tau_i \,..\, \tau_{i+1}]$, we have f agree with some polynomial P_i of order 4,

(1)
$$f(x) = P_i(x) \quad \text{for } \tau_i \le x \le \tau_{i+1} \text{ for some } P_i \in \Pi_{<4}, \quad i = 1, \ldots, n-1.$$

The ith polynomial piece P_i is made to satisfy the conditions

(2)
$$\begin{aligned} P_i(\tau_i) &= g(\tau_i), \quad & P_i(\tau_{i+1}) &= g(\tau_{i+1}) \\ P_i'(\tau_i) &= s_i, & P_i'(\tau_{i+1}) &= s_{i+1} \end{aligned} \qquad i = 1, \ldots, n-1.$$

Here, $s_1, \ldots, s_n$ are free parameters. The resulting piecewise cubic function f agrees with g at $\tau_1, \ldots, \tau_n$ and is in $C^{(1)}[a \,..\, b]$, that is, is continuous and has a continuous first derivative on $[a \,..\, b]$, regardless of how we choose the free "slopes" $(s_i)_1^n$.

In order to compute the coefficients of the ith polynomial piece P_i, we use its Newton form

(3)
$$\begin{aligned} P_i(x) = P_i(\tau_i) &+ (x - \tau_i)[\tau_i, \tau_i]P_i + (x - \tau_i)^2[\tau_i, \tau_i, \tau_{i+1}]P_i \\ &+ (x - \tau_i)^2(x - \tau_{i+1})[\tau_i, \tau_i, \tau_{i+1}, \tau_{i+1}]P_i. \end{aligned}$$

We determine its coefficients from a divided difference table for P_i based on the data (2) for P_i:

$$\begin{array}{llll} [\,]P & [\,,\,]P & [\,,\,,\,]P & [\,,\,,\,,\,]P \\ \hline \tau_i \quad g(\tau_i) & & & \\ & s_i & & \\ \tau_i \quad g(\tau_i) & & \dfrac{[\tau_i,\tau_{i+1}]g - s_i}{\Delta\tau_i} & \\ & [\tau_i,\tau_{i+1}]g & & \dfrac{s_{i+1}+s_i-2[\tau_i,\tau_{i+1}]g}{(\Delta\tau_i)^2} \\ \tau_{i+1} \quad g(\tau_{i+1}) & & \dfrac{s_{i+1}-[\tau_i,\tau_{i+1}]g}{\Delta\tau_i} & \\ & s_{i+1} & & \\ \tau_{i+1} \quad g(\tau_{i+1}) & & & \end{array}$$

This shows that, in terms of the *shifted* powers $(x-\tau_i)^r$,

$$P_i(x) = c_{1,i} + c_{2,i}(x-\tau_i) + c_{3,i}(x-\tau_i)^2 + c_{4,i}(x-\tau_i)^3 \tag{4}$$

with

(5)
$$\begin{aligned} c_{1,i} &= P_i(\tau_i) &&= g(\tau_i), \\ c_{2,i} &= P_i'(\tau_i) &&= s_i, \\ c_{3,i} &= P_i''(\tau_i)/2 &&= [\tau_i,\tau_i,\tau_{i+1}]P_i - \Delta\tau_i([\tau_i,\tau_i,\tau_{i+1},\tau_{i+1}]P_i), \\ &&&= ([\tau_i,\tau_{i+1}]g - s_i)/\Delta\tau_i - c_{4,i}\Delta\tau_i, \\ c_{4,i} &= P_i'''(\tau_i)/6 &&= (s_i + s_{i+1} - 2[\tau_i,\tau_{i+1}]g)/(\Delta\tau_i)^2. \end{aligned}$$

Different piecewise cubic interpolation schemes differ in the choice of the slopes $(s_i)_1^n$. We discuss the best known choices.

Piecewise cubic Hermite interpolation Here, one chooses $s_i = g'(\tau_i)$, all i, thereby making the approximation **local**. This means that the ith piece P_i depends only on information from, or near, the interval $[\tau_i \,..\, \tau_{i+1}]$. From Theorem I(14), we have, for $\tau_i \le x \le \tau_{i+1}$, that

$$\begin{aligned} |g(x) - f(x)| &= |(x-\tau_i)^2(x-\tau_{i+1})^2[\tau_i,\tau_i,\tau_{i+1},\tau_{i+1},x]g| \\ &\le (\Delta\tau_i/2)^4 \max_{\tau_i\le\zeta\le\tau_{i+1}} |g^{(4)}(\zeta)|/4!, \end{aligned}$$

hence, with $|\tau| := \max_i \Delta\tau_i$,

$$\|g - f\| \le (1/384)|\tau|^4 \|g^{(4)}\| \tag{6}$$

in case $g \in C^{(4)}[a \,..\, b]$. For equaliy spaced sites ($\Delta\tau_i = (b-a)/(n-1)$, all i), we get

$$\|g - f\| = \mathcal{O}\left(n^{-4}\right) \tag{7}$$

that is, *the decay exponent for the error in piecewise cubic Hermite interpolation is at least* -4 *if the function* g *has four continuous derivatives.*

(8) Runge example continued

We consider again the Runge example II(1): We interpolate $g(x) := (1+25x^2)^{-1}$ on $[-1..1]$ at $\tau_i = 1+(i-1)h$, $i = 1, \ldots, n$, with $h := 2/(n-1)$, but this time by piecewise cubic Hermite interpolation. The two major changes in the program are outlined.

```
CHAPTER IV. RUNGE EXAMPLE, WITH CUBIC HERMITE INTERPOLATION
      INTEGER I,ISTEP,J,N,NM1
      REAL ALOGER,ALGERP,C(4,20),DECAY,DIVDF1,DIVDF3,DTAU,DX,ERRMAX,G,H
     *     ,PNATX,STEP,TAU(20)
      DATA STEP, ISTEP /20., 20/
      G(X) = 1./(1.+(5.*X)**2)
      PRINT 600
  600 FORMAT(28H  N   MAX.ERROR   DECAY EXP.//)
      DECAY = 0.
      DO 40 N=2,20,2
C        CHOOSE INTERPOLATION POINTS  TAU(1), ..., TAU(N) , EQUALLY
C        SPACED IN (-1 .. 1), AND SET  C(1,I) = G(TAU(I)), C(2,I) =
C        GPRIME(TAU(I)) = -50.*TAU(I)*G(TAU(I))**2, I=1,...,N.
         NM1 = N-1
         H = 2./FLOAT(NM1)
         DO 10 I=1,N
            TAU(I) = FLOAT(I-1)*H - 1.
            C(1,I) = G(TAU(I))
   10       C(2,I) = -50.*TAU(I)*C(1,I)**2
C        CALCULATE THE COEFFICIENTS OF THE POLYNOMIAL PIECES
C
         DO 20 I=1,NM1
            DTAU = TAU(I+1) - TAU(I)
            DIVDF1 = (C(1,I+1) - C(1,I))/DTAU
            DIVDF3 = C(2,I) + C(2,I+1) - 2.*DIVDF1
            C(3,I) = (DIVDF1 - C(2,I) - DIVDF3)/DTAU
   20       C(4,I) = (DIVDF3/DTAU)/DTAU
C
C        ESTIMATE MAX.INTERPOLATION ERROR ON (-1 .. 1).
         ERRMAX = 0.
         DO 30 I=2,N
            DX = (TAU(I)-TAU(I-1))/STEP
            DO 30 J=1,ISTEP
               H = FLOAT(J)*DX
C              EVALUATE (I-1)ST CUBIC PIECE
C
               PNATX = C(1,I-1)+H*(C(2,I-1)+H*(C(3,I-1)+H*C(4,I-1)))
C
   30          ERRMAX = AMAX1(ERRMAX,ABS(G(TAU(I-1)+H)-PNATX))
         ALOGER = ALOG(ERRMAX)
         IF (N .GT. 2)  DECAY =
     *       (ALOGER - ALGERP)/ALOG(FLOAT(N)/FLOAT(N-2))
         ALGERP = ALOGER
   40    PRINT 640,N,ERRMAX,DECAY
  640 FORMAT(I3,E12.4,F11.2)
                                          STOP
      END
```

N	MAX.ERROR	DECAY EXP.
2	0.9601+00	0.00
4	0.7275+00	-0.40
6	0.4900+00	-0.97
8	0.3289+00	-1.39
10	0.2286+00	-1.63
12	0.1656+00	-1.77
14	0.1244+00	-1.86
16	0.9640-01	-1.91
18	0.7669-01	-1.94
20	0.6234-01	-1.97

N	MAX.ERROR	DECAY EXP.
2	0.9246+00	0.00
4	0.5407+00	-0.77
6	0.2500+00	-1.90
8	0.1141+00	-2.73
10	0.5562-01	-3.22
12	0.2932-01	-3.51
14	0.1661-01	-3.69
16	0.1000-01	-3.80
18	0.6339-02	-3.87
20	0.4195-02	-3.92

The resulting output is shown above, to the right. The output on the left came from an incorrect first run, and I decided to include it here to show the usefulness of computing decay exponents. I knew I must have made a mistake since the output showed only $\mathcal{O}\left(n^{-2}\right)$ convergence in contrast to (7). As it turned out, I had used -2 instead of -50 in the statement labeled 10 in the program, hence had computed the slopes of g incorrectly; see Problem 3(e).

Piecewise cubic Bessel interpolation Here, one chooses s_i as the slope at τ_i of the polynomial p of order 3 that agrees with g at $\tau_{i-1}, \tau_i, \tau_{i+1}$. A short calculation gives

$$s_i = \frac{\Delta\tau_i[\tau_{i-1},\tau_i]g + \Delta\tau_{i-1}[\tau_i,\tau_{i+1}]g}{\Delta\tau_i + \Delta\tau_{i+1}} \tag{9}$$

which shows that Bessel interpolation is also a *local* method. Bessel interpolation provides only an $\mathcal{O}\left(|\tau|^3\right)$-approximation since s_i, as given by (9), is only an $\mathcal{O}\left(|\tau|^2\right)$-approximation to $g'(\tau_i)$; see Problems 3 and 4 (or try it right now for $g := (\cdot - \tau_{i-1})(\cdot - \tau_i)(\cdot - \tau_{i+1})$).

Akima's interpolation Akima's interpolation was developed by Akima [1970] in order to combat wiggles in the interpolant. It, too, is a *local* method, but it is *not* additive, that is, the interpolant to a sum of two functions is, in general, different from the sum of the corresponding interpolants. Akima chooses

$$s_i = \frac{w_{i+1}[\tau_{i-1},\tau_i]g + w_{i-1}[\tau_i,\tau_{i+1}]g}{w_{i+1} + w_{i-1}}, \tag{10}$$

with

$$w_j := |[\tau_j,\tau_{j+1}]g - [\tau_{j-1},\tau_j]g|. \tag{11}$$

The approximation provided by this scheme is, in general, only of order $\mathcal{O}\left(|\tau|^2\right)$; see Problems 3 and 5. The subroutine IQHSCU in the IMSL subroutine library [1977] embodies this scheme.

Cubic spline interpolation Here, the free slopes $s_2, \ldots, s_{n-1}$ are determined from the condition that f should be *twice* continuously differentiable, that is, so that f has also a continuous second derivative. This gives the conditions that, for $i = 2, \ldots, n-1$,

$$P''_{i-1}(\tau_i) = P''_i(\tau_i)$$

or

$$2c_{3,i-1} + 6c_{4,i-1}\Delta\tau_{i-1} = 2c_{3,i}$$

or

$$\frac{2([\tau_{i-1}, \tau_i]g - s_{i-1})}{\Delta\tau_{i-1}} + 4c_{4,i-1}\Delta\tau_{i-1} = \frac{2([\tau_i, \tau_{i+1}]g - s_i)}{\Delta\tau_i} - 2c_{4,i}\Delta\tau_i,$$

leading to the linear system

$$\begin{aligned} (12) \qquad & s_{i-1}\Delta\tau_i + s_i 2(\Delta\tau_{i-1} + \Delta\tau_i) + s_{i+1}\Delta\tau_{i-1} = \\ & b_i := 3(\Delta\tau_i[\tau_{i-1}, \tau_i]g + \Delta\tau_{i-1}[\tau_i, \tau_{i+1}]g), \qquad i = 2, \ldots, n-1. \end{aligned}$$

Assuming that the endslopes, s_1 and s_n, have been chosen somehow (see below), we now have in (12) a tridiagonal linear system of $n-2$ equations for the $n-2$ unknowns $s_2, \ldots, s_{n-1}$ which is strictly row diagonally dominant. Hence the system has exactly one solution, and this solution can be found without any difficulty by Gauss elimination *without* pivoting (see Forsythe & Moler [1967]).

Boundary conditions The various choices for s_1 and s_n about to be discussed are useful for piecewise cubic Bessel interpolation, Akima's interpolation scheme and similar schemes as well as for cubic spline interpolation. But we discuss them only in terms of spline interpolation.

(i) If g' is known at τ_1 and τ_n, then it is natural to choose $s_1 = g'(\tau_1)$ and $s_n = g'(\tau_n)$. The resulting spline interpolant

$$f = I_4 g$$

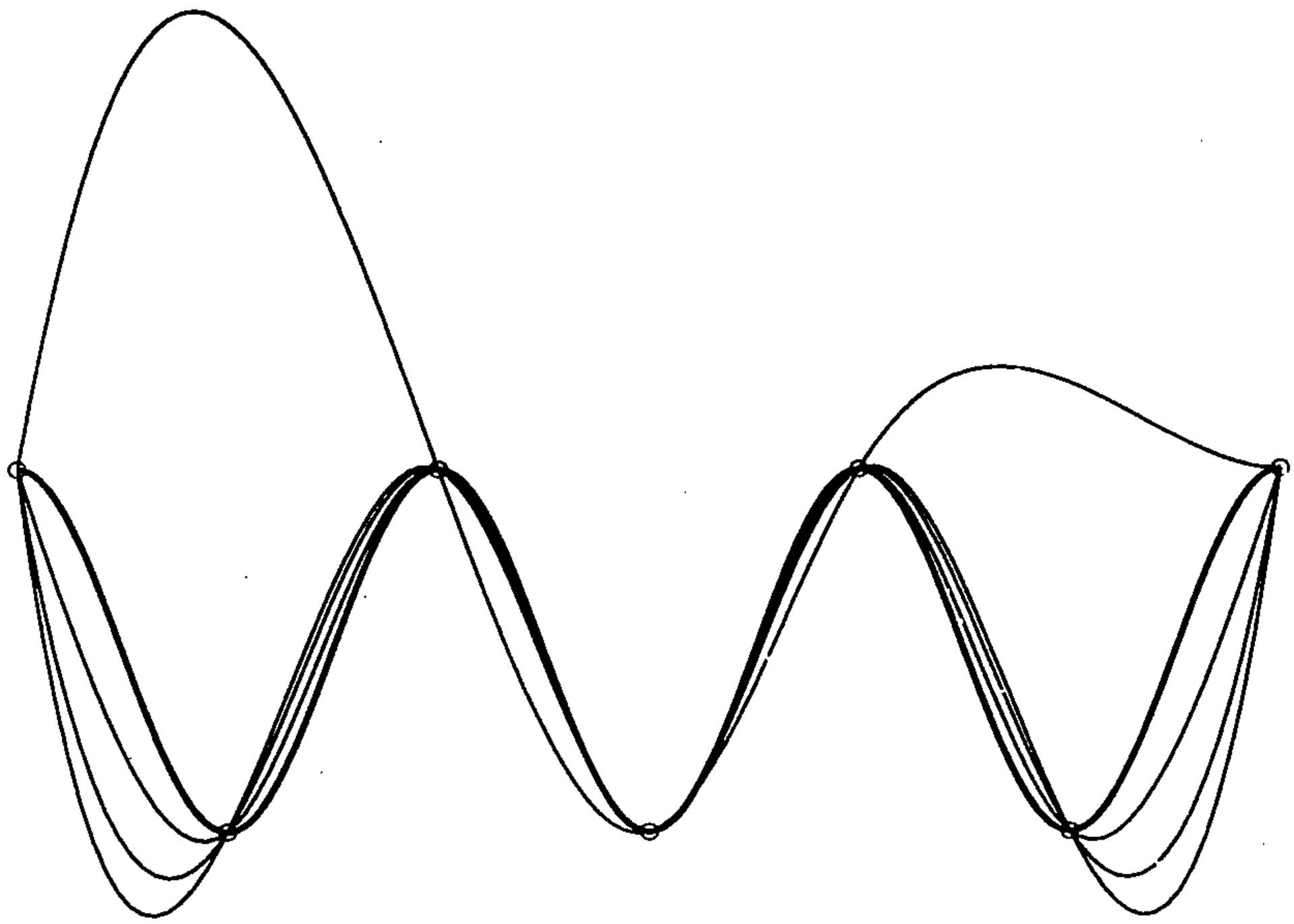
agrees with g at $\tau_0, \ldots, \tau_{n+1}$ (with $\tau_0 := \tau_1$, $\tau_{n+1} := \tau_n$) and has been called the **complete** cubic spline interpolant of g.

(ii) If g'' is known at the end points, then one can force $f'' = g''$ at the end points by adding the equation

$$(13) \qquad 2s_1 + s_2 = 3[\tau_1, \tau_2]g + (\Delta\tau_1)g''(\tau_1)/2$$

at the beginning of (12) and appending the corresponding equation

$$(14) \qquad s_{n-1} + 2s_n = 3[\tau_{n-1}, \tau_n]g + (\Delta\tau_{n-1})g''(\tau_n)/2.$$



(15) FIGURE. Cubic spline interpolation to $f(x) = \cos(x)$ at $\tau = ((i-1)\pi : i-1,\dots,7)$ and with various end conditions. Going from top to bottom at either end, the conditions used are: 1. match second (first) derivative at left (right) end (but don't interpolate to second nor second-last data point); 2. periodic, complete (both indistinguishable from f); 3. free-end or natural (note the straight ends); 4. first end derivatives estimated by local cubic interpolation; 5. not-a-knot.

(iii) So-called *natural* spline interpolation results from the **free-end** conditions

$$f''(\tau_1) = f''(\tau_n) = 0. \tag{16}$$

In spite of its positive sounding name, natural spline interpolation has little to recommend it from an approximation-theoretic point of view. The arbitrary assignment (16) produces $\mathcal{O}(|\tau|^2)$-errors near the ends (unless also $g''(\tau_1) = g''(\tau_n) = 0$) and so dramatically reduces the overall rate of convergence of the interpolation method.

(iv) If one knows nothing about the end point derivatives, then one should try the **not-a-knot** condition (from de Boor [1966]). Here, one chooses s_1 and s_n so that $P_1 = P_2$ and $P_{n-2} = P_{n-1}$ (that is, the first and the last interior knots are not active). This requires that f''' be continuous across

τ_2 and τ_{n-1} and so means adding the equation

$$s_1\Delta\tau_2 + s_2(\tau_3 - \tau_1) = \frac{(\Delta\tau_1 + 2(\tau_3 - \tau_1))\Delta\tau_2[\tau_1,\tau_2]g + (\Delta\tau_1)^2[\tau_2,\tau_3]g}{\tau_3 - \tau_1} \tag{17}$$

at the beginning of (12) and appending the corresponding equation

$$s_{n-1}(\tau_n - \tau_{n-2}) + s_n\Delta\tau_{n-2} = \frac{(\Delta\tau_{n-1})^2[\tau_{n-2},\tau_{n-1}]g + (2(\tau_n - \tau_{n-2}) + \Delta\tau_{n-1})\Delta\tau_{n-2}[\tau_{n-1},\tau_n]g}{\tau_n - \tau_{n-2}} \tag{18}$$

One may alternatively think of this boundary condition as requiring both the first and the last polynomial piece to interpolate g at an additional site that is not a break. This means that we have $n-3$ polynomial pieces rather than $n-1$, with the first piece P_1 agreeing with f on $[\tau_1 \,..\, \tau_3]$ and such that $P_1(\tau_i) = g(\tau_i)$, $i = 1,2,3$, and $P_1'(\tau_3) = s_3$, and, similarly, the last piece, P_{n-3}, making up f on $[\tau_{n-2} \,..\, \tau_n]$ and such that $P_{n-3}(\tau_i) = g(\tau_i)$ for $i = n-2, n-1, n$ and $P_{n-3}'(\tau_{n-2}) = s_{n-2}$. This would change the linear system (and the notation) slightly, but the resulting function f would be identical with the one constructed the earlier way.

Interpreted this way, we have here the first illustration of the fact that, in piecewise polynomial interpolation, *interpolation sites and breaks need not coincide.*

(v) A somewhat different technique in the absence of derivative information at the boundary consists in computing an estimate for g', g'' or even g''' at the boundary points a and b by computing the corresponding derivatives of the cubic polynomial that matches g at the four data sites nearest to the boundary point, and then forcing f to match that derivative value. We leave it to the reader to work out the requisite additional equations. As with the not-a-knot condition, the resulting approximation is $\mathcal{O}(|\tau|^4)$ (see Swartz & Varga [1972]).

(vi) Curtis & Powell (see Hayes [1970:Ch.4]) have an interesting justification for the end condition $(f-g)(\tau_{3/2}) = (f-g)(\tau_{5/2})$, with $\tau_{(2i+1)/2} := (\tau_i + \tau_{i+1})/2$.

(vii) Note that one can freely choose to have different conditions at the two ends of the interval.

There follows a FORTRAN program CUBSPL for computing the cubic spline interpolant. It allows for prescribing end slopes, end second derivatives, or for prescribing nothing at all, in which case the not-a-knot condition is used. In anticipation of later programs, the ith polynomial piece P_i is written here as

$$P_i(x) = C_{1,i} + C_{2,i}(x - \tau_i) + C_{3,i}(x - \tau_i)^2/2 + C_{4,i}(x - \tau_i)^3/6$$

instead of in the earlier way given in (4).

```
      SUBROUTINE CUBSPL ( TAU, C, N, IBCBEG, IBCEND )
C     ********************  INPUT  ****************************
C     N = NUMBER OF DATA POINTS. ASSUMED TO BE .GE. 2.
C     (TAU(I), C(1,I), I=1,...,N) = ABSCISSAE AND ORDINATES OF THE
C        DATA POINTS. TAU IS ASSUMED TO BE STRICTLY INCREASING.
C     IBCBEG, IBCEND = BOUNDARY CONDITION INDICATORS, AND
C     C(2,1), C(2,N) = BOUNDARY CONDITION INFORMATION. SPECIFICALLY,
C        IBCBEG = 0  MEANS NO BOUNDARY CONDITION AT TAU(1) IS GIVEN.
C           IN THIS CASE, THE NOT-A-KNOT CONDITION IS USED, I.E. THE
C           JUMP IN THE THIRD DERIVATIVE ACROSS TAU(2) IS FORCED TO
C           ZERO, THUS THE FIRST AND THE SECOND CUBIC POLYNOMIAL PIECES
C           ARE MADE TO COINCIDE.)
C        IBCBEG = 1  MEANS THAT THE SLOPE AT TAU(1) IS MADE TO EQUAL
C           C(2,1), SUPPLIED BY INPUT.
C        IBCBEG = 2  MEANS THAT THE SECOND DERIVATIVE AT TAU(1) IS
C           MADE TO EQUAL C(2,1), SUPPLIED BY INPUT.
C        IBCEND = 0, 1, OR 2 HAS ANALOGOUS MEANING CONCERNING THE
C           BOUNDARY CONDITION AT TAU(N), WITH THE ADDITIONAL INFOR-
C           MATION TAKEN FROM C(2,N).
C     ********************  OUTPUT  ****************************
C     C(J,I), J=1,...,4; I=1,...,L (= N-1) = THE POLYNOMIAL COEFFICIENTS
C        OF THE CUBIC INTERPOLATING SPLINE WITH INTERIOR KNOTS (OR
C        JOINTS) TAU(2), ..., TAU(N-1). PRECISELY, IN THE INTERVAL
C        (TAU(I) .. TAU(I+1)), THE SPLINE F IS GIVEN BY
C           F(X) = C(1,I)+H*(C(2,I)+H*(C(3,I)+H*C(4,I)/3.)/2.)
C        WHERE H = X - TAU(I). THE FUNCTION PROGRAM *PPVALU* MAY BE
C        USED TO EVALUATE F OR ITS DERIVATIVES FROM TAU,C, L = N-1,
C        AND K=4.
      INTEGER IBCBEG,IBCEND,N,   I,J,L,M
      REAL C(4,N),TAU(N),   DIVDF1,DIVDF3,DTAU,G
C****** A TRIDIAGONAL LINEAR SYSTEM FOR THE UNKNOWN SLOPES S(I) OF
C  F  AT TAU(I), I=1,...,N, IS GENERATED AND THEN SOLVED BY GAUSS ELIM-
C  INATION, WITH S(I) ENDING UP IN C(2,I), ALL I.
C     C(3,.) AND C(4,.) ARE USED INITIALLY FOR TEMPORARY STORAGE.
      L = N-1
COMPUTE FIRST DIFFERENCES OF TAU SEQUENCE AND STORE IN C(3,.). ALSO,
COMPUTE FIRST DIVIDED DIFFERENCE OF DATA AND STORE IN C(4,.).
      DO 10 M=2,N
         C(3,M) = TAU(M) - TAU(M-1)
   10    C(4,M) = (C(1,M) - C(1,M-1))/C(3,M)
CONSTRUCT FIRST EQUATION FROM THE BOUNDARY CONDITION, OF THE FORM
C             C(4,1)*S(1) + C(3,1)*S(2) = C(2,1)
      IF (IBCBEG-1)                          11,15,16
   11 IF (N .GT. 2)                          GO TO 12
C     NO CONDITION AT LEFT END AND N = 2.
      C(4,1) = 1.
      C(3,1) = 1.
      C(2,1) = 2.*C(4,2)
                                             GO TO 25
C     NOT-A-KNOT CONDITION AT LEFT END AND N .GT. 2.
   12 C(4,1) = C(3,3)
      C(3,1) = C(3,2) + C(3,3)
      C(2,1) =((C(3,2)+2.*C(3,1))*C(4,2)*C(3,3)+C(3,2)**2*C(4,3))/C(3,1)
                                             GO TO 19
C     SLOPE PRESCRIBED AT LEFT END.
   15 C(4,1) = 1.
      C(3,1) = 0.
                                             GO TO 18
```

```
C     SECOND DERIVATIVE PRESCRIBED AT LEFT END.
   16 C(4,1) = 2.
      C(3,1) = 1.
      C(2,1) = 3.*C(4,2) - C(3,2)/2.*C(2,1)
   18 IF(N .EQ. 2)                                  GO TO 25
C  IF THERE ARE INTERIOR KNOTS, GENERATE THE CORRESP. EQUATIONS AND CAR-
C  RY OUT THE FORWARD PASS OF GAUSS ELIMINATION, AFTER WHICH THE M-TH
C  EQUATION READS    C(4,M)*S(M) + C(3,M)*S(M+1) = C(2,M).
   19 DO 20 M=2,L
         G = -C(3,M+1)/C(4,M-1)
         C(2,M) = G*C(2,M-1) + 3.*(C(3,M)*C(4,M+1)+C(3,M+1)*C(4,M))
   20    C(4,M) = G*C(3,M-1) + 2.*(C(3,M) + C(3,M+1))
CONSTRUCT LAST EQUATION FROM THE SECOND BOUNDARY CONDITION, OF THE FORM
C           (-G*C(4,N-1))*S(N-1) + C(4,N)*S(N) = C(2,N)
C     IF SLOPE IS PRESCRIBED AT RIGHT END, ONE CAN GO DIRECTLY TO BACK-
C     SUBSTITUTION, SINCE C ARRAY HAPPENS TO BE SET UP JUST RIGHT FOR IT
C     AT THIS POINT.
      IF (IBCEND-1)                              21,30,24
   21 IF (N .EQ. 3 .AND. IBCBEG .EQ. 0) GO TO 22
C     NOT-A-KNOT AND N .GE. 3, AND EITHER N.GT.3 OR  ALSO NOT-A-KNOT AT
C     LEFT END POINT.
      G = C(3,N-1) + C(3,N)
      C(2,N) = ((C(3,N)+2.*G)*C(4,N)*C(3,N-1)
     *             + C(3,N)**2*(C(1,N-1)-C(1,N-2))/C(3,N-1))/G
      G = -G/C(4,N-1)
      C(4,N) = C(3,N-1)
                                                 GO TO 29
C     EITHER (N=3 AND NOT-A-KNOT ALSO AT LEFT) OR (N=2 AND NOT NOT-A-
C     KNOT AT LEFT END POINT).
   22 C(2,N) = 2.*C(4,N)
      C(4,N) = 1.
                                                 GO TO 28
C     SECOND DERIVATIVE PRESCRIBED AT RIGHT ENDPOINT.
   24 C(2,N) = 3.*C(4,N) + C(3,N)/2.*C(2,N)
      C(4,N) = 2.
                                                 GO TO 28
   25 IF (IBCEND-1)                              26,30,24
   26 IF (IBCBEG .GT. 0)                         GO TO 22
C     NOT-A-KNOT AT RIGHT ENDPOINT AND AT LEFT ENDPOINT AND N = 2.
      C(2,N) = C(4,N)
                                                 GO TO 30
   28 G = -1./C(4,N-1)
COMPLETE FORWARD PASS OF GAUSS ELIMINATION.
   29 C(4,N) = G*C(3,N-1) + C(4,N)
      C(2,N) = (G*C(2,N-1) + C(2,N))/C(4,N)
CARRY OUT BACK SUBSTITUTION
   30 J = L
   40    C(2,J) = (C(2,J) - C(3,J)*C(2,J+1))/C(4,J)
         J = J - 1
         IF (J .GT. 0)                           GO TO 40
C****** GENERATE CUBIC COEFFICIENTS IN EACH INTERVAL, I.E., THE DERIV.S
C  AT ITS LEFT ENDPOINT, FROM VALUE AND SLOPE AT ITS ENDPOINTS.
      DO 50 I=2,N
      DTAU = C(3,I)
      DIVDF1 = (C(1,I) - C(1,I-1))/DTAU
      DIVDF3 = C(2,I-1) + C(2,I) - 2.*DIVDF1
      C(3,I-1) = 2.*(DIVDF1 - C(2,I-1) - DIVDF3)/DTAU
50    C(4,I-1) = (DIVDF3/DTAU)*(6./DTAU)
                                                 RETURN
   END
```

Problems

1. Another popular piecewise cubic interpolation scheme is based on choosing s_i so that the tangent to f at τ_i has equal angle with the secant at τ_{i-1}, τ_i and the secant at τ_i, τ_{i+1}. Work out the formula for s_i in terms of $[\tau_{i-1}, \tau_i]g$ and $[\tau_i, \tau_{i+1}]g$.

2. (The cubic Hermite basis) Prove that an arbitrary cubic polynomial p can be written uniquely as

$$p(x) = p(0)\varphi_1(x) + p(h)\varphi_2(x) + p'(0)\varphi_3(x) + p'(h)\varphi_4(x)$$

for given $h \neq 0$, with

$$\begin{aligned} \varphi_1(x) &:= 1 + y^2(2y-3), & \varphi_2(x) &:= \varphi_1(h-x) = 1 - \varphi_1(x), \\ \varphi_3(x) &:= x(1-y)^2, & \varphi_4(x) &:= -\varphi_3(h-x), \end{aligned} \qquad y := x/h$$

Hence,

$$p(x) = p(0) + (p(h) - p(0))y^2(3-2y) + (p'(0)(1-y) - p'(h)y)x(1-y).$$

3. (Error in piecewise cubic interpolation)
(a) Use Problem 2 to show that the error in a piecewise cubic interpolant f to g can be written

$$e := g - f = g - f^H + E,$$

with f^H the piecewise cubic Hermite interpolant and

$$E(x) := \left(e'(\tau_i)\frac{\tau_{i+1} - x}{\Delta\tau_i} - e'(\tau_{i+1})\frac{x - \tau_i}{\Delta\tau_i} \right) \frac{(x-\tau_i)(\tau_{i+1}-x)}{\Delta\tau_i},$$
$$\tau_i \leq x \leq \tau_{i+1}.$$

Note that $e'(\tau_j) = g'(\tau_j) - s_j$, all j.
(b) Conclude from (a) and from (6) above that, for a sufficiently smooth g, $\|g - f\| \leq \|g - f^H\| + |\tau| \max_i |e'(\tau_i)|/4$, or

$$\|g - f\| = \mathcal{O}\left(|\tau|^4\right) + \max_i |e'(\tau_i)| \mathcal{O}\left(|\tau|\right).$$

(c) Prove that, in general, piecewise cubic Bessel interpolation (with prescribed endslopes) is $\mathcal{O}\left(|\tau|^3\right)$.
(d) Prove that Akima's interpolation scheme (for example, with prescribed endslopes) is $\mathcal{O}\left(|\tau|^2\right)$.
(e) In the incorrect first run of the program for the Runge example (8) with piecewise cubic Hermite interpolation, we have $\max_i |e'(\tau_i)| \approx 3.25$

regardless of τ, hence (b) would allow only the conclusion that $\|E\| = \mathcal{O}(|\tau|)$, yet the numbers showed $\|E\| = \mathcal{O}(|\tau|^2)$. Explain! (Hint: Show by additional printout that, for the range of n considered, the maximum error occurs in the middle interval that is centered around zero, then show that $\|E\| = \mathcal{O}(h^2)$ on that interval $[-h/2 \,..\, h/2]$, because of the systematic nature of the error in the slopes).

4. Adapt the experimental program in Example (8) to carry out piecewise cubic Bessel interpolation (with prescribed endslopes). Then verify that the error in Bessel interpolation is in general no better than $\mathcal{O}(|\tau|^3)$ by interpolating $g(x) = x^3$ on $[0 \,..\, 1]$ on a mesh τ that is uniform except for one site, say the first interior site, that is twice as far from its left neighbor than from its right.

5. Adapt the experimental program in Example (8) to carry out Akima's interpolation (with $s_1 = [\tau_0, \tau_1]g = g'(\tau_1)$ and $s_n = [\tau_n, \tau_{n+1}]g = g'(\tau_n)$). Then verify that Akima's scheme is, in general, no better than $\mathcal{O}(|\tau|^2)$ by applying it to $g(x) = x^2$ on $[0 \,..\, 1]$ for a uniform τ.

Also, appreciate the effect of a nonuniform τ by interpolating $g(x) = x^2$ on $[0 \,..\, 1]$ on a mesh derived from a uniform one by moving every other site halfway toward its neighbor on the right. Look at the error!

6. A common approach to cubic spline interpolation expresses the cubic piece P_i in terms of its values at τ_i and τ_{i+1} and its *second* derivative, m_i and m_{i+1} at τ_i and τ_{i+1}, respectively. The free parameters $m_1, \ldots, m_n$ are then determined so that $P'_{i-1}(\tau_i) = P'_i(\tau_i)$, $i = 2, \ldots, n-1$.
(a) Derive the formula corresponding to (4) and (5) for this approach.
(b) Determine the linear system for the m_i's. (Remark: I have given preference to the approach in the text because it relates cubic spline interpolation explicitly to a host of other piecewise cubic interpolation schemes.)

7. (Convergence of complete cubic spline interpolation)
(a) Use the techniques of Chapter III to show that the solution $(s_i)_2^{n-1}$ of (12) for given s_1, s_n satisfies

$$\max_i |s_i| \le \max\{\, |s_1|, \max_{1<i<n} |b_i|/(\tau_{i+1} - \tau_{i-1}), |s_n| \,\}.$$

(b) Conclude that the complete cubic spline interpolant satisfies

$$\|(I_4 g)'\| \le 3.75 \max_{0 \le i \le n} |[\tau_i, \tau_{i+1}]g|.$$

(Hint: show that $(I_4 g)'(\tau_{i+1/2}) = \frac{3}{2}[\tau_i, \tau_{i+1}]g - (s_i + s_{i+1})/4$, with $\tau_{i+1/2} := (\tau_i + \tau_{i+1})/2$. Then use the fact that, for $p \in \Pi_{<3}$, $\max\{\, |p(x)| : a \le x \le b \,\} \le (5/4)\max\{\, |p(a)|, |p(\frac{a+b}{2})|, |p(b)| \,\}$; see Problem II.4).
(c) Deduce from (b) and from Problem III.2 that

$$\|g' - (I_4 g)'\| \le 4.75 \operatorname{dist}(g', \$_3),$$

with $\$_3 := \{ f \in C^{(1)}[\tau_1 \,..\, \tau_n] : f_{|[\tau_i..\tau_{i+1}]} \in \Pi_{<3}$, all $i \}$.
(d) Deduce from (c) that $\|g - I_4 g\| \le (19/8)|\tau|\|g'\|$, in case g has a bounded first derivative.

8. Derive (17) from (2), (3), and $P_1'''(\tau_2) = P_2'''(\tau_2)$, then derive (18) from (17) by an appropriate change of variables.

9. Here is the cumulative distribution $N = N(\text{age})$, i.e., as a function of age, of Bulgarian women giving birth, tabulated in five-year intervals

age	15	20	25	30	35	40	45	50
N	0	7,442	26,703	41,088	47,635	49,758	50,209	50,266

Fit these data with a cubic spline, using CUBSPL and the end conditions $f'(15) = f'(50) = 0$. Then draw f' on [15..50], and also draw the histogram or piecewise constant function that, on $[\tau_i \,..\, \tau_{i+1}]$, has the value $(N(\tau_{i+1}) - N(\tau_i))/5$, with $\tau_i := 10 + 5i$, $i = 1, \ldots, 8$. Compare with Figure VIII(2).

This example is taken from L. I. Boneva, D. G. Kendall & I. Stefanov [1971].

V

Best Approximation Properties of Complete Cubic Spline Interpolation and Its Error

We show that the complete cubic spline interpolant $f = I_4 g$ to g, constructed in the preceding chapter, minimizes $\int_a^b [f''(x)]^2 \, dx$ over all twice differentiable functions f that agree with g at $\tau_0, \ldots, \tau_{n+1}$. This is the property that gave rise to the name "spline". We also show that the error $\|g - I_4 g\|$ is $\mathcal{O}(|\tau|^4)$.

Consider again the data sites

$$a = \tau_0 = \tau_1 < \cdots < \tau_n = \tau_{n+1} = b.$$

In the preceding chapter, we constructed for the given g its complete cubic spline interpolant $I_4 g$ for those data sites. To recall, $I_4 g$ agrees with g at $(\tau_i)_0^{n+1}$ (in particular, $(I_4 g)' = g'$ at τ_1 and τ_n since $\tau_0 = \tau_1$ and $\tau_{n+1} = \tau_n$), and is **a cubic spline on** $[a\,..\,b]$ **with interior knots** $\tau_2, \ldots, \tau_{n-1}$, that is a twice continuously differentiable, piecewise cubic function on $[a\,..\,b]$ with breaks $\tau_2, \ldots, \tau_{n-1}$. We denote the class or linear space of all cubic splines on $[a\,..\,b]$ with interior knots at $\tau_2, \ldots, \tau_{n-1}$ by

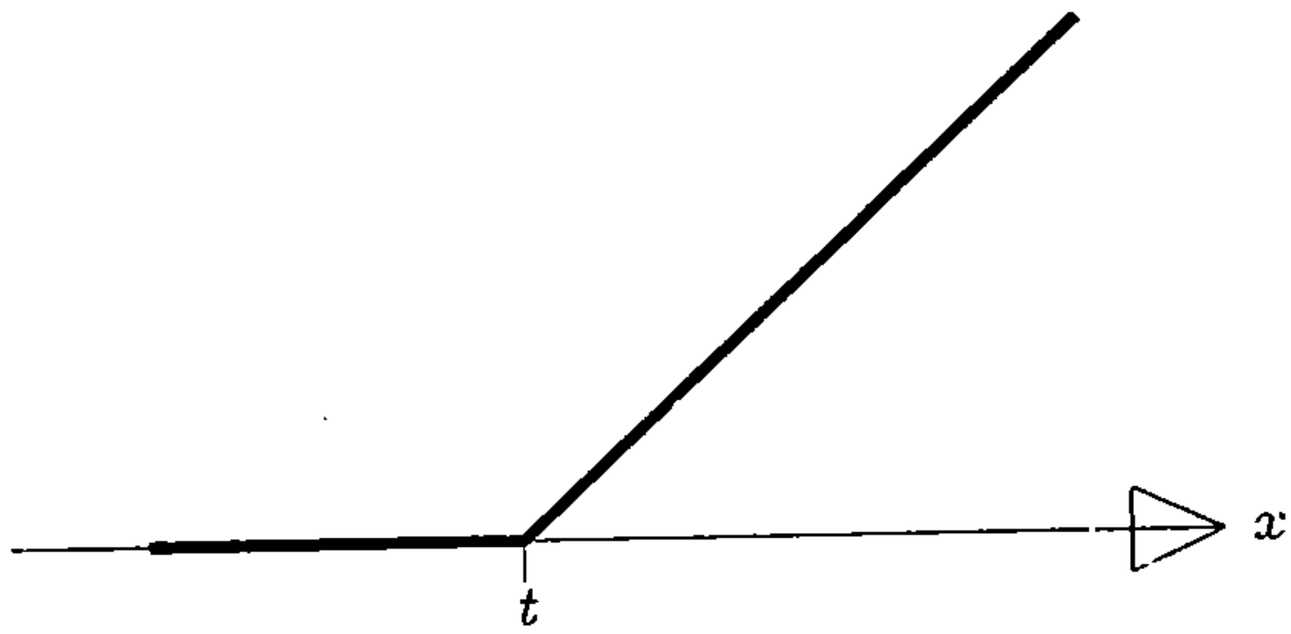
$$\$_4.$$

We derive the various best approximation properties of the complete cubic spline interpolant from the following simple lemma which relates the interpolant to least-squares approximation to its second derivative from $\$_2$, the linear space of continuous broken lines on $[a\,..\,b]$ with breaks $\tau_2, \ldots, \tau_{n-1}$ as introduced in Chapter III.

(1) **Lemma.** *If g is twice differentiable, then the second derivative of the interpolation error $e := g - I_4 g$ is orthogonal to $\$_2$, that is,*

$$(2) \qquad \int_a^b e''(x)\varphi(x)\,dx = 0 \qquad \text{for all } \varphi \in \$_2.$$

PROOF. The customary proof of this lemma uses nothing more than integration by parts and can therefore be supplied directly by the reader.



(3) FIGURE. The function $f(x) = (x - t)_+$

In anticipation of material in subsequent chapters, I prefer to prove the lemma via the integral representation for the second divided difference, as follows.

From Taylor's expansion with integral remainder, we have

$$h(x) = h(a) + (x - a)h'(a) + \int_a^x (x - t)h''(t)\,\mathrm{d}t$$

for any twice differentiable function h. Since the second divided difference of any straight line is zero, we obtain

$$[\tau_{i-1}, \tau_i, \tau_{i+1}]h = [\tau_{i-1}, \tau_i, \tau_{i+1}] \int_a^x (x - t)h''(t)\,\mathrm{d}t, \tag{4}$$

where the divided difference on the right is taken of the function of x defined by the integral. Note that the independent variable x appears in two places there. In order to simplify things and take the divided difference inside of the integral, we introduce the **truncated** function

$$(x - t)_+ := \max\{0, x - t\},$$

depicted in Figure (3). With this function, we can write

$$\int_a^x (x - t)h''(t)\,\mathrm{d}t = \int_a^b (x - t)_+ h''(t)\,\mathrm{d}t, \qquad \text{for } a \le x \le b,$$

and so may rewrite (4) as

$$[\tau_{i-1}, \tau_i, \tau_{i+1}]h = \int_a^b [\tau_{i-1}, \tau_i, \tau_{i+1}](x - t)_+ h''(t)\,\mathrm{d}t = \int_a^b \hat{H}_i(t)h''(t)\,\mathrm{d}t/2$$

where $\hat{H}_i$ is the piecewise linear "hat" function III(16); that is,

$$\begin{aligned}\hat{H}_i(t) &:= 2[\tau_{i-1}, \tau_i, \tau_{i+1}](x - t)_+ \\ &= \frac{2}{\tau_{i+1} - \tau_{i-1}} \begin{cases} (t - \tau_{i-1})/\Delta\tau_{i-1}, & \tau_{i-1} < t \le \tau_i, \\ (\tau_{i+1} - t)/\Delta\tau_i, & \tau_i \le t < \tau_{i+1}, \\ 0, & \text{otherwise,} \end{cases}\end{aligned}$$

is twice a second divided difference of $(x-t)_+$ for each fixed t as a function of x.

Now, since the interpolation error $e = g - I_4 g$ vanishes at all the τ_i's, its second divided differences at these sites are therefore also zero, that is,

$$0 = [\tau_{i-1}, \tau_i, \tau_{i+1}]e = \int_a^b \hat{H}_i(t) e''(t)\,\mathrm{d}t, \qquad i = 1, \ldots, n.$$

This shows e'' to be orthogonal to each of the n functions $\hat{H}_1, \ldots, \hat{H}_n$, and, since these span $\$_2$, the lemma follows. □

(5) Theorem. (Pythagoras) *For every twice continuously differentiable function g on $[a\,..\,b]$,*

$$\text{(6)} \qquad \int_a^b [g''(x)]^2\,\mathrm{d}x = \int_a^b [(I_4 g)''(x)]^2\,\mathrm{d}x + \int_a^b [(g - I_4 g)''(x)]^2\,\mathrm{d}x.$$

PROOF. We have

$$\begin{aligned}\int_a^b [g''(t)]^2\,\mathrm{d}t &= \int_a^b [(I_4 g)''(t) + e''(t)]^2\,\mathrm{d}t \\ &= \int_a^b [(I_4 g)''(t)]^2\,\mathrm{d}t + 2\int_a^b (I_4 g)''(t) e''(t)\,\mathrm{d}t + \int_a^b [e''(t)]^2\,\mathrm{d}t.\end{aligned}$$

But $\int_a^b (I_4 g)''(t) e''(t)\,\mathrm{d}t = 0$ by Lemma (1), since $(I_4 g)'' \in \$_2$. □

(7) Corollary. *Among all functions f with two continuous derivatives that agree with the function g at the sites $\tau_0, \ldots, \tau_{n+1}$, the particular function $I_4 g$ uniquely minimizes $\int_a^b [f''(t)]^2\,\mathrm{d}t$.*

PROOF. For any such function f, we must have $I_4 f = I_4 g$, therefore, from Theorem (5),

$$\text{(8)} \qquad \begin{aligned}\int_a^b [f''(t)]^2\,\mathrm{d}t &= \int_a^b [(I_4 g)''(t)]^2\,\mathrm{d}t + \int_a^b [f''(t) - (I_4 g)''(t)]^2\,\mathrm{d}t \\ &\geq \int_a^b [(I_4 g)''(t)]^2\,\mathrm{d}t,\end{aligned}$$

and the inequality becomes equality if and only if $f'' = (I_4 g)''$, which, because of the interpolation conditions, is equivalent to $f = I_4 g$. □

This is the **smoothest interpolation property** of cubic spline interpolation. The function $f = I_4 g$ minimizes (approximately only!) the strain energy

$$\int_a^b \frac{[f''(t)]^2}{[1+(f'(t))^2]^{5/2}}\,\mathrm{d}t$$

over all function curves passing through the given data. In this sense $I_4 g$ approximates the position of a flexible thin beam or **draftman's spline** forced through the given data. This is the reason behind Schoenberg's [1946] choice of the words "spline" curve and "spline" function.

(9) Corollary. *The second derivative of the complete cubic spline interpolant to the function g is the least-squares approximation from the space $\$_2$ of broken lines to the second derivative of g, that is,*

$$(I_4 g)'' = L_2(g'').$$

PROOF. Let $s \in \$_2$, and take $h(x) := \int_a^b (x-t)_+ s(t)\,\mathrm{d}t$. Then $h'' = s$, and $h \in \$_4$, consequently $I_4 h = h$ and so $(I_4 h)'' = h'' = s$ and $h - I_4 h = 0$. Therefore, substitution of $g - h$ for g in Theorem (5) gives

$$\int_a^b [g''(x) - s(x)]^2\,\mathrm{d}x = \int_a^b [((I_4 g)'' - s)(x)]^2\,\mathrm{d}x + \int_a^b [(g - I_4 g)''(x)]^2\,\mathrm{d}x$$
$$\geq \int_a^b [g''(x) - (I_4 g)''(x)]^2\,\mathrm{d}x$$

with equality if and only if $(I_4 g)'' = s$. Hence $L_2(g'') = I_4 g''$.

An alternative proof is based directly on Lemma (1). We know by that lemma that there are coefficients $(\alpha_i)_1^n$ so that $(I_4 g)'' = \sum_1^n \alpha_j H_j$ and $\int_a^b (g'' - \sum \alpha_j H_j) H_i = 0$ for $i = 1, \ldots, n$ (with H_j given by III(6)). But these last orthogonality conditions can also be written

$$\sum_{j=1}^n \Big(\int H_i H_j \Big) \alpha_j = \int H_i g'', \qquad i = 1, \ldots, n,$$

which, on comparing with III(10), we recognize as the normal equations for the determination of the least-squares approximation $L_2 g'' = \sum_j \alpha_j H_j$ to g'' from $\$_2$. □

The corollary comprises the **best approximation property** of complete cubic spline interpolation. As a bonus, we obtain at once from Theorem III(12) that

$$\|(I_4 g)''\| \leq 3\|g''\|$$

and that, for a function g with four continuous derivatives,

$$\|e''\| = \|g'' - (I_4 g)''\| \leq 4\ \mathrm{dist}\,(g'', \$_2) \leq \frac{1}{2}|\tau|^2 \|g^{(4)}\|.$$

From this, we obtain a bound on the maximum interpolation error $\|e\|$ as follows. If $\tau_i \le x \le \tau_{i+1}$, then

$$\begin{aligned} e(x) &= e(\tau_i) + (x-\tau_i)[\tau_i,\tau_{i+1}]e + (x-\tau_i)(x-\tau_{i+1})[\tau_i,\tau_{i+1},x]e \\ &= \quad 0 \quad + \qquad 0 \qquad + \quad (x-\tau_i)(x-\tau_{i+1})e''(\zeta_x)/2 \end{aligned}$$

for some ζ_x in $(\tau_i \,..\, \tau_{i+1})$. Therefore, we then have

$$|e(x)| \le (\Delta\tau_i/2)^2 \max_{\tau_i \le \zeta \le \tau_{i+1}} |e''(\zeta)|/2.$$

Thus $\|e\| \le \frac{1}{8}|\tau|^2\|e''\| \le \frac{1}{8}|\tau|^2 4\,\mathrm{dist}\,(g'', \$_2)$, which proves

(10) Corollary. *For a twice continuously differentiable function g,*

$$\|g - I_4 g\| \le \frac{1}{2}|\tau|^2\,\mathrm{dist}\,(g'', \$_2). \tag{11}$$

Therefore, from equation III(2),

$$\|g - I_4 g\| \le \frac{1}{16}|\tau|^4\|g^{(4)}\| \tag{12}$$

in case g has four continuous derivatives.

Actually, as proved by Hall [1968],

$$\|g - I_4 g\| \le \frac{5}{384}|\tau|^4\|g^{(4)}\|, \tag{13}$$

and the constant (5/384) is best possible, as proved by Hall & Meyer [1976] (see Problem 2(c) and Problem VI.5(c)).

Finally, for completeness, we recall from Problem IV.7 that

$$\|g - I_4 g\| \le \frac{19}{8}|\tau|\,\mathrm{dist}\,(g', \$_3) \tag{14}$$

in case g is differentiable.

Cubic spline interpolation also provides acceptable approximations to derivatives and is therefore a popular means for *numerical differentiation*. We saw already that

$$\|g'' - (I_4 g)''\| \le 4\,\mathrm{dist}\,(g'', \$_2) \le \frac{1}{2}|\tau|^2\|g^{(4)}\|. \tag{15}$$

Actually, Hall & Meyer [1976] proved that

$$\|g'' - (I_4 g)''\| \le \frac{3}{8}|\tau|^2\|g^{(4)}\|. \tag{16}$$

Also, from Problem IV.7 and Problem III.2,

$$\|g' - (I_4 g)'\| \leq \frac{19}{4} \operatorname{dist}(g', \$_3) \tag{17}$$

while, from Problem 2,

$$\max_i |g'(\tau_i) - (I_4 g)'(\tau_i)| \leq \frac{1}{24} |\tau|^3 \|g^{(4)}\| \tag{18}$$

and, by Problem 5, for a uniform τ, even

$$\max_i |g'(\tau_i) - (I_4 g)'(\tau_i)| \leq \frac{1}{60} |\tau|^4 \|g^{(5)}\|, \tag{19}$$

that is, one order higher. Somewhat more effort shows that

$$\|g' - (I_4 g)'\| \leq \frac{1}{24} |\tau|^3 \|g^{(4)}\| \tag{20}$$

and that this estimate is best possible (see Problem VI.5); see Hall & Meyer [1976]. Finally, the quality of approximation to the third derivative may depend on the spacing of τ. Hall & Meyer [1976] show that

$$\|g^{(3)} - (I_4 g)^{(3)}\| \leq \frac{1}{2}(M_\tau + 1/M_\tau)|\tau| \|g^{(4)}\|, \tag{21}$$

with the **global mesh ratio** M_τ given by

$$M_\tau := |\tau| / \min_{i=1,\ldots,n-1} \Delta\tau_i. \tag{22}$$

Problems

1. Derive (14) from Problem IV.7(d). (Hint: in Problem IV.7(d), substitute $g - h$ for g, with $h(x) := \int_a^x s(t)\,dt$ and $s \in \$_3$ arbitrary, then minimize over s.)

2. (Proof of (13)) Let $e_i' := e'(\tau_i)$, all i, with $e := g - I_4 g$, and assume that g has four continuous derivatives.
(a) Prove that (e_i') satisfies the equation

$$\delta_i e_{i-1}' + 2e_i' + (1 - \delta_i)\epsilon_{i+1}' = \beta_i$$

with $\delta_i := \Delta\tau_i/(\tau_{i+1} - \tau_{i-1})$,

$$\beta_i := [\delta_i(-\Delta\tau_{i-1})^3 g^{(4)}(\zeta_i^-) + (1 - \delta_i)(\Delta\tau_i)^3 g^{(4)}(\zeta_i^+)]/24,$$

and $\zeta_i^-, \zeta_i^+ \in [\tau_{i-1} \,..\, \tau_{i+1}]$, $i = 2, \ldots, n-1$. (Hint: By IV(12), $\delta_i e_{i-1}' +$

$2e_i' + (1-\delta_i)e_{i+1}' = \delta_i g'(\tau_{i-1}) + 2g'(\tau_i) + (1-\delta_i)g'(\tau_{i+1}) - 3(\delta_i[\tau_{i-1},\tau_i]g + (1-\delta_i)[\tau_i,\tau_{i+1}]g)$, all i. Now verify (for example, by Taylor expansion or by Problem 3), and then use, the fact that, for a sufficiently smooth function F,

$$(*) \qquad 2F'(0) + F'(h) - 3[0,h]F = -\frac{h}{2}F''(0) + (h^3/24)F^{(4)}(\zeta)$$

for some ζ between 0 and h. (This fixes a flaw in the argument for (13) given in Hall [1968].))

(b) Use the considerations of Problem IV.7(a) to conclude from (a) that $\max_i |e_i'| \le |\tau|^3 \|g^{(4)}\|/24$.

(c) Prove (13) from (b) using Problem IV.3(b).

3. Derive the identity (*) in Problem 2(a) above in the following way: Integrate the identity $F'(x) = F'(0) + F''(0)x + [0,0,h]F' \cdot x^2 + [0,0,h,x]F' \cdot x^2(x-h)$ (obtained from Theorem I(14)) from 0 to h, then use the mean value theorem for integrals and the fact that $[0,0,h,x]F' = (F')'''(\zeta)/3!$ for some ζ between 0 and h (if x is in that interval).

4. Identify the linear system III(11) or III(15) with the linear system derived in Problem IV.6(b).

5. Carry the expansion (*) in Problem 2(a) one step further, that is, prove that

$$2F'(0) + F'(h) - 3[0,h]F = -\frac{h}{2}F''(0) + \frac{h^3}{24}F^{(4)}(0) + \frac{h^4}{60}F^{(5)}(\zeta)$$

for some ζ between 0 and h. Then use this expansion to show that, for a *uniform* τ, even $\max_i |e_i'| \le |\tau|^4 \|g^{(5)}\|/60$. In words, *complete cubic spline interpolation at uniformly spaced data sites gives* $\mathcal{O}(n^{-4})$ *approximation even to the slopes at the data sites.*

6. Let $\tau_i := (i-1)/(n-1)$, $i = 1, \ldots, n$.

(a) Show that the function $e(x) := (x-\tau_i)^2(x-\tau_{i+1})^2$, $\tau_i \le x \le \tau_{i+1}$, $i = 1, \ldots, n-1$, is of the form $x^4 - f(x)$ for some $f \in \$_4$.

(b) Conclude that e is the error in the complete cubic spline interpolant to x^4. Then conclude that the overall error in f' as an approximation of g' (with $g(x) = x^4$ in this example) is no better than $\mathcal{O}(n^{-3})$ even though $\max_i |e_i'| = \mathcal{O}(n^{-4})$.

A function of the form $x^4 - f(x)$ for some $f \in \$_4$ is called a **monospline of degree 4.**

7. Let $I_2 g$ be the broken line interpolant to g, at the sites $a = \tau_1 < \cdots < \tau_n = b$, as introduced in III(8). Prove that, for every piecewise continuously differentiable function g,

$$\int_a^b [g'(x)]^2\,dx = \int_a^b [(I_2 g)'(x)]^2\,dx + \int_a^b [(g - I_2 g)'(x)]^2\,dx.$$

Conclude that $(I_2 g)' = L_1(g')$, with L_1 least-squares approximation from $\$_1 := \{ f' : f \in \$_2 \}$ = all piecewise constant functions on $[a \,..\, b]$ with breaks $\tau_2, \ldots, \tau_{n-1}$. Conclude also that broken line interpolation is "smoothest" in the sense that $\int_a^b [(I_2 g)'(x)]^2 \, dx \leq \int_a^b [f'(x)]^2 \, dx$ for all piecewise continuously differentiable f that agree with g at τ.

VI

Parabolic Spline Interpolation

In this chapter, we discuss briefly interpolation by parabolic splines in order to make the point that, in piecewise polynomial interpolation, it might be advantageous at times to interpolate at sites other than the breaks. In the process, we come upon the exponential decay of certain influence functions, and expand on this in the problems.

We begin again with given data $g(\tau_1), \ldots, g(\tau_n)$ at data sites $a = \tau_1 < \ldots < \tau_n = b$. Suppose first that we proceed exactly as in the construction of a piecewise cubic interpolant in Chapter IV. We choose the interpolant f to g to consist of parabolic pieces,

$$f(x) = P_i(x) \quad \text{on } \tau_i \le x \le \tau_{i+1} \text{ for some } P_i \in \Pi_{<3},\ i = 1, \ldots, n-1.$$

Then

$$P_i(\tau_i) = g(\tau_i), \quad P_i(\tau_{i+1}) = g(\tau_{i+1}), \tag{1}$$

which leaves one degree of freedom for each piece still to be determined. We take this degree of freedom in the form

$$P_i(\tau_{i+1/2}) = v_{i+1}, \quad \text{with } \tau_{i+1/2} := (\tau_i + \tau_{i+1})/2, \tag{2}$$

and now look for ways of choosing these numbers $v_2, \ldots, v_n$. We could determine these additional parameters from some local considerations. For instance, we could choose $v_{i+1} = g(\tau_{i+1/2})$ if the latter numbers are available, thus interpolating g at the sites $\tau_{i+1/2}$, $i = 2, \ldots, n$, as well. The reader is familiar with the resulting interpolation scheme in a somewhat disguised form: If we integrate the resulting piecewise parabolic interpolant f for g, we obtain a general form of Simpson's rule for $\int_a^b g(x)\,dx$.

Another idea is to determine v_i, $i = 2, \ldots, n$, so as to make the interpolating function f a *parabolic spline*, that is, so as to make the function f continuously differentiable. In order to derive the resulting linear system

for the numbers v_i, we write P_i in Newton form

$$\begin{aligned} P_i(x) &= P_i(\tau_i) + (x-\tau_i)[\tau_i, \tau_{i+1/2}]P_i \\ &\qquad + (x-\tau_i)(x-\tau_{i+1/2})[\tau_i, \tau_{i+1/2}, \tau_{i+1}]P_i \\ &= P_i(\tau_i) + (x-\tau_i)s_i^+ + (x-\tau_i)(x-\tau_{i+1/2})(s_{i+1}^- - s_i^+)/\Delta\tau_i \end{aligned}$$

with the abbreviations

$$\begin{aligned} s_i^+ &:= [\tau_i, \tau_{i+1/2}]P_i = (v_{i+1} - g(\tau_i))/(\Delta\tau_i/2), \\ s_{i+1}^- &:= [\tau_{i+1/2}, \tau_{i+1}]P_i = (g(\tau_{i+1}) - v_{i+1})/(\Delta\tau_i/2). \end{aligned}$$

Then

$$\begin{aligned} P_i'(\tau_i) &= s_i^+ + (\tau_i + \tau_{i+1/2})(s_{i+1}^- - s_i^+)/\Delta\tau_i = (3s_i^+ - s_{i+1}^-)/2, \\ P_{i-1}'(\tau_i) &= s_{i-1}^+ + (\tau_i - \tau_{i-1} + \tau_i - \tau_{i-1/2})(s_i^- - s_{i-1}^+)/\Delta\tau_{i-1} \\ &= (3s_i^- - s_{i-1}^+)/2. \end{aligned}$$

The continuity of the first derivative of f is therefore ensured if, for $i = 2, \ldots, n-1$,

$$P_{i-1}'(\tau_i) = P_i'(\tau_i)$$

or

$$(3s_i^- - s_{i-1}^+)/2 = (3s_i^+ - s_{i+1}^-)/2.$$

A reordering of the terms produces the linear system

$$\begin{aligned} &4v_i/\Delta\tau_{i-1} + 4v_{i+1}/\Delta\tau_i \\ &= (3g(\tau_i) + g(\tau_{i-1}))/\Delta\tau_{i-1} + (3g(\tau_i) + g(\tau_{i+1}))/\Delta\tau_i, \\ &\qquad i = 2, \ldots, n-1, \end{aligned} \tag{3}$$

of $n-2$ equations in the $n-1$ unknowns $v_2, \ldots, v_n$. If we now determine one of the unknowns, for example, v_2, explicitly by some additional condition, then the system (3) is uniquely solvable for $v_3, \ldots, v_n$. We abbreviate the right side of the ith equation of (3) to b_i, and then obtain simply

$$v_{i+1} = (\Delta\tau_i/4)b_i - (\Delta\tau_i/\Delta\tau_{i-1})v_i, \qquad i = 2, \ldots, n-1. \tag{4}$$

In order to understand the one remaining degree of freedom better, consider *two* interpolating parabolic splines f and $\hat{f}$ determined by (3) from the same data, but with different choices for the parameter v_2. Their difference

$$d := f - \hat{f}$$

is then a parabolic spline on $[a\,..\,b]$ that vanishes at its breaks $\tau_2, \ldots, \tau_{n-1}$ and vanishes also at a and at b. Further, from (4),

$$d(\tau_{i+1/2}) = -(\Delta\tau_i/\Delta\tau_{i-1})d(\tau_{i-1/2}) = (-)^{i-1}(\Delta\tau_i/\Delta\tau_1)d(\tau_{3/2}).$$

Therefore, for uniformly spaced data, that is, for $\tau_i = a + (i-1)h$, all i, with $h := (b-a)/(n-1)$, we have

$$(5)\quad f(x) - \hat{f}(x) = [f(\tau_{3/2}) - \hat{f}(\tau_{3/2})](-)^{i-1} p((x-\tau_i)/h) \quad \text{on } \tau_i \le x \le \tau_{i+1}$$

with

$$p(x) := 4x(1-x).$$

This shows that the choice of v_2 affects the resulting interpolant f more or less equally in the entire interval and should therefore probably not be based on some *local* datum such as the value of g at some additional site. This behavior of the difference $f(x) - \hat{f}(x)$ due to different choices of the one remaining degree of freedom also reflects the fact that the linear system (3) fails to be strictly diagonally dominant regardless of how we pick the required additional condition.

We do not encounter such difficulties if we construct the interpolant instead in the form

$$f(x) = P_i(x) \quad \text{on } \xi_i \le x \le \xi_{i+1}, \text{ for some } P_i \in \Pi_{<3},\ i = 1, \ldots, n,$$

with the *breaks* $\xi_1, \ldots, \xi_{n+1}$ chosen so that

$$\xi_1 \le a = \tau_1 < \xi_2 < \tau_2 < \xi_3 < \tau_3 < \cdots < \tau_{n-1} < \xi_n < \tau_n = b \le \xi_{n+1}.$$

For instance, we might (as first proposed, in the case of uniformly spaced data sites, by Subbotin [1967]) choose the breaks midway between the data sites,

$$\xi_i = \tau_{i-1/2} = (\tau_i + \tau_{i-1})/2, \qquad i = 2, \ldots, n.$$

The interpolation conditions impose just one condition on each polynomial piece,

$$(6)\qquad P_i(\tau_i) = g(\tau_i), \qquad i = 1, \ldots, n.$$

We are therefore free to choose two additional parameters per polynomial piece. We choose the conditions

$$(7)\qquad P_i(\xi_i) = v_i, \quad P_i(\xi_{i+1}) = v_{i+1}, \qquad i = 1, \ldots, n,$$

which then pins down each piece *and* makes the resulting f continuous, regardless of how these additional parameters, $v_1, \ldots, v_{n+1}$, are chosen. Of course, we do assume now that $\xi_1 < a$ and $b < \xi_{n+1}$.

If we choose the parameters $v_1, \ldots, v_{n+1}$ by interpolation, that is, $v_i = g(\xi_i)$, all i, then we are back to the scheme connected with Simpson's rule mentioned earlier. Instead, we chose $v_2, \ldots, v_n$ to make f again a

parabolic spline, that is, so as to make f' continuous. We derive the requisite equations for the v_i's as follows. Write P_i in Newton form,

$$P_i(x) = P_i(\xi_i) + (x - \xi_i)[\xi_i, \tau_i]P_i + (x - \xi_i)(x - \tau_i)[\xi_i, \tau_i, \xi_{i+1}]P_i.$$

From the data (6) and (7), we get the divided difference table for P_i,

$$\begin{array}{lll} \xi_i & v_i & \\ & & s_i^+ \\ \tau_i & g(\tau_i) & \qquad (s_{i+1}^- - s_i^+)/\Delta\xi_i \\ & & s_{i+1}^- \\ \xi_{i+1} & v_{i+1} & \end{array}$$

with the abbreviations

$$\begin{aligned} s_i^+ &:= [\xi_i, \tau_i]P_i = (g(\tau_i) - v_i)/(\tau_i - \xi_i), \\ s_{i+1}^- &:= [\tau_i, \xi_{i+1}]P_i = (v_{i+1} - g(\tau_i))/(\xi_{i+1} - \tau_i), \end{aligned}$$

in terms of which

$$\begin{aligned} P_i(x) &= v_i + (x - \xi_i)s_i^+ + (x - \xi_i)(x - \tau_i)(s_{i+1}^- - s_i^+)/\Delta\xi_i \\ &= c_{1,i} + (x - \xi_i)c_{2,i} + (x - \xi_i)^2 c_{3,i} \end{aligned} \tag{8}$$

where

$$c_{1,i} = v_i, \quad c_{2,i} = s_i^+ + (\xi_i - \tau_i)c_{3,i}, \quad c_{3,i} = (s_{i+1}^- - s_i^+)/\Delta\xi_i. \tag{9}$$

From (8) and (9), we obtain

$$P_i'(\xi_i) = s_i^+ + (\xi_i - \tau_i)(s_{i+1}^- - s_i^+)/\Delta\xi_i$$

and

$$\begin{aligned} P_{i-1}'(\xi_i) &= s_{i-1}^+ + (\xi_i - \xi_{i-1} + \xi_i - \tau_{i-1})(s_i^- - s_{i-1}^+)/\Delta\xi_{i-1} \\ &= s_i^- + (\xi_i - \tau_{i-1})(s_i^- - s_{i-1}^+)/\Delta\xi_{i-1}. \end{aligned}$$

The continuity of f' is therefore ensured if, for $i = 2, \ldots, n$,

$$P_i'(\xi_i) = P_{i-1}'(\xi_i)$$

or

$$s_i^+ + (\xi_i - \tau_i)(s_{i+1}^- - s_i^+)/\Delta\xi_i = s_i^- + (\xi_i - \tau_{i-1})(s_i^- - s_{i-1}^+)/\Delta\xi_{i-1}$$

or, after collecting terms,

$$\frac{\xi_i - \tau_{i-1}}{\Delta\xi_{i-1}} s_{i-1}^+ - \left(1 + \frac{\xi_i - \tau_{i-1}}{\Delta\xi_{i-1}}\right) s_i^- + \left(1 - \frac{\xi_i - \tau_i}{\Delta\xi_i}\right) s_i^+ + \frac{\xi_i - \tau_i}{\Delta\xi_i} s_{i+1}^- = 0.$$

After appropriate reordering of terms, this gives the linear system

(10)
$$(\alpha_{i-1} - \frac{1}{\Delta\xi_{i-1}})v_{i-1} + (\frac{1}{\Delta\xi_{i-1}} + \beta_{i-1} + \alpha_i + \frac{1}{\Delta\xi_i})v_i + (\beta_i - \frac{1}{\Delta\xi_i})v_{i+1}$$
$$= (\alpha_{i-1} + \beta_{i-1})g(\tau_{i-1}) + (\alpha_i + \beta_i)g(\tau_i), \qquad i = 2, \ldots, n,$$

where we have used the abbreviations

$$\alpha_i := 1/(\tau_i - \xi_i), \quad \beta_i := 1/(\xi_{i+1} - \tau_i).$$

Note that (10) is strictly column diagonally dominant, hence safely solvable for $v_2, \ldots, v_n$ by Gauss elimination *without* pivoting once we prescribe the two free parameters, v_1 and v_{n+1}.

In order to demonstrate the promised contrast to the earlier scheme of parabolic spline interpolation *at* breaks, we now consider the effect of a particular choice for v_1 and v_{n+1} on the interpolant from (10). For this, we choose the data sites again equally spaced, and choose the breaks midway between the data sites,

$$\tau_i = a + (i-1)h, \quad \xi_i = a + (i - \frac{3}{2})h, \quad \text{all } i, \qquad \text{with } h := (b-a)/(n-1).$$

One verifies directly that, on $[a\,..\,b]$, the function d_1, given by

(11)
$$d_1(x) = p((x - \xi_i)/h)/\lambda^{i-1} \text{ on } \xi_i \le x \le \xi_{i+1}, \quad \text{all } i,$$

with λ the absolutely larger solution of $\lambda^2 + 6\lambda + 1 = 0$, that is, $\lambda := -3 - \sqrt{8} = -5.828427\cdots$, and

$$p(x) := 2(\lambda+1)x^2 - (3\lambda+1)x + \lambda,$$

is a parabolic spline with breaks $\xi_2, \ldots, \xi_n$ that vanishes at all the data sites $\tau_1, \ldots, \tau_n$ and has value 1 at ξ_1. Also, d_1 *decays exponentially* as x increases, by a factor of almost 1/6 per interval $(\xi_i\,..\,\xi_{i+1})$. The function d_{n+1}, obtained by reflecting d_1 across the midpoint $(a+b)/2$, that is,

$$d_{n+1} := p((\xi_{i+1} - x)/h)/\lambda^{n+1-i} \text{ on } \xi_i \le x \le \xi_{i+1}, \quad \text{all } i,$$

is then also a parabolic spline with breaks $\xi_2, \ldots, \xi_n$ and vanishes at $\tau_1, \ldots, \tau_n$, but has the value 1 at ξ_{n+1} and decays exponentially as x decreases away from b.

If now f and $\hat{f}$ are two interpolating parabolic splines obtained from (10) from the same data but with possibly different choices for v_1 and v_{n+1}, then their difference is necessarily of the form

$$f - \hat{f} = \alpha d_1 + \beta d_{n+1}$$

with α and β satisfying

$$\begin{aligned} \alpha d_1(\xi_1) \quad &+ \beta d_{n+1}(\xi_1) \quad = \delta v_1 \quad := f(\xi_1) - \hat{f}(\xi_1), \\ \alpha d_1(\xi_{n+1}) &+ \beta d_{n+1}(\xi_{n+1}) = \delta v_{n+1} := f(\xi_{n+1}) - \hat{f}(\xi_{n+1}). \end{aligned}$$

Since $d_1(\xi_1) = d_{n+1}(\xi_{n+1}) = 1$, and $d_{n+1}(\xi_1) = d_1(\xi_{n+1}) = 1/\lambda^n$, we conclude that $\alpha \approx \delta v_1$ and $\beta \approx \delta v_{n+1}$. This shows that the choice of v_1 and v_{n+1} only affects the interpolant f "near" a and b, respectively. It therefore makes sense to derive these parameters from local considerations.

Typically, one might choose v_1 as the value of g at ξ_1. This one might do even in the limiting case when ξ_1 approaches $a = \tau_1$, in which case one ends up interpolating to g in value and slope at $a = \tau_1$. In fact, every one of the ways of choosing the additional boundary conditions in cubic spline interpolation discussed in Chapter IV is, with appropriate changes, applicable here, too.

Problems

1. Parabolic spline interpolation at midpoints, that is, at sites halfway between breaks, was introduced and thoroughly studied by Marsden [1974] from which all but (d) below are taken. However, for uniformly spaced data, the idea and the analysis go back at least to Subbotin [1967].

Let $\tau_i = (\xi_i + \xi_{i+1})/2$, $i = 1, \ldots, n$, with $\xi_1 < \cdots < \xi_{n+1}$ arbitrary, and denote the resulting parabolic spline interpolant (with $v_1 = g(\xi_1)$, $v_{n+1} = g(\xi_{n+1})$) to g by $I_3 g$.

(a) Verify that (10) simplifies to

$$\begin{gathered} \delta_i v_{i-1} + 3v_i + (1-\delta_i)v_{i+1} = 4(\delta_i g(\tau_{i-1}) + (1-\delta_i)g(\tau_i)), \\ \text{with } \delta_i := \Delta\xi_i/(\xi_{i+1} - \xi_{i-1}), \quad i = 2, \ldots, n-1. \end{gathered}$$

(b) Use the diagonal dominance of the linear system in (a) to conclude that $\max_i |v_i| \le 2 \max_i |g(\tau_i)|$ (with $\tau_0 = \xi_1$, $\tau_{n+1} = \xi_{n+1}$).

(c) Conclude that even $\|I_3 g\| \le 2\|g\|$. (Hint: With $f := I_3 g$, show that, for $\xi_i \le x \le \xi_{i+1}$,

$$(\Delta\xi_i)^2 f(x) = 2(x-\tau_i)(v_i(x-\xi_{i+1}) + v_{i+1}(x-\xi_i)) + g(\tau_i)4(x-\xi_i)(\xi_{i+1}-x)$$

(recall the Lagrange form!), then infer, using (b), that

$$|f(x)| \le (\max_j |v_j|)2|x-\tau_i|/\Delta\xi_i + |g(\tau_i)|4|x-\xi_i||\xi_{i+1}-x|/(\Delta\xi_i)^2 \le 2\|g\|.)$$

(d) Deduce from (c) and from V(21) that

$$\|g - I_3 g\| \le \frac{1}{8}|\xi|^3 \|g^{(3)}\|.$$

2. (a) Give a sequence τ for which there does not exist ξ so that $\tau_i = (\xi_i + \xi_{i+1})/2$, all i. What does that imply about the scheme in Problem 1?
(b) Simplify (10) for the case when $\xi_{i+1} = \tau_{i+1/2} := (\tau_i + \tau_{i+1})/2$, $i = 1, \ldots, n-1$, with $\xi_1 = \tau_1$, $\xi_{n+1} = \tau_n$, and $f' = g'$ at τ_1 and τ_n.
Call the resulting interpolant $\hat{I}_3 g$.
(c) Prove that $\|g - \hat{I}_3 g\| \leq \text{const}|\tau|\|g'\|$, and determine an appropriate const. (Hint: With $f = \hat{I}_3 g$, establish the tridiagonal linear system for the quantities $s_i := f'(\xi_i)$, $i = 1, \ldots, n$, then proceed as in Problem IV.7.)

3. **(Euler splines)** The parabolic spline (5) is an example of a (scaled and shifted) Euler spline (see, for example, Schoenberg [1973:Lecture 4]).

(a) The Euler polynomials $\mathrm{E}_0, \mathrm{E}_1, \mathrm{E}_2, \ldots$, are characterized by the fact that $\mathrm{E}_0(x) = 1$ and, for $n > 0$, $\mathrm{E}_n' = \mathrm{E}_{n-1}$ and $\varphi(\mathrm{E}_n) := (\mathrm{E}_n(1) + \mathrm{E}_n(0))/2 = 0$. Thus, if E_{n-1} is already known, then E_n can be generated as $\mathrm{E}_n = \mathrm{E} - \varphi(\mathrm{E})$, with $\mathrm{E}(x) := \int_0^x \mathrm{E}_{n-1}(t)\,dt$. Verify that

$$\mathrm{E}_1(x) = x - 1/2, \quad \mathrm{E}_2(x) = x^2/2 - x/2, \quad \mathrm{E}_3(x) = x^3/6 - x^2/4 + 1/24,$$

and construct E_4.
(b) Let $\hat{\mathrm{E}}_n$ be the extension of E_n on $[0\,..\,1]$ to all of $\mathbb{R}$ via the functional equation $\hat{\mathrm{E}}_n(x+1) = -\hat{\mathrm{E}}_n(x)$, that is,

$$\hat{\mathrm{E}}_n(x) := (-)^{\lfloor x \rfloor} \hat{\mathrm{E}}_n(x - \lfloor x \rfloor).$$

Prove that $\hat{\mathrm{E}}_n$ has $n-1$ continuous derivatives.
$\hat{\mathrm{E}}_n$ is a close cousin of Schoenberg's Euler spline $\mathcal{E}_n$. Precisely, $\mathcal{E}_n = \hat{\mathrm{E}}_n/\mathrm{E}_n(0)$ for odd n, and $\mathcal{E}_n = \hat{\mathrm{E}}_n(\cdot - \frac{1}{2})/\mathrm{E}_n(\frac{1}{2})$ for even n.
(c) Verify (by induction on n) that $\mathrm{E}_n(x)$ vanishes at $\frac{1}{2}$ if n is odd and at 0 if n is even (and positive) and nowhere else in $[0\,..\,1)$. Conclude that $\hat{\mathrm{E}}_n$ is strictly monotone between its extreme values that occur at the integers if n is odd and at the half integers if n is even. In particular,

$$\|\hat{\mathrm{E}}_n\| = \begin{cases} |\mathrm{E}_n(0)|, & n \text{ odd}; \\ |\mathrm{E}_n(\frac{1}{2})|, & n \text{ even}. \end{cases}$$

4. The parabolic spline (11) is an example of a (scaled and shifted) **exponential Euler spline** (see, for example, Schoenberg [1973:Lecture 4]). Let $\lambda \neq 1$ be a number, real or complex. The λ-Euler polynomials $\mathrm{E}_0^\lambda, \mathrm{E}_1^\lambda, \mathrm{E}_2^\lambda, \ldots$ are characterized by the fact that $\mathrm{E}_0^\lambda(x) = 1$, and for $n > 0$, $(\mathrm{E}_n^\lambda)' = \mathrm{E}_{n-1}^\lambda$ and $\varphi_\lambda(\hat{\mathrm{E}}_n) := (\mathrm{E}_n^\lambda(1) - \lambda \mathrm{E}_n^\lambda(0))/(1-\lambda) = 0$. Thus, for $\lambda = -1$, we get again the Euler polynomials of Problem 3.

Let $\hat{\mathrm{E}}_n^\lambda$ be the extension of E_n^λ on $[0\,..\,1]$ to all of $\mathbb{R}$ by the functional equation $\hat{\mathrm{E}}_n^\lambda(x+1) = \lambda \hat{\mathrm{E}}_n^\lambda(x)$, that is,

$$\hat{\mathrm{E}}_n^\lambda(x) := \lambda^{\lfloor x \rfloor} \mathrm{E}_n^\lambda(x - \lfloor x \rfloor).$$

(a) Prove that $\hat{E}_n^\lambda$ has $n-1$ continuous derivatives.
(b) Construct $\hat{E}_2^\lambda$ in general, then determine λ so that $\hat{E}_2^\lambda(\frac{1}{2}) = 0$ and compare the resulting exponential Euler spline $\hat{E}_2^\lambda$ with the function d_1 in (11).
(c) Construct also E_3^λ in general, then determine λ so that $E_3^\lambda(0) = 0$. There should be two solutions, λ_1 and λ_2, with $\lambda_2 < -1 < \lambda_1$ and $\lambda_1 = 1/\lambda_2$.

5. (V(13) and V(21) are sharp). Consider complete cubic spline interpolation at the sites $\tau_i = i$, $i = 1, \ldots, n$ (with $\tau_0 = 1$, $\tau_{n+1} = n$). Let $d_0 := \hat{E}_3^{\lambda_1}$ (with λ_1 as determined in Problem 4(c)) and set $d_{n+1}(x) := -d_0(n+1-x)$.
(a) Verify that d_0, $d_{n+1} \in \$_4$ and that $d_0'(1) = d_{n+1}'(n) \neq 0$, $d_0'(n) = d_{n+1}'(1) = d_0'(1)\lambda_1^{n-1}$. Conclude that, with $\hat{E}_4$ the Euler spline of Problem 3, the system

$$\alpha d_0' + \beta d_{n+1}' = \hat{E}_4' \quad \text{at } 1 \text{ and } n$$

has exactly one solution and that, for large n,

$$(-)^{n-1}\beta = \alpha \approx \hat{E}_4'(1)/d_0'(1).$$

(b) Conclude that $I_4(\hat{E}_4) = \alpha d_0 + \beta d_{n+1}$.
(c) Let $n = 2r$. Verify that, on $[r \,..\, r+1]$, and for $g = \hat{E}_4$,

$$g - I_4 g = \hat{E}_4 + \mathcal{O}(\lambda_1^r),$$

while $\|\hat{E}_4\|/\|\hat{E}_4^{(4)}\| = 5/384$ and $\|\hat{E}_4'\|/\|\hat{E}_4'\| = 1/24$. Infer (by letting $n \to \infty$) that the constants in V(13) and V(21) are best possible.

6. Write the complete cubic spline interpolant in Lagrange form,

$$I_4 g = g'(\tau_1)C_0 + \sum_{i=1}^{n} g(\tau_i)C_i + g'(\tau_n)C_{n+1},$$

with C_i the cubic spline that vanishes at τ_j for all $j \neq i$ and has the value 1 at τ_i, $i = 1, \ldots, n$ and C_0, C_{n+1} analogously defined.
(a) Construct C_i (for example, by CUBSPL) for a *uniform* τ and verify numerically that

$$C_i(x) \approx C((x - \tau_i)/h)$$

with $h := |\tau|$ and C the cubic spline with breaks at the integers and of the form

$$C(x) := \begin{cases} \alpha \hat{E}_3^{\lambda_1}(x), & 1 < x; \\ \alpha \hat{E}_3^{\lambda_1}(x) + (1-x)^3, & 0 \le x \le 1; \\ C(-x), & x < 0. \end{cases}$$

Here, $\alpha := 3/(E_3^{\lambda_1})'(0)$, and $\hat{E}_3^{\lambda_1}$ is the exponential Euler spline determined in Problem 4. In particular, verify that the function C is a cubic spline,

that is, a piecewise cubic function with two continuous derivatives.
(b) Conclude that a satisfactory approximation to the cubic spline interpolant $I_4 g$ at a *uniform* τ can be obtained by the *local* scheme

$$(I_4^{\mathrm{loc}} g)(x) := \sum_{j-r<i\le j+r} g(\tau_i) C\left(\frac{x-\tau_i}{h}\right), \qquad \text{for } \tau_j \le x \le \tau_{j+1},$$

(see Buneman [1973]). Specifically, show that $r = 4$ gives about 1% accuracy.

7. Develop such a local scheme for parabolic spline interpolation of Problem 1.

VII

A Representation for Piecewise Polynomial Functions;

PPVALU, INTERV

Experience has shown that piecewise polynomial functions of order higher than four supply at times much more efficient approximations than do piecewise parabolic or piecewise cubic functions, especially when the breaks are chosen properly. For this reason, we discuss in this chapter ways to represent piecewise polynomial functions of arbitrary order in a computer.

We begin with a formal definition.

Piecewise polynomial functions Let $\xi := (\xi_i)_1^{l+1}$ be a strictly increasing sequence of points, and let k be a positive integer. If $P_1, \ldots, P_l$ is any sequence of l polynomials, each of order k (that is, of degree $< k$), then we define the corresponding **piecewise polynomial** or **pp function** f **of order** k by the prescription

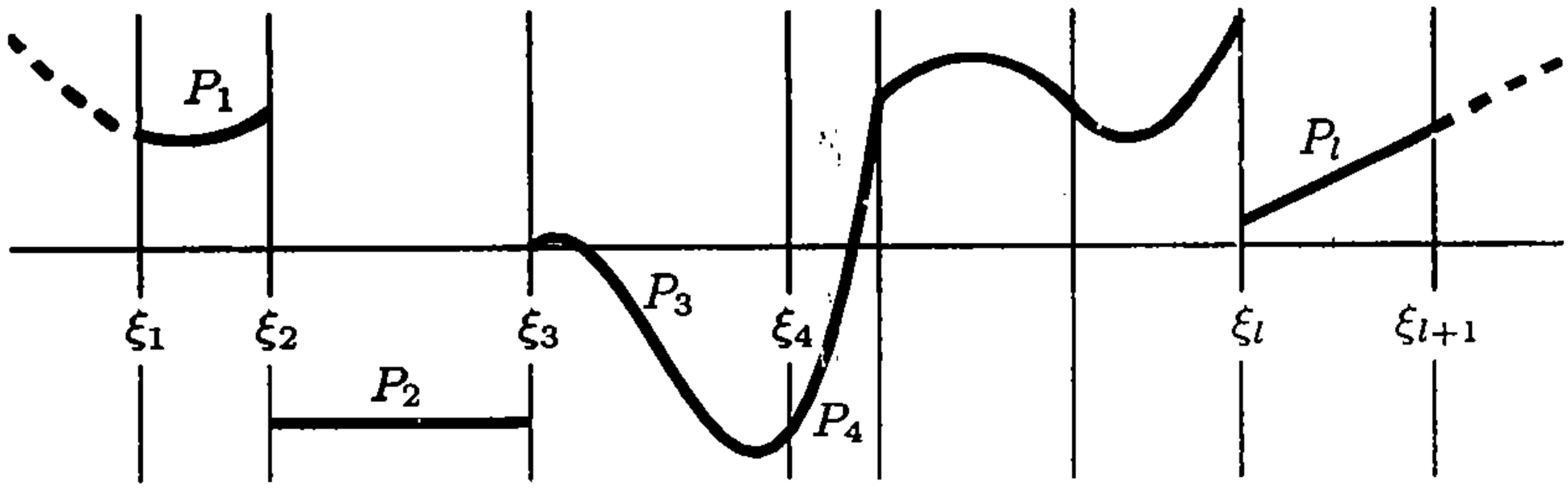
$$(1) \qquad f(x) := P_i(x) \quad \text{if } \xi_i < x < \xi_{i+1}; \quad i = 1, \ldots l.$$

The points ξ_i are called the **breaks** (or, **breakpoints**) of f. Whenever convenient, we think of such a function f as defined on the whole real line $\mathbb{R}$ by extension of the first and the last piece, that is,

$$(2) \qquad f(x) := \begin{cases} P_1(x), & \text{if } x \le \xi_1; \\ P_l(x), & \text{if } \xi_{l+1} \le x. \end{cases}$$

With this, ξ_1 and ξ_{l+1} are, strictly speaking, not breaks, and the pp function f could be defined without any reference to them. We retain these two points nevertheless as part of the specification of a pp function since they specify the interval on which such a pp function is originally defined, and which we will refer to as the **basic interval** for that pp function. Also, we use the point ξ_1 in the "ppform" to be described below.

It should be noted that (2) amounts to *extrapolation*. Therefore, away from the original interval $[\xi_1 \,..\, \xi_{l+1}]$, f, as extended by (2), may reflect



(3) FIGURE. A pp function of some order, with $l = 7$.

the original intent of its construction as badly as would the product of any other extrapolation technique. To reiterate, we use (2) for convenience and not because of any inherent quality or truth.

At the (interior) breaks $\xi_2, \ldots, \xi_l$, the function f is as yet undefined. In a sense, the pp function f has *two* values at such a site, namely the value $f(\xi_i^-) = P_{i-1}(\xi_i)$ it gets from the left piece and the value $f(\xi_i^+) = P_i(\xi_i)$ it gets from the right piece. For definiteness and in order to obtain a (single-valued) function, the programs below *arbitrarily* choose to make f continuous from the right, that is,

$$f(\xi_i) := f(\xi_i^+), \quad \text{for } i = 2, \ldots, l. \tag{4}$$

We will continue nevertheless to think of the pp function f as having two values at each break. Of course, this becomes a moot point in case the function f is continuous. But, unless f consists of just one polynomial, some derivative of f is discontinuous and is a pp function so that this point has to be discussed and settled in any case.

It follows that *two pp functions agree iff they consist of the same polynomial pieces, broken at the same sites.* Of course, if one is made right-continuous and the other left-continuous, then they may not agree at the breaks, but we will nevertheless think of them as being the same pp function.

We denote the collection of all such pp functions of order k with break sequence $\xi = (\xi_i)_1^{l+1}$ by

$$\Pi_{<k,\xi}.$$

It is clear that $\Pi_{<k,\xi}$ is a linear space. Its dimension is kl since each of its elements consists of l polynomial pieces and each polynomial piece has k freely choosable polynomial coefficients (see Problem 4). Abstractly, $\Pi_{<k,\xi}$ is the direct sum of l copies of $\Pi_{<k}$.

As already implied by an earlier remark, we consider the jth derivative

$$D^j f$$

of the pp function f to be the pp function of order $k - j$ with the same break sequence and put together from the jth derivatives of the polynomial pieces that make up f. This definition avoids a lot of fancy mathematical footwork when it comes to a discussion of the derivatives of a pp function at a break. In return, the definition has to be treated with care in the context of the fundamental theorem of calculus.

(5) Proposition. *The pp function f satisfies*

$$f(x) - f(a) = \int_a^x (Df)(t)\,\mathrm{d}t \quad \text{for all } x$$

if and only if f is a continuous function.

The first derivative Df of a piecewise constant function f, for example, is identically zero by our definition, therefore equal to the usual derivative of the function f if and only if f is actually a constant function.

A pp function can be represented in a computer in a variety of ways. If such a function f and some of its derivatives are to be evaluated at many sites (for graphing purposes, say), then the following representation seems most convenient and efficient:

(6) Definition. *The* **ppform** *for $f \in \Pi_{<k,\xi}$ consists of*

(i) the integers k and l, giving order and number of its pieces, respectively;

(ii) the strictly increasing sequence $\xi_1, \xi_2, \ldots, \xi_{l+1}$ of its breaks; and

(iii) the matrix $C = (C_{ji})_{j=1}^{k};{}_{i=1}^{l}$ of its right derivatives at the breaks, that is, the numbers

$$C_{ji} := D^{j-1} f(\xi_i^+), \quad j = 1, \ldots, k; \quad i = 1, \ldots, l.$$

In terms of these numbers, the value of the jth derivative $D^j f$ of f at a site x is found (in PPVALU) as

$$D^j f(x) = \sum_{m=j}^{k-1} C_{m+1,i}(x - \xi_i)^{m-j}/(m-j)!, \tag{7}$$

where (with the earlier conventions) i is the integer such that

$$\begin{array}{rll} \text{either} : i = 1 & \text{and} & x < \xi_2 \\ \text{or} \quad : 1 < i < l & \text{and} & \xi_i \le x < \xi_{i+1} \\ \text{or} \quad : i = l & \text{and} & \xi_l \le x. \end{array} \tag{8}$$

To be sure, the ppform in the SPLINE TOOLBOX (de Boor [1990]$_2$) stores differently normalized coefficients and stores them differently, in part because, as mentioned at the beginning of Chapter I, MATLAB stores poly-

nomial coefficients in order from highest to lowest. Explicitly, if c is the coefficient array in the ppform of some pp function f of order k generated in the SPLINE TOOLBOX (de Boor [1990]$_2$) and with break sequence ξ, then

$$f(x) = \sum_{j=1}^{k} \mathrm{c}(i,j)(x-\xi_i)^{k-j}, \quad \text{for } \xi_i \le x < \xi_{i+1}.$$

The following MATLAB statement would convert such an array c into the corresponding array C needed in this book's ppform:

```
C = (c(:,k:-1:1).').*repmat(cumprod([1 1:k-1]).',1,1);
```

The subroutine PPVALU It follows that a FORTRAN program for the evaluation of a pp function requires as *input* the integers k and l, some one-dimensional array BREAK containing ξ and some two-dimensional array COEF containing the matrix C. Here is such a function subprogram, for the evaluation of a pp function or its derivatives according to (7).

```
      REAL FUNCTION PPVALU (BREAK, COEF, L, K, X, JDERIV )
CALLS  INTERV
CALCULATES VALUE AT  X  OF  JDERIV-TH DERIVATIVE OF PP FCT FROM PP-REPR.
C
C******  I N P U T  ******
C  BREAK, COEF, L, K.....FORMS THE PP-REPRESENTATION OF THE FUNCTION  F
C        TO BE EVALUATED. SPECIFICALLY, THE J-TH DERIVATIVE OF  F  IS
C        GIVEN BY
C
C     (D**J)F(X) = COEF(J+1,I) + H*(COEF(J+2,I) + H*( ... (COEF(K-1,I) +
C                                + H*COEF(K,I)/(K-J-1))/(K-J-2) ... )/2)/1
C
C
C        WITH  H = X - BREAK(I),  AND
C
C       I  =  MAX( 1 , MAX( J ,  BREAK(J) .LE. X , 1 .LE. J .LE. L ) ).
C
C  X.....THE POINT AT WHICH TO EVALUATE.
C  JDERIV.....INTEGER GIVING THE ORDER OF THE DERIVATIVE TO BE EVALUAT-
C        ED.  A S S U M E D  TO BE ZERO OR POSITIVE.
C
C******  O U T P U T  ******
C  PPVALU.....THE VALUE OF THE (JDERIV)-TH DERIVATIVE OF  F  AT  X.
C
C******  M E T H O D  ******
C     THE INTERVAL INDEX  I , APPROPRIATE FOR  X , IS FOUND THROUGH A
C  CALL TO  INTERV . THE FORMULA ABOVE FOR THE  JDERIV-TH DERIVATIVE
C  OF  F  IS THEN EVALUATED (BY NESTED MULTIPLICATION).
C
      INTEGER JDERIV,K,L,   I,M,NDUMMY
      REAL BREAK(L+1),COEF(K,L),X,   FMMJDR,H
      PPVALU = 0.
      FMMJDR = K - JDERIV
C           DERIVATIVES OF ORDER  K  OR HIGHER ARE IDENTICALLY ZERO.
      IF (FMMJDR .LE. 0.)                  GO TO 99
C
```

```
C               FIND INDEX  I  OF LARGEST BREAKPOINT TO THE LEFT OF  X .
      CALL INTERV ( BREAK, L+1, X, I, NDUMMY )
C
C       EVALUATE  JDERIV-TH DERIVATIVE OF  I-TH POLYNOMIAL PIECE AT  X .
      H = X - BREAK(I)
      M = K
    9    PPVALU = (PPVALU/FMMJDR)*H + COEF(M,I)
         M = M - 1
         FMMJDR = FMMJDR - 1.
         IF (FMMJDR .GT. 0.)              GO TO 9
   99                                     RETURN
      END
```

The choice of the derivatives $C_{ji} = D^{j-1}f(\xi_i^+)$ rather than the usual polynomial coefficients $c_{ji} := D^{j-1}f(\xi_i^+)/(j-1)!$ in the ppform allows for a uniform treatment of the function f and all its derivatives during evaluation. If one is merely interested in values of f, then it would be more efficient to have available the polynomial coefficients c_{ji} since this would avoid the formation of, and division by, the numbers $1, \ldots, k-1$ when evaluating the function f at a site (thus cutting the work in two). On the other hand, evaluation of the jth derivative for $j > 0$ would become much trickier since

$$D^j f(x) = \sum_{m=j}^{k-1} c_{m+1,i}(x-\xi_i)^{m-j} m!/(m-j)!, \quad \text{for } \xi_i \le x < \xi_{i+1}.$$

One would compute the value of f from the coefficients CSMALL(j,I) $= c_{j,\mathrm{I}}$ simply by

```
      DO 8   M=K,1,-1
    8        PPVALU = PPVALU*H + CSMALL(M,I)
```

and the value of the first derivative Df of f by

```
      DO 9   M=K-1,1,-1
    9        PPVALU = PPVALU*H + FLOAT(M)*CSMALL(M+1,I)
```

and these would be special cases. For the general case of computing $D^{\mathrm{JDERIV}}f(\mathrm{X})$ for JDERIV > 1, the 10-loop in PPVALU would have to be changed to something like the following:

```
      DO 10 M=K,JDERIV+1,-1
         PPVALU = (PPVALU/FMMJD)*FLOAT(M)*H + CSMALL(M,I)
   10    FMMJD = FMMJD - 1.
      DO 11 M=2,JDERIV
   11    PPVALU = PPVALU*FLOAT(M)
```

PPVALU uses a subroutine INTERV, given below, to place the argument X within the break sequence. Some people have objected to the use of INTERV in PPVALU, contending that the added efficiency achieved is not worth being

saddled with a routine as complex as INTERV appears to be. A possible alternative would be to carry out bisection (or, binary search) directly in PPVALU. One could even bring in one of the features of INTERV, namely retain the interval index I from one call to the next and first test whether I < L and BREAK(I) ≤ X < BREAK(I + 1), going into the binary search only if this test fails. I have nevertheless kept INTERV since the problem it solves comes up repeatedly later on and since I cannot see how its complexity could be of any concern to the user, as long as it is efficient.

It may happen, though, that the interval index I appropriate for a given X is obvious from the context, in which case I would make use of the routine PVALUE, discussed in Problem 1, instead.

The subroutine INTERV Here is a routine for determining the interval index I for X with respect to the break sequence ξ.

```
      SUBROUTINE INTERV ( XT, LXT, X, LEFT, MFLAG )
COMPUTES  LEFT = MAX( I :  XT(I) .LT. XT(LXT) .AND.  XT(I) .LE. X )  .
C
C******  I N P U T  ******
C  XT.....A REAL SEQUENCE, OF LENGTH  LXT , ASSUMED TO BE NONDECREASING
C  LXT.....NUMBER OF TERMS IN THE SEQUENCE  XT .
C  X.....THE POINT WHOSE LOCATION WITH RESPECT TO THE SEQUENCE  XT  IS
C        TO BE DETERMINED.
C
C******  O U T P U T  ******
C  LEFT, MFLAG.....BOTH INTEGERS, WHOSE VALUE IS
C
C     1      -1       IF                    X .LT.  XT(1)
C     I       0       IF   XT(I)   .LE. X .LT. XT(I+1)
C     I       0       IF   XT(I)   .LT. X .EQ. XT(I+1) .EQ. XT(LXT)
C     I       1       IF   XT(I)   .LT.       XT(I+1) .EQ. XT(LXT) .LT. X
C
C        IN PARTICULAR,  MFLAG = 0  IS THE 'USUAL' CASE.  MFLAG .NE. 0
C        INDICATES THAT  X  LIES OUTSIDE THE CLOSED INTERVAL
C        XT(1) .LE. Y .LE. XT(LXT) . THE ASYMMETRIC TREATMENT OF THE
C        INTERVALS IS DUE TO THE DECISION TO MAKE ALL PP FUNCTIONS CONT-
C        INUOUS FROM THE RIGHT, BUT, BY RETURNING  MFLAG = 0  EVEN IF
C        X = XT(LXT), THERE IS THE OPTION OF HAVING THE COMPUTED PP FUNCTION
C        CONTINUOUS FROM THE LEFT AT  XT(LXT) .
C
C******  M E T H O D  ******
C  THE PROGRAM IS DESIGNED TO BE EFFICIENT IN THE COMMON SITUATION THAT
C  IT IS CALLED REPEATEDLY, WITH  X  TAKEN FROM AN INCREASING OR DECREA-
C  SING SEQUENCE. THIS WILL HAPPEN, E.G., WHEN A PP FUNCTION IS TO BE
C  GRAPHED. THE FIRST GUESS FOR  LEFT  IS THEREFORE TAKEN TO BE THE VAL-
C  UE RETURNED AT THE PREVIOUS CALL AND STORED IN THE  L O C A L  VARIA-
C  BLE  ILO . A FIRST CHECK ASCERTAINS THAT  ILO .LT. LXT (THIS IS NEC-
C  ESSARY SINCE THE PRESENT CALL MAY HAVE NOTHING TO DO WITH THE PREVI-
C  OUS CALL). THEN, IF  XT(ILO) .LE. X .LT. XT(ILO+1), WE SET  LEFT =
C  ILO  AND ARE DONE AFTER JUST THREE COMPARISONS.
C     OTHERWISE, WE REPEATEDLY DOUBLE THE DIFFERENCE  ISTEP = IHI - ILO
C  WHILE ALSO MOVING  ILO  AND  IHI  IN THE DIRECTION OF  X , UNTIL
C                      XT(ILO) .LE. X .LT. XT(IHI) ,
C  AFTER WHICH WE USE BISECTION TO GET, IN ADDITION, ILO+1 = IHI .
C  LEFT = ILO  IS THEN RETURNED.
C
```

```
      INTEGER LEFT,LXT,MFLAG,    IHI,ILO,ISTEP,MIDDLE
      REAL X,XT(LXT)
      DATA ILO /1/
      SAVE ILO
      IHI = ILO + 1
      IF (IHI .LT. LXT)                    GO TO 20
         IF (X .GE. XT(LXT))               GO TO 110
         IF (LXT .LE. 1)                   GO TO 90
         ILO = LXT - 1
         IHI = LXT
C
   20 IF (X .GE. XT(IHI))                  GO TO 40
         IF (X .GE. XT(ILO))               GO TO 100
C
C                ***** NOW X .LT. XT(ILO) . DECREASE  ILO  TO CAPTURE  X .
         ISTEP = 1
   31       IHI = ILO
            ILO = IHI - ISTEP
            IF (ILO .LE. 1)                GO TO 35
            IF (X .GE. XT(ILO))            GO TO 50
            ISTEP = ISTEP*2
                                           GO TO 31
   35    ILO = 1
         IF (X .LT. XT(1))                 GO TO 90
                                           GO TO 50
C                ***** NOW X .GE. XT(IHI) . INCREASE  IHI  TO CAPTURE  X .
   40    ISTEP = 1
   41       ILO = IHI
            IHI = ILO + ISTEP
            IF (IHI .GE. LXT)              GO TO 45
            IF (X .LT. XT(IHI))            GO TO 50
            ISTEP = ISTEP*2
                                           GO TO 41
   45    IF (X .GE. XT(LXT))               GO TO 110
         IHI = LXT
C
C              **** NOW XT(ILO) .LE. X .LT. XT(IHI) . NARROW THE INTERVAL.
   50    MIDDLE = (ILO + IHI)/2
         IF (MIDDLE .EQ. ILO)              GO TO 100
C     NOTE. IT IS ASSUMED THAT MIDDLE = ILO IN CASE IHI = ILO+1 .
         IF (X .LT. XT(MIDDLE))            GO TO 53
            ILO = MIDDLE
                                           GO TO 50
   53       IHI = MIDDLE
                                           GO TO 50
C**** SET OUTPUT AND RETURN.
   90    MFLAG = -1
         LEFT = 1
                                           RETURN
  100    MFLAG = 0
         LEFT = ILO
                                           RETURN
  110    MFLAG = 1
         IF (X .EQ. XT(LXT)) MFLAG = 0
         LEFT = LXT
  111    IF (LEFT .EQ. 1)                  RETURN
            LEFT = LEFT - 1
            IF (XT(LEFT) .LT. XT(LXT))     RETURN
                                           GO TO 111
         END
```

The problem of locating a site within an increasing sequence of sites is solved quite differently in the SPLINE TOOLBOX (de Boor [1990]$_2$), in

part because there one wants to take advantage of the fact that MATLAB provides *vector arithmetic*, hence encourages the 'simultaneous' evaluation of a function at all the sites in a given sequence (or matrix). This therefore requires the determination of the correct index *sequence* I for given site sequence X with respect to the break sequence XI. This is accomplished (in `sorted`) by computing first

```
[ignored,index] = sort([XI(1:L) X]);
```

which supplies, in `ignored`, the entries of the concatenated sequence s :=(`XI(1:L),X`) in nondecreasing order. More importantly, it supplies, in `index`, the information of how to reorder the sequence s to obtain that nondecreasing sequence `ignored`. Precisely, `ignored` = s(`index`). For example, if `XI(1:L)` = (1,2,3) while `X` = (0,1,2,3,4), then the above command produces

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

Since `XI(1:L)` comes first in s, we know that, in the sequence `index`, the numbers `1,...,L` refer to entries of `XI`, while the numbers greater than `L` refer to entries of `X`. Hence, the `MATLAB` command `j=find(index>L)` returns the positions in the sorted sequence of the entries from `X`. So, assuming `X` itself to be ordered, the only reason why `X(i)` would not occur in position `i` in the sorted sequence is because there are entries from `XI` to the left of it. In fact, the difference, `j(i)-i`, of the position of `X(i)` in the sorted sequence and `i`, equals the number of entries from `XI` to the left of it. Therefore, `XI(j(i)-i)` is the breakpoint closest to `X(i)` from the left, hence `j(i)-i` is the index we are looking for.

In our example, the corresponding calculation

```
I = find(index>L) - (1:length(X));
```

gives `I = 0  1  2  3  3`, which is exactly the sequence we want for this example, – except for that initial `0` which tells us that there are 0 entries of `XI(1:L)` to the left of `X(1)`, that is, `X(1) < XI(1)`. For such `X(1)`, we want to use the first break interval, and we can achieve that using the `max` function to ensure that we get at least the value 1. Altogether, this gives the simple command sequence

```
[ignored,index] = sort([XI(1:L) X]);
I = max(find(index>L) - (1:length(X)) , 1 );
```

which, for our example, produces the desired `I = 1  1  2  3  3`.

Problems

1. Write an abbreviated version

```
FUNCTION PVALUE( XLEFT, COEF, K, X, JDERIV )
```

of PPVALU that returns the JDERIV-th derivative at X of the function f given by

$$f(x) := \sum_{m=1}^{k} \texttt{COEF}(m)(x - \texttt{XLEFT})^{m-1}/(m-1)! \quad .$$

How would you use it in place of PPVALU to evaluate a pp function at some x, given the ppform for f in BREAK, COEF, L, K, in case you know already the interval index I appropriate for x?

2. Consider the problem of making pp functions *continuous from the left.* Possible solutions:

(i) Write a subprogram

```
SUBROUTINE INTRVL ( XT, LXT, K, LEFT, MFLAG )
```

that returns LEFT = $\min\{\,\texttt{LXT}, \min\{\,j : 1 \le j \le \texttt{LXT}, \texttt{X} \le \texttt{XT}(j)\,\}\,\}$, and use it in PPVALU in place of INTERV.

(ii) Continue to use INTERV in PPVALU, but decrease I by one in case X = BREAK(I) and I ≤ 2.

Both (i) and (ii) require, in effect, a modified version of PPVALU that we will call, for later reference,

```
FUNCTION PPVLLC ( BREAK, COEF, L, K, X, JDERIV )
```

(iii) Give up on PPVALU and PPVLLC altogether, but evaluate f by *two* subroutine calls:

```
CALL IOFX ( BREAK, I, X, I, MFLAG )
FX = PVALUE (BREAK(I), COEF(1,I), K, X, JDERIV )
```

with PVALUE as constructed in Problem 1, and IOFX equal to INTERV or INTRVL depending on what is wanted.

Discuss the relative advantages and disadvantages of these approaches and propose, perhaps, alternatives of your own, given that a pp function is to be considered, in the *same* program, at times left-continuous and at times right-continuous.

3. Write a

```
FUNCTION PCVALU ( BREAK, COEF, L, K, X )
```

that would use Problem I.7 to evaluate a pp function from its *piecewise Chebyshev form*, contained in BREAK, COEF, L, K. (This corresponds to the ppform for f except that now COEF(·,I) contains the Chebyshev coefficients for the I-th polynomial piece.) How would you construct the more

general routine that would also return the value of some derivative if desired?

4. Prove that, with $\xi_1 < \cdots < \xi_{l+1}$, $\Pi_{<k,\xi}$ is a linear space of dimension

$$n := kl.$$

This is not really a problem for anyone familiar with basic linear algebra. The proof requires the following (standard) steps:

(a) Verify that $\Pi_{<k,\xi}$ is a linear space. (You may take as given that the collection $\mathbb{R}^{\mathbb{R}}$ of all real-valued functions on $\mathbb{R}$ is a linear space if vector addition and scalar multiplication are defined *pointwise*, that is, $(f+g)(x) := f(x)+g(x)$, $(\alpha f)(x) := \alpha(f(x))$, all $x \in \mathbb{R}$, all functions f, g, all $\alpha \in \mathbb{R}$. Then it is only necessary to show that $\Pi_{<k,\xi}$ is a subspace of $\mathbb{R}^{\mathbb{R}}$, that is, $\Pi_{<k,\xi}$ is nonempty and is closed under addition and scalar multiplication: $f, g \in \Pi_{<k,\xi}$, $\alpha \in \mathbb{R}$ implies $f+g, \alpha f \in \Pi_{<k,\xi}$.)

(b) Verify that $\dim \Pi_{<k,\xi} \le n = kl$. (This requires you to show that $\Pi_{<k,\xi}$ is generated by n (or fewer) functions, that is, you must exhibit functions $\varphi_1, \ldots, \varphi_n$, all in $\Pi_{<k,\xi}$, with the property that every f in $\Pi_{<k,\xi}$ can be written as a linear combination of them, that is, $f = \sum_i \alpha_i \varphi_i$ for some suitable coefficients (α_i).) Note: it is possible to combine (a) and (b) by showing that $\Pi_{<k,\xi}$ consists of exactly all possible linear combinations of certain n functions.

(c) Verify that $\dim \Pi_{<k,\xi} \ge n$. (This requires you to exhibit a function sequence $\varphi_1, \ldots, \varphi_n$ that is linearly independent, that is, for which $\sum \alpha_i \varphi_i = 0$ is possible only if $\alpha_1 = \cdots = \alpha_n = 0$. Such linear independence is invariably shown by exhibiting a corresponding sequence $\lambda_1, \ldots, \lambda_n$ of linear functionals for which the matrix $(\lambda_i \varphi_j : i, j = 1, \ldots, n)$ is obviously invertible, that is, is triangular with nonzero diagonal terms. For the case at hand, you might consider the linear functionals λ given by $\lambda f := f^{(r)}(\xi_s^+)$.)

The bonus of all this work is a basis for the space, that is, a sequence $\varphi_1, \ldots, \varphi_{kl}$ that is both generating and linearly independent.

5. Prove Proposition (5).

VIII

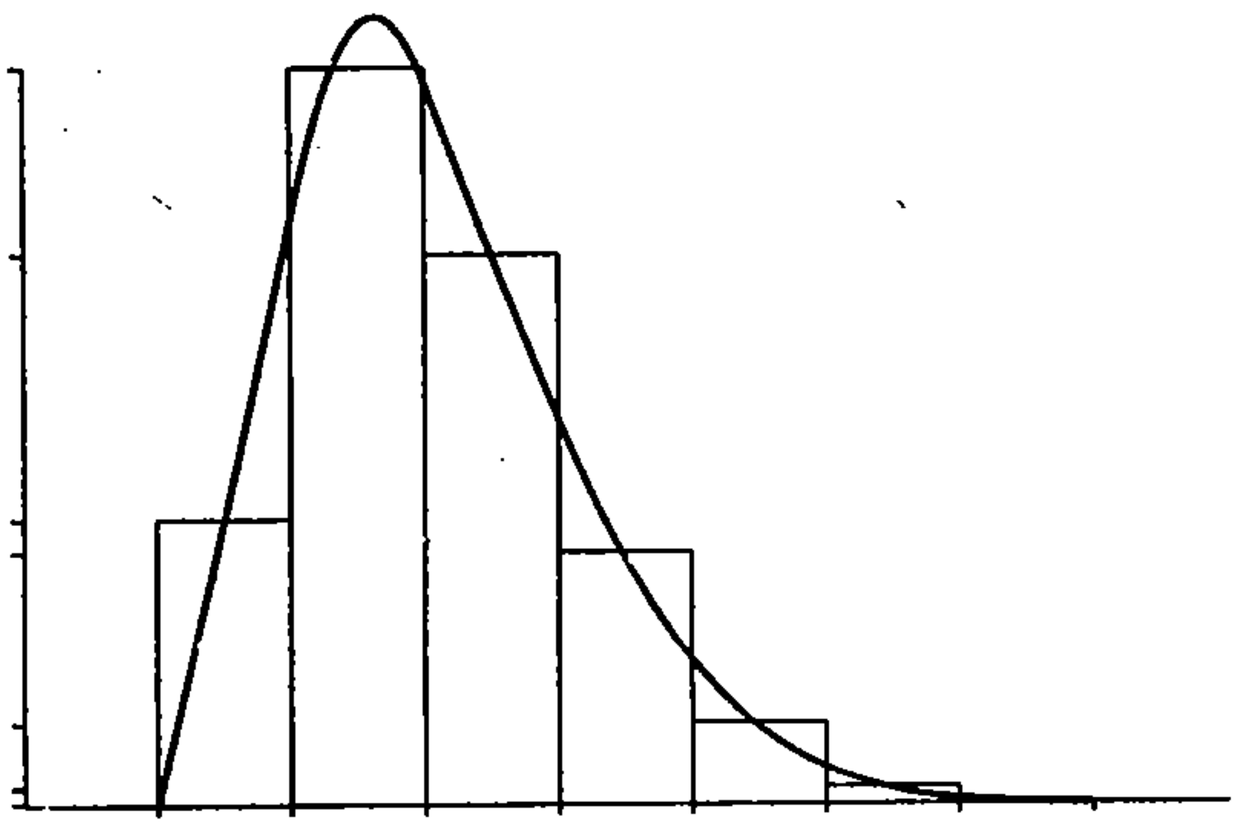
The Spaces $\Pi_{<k,\xi,\nu}$ and the Truncated Power Basis

The typical computational problem involving pp functions can be stated as follows: We are given some information about a certain function g and are required to construct a particular function f in $\Pi_{<k,\xi}$ that satisfies the same conditions g is known to satisfy (see Chapter XIII– XVII for examples). In addition, the function f is to have a certain number of continuous derivatives. Formalization of these latter, homogeneous conditions leads to the subspaces $\Pi_{<k,\xi,\nu}$ of $\Pi_{<k,\xi}$, and computational efficiency demands the construction of a convenient basis for these subspaces.

In this chapter, we introduce the somewhat popular truncated power basis for the space $\Pi_{<k,\xi,\nu}$ and point out its failings.

We begin with an example.

(1) Example: The smoothing of a histogram by parabolic splines



(2) FIGURE. Parabolic "area matching" spline approximation to a histogram.

Some people object to histograms or bar graphs and would draw a smooth

curve through them as a more satisfying representation of the underlying distribution g whose sampling produces the histogram. One can argue the point whether anything (other than æsthetic satisfaction) is gained by the procedure. But, suppose we are given such a histogram and wish to smooth it. This means that we are given sites

$$\tau_1 < \tau_2 < \cdots < \tau_{n+1}$$

and (usually nonnegative) numbers $h_1, h_2, \ldots, h_n$, with h_i the height over the open interval $(\tau_i \,..\, \tau_{i+1})$, all i. The usual interpretation of these numbers is that $h_i \Delta\tau_i$ is (approximately) equal to the integral of the underlying distribution g over the interval $[\tau_i \,..\, \tau_{i+1}]$ (recall that $\Delta\tau_i := \tau_{i+1} - \tau_i$). It therefore makes sense to demand of our smooth version f that it satisfy the "interpolation" conditions

$$\int_{\tau_i}^{\tau_{i+1}} f(x)\,\mathrm{d}x = h_i \Delta\tau_i, \qquad i = 1, \ldots, n.$$

We choose the function f to be a parabolic spline, that is, a continuous pp function of order 3 with a continuous first derivative,

$$f \in \Pi_{<3,\boldsymbol{\xi}} \cap C^{(1)},$$

and choose the break sequence $\boldsymbol{\xi}$ to coincide with the sequence $\boldsymbol{\tau}$. If the underlying distribution g is smooth and vanishes outside the interval $[\tau_1 \,..\, \tau_{n+1}]$, then we would have $g^{(j)}(\tau_1) = g^{(j)}(\tau_{n+1}) = 0$ for $j = 0, 1, \ldots$ to the extent of the smoothness of the distribution g. So we impose the additional interpolation conditions

$$f(\tau_1) = f(\tau_{n+1}) = 0.$$

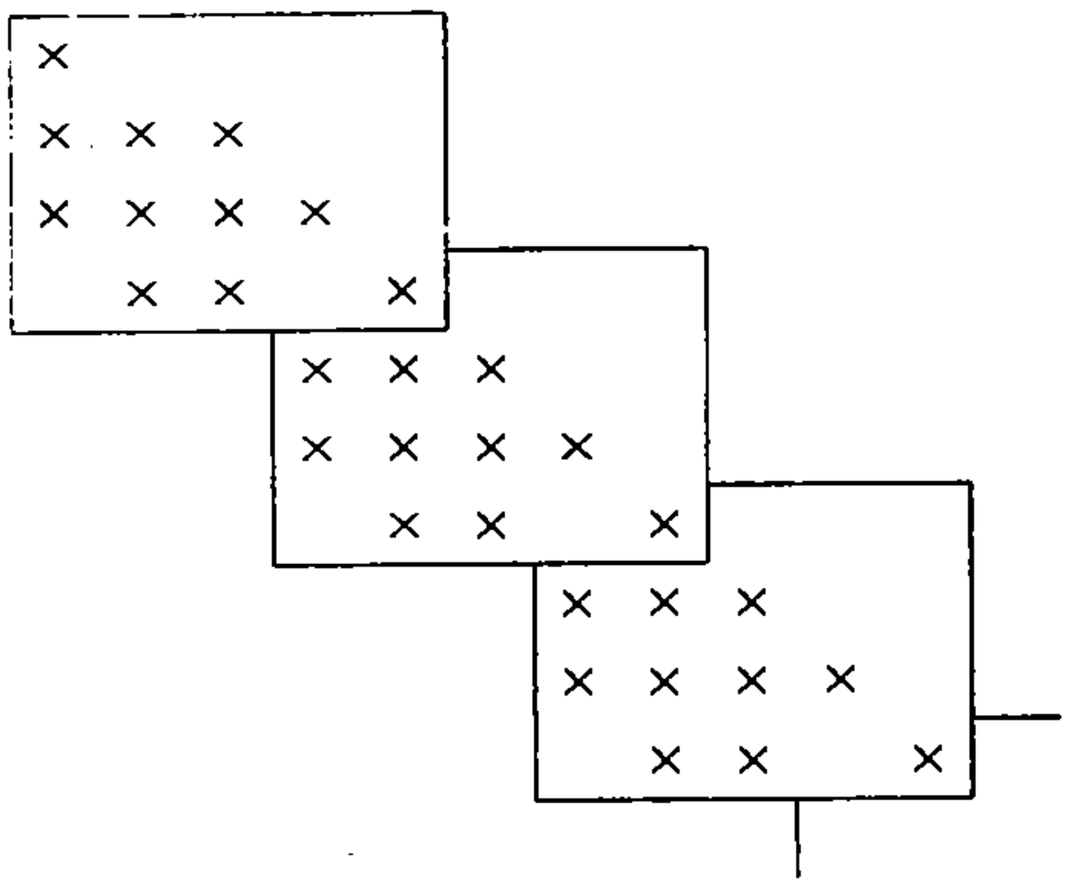
This gives altogether $n + 2$ interpolation conditions and $2(n-1)$ homogeneous conditions, for a total of $3n$ conditions on the $3n$ polynomial coefficients $(C_{ji})_{j=1}^{3};{}_{i=1}^{n}$ in the ppform for the function f (recall that $C_{ji} := D^{j-1} f(\xi_i^+)$). The resulting linear system has the following appearance.

(3)

$$\begin{array}{llll}
C_{11} & & & = 0 \quad \text{interpolation at } \tau_1 \\
C_{11} + C_{21}\frac{\Delta\tau_1}{2} & + C_{31}\frac{\Delta\tau_1^2}{6} & & = h_1 \quad \text{area matching on } [\tau_1 \,..\, \tau_2] \\
C_{11} + C_{21}\Delta\tau_1 & + C_{31}\frac{\Delta\tau_1^2}{2} & - C_{12} & = 0 \quad \text{continuity of } f \text{ across } \tau_2 \\
C_{21} & + C_{31}\Delta\tau_1 & - C_{22} & = 0 \quad \text{continuity of } Df \text{ across } \tau_2 \\
 & & C_{12} + C_{22}\frac{\Delta\tau_2}{2} + C_{32}\frac{\Delta\tau_2^2}{6} & = h_2 \quad \text{area matching on } [\tau_2 \,..\, \tau_3] \\
 & & C_{12} + C_{22}\Delta\tau_2 + C_{32}\frac{\Delta\tau_2^2}{2} - \cdots & = 0 \quad \text{continuity of } f \text{ across } \tau_3 \\
 & & C_{22} + C_{32}\Delta\tau_2 - \cdots & = 0 \quad \text{continuity of } Df \text{ across } \tau_3 \\
 & & \cdots & = \cdots \quad \textit{etc.}
\end{array}$$

□

Schematically, this linear system has the following **almost block diagonal** form (all nonzero matrix entries are indicated by a ×):



The package SOLVEBLOK given in the Appendix is designed to solve such systems efficiently, by Gauss elimination with partial pivoting. Incidentally, the factorization methods of Carasso & Laurent [1969] and Munteanu & Schumaker [1973] for such systems also use Gauss elimination with partial pivoting, but in so cleverly disguised a form that probably even their originators didn't notice this.

There is really no objection to determining the solution of such problems by solving linear systems such as (3), except for the fact that such systems contain many homogeneous equations. For instance, (3) is two-thirds homogeneous. This means that two-thirds of the equations that make up (3) could be solved once and for all, leaving a much smaller system (one-third the size of (3)) to be solved for any particular g. One accomplishes this reduction by constructing a linearly independent function sequence $\varphi_1, \varphi_2, \ldots$, with as many entries as there are interpolation conditions, and each satisfying all the homogeneous conditions. Our function f is then found in the form

$$\sum_j \alpha_j \varphi_j$$

with the coefficients (α_j) determined from the information about g. In effect, the homogeneous conditions tell us that f is to belong to some *subspace* of $\Pi_{<k,\xi}$ (namely the subspace of all functions in $\Pi_{<k,\xi}$ satisfying these homogeneous conditions), and the above construction amounts to finding some *basis* for that subspace.

We identify the subspaces of interest first, and then give a basis for each of them.

The space $\Pi_{<k,\xi,\nu}$ The typical homogeneous conditions require that the pp function $f \in \Pi_{<k,\xi}$ be constructed to have a certain number of continuous derivatives. We write these conditions in the form

$$\text{(4)} \qquad \text{jump}_{\xi_i} D^{j-1} f = 0 \quad \text{for} \quad j = 1, \ldots, \nu_i \quad \text{and} \quad i = 2, \ldots, l,$$

for some vector $\nu := (\nu_i)_2^l$ of nonnegative integers. Here, ν_i counts the number of continuity conditions required at ξ_i. In particular, $\nu_i = 0$ means that we impose no continuity condition whatever at ξ_i. Further, we are using here the abbreviation (or linear functional)

$$\text{(5)} \qquad \text{jump}_\alpha f := f(\alpha^+) - f(\alpha^-)$$

which is read "the jump of the function f at, or across, the site α".

As the conditions (4) are linear and homogeneous, the subset of all $f \in \Pi_{<k,\xi}$ satisfying (4) for a given vector ν is a linear subspace of $\Pi_{<k,\xi}$. The reader should verify this fact if it seems unfamiliar. We denote this subspace by

$$\Pi_{<k,\xi,\nu}.$$

In this notation, the set of parabolic splines needed in the smoothing of a histogram earlier becomes the set $\Pi_{<3,\xi,\nu}$ with $\nu = 2 := (2, \ldots, 2)$.

We now have a convenient notation for the set of all pp functions satisfying certain common homogeneous conditions. But, if we are to make use of this space $\Pi_{<k,\xi,\nu}$ in computations, we need a *basis* for it, that is, we need a sequence $\varphi_1, \varphi_2, \ldots$, of functions, all in $\Pi_{<k,\xi,\nu}$ and such that every element f of $\Pi_{<k,\xi,\nu}$ can be written in one and only one way as a linear combination

$$\sum_j \alpha_j \varphi_j$$

of the sequence $\varphi_1, \varphi_2, \ldots$.

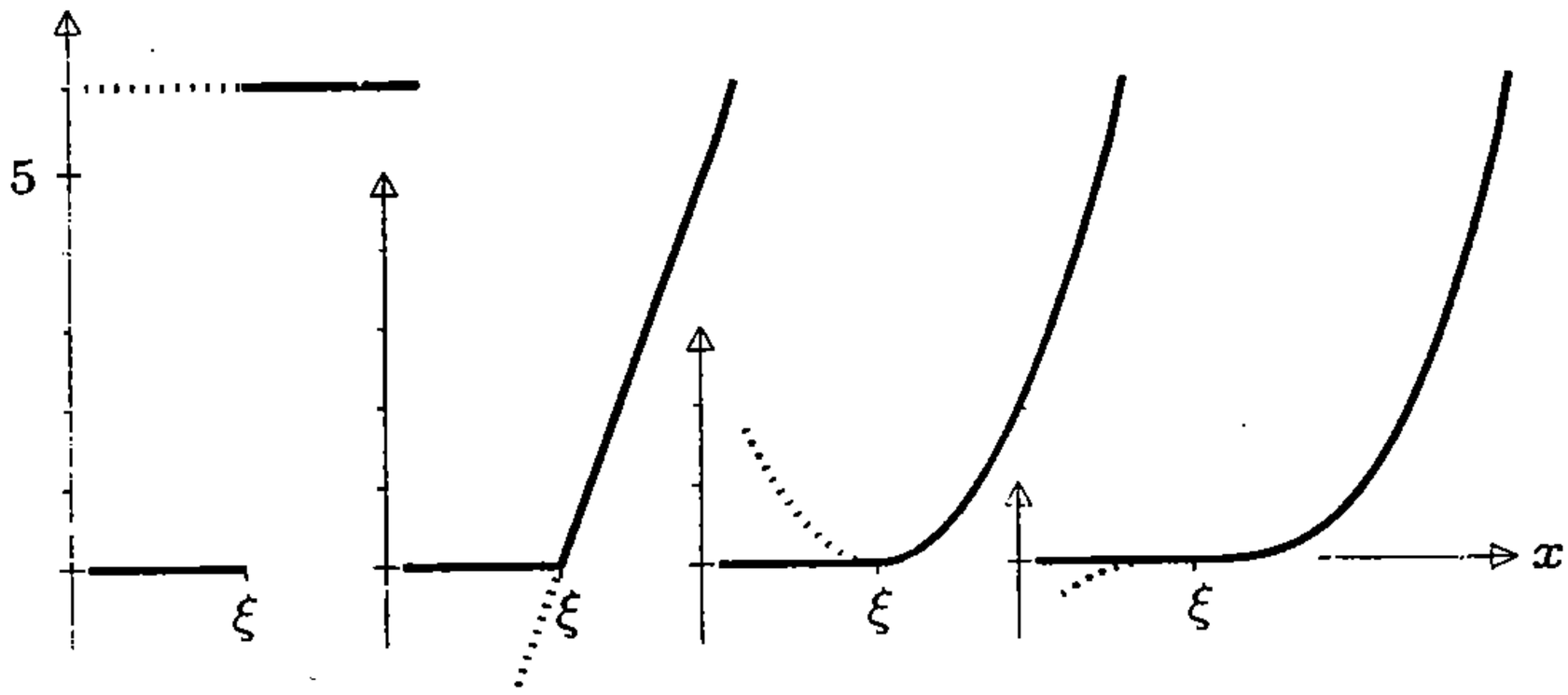
The truncated power basis for $\Pi_{<k,\xi}$ and $\Pi_{<k,\xi,\nu}$ Recall from V(3) the definition

$$(x-t)_+ := \max\{x-t, 0\},$$

in terms of which we define the **truncated power function**

$$(x)_+^r := (x_+)^r, \quad r = 0, 1, 2, \ldots.$$

The function $f(x) := (x-\xi)_+^r$ is a piecewise polynomial, of order $r+1$, with just one (active) break, at ξ, and is continuous at ξ in case $r > 0$, while, for $r = 0$, it has a jump across ξ, of size 1. Since $D(\cdot - \xi)_+^r = r(\cdot - \xi)_+^{r-1}$, we see that $(\cdot - \xi)_+^r$ has $r-1$ continuous derivatives, with a jump in the rth derivative across ξ, of size $r!$.



(6) FIGURE. The functions $6(x-\xi)_+^0$, $6(x-\xi)_+^1$, $3(x-\xi)_+^2$, $(x-\xi)_+^3$. Note the increasing smoothness across $x=\xi$, as evidenced by the fact that each is the derivative of its neighbor to the right.

What about $(x)_+^0$? Since $a^0 = 1$ for any nonzero a, we have $(x)_+^0 = 1$ for $x > 0$, but for $x \le 0$ we would need to know the value of 0^0, something undefined. That being so, we are free to define it, as follows:

$$0^0 := 0.$$

Thus, $(x)_+^0 = 0$ for $x < 0$.

With this, $(x-\xi)_+^r$ is a pp function of x even for $r = 0$ and, as a pp function, is determined by its polynomial pieces and break(s). In particular, it has, in general, *two* values at its sole break. If it is evaluated in PPVALU, then $(0)_+^0 = (0^+)_+^0 = 1$, while PPVLLC would give $(0)_+^0 = (0^-)_+^0 = 0$.

Now define the linear functionals λ_{ij} and corresponding functions φ_{ij} by

$$\lambda_{ij} f := \begin{cases} D^j f(\xi_1), & i = 1; \\ \text{jump}_{\xi_i} D^j f, & i = 2, \dots, l; \end{cases}$$

(7)

$$\varphi_{ij}(x) := \begin{cases} (x-\xi_1)^j/j!, & i = 1; \\ (x-\xi_i)_+^j/j!, & i = 2, \dots, l, \end{cases}$$

for $j = 0, \dots, k-1$. It is clear that each function φ_{ij} is in $\Pi_{<k,\xi}$. Also, from what has been said already,

$$\lambda_{ij}\varphi_{rs} = \delta_{ir}\delta_{js} = \begin{cases} 1, & \text{if } i = r \text{ and } j = s, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

This shows that the double sequence (φ_{ij}) is linearly independent. Since (φ_{ij}) consists of kl functions and $\Pi_{<k,\xi}$ has dimension kl, as proved in

Problem VII.4, we conclude that (φ_{ij}) is a basis for $\Pi_{<k,\xi}$. This means that every $f \in \Pi_{<k,\xi}$ has a unique representation of the form

$$f = \sum_{ij} (\lambda_{ij} f)\varphi_{ij}$$

and that, by (7) and (8), this representation can be written quite explicitly as

$$(9) \quad f(x) = \sum_{j<k} f^{(j)}(\xi_1)(x-\xi_1)^j/j! + \sum_{i=2}^{l} \sum_{j<k} (\mathrm{jump}_{\xi_i} D^j f)(x-\xi_i)_+^j/j!.$$

Now note that the jumps of the various derivatives of the function f across the various breaks appear *explicitly* as coefficients in the representation (9) for f. This makes the enforcement of the constraints

$$(10) \qquad \mathrm{jump}_{\xi_i} D^{j-1} f = 0 \quad \text{for } j = 1, \dots, \nu_i \text{ and } i = 2, \dots, l$$

very easy: We simply restrict attention to those functions f of the form (9) for which these coefficients are zero. This means that *the double sequence* (or set)

$$(11) \qquad \varphi_{ij}, \quad j = \nu_i, \dots, k-1 \text{ and } i = 2, \dots, l$$

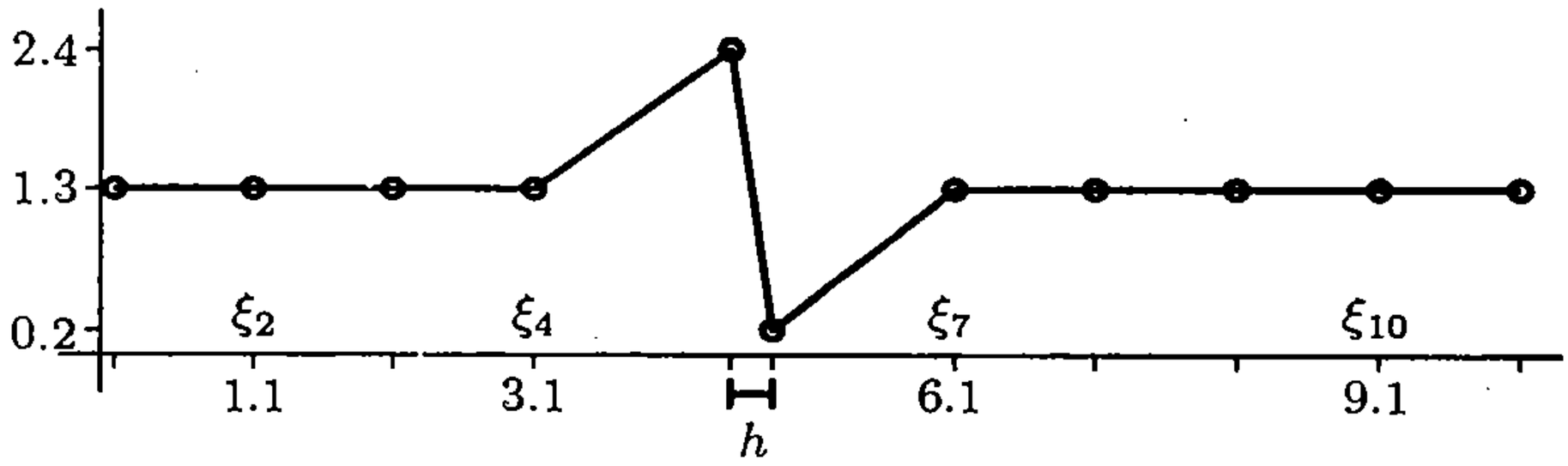
is a basis for $\Pi_{<k,\xi,\nu}$. Here, we have set $\nu_1 := 0$. In other words, every function f in $\Pi_{<k,\xi,\nu}$ can be written in exactly one way in the form

$$f = \sum_{i=1}^{l} \sum_{j=\nu_i}^{k-1} \alpha_{ij}\varphi_{ij}.$$

At first glance, the basis (φ_{ij}) seems tailormade for our problems involving pp functions of some smoothness. But, in comparing the representation (9) with the ppform VII(6), two complaints come to mind:

(i) The value of f at a site x can involve considerably more than just k of the coefficients if l is "large".

(ii) For very nonuniform ξ, some of the basis functions φ_{ij} become nearly linearly dependent on the others. This means that small changes in the coefficients might produce much smaller or much larger changes in the function represented. This possible bad *condition* (see pp. 12ff) of the truncated power basis will also result in badly conditioned linear systems (analogous to the system (3)) for the coefficients of f with respect to the truncated power basis.



(12) FIGURE. A broken line that causes difficulties for the truncated power basis.

(13) Example: The truncated power basis can be bad Construct the function f as a broken line, that is, $f \in \Pi_{<2,\xi} \cap C^{(0)}$, such that $f(\xi_i) = 1.3$, all i, except that $f(\xi_5) = 2.4$ and $f(\xi_6) = .2$. We choose the breaks $(\xi_i)_1^{11}$ as indicated in Figure (12). We have $f \in \Pi_{<2,\xi,\nu}$ with $\nu = (1, \ldots, 1)$. Therefore

$$f(x) = \alpha + \beta(x - \xi_1) + \sum_2^l \alpha_i (x - \xi_i)_+$$

and $\alpha = 1.3$, $\beta = f'(\xi_1) = 0$, and $\alpha_i = 0$ except that

$$(\alpha_4, \ldots, \alpha_7) = (1.1/\Delta\xi_4, -2.2/h - 1.1/\Delta\xi_4, 1.1/\Delta\xi_6 + 2.2/h, -1.1/\Delta\xi_6).$$

As $h := \Delta\xi_5$ becomes small, we get $(x - \xi_5)_+ \sim (x - \xi_6)_+$, and, correspondingly, $-\alpha_5 \sim \alpha_6 \gg 1$. This leads to loss of significance in the evaluation of the function f. If, for example, we choose $\xi_5 = 4.5$ and $\xi_6 = 4.8$, so that $h = .3$, and we use two significant decimal digit arithmetic (with rounding, a tie going to the nearest even number), then we obtain

$$\alpha = 1.3,\ \beta = 0,\ \alpha_4 = .79,\ \alpha_5 = -8.1,\ \alpha_6 = 8.2,\ \alpha_7 = -.85,$$

while all other α_i's are zero. If we then evaluate f at $x = 9.5$, we get the value 2.1 instead of the correct 1.3. □

For this example, the remedy is obvious: Use the hat functions III(6), given by the rule

$$H_i(x) = \begin{cases} (x - \xi_{i-1}/\Delta\xi_{i-1}, & \xi_{i-1} < x \le \xi_i, \\ (\xi_{i+1} - x)/\Delta\xi_i, & \xi_i \le x < \xi_{i+1}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, we use some $\xi_0 \le \xi_1$ and some $\xi_{12} \ge \xi_{11}$. Now

$$f(x) = 1.3H_1(x) + \cdots + 1.3H_4(x) + 2.4H_5(x) + .2H_6(x) + 1.3H_7(x) + \cdots$$

and, again with 2 decimal digit arithmetic, we find

$$f(9.5) = 1.3(.4/.9) + 1.3(.5/.9) = 1.3 \cdot .44 + 1.33 \cdot .56 = .57 + .73 = 1.3 \ .$$

In fact, one could even let $h = \Delta\xi_5$ approach 0 and still have the function f well represented by the basis functions (H_i).

In the general case, both objections to the truncated power basis mentioned earlier can be overcome (at least for moderate k) by a generalization of the "hat" functions, the so called B-splines. These are obtained by forming *analytically* certain linear combinations of the truncated power functions to obtain a new basis for $\Pi_{<k,\xi,\nu}$ whose elements each vanish outside a "small" interval, as described in the next chapter.

Problems

1. Show that $\Pi_{<3,\tau} \cap C^{(1)} = \{ DF : F \in \Pi_{<4,\tau} \cap C^{(2)} \}$, and that $\Pi_{<4,\tau} \cap C^{(2)} = \$_4$ (of p. 51, with $(\tau_i)_1^n$ replaced by $(\tau_i)_1^{n+1}$). Conclude that the approximation f to the distribution g in Example (1) can be constructed as the derivative $f = DF$ of $F \in \$_4$ for which $F(\tau_i) = \sum_{j<i} h_j \Delta\tau_j$, $i = 1, \ldots, n+1$ (in particular, $F(\tau_1) = 0$) and $F'(\tau_1) = F'(\tau_{n+1}) = 0$. Hence identify Figure (2) with the results of Problem IV.9.

2. Prove that $\Pi_{<k,\xi,\nu}$ *is a linear subspace of* $\Pi_{<k,\xi}$ (see Problem VII.4).

3. Prove: *If* $\varphi_1, \ldots, \varphi_n$ *is linearly independent and* $A := (a_{ij})$ *is an* $m \times n$ *matrix whose* m *rows form a linearly independent sequence, then the sequence* $\psi_1, \ldots, \psi_m$, *with* $\psi_i := \sum a_{ij}\varphi_j$, *all* i, *is also linearly independent.*

4. (a) Use Problem IV.2 to construct a basis (φ_i^H) for $\Pi_{<4,\xi,2}$, with $\mathbf{2} := (2, \ldots, 2)$, in such a way that the coordinates for $f \in \Pi_{<4,\xi,2}$ with respect to this basis are $\big(f(\xi_1), f'(\xi_1), f(\xi_2), f'(\xi_2), \ldots, f(\xi_{l+1}), f'(\xi_{l+1})\big)$.
(b) Express the homogeneous conditions

$$\text{jump}_{\xi_i} D^2 f = 0, \qquad i = 2, \ldots, l,$$

in terms of the coordinates for f with respect to the basis (φ_i^H) found in (a). Where have you seen these equations before?
(c) Construct f_1, f_2 as linear combinations of $(\varphi_1^H, \ldots, \varphi_6^H)$ both with continuous second derivatives and so that (f_1, f_2) is linearly independent. Can you choose f_1, f_2 to be nonnegative on $[\xi_1 \,..\, \xi_3]$?
(d) Use (c) to get a basis for $\Pi_{<4,\xi,3}$ (you might need Problem 3 to show that you have gotten a *basis* for $\Pi_{<4,\xi,3}$). What is the relationship of $\Pi_{<4,\xi,3}$ to $\$_4$ on p. 51?

IX

The Representation of PP Functions by B-Splines

In this chapter, we define kth order B-splines as appropriately scaled kth divided differences of the truncated power function and prove that every space $\Pi_{<k,\xi,\nu}$ has a basis consisting of such **basis splines** or B-splines. This gives rise to the B-form for a pp function.

Here is the definition of a B-spline as originally given by Curry & Schoenberg [1947] (though with a different normalization).

(1) **Definition.** *Let* $\mathbf{t} := (t_j)$ *be a nondecreasing sequence (which may be finite, infinite or biinfinite).* **The jth (normalized) B-spline of order k for the knot sequence $\mathbf{t}$** *is denoted by* $B_{j,k,\mathbf{t}}$ *and is defined by the rule*

$$(2) \qquad B_{j,k,\mathbf{t}}(x) := (t_{j+k} - t_j)[t_j, \ldots, t_{j+k}](\cdot - x)_+^{k-1}, \quad \text{all } x \in \mathbb{R}.$$

Note that, by I(13),

$$(3) \qquad B_{j,k,\mathbf{t}}(x) = [t_{j+1}, \ldots, t_{j+k}](\cdot - x)_+^{k-1} - [t_j, \ldots, t_{j+k-1}](\cdot - x)_+^{k-1}.$$

The "placeholder" notation is used to indicate that the kth divided difference of the function $(t-x)_+^{k-1}$ of the *two* variables t and x is to be taken by fixing x and considering $(t-x)_+^{k-1}$ as a function of t alone. The resulting number depends, of course, on the particular value of x we choose, that is, the resulting number varies as we vary x, and so we obtain eventually the function $B_{j,k,\mathbf{t}}$ of x.

We will usually write

$$B_j \text{ or } B_{jk} = B_{j,k} \quad \text{instead of} \quad B_{j,k,\mathbf{t}}$$

as long as the k and the $\mathbf{t}$ can be inferred from the context.

In this chapter and in the next two, we will record various properties of the B-spline, in hopes of making it thereby as familiar and real an object for the reader as, say, the sine function.

All these properties could (see, e.g., the first edition of this book) be derived directly from the above definition with the aid of the various di-

vided difference properties recorded in Chapter I. However, it turns out (see de Boor & Höllig [1987] and de Boor [1993]) to be just as efficient (and less hard on readers not familiar with divided differences) to establish just one property, namely the B-spline recurrence relation, and then derive everything else from this without recourse to divided differences. It is even possible (see (41)) to derive the equality (2) from the recurrence relation, thus making it possible to *start* B-spline theory with the recurrence relations, though this may leave open the question of how one would come up with the recurrence relation in the first place.

The Curry-Schoenberg definition was based on the observation that, at least for $t_j < \cdots < t_{j+k}$, the function defined in (2) is obviously pp of order k and, in any case, has small support in the sense that

$$\mathrm{B}_{j,k,\mathbf{t}}(x) = 0 \quad \text{for} \quad x \notin [t_j \,..\, t_{j+k}]. \tag{4}$$

For, if $x \notin [t_j \,..\, t_{j+k}]$, then $g := (\cdot - x)_+^{k-1}$ is a polynomial of order k on $[t_j \,..\, t_{j+k}]$ and therefore, by I(v), we have $[t_j, \ldots, t_{j+k}]g = 0$. Further, the Curry-Schoenberg B-spline

$$M_{j,k,\mathbf{t}} := \frac{k}{t_{j+k} - t_j} \mathrm{B}_{j,k,\mathbf{t}} \tag{5}$$

is differently normalized. It arises naturally when one applies the divided difference to both sides of the Taylor identity

$$f = \sum_{r<k} D^r f(a)\,(\cdot - a)^r/r! \;+\; \int_a^b (\cdot - s)_+^{k-1}\, D^k f \; \mathrm{d}s/k!$$

to obtain (under the assumption that $t_j, \ldots, t_{j+k} \in [a \,..\, b]$)

$$[t_j, \ldots, t_{j+k}]f \;=\; \int_{\mathbb{R}} M_{j,k,\mathbf{t}} D^k f/k!, \tag{6}$$

showing that $M_{j,k,\mathbf{t}}$ *is the Peano kernel for the divided difference*. In particular, with $f(x) = x^k$,

$$\int_{\mathbb{R}} M_{j,k,\mathbf{t}} = 1. \tag{7}$$

The notation $N_{j,k,\mathbf{t}}$ for $\mathrm{B}_{j,k,\mathbf{t}}$ is quite common, in distinction to $M_{j,k,\mathbf{t}}$. The B_{jk} are so normalized that

$$\sum_{j=r}^{s} \mathrm{B}_{jk} = 1 \quad \text{on } [t^+_{r+k-1} \,..\, t^-_{s+1}], \tag{8}$$

that is, on $(t_{r+k-1} \,..\, t_{s+1})$, hence also in the (one-sided) limit at the endpoints of this interval; see (36).

Two special knot sequences

For two frequently used knot sequences, the various details to follow simplify significantly.

One is the **uniform knot sequence**, $\mathbf{t} = \mathbb{Z} = (\ldots, -1, 0, 1, 2, \ldots)$. The corresponding B-splines are called **cardinal**. For a given k, these cardinal B-splines are integer translates of each other since (see Problem 2)

$$\mathrm{B}_{j,k,\mathbb{Z}}(x) = \sum_{r=0}^{k} (-)^{k-r} \binom{k}{r} (r - x - j)_+^{k-1}/(k-1)!, \tag{9}$$

thus greatly simplifying their study.

The other is the knot sequence

$$\mathbb{B} := (\ldots, 0, 0, 0, 1, 1, 1, \ldots)$$

which has just the two knots, 0 and 1, but both with infinite multiplicity. For this knot sequence, there are just k nontrivial B-splines of order k, namely the restriction to $[0\,..\,1]$ of the polynomials

$$b_{j,k-1}(x) := \binom{k-1}{j} x^{k-1-j}(1-x)^j, \quad j = 0, \ldots, k-1,$$

familiar from the **Bernstein polynomial** (see, e.g., Rivlin [1969:p. 12])

$$\sum_{j=0}^{k-1} f\left(\frac{j}{k-1}\right) b_{j,k-1} \tag{10}$$

of order k for the function f. The sequence $(b_{j,k-1} : j = 0, \ldots, k-1)$ is a basis for $\Pi_{<k}$, giving rise to the **Bernstein-Bézier form**, or **BBform**, much used in CAGD.

It is very instructive to specialize the various discussions to follow to these particular two knot sequences.

A recurrence relation for B-splines

Directly from the definition,

$$\mathrm{B}_{j1} = (\cdot - t_{j+1})_+^0 - (\cdot - t_j)_+^0$$

is the characteristic function of the jth knot interval, that is,

$$\mathrm{B}_{j1}(x) = \begin{cases} 1, & \text{if } t_j \le x < t_{j+1}; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Note that, in conformity with the convention adopted in Chapter VII, these functions are continuous from the right. Other choices could have been made. The only constraint is that these B_j should form a **partition of unity**, that is,

$$\sum_j \mathrm{B}_{j1} = 1. \tag{12}$$

In particular,

$$t_j = t_{j+1} \quad \Longrightarrow \quad \mathrm{B}_{j1} = 0.$$

Starting with these first-order B-splines, one may construct higher-order B-splines with the aid of the following

(13) B-spline Property (i): Recurrence relation. *For* $k > 1$,

$$\mathrm{B}_{jk} = \omega_{jk}\mathrm{B}_{j,k-1} + (1 - \omega_{j+1,k})\mathrm{B}_{j+1,k-1}, \tag{14}$$

with

$$\omega_{jk}(x) := \frac{x - t_j}{t_{j+k-1} - t_j}. \tag{15}$$

PROOF. Apply Leibniz' formula I(iv) for the kth divided difference of a product to the particular product

$$(t - x)_+^{k-1} = (t - x)(t - x)_+^{k-2}.$$

This gives

$$\begin{aligned}[t_j, \ldots, t_{j+k}](\cdot - x)_+^{k-1} \; = \\ (t_j - x)[t_j, \ldots, t_{j+k}](\cdot - x)_+^{k-2} + 1[t_{j+1}, \ldots, t_{j+k}](\cdot - x)_+^{k-2}\end{aligned} \tag{16}$$

since $[t_j](\cdot - x) = (t_j - x)$, $[t_j, t_{j+1}](\cdot - x) = 1$, while $[t_j, \ldots, t_r](\cdot - x) = 0$ for $r > j + 1$. Now,

$$(t_j - x)[t_j, \ldots, t_{j+k}] = \frac{t_j - x}{t_{j+k} - t_j}\left([t_{j+1}, \ldots, t_{j+k}] - [t_j, \ldots, t_{j+k-1}]\right).$$

Therefore, (16) can also be stated as

(17)

$$\begin{aligned}&[t_j, \ldots, t_{j+k}](\cdot - x)_+^{k-1} \; = \\ &\frac{x - t_j}{t_{j+k} - t_j}[t_j, \ldots, t_{j+k-1}](\cdot - x)_+^{k-2} + \frac{t_{j+k} - x}{t_{j+k} - t_j}[t_{j+1}, \ldots, t_{j+k}](\cdot - x)_+^{k-2},\end{aligned}$$

and this, after multiplication by $(t_{j+k} - t_j)$, is (14). □

Thus, the second-order B-spline is given by

$$\mathrm{B}_{j2} = \omega_{j2}B_{j1} + (1 - \omega_{j+1,2})B_{j+1,1},$$

hence consists, in general, of two nontrivial linear pieces which join continuously to form a piecewise linear function that vanishes outside the interval $[t_j \,..\, t_{j+1})$. For this reason, B_{j2} is also called a *linear* B-spline. If, however, for example $t_j = t_{j+1}$, then B_{j2} consists of only one nontrivial linear piece, has a jump at t_j, but is still continuous at t_{j+1}.

The third-order B-spline is given by

$$\begin{aligned}\mathrm{B}_{j3} \; &= \omega_{j3}\mathrm{B}_{j2} + (1 - \omega_{j+1,3})\mathrm{B}_{j+1,2} \\ &= \omega_{j3}\omega_{j2}\mathrm{B}_{j1} + (\omega_{j3}(1 - \omega_{j+1,2}) + (1 - \omega_{j+1,3})\omega_{j+1,2})\mathrm{B}_{j+1,1} \\ &\quad + (1 - \omega_{j+1,3})(1 - \omega_{j+2,2})\mathrm{B}_{j+2,1},\end{aligned} \tag{18}$$

hence, in general, consists of three (nontrivial) parabolic pieces that join to form a $C^{(1)}$ function that vanishes outside $[t_j \,..\, t_{j+2})$.

After $k-1$ applications of the recurrence, we obtain B_{jk} in the form

$$\mathrm{B}_{jk} = \sum_{r=j}^{j+k-1} b_{rk}\mathrm{B}_{r1}, \tag{19}$$

with each of the k b_{rk} a polynomial of order k (as the sum of products of $k-1$ linear polynomials; in fact, each b_{rk} in (19) is of degree $k-1$). This establishes most of the following:

(20) B-spline Property (ii): Support and positivity. *The B-spline $\mathrm{B}_{j,k,\mathbf{t}}$ is pp of order k with breaks $t_j, \ldots, t_{j+k}$, hence made up of at most k nontrivial polynomial pieces, vanishes outside the interval $[t_j \,..\, t_{j+k})$, and is positive on the interior of that interval, that is,*

$$\mathrm{B}_{j,k,\mathbf{t}}(x) > 0, \quad t_j < x < t_{j+k} \tag{21}$$

while

$$t_j = t_{j+k} \quad \Longrightarrow \quad \mathrm{B}_{jk} = 0. \tag{22}$$

PROOF. Only (21) still needs proof. Certainly B_{j1} is positive on $(t_j \,..\, t_{j+1})$. Assuming (21) to hold for $k < r$, the positivity of both ω_{jr} and $(1-\omega_{j+1,r})$ on $(t_j \,..\, t_{j+r})$ implies with (14) that (21) also holds for $k = r$. □

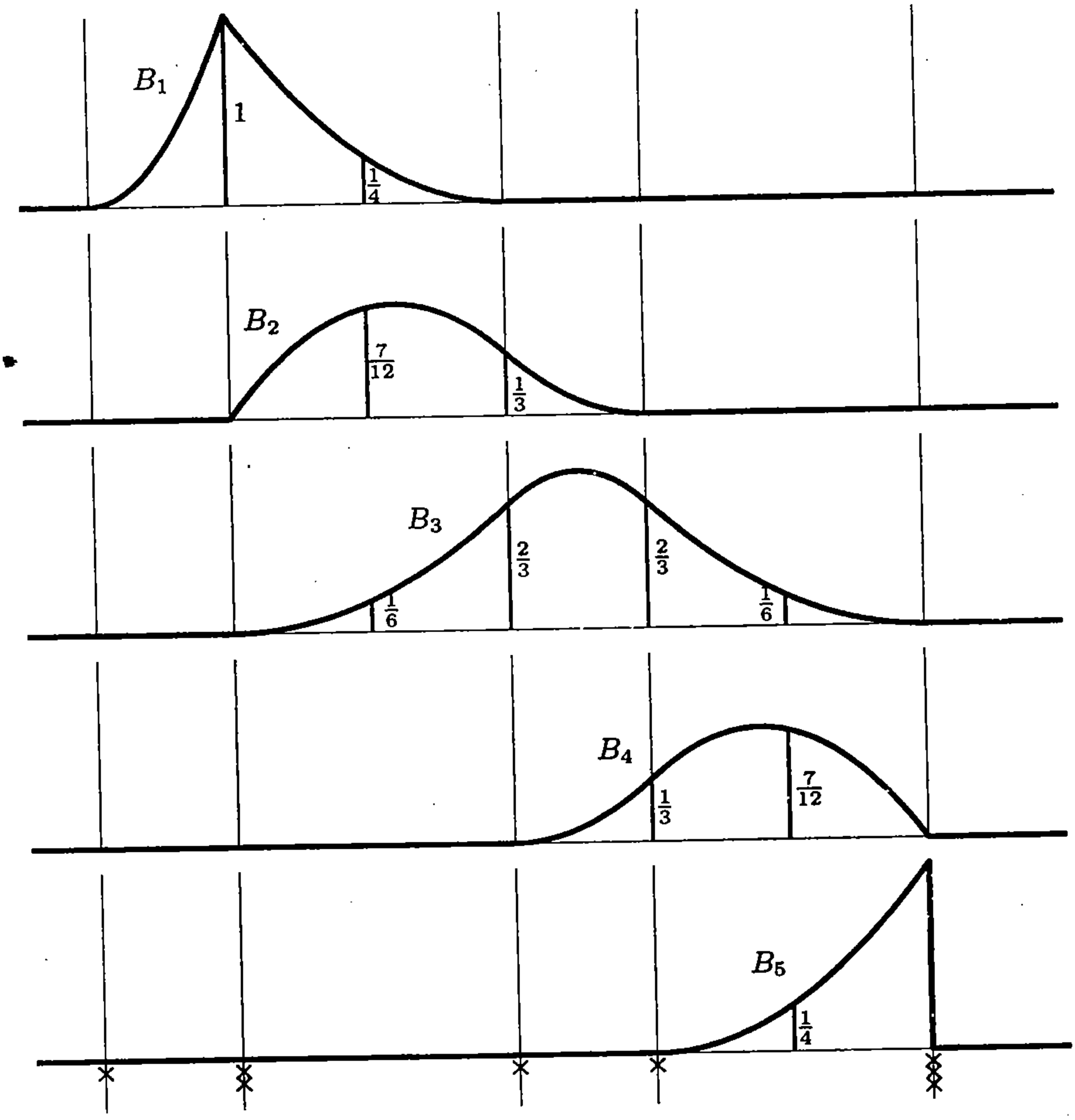
The B-spline $\mathrm{B}_{j,k,\mathbf{t}}$ depends only on the $k+1$ knots $t_j, \ldots, t_{j+k}$. For this reason, the alternative notation

$$\mathrm{B}_{j,k,\mathbf{t}} =: B(\cdot|t_j, \ldots, t_{j+k}) \tag{23}$$

is also quite common.

The actual smoothness of B_{jk} depends on the multiplicity with which the break ξ_i appears in the knot sequence $(t_j, \ldots, t_{j+k})$; see (49).

(24) Example: A sequence of parabolic B-splines Figure (25) shows the five parabolic B-splines for the knot sequence $(0, 1, 1, 3, 4, 6, 6, 6)$. Property (ii) is clearly indicated. Also (8) is illustrated at a few sites, where function values are given numerically. Note that, in conformity with (8), $\sum_1^5 \mathrm{B}_j(x) = 1$ only on $[t_k \,..\, t_{n+1}] = [1 \,..\, 6]$. In particular, on $[0 \,..\, 1)$, the sum is *not* equal to 1.



(25) FIGURE. The parabolic B-splines for the knot sequence $(t_1, \ldots, t_{5+3}) = (0, 1, 1, 3, 4, 6, 6, 6)$. Note the connection between knot multiplicity and smoothness.

Each B_j is piecewise parabolic, and the breaks are clearly visible as places of discontinuity in the B-spline or one of its derivatives. Only one of the B-splines appears to be discontinuous, namely B_5, with a jump discontinuity at 6, corresponding to the fact that the number 6 appears *three* times in the knot sequence $t_j, \ldots, t_{j+k}$ involved in the definition of B_5. Only three B-splines appear to have a discontinuous first derivative, namely B_1 and B_2, at 1, corresponding to the fact that the number 1 appears *twice* in the sequence $t_j, \ldots, t_{j+k}$ of knots involved in the definition of B_j for $j = 1, 2$, and B_4 at 6,

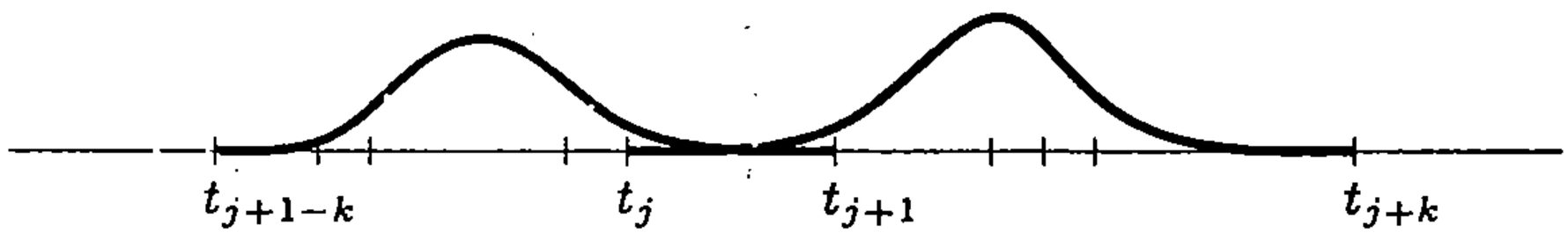
with 6 appearing twice in the knot sequence $(3, 4, 6, 6)$ corresponding to B_4. Note that B_3 appears to have a continuous first derivative at 1; its defining knot sequence, $(t_3, t_4, t_5, t_6) = (1, 3, 4, 6)$, contains the number 1 only once. At all other breaks, the functions B_j have, at worst, a discontinuity in the second derivative. E.g., on $[1 \,..\, 3]$, B_3 is a parabola, concave upward, hence has a positive (constant) second derivative there, while, on $[3 \,..\, 4]$, it is concave downward, so $\text{jump}_3 D^2 B_3 < 0$. This connection between knot multiplicity and smoothness is basic to the use of B-splines and will be explored further below. □

The spline space $\$_{k,t}$

(26) Definition. *A* **spline function of order** k **with knot sequence** t *is any linear combination of B-splines of order* k *for the knot sequence* t. *The collection of all such functions is denoted by* $\$_{k,t}$. *In symbols,*

$$\$_{k,t} := \left\{ \sum_i \alpha_i B_{i,k,t} : \alpha_i \text{ real, all } i \right\}.$$

We will have little occasion in this book to consider infinite or biinfinite knot sequences. But if t is infinite or biinfinite, then the infinite sum in this definition is meant to be taken *pointwise*, that is, the value of the function $\sum_i \alpha_i B_i$ at the site x is simply the value of the sum $\sum_i \alpha_i B_i(x)$ which makes good sense since the latter sum has at most k nonzero terms, by B-spline Property (ii); see (27).



(27) FIGURE. Support of the leftmost and the rightmost B-spline of order k that is nonzero on $(t_j \,..\, t_{j+1})$.

This definition of "spline function" may leave the reader in doubt as to what exactly spline functions "are". The reader may further protest that Chapters III-VI already contain definitions of such things as cubic splines or parabolic splines and that those definitions do not jibe completely with the above definition. This latter objection is justified. But, before discussing it, we take up the more fundamental question of what exactly splines, according to the Definition (26), "are", i.e., exactly what kind of functions make up $\$_{k,t}$.

It is obvious from B-spline Property (ii) that

$$\$_{k,t} \subseteq \Pi_{<k,t},$$

but it takes some work to prove the Curry-Schoenberg Theorem (44) below, to the effect that

$$\$_{k,\mathbf{t}} = \Pi_{<k,\xi,\nu}$$

for some break sequence ξ and some sequence ν, with this equality only holding on the "basic interval" associated with the knot sequence t.

This last *caveat* is related to the difficulty that already occurs when we want to make use of the following immediate consequence of the recurrence relation (14):

$$\sum_j \alpha_j B_{jk} = \sum_j (\omega_{jk}\alpha_j + (1-\omega_{jk})\alpha_{j-1})\mathrm{B}_{j,k-1}. \tag{28}$$

If t has a first knot, that is, $\mathbf{t} = (t_1, t_2, \ldots)$, then the sum on the left in (28) starts with $j = 1$, but the sum on the right can only start with $j = 2$ since it contains the term α_{j-1}. To circumvent this difficulty *we* agree in this case (and in the corresponding case when t has a last entry) *to extend* t *to a biinfinite knot sequence in any way whatsoever*, denoting the extension again by t. However, this increases the number of available B-splines, hence also increases the spline space. Since we are still only interested in our original spline space, *we further agree to choose the additional B-spline coefficients to be zero*. But, to be quite certain that none of the additional knots or B-splines matter, we restrict attention to the largest interval not intersected by the interior of the support of any of the additional B-splines. We call this the **basic interval for $\$_{k,\mathbf{t}}$** and denote it

$$I_{k,\mathbf{t}} = (\mathbf{t}_- \,..\, \mathbf{t}_+), \tag{29}$$

with

$$\mathbf{t}_- := \begin{cases} t_k, & \text{if } \mathbf{t} = (t_1, \ldots); \\ \inf_j t_j, & \text{otherwise,} \end{cases} \qquad \mathbf{t}_+ := \begin{cases} t_{n+1}, & \text{if } \mathbf{t} = (\ldots, t_{n+k}); \\ \sup_j t_j, & \text{otherwise.} \end{cases}$$

In practice, the knot sequence is finite, having both a first and a last knot. In this case, one chooses $I_{k,\mathbf{t}}$ to be closed, and this is fine for the left endpoint, since we have long agreed to make all pp functions right-continuous. However, for this to work properly at the right endpoint, we modify the above B-spline definition to make each B-spline left-continuous at the right endpoint of the basic interval. (The first edition of this book failed to do this, unfortunately.)

The polynomials in $\$_{k,\mathbf{t}}$ We now use (28) to show that $\$_{k,\mathbf{t}}$ contains all polynomials of order k, and even give a formula for the B-spline coefficients for $p \in \Pi_{<k}$.

(30) B-spline Property (iii): Marsden's Identity. *For any* $\tau \in \mathbb{R}$,

$$(\cdot - \tau)^{k-1} = \sum_j \psi_{jk}(\tau) \mathrm{B}_{jk} \tag{31}$$

with

$$\psi_{jk}(\tau) := (t_{j+1} - \tau) \cdots (t_{j+k-1} - \tau). \tag{32}$$

PROOF. Consider (28) for the special sequence

$$\alpha_j = \psi_{jk}(\tau), \quad \text{all } j.$$

Then, for $\mathrm{B}_{j,k-1} \neq 0$, i.e., for $t_j < t_{j+k-1}$,

$$\begin{aligned} \omega_{jk}\alpha_j + (1 - \omega_{jk})\alpha_{j-1} &= \psi_{j,k-1}(\tau)\big(\omega_{jk}\,(t_{j+k-1} - \tau) + (1 - \omega_{jk})(t_j - \tau)\big) \\ &= \psi_{j,k-1}(\tau)(\cdot - \tau) \end{aligned}$$

since, for any f,

$$\omega_{jk} f(t_{j+k-1}) + (1 - \omega_{jk}) f(t_j)$$

is the unique straight line that agrees with f at t_j and t_{j+k-1}, hence must equal f if, as is the case for us here, f is itself a straight line. Therefore, by induction,

$$\sum_j \psi_{jk}(\tau)\mathrm{B}_{jk} = (\cdot - \tau)^{k-1} \sum_j \psi_{j1}(\tau)\mathrm{B}_{j1},$$

and the last sum equals 1, by (12), since $\psi_{j1} = 1$. □

To be sure, from the definition (32), ψ_{j1} is the product of *no* factors, hence equals 1 by definition, as that is the definition appropriate in inductive arguments such as the one just given.

Since τ in (31) is arbitrary, it follows (see Problem 11) that $\$_{k,\mathbf{t}}$ contains all polynomials of order k. More than that, we can easily obtain an explicit expression in terms of the B_{jk} for any $p \in \Pi_{<k}$, as follows. Divide (31) by $(k-1)!$, then differentiate it $\nu - 1$ times with respect to τ to obtain

$$\frac{(\cdot - \tau)^{k-\nu}}{(k-\nu)!} = \sum_j \frac{(-D)^{\nu-1}\psi_{jk}(\tau)}{(k-1)!}\,\mathrm{B}_{jk}, \quad \nu > 0. \tag{33}$$

Now use this identity in the Taylor formula

$$p = \sum_{\nu=1}^{k} \frac{(\cdot - \tau)^{k-\nu}}{(k-\nu)!}\, D^{k-\nu}p(\tau),$$

valid for any $p \in \Pi_{<k}$, to conclude that

$$p = \sum_j \lambda_{jk} p\, \mathrm{B}_{jk}, \quad p \in \Pi_{<k}, \tag{34}$$

with the linear functional λ_{jk} given by the rule

$$\lambda_{jk} f := \sum_{\nu=1}^{k} \frac{(-D)^{\nu-1}\psi_{jk}(\tau)}{(k-1)!}\, D^{k-\nu} f(\tau). \tag{35}$$

To be sure, (34) holds only on the basic interval $I_{k,\mathbf{t}}$.

The following two special cases of (34) deserve special attention.

(36) B-spline Property (iv): (positive and local) partition of unity. *The sequence* (B_{jk}) *provides a positive and local partition of unity, that is, each* B_{jk} *is positive on* $(t_j \,..\, t_{j+k})$, *is zero off* $[t_j \,..\, t_{j+k}]$, *and*

$$\sum_j B_{jk} = 1 \qquad \text{on } I_{k,\mathbf{t}}. \tag{37}$$

PROOF. Only the last assertion needs proof. It follows at once from (34) with the choice $p = 1$. □

By choosing $p \in \Pi_{<2}$ (and choosing $k > 1$), we obtain from (34)

(38) B-spline Property (v): Knot averages. *For* $k > 1$ *and any* $\ell \in \Pi_{<2}$,

$$\ell = \sum_j \ell(t^*_{jk})\, B_{jk}$$

with t^*_{jk} *the* **Greville sites**:

$$t^*_{jk} := \frac{t_{j+1} + \cdots + t_{j+k-1}}{k-1}, \quad \text{all } j. \tag{39}$$

PROOF. Indeed, since $D^{k-2}\psi_{jk}$ is a linear polynomial that vanishes at t^*_{jk}, the assertion follows from (34). □

Because of the importance of these knot averages, the SPLINE TOOLBOX (de Boor [1990]$_2$) contains the command **aveknt**(t,k) which supplies them.

The pp functions in $\$_{k,\mathbf{t}}$ Marsden's identity (31) even provides a B-spline expansion for certain *truncated* powers, because of the following simple observation.

Since $t_j < t_i < t_{j+k}$ implies $D^{\nu-1}\psi_{jk}(t_i) = 0$ for $\nu \le \#t_i$, with

$$\#t_i := \#\{r : t_r = t_i\}$$

the **multiplicity** of t_i in the knot sequence $\mathbf{t}$, the choice $\tau = t_i$ in (33) leaves as possibly nonzero terms in that sum only terms with support either entirely to the left of t_i or else entirely to the right of t_i. This implies that

$$\frac{(\cdot - \tau)_+^{k-\nu}}{(k-\nu)!} = \sum_{j \ge i} \frac{(-D)^{\nu-1}\psi_{jk}(\tau)}{(k-1)!} B_{jk}, \quad 0 < \nu \le \#t_i,\ \tau = t_i. \tag{40}$$

(41) Remark Equation (40) provides all the information needed to *deduce* the equation (2) which is traditionally used to *define* the B-spline,

thus closing the loop that has led us from that definition to the recurrence relation and thence, via Marsden's identity, to (40). Indeed, by defining

$$\psi_{jk}^+ := (t_{j+1} - \cdot)_+ \cdots (t_{j+k-1} - \cdot)_+, \quad \text{all } j,$$

we are entitled to write (40) (for the case $\nu = 1$) as

$$(\cdot - \tau)_+^{k-1} = \sum_j \psi_{jk}^+(\tau) \mathrm{B}_{jk}, \quad \tau \in \mathbf{t}, \tag{42}$$

in which the summation is, once again, over all j. The function ψ_{jk}^+ agrees with ψ_{jk} at $(t_i : i < j + k)$, and agrees with the zero polynomial at $(t_i : i > j)$. Since both ψ_{jk} and 0 are polynomials of order k, it follows that

$$[t_i, \ldots, t_{i+k}]\psi_{jk}^+ = 0 \quad \text{for} \quad j \neq i.$$

Therefore, applying $[t_i, \ldots, t_{i+k}]$ to both sides of (42) as functions of τ, we find that

$$[t_i, \ldots, t_{i+k}](x - \cdot)_+^{k-1} = [t_i, \ldots, t_{i+k}]\psi_{ik}^+ \, \mathrm{B}_{ik}(x).$$

Since ψ_{ik}^+ agrees at $(t_i, \ldots, t_{i+k})$ with the kth degree polynomial

$$\tau \mapsto \frac{(\tau - t_i)}{t_{i+k} - t_i} \psi_{ik}(\tau) = \frac{(-)^{k-1}}{t_{i+k} - t_i} \tau^k + \text{lower order terms},$$

it follows, with $(-)^{k-1}(x - \cdot)_+^{k-1} = (\cdot - x)_+^{k-1}$, that

$$(t_{i+k} - t_i)[t_i, \ldots, t_{i+k}](\cdot - x)_+^{k-1} = \mathrm{B}_{ik}(x).$$

□

It follows from (40) that

$$(\cdot - t_i)_+^{k-\nu} \in \$_{k,\mathbf{t}} \quad \text{for} \quad 1 \leq \nu \leq \#t_i \tag{43}$$

(on $I_{k,\mathbf{t}}$), and so implies the major part of the following basic theorem.

(44) Theorem. (Curry and Schoenberg) *For a given strictly increasing sequence* $\boldsymbol{\xi} = (\xi_i)_1^{l+1}$, *and a given nonnegative integer sequence* $\boldsymbol{\nu} = (\nu_i)_2^l$ *with* $\nu_i \leq k$, *all* i, *set*

$$n := k + \sum_{i=2}^{l}(k - \nu_i) = kl - \sum_{i=2}^{l} \nu_i = \dim \Pi_{<k,\boldsymbol{\xi},\boldsymbol{\nu}}$$

and let $\mathbf{t} := (t_i)_1^{n+k}$ *be the nondecreasing sequence obtained from* $\boldsymbol{\xi}$ *by the*

following two requirements:
(i) for $i = 2, \dots, l$, *the number* ξ_i *occurs exactly* $k - \nu_i$ *times in* $\mathbf{t}$;
(ii) $t_1 \le t_2 \le \cdots \le t_k \le \xi_1$ *and* $\xi_{l+1} \le t_{n+1} \le \cdots \le t_{n+k}$.
Then, the sequence $\mathrm{B}_1, \dots, \mathrm{B}_n$ *of B-splines of order* k *for the knot sequence* $\mathbf{t}$ *is a basis for* $\Pi_{<k,\boldsymbol{\xi},\boldsymbol{\nu}}$, *considered as functions on* $I_{k,\mathbf{t}} = [t_k \,..\, t_{n+1}]$. *In symbols, then*

$$\$_{k,\mathbf{t}} = \Pi_{<k,\boldsymbol{\xi},\boldsymbol{\nu}} : \text{ on } I_{k,\mathbf{t}}.$$

PROOF. The proof is basic Linear Algebra. Let $I = [a \,..\, b] := [t_k \,..\, t_{n+1}]$ be the basic interval for $\$_{k,\mathbf{t}}$. By its very definition, the space $\$_{k,\mathbf{t}}$ is spanned by the n-sequence $(\mathrm{B}_j : j = 1, \dots, n)$, and, by (40), contains the sequence

$$(45) \quad (\cdot - a)^{k-\mu}, \mu = 1, \dots, k; (\cdot - t_i)_+^{k-\mu}, \mu = 1, \dots, \#t_i, \text{ for } a < t_i < b,$$

of

$$k + \sum_{a<t_i<b} \#t_i = k + \sum_{i=2}^{l} (k - \nu_i) = n$$

terms which, from Chapter VIII, is known to be a basis for $\Pi_{<k,\boldsymbol{\xi},\boldsymbol{\nu}}$. □

A different proof can be found in the original paper, Curry & Schoenberg [1966]. Note the following remarkable consequence of this theorem: B_{jk} has one more continuous derivative than do the two functions, $\mathrm{B}_{j,k-1}$ and $\mathrm{B}_{j+1,k-1}$, hence, the two weights, ω_{jk} and $(1 - \omega_{j+1,k})$, in the recurrence relation (14) must be exactly right to cancel, at each t_i, $i = j, \dots, j+k$, the jump discontinuities in a certain derivative of the two lower-order B-splines.

(46) Remark The argument from Linear Algebra used in the proof of the Curry-Schoenberg theorem is the following: Suppose that we know a basis, $(f_1, \dots, f_n)$, for the linear subspace F and that we further know a sequence $(g_1, \dots, g_m)$ whose span, G, contains each of the f_i. Then, of course, $F \subseteq G$ and so

$$n = \dim F \le \dim G \le m.$$

If we now know, in addition, that $n = m$, then necessarily $F = G$. Moreover, then necessarily $\dim G = n$, hence the sequence $(g_1, \dots, g_n)$ must be a basis for G since it is minimally spanning. In particular, it must be linearly independent. □

(47) B-spline Property (vi): Local linear independence. *For any knot sequence* $\mathbf{t}$, *and any interval* $I = [a \,..\, b] \subseteq I_{k,\mathbf{t}}$ *containing finitely many of the* t_i, *the sequence*

$$B := (\mathrm{B}_{j,k,\mathbf{t}|I} : \mathrm{B}_{j,k,\mathbf{t}|I} \neq 0)$$

is a basis for $\Pi_{<k,\boldsymbol{\xi},\boldsymbol{\nu}|I}$, *with* $\boldsymbol{\xi}$ *the strictly increasing sequence containing* a,

b, as well as every $t_i \in I$, *and* $\nu_i := k - \min(k, \#\{r : t_r = \xi_i\})$. *In particular, B is linearly independent.*

PROOF. If t contains any knot of multiplicity $> k$, reduce its multiplicity to k. This will only remove B-splines that are 0, hence will not change the sequence B. Further, omit from t any t_i for which neither B_i nor B_{i-k} has support in I. This, too, will not change B. At this point, B consists of the restriction to I of all B-splines of order k for the knot sequence t, with each interior break ξ_i occurring exactly $k - \nu_i$ times in t, hence Theorem (44) finishes the proof. □

(48) Corollary. *For all* $p \in \Pi_{<k}$, $D_\tau(\lambda_{jk}p) = 0$.

PROOF. Since the B-spline sequence (B_{jk}) is linearly independent, the coefficients in (34) are uniquely determined. In particular, they cannot depend on the free parameter τ that appears in the Definition (35) of λ_{jk}. □

B stands for basis The Curry-Schoenberg Theorem enables the construction of a B-spline basis for any particular pp space $\Pi_{<k,\xi,\nu}$, by providing a recipe for an appropriate knot sequence t. This choice of t translates the desired amount of smoothness at a break (as specified by ν) into a corresponding number of knots at that site, with *fewer* knots corresponding to *more* continuity conditions, in such a way that always

(49) number of continuity conditions at ξ + number of knots at $\xi = k$.

Thus a k-fold knot at a site corresponds to no continuity conditions whatever at that site while, on the other extreme, no knot at a site enforces k continuity conditions there, that is, the two polynomial pieces meeting there must agree identically (by Theorem I(14)). In addition to these $\sum_2^l (k - \nu_i)$ (interior) knots $\xi_2 \le t_{k+1} \le \cdots \le t_n \le \xi_l$, there are k initial and k final knots that are arbitrary except that they must not lie in the interior of the *basic interval* $[\xi_1 \,..\, \xi_{l+1}]$.

The theorem leaves open the choice of the first k and of the last k knots. A convenient choice is

$$t_1 = \cdots = t_k = \xi_1, \quad t_{n+1} = \cdots = t_{n+k} = \xi_{l+1}$$

which then allows one to include the choice of these knots under the same pattern as the choice of the other knots, by using

$$\nu_1 = 0 = \nu_{l+1}.$$

In a way, we impose *no* continuity conditions at the endpoints ξ_1 and ξ_{l+1}

of the interval of interest. This is consistent with the fact that the B-spline basis provides a valid representation for elements of $\Pi_{<k,\xi,\nu}$ only on the interval $[t_k \,..\, t_{n+1}]$, that is, it makes the basic interval for $\$_{k,\mathbf{t}}$ coincide with the basic interval for $\Pi_{<k,\xi,\nu}$.

With this choice for the end knots, the construction of the knot sequence $\mathbf{t} = (t_i)_1^{n+k}$ from the break sequence $\xi = (\xi_i)_1^{l+1}$ and the integer sequence $\nu = (\nu_i)_1^{l+1}$ is carried out in the SPLINE TOOLBOX (de Boor [1990]$_2$) by the command `t = augknt(`ξ, ν`)`; this can be visualized as in the following diagram.

breaks	ξ_1	ξ_2	ξ_3	$\cdots$	ξ_l	ξ_{l+1}
no. of continuity conditions	$\nu_1 = 0$	ν_2	ν_3	$\cdots$	ν_l	$\nu_{l+1} = 0$
corresp. knot multiplicity	$k - \nu_1 = k$	$k - \nu_2$	$k - \nu_3$	$\cdots$	$k - \nu_l$	$k - \nu_{l+1} = k$
resulting knots	t_1	t_{k+1}	$t_{2k-\nu_2+1}$	$\cdots$	.	t_{n+1}
	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	t_k	$t_{2k-\nu_2}$	$t_{3k-\nu_2-\nu_3}$		t_n	t_{n+k}

(50) FIGURE. Converting breaks and continuity conditions into knots of appropriate multiplicity.

Theorem (44) allows us to represent pp functions in terms of B-splines and so gives rise to the B-form for a pp function.

(51) Definition. *The* **B-form** *for* $f \in \Pi_{<k,\xi,\nu}$ *consists of*

(i) the integers k *and* n*, giving the order of* f *(as a pp function) and the number of linear parameters (that is,* $n = kl - \sum_i \nu_i = \dim \Pi_{<k,\xi,\nu}$*), respectively;*

(ii) the vector $\mathbf{t} = (t_i)_1^{n+k}$ *containing the knots (possibly partially coincident and constructed from* ξ *and* ν *as in Theorem (44)) in increasing order; and*

(iii) the vector $\alpha = (\alpha_i)_1^n$ *of the coefficients of* f *with respect to the B-spline basis* $(\mathrm{B}_i)_1^n$ *for* $\Pi_{<k,\xi,\nu}$ *on the knot sequence* $\mathbf{t}$.

In terms of these quantities, the value of f at a site x in $[t_k \,..\, t_{n+1}]$ is given by

$$f(x) = \sum_{i=1}^{n} \alpha_i \mathrm{B}_i(x). \tag{52}$$

In particular, if $t_j \le x \le t_{j+1}$ for some $j \in \{k, \ldots, n\}$, then

$$f(x) = \sum_{i=j-k+1}^{j} \alpha_i \mathrm{B}_i(x).$$

This means that we have recaptured an important feature of the ppform, nameiy that the value at a site depends only on k of the coefficients.

Note that the B-form for a pp function of order k involves at least $2k$ knots (and usually more). By contrast, a spline function of order k may have as few as $k+1$ knots (but not fewer). If we construct the ppform (see Definition VII(6)) for a spline of order k with the $n+k$ knots $t_1, \ldots, t_{n+k}$ on the interval $[a\,..\,b] = [t_1\,..\,t_{n+k}]$ of interest, we may end up with as many as $n+k-1$ polynomial pieces. If now $t_1 < t_2$ and $t_{n+k-1} < t_{n+k}$, then, on constructing the B-form for this pp function on $[a\,..\,b] = [t_1\,..\,t_{n+k}]$, we obtain a spline with $n+k+2(k-1)$ knots. The additional knots derive from those arbitrary additional $k-1$ knots at each end. But, if all computations have been carried out exactly (that is, without rounding errors), then we would find that the first $k-1$ and the last $k-1$ B-spline coefficients in the B-form obtained are zero as they must be, because of the linear independence of the B-spline sequence (see Problem X.4).

Conversion from one form to the other Conversion from a B-form to the ppform for f is easily accomplished *if* one knows how to evaluate and differentiate (52) stably and efficiently: the $l+1$ distinct sites among the numbers $t_k, \ldots, t_{n+1}$ are found (with l determined by such a search) and stored in $\xi_1, \ldots, \xi_{l+1}$ in increasing order and, for $i = 1, \ldots, l$, the number $D^{j-1}f(\xi_i^+)$ is computed and stored in C_{ij}, $j = 1, \ldots, k$. We discuss the subroutine BSPLPP which accomplishes this conversion in the next chapter after we have described how to evaluate B-splines stably and efficiently.

The conversion from the ppform to a B-form for f is more difficult because the ppform contains no *explicit* information about the smoothness of f at breaks, that is, about the (minimal) knot multiplicity necessary to represent f as a spline function, nor could such information be derived *reliably* numerically, that is, in finite precision arithmetic, from the ppform. But if f is, by some means, *known* to lie in $\Pi_{<k,\xi,\nu}$ for a certain ν, then the appropriate knot sequence $\mathbf{t}$ can be constructed from ξ and ν as in Theorem (44) and the corresponding coefficient sequence α can be obtained from (55) below. If f is so representable, then the sites τ_j that appear (implicitly) in that formula can always be chosen to be one of the breaks ξ_i so that the required derivatives can be read off directly from the ppform of f. This is exactly what is done, with some care, in the conversion command `fn2fm(pp,'B-')` in the SPLINE TOOLBOX (de Boor [1990]$_2$).

The formula (55) is based on the observation that (34) holds, in fact, for every $p \in \$_{k,\mathbf{t}}$ provided that the point τ appearing in the Definition (35) of λ_{jk} is chosen from the interval $[t_j^+\,..\,t_{j+k}^-]$. Here, having, for example, $\tau = t_j^+$ means that $D^{\nu-1}f(\tau) = \lim_{h\to 0+} D^{\nu-1}f(t_j+h)$.

(53) B-spline Property (vii): Dual functionals. (de Boor & Fix [1973]) *For any* $f \in \$_{k,\mathbf{t}}$,

$$f = \sum_j \lambda_{jk} f \, \mathrm{B}_{jk},$$

with

$$\lambda_{jk} f := \sum_{\nu=1}^{k} \frac{(-D)^{k-\nu}\psi_{jk}(\tau_j)}{(k-1)!} D^{\nu-1} f(\tau_j) \tag{54}$$

and $t_j^+ \le \tau_j \le t_{j+k}^-$, *all* j. *Hence*

$$\lambda_{ik}\Big(\sum_j \alpha_j \, \mathrm{B}_j\Big) = \alpha_i, \quad \text{all } i. \tag{55}$$

It is remarkable that τ_j can be chosen arbitrarily in the interval $[t_j^+ \,..\, t_{j+k}^-]$. The reason behind this is Corollary (48).

PROOF. It is sufficient to prove that, for any i,

$$\lambda_{ik}\mathrm{B}_j = \delta_{ij}, \quad \text{all } j. \tag{56}$$

For this, assume that $\tau_i \in [t_r^+ \,..\, t_{r+1}^-]$. Then (56) is trivially satisfied for every

$$j \notin \{r-k+1, \ldots, r\} =: J.$$

For $j \in J$, let p_j be the polynomial of order k that agrees with B_j on $[t_r \,..\, t_{r+1}]$. Then

$$\lambda_{ik}\mathrm{B}_j = \lambda_{ik} p_j.$$

On the other hand,

$$p_j = \sum_{s \in J} (\lambda_{sk} p_j) p_s, \tag{57}$$

since this holds on $[t_r \,..\, t_{r+1}]$ by (34). This forces $\lambda_{ik}p_j$ $(= \lambda_{ik}\mathrm{B}_j)$ to equal δ_{ij}, since the sequence $(p_s : s \in J)$ is linearly independent, by (34) (since (34) shows that the k-sequence $(p_s : s \in J)$ spans the k-dimensional space $\Pi_{<k}$, hence must be a basis for $\Pi_{<k}$) or, more directly, by the local linear independence (47) of the B-splines. □

(58) Example: Conversion to B-form

Find the B-form of

$$f(x) = (x-3)(x-6)(x-9) \tag{59}$$

on $[0..10]$ as a cubic spline with simple interior knots $1, 2, \ldots, 9$. Then $t_1 = \cdots = t_4 = 0$; $t_{4+i} = i$ for $i = 1, \ldots, 9$; and, finally, $t_{n+1} = \cdots = t_{n+4} = 10$ with $k = 4$ and $n = 13$. In (54), we choose $\tau_j = t_{j+2}$, all j, in which case the formula specializes to

$$\alpha_j = f(t_{j+2}) + \frac{1}{3}(\Delta t_{j+2} - \Delta t_{j+1}) f'(t_{j+2}) - \frac{1}{3}\Delta t_{j+1}\Delta t_{j+2} f''(t_{j+2})/2. \tag{60}$$

By way of an exercise, we find the truncated Taylor expansion for f around 0, that is the numbers $D^i f(0)/i!$, $i = 1, 2, 3, 4$, from its Newton form (59) by the Nested Multiplication Algorithm I(23). Starting from the boldfaced diagonal, this produces the next three diagonals in a divided difference table for f as follows.

x	$f(x)$	1.d.d	2.d.d.	3.d.d
3				
		0		**1**
6			**0**	
	0			1
9			−3	
		18		1
0	−162		−9	
		99		**1**
0	−162		−18	
		99		
0	−162			

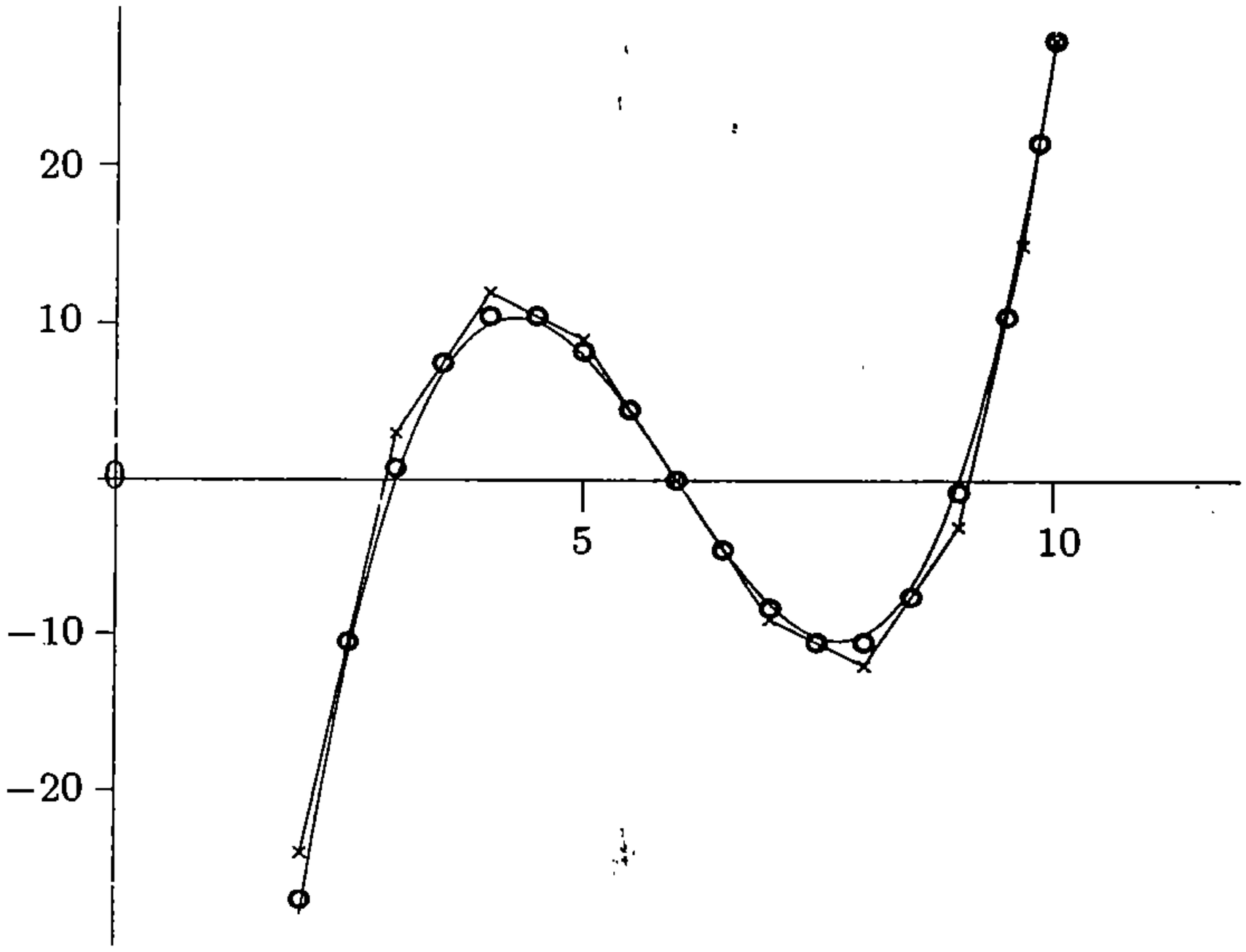
This shows that $f(x) = -162 + x(99 + x(-18 + x))$. The program below uses nested multiplication to find from this the numbers $f(t_{j+2})$, $f'(t_{j+2})$ and $f''(t_{j+2})/2$ and then calculates α_j by (60). The program also computes the value of f at the Greville sites

$$t^*_{jk} = (t_{j+1} + \cdots + t_{j+k-1})/(k-1) =: \texttt{TAVE}(j) \tag{61}$$

from (39) in order to bring out the important point that $\alpha_j \sim f(t^*_{jk})$. This becomes even more striking when we increase the number of knots, using the points $1/2, 1, 3/2, \ldots, 9, 19/2$ as interior knots; see Figure (62). We pick up on this very important point in Chapter XI.

```
CHAPTER IX. EXAMPLE COMPARING THE B-REPRESENTATION OF A CUBIC  F  WITH
C  ITS VALUES AT KNOT AVERAGES.
C
      INTEGER I,ID,J,JJ,N,NM4
      REAL BCOEF(23),D(4),DO(4),DTIP1,DTIP2,F(23),T(27),TAVE(23),X
C            THE TAYLOR COEFFICIENTS AT  0  FOR THE POLYNOMIAL  F  ARE
      DATA DO /-162.,99.,-18.,1./
C
C                    SET UP KNOT SEQUENCE IN THE ARRAY  T .
      N = 13
      DO 5 I=1,4
         T(I) = 0.
    5    T(N+I) = 10.
      NM4 = N-4
      DO 6 I=1,NM4
    6    T(I+4) = FLOAT(I)
C
      DO 50 I=1,N
C        USE NESTED MULTIPLICATION TO GET TAYLOR COEFFICIENTS  D  AT
C              T(I+2)  FROM THOSE AT  0 .
         DO 20 J=1,4
   20       D(J) = DO(J)
         DO 21 J=1,3
            ID = 4
            DO 21 JJ=J,3
               ID = ID-1
   21          D(ID) = D(ID) + D(ID+1)*T(I+2)
C
C                 COMPUTE B-SPLINE COEFFICIENTS BY FORMULA (9).
         DTIP1 = T(I+2) - T(I+1)
         DTIP2 = T(I+3) - T(I+2)
         BCOEF(I) = D(1) + (D(2)*(DTIP2-DTIP1)-D(3)*DTIP1*DTIP2)/3.
C
C                   EVALUATE  F  AT CORRESP. KNOT AVERAGE.
         TAVE(I) = (T(I+1) + T(I+2) + T(I+3))/3.
         X = TAVE(I)
   50    F(I) = DO(1) + X*(DO(2) + X*(DO(3) + X*DO(4)))
C
      PRINT 650, (I,TAVE(I), F(I), BCOEF(I),I=1,N)
  650 FORMAT(45H  I   TAVE(I)       F AT TAVE(I)      BCOEF(I)//
     *       (I3,F10.5,2F16.5))
                                        STOP
      END
```

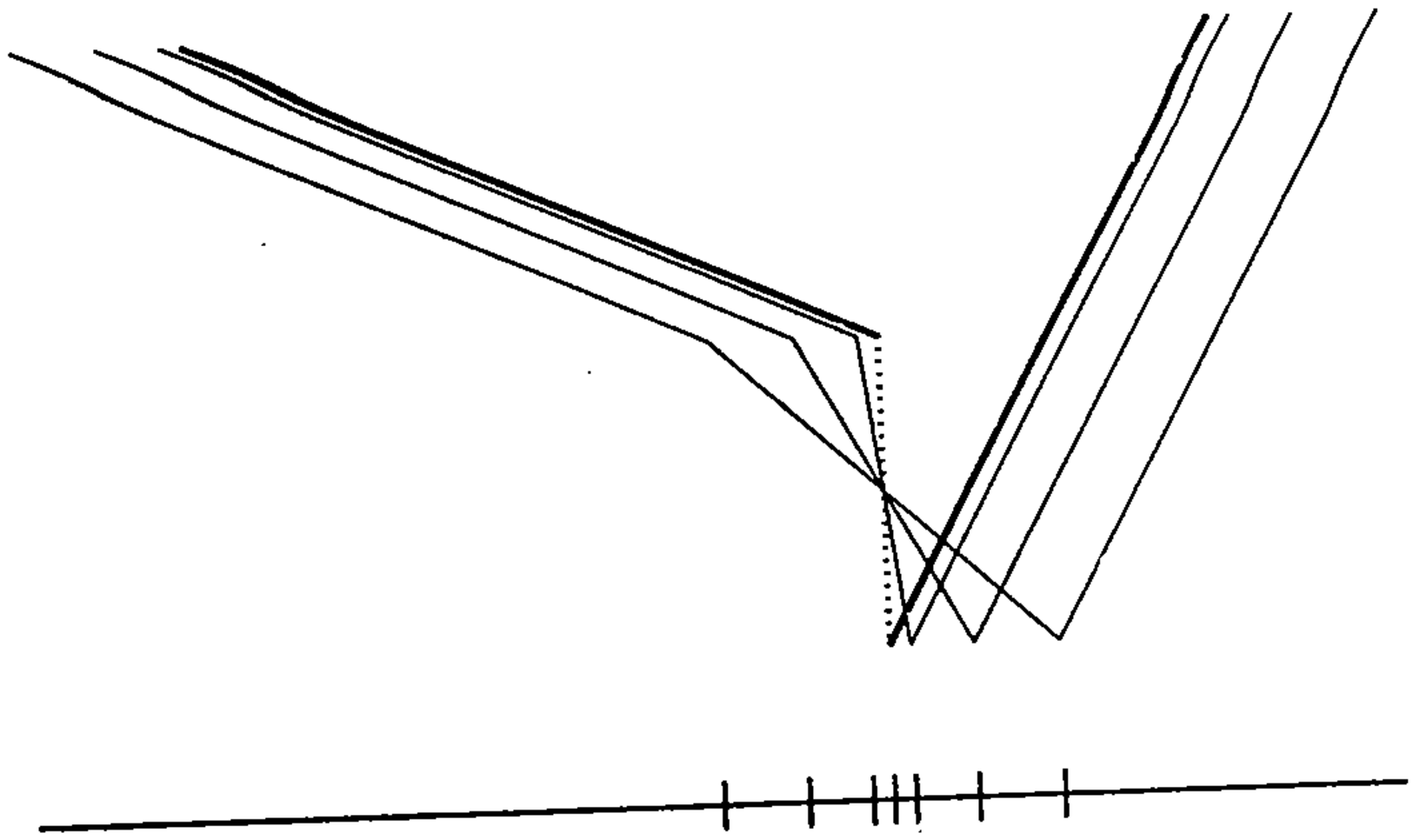
I	TAVE(I)	F AT TAVE(I)	BCOEF(I)
1	0.00000	-162.00000	-162.00000
2	0.33333	-130.96297	-129.00000
3	1.00000	-80.00000	-75.00000
4	2.00000	-28.00000	-24.00000
5	3.00000	0.00000	3.00000
6	4.00000	10.00000	12.00000
7	5.00000	8.00000	9.00000
8	6.00000	0.00000	0.00000
9	7.00000	-8.00000	-9.00000
10	8.00000	-10.00000	-12.00000
11	9.00000	0.00000	-3.00000
12	9.66667	16.29628	15.00000
13	10.00000	28.00000	28.00000



(62) FIGURE. B-spline coefficients model the function they represent. A cubic polynomial f on $[0\,..\,10]$ and its B-spline coefficients (α_j) plotted to show that $\alpha_j \sim f(t^*_{jk})$. The points marked $\times$ come from the uniform knot sequence $0, 1, \ldots, 10$, those marked $\odot$ from the finer uniform knot sequence $0, 1/2, \ldots, 10$.

□

We now return to the objection voiced earlier that our Definition (26) of a spline function does not completely jibe with the definition of a cubic spline or a parabolic spline given in Chapter III–VI. In these earlier chapters, and in the early history of spline theory (see, for example, Schoenberg [1946]), a spline function of order k was defined to be a pp function of order k on some (finite or infinite) interval with $k-2$ continuous derivatives there. In other words, "spline function" meant a pp function that was as smooth as it could be without simply reducing to a polynomial. But it was soon found that pp functions of less than this maximum smoothness were also quite interesting and useful. For instance, piecewise cubic Hermite interpolation or piecewise cubic Bessel interpolation is at times preferred to cubic spline interpolation. Some people have called such pp functions with less than maximum (nontrivial) smoothness **deficient splines.** I will not use this term here since it contains a value judgement for which I have no justification. Rather, I call such functions splines with multiple knots since



(63) FIGURE. The formation of a double knot.

we can think of them as having been obtained from a spline with simple knots (that is, one of the original splines) by letting some knots coalesce. This is illustrated in Figure (63), where a discontinuous piecewise linear function is shown as the limit of a linear spline with simple knots as two knots coalesce.

But, with this extension of the word "spline", *all* pp functions become splines, and one wonders why one should bother with the term "spline" at all. In order to retain some special and useful meaning for the word "spline", I have adopted the definition of spline function, given earlier in this chapter, as linear combination of B-splines. In this way, the notion of "spline" is simply a particular way of looking at pp functions. This way makes it particularly easy to see the nearby splines with simple knots when looking at a particular pp function. This is so because a B-spline doesn't change much if one changes its $k+1$ knots a little bit. Therefore, if one has multiple knots, then it is very easy to find a B-spline almost like it with simple knots: Simply replace each knot of multiplicity $r > 1$ by r simple knots nearby.

Problems

1. Verify the numerical values given in Figure (25) by constructing the appropriate divided difference tables.

2. (a) Prove formula (9) for the cardinal B-spline. (Hint: Verify, by induction on n, the formula $n![0,1,\ldots,n]f = \Delta^n f(0) := \sum_{i=0}^{n}(-)^{n-i}\binom{n}{i}f(i)$.)
(b) Use the formula in a program to graph B_i for $k = 5$, 10, 15, 20. (Any problems?)

3. Prove: If f is pp of order k, with

$$f(x) = \begin{cases} p(x), & x < \tau \\ q(x), & x > \tau \end{cases}, \quad p, q \in \Pi_{<k},$$

and $\text{jump}_\tau D^j f = 0$ for $j = 0, \ldots, k-1$, then $f = q = p$.

4. Verify the assertion in Theorem (44) that $\dim \Pi_{<k,\xi,\nu} = k + \sum_2^l (k - \nu_i)$.

5. Prove: If $f \in \$_{k,\mathbf{t}}$ vanishes outside the interval $[t_r \,..\, t_s]$ for some $r < s < r + k$, then $f = 0$. Conclude that B-splines are *splines of minimal support*. (Hint: Use (53).)

6. Consider the linear functional λ_i of (53) as a function of τ_i, that is, consider the function

$$A_f(\tau) := \sum_{r=1}^{k} (-)^{k-r} \psi_{ik}^{(k-r)}(\tau) D^{r-1} f(\tau).$$

(a) Prove that $A_f(\tau)$ does not depend on τ in case $f \in \Pi_{<k}$. (Hint: Prove that $A_f \in \Pi_{<k}$ and calculate A_f'.)
(b) Prove directly, that is, without recourse to Theorem (44) or (53), that $A_f(\tau)$ does not depend on τ in case $t_i < \tau < t_{i+k}$ and $f \in \Pi_{<k,\xi,\nu}$, assuming $\mathbf{t}$ is derived from ξ and ν as in Theorem (44). (Because of (a), this only requires showing that $\text{jump}_{t_j} A_f = 0$ for t_j in $(t_i \,..\, t_{i+k})$.)

7. Verify that the spline approximation $Ag := \sum_i (\lambda_i g) \mathrm{B}_{i,k,\mathbf{t}}$ is *local* and satisfies $Ag = g$ for all $g \in \$_{k,\mathbf{t}}$.

8. (a) Verify that $\mu_i \mathrm{B}_j = \delta_{ij}$ for $k = 3$, $\mathbf{t}$ arbitrary, and

$$\mu_i g := (-g(t_{i+1}) + 4g(t_{i+3/2}) - g(t_{i+2}))/2,$$

with $t_{i+3/2} := (t_{i+1} + t_{i+2})/2$.
(b) Verify that the approximation $Ag := \sum_i (\mu_i g) \mathrm{B}_{i,3,\mathbf{t}}$ satisfies $\|g - Ag\| \leq 4 \,\text{dist}\,(g, \$_{k,\mathbf{t}})$. (Hint: Use the fact that the B_i are nonnegative and form a partition of unity, also use Problem III.2.)
(c) Prove that the approximation scheme in (b) is local.

9. In Figure (62), half of the B-coefficients marked $\odot$ seem to lie on the broken line defined by the B-coefficients marked $\times$. Prove that this is no accident.

10. Use Problem I.8 to obtain the following truncated-power representation of the B-spline:

$$\mathrm{B}_j(x) = (t_{j+k} - t_j) \sum_{r=j}^{j+k} d_r \, (t_r - x)_+^{k-1-m(r)} \frac{(k-1)!}{(k-1-m(r))}$$

for certain weights d_r and with $m(r) := \max\{s : r - s \geq j, t_{r-s} = t_r\}$, $r = j, \ldots, j + k$.

11. Prove that, for any $\tau_1 < \cdots < \tau_k$, the sequence $((\cdot - \tau_j)^{k-1} : j = 1, \ldots, k)$ is a basis for $\Pi_{<k}$.

X

The Stable Evaluation of B-Splines and Splines;

BSPLVB, BVALUE, BSPLPP

In this chapter, we continue our discussion of the properties of B-splines, albeit in terms of splines, that is, linear combinations of B-splines or the B-spline series. We concentrate on the differentiation and integration of splines and on their stable evaluation with the aid of the recurrence relation. Also, once we are able to evaluate a spline and its derivatives safely, we can convert from B-form to ppform as outlined in the preceding chapter.

Stable evaluation of B-splines The direct evaluation of the B-spline B_i from its definition IX(2) as a divided difference has to be carried out with caution because of the possible loss of significance during the computation of the various difference quotients (Problem IX.2). Also, special provisions have to be made in the case of repeated or multiple knots (Problem IX.1). Such a calculation would, in fact, amount to evaluating B_i from its representation in terms of the truncated power basis (see Problem IX.10) and would, therefore, be beset with precisely the difficulties that we hoped to avoid by introducing the B-spline basis in the first place.

Fortunately, it is possible to evaluate B-splines with the aid of the recurrence relation IX(14) which requires no special arrangement for multiple knots and does not suffer (unnecessary) loss of significance (see Cox [1972] for a rounding error analysis of the process). B-splines became a viable computational tool only after the discovery (e.g., de Boor [1971], Cox [1972]) of these recurrence relations.

The subroutine BSPLVB The recurrence relation IX(14) leads directly to an algorithm for the *simultaneous* generation of the values at x of the k B-splines of order k that are possibly not zero there.

Suppose $t_i < t_{i+1}$ and $x \in [t_i \,..\, t_{i+1}]$. The values of all B-splines not automatically zero at x fit into a triangular array as follows (we write B_{jr} instead of $B_{jr}(x)$ to keep the table simple).

				0	
			0		$B_{i-k+1,k}$
		0		$B_{i-k+2,k-1}$	
	0		$B_{i-2,3}$		$B_{i-k+2,k}$
0		$B_{i-1,3}$		$B_{i-k+3,k-1}$	
	$B_{i-1,2}$				$B_{i-k+3.k}$
B_{i1}					
	B_{i2}				
0		B_{i3}		$B_{i-1,k-1}$	
	0				$B_{i-1,k}$
		0		$B_{i,k-1}$	
					B_{ik}
				0	
					0

(1) FIGURE. The triangular array of B-splines of order $\leq k$ that are nonzero on $[t_i \,..\, t_{i+1}]$.

The boundary of 0's in this table is a reminder of the fact that all other B-splines (of order k or less) not mentioned explicitly in the table vanish at x.

It is clear that we can generate this triangular table column by column, since we know the first column by IX(11), and can compute each entry in a subsequent column by IX(14). Of course, in the computation of the first and the last nonzero entry in a column, we use the fact that one of their neighbors to the left is zero. Suppose we have already computed the j numbers $B_{i-j+1,j}(x), \ldots, B_{ij}(x)$ and have stored them in the vector $\mathbf{b} := (b_r)_1^j$. If the vector $\mathbf{b}' := (b'_r)_1^{j+1}$ is to contain the $j+1$ numbers $B_{i-j,j+1}(x), \ldots, B_{i,j+1}(x)$ of the next column in that order, then, by IX(14),

$$(2) \qquad b'_r = (x - t_{i-j+r-1}) \frac{b_{r-1}}{t_{i+r-1} - t_{i-j+r-1}} + (t_{i+r} - x) \frac{b_r}{t_{i+r} - t_{i-j+r}}$$

for $r = 1, \ldots, j+1$ and with $b_0 := b_{j+1} := 0$. Introduce the quantities

$$\delta^L_{sj} := x - t_{i-j+s}, \quad \delta^R_s := t_{i+s} - x, \qquad s = 1, \ldots, k-1.$$

Then we can write (2) in the form

$$(3) \qquad b'_r = \delta^L_{r-1,j} \frac{b_{r-1}}{\delta^L_{r-1,j} + \delta^R_{r-1}} + \delta^R_r \frac{b_r}{\delta^L_{rj} + \delta^R_r}, \quad r = 1, \ldots, j+1,$$

Note that our assumption $t_i < t_{i+1}$ ensures that none of the denominators in (3) vanishes; in fact, they are all at least as big as Δt_i. Note also that $\delta^L_{rj} = \delta^L_{j+1-r}$, with

$$\delta^L_s := x - t_{i+1-s}$$

conveniently independent of j.

The computation of (b'_r) and (b_r) as in (3) can be arranged as follows.

```
2/3    b'_1 := 0
2/4    for r = 1,...,j, do:
   2/4/1  term := b_r/(δ^L_{j+1-r} + δ^R_r)
   2/4/2  b'_r := b'_r + δ^R_r·term
   2/4/3  b'_{r+1} := δ^L_{j+1-r}·term
```

Here, it is possible to store $\mathbf{b}'$ over $\mathbf{b}$ by using an additional word of temporary storage. Starting with $j = 1$, in which case $\mathbf{b}$ has the single entry $b_1 = 1$, repetition of the 2/4-loop for $j = 1, \ldots, k-1$ produces eventually in $\mathbf{b}$ the desired vector of k numbers $\mathrm{B}_{i-k+1,k}(x), \ldots, \mathrm{B}_{ik}(x)$, as follows.

(4)

```
1   b_1 := 1
2   for j = 1,...,k-1, do:
      2/1  δ^R_j := t_{i+j} - x
      2/2  δ^L_j := x - t_{i+1-j}
      2/3  saved := 0
      2/4  for r = 1,...,j, do:
         2/4/1  term := b_r/(δ^R_r + δ^L_{j+1-r})
         2/4/2  b_r := saved + δ^R_r·term
         2/4/3  saved := δ^L_{j+1-r}·term
    2/5  b_{j+1} := saved
```

This algorithm is carried out in the following subprogram BSPLVB.

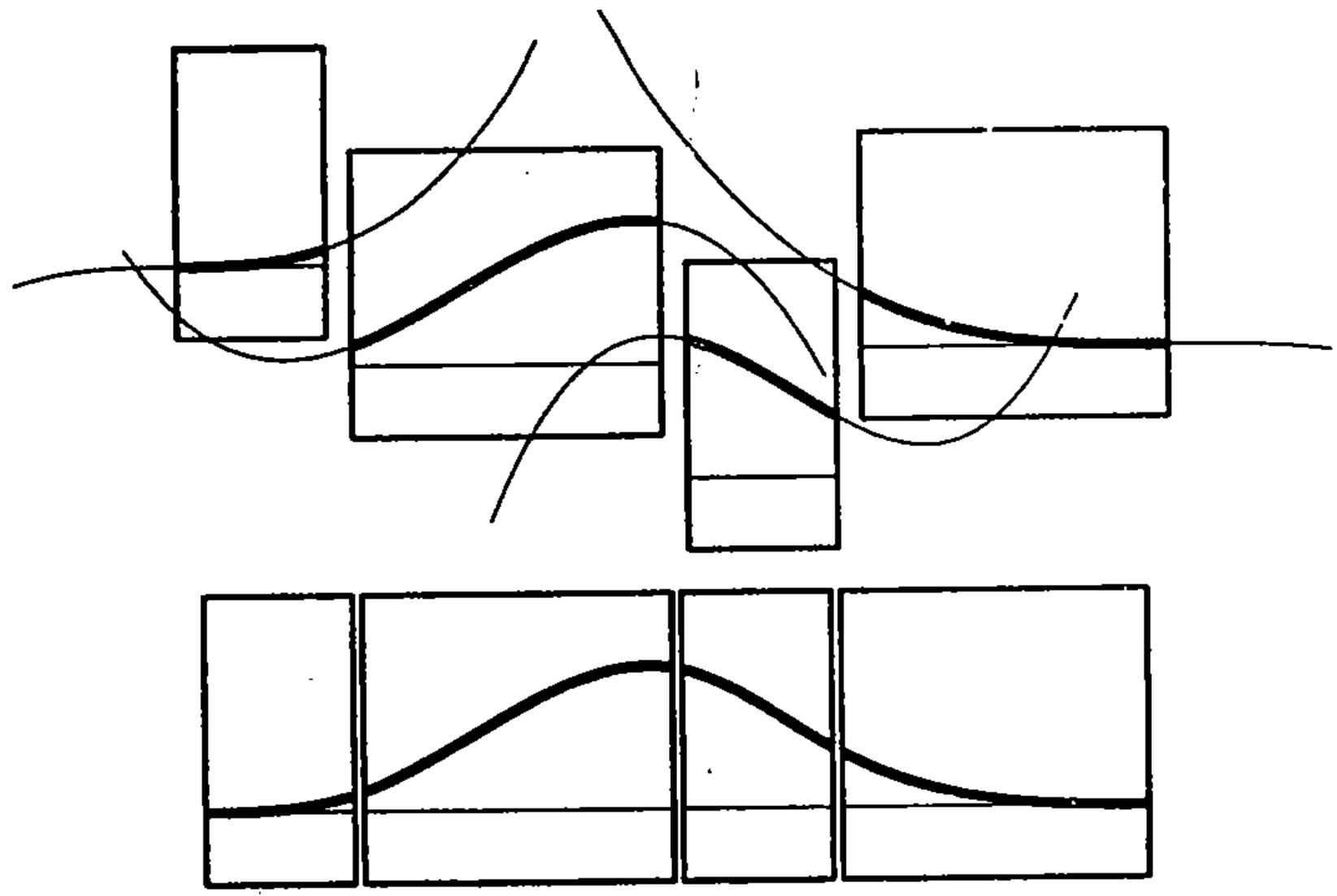
```
      SUBROUTINE BSPLVB ( T, JHIGH, INDEX, X, LEFT, BIATX )
C  FROM  * A PRACTICAL GUIDE TO SPLINES *  BY C. DE BOOR
CALCULATES THE VALUE OF ALL POSSIBLY NONZERO B-SPLINES AT  X  OF ORDER
C
C                JOUT  =  MAX( JHIGH , (J+1)*(INDEX-1) )
C
C  WITH KNOT SEQUENCE  T .
C
C******  I N P U T  ******
C  T.....KNOT SEQUENCE, OF LENGTH  LEFT + JOUT  , ASSUMED TO BE NONDE-
C        CREASING.  A S S U M P T I O N . . .
C                         T(LEFT)  .LT.  T(LEFT + 1)    .
C   D I V I S I O N  B Y  Z E R O  WILL RESULT IF  T(LEFT) = T(LEFT+1)
C  INDEX.....INTEGERS WHICH DETERMINE THE ORDER  JOUT = MAX(JHIGH,
C        (J+1)*(INDEX-1))  OF THE B-SPLINES WHOSE VALUES AT  X  ARE TO
C        BE RETURNED.  INDEX  IS USED TO AVOID RECALCULATIONS WHEN SEVE-
C        RAL COLUMNS OF THE TRIANGULAR ARRAY OF B-SPLINE VALUES ARE NEE-
C        DED (E.G., IN  BSPLPP  OR IN  BSPLVD ). PRECISELY,
C                      IF  INDEX = 1 ,
C        THE CALCULATION STARTS FROM SCRATCH AND THE ENTIRE TRIANGULAR
C        ARRAY OF B-SPLINE VALUES OF ORDERS 1,2,...,JHIGH  IS GENERATED
```

```
C         ORDER BY ORDER , I.E., COLUMN BY COLUMN .
C                       IF  INDEX = 2 ,
C         ONLY THE B-SPLINE VALUES OF ORDER  J+1, J+2, ..., JOUT  ARE GE-
C         NERATED, THE ASSUMPTION BEING THAT  BIATX , J , DELTAL , DELTAR
C         ARE, ON ENTRY, AS THEY WERE ON EXIT AT THE PREVIOUS CALL.
C  JHIGH,
C            IN PARTICULAR, IF  JHIGH = 0, THEN  JOUT = J+1, I.E., JUST
C         THE NEXT COLUMN OF B-SPLINE VALUES IS GENERATED.
C
C  W A R N I N G . . .  THE RESTRICTION   JOUT .LE. JMAX (= 20)  IS IM-
C         POSED ARBITRARILY BY THE DIMENSION STATEMENT FOR  DELTAL  AND
C         DELTAR  BELOW, BUT IS  N O W H E R E  C H E C K E D  FOR .
C
C  X.....THE POINT AT WHICH THE B-SPLINES ARE TO BE EVALUATED.
C  LEFT.....AN INTEGER CHOSEN (USUALLY) SO THAT
C                  T(LEFT) .LE. X .LE. T(LEFT+1)   .
C
C******  O U T P U T  ******
C  BIATX.....ARRAY OF LENGTH  JOUT , WITH  BIATX(I)  CONTAINING THE VAL-
C         UE AT  X  OF THE POLYNOMIAL OF ORDER  JOUT  WHICH AGREES WITH
C         THE B-SPLINE  B(LEFT-JOUT+I,JOUT,T)  ON THE INTERVAL (T(LEFT)..
C         T(LEFT+1)) .
C
C******  M E T H O D  ******
C  THE RECURRENCE RELATION
C
C                          X - T(I)                    T(I+J+1) - X
C     B(I,J+1)(X)  =  -----------B(I,J)(X) + ---------------B(I+1,J)(X)
C                     T(I+J)-T(I)              T(I+J+1)-T(I+1)
C
C  IS USED (REPEATEDLY) TO GENERATE THE (J+1)-VECTOR  B(LEFT-J,J+1)(X),
C  ...,B(LEFT,J+1)(X)  FROM THE J-VECTOR  B(LEFT-J+1,J)(X),...,
C  B(LEFT,J)(X), STORING THE NEW VALUES IN  BIATX  OVER THE OLD. THE
C  FACTS THAT
C            B(I,1) = 1  IF  T(I) .LE. X .LT. T(I+1)
C  AND THAT
C            B(I,J)(X) = 0  UNLESS  T(I) .LE. X .LT. T(I+J)
C  ARE USED. THE PARTICULAR ORGANIZATION OF THE CALCULATIONS FOLLOWS AL-
C  GORITHM  (9)  IN CHAPTER X OF THE 1998 TEXT.
C
      INTEGER INDEX,JHIGH,LEFT,   I,J,JMAX,JP1
      PARAMETER (JMAX = 20)
      REAL BIATX,T,X,   DELTAL(JMAX),DELTAR(JMAX),SAVED,TERM
      DIMENSION BIATX(JOUT), T(LEFT+JOUT)
      DATA J/1/
      SAVE J,DELTAL,DELTAR
C
                                          GO TO (10,20), INDEX
   10 J = 1
      BIATX(1) = 1.
      IF (J .GE. JHIGH)                   GO TO 99
C
   20    JP1 = J + 1
         DELTAR(J) = T(LEFT+J) - X
         DELTAL(J) = X - T(LEFT+1-J)
         SAVED = 0.
         DO 26 I=1,J
            TERM = BIATX(I)/(DELTAR(I) + DELTAL(JP1-I))
            BIATX(I) = SAVED + DELTAR(I)*TERM
   26       SAVED = DELTAL(JP1-I)*TERM
         BIATX(JP1) = SAVED
         J = JP1
         IF (J .LT. JHIGH)                GO TO 20
C
   99                                     RETURN
      END
```

(5) Example: To plot B-splines We should like to encourage the reader to look at kth order B-splines for various values of k and various choices of knots. For this reason, we supply here a sample program that uses BSPLVB in order to plot all the B-splines of a given order for a given knot sequence. It happens to be set up to supply the data necessary to plot the parabolic B-splines in Figure IX(25), but is easily changed. For the plotting of a single B-spline, we recommend the function BVALUE discussed later in this chapter.

```
CHAPTER X. EXAMPLE 1. PLOTTING SOME B-SPLINES
CALLS  BSPLVB, INTERV
      INTEGER I,J,K,LEFT,LEFTMK,MFLAG,N,NPOINT
      REAL DX,T(10),VALUES(7),X,XL
C  DIMENSION, ORDER AND KNOT SEQUENCE FOR SPLINE SPACE ARE SPECIFIED...
      DATA N,K /7,3/, T /3*0.,2*1.,3.,4.,3*6./
C  B-SPLINE VALUES ARE INITIALIZED TO 0., NUMBER OF EVALUATION POINTS...
      DATA VALUES /7*0./, NPOINT /31/
C  SET LEFTMOST EVALUATION POINT  XL , AND SPACING  DX  TO BE USED...
      XL = T(K)
      DX = (T(N+1)-T(K))/FLOAT(NPOINT-1)
C
      PRINT 600,(I,I=1,5)
  600 FORMAT('1  X',8X,5('B',I1,'(X)',7X))
C
      DO 10 I=1,NPOINT
         X = XL + FLOAT(I-1)*DX
C                     LOCATE  X  WITH RESPECT TO KNOT ARRAY  T .
         CALL INTERV ( T, N+1, X, LEFT, MFLAG )
         LEFTMK = LEFT - K
C           GET B(I,K)(X)  IN  VALUES(I) , I=1,...,N .  K  OF THESE,
C        VIZ.  B(LEFT-K+1,K)(X), ..., B(LEFT,K)(X),  ARE SUPPLIED BY
C        BSPLVB . ALL OTHERS ARE KNOWN TO BE ZERO A PRIORI.
         CALL BSPLVB ( T, K, 1, X, LEFT, VALUES(LEFTMK+1) )
C
         PRINT 610, X, (VALUES(J),J=3,7)
  610    FORMAT(F7.3,5F12.7)
C
C           ZERO OUT THE VALUES JUST COMPUTED IN PREPARATION FOR NEXT
C        EVALUATION POINT .
         DO 10 J=1,K
   10       VALUES(LEFTMK+J) = 0.
                                          STOP
      END
```

□



(6) FIGURE. The four cubic polynomials that make up a certain cubic B-spline.

(7) Example: To plot the polynomials that make up a B-spline
The input parameter LEFT for BSPLVB is to be chosen so that

$$\mathrm{T(LEFT)} < \mathrm{T(LEFT+1)}$$

and so that

$$\mathrm{T(LEFT)} \le \mathrm{X} \le \mathrm{T(LEFT+1)}.$$

The first condition is absolutely necessary to avoid division by zero, but what if the second condition is violated? A check of equation (2) assures us, by induction on j, that the algorithm (4) generates the value at x of certain k polynomials, each of order k, regardless of any relationship that the specific value x might have to the (fixed) index i. Since the process (4) does give us the value at x of the B-splines $\mathrm{B}_{i-k+1,k},\ldots,\mathrm{B}_{ik}$ if we choose $t_i \le x \le t_{i+1}$, it follows that the algorithm (4) generates, for given x, the value at x of the k polynomials $p_{i-k+1},\ldots,p_i$ of order k for which

$$p_{i-r}(x) = \mathrm{B}_{i-r,k}(x) \qquad \text{for } t_i \le x \le t_{i+1}, \quad r = 0,\ldots,k-1.$$

This observation allows us to compute these polynomials, and so permits us to illustrate how a function as smooth as a cubic B-spline is put together from polynomial pieces. The data for Figure (6) were obtained from BSPLVB with the aid of the following program.

```
CHAPTER X. EXAMPLE 2. PLOTTING THE POL,S WHICH MAKE UP A B-SPLINE
CALLS  BSPLVB
C
      INTEGER IA,LEFT
      REAL BIATX(4),T(11),VALUES(4),X
C                    KNOT SEQUENCE SET HERE....
      DATA T / 4*0.,1.,3.,4.,4*6. /
      DO 20 IA=1,40
         X = FLOAT(IA)/5. - 1.
         DO 10 LEFT=4,7
            CALL BSPLVB ( T, 4, 1, X, LEFT, BIATX )
C
C           ACCORDING TO  BSPLVB  LISTING,  BIATX(.) NOW CONTAINS VALUE
C           AT  X  OF POLYNOMIAL WHICH AGREES ON THE INTERVAL  ( T(LEFT)
C           ..T(LEFT+1) ) WITH THE B-SPLINE  B(LEFT-4 + . ,4,T) . HENCE,
C           BIATX(8-LEFT)  NOW CONTAINS VALUE OF THAT POLYNOMIAL FOR
C           B(LEFT-4 +(8-LEFT) ,4,T) = B(4,4,T) . SINCE THIS B-SPLINE
C           HAS SUPPORT  (T(4),T(8)), IT MAKES SENSE TO RUN  LEFT = 4,
C           ...,7, STORING THE RESULTING VALUES IN  VALUES(1),...,
C           VALUES(4)  FOR LATER PRINTING.
C
   10       VALUES(LEFT-3) = BIATX(8-LEFT)
   20    PRINT 620, X, VALUES
  620 FORMAT(F10.1,4F20.8)
                                        STOP
      END
```

□

Differentiation The fact that the first derivative $D_x(t-x)_+^{k-1}$ with respect to x of the pp function $g(x) = (t-x)_+^{k-1}$ is given by

$$D_x(t-x)_+^{k-1} = -(k-1)(t-x)_+^{k-2}$$

may be used to conclude from IX(3) that

$$\begin{aligned} D\mathrm{B}_{ik}(x) &= ([t_{i+1},\ldots,t_{i+k}] - [t_i,\ldots,t_{i+k-1}])D_x(\cdot - x)_+^{k-1} \\ &= -(k-1)([t_{i+1},\ldots,t_{i+k}] - [t_i,\ldots,t_{i+k-1}])(\cdot - x)_+^{k-2} \\ &= (k-1)\left(\frac{-\mathrm{B}_{i+1,k-1}(x)}{t_{i+k}-t_{i+1}} + \frac{\mathrm{B}_{i,k-1}(x)}{t_{i+k-1}-t_i}\right), \end{aligned} \tag{8}$$

though this requires justification of the interchange $D_x[\ldots](\cdot - x)_+^{k-1} = [\ldots]D_x(\cdot - x)_+^{k-1}$ (for example via Problem IX.10). In accordance with our promise to develop the entire B-spline theory directly from the recurrence relation, without recourse to divided differences, we now prove (8) with the aid of the dual functionals IX(54) and via the B-spline series.

We begin with the observation that

$$D\$_{k,\mathbf{t}} = \{Df : f \in \$_{k,\mathbf{t}}\} = \$_{k-1,\mathbf{t}}, \tag{9}$$

as follows from the Curry-Schoenberg Theorem IX(44). Therefore, by IX(53), for any $f \in \$_{k,\mathbf{t}}$,

$$Df = \sum_j (\lambda_{j,k-1} Df)\, \mathrm{B}_{j,k-1,\mathbf{t}}$$

with

$$\lambda_{j,k-1} Df = \sum_{r=1}^{k-1} (-D)^{r-1} \psi_{j,k-1}(\tau) D^{k-1-r} Df(\tau)/(k-2)!, \quad \text{all } j.$$

Compare this with the coefficients

$$\lambda_{jk} f = \sum_{r=1}^{k} (-D)^{r-1} \psi_{jk}(\tau) D^{k-r} f(\tau)/(k-1)!, \quad \text{all } j,$$

in the B-spline expansion $f = \sum_j \lambda_{jk} f\, \mathrm{B}_{jk}$ for f. Since

$$\begin{aligned} (t_{j+k-1} - \cdot)\psi_{j,k-1} &= \psi_{jk} \\ (t_j - \cdot)\psi_{j,k-1} &= \psi_{j-1,k}, \end{aligned}$$

subtracting the latter from the former gives

$$(t_{j+k-1} - t_j)\psi_{j,k-1} = \psi_{jk} - \psi_{j-1,k}.$$

Therefore,

$$\lambda_{j,k-1} D = (k-1) \frac{\lambda_{jk} - \lambda_{j-1,k}}{t_{j+k-1} - t_j}. \tag{10}$$

This proves

(11) B-spline Property (viii): Differentiation.

$$D\Big(\sum_j \alpha_j \mathrm{B}_{jk} \Big) = (k-1) \sum_j \frac{\alpha_j - \alpha_{j-1}}{t_{j+k-1} - t_j} \mathrm{B}_{j,k-1}. \tag{12}$$

This shows that *the first derivative of a spline function* $\sum_j \alpha_j \mathrm{B}_{jk}$ *can be found simply by differencing its B-spline coefficients*, thereby obtaining the B-spline coefficients of its first derivative, a spline of one order lower.

We have left the limits of summation in (12) unspecified. The precise formulation is

$$D\Big(\sum_{j=r}^{s} \alpha_j \mathrm{B}_{jk} \Big) = \sum_{j=r}^{s+1} (k-1) \frac{\alpha_j - \alpha_{j-1}}{t_{j+k-1} - t_j} \mathrm{B}_{j,k-1}, \tag{12'}$$

$$\alpha_{r-1} := 0 =: \alpha_{s+1}.$$

In a way, formula (12) and subsequent formulas are written for biinfinite sums. They apply to finite sums after these have been made formally biinfinite through the adjunction of zero terms.

On the other hand, if we are interested in the fixed interval $[t_r \,..\, t_s]$, then

$$\text{(13)} \qquad \sum_j \alpha_j B_{jk} = \sum_{j=r-k+1}^{s-1} \alpha_j B_{jk} \quad \text{on } [t_r \,..\, t_s],$$

from which we have

$$\text{(14)} \quad D\Big(\sum_j \alpha_j B_{jk}\Big) = \sum_{j=r-k+2}^{s-1} (k-1)\,\frac{\alpha_j - \alpha_{j-1}}{t_{j+k-1} - t_j} B_{j,k-1} \quad \text{on } [t_r \,..\, t_s],$$

since $B_{j,k-1}$ vanishes identically on $[t_r \,..\, t_s]$ for $j \notin [r-k+2 \,..\, s-1]$, that is, we need not bring in additional coefficients as we did in (12′).

Repeated application of (12) produces the following formula for the mth derivative of a spline:

$$\text{(15)} \qquad D^m\Big(\sum_j \alpha_j B_{jk}\Big) = \sum_j \alpha_j^{(m+1)} B_{j,k-m}$$

with

$$\text{(16)} \qquad \alpha_r^{(m+1)} := \begin{cases} \alpha_r, & \text{for } m = 0; \\ \dfrac{\alpha_r^{(m)} - \alpha_{r-1}^{(m)}}{(t_{r+k-m} - t_r)/(k-m)}, & \text{for } m > 0\,. \end{cases}$$

We have taken here the factor $(k-m)$ as a divisor into the denominator in order to stress the fact that we are indeed computing a difference quotient of sorts, with the step $(t_{r+k-m} - t_r)/(k-m)$ an "average mesh length". This implies that

$$\text{(17)} \quad \alpha_r^{(m+1)} = \nabla^m \alpha_r / h^m \quad \textit{in case } \mathbf{t} \textit{ is uniform, that is, } \Delta t_r = h, \textit{ all } r.$$

Here, ∇^m denotes the mth backward difference, that is, $\nabla^m := \nabla(\nabla^{m-1})$ and $\nabla \alpha_r := \alpha_r - \alpha_{r-1}$.

In general, the formula (16) may lead to division by zero. But, if indeed $t_{r+k-m} - t_r = 0$, then $B_{r,k-m} = 0$ by IX(22), and, as we will follow the useful maxim

anything times zero is zero,

it really doesn't matter how we interpret the definition of $\alpha_r^{(m+1)}$ in this case. In computations, one simply refrains from attempting to compute $\alpha_r^{(m+1)}$ in case $t_{r+k-m} - t_r = 0$.

The subroutine BSPLPP We are now in a position to discuss the subroutine BSPLPP for the construction of the ppform for a pp function f from a B-form for f.

```
      SUBROUTINE BSPLPP ( T, BCOEF, N, K, SCRTCH, BREAK, COEF, L )
CALLS  BSPLVB
C
CONVERTS THE B-REPRESENTATION  T, BCOEF, N, K  OF SOME SPLINE INTO ITS
C  PP-REPRESENTATION  BREAK, COEF, L, K .
C
C******  I N P U T  ******
C  T.....KNOT SEQUENCE, OF LENGTH  N+K
C  BCOEF.....B-SPLINE COEFFICIENT SEQUENCE, OF LENGTH  N
C  N.....LENGTH OF  BCOEF  AND  DIMENSION OF SPLINE SPACE  SPLINE(K,T)
C  K.....ORDER OF THE SPLINE
C
C  W A R N I N G  . . .  THE RESTRICTION  K .I.E. KMAX (= 20)  IS IMPO-
C       SED BY THE ARBITRARY DIMENSION STATEMENT FOR  BIATX  BELOW, BUT
C       IS  N O W H E R E  C H E C K E D  FOR.
C
C******  W O R K  A R E A  ******
C  SCRTCH.....OF SIZE  (K,K) , NEEDED TO CONTAIN BCOEFFS OF A PIECE OF
C       THE SPLINE AND ITS  K-1  DERIVATIVES
C
C******  O U T P U T  ******
C  BREAK.....BREAKPOINT SEQUENCE, OF LENGTH  L+1, CONTAINS (IN INCREAS-
C       ING ORDER) THE DISTINCT POINTS IN THE SEQUENCE  T(K),...,T(N+1)
C  COEF.....ARRAY OF SIZE (K,L), WITH  COEF(I,J) = (I-1)ST DERIVATIVE OF
C       SPLINE AT BREAK(J) FROM THE RIGHT
C  L.....NUMBER OF POLYNOMIAL PIECES WHICH MAKE UP THE SPLINE IN THE IN-
C       TERVAL  (T(K), T(N+1))
C
C******  M E T H O D  ******
C     FOR EACH BREAKPOINT INTERVAL, THE  K  RELEVANT B-COEFFS OF THE
C  SPLINE ARE FOUND AND THEN DIFFERENCED REPEATEDLY TO GET THE B-COEFFS
C  OF ALL THE DERIVATIVES OF THE SPLINE ON THAT INTERVAL. THE SPLINE AND
C  ITS FIRST  K-1  DERIVATIVES ARE THEN EVALUATED AT THE LEFT END POINT
C  OF THAT INTERVAL, USING  BSPLVB  REPEATEDLY TO OBTAIN THE VALUES OF
C  ALL B-SPLINES OF THE APPROPRIATE ORDER AT THAT POINT.
C
      INTEGER K,L,N,   I,J,JP1,KMAX,KMJ,LEFT,LSOFAR
      PARAMETER (KMAX = 20)
      REAL BCOEF(N),BREAK,COEF,T,   SCRTCH(K,K)
     *                             ,BIATX(KMAX),DIFF,FACTOR,SUM
      DIMENSION BREAK(L+1),COEF(K,L),T(N+K)
      LSOFAR = 0
      BREAK(1) = T(K)
      DO 50 LEFT=K,N
C                                  FIND THE NEXT NONTRIVIAL KNOT INTERVAL.
         IF (T(LEFT+1) .EQ. T(LEFT))    GO TO 50
         LSOFAR = LSOFAR + 1
         BREAK(LSOFAR+1) = T(LEFT+1)
         IF (K .GT. 1)                  GO TO 9
         COEF(1,LSOFAR) = BCOEF(LEFT)
                                        GO TO 50
C        STORE THE K B-SPLINE COEFF.S RELEVANT TO CURRENT KNOT INTERVAL
C                                  IN  SCRTCH(.,1) .
    9    DO 10 I=1,K
   10       SCRTCH(I,1) = BCOEF(LEFT-K+I)
C
C        FOR J=1,...,K-1, COMPUTE THE  K-J  B-SPLINE COEFF.S RELEVANT TO
C        CURRENT KNOT INTERVAL FOR THE J-TH DERIVATIVE BY DIFFERENCING
C        THOSE FOR THE (J-1)ST DERIVATIVE, AND STORE IN SCRTCH(.,J+1) .
```

```
      DO 20 JP1=2,K
         J = JP1 - 1
         KMJ = K - J
         DO 20 I=1,KMJ
            DIFF = T(LEFT+I) - T(LEFT+I - KMJ)
            IF (DIFF .GT. 0.)  SCRTCH(I,JP1) =
     *                        (SCRTCH(I+1,J)-SCRTCH(I,J))/DIFF
20          CONTINUE
C
C     FOR  J = 0, ..., K-1, FIND THE VALUES AT  T(LEFT)  OF THE  J+1
C     B-SPLINES OF ORDER  J+1  WHOSE SUPPORT CONTAINS THE CURRENT
C     KNOT INTERVAL FROM THOSE OF ORDER  J  (IN  BIATX ), THEN COMB-
C     INE WITH THE B-SPLINE COEFF.S (IN SCRTCH(.,K-J) ) FOUND EARLIER
C     TO COMPUTE THE (K-J-1)ST DERIVATIVE AT  T(LEFT)  OF THE GIVEN
C     SPLINE.
C        NOTE. IF THE REPEATED CALLS TO  BSPLVB  ARE THOUGHT TO GENE-
C     RATE TOO MUCH OVERHEAD, THEN REPLACE THE FIRST CALL BY
C        BIATX(1) = 1.
C     AND THE SUBSEQUENT CALL BY THE STATEMENT
C        J = JP1 - 1
C     FOLLOWED BY A DIRECT COPY OF THE LINES
C        DELTAR(J) = T(LEFT+J) - X
C                  .......
C        BIATX(J+1) = SAVED
C     FROM  BSPLVB . DELTAL(KMAX)  AND  DELTAR(KMAX)  WOULD HAVE TO
C     APPEAR IN A DIMENSION STATEMENT, OF COURSE.
C
      CALL BSPLVB ( T, 1, 1, T(LEFT), LEFT, BIATX )
      COEF(K,LSOFAR) = SCRTCH(1,K)
      DO 30 JP1=2,K
         CALL BSPLVB ( T, JP1, 2, T(LEFT), LEFT, BIATX )
         KMJ = K+1 - JP1
         SUM = 0.
         DO 28 I=1,JP1
28          SUM = BIATX(I)*SCRTCH(I,KMJ) + SUM
30       COEF(KMJ,LSOFAR) = SUM
50    CONTINUE
   L = LSOFAR
   IF (K .EQ. 1)                          RETURN
   FACTOR = 1.
   DO 60 I=2,K
      FACTOR = FACTOR*FLOAT(K+1-I)
      DO 60 J=1,LSOFAR
60       COEF(I,J) = COEF(I,J)*FACTOR
                                          RETURN
   END
```

As already mentioned in Chapter IX, we must find the L + 1 distinct points $\xi_1 < \cdots < \xi_{L+1}$ among the numbers T(K), ..., T(N+1), store them (in increasing order) in BREAK(1), ..., BREAK(L+1) and then compute, for $i = 1, \ldots, L$ and for $j = 1, \ldots, K$, the number $D^{j-1}f(\xi_i^+)$ and store it in COEF(j, i).

Suppose ξ_i = T(LEFT) < T(LEFT + 1) for some $i \leq$ L. Then ξ_{i+1} = T(LEFT + 1), and

$$f = \sum_{r=1}^{k} \alpha_{\mathtt{LEFT}-k+r} \mathrm{B}_{\mathtt{LEFT}-k+r,k} \quad \text{on} \quad [\xi_i^+ \,..\, \xi_{i+1}^-],$$

with agreement 'at' ξ_i^+ and ξ_{i+1}^- (that is, of the one-sided limits) justified

since, on $(\xi_i \,..\, \xi_{i+1})$, both sides are just polynomials. Therefore, by (15),

$$D^j f = \sum_{r=j+1}^{k} \alpha^{(j+1)}_{\text{LEFT}-k+r} \text{B}_{\text{LEFT}-k+r,k-j} \quad \text{on} \quad [\xi_i^+ \,..\, \xi_{i+1}^-]$$

for $j = 0, \ldots, k-1$, with $\alpha_r^{(j)}$ as given by (16). If we put the $k-j$ numbers $\alpha^{(j+1)}_{\text{LEFT}-k+r}$ into the $(j+1)$st column of the two-dimensional array A as follows,

$$\text{A}(r, j+1) \leftarrow \alpha^{(j+1)}_{\text{LEFT}-k+j+r}, \qquad r = 1, \ldots, k-j,$$

then (16) translates into

$$\text{A}(r, 1) \leftarrow \alpha_{\text{LEFT}-k+r}, \qquad r = 1, \ldots, k \tag{18}$$

and

$$\text{A}(r, j+1) \leftarrow (k-j) \frac{\text{A}(r+1, j) - \text{A}(r, j)}{\text{T}(\text{LEFT}+r) - \text{T}(\text{LEFT}+r-k+j)}, \tag{19}$$
$$r = 1, \ldots, k-j; \quad j = 1, \ldots, k-1.$$

Once these coefficients are computed, we then find

$$D^j f(\xi_i^+) = \sum_{r=1}^{k-j} \text{A}(r, j+1) \cdot b_r$$

with $b_r := \text{B}_{\text{LEFT}-k+j+r,k-j}(\xi_i^+)$, $r = 1, \ldots, k-j$, obtained from the subroutine BSPLVB mentioned earlier by a `CALL BSPLVB ( T, k-j, 1, ξi, LEFT, b )`. A final change of variables, in which $k-j-1$ is substituted for j, then leads to the `DO 30` loop.

(20) Example: Computing a B-spline once again This time, we use the subprogram just discussed to recompute the cubic B-spline whose polynomial pieces we obtained in Example (7) with the aid of BSPLVB.

After we obtain the ppform for the spline

$$f := \sum_{i=1}^{7} \delta_{i4} \text{B}_{i4} = \text{B}_{44}$$

with the knot sequence $\mathbf{t} := (0, 0, 0, 0, 1, 2, 3, 4, 6, 6, 6, 6)$ via BSPLPP, we evaluate the B-spline f on a fine mesh with the aid of PPVALU (and INTERV) introduced in Chapter VII.

```
CHAPTER X. EXAMPLE 3. CONSTRUCTION AND EVALUATION OF THE PP-REPRESENTAT-
C                     ION OF A B-SPLINE.
CALLS BSPLPP(BSPLVB),PPVALU(INTERV)
C
      INTEGER IA,L
      REAL BCOEF(7),BREAK(5),COEF(4,4),SCRTCH(4,4),T(11),VALUE,X
C               SET KNOT SEQUENCE  T  AND B-COEFFS FOR  B(4,4,T)  ....
      DATA T / 4*0.,1.,3.,4.,4*6. /, BCOEF / 3*0.,1.,3*0. /
C                           CONSTRUCT PP-REPRESENTATION  ....
      CALL BSPLPP ( T, BCOEF, 7, 4, SCRTCH, BREAK, COEF, L )
C
C     AS A CHECK, EVALUATE  B(4,4,T)  FROM ITS PP-REPR. ON A FINE MESH.
C           THE VALUES SHOULD AGREE WITH (SOME OF) THOSE GENERATED IN
C                       EXAMPLE 2 .
      DO 20 IA=1,40
         X = FLOAT(IA)/5. - 1.
         VALUE = PPVALU ( BREAK, COEF, L, 4, X, 0 )
   20    PRINT 620, X, VALUE
  620 FORMAT(F10.1,F20.8)
                                          STOP
      END
```

□

The subroutine BVALUE The construction used in BSPLPP can be exploited in a slightly more general setup to provide the value of the first so many derivatives of a spline from its B-form. This is done, for example, in the subprogram BSPLEV of de Boor [1971] and de Boor [1977]$_1$, but we do not consider it here. Instead, we now discuss the subprogram BVALUE which is analogous to PPVALU in that it provides the value of the jth derivative of a spline at some site, but from its B-form rather than its ppform.

```
      REAL FUNCTION BVALUE ( T, BCOEF, N, K, X, JDERIV )
CALLS  INTERV
C
CALCULATES VALUE AT  X  OF  JDERIV-TH DERIVATIVE OF SPLINE FROM B-REPR.
C  THE SPLINE IS TAKEN TO BE CONTINUOUS FROM THE RIGHT, EXCEPT AT THE
C  RIGHTMOST KNOT, WHERE IT IS TAKEN TO BE CONTINUOUS FROM THE LEFT.
C
C******  I N P U T ******
C  T, BCOEF, N, K......FORMS THE B-REPRESENTATION OF THE SPLINE  F  TO
C        BE EVALUATED. SPECIFICALLY,
C  T.....KNOT SEQUENCE, OF LENGTH  N+K, ASSUMED NONDECREASING.
C  BCOEF.....B-COEFFICIENT SEQUENCE, OF LENGTH  N .
C  N.....LENGTH OF  BCOEF  AND DIMENSION OF SPLINE(K,T),
C        A S S U M E D  POSITIVE .
C  K.....ORDER OF THE SPLINE .
C
C  W A R N I N G . . .  THE RESTRICTION  K .LE. KMAX (=20)  IS IMPOSED
C        ARBITRARILY BY THE DIMENSION STATEMENT FOR  AJ, DL, DR  BELOW,
C        BUT IS  N O W H E R E  C H E C K E D  FOR.
C
```

```
C  X.....THE POINT AT WHICH TO EVALUATE .
C  JDERIV.....INTEGER GIVING THE ORDER OF THE DERIVATIVE TO BE EVALUATED
C        A S S U M E D  TO BE ZERO OR POSITIVE.
C
C******  O U T P U T  ******
C  BVALUE.....THE VALUE OF THE (JDERIV)-TH DERIVATIVE OF  F  AT  X .
C
C******  M E T H O D  ******
C     THE NONTRIVIAL KNOT INTERVAL  [T(I)..T(I+1))  CONTAINING  X  IS LO-
C  CATED WITH THE AID OF  INTERV . THE  K  B-COEFFS OF  F  RELEVANT FOR
C  THIS INTERVAL ARE THEN OBTAINED FROM  BCOEF (OR TAKEN TO BE ZERO IF
C  NOT EXPLICITLY AVAILABLE) AND ARE THEN DIFFERENCED  JDERIV  TIMES TO
C  OBTAIN THE B-COEFFS OF  (D**JDERIV)F  RELEVANT FOR THAT INTERVAL.
C  PRECISELY, WITH  J = JDERIV, WE HAVE FROM X.(15-16) OF THE TEXT THAT
C
C     (D**J)F  =  SUM ( BCOEF(.,J)*B(.,K-J,T) )
C
C  WHERE
C                    / BCOEF(.),                                  ,  J .EQ. 0
C                    /
C     BCOEF(.,J)  =  / BCOEF(.,J-1) - BCOEF(.-1,J-1)
C                    / ----------------------------               ,  J .GT. 0
C                    /    (T(.+K-J) - T(.))/(K-J)
C
C     THEN, WE USE REPEATEDLY THE FACT THAT
C
C     SUM ( A(.)*B(.,M,T)(X) )  =  SUM ( A(.,X)*B(.,M-1,T)(X) )
C  WITH
C                    (X - T(.))*A(.) + (T(.+M-1) - X)*A(.-1)
C     A(.,X)  =      ---------------------------------------
C                    (X - T(.))      + (T(.+M-1) - X)
C
C  TO WRITE  (D**J)F(X)  EVENTUALLY AS A LINEAR COMBINATION OF B-SPLINES
C  OF ORDER  1 , AND THE COEFFICIENT FOR  B(I,1,T)(X)  MUST THEN BE THE
C  DESIRED NUMBER  (D**J)F(X). (SEE X.(25-29) OF TEXT).
C
      INTEGER JDERIV,K,N,    I,ILO,IMK,J,JC,JCMIN,JCMAX,JJ,KMAX,KMJ,KM1
     *                      ,MFLAG,NMI,JDRVP1
      PARAMETER (KMAX = 20)
      REAL BCOEF(N),T(N+K),X,    AJ(KMAX),DL(KMAX),DR(KMAX),FKMJ
      BVALUE = 0.
      IF (JDERIV .GE. K)                          GO TO 99
C
C  *** FIND  I   S.T.   1 .LE. I .LT. N+K   AND   T(I) .LT. T(I+1)   AND
C      T(I) .LE. X .LT. T(I+1) . IF NO SUCH I CAN BE FOUND,  X  LIES
C      OUTSIDE THE SUPPORT OF  THE SPLINE  F , HENCE  BVALUE = 0.
C      (THE ASYMMETRY IN THIS CHOICE OF  I  MAKES  F  RIGHTCONTINUOUS, EXCEPT
C      AT  T(N+K) WHERE IT IS LEFTCONTINUOUS.)
      CALL INTERV ( T, N+K, X, I, MFLAG )
      IF (MFLAG .NE. 0)                           GO TO 99
C  *** IF K = 1 (AND JDERIV = 0), BVALUE = BCOEF(I).
      KM1 = K - 1
      IF (KM1 .GT. 0)                             GO TO 1
      BVALUE = BCOEF(I)                           GO TO 99
C
C  *** STORE THE K B-SPLINE COEFFICIENTS RELEVANT FOR THE KNOT INTERVAL
C      (T(I)..T(I+1)) IN AJ(1),...,AJ(K) AND COMPUTE DL(J) = X - T(I+1-J),
C      DR(J) = T(I+J) - X, J=1,...,K-1 . SET ANY OF THE AJ NOT OBTAINABLE
C      FROM INPUT TO ZERO. SET ANY T.S NOT OBTAINABLE EQUAL TO T(1) OR
C      TO T(N+K) APPROPRIATELY.
```

```
    1 JCMIN = 1
      IMK = I - K
      IF (IMK .GE. 0)                    GO TO 8
      JCMIN = 1 - IMK
      DO 5 J=1,I
    5    DL(J) = X - T(I+1-J)
      DO 6 J=I,KM1
         AJ(K-J) = 0.
    6    DL(J) = DL(I)
                                         GO TO 10
    8 DO 9 J=1,KM1
    9    DL(J) = X - T(I+1-J)
C
   10 JCMAX = K
      NMI = N - I
      IF (NMI .GE. 0)                    GO TO 18
      JCMAX = K + NMI
      DO 15 J=1,JCMAX
   15    DR(J) = T(I+J) - X
      DO 16 J=JCMAX,KM1
         AJ(J+1) = 0.
   16    DR(J) = DR(JCMAX)
                                         GO TO 20
   18 DO 19 J=1,KM1
   19    DR(J) = T(I+J) - X
C
   20 DO 21 JC=JCMIN,JCMAX
   21    AJ(JC) = BCOEF(IMK + JC)
C
C                *** DIFFERENCE THE COEFFICIENTS  JDERIV  TIMES.
      IF (JDERIV .EQ. 0)                 GO TO 30
      DO 23 J=1,JDERIV
         KMJ = K-J
         FKMJ = FLOAT(KMJ)
         ILO = KMJ
         DO 23 JJ=1,KMJ
            AJ(JJ) = ((AJ(JJ+1) - AJ(JJ))/(DL(ILO) + DR(JJ)))*FKMJ
   23       ILO = ILO - 1
C
C  *** COMPUTE VALUE AT  X  IN (T(I),T(I+1)) OF JDERIV-TH DERIVATIVE,
C     GIVEN ITS RELEVANT B-SPLINE COEFFS IN AJ(1),...,AJ(K-JDERIV).
   30 IF (JDERIV .EQ. KM1)               GO TO 39
      JDRVP1 = JDERIV + 1
      DO 33 J=JDRVP1,KM1
         KMJ = K-J
         ILO = KMJ
         DO 33 JJ=1,KMJ
            AJ(JJ) = (AJ(JJ+1)*DL(ILO) + AJ(JJ)*DR(JJ))/(DL(ILO)+DR(JJ))
   33       ILO = ILO - 1
   39 BVALUE = AJ(1)
C
   99                                    RETURN
      END
```

The calculation is based on the formula (15) for the derivatives of a spline and on the recurrence relation IX(14).

We locate i so that $t_i \le \mathtt{X} < t_{i+1}$ and then compute the K-JDERIV relevant B-spline coefficients for the JDERIV-th derivative of f according to (18) and (19). This means that we initialize

$$\mathtt{A}(r,1) \leftarrow \mathtt{BCOEF}(i - \mathtt{K} + r), \qquad r = 1, \ldots, \mathtt{K}, \tag{21}$$

and then compute

$$(22)\qquad \mathtt{A}(r,j+1) \leftarrow (\mathtt{K}-j)\,\frac{\mathtt{A}(r+1,j)-\mathtt{A}(r,j)}{\delta^L_{\mathtt{K}-j+1-r}+\delta^R_r}, \qquad \begin{matrix} r=1,\dots,\mathtt{K}-j; \\ j=1,\dots,\mathtt{JDERIV}. \end{matrix}$$

Here, we have used the abbreviations

$$\delta^L_s := \mathtt{X}-\mathtt{T}(i+1-s), \quad \delta^R_s := \mathtt{T}(i+s)-\mathtt{X}, \quad s=1,\dots,\mathtt{K}-1$$

introduced earlier in the discussion of BSPLVB.

Now we know that, with $m := \mathtt{JDERIV}$, $x := \mathtt{X}$, and $k := \mathtt{K}$,

$$(23)\qquad D^m f(x) = \sum_{r=1}^{k-m} \mathtt{A}(r,m+1)\mathrm{B}_{i-(k-m)+r,k-m}(x).$$

We could therefore finish the calculations with an appeal to the subprogram BSPLVB for the values of the requisite B-splines. It is more efficient, though, to make use of the first algorithm in de Boor [1972], cf which the derivation of IX(19) is a particular example and which is as follows.

Consider first the case $m = \mathtt{JDERIV} = 0$, that is, we want to evaluate the spline f itself. From the recurrence relation IX(14), we obtain that

$$\begin{aligned} f(x) &= \sum_r \alpha_r \mathrm{B}_{rk}(x) \\ &= \sum_r \alpha_r \frac{x-t_r}{t_{r+k-1}-t_r}\mathrm{B}_{r,k-1}(x) \;+\; \sum_r \alpha_r \frac{t_{r+k}-x}{t_{r+k}-t_{r+1}}\mathrm{B}_{r+1,k-1}(x). \end{aligned}$$

Now change the dummy variable of the summation in the second sum from r to $r-1$ and recombine the two sums to get that

$$f(x) = \sum_r \alpha_r^{[2]}(x)\mathrm{B}_{r,k-1}(x)$$

with

$$(24)\qquad \alpha_r^{[2]}(x) := \frac{(x-t_r)\alpha_r + (t_{r+k-1}-x)\alpha_{r-1}}{t_{r+k-1}-t_r}.$$

If we repeat this process again and again, then we obtain the fact that

$$(25)\qquad f(x) = \sum_r \alpha_r^{[j+1]}(x)\mathrm{B}_{r,k-j}(x), \qquad j=0,\dots,k-1$$

with

$$(26)\qquad \alpha_r^{[j+1]}(x) := \begin{cases} \dfrac{(x-t_r)\alpha_r^{[j]}(x) + (t_{r+k-j}-x)\alpha_{r-1}^{[j]}(x)}{t_{r+k-j}-t_r}, & j>0; \\ \alpha_r, & j=0. \end{cases}$$

This means that we have rewritten the spline f of order k as a linear combination of B-splines of order $k-j$, but with coefficients which are not constant anymore but are polynomials of order $j+1$ in the independent variable x.

Now take $j = k-1$ in (25) and take $t_i \le x < t_{i+1}$. Then

$$\mathrm{B}_{r,k-j}(x) = \mathrm{B}_{r1}(x) = \begin{cases} 1, & r = i; \\ 0, & r \neq i. \end{cases}$$

Therefore

$$f(x) = \alpha_i^{[k]}(x) \qquad \text{for } t_i \le x < t_{i+1}.$$

This makes it possible to evaluate our spline f at the site x in $[t_i \,..\, t_{i+1})$ by constructing successively the entries in the following table.

$$\begin{array}{lllll}
\alpha_{i-k+1} =: \alpha_{i-k+1}^{[1]}(x) & & & & \\
 & \alpha_{i-k+2}^{[2]}(x) & & & \\
\alpha_{i-k+2} =: \alpha_{i-k+2}^{[1]}(x) & & & & \\
 & \alpha_{i-k+3}^{[2]}(x) & & \alpha_{i-1}^{[k-1]}(x) & \\
 & & & & \alpha_i^{[k]}(x) \\
 & & & \alpha_i^{[k-1]}(x) & \\
\alpha_{i-1} =: \alpha_{i-1}^{[1]}(x) & & & & \\
 & \alpha_i^{[2]}(x) & & & \\
\alpha_i =: \alpha_i^{[1]}(x) & & & &
\end{array}$$

(27) FIGURE. Table of polynomials $\alpha_r^{[j]}$ of order j, constructible column by column by (26) and producing eventually the polynomial $\alpha_i^{[k]}$ of order k that agrees with the spline $\sum_s \alpha_s \mathrm{B}_{sk}$ on the open interval $(t_i \,..\, t_{i+1})$.

The leftmost column consists of the B-spline coefficients for f pertinent to the interval $[t_i \,..\, t_{i+1}]$. Every entry in a subsequent column is obtainable, according to (26), as a *convex* combination of its two neighbors to the left if, as we assume, $t_i \le x \le t_{i+1}$.

Suppose we put the $k-j+1$ numbers $\alpha_{i-k+j}^{[j]}, \ldots, \alpha_i^{[j]}$ into the jth column of the two-dimensional array A,

$$\mathtt{A}(r,j) \leftarrow \alpha_{i-k+j-1+r}^{[j]}, \qquad r = 1, \ldots, k-j+1.$$

Then (26) can be written

$$\mathtt{A}(r,j+1) \leftarrow \frac{\delta_{k-j+1-r}^{L}\mathtt{A}(r+1,j) + \delta_r^R \mathtt{A}(r,j)}{\delta_{k-j+1-r}^{L} + \delta_r^R}, \tag{28}$$

$$r = 1, \ldots, k-j;\ j = 1, \ldots, k-1,$$

where we, once again, use the abbreviations

$$\delta_r^L := x - t_{i+1-r}, \quad \delta_r^R := t_{i+r} - x, \qquad r = 1, \ldots, k-1.$$

We have shown that, if we initialize

$$A(r,1) \leftarrow \alpha_{i-k+r}, \qquad r = 1, \ldots, k$$

and then compute the remaining entries in the triangular array A according to (28), then

$$\mathtt{A}(1,k) = f(x) = \sum_r \alpha_r \mathrm{B}_{rk}(x)$$

for $t_i \le x < t_{i+1}$. Consequently, if we compute the $k-m$ numbers $\mathtt{A}(r, m+1)$, $r = 1, \ldots, k-m$, by (22), and then compute, according to (28),

$$\mathtt{A}(r, j+1) \leftarrow \frac{\delta^L_{k-j+1-r}\mathtt{A}(r+1,j) + \delta^R_r \mathtt{A}(r,j)}{\delta^L_{k-j+1-r} + \delta^R_r}, \qquad r = 1, \ldots, k-j;\ j = m+1, \ldots, k-1, \tag{29}$$

then

$$\mathtt{A}(1,k) = D^m f(x) \qquad \text{for } t_i \le x < t_{i+1}.$$

The subprogram BVALUE employs the two steps (22) and (29). The first part of the program takes care of the possibility that there might not be k B-spline coefficients for the interval $[t_i, .t_{i+1}]$. This happens when $1 \le i < k$ and/or $n < i \le n+k$. In this case, we could suitably restrict the indices appearing in (22) and (29). But, since this case is likely to be exceptional, it seems inefficient to burden the calculations (22) and (29) in this way with additional index calculations. Rather, we simply choose the "missing" coefficients to be zero and then choose the correspondingly missing knots (whose choice is then entirely immaterial except for the necessity of avoiding division by zero) to equal the first or the last given knot as the case may be.

One would expect to use BVALUE if a spline is to be evaluated no more than about two or three times per polynomial piece (see problem X.3). Otherwise, it would be more efficient to convert first to ppform via BSLPP and then use PPVALU for the evaluation.

(30) Example: Computing a B-spline one more time We compute once more the values of the B-spline of order 4 with knots (0, 1, 3, 4, 6) on a fine mesh, but use BVALUE this time. Note that, in contrast to Examples (7) and (20), we do not need to invent (explicitly) additional knots.

```
CHAPTER X.  EXAMPLE 4.  CONSTRUCTION OF A B-SPLINE VIA  BVALUE
CALLS  BVALUE(INTERV)
      INTEGER IA
      REAL BCOEF(1),T(5),VALUE,X
C                    SET KNOT SEQUENCE  T  AND  B-COEFFS FOR  B(1,4,T)
      DATA T / 0.,1.,3.,4.,6. / , BCOEF / 1. /
C       EVALUATE  B(1,4,T)  ON A FINE MESH.  ON (0,6), THE VALUES SHOULD
C              COINCIDE WITH THOSE OBTAINED IN EXAMPLE 3 .
      DO 20 IA=1,40
         X = FLOAT(IA)*.2 - 1.
         VALUE = BVALUE ( T, BCOEF, 1, 4, X, 0 )
   20    PRINT 620, X, VALUE
  620 FORMAT(F10.1,F20.8)
                                          STOP
      END
```

□

Integration We know from formula (12) for the derivative of a spline that

$$D\Big(\sum_i \beta_i \mathrm{B}_{i,k+1}\Big) = \sum_i k\,\frac{\beta_i - \beta_{i-1}}{t_{i+k} - t_i}\,\mathrm{B}_{ik}.$$

This implies that $\sum_i \alpha_i \mathrm{B}_{ik}$ is the first derivative of the spline $\sum_i \beta_i \mathrm{B}_{i,k+1}$ provided $k(\beta_i - \beta_{i-1})/(t_{i+k} - t_i) = \alpha_i$, all i, or

$$\beta_i = \beta_{i-1} + \alpha_i(t_{i+k} - t_i)/k, \qquad \text{all } i. \tag{31}$$

This allows us to prescribe one B-spline coefficient in the anti-derivative for a spline arbitrarily, and all other coefficients are then determined by (31). In formulae, the most general anti-derivative for the spline $\sum_i \alpha_i \mathrm{B}_{ik}$ is given by $\sum_i \beta_i \mathrm{B}_{i,k+1}$ with

$$\beta_i = c + \begin{cases} \sum_{i_0}^{i} \alpha_j(t_{j+k} - t_j)/k, & i \geq i_0; \\ -\sum_{i+1}^{i_0-1} \alpha_j(t_{j+k} - t_j)/k, & i < i_0, \end{cases} \tag{32}$$

and c and i_0 arbitrary.

Here, we have again taken the tack that even if our spline $\sum_i \alpha_i \mathrm{B}_{ik}$ has only finitely many knots, we nevertheless treat it as a spline with infinitely many knots by adjoining suitable zero terms. This is quite necessary because a spline with finitely many knots usually has *no* anti-derivative with finitely many knots. Indeed, for a specific spline $f = \sum_1^n \alpha_i \mathrm{B}_{ik}$, we manage to construct an antiderivative $\sum_i \beta_i \mathrm{B}_{i,k+1}$ with $\beta_i = 0$ for all $i < 1$ by choosing $c = 0$, $i_0 = 1$ in (32). But then

$$\beta_i = \sum_{j=1}^{n} \alpha_j(t_{j+k} - t_j)/k \qquad \text{for } i = n, n+1, \ldots .$$

It follows that the spline $\sum_1^n \alpha_i B_{ik}$ is the derivative of a spline with finitely many knots if and only if $\sum_1^n \alpha_j(t_{j+k} - t_j) = 0$.

Of course, these infinitely many coefficients need not worry us if we are only interested in the integral over an interval containing a finite number of knots. We find that

$$\int_{t_1}^{x} \sum_{i=1}^{n} \alpha_i B_{ik}(y)\,dy = \sum_{i=1}^{s-1} \left(\sum_{j=1}^{i} \alpha_j(t_{j+k} - t_j)/k \right) B_{i,k+1}(x) \tag{33}$$

on $t_1 \le x \le t_s$.

Problems

1. *Partial B-form.* It is easy to pick out of the B-form for a spline f in `T, BCOEF, N, K` a partial representation.
(a) Verify that the input list `T(I-K+1), BCOEF(I-K+1), K, K` (for BSPLPP) describes the B-form for $f|_{(T(I)..T(I+1))}$.
(b) What part of `T, BCOEF, N, K` would give the B-form for $f|_{(T(I)..T(J))}$?
(c) What if `I < K` in (a) or (b)? Could an appropriate representation usable in BVALUE still be picked out?

2. Use BSPLPP to calculate the value and first $k-1$ derivatives of a spline at just *one* knot `T(i)`. (Don't cause any more computations than absolutely necessary).

3. Give an operation count (floating-point operations) as a function of k for the evaluation of a spline of order k at one site
(a) by BVALUE;
(b) by PPVALU.
(c) Give such a count as a function of k and l for the cost of converting from B-form to ppform by BSPLPP.
(d) For how many interpolation sites (per interval) is it cheaper to convert and use PPVALU than to use BVALUE?

4. Take a spline in B-form (for example, a B-spline), convert to ppform by BSPLPP, then use IX(53) and Theorem IX(44) to construct a B-form for the resulting pp function and compare with the B-form you started with.

5. Use Leibniz' formula I(iv) to establish a recurrence relation that relates the jth derivative of a B-spline to that of two B-splines of one order less.

6. Change BVALUE into a routine BVALLC that treats splines as left-continuous functions (cf. Problem VII.2).

7. Prove: If z occurs exactly m times in the sequence $t_{i+1}, \ldots, t_{i+k-1}$, and $r := k - 1 - m$, then

$$\frac{z - t_i}{t_{i+k-1} - t_i} \operatorname{jump}_z D^r B_{i,k-1} + \frac{t_{i+k} - z}{t_{i+k} - t_{i+1}} \operatorname{jump}_z D^r B_{i+1,k-1} = 0.$$

8. (a) Prove that $f := \sum_1^n \alpha_i B_{i,k,\mathbf{t}}$ is the first derivative of a spline (with finite support) if and only if $\sum_1^n \alpha_i(t_{i+k} - t_i) = 0$.
(b) Prove that f is also a second derivative of a spline (with finite support) if and only if in addition $\sum_{i=1}^{n} \alpha_i(t_{i+k} - t_i) \sum_{j=i}^{n-1} (t_{j+k-1} - t_j) = 0$.
(c) If you have the courage, state and prove the r conditions on (α_i) necessary and sufficient for f to be the rth derivative of a spline of finite support.

9. Use (15) and (16) to show that, for $f = \sum_i \alpha_i B_{i,k,\mathbf{t}}$ with $t_{r-k+1} = \cdots = t_r < t_{r+1} = \cdots = t_{r+s}$,

$$(D^j f)(t_r) = \frac{(k-1)\cdots(k-j)}{(\Delta t_r)^j} \Delta^j \alpha_{r-k+1}, \quad \text{for } j = 0, \ldots, s.$$

This shows that the choice $t_1 = \cdots = t_k = a$, and $t_{n+1} = \cdots = t_{n+k} = b$, of the end knots as proposed in Chapter IX makes the imposition of end conditions on f and its derivatives particularly easy.

10. Develop an alternate routine ESPLPP that makes no use of BSPLVB but rather uses algorithm (25) - (26) to derive the local polynomial coefficients of the polynomial piece of $f = \sum_i \alpha_i B_{i,k}$ on $[t_m \,..\, t_{m+1}]$ from $\alpha_{m-k+1}, \ldots, \alpha_m$. (Hint: Express each of the polynomials $\alpha_r^{[j]}$ in local polynomial form.)

How could this idea be used to derive a routine that evaluates simultaneously f and its first s derivatives at some site z in $[t_m \,..\, t_{m+1}]$? (Hint: $\alpha_m^{[k]}(x) = \sum_{i=1}^{k} f^{(i-1)}(z)(x-z)^{i-1}/(i-1)!$.)

XI

The B-Spline Series, Control Points, and Knot Insertion

In Example IX(58), we made the point that *B-spline coefficients model the function they represent.* In the present chapter, we document this assertion in more detail by continuing the list of properties of B-splines begun in Chapter IX, paying special attention to the *control points* of a spline and introducing *knot insertion* as a remarkably efficient tool for establishing the shape-preserving properties of a spline fit and the *total positivity* of the B-spline collocation matrix that underlies them.

Throughout this chapter, let $\mathbf{t} := (t_i)_1^{n+k}$ with $t_i < t_{i+k}$, all i, and $t_1 = \cdots = t_k = a$, $t_{n+1} = \cdots = t_{n+k} = b$ so that, in particular

$$[a\,..\,b] = [t_k\,..\,t_{n+1}]$$

is the basic interval; also, let $\mathrm{B}_i = \mathrm{B}_{i,k,\mathbf{t}}$, $i = 1, \ldots, n$, and let $\$ = \$_{k,\mathbf{t}}$.

Bounding spline values in terms of "nearby" coefficients Recall from B-spline Property IX(iv) (see IX(36)) that the B-spline sequence $(\mathrm{B}_i)_1^n$ provides a "local" and nonnegative partition of unity on $[a\,..\,b]$. This gives the following property (to add to the list begun in Chapter IX):

(1) B-spline Property (ix): Convex hull. *For $t_i < x < t_{i+1}$, the value of the spline function $f := \sum_j \alpha_j \mathrm{B}_j$ at the site x is a strictly convex combination of the k numbers $\alpha_{i+1-k}, \ldots, \alpha_i$.*

Indeed, if $t_i \le x \le t_{i+1}$, then $f(x) = \sum_{j=i-k+1}^{i} \alpha_j \mathrm{B}_j(x)$, while all $\mathrm{B}_j(x)$ are nonnegative and sum to 1.

(2) Corollary. *If $t_i \le x \le t_{i+1}$ and $f = \sum_j \alpha_j \mathrm{B}_j$, then*

$$\text{(3)} \qquad \min\{\,\alpha_{i+1-k}, \ldots, \alpha_i\,\} \le f(x) \le \max\{\,\alpha_{i+1-k}, \ldots, \alpha_i\,\}.$$

In words, the values of the spline $f = \sum \alpha_j \mathrm{B}_j$ on the interval $[t_i\,..\,t_{i+1}]$ are bounded, above and below, by the k B-spline coefficients "nearby" (see

Problem 1). On the other hand, any B-spline coefficient is closely related to the values of f "nearby", as the following property shows. We use the abbreviation

$$\|f\|_I := \sup_{x\in I} |f(x)|.$$

(4) B-spline Property (x): Good condition. $(\mathrm{B}_i)_1^n$ *is a relatively well conditioned basis for* \$ *in the sense that there exists a positive constant* $D_{k,\infty}$, *which depends only on* k *and not on the particular knot sequence* $\mathbf{t}$, *so that for all* i,

$$|\alpha_i| \le D_{k,\infty} \|\sum_j \alpha_j \mathrm{B}_j\|_{[t_{i+1}..t_{i+k-1}]}. \tag{5}$$

Smallest possible values for $D_{k,\infty}$ are

k	2	3	4	5	6
$D_{k,\infty}$	1	3	$5.5680\cdots$	$12.0886\cdots$	$22.7869\cdots$

Based on numerical calculations, it is conjectured that, in general,

$$D_{k,\infty} \sim 2^{k-3/2}.$$

PROOF SKETCH. Choose τ_i in the dual functional formula IX(54),

$$\lambda_{ik} f := \sum_{\nu=1}^{k} \frac{(-D)^{k-\nu}\psi_{ik}(\tau_i)}{(k-1)!}\, D^{\nu-1} f(\tau_i),$$

for the ith B-spline coefficient α_i of $f \in \$_{k,\mathbf{t}}$ to lie in a largest knot interval $[t_\tau \,..\, t_{\tau+1}]$ in $[t_{i+1} \,..\, t_{i+k-1}]$. Then $|(-D)^{k-\nu}\psi_{ik}(\tau_i)| \le \mathrm{const}|\Delta t_\tau|^{\nu-1}$ (since $D^{k-\nu}\psi_{ik}(\tau_i)$ is a certain sum of products of $\nu-1$ factors of the form $t_s - \tau_i$ with $s \in \{i+1,\ldots,i+k-1\}$), while, by Markov's Inequality (see Rivlin [1969]), for τ_i in any knot interval $[t_\tau \,..\, t_{\tau+1}]$ in $[t_i \,..\, t_{i+k}]$,

$$|D^{\nu-1} f(\tau_i)| \le \mathrm{const}\|f\|_{[t_\tau..t_{\tau+1}]}/|\Delta t_\tau|^{\nu-1},$$

both inequalities for certain $\mathbf{t}$- and i-independent constants. This shows that $|\alpha_i| \le \mathrm{const}\|f\|_{[t_\tau..t_{\tau+1}]} \le \mathrm{const}\|f\|_{[t_{i+1}..t_{i+k-1}]}$ for a certain constant independent of i and $\mathbf{t}$. □

The existence of such a constant $D_{k,\infty}$ goes back to de Boor [1968], but the proof there is unnecessarily involved. A different proof (along the line of the above proof sketch) can be found in de Boor $[1976]_2$. The specific numerical values for $D_{k,\infty}$ given above are computed with the aid of de Boor $[1990]_1$. The best bound so far is obtained in Scherer & Shadrin [1999]:

$$D_{k,\infty} \le k2^{k-1}.$$

Let $f := \sum_j \alpha_j \mathrm{B}_j$. Since B-splines sum up to one, (5) implies that $|\alpha_i - c| \le D_{k,\infty}\|f - c\|_{[t_{i+1}..t_{i+k-1}]}$ for any particular constant c. For the particular constant $c := (M+m)/2$ with $\overset{M}{m} := \overset{\max}{\min}\{\, f(x) : t_{i+1} \le x \le t_{i+k-1}\,\}$, we get $\|f - c\|_{[t_{i+1}..t_{i+k-1}]} = (M-m)/2$. This give the following converse to Corollary (2).

(6) Corollary. *Let* $f = \sum_j \alpha_j B_j$ *and* $f([t_{i+1} \,..\, t_{i+k-1}]) = [m \,..\, M]$. *Then*

$$(7) \qquad \left| \alpha_i - \frac{M+m}{2} \right| \le D_{k,\infty} \frac{M-m}{2}.$$

Finally, the assertion that (B_j) is a relatively well conditioned basis for \$ is properly expressed by the following corollary which combines (5) and (3).

(8) Corollary. $D_{k,\infty}^{-1} \|\alpha\| \le \| \sum \alpha_j B_j \| \le \|\alpha\| := \max_j |\alpha_j|$.

The point to note here once again is that $D_{k,\infty}$ does not depend on **t**.

Control points and control polygon The close relationship between the value of a spline and the "nearby" B-spline coefficients has led to the definition and use of "control point". This term had its origin in Computer-Aided Geometric Design, where spline *curves* rather than spline *functions* are used. Here, a **spline curve** is, by definition, a **vector-valued** spline, that is, a spline with values in the plane or in 3-space. Correspondingly, its B-spline coefficients are then points in the plane or in 3-space, and their sequence is called the **control point sequence** of that spline curve.

Now, the graph of a spline *function* (or, *scalar-valued* spline) $f \in \$$ defines a curve, namely the planar curve

$$x \mapsto (x, f(x)),$$

and this is actually a *spline* curve, that is, a spline with *vector*-valued coefficients, since, by B-spline Property IX(v),

$$x = \sum_j t_{jk}^* B_j(x), \quad x \in [a \,..\, b],$$

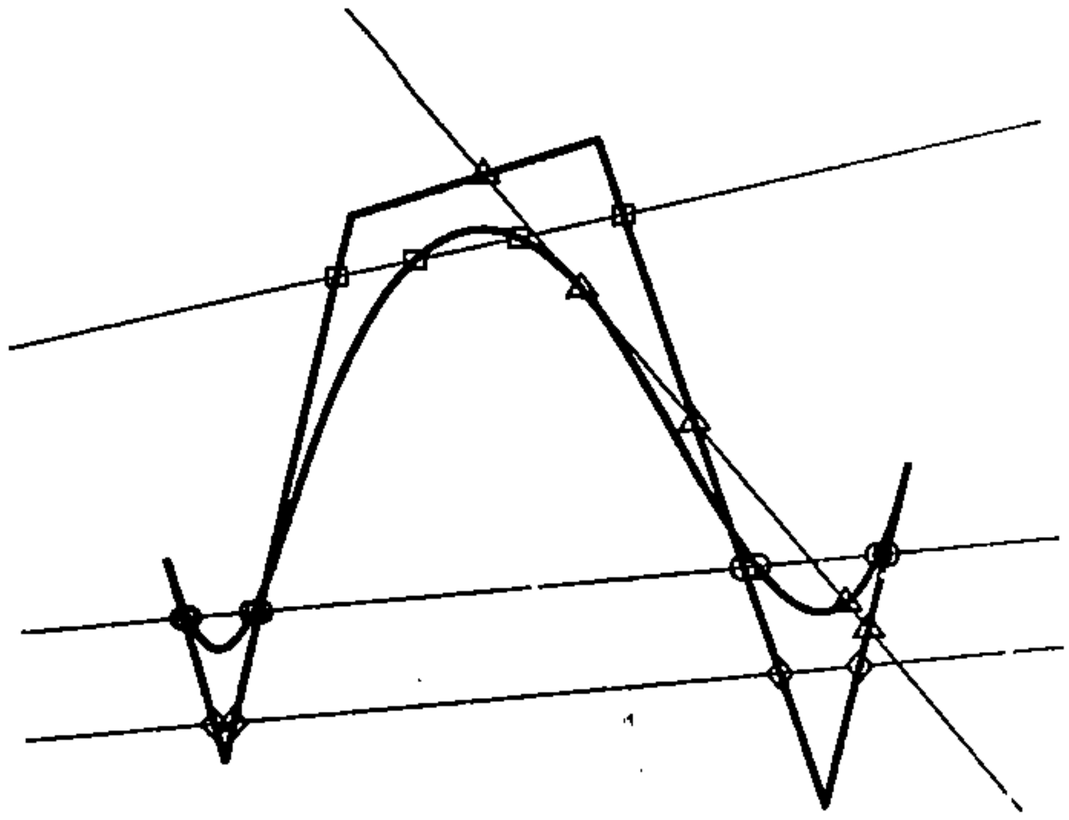
where (see IX(39))

$$t_{jk}^* = \frac{t_{j+1} + \cdots + t_{j+k-1}}{k-1}, \quad \text{all } j.$$

For this reason, one calls

$$(9) \qquad (P_j := (t_{jk}^*, \alpha_j) \in \mathbb{R}^2 \,:\, j = 1, \ldots, n)$$

the **control point sequence** of the spline function $\sum_j \alpha_j B_j \in \$_{k,\mathbf{t}}$.



(10) FIGURE. A control polygon exaggerates the corresponding spline. Any crossing of the spline by some straight line is bracketed by crossings of the control polygon by that straight line.

Finally, the **control polygon**

$$C_{k,\mathbf{t}} f \tag{11}$$

of a spline $f \in \$$ is the broken line with the spline's control point sequence as vertex sequence. The control polygon is an exaggerated version or caricature of the spline itself, hence makes it easy to locate certain important features (zeros, regions of convexity/concavity, etc) of the spline; see, e.g., Figure (10).

The close connection between a spline and its control polygon is also evidenced by the following proposition.

(12) Proposition. *If the spline $f \in \$$ is continuously differentiable, then*

$$|\alpha_j - f(t^*_{jk})| \le \text{const}_k |\mathbf{t}|^2 \|D^2 f\|_{[t_{j+1}..t_{j+k-1}]}. \tag{13}$$

PROOF. We know from (5) that

$$|\alpha_j| = |\lambda_{jk} f| \le D_{k\infty} \|f\|_{[t_{j+1}..t_{j+k-1}]}.$$

Further, recall from the knot-averages B-spline property (see IX(38)) that

$$\lambda_{jk} p = p(t^*_{jk}), \quad \text{all } p \in \Pi_{<2}.$$

Choosing, in particular,

$$p := f(t^*_{jk}) + (\cdot - t^*_{jk}) Df(t^*_{jk}),$$

that is, the linear Taylor polynomial for f at t^*_{jk}, we get

$$\begin{aligned} |\alpha_j - f(t^*_{jk})| = |\lambda_{jk}(f-p)| &\le \text{const} \|f - p\|_{[t_{j+1}..t_{j+k-1}]} \\ &\le \text{const}\, \frac{1}{2} (t_{j+k-1} - t_{j+1})^2 \, \|D^2 f\|_{[t_{j+1}..t_{j+k-1}]}. \end{aligned} \tag{14}$$

□

(15) Corollary. *For any continuously differentiable $f \in \$$,*

$$\|f - C_{k,\mathbf{t}} f\| \;\leq\; \text{const}|\mathbf{t}|^2 \|D^2 f\|.$$

PROOF. Let $t^*_{jk} \leq x \leq t^*_{j+1,k}$ and let p be the linear polynomial that agrees with f at t^*_{jk} and $t^*_{j+1,k}$. Then

$$|f(x) - p(x)| \leq \frac{1}{8}(t^*_{j+1,k} - t^*_{jk})^2 \|D^2 f\|_{[t^*_{jk}..t^*_{j+1,k}]},$$

while

$$|p(x) - C_{k,\mathbf{t}} f(x)| \leq \max\{|f(t^*_{jk} - \alpha_j)|, |f(t^*_{j+1,k} - \alpha_{j+1})|\} \leq \text{const}|\mathbf{t}|^2 \|D^2 f\|,$$

by the Proposition. □

The proposition and its corollary are nicely illustrated in Figure IX(62).

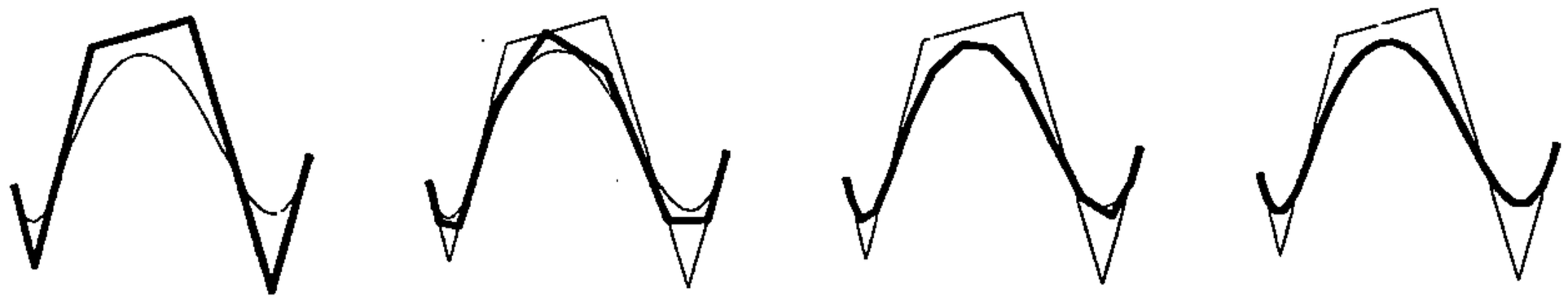
Now, directly from the Curry-Schoenberg theorem IX(44), the spline space $\$ = \$_{k,\mathbf{t}}$ is a subspace of $\$_{k,\widehat{\mathbf{t}}}$ for any knot sequence $\widehat{\mathbf{t}}$ that is a refinement of the knot sequence $\mathbf{t}$. In symbols:

$$(16) \qquad \mathbf{t} \subset \widehat{\mathbf{t}} \quad \Longrightarrow \quad \$ = \$_{k,\mathbf{t}} \subset \$_{k,\widehat{\mathbf{t}}}.$$

This means that we can rewrite any $f \in \$$ as a spline with a *refined* knot sequence $\widehat{\mathbf{t}}$ and, by choosing the meshsize $|\widehat{\mathbf{t}}|$ small enough, be certain that the control polygon $C_{k,\widehat{\mathbf{t}}} f$ is close to the spline itself. This has immediate appeal when generating a computer graph of a spline since most graphing programs only plot broken lines anyway. Have a look at Figure (17) which shows the spline from Figure (10) along with its control polygons obtained by repeated **midpoint refinement** in which the next knot sequence is obtained by inserting into the current knot sequence all its knot interval midpoints.

Of course, this requires a way, preferably fast, for constructing the B-form of $f \in \$$ as an element of $\$_{k,\widehat{\mathbf{t}}}$ from its B-form based on $\mathbf{t}$. One such way is provided by knot insertion, to which we turn next.

Knot insertion Any refinement $\widehat{\mathbf{t}}$ of the knot sequence $\mathbf{t}$ can be reached by adding or inserting knots one at a time, as first proposed by Wolfgang Boehm [1980]. We derive his algorithm with the aid of the dual functional formula IX(54), much as we derived the differentiation formula X(12), by comparing coefficients. There being nothing to calculate for the case $k = 1$, we avoid certain mostly notational complications by assuming $k > 1$ throughout this discussion.



(17) FIGURE. After three midpoint refinements, the control polygon of this spline is graphically indistinguishable from the spline.

Assume that $\widehat{\mathbf{t}}$ is obtained from $\mathbf{t}$ by the insertion of one additional term, x say, with

$$\widehat{t}_j = \begin{cases} t_j, & \text{for } t_j < x; \\ t_{j-1}, & \text{for } t_{j-1} > x. \end{cases}$$

Then (see Problem 11), for any j,

$$\widehat{t}^*_{jk} = (1 - \widehat{\omega}_{jk}(x))t^*_{j-1,k} + \widehat{\omega}_{jk}(x)t^*_{jk}, \tag{18}$$

with

$$\widehat{\omega}_{jk} : x \mapsto \begin{cases} 0, & \text{for } x \le t_j; \\ \omega_{jk}(x) = \dfrac{x - t_j}{t_{j+k-1} - t_j}, & \text{for } t_j < x < t_{j+k-1}; \\ 1, & \text{for } t_{j+k-1} \le x \end{cases} \tag{19}$$

the broken line that agrees with ω_{jk} (see IX(15)) on the interval $(t_j \,..\, t_{j+k-1})$ and is constant outside it; for $t_j = t_{j+k-1}$, it's just $(\cdot - t_j)^0_+$.

We know from the dual functional formula IX(54) that the jth B-spline coefficient for $f \in \$$ as an element of $\$_{k,\widehat{\mathbf{t}}}$ can be computed as

$$\widehat{\alpha}_j = \lambda_{j,k,\widehat{\mathbf{t}}} f = \sum_{\nu=1}^{k} \frac{(-D)^{k-\nu}\widehat{\psi}_{jk}(\tau_j)}{(k-1)!} D^{\nu-1} f(\tau_j) \tag{20}$$

with

$$\widehat{\psi}_{jk} := (\widehat{t}_{j+1} - \cdot) \cdots (\widehat{t}_{j+k-1} - \cdot), \quad \text{all } j,$$

while its jth B-spline coefficient as an element of $\$$ is

$$\alpha_j = \lambda_{j,k,\mathbf{t}} f := \sum_{\nu=1}^{k} \frac{(-D)^{k-\nu}\psi_{jk}(\tau_j)}{(k-1)!} D^{\nu-1} f(\tau_j).$$

Now note that $\widehat{\psi}_{jk}$ appears *linearly* in (20), hence all we need to do in order to relate $\widehat{\alpha}_j$ to the α_i's is to express, if possible, $\widehat{\psi}_{jk}$ as a weighted sum of the ψ_{ik}'s, substitute the resulting expression into (20) and multiply out to obtain $\widehat{\alpha}_j$ as the *corresponding* weighted sum of the α_i's.

For this, note that

$$\widehat{\psi}_{jk} = \begin{cases} \psi_{jk}, & \text{if } t_{j+k-1} \le x; \\ \psi_{j-1,k}, & \text{if } x \le t_j. \end{cases}$$

Hence, correspondingly, $\widehat{\alpha}_j = \alpha_j$ or $= \alpha_{j-1}$, respectively, for these cases. There is therefore some thought needed only for the case $t_j < x < t_{j+k-1}$. In this case, we look at the particular weighted sum

$$w_1\psi_{j-1,k} + w_2\psi_{jk} = (t_{j+1} - \cdot)\cdots(t_{j+k-2} - \cdot)\big(w_1(t_j - \cdot) + w_2(t_{j+k-1} - \cdot)\big).$$

This agrees with $\widehat{\psi}_{jk}$ exactly when

$$\big(w_1\,(t_j - \cdot) + w_2\,(t_{j+k-1} - \cdot)\big) = (x - \cdot),$$

that is, when

$$w_1 = 1 - \omega_{jk}(x) \quad \text{and} \quad w_2 = \omega_{jk}(x).$$

By using $\widehat{\omega}_{jk}$ instead of ω_{jk}, we obtain the formula

$$\widehat{\lambda}_j := \lambda_{j,k,\widehat{\mathrm{t}}} = (1 - \widehat{\omega}_{jk}(x))\lambda_{j-1,k} + \widehat{\omega}_{jk}(x)\lambda_{jk}, \quad \text{all } j, \tag{21}$$

that covers *all* cases. This proves the following.

(22) B-spline Property (xi): Knot insertion. (W. Boehm [1980]) *If the knot sequence* $\widehat{\mathrm{t}}$ *is obtained from the knot sequence* t *by the insertion of just one term,* x *say, then, for any* $f \in \$$, $\sum_j \alpha_j \mathrm{B}_{j,k,\mathrm{t}} := f =: \sum_j \widehat{\alpha}_j \mathrm{B}_{j,k,\widehat{\mathrm{t}}}$ *with*

$$\widehat{\alpha}_j = (1 - \widehat{\omega}_{jk}(x))\alpha_{j-1} + \widehat{\omega}_{jk}(x)\alpha_j, \quad \text{all } j. \tag{23}$$

The calculation (23) has the following pretty graphical interpretation (and is usually so described in CAGD); see Figure (26).

(24) Corollary. *If the knot sequence* $\widehat{\mathrm{t}}$ *is obtained from the knot sequence* t *by the insertion of just one term, then, for any* $f \in \$$, *the vertices* $\widehat{P}_j = (\widehat{t}^*_j, \widehat{\alpha}_j)$ *of* $C_{k,\widehat{\mathrm{t}}}f$ *lie on* $C_{k,\mathrm{t}}f$, *that is,*

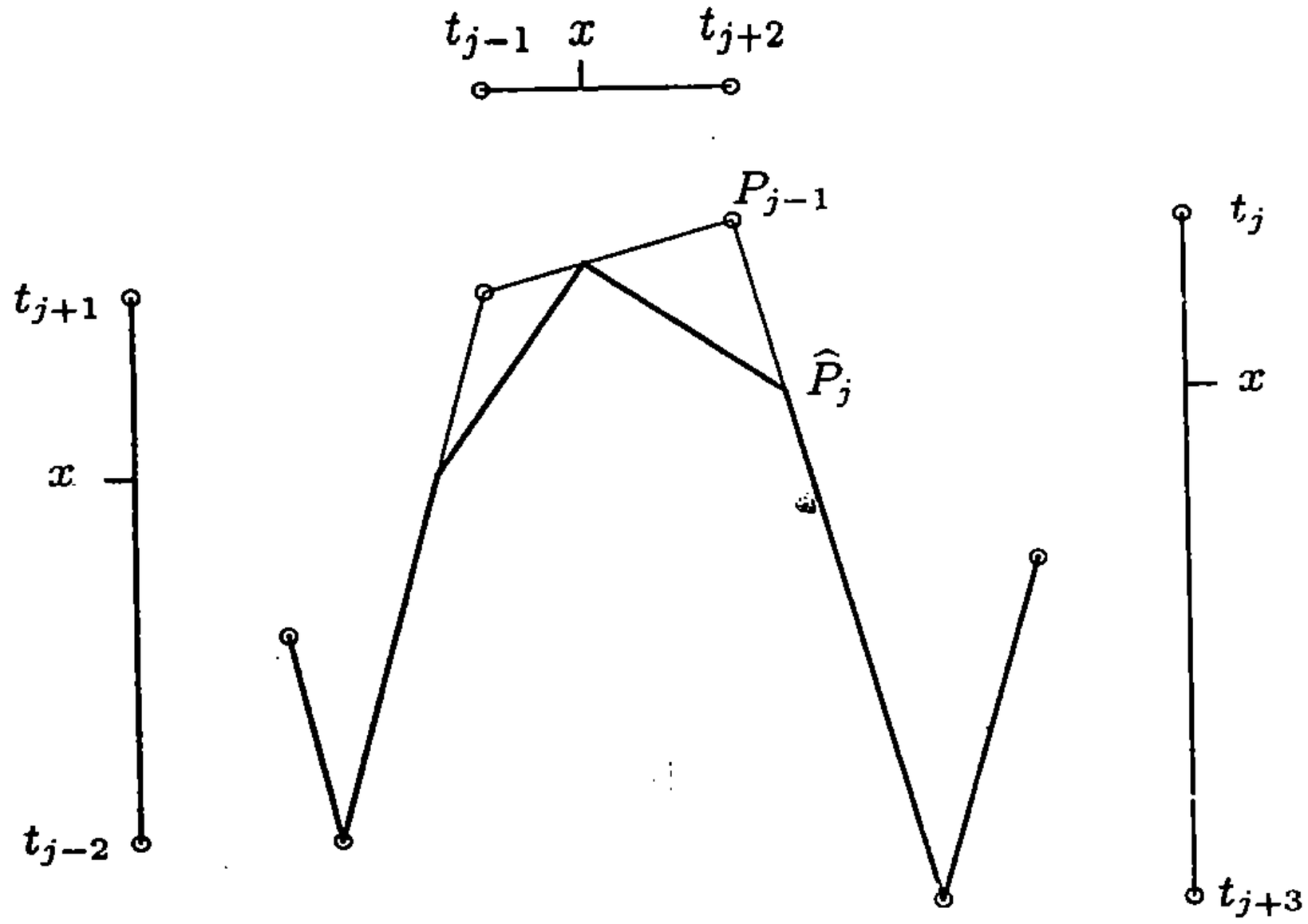
$$C_{k,\widehat{\mathrm{t}}}f(\widehat{t}^*_{jk}) = C_{k,\mathrm{t}}f(\widehat{t}^*_{jk}), \quad \text{all } j.$$

In other words, the refined control polygon can be constructed as the broken line interpolant at the sites $(\widehat{t}^*_j)$ *to the original control polygon.*

PROOF. By IX(38), the particular spline $p : x \mapsto x$ on $[a\,..\,b]$ has the numbers t^*_{jk} as its B-spline coefficients, hence (22) implies that

$$\widehat{P}_j = (1 - \widehat{\omega}_{jk}(x))P_{j-1} + \widehat{\omega}_{jk}(x)P_j, \quad \text{all } j \tag{25}$$

(with $P_j = (t^*_j, \alpha_j)$, all j). □



(26) FIGURE. Insertion of the point $x = 2$ into the knot sequence $(0, 0, 0, 0, \mathbf{1}, \mathbf{3}, 5, 5, 5, 5)$ with $k = 4$ produces the heavily drawn control polygon from that of Figure (10).

W. Boehm [1980] already points out that, for $t_j < x < t_{j+k-1}$ and with $j = r$, the calculation (23) is exactly the one for $\alpha_r^{[2]}(x)$ in X(24), the first step in the evaluation of $f \in \$$ at the site x. More than that, when (22) is used to insert x $k-1$ times, then the calculations carried out are exactly those of the first algorithm in de Boor [1972] and described in Chapter X. This conforms with the fact that, with $\widetilde{\mathbf{t}}$ the knot sequence obtained from $\mathbf{t}$ by inserting x exactly as many times as are needed to have $\widetilde{t}_j < \widetilde{t}_{j+1} = x \leq \widetilde{t}_{j+k-1} < \widetilde{t}_{j+k}$,

$$\widetilde{\mathrm{B}}_j := \mathrm{B}_{j,k,\widetilde{\mathbf{t}}}$$

is the only B-spline of order k for the knot sequence $\widetilde{\mathbf{t}}$ that is not zero at x, hence must equal 1 there, and therefore we must have $\widetilde{\alpha}_j = f(x)$ for the corresponding B-spline coefficient for $f \in \$_{k,\widetilde{\mathbf{t}}}$.

Variation diminution It is customary to denote the number of sign changes in a sequence $\alpha = (\alpha_i)_1^n$ by $\mathrm{S}^-\alpha$. To be precise, $\mathrm{S}^-\alpha$ is the largest integer r with the property that for some $1 \leq j_1 < \cdots < j_{r+1} \leq n$, $\alpha_{j_i}\alpha_{j_{i+1}} < 0$ for $i = 1, \ldots, r$.

(27) Lemma. (Lane & Riesenfeld [1983]) *If $\widetilde{\alpha}$ is the B-spline coefficient sequence obtained from the coefficient sequence α for $f \in \$$ by the insertion of (zero or more) knots into* t, *then*

$$S^- \widetilde{\alpha} \leq S^- \alpha.$$

PROOF. It is sufficient to prove this for the special case of insertion of just one knot. In that case, though, it is an immediate consequence of Corollary (24). □

One denotes similarly, for a function f, the number of its sign changes by $S^- f$ and defines it as the supremum over all numbers $S^-(f(\tau_1), \ldots, f(\tau_r))$ with r arbitrary and $\tau_1 < \cdots < \tau_r$ arbitrary in the domain of f.

(28) Corollary. (Schoenberg [1967]) *The number of sign changes in the spline function $\sum_j \alpha_j B_j$ is not bigger than the number of sign changes in its B-spline coefficient sequence $\alpha = (\alpha_j)$, that is,*

$$S^-\Big(\sum_j \alpha_j B_j\Big) \leq S^- \alpha.$$

PROOF. Set $f := \sum \alpha_j B_j$. Insert into t each of the entries of a given sequence $\tau = (\tau_1 < \cdots < \tau_r)$ enough times to have each appear in the resulting refined knot sequence $\widetilde{t}$ exactly $k-1$ times. Then $(f(\tau_1), \ldots, f(\tau_r))$ is a subsequence of the resulting B-spline coefficient sequence $\widetilde{\alpha}$ for f, hence, by the Lemma,

$$S^-(f(\tau_1), \ldots, f(\tau_r)) \leq S^- \widetilde{\alpha} \leq S^- \alpha.$$

□

With the formula X(12) for the derivative of a spline, this implies that a B-spline of order $k > 1$ is **unimodal** in the sense that it has exactly one maximum. More than that, it is **bell-shaped** in the sense that, for $j = 0, \ldots, k-1$, its jth derivative has at most j sign changes.

Actually, Corollary (28) is a special case of the following slightly more sophisticated statement.

(29) B-spline Property (xii): Variation diminution. *If $f = \sum_j \alpha_j B_j$ and $\tau_1 < \cdots < \tau_r$ are such that $f(\tau_{i-1}) f(\tau_i) < 0$, all i, then one can find indices $1 \leq j_1 < \cdots < j_r \leq n$ so that*

$$\alpha_{j_i} B_{j_i}(\tau_i) f(\tau_i) > 0 \qquad \text{for } i = 1, \ldots, r. \tag{30}$$

In this particular form, Property (xii) is proved in de Boor [1976]$_1$. But the proof there is merely a refinement of arguments by Karlin [1968: Ch. 10, Thm 4.1 & Ch. 2, Thm. 3.2] in support of the above Corollary (28). Here is a simple proof, based on knot insertion.

PROOF OF (29). Let $\widetilde{\mathbf{t}}$ be the refinement of the knot sequence $\mathbf{t}$ that contains each of the sites $\tau_{i+1/2} = (\tau_i + \tau_{i+1})/2$ exactly k times. Then the index sets $I_i := \{i : \widetilde{\mathrm{B}}_{jk}(\tau_i) \neq 0\}$, $i = 1, \ldots, r$, are pairwise disjoint. Further, assuming without loss of generality that $f(\tau_1) < 0$, we must have $(-)^i \widetilde{\alpha}_{j_i} > 0$ for some $j_i \in I_i$, all i. Thus

$$\widetilde{\alpha}_{j_i} \widehat{\mathrm{B}}_{j_i}(\tau_i)\, f(\tau_i) > 0 \quad \text{for} \quad i = 1, \ldots, r,$$

with $j_1 < \cdots < j_r$. Inductive application of the following lemma therefore finishes the proof. □

(31) Lemma. *Let $f =: \sum_j \alpha_j \mathrm{B}_j$ and assume that, for some $\tau_1 < \cdots < \tau_r$, $f(\tau_i) f(\tau_{i+1}) < 0$, all i. If there exist $1 \leq \widehat{j}_1 < \cdots < \widehat{j}_r \leq n+1$ so that*

$$\widehat{\alpha}_{\widehat{j}_i} \widehat{\mathrm{B}}_{\widehat{j}_i}(\tau_i) f(\tau_i) > 0 \quad \text{for} \quad i = 1, \ldots, r,$$

with $\widehat{\alpha}$ the B-spline coefficient sequence for f with respect to the knot sequence $\widehat{\mathbf{t}}$ obtained from $\mathbf{t}$ by the insertion of just one knot, then there also exist $1 \leq j_1 < \cdots < j_r \leq n$ so that

$$\alpha_{j_i} \mathrm{B}_{j_i}(\tau_i) f(\tau_i) > 0 \quad \text{for} \quad i = 1, \ldots, r. \tag{32}$$

PROOF. By (23), we must have $\alpha_j \widehat{\alpha}_{\widehat{j}_i} > 0$ for either $j = \widehat{j}_i - 1$ or $j = \widehat{j}_i$, while $\mathrm{B}_j(\tau_i) > 0$ for both (since $\widehat{\mathrm{B}}_{\widehat{j}_i}(\tau_i) > 0$). Therefore, we can choose $j_i \in \{\widehat{j}_i - 1, \widehat{j}_i\}$, all i, so that (32) holds, and certainly $j_1 \leq \cdots \leq j_r$ since $(\widehat{j}_i)$ is *strictly* increasing. However, since $\alpha_{j_i} \alpha_{j_{i+1}} < 0$, all i, there can be no equality here. □

B-spline Property (xii) is more precise than Corollary (28) in that it also connects the places at which the spline function f changes sign with the places at which its B-spline coefficient sequence changes sign. For each i, the index j_i corresponding to the site τ_i must be "near" τ_i in the sense that $\mathrm{B}_{j_i}(\tau_i) \neq 0$, that is, $t_{j_i} < \tau_i < t_{j_i+k}$, and also the coefficient α_{j_i} must be nonzero and have the same sign as the number $f(\tau_i)$. To illustrate this point further, suppose that the spline function $f = \sum_j \alpha_j \mathrm{B}_j$ has $k-1$ simple zeros in the open interval $(t_r \,..\, t_{r+1})$. It then follows that we can find sites $\tau_1 < \cdots < \tau_k$ all in the open interval $(t_r \,..\, t_{r+1})$ over which the function f changes sign, that is, for which $f(\tau_{i-1}) f(\tau_i) < 0$, $i = 2, \ldots, k$. Property (xii) then requires the existence of k indices $1 \leq j_1 < \cdots < j_k \leq n$ so that $\mathrm{B}_{j_i}(\tau_i) \neq 0$ and $\alpha_{j_i} f(\tau_i) > 0$ for $i = 1, \ldots, k$. But, since the k sites $\tau_1, \ldots, \tau_k$ all lie in the same interval $[t_r \,..\, t_{r+1}]$, there are only k distinct indices m for which $\mathrm{B}_m(\tau_i) \neq 0$ for some $i = 1, \ldots, k$, namely the indices $r+1-k, \ldots, r$. Consequently, $j_i = (r+1-k) + (i-1) = r-k+i$, and so

$$\alpha_{r-k+i} f(\tau_i) > 0, \qquad i = 1, \ldots, k,$$

that is, the k coefficients that determine the spline f on the interval $[t_r \,..\, t_{r+1}]$ must strictly alternate in sign and with the same orientation as that of f. If f is positive on the left end portion of the interval $[t_r \,..\, t_{r+1}]$, then the leftmost coefficient for that interval, that is, the number α_{r+1-k}, must also be positive.

We conclude that, for moderate k (that is, for moderately small $D_{k,\infty}$), the sequence α of B-spline coefficients (or, more precisely the control polygon) for a spline function f gives a fair idea of the graph of f.

Schoenberg's variation diminishing spline approximation

We continue to assume that $k > 1$. Corollary (28) and Proposition (12) form the basis for a very effective "shape preserving" spline approximation method of much use in Computer Aided Design (cf., for example, Riesenfeld [1973] or Barnhill & Riesenfeld [1974]). For a given function g on $[a \,..\, b]$, this spline approximation is defined by

$$Vg := \sum_{i=1}^{n} g(t^*_{ik})\mathrm{B}_i \qquad \text{on } [a \,..\, b], \tag{33}$$

with $\mathrm{t}^* := (t^*_{ik})_1^n$ the knot averages given in IX(39). From Corollary (28) and the definition of $\mathrm{S}^- g$, we know that

$$\mathrm{S}^- Vg \le \mathrm{S}^-(g(t^*_{ik})) \le \mathrm{S}^- g. \tag{34}$$

But more is true. By IX(38),

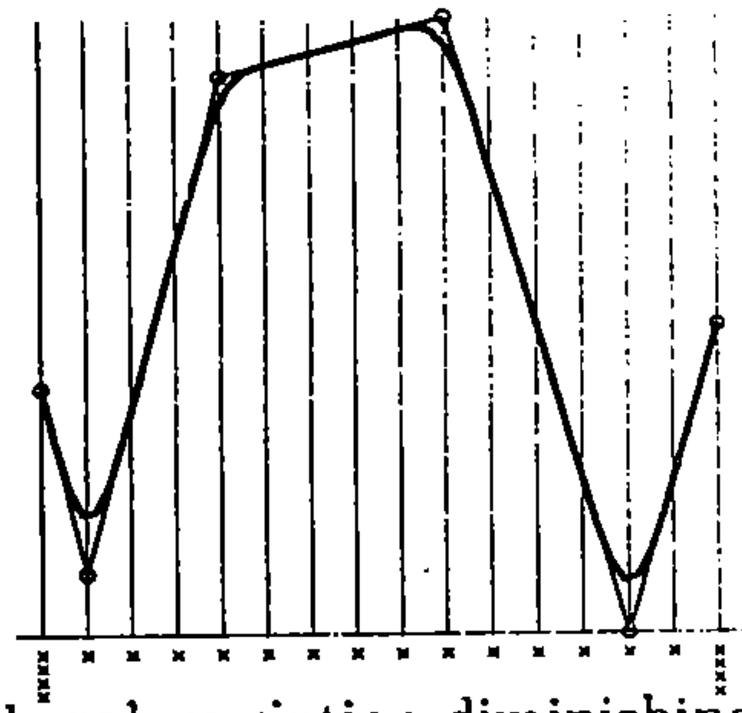
$$V\ell = \ell \quad \textit{for all straight lines } \ell. \tag{35}$$

We combine this fact with (34) to obtain the following description of the shape-preserving or *variation diminishing* character of Schoenberg's transformation V:

$$\mathrm{S}^-(Vg - \ell) \le \mathrm{S}^-(g - \ell) \qquad \text{for all } \ell \in \Pi_{<2}. \tag{36}$$

In words, *the spline approximation Vg to g crosses any particular straight line at most as many times as does g itself.* This is illustrated in Figure (10) for the special case that g happens to be the control polygon for Vg.

This suggests that V maps nonnegative functions to nonnegative functions, monotone functions to monotone functions, and convex functions to convex functions. A proof of these claims is immediate: Since the B-splines are nonnegative, so is Vg for nonnegative g. Therefore, if g is monotone nondecreasing, then, by the formula X(12) for the derivative of a spline, DVg is nonnegative, hence Vg is monotone nondecreasing. Finally, using



(37) FIGURE. Schoenberg's variation diminishing (and smooth) cubic spline approximation to the broken line sketch of a curve.

that differentiation formula twice, if g is convex, then D^2Vg is nonnegative, that is, Vg is convex.

Also, Vg provides a *local* approximation to g. The function Vg on the interval $[t_i \,..\, t_{i+1}]$ depends only on the values of g at the k "nearby" sites $t^*_{i-k+1,k}, \ldots, t^*_{ik}$. In particular, if the points $(t^*_{i-k+j,k}, g(t^*_{i-k+j,k}))$, $j = 1, \ldots, k$, lie on a straight line, then Vg on the interval $[t_i \,..\, t_{i+k}]$ coincides with that same straight line.

These various properties of Schoenberg's variation diminishing spline approximation are illustrated in Figure (37) which shows a broken line sketch g of some shape (see Figure (10)) and its smooth cubic spline approximant Vg. The knots of the spline are indicated by $\times$'s.

Warning: Vg is a shape-preserving linear approximation, hence V is, in particular, a positive linear map and, for that reason, Vg cannot be a very high order approximation. If g has r continuous derivatives for some $r \geq 2$, then

$$\|g - Vg\| \leq \text{const}_{g,k} |\mathrm{t}|^2$$

and no exponent higher than 2 can be put there, even if r is greater than 2; see Example XII(11). This means that Vg is not as good an approximation to a smooth function as splines are capable of providing, as shown in the next chapter.

Problems

1. Sharpen (3) in case $t_i = x$.

2. From (5),

$$(*) \qquad |\alpha_i| \leq \gamma \| \sum \alpha_j B_{j,3} \|_{[t_{i+1} \,..\, t_{i+2}]}$$

with $\gamma = 3$ the best possible constant.

(i) Prove that, if α_{i-1}, α_i, α_{i+1} all have the same sign, then, (*) holds

even with $\gamma = 2$. (Hint: Make use of Problem IX.8.)

(ii) What is the smallest value of γ we can choose in (*) in case $\alpha_{i-1} = \alpha_i = \alpha_{i+1}$?

3. Verify that Corollary (8) carries real information by the following numerical experiment: Calculate $\|f\|$ for the spline of order 20 with a bi-infinite uniform knot sequence (say, with knots at the integers) and with B-coefficients $\alpha_i = (-)^i$, all i. How big a γ must you choose to have $|\alpha_i| \leq \gamma\|f\|$ be true? Now change just one coefficient from $(-)^i$ to $(-)^i + .0001$ (that is, in the fourth significant place) and calculate the resulting change in $\|f\|$. In what significant digit did $\|f\|$ change?

4. Prove: (i) If $f = \sum_i \alpha_i B_{i,3}$ (that is, f is a parabolic spline), then f is monotone on $[t_r \,..\, t_s]$ if and only if the corresponding B-coefficient sequence $(\alpha_i)_{r-2}^{s-1}$ is monotone. (ii) Prove that (i) does not generalize to higher order splines by constructing a parabolic spline that is positive on $[t_r \,..\, t_s]$ while some of the coefficients $\alpha_{r-1}, \ldots, \alpha_{s-2}$ are negative. To what an extent does (i) generalize to higher order splines?

5. Verify B-spline Property (xii) directly for the coefficients found in Example IX(58).

6. Let $f = \sum_i \alpha_i B_{i,k}$; then $Df = \sum_i \alpha_i^{(2)} B_{i,k-1}$, by X(15)-X(16). Prove that $S^-(\boldsymbol{\alpha}^{(2)}) \geq 1 + S^-(\boldsymbol{\alpha})$, in case $\boldsymbol{\alpha}$ has only finitely many nonzero entries. Then conclude that the B-spline coefficient sequence for the jth derivative of a B-spline has *exactly* j strong sign changes.

7. Prove that $Vg = g$ at every t_i with $t_i = \cdots = t_{i+k-1}$, and, in particular, at a and b.

8. Verify that, for $n = k$ and $[a \,..\, b] = [0 \,..\, 1]$, Vg is the Bernstein polynomial of order k for g (see IX(10)).

9. Generate the spline curve of Figure (37) with the aid of BVALUE from the function table

τ	0	1/3	4/3	3	13/3	5
$g(\tau)$	2	1/2	9/2	5	0	5/2

.

10. Prove that, for any continuous $f \in \$$ with B-spline coefficient sequence α and any $x \in [t_j \,..\, t_{j+k}]$, $|\alpha_j - f(x)| \leq \text{const}|\mathbf{t}|\,\|Df\|_{[t_j..t_{j+k}]}$ for some const that is independent of f or $\mathbf{t}$.

11. When the single knot ζ is inserted into $\mathbf{t}$ to obtain the knot sequence $\widehat{\mathbf{t}}$, then, for some i,

$$\widehat{t}_j = \begin{cases} t_j, & \text{for } j < i; \\ \zeta, & \text{for } j = i; \\ t_{j-1}, & \text{for } j > i, \end{cases}$$

with i uniquely determined by the condition $t_{i-1} < \zeta < t_i$ *if* that condition obtains. In the contrary case, there are, offhand, several choices for i. Does

it matter which one one chooses? Test this question by proving (18).

12. Work out the details of knot insertion for the special case $k = 1$, being sure to identify the places in the text where the assumption $k > 1$ was explicitly needed.

13. (Blossoming) Consider the dual functional formula IX(54) *as it depends on* t, and as a linear functional on $\Pi_{<k}$. Specifically, prove the following.
(a) λ_j is a function of $t_{j+1}, \ldots, t_{j+k-1}$ only, that is,

$$\lambda_j = \lambda(t_{j+1}, \ldots, t_{j+k-1}), \quad \text{all } j,$$

for a certain linear functional λ on $\Pi_{<k}$ parametrized by $k-1$ variables. (Recall that, for $p \in \Pi_{<k}$, $\lambda_j p$ is independent of τ_j.)
(b) λ is **symmetric**, that is, $\lambda(s_1, \ldots, s_{k-1}) = \lambda(s_{\pi(1)}, \ldots, s_{\pi(k-1)})$ for every permutation π of order $k-1$.
(c) λ is **multiaffine**, that is, an affine function of each of its $k-1$ arguments. (To recall, a map $f : V \to W$ is **affine** if $f(\alpha v + (1-\alpha)w) = \alpha f(v) + (1-\alpha)f(w)$ for all $v, w \in V$ and all real scalars α.)
(c) For any $x \in \mathbb{R}$, $\lambda(x, x, \ldots, x) = [x]$, that is, equal to evaluation at x.
It follows that, for each $p \in \Pi_{<k}$, the scalar-valued map $blossom(p) : (s_1, \ldots, s_{k-1}) \mapsto \lambda(s_1, \ldots, s_{k-1})p$ is symmetric, multiaffine, and satisfies $blossom(p)(x, .., x) = p(x)$, all x. This identifies $blossom(p)$ as the *polar form* or *blossom* of p, the latter a term introduced by L. Ramshaw [1989] who uses this insight for an alternative approach to knot refinement that has become very popular in CAGD. These observations, using the term 'polar form', were already made by de Casteljau in the 60's (see, e.g., de Casteljau [1963]), but did not become public knowledge until Ramshaw's work.

XII

Local Spline Approximation and the Distance from Splines;

NEWNOT

In this chapter, we use Schoenberg's variation diminishing spline approximation and other local approximation schemes in order to establish how well a function of a certain smoothness can possibly be approximated by splines of a certain order. Such results provide an ideal against which one may then measure the actual performance of a specific spline approximation scheme.

We continue to use the setup of the preceding chapter. We have a knot sequence $\mathrm{t} = (t_i)_1^{n+k}$ with $t_i < t_{i+k}$, all i, and $t_1 = \cdots = t_k = a$, $t_{n+1} = \cdots = t_{n+k} = b$ so that

$$[a\,..\,b] = [t_k\,..\,t_{n+1}].$$

We are interested in approximating a certain function g on $[a\,..\,b]$ by splines of order k with knot sequence $\mathbf{t}$, that is, by elements of $\$_{k,\mathbf{t}}$. We use the abbreviation

$$\|g\| := \max_{a\le x\le b} |g(x)|$$

and the modulus of continuity

$$\omega(g;h) := \max\{\,|g(x)-g(y)| : |x-y|\le h, x, y\in[a\,..\,b]\,\}$$

of the function g, both familiar from II(18).

The distance of a continuous function from $\$_{k,\mathrm{t}}$ Choose $\tau_1 \le \cdots \le \tau_n$ in any way whatsoever in $[a\,..\,b]$ and consider the kth order spline approximation Ag to the continuous function g on $[a\,..\,b]$, constructed simply as follows:

$$Ag := \sum_{i=1}^{n} g(\tau_i)\mathrm{B}_i \quad \text{on } [a\,..\,b]. \tag{1}$$

The transformation A reproduces constants, that is, $Ag = g$ in case g

is a constant function, $g(x) = c$ for all x in $[a\,..\,b]$. This is so because, by IX(37), B-splines sum up to one. This property, together with the fact that B-splines are non-negative and have small support (that is, the rest of Property IX(iv)), make it very easy to establish a useful estimate for the error in the approximation scheme (1). The argument is entirely analogous to the one given in Chapter III for the error estimate III(18).

Take a site $\hat{x}$ in some interval $[t_j\,..\,t_{j+1}]$ in $[a\,..\,b]$. Then

$$(Ag)(\hat{x}) = \sum_{i=j+1-k}^{j} g(\tau_i)\mathrm{B}_i(\hat{x})$$

(see Figure IX(27)), while also

$$g(\hat{x}) = g(\hat{x}) \sum_{i=j+1-k}^{j} \mathrm{B}_i(\hat{x}) = \sum_{i=j+1-k}^{j} g(\hat{x})\mathrm{B}_i(\hat{x})$$

since B-splines sum up to one. Therefore

$$g(\hat{x}) - (Ag)(\hat{x}) = \sum_{i=j+1-k}^{j} \big(g(\hat{x}) - g(\tau_i)\big)\mathrm{B}_i(\hat{x}),$$

and, taking absolute values on both sides and using the non-negativity of the B-splines,

$$\begin{aligned}\big|g(\hat{x}) - (Ag)(\hat{x})\big| &\le \sum_{i=j+1-k}^{j} \big|g(\hat{x}) - g(\tau_i)\big|\mathrm{B}_i(\hat{x}) \\ &\le \max\{\,|g(\hat{x}) - g(\tau_i)| : j-k < i \le j\,\}.\end{aligned}$$

Now choose the τ_i's appropriately, each τ_i "near" the support of B_i. For instance, we might choose

$$\tau_i = t_{i+k/2}, \qquad i = 1, \ldots, n$$

where we declare

$$t_{i+k/2} := \big(t_{i+(k-1)/2} + t_{i+(k+1)/2}\big)/2$$

in case k is odd. Then

$$\begin{aligned}&\max\{\,|g(\hat{x}) - g(\tau_i)| : j-k < i \le j\,\} \\ &\quad \le \max\{\,|g(x) - g(y)| : x, y \in [t_{j+1-k/2}\,..\,t_{j+1}] \text{ or } x, y \in [t_j\,..\,t_{j+k/2}]\,\} \\ &\quad \le \omega(g; k|\mathrm{t}|/2) \\ &\quad \le \lfloor (k+1)/2 \rfloor \omega(g; |\mathrm{t}|),\end{aligned}$$

(using the monotonicity and subadditivity of $\omega(g;\cdot)$, see Problem II.6), with

$$|\mathbf{t}| := \max_i \Delta t_i$$

the **mesh size** of the knot sequence $\mathbf{t}$, as before. We conclude that

$$\|g - Ag\| \leq \lfloor (k+1)/2 \rfloor \omega(g;|\mathbf{t}|)$$

and therefore

$$\text{(2)} \qquad \operatorname{dist}(g, \$_{k,\mathbf{t}}) := \min\{\, \|g - s\| : s \in \$_{k,\mathbf{t}} \,\} \leq \operatorname{const}_k \omega(g;|\mathbf{t}|).$$

If we choose $\tau_i = t^*_{ik} = (t_{i+1} + \cdots + t_{i+k-1})/(k-1)$, all i, then the transformation A becomes Schoenberg's variation diminishing approximation method V XI(33). A more careful analysis by Marsden [1972] of the error $g - Vg$ produces the following striking estimate:

$$\text{(3)} \qquad \operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \|g - Vg\| \leq 2\omega\Big(g; \min\Big\{ \frac{b-a}{\sqrt{2k-2}}, |\mathbf{t}|\sqrt{(k/12)} \Big\}\Big)$$

which brings in the order of the spline as well as the mesh size $|\mathbf{t}|$ of the knot sequence and so establishes a kind of connection between the estimate (2) and Jackson's theorem II(22).

The estimate (2) shows that the distance of any continuous function g from $\$_{k,\mathbf{t}}$ goes to zero with the mesh size $|\mathbf{t}|$. This says that we can approximate continuous functions arbitrarily well by splines of a fixed order if we are willing to use many knots. The estimate shows further that $\operatorname{dist}(g, \$_{k,\mathbf{t}})$ goes to zero at least as fast as the modulus of continuity $\omega(g;|\mathbf{t}|)$ of g at $|\mathbf{t}|$ goes to zero with $|\mathbf{t}|$. It is possible to show that this estimate is sharp in the sense that, for some continuous function g and some sequence $(\mathbf{t})$ of knot sequences with $|\mathbf{t}| \to 0$, the best approximation error goes to zero no faster than the bound in (2). The reader is reminded of the discussion of broken line approximation with a uniform knot sequence on $[-1\,..\,1]$ to the function $g(x) = \sqrt{|x|}$ in Chapter III (see III(19)) and taken up again in Problem III.5 there. To recall, for $g(x) = \sqrt{|x|}$, $\omega(g;h) = h^{1/2}$ while, with $h = 2/N$ and with ih, $i = -N+1, \ldots, N-1$, the interior knots in $[-1\,..\,1]$ for our spline, with I_2 broken line interpolation,

$$\operatorname{dist}(g, \$_{2,\mathbf{t}}) \geq \|g - I_2 g\|/2 = (\sqrt{h}/4)/2 = |\mathbf{t}|^{1/2}/8.$$

In this example, the continuous function g has a singularity, and this is typical for functions for which the bound (2) is sharp as far as the order in $|\mathbf{t}|$ is concerned. For smooth functions, that is, for functions with many continuous derivatives, much better estimates can be given.

The distance of a smooth function from $\$_{k,\mathbf{t}}$ can be bounded in various ways. Here is a quickie.

The distance of the function g from $\$_{k,\mathbf{t}}$ is the same as the distance of the function $g - s$ from $\$_{k,\mathbf{t}}$ in case s is itself a spline of order k with knots $\mathbf{t}$. In symbols,

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) = \operatorname{dist}(g - s, \$_{k,\mathbf{t}}) \qquad \text{for all } s \in \$_{k,\mathbf{t}}.$$

Therefore, by (2),

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \operatorname{const}_k \omega(g - s; |\mathbf{t}|) \quad \text{for all } s \in \$_{k,\mathbf{t}} \cap C[a\,..\,b].$$

Further, from II(20),

$$\omega(g - s; h) \leq h\|Dg - Ds\|$$

in case g and s are sufficiently smooth; having them both piecewise continuously differentiable suffices. We conclude that

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \operatorname{const}_k |\mathbf{t}|\,\|Dg - Ds\| \quad \text{for all } s \in \$_{k,\mathbf{t}} \cap C[a\,..\,b]$$

and, on choosing s so as to make this bound as small as possible and recalling from X(9) that

$$\$_{k-1,\mathbf{t}} = \{\, Ds : s \in \$_{k,\mathbf{t}} \cap C[a\,..\,b] \,\} \text{ on } [a\,..\,b],$$

we finally obtain the estimate

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \operatorname{const}_k |\mathbf{t}| \operatorname{dist}(Dg, \$_{k-1,\mathbf{t}}) \tag{4}$$

in case g has a piecewise continuous derivative.

Of course, we can use (2) again, this time to estimate $\operatorname{dist}(Dg, \$_{k-1,\mathbf{t}})$, in case Dg is continuous and so get

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \operatorname{const}'_k |\mathbf{t}| \omega(Dg; |\mathbf{t}|) \tag{5}$$

with $\operatorname{const}'_k := \operatorname{const}_k \operatorname{const}_{k-1}$. If the function g is even smoother and if $k - 1 > 1$, we can now repeat the procedure and find that

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \operatorname{const}''_k |\mathbf{t}|^2 \omega(D^2 g; |\mathbf{t}|)$$

with $\operatorname{const}''_k := \operatorname{const}'_k \operatorname{const}_{k-2}$. Proceeding in this way, we obtain the analog of Jackson's theorem for polynomials (Theorem II(22)).

(6) Theorem (Jackson type). *For $j = 0, \dots, k-1$, there exists $\mathrm{const}_{k,j}$ so that, for all $\mathbf{t} = (t_i)_1^{n+k}$ with*

$$t_1 = \cdots = t_k = a < t_{k+1} \le \cdots < b = t_{n+1} = \cdots = t_{n+k} \tag{7}$$

and for all $g \in C^{(j)}[a \,..\, b]$,

$$\mathrm{dist}\,(g, \$_{k,\mathbf{t}}) \le \mathrm{const}_{k,j}\, |\mathbf{t}|^j \omega(D^j g; |\mathbf{t}|). \tag{8}$$

In particular, for $j = k-1$, we get

$$\mathrm{dist}\,(g, \$_{k,\mathbf{t}}) \le \mathrm{const}_k\, |\mathbf{t}|^k \|D^k g\| \tag{9}$$

in case g has k continuous derivatives (since then $\omega(D^{k-1}g; h) \le h\|D^k g\|$).

The theorem states that the distance of a smooth function from $\$_{k,\mathbf{t}}$ goes to zero at least as fast as the kth power of the mesh size $|\mathbf{t}|$. This order cannot be improved except in a trivial case in which the distance is obviously zero, as the following theorem shows.

(10) Theorem (Saturation theorem). *Suppose $(\xi^{(n)})$ is a sequence of breaks sequences satisfying the following* **mixing condition**: *For some positive number ρ and all n and all interior breaks $\xi_i^{(n)}$ there exists $\hat{n} > n$ with* $\mathrm{dist}\,(\xi^{(n)}, \xi^{(\hat{n})}) > \rho|\xi^{(n)}|$. (This condition is, for example, satisfied by the sequence of uniform partitions.) *Then* $\mathrm{dist}\,(g, \Pi_{<k,\xi^{(n)}}) = o\left(|\xi^{(n)}|^k\right)$ *implies that $g \in \Pi_{<k}$.*

For a proof and further such results, see DeVore & Richards [1973].

Theorem (6) only tells us at what rate a best approximation to a smooth function g from $\$_{k,\mathbf{t}}$ converges to g as we make the mesh size $|\mathbf{t}|$ small. It does not tell us how to construct such an approximation. The reader may feel that the proof of Theorem (6) *is* based on a construction, namely the approximation $Ag = \sum_i g(\tau_i)\mathrm{B}_i$ to g that gave us the bound (2). In response to this, we illustrate by an example the important fact that the error in the approximation Ag to g is, in general, no better than $\mathcal{O}\left(|t|^2\right)$ no matter how smooth the function g might be.

(11) Example: The degree of approximation of Schoenberg's variation diminishing spline approximation We take the interval of interest to be $[a \,..\, b] = [0 \,..\, 1]$, take the knots in $(0 \,..\, 1)$ uniform, that is, $t_{k+i} = ih$, $i = 1, \dots, N-1$, with $Nh = 1$, and consider Schoenberg's spline approximation

$$Vg = \sum_i g(t_{ik}^*)\mathrm{B}_i$$

to the specific function $g(x) = x^2$. Clearly, this function g has derivatives

of all orders, it is even analytic, and so offers the transformation V a fair chance for showing just how well it can approximate to smooth functions.

Take $k > 2$. Then, on the interval $[0\,..\,1]$, our parabola g agrees with some spline of order k whatever knots $\mathbf{t}$ we might pick. We compute its B-spline coefficients directly from IX(55):

$$\alpha_i = g(t^*_{ik}) + \frac{(-)^{k-3} D^{k-3}\psi_{ik}(t^*_{ik})2}{(k-1)!} \qquad (\text{since } D^2 g = 2)$$

and so conclude that the error on $[0\,..\,1]$ is given by

$$g - Vg = \sum_i \varepsilon_i \mathrm{B}_i \quad \text{with } \varepsilon_i := (-)^{k-3} D^{k-3}\psi_{ik}(t^*_{ik})2/(k-1)!, \quad \text{all } i.$$

Further,

$$\psi_{ik}(x) = \Big[(-x)^{k-1} + \Big(\sum_{j=1}^{k-1} t_{i+j}\Big)(-x)^{k-2} + \sum_{j=1}^{k-2}\sum_{r=j+1}^{k-1} t_{i+j}t_{i+r}(-x)^{k-3} + \text{l.o.t.}\Big].$$

Therefore, using the fact that $t^*_{ik} = \sum_{j=1}^{k-1} t_{i+j}/(k-1)$, we obtain

$$\frac{(-)^{k-3}\psi^{(k-3)}_{ik}(x)}{(k-1)!} = \frac{x^2}{2} - t^*_{ik}x + \frac{1}{2}\left[\big((k-1)t^*_{ik}\big)^2 - \sum_{j=1}^{k-1} t^2_{i+j}\right]/[(k-1)(k-2)]$$

and so

$$\begin{aligned}\varepsilon_i &= (-)^{k-3}\psi^{(k-3)}_{ik}(t^*_{ik})2/(k-1)! \\ &= 2(t^*_{ik})^2\left(\frac{1}{2} - 1 + \frac{1}{2}\,\frac{k-1}{k-2}\right) - \sum_{j=1}^{k-1} t^2_{i+j}/((k-1)(k-2)) \\ &= \Big[(t^*_{ik})^2 - \sum_{j=1}^{k-1} t^2_{i+j}/(k-1)\Big]/(k-2).\end{aligned}$$

For a uniformly spaced knot sequence $\mathbf{t}$ with mesh size $|\mathbf{t}| = h$, this expression simplifies. If $t_{i+j} = t_i + jh$, $j = 1, \ldots, k-1$, then

$$\varepsilon_i = \left[\left(\frac{hk}{2}\right)^2 - h^2\,\frac{k(2k-1)}{6}\right]/(k-2) = -h^2 k/12.$$

We conclude that the error in the approximation $Vg = \sum_i g(t^*_{ik})\mathrm{B}_i$ to our parabola g is given by

$$g - Vg = \sum_i \varepsilon_i \mathrm{B}_i$$

with

$$\varepsilon_i = -h^2 k/12 \quad \text{for} \quad i = k-1, \ldots, n-k+2$$

in case of a knot sequence t with uniform mesh size h on $[0\,..\,1]$. But this implies that *the error is constantly equal to* $-h^2k/12$ *on the interval* $[t_{2(k-1)}\,..\,t_{n+1-k}]$. In particular, the error is no better than $\mathcal{O}\left(h^2\right)$ even though our function g has infinitely many derivatives.

On the other hand, *the error* $g - Vg$ *is always* $\mathcal{O}\left(|\mathrm{t}|^2\right)$ *in case the function* g *has two continuous derivatives.* This can be seen as follows: Take $\hat{x} \in [t_j\,..\,t_{j+1}]$ and let p be the straight line that agrees with the function g at the site $\hat{x}$ twice, that is, $p(x) = g(\hat{x}) + g'(\hat{x})(x - \hat{x})$. Therefore $Vp = p$, hence

$$g - Vg = g - p - V(g - p).$$

But, since the function $g - p$ vanishes at $\hat{x}$, it follows that the error at $\hat{x}$ is given by

$$g(\hat{x}) - (Vg)(\hat{x}) = -V(g-p)(\hat{x}) = -\sum_{i=j+1-k}^{j} (g-p)(t^*_{ik})\mathrm{B}_i.$$

Consequently,

$$\left|g(\hat{x}) - (Vg)(\hat{x})\right| \le \max\{\, |(g-p)(x)| : t^*_{j+1-k,k} \le x \le t^*_{jk} \,\}.$$

Now, the function $g - p$ vanishes twofold at $\hat{x}$, therefore is of order h^2 in any interval of length h containing $\hat{x}$. Precisely, we have from Theorem I(14) that

$$|(g-p)(x)| = (x - \hat{x})^2 |[\hat{x}, \hat{x}, x]g|$$

and therefore, with I(vii),

$$\begin{aligned} \left|g(\hat{x}) - (Vg)(\hat{x})\right| &\le \max\{\, (t_{j+1} - t^*_{j+1-k,k}, (t^*_{jk} - t_j)^2 \,\} \|g''\|/2 \\ &\le \mathrm{const}_k |\mathrm{t}|^2 \|g''\|. \end{aligned}$$

We conclude that

$$\|g - Vg\| \le \mathrm{const}_k |\mathrm{t}|^2 \|g''\| \qquad \text{for all } g \in C^{(2)}. \tag{12}$$

□

Local spline approximation schemes that provide the best possible order of approximation The argument for the $|t|^2$-convergence of Schoenberg's approximation Vg to a smooth function g just given contains a very simple idea for the construction of spline approximation schemes that realize the potential order of approximation to smooth functions as expressed in Theorem (6). We take the time to discuss this idea in some detail because such schemes also allow us to gain some insight into the effect of *knot placement* on the achievable accuracy of approximation.

Suppose we constructed a spline approximation Ag in $\$_{k,t}$ in the form

$$Ag = \sum_{i=1}^{n} (\mu_i g) \mathrm{B}_i \tag{13}$$

with each $\mu_i g$ a number that depends linearly and continuously on the continuous function g, that is, μ_i is a continuous linear functional on $C[a\,..\,b]$. This means that

$$\mu_i(g+f) = \mu_i g + \mu_i f, \quad \mu_i(\alpha g) = \alpha \mu_i g$$

and that

$$|\mu_i g| \le \|\mu_i\|\,\|g\| \quad \text{for all} \quad g \in C[a\,..\,b] \tag{14}$$

with $\|\mu_i\| := \sup\{\, |\mu_i g|/\|g\| : g \in C[a\,..\,b] \,\}$ a *finite* constant, also called the *norm* of the linear functional μ_i. It then follows that our approximation scheme A is linear,

$$\begin{aligned} A(\alpha g) &= \alpha Ag \quad , \quad \text{for all} \quad \alpha \in \mathbb{R},\ g \in C[a\,..\,b], \\ A(g+f) &= Ag + Af \ , \quad \text{for all} \quad f, g \in C[a\,..\,b], \end{aligned}$$

and is continuous or bounded, that is,

$$\|A\| := \sup_{g \in C[a..b]} \|Ag\|/\|g\| \le \max_i \|\mu_i\| < \infty \tag{15}$$

since, for any site x and any function g,

$$|(Ag)(x)| \le \max_i |\mu_i g| \le \left(\max_i \|\mu_i\|\right) \|g\|$$

by XI(3) and (14).

The approximation (1) is of this form, with $\mu_i g = g(\tau_i)$, all i, hence $\|\mu_i\| = 1$, all i. More generally, one might choose

$$\mu_i g = \sum_{j=1}^{K} \beta_{ij}\, g(\tau_{ij}) \tag{16}$$

with τ_{ij} sites in $[a\,..\,b]$ and β_{ij} certain specified real numbers. For such μ_i, we then have $\|\mu_i\| \le \sum_j |\beta_{ij}|$. Even more generally than this, one might choose

$$\mu_i g = \int_a^b g(x)\,\mathrm{d}M_i(x) \tag{17}$$

with M_i a function of bounded variation, that is, the difference of two monotone functions, and the integral a Stieltjes integral. This is, in fact, the most general form for such a number $\mu_i g$ and we mention it here only for the sake of completeness. A special case of (17) is the choice

$$\mu_i g = \int_a^b m_i(x) g(x)\,\mathrm{d}x \tag{18}$$

for some integrable function m_i, that is, some piecewise continuous function m_i, in which case $\|\mu_i\| = \int_a^b |m_i(x)|\,\mathrm{d}x$.

Suppose now, secondly, that, for each i, μ_i has its support in the interval $[t_i\,..\,t_{i+k}]$, that is, in the support of B_i. By this we mean that (14) can be strengthened to

$$|\mu_i g| \le \|\mu_i\|\,\|g\|_{[t_i..t_{i+k}]} \quad \text{for all} \quad g \in C[a\,..\,b]. \tag{19}$$

If $\mu_i g$ is of the form (16), then it will satisfy (19) provided $\tau_{ij} \in [t_i\,..\,t_{i+k}]$, all j. If $\mu_i g$ is of the form (18), then it will satisfy (19) provided m_i has its support in $[t_i\,..\,t_{i+k}]$, that is, provided $m_i(x) = 0$ for $x \notin [t_i\,..\,t_{i+k}]$. A different way of stating that the linear functional μ_i has its support in $[t_i\,..\,t_{i+k}]$ is to say that $\mu_i g = 0$ whenever the function g vanishes identically on $[t_i\,..\,t_{i+k}]$.

The resulting spline approximation $Ag = \sum_i (\mu_i g)\mathrm{B}_i$ is then *local* in the sense that, on the interval $[t_j\,..\,t_{j+1}]$, the approximation Ag to g depends only on the values of the function g on the interval $[t_{j+1-k}\,..\,t_{j+k}]$. In fact, we know that on $[t_j\,..\,t_{j+1}]$,

$$Ag = \sum_{i=j+1-k}^{j} (\mu_i g)\mathrm{B}_i \quad \text{and} \quad \sum_{i=j+1-k}^{j} \mathrm{B}_i = 1, \quad \mathrm{B}_i \ge 0, \text{ all } i,$$

therefore, (19) implies the bound

$$\|Ag\|_{[t_j..t_{j+1}]} \le \big(\max_i \|\mu_i\|\big)\|g\|_{[t_{j+1-k}..t_{j+k}]}, \quad \text{all } g \in C[a\,..\,b]. \tag{20}$$

In particular, the approximation Ag vanishes on the interval $[t_j\,..\,t_{j+1}]$ identically in case the function g vanishes identically on the slightly larger interval $[t_{j+1-k}\,..\,t_{j+k}]$.

The last ingredient for our successful local spline approximation scheme A is the requirement that it reproduce polynomials of order k, that is,

$$Ap = p \qquad \text{for all } p \in \Pi_{<k}. \tag{21}$$

Schoenberg's transformation V, for instance, reproduces straight lines and, in consequence, we were able to prove that $\|g - Vg\| = \mathcal{O}(|\mathrm{t}|^2)$ for all sufficiently smooth functions g. The combination of (20) and (21) allows us to prove, in the same way, that $\|g - Ag\| = \mathcal{O}(|\mathrm{t}|^k)$.

(22) Theorem. *Let $Ag := \sum_{i=1}^{n}(\mu_i g)\mathrm{B}_i$, all $g \in C[a\,..\,b]$, with $(\mathrm{B}_i)_1^n$ the B-spline basis for $\$_{k,\mathrm{t}}$, t satisfying (7), and μ_i a linear functional on the continuous functions $C[a\,..\,b]$ satisfying (19), all i. If the transformation A reproduces $\Pi_{<k}$, that is, if $Ap = p$ for all $p \in \Pi_{<k}$, then*

$$\|g - Ag\| \le (1 + \max_i \|\mu_i\|)\,\mathrm{const}_k \|D^k g\|\,|\mathrm{t}|^k, \quad \text{for all } g \in C^{(k)}[a..b]. \tag{23}$$

PROOF. For any polynomial p of order k, we have, by assumption,

$$g - Ag = g - p - (Ag - p) = g - p - A(g - p).$$

Therefore, from (20),

$$\begin{aligned}\|g - Ag\|_{[t_j..t_{j+1}]} &\le \|g - p\|_{[t_j..t_{j+1}]} + (\max_i \|\mu_i\|)\,\|g - p\|_{[t_{j+1-k}..t_{j+k}]} \\ &\le (1 + \max_i \|\mu_i\|)\,\|g - p\|_{[t_{j+1-k}..t_{j+k}]}\end{aligned}$$

Now choose the polynomial p of order k to make $g - p$ as small as possible on $[t_{j+1-k}\,..\,t_{j+k}]$ to get

$$\|g - Ag\|_{[t_j..t_{j+1}]} \le (1 + \max_i \|\mu_i\|)\,\mathrm{dist}\,(g, \Pi_{<k})_{[t_{j+1-k}..t_{j+k}]}, \tag{24}$$

and our assertion (23) follows from this by a reference to II(16). □

It is not very difficult to construct such local approximation schemes satisfying (21). But, to make them theoretically and practically useful schemes, we must construct them in such a way that the number $\max_i \|\mu_i\|$ appearing in (23) or (24) can be bounded *independently of* t. Only then can such a scheme claim to achieve the best possible order of approximation. This makes the construction of such schemes something of a challenge.

The first such scheme can be found in de Boor [1968], with μ_i of the form

$$\mu_i g = \sum_{j=1}^{k} \beta_{ij}\, g(\tau_{ij})$$

and $\tau_{i1}, \ldots, \tau_{ik}$ the extreme sites of the Chebyshev polynomial of order k

for the largest interval in $[t_{i+1} \,..\, t_{i+k-1}]$ of the form $[t_j \,..\, t_{j+1}]$, while the weights β_{ij} are so chosen that $Ap = p$ for all $p \in \Pi_{<k}$. For $k = 2$, this gives A as broken line interpolation (of Chapter III) (cf. Problem 2). For $k = 3$, this gives

$$g = \sum_{i=1}^{n} \frac{1}{2}\big(-g(t_{i+1}) + 4g(t_{i+3/2}) - g(t_{i+2})\big)\mathrm{B}_{i,3}$$

with $t_{i+3/2} := (t_{i+1} + t_{i+2})/2$ (cf. Problem IX.8). For $k > 3$, the weights β_{ij} depend on $\mathbf{t}$, but the choice of the sites $\tau_{i1}, \ldots, \tau_{ik}$ in a largest interval $[t_j \,..\, t_{j+1}]$ in $[t_{i+1} \,..\, t_{i+k-1}]$ ensures that the number $\max_i \|\mu_i\|$ can be bounded independently of $\mathbf{t}$. Also, it can be shown that this scheme reproduces not only $\Pi_{<k}$ but all of $\$_{k,\mathbf{t}}$ (on $[a \,..\, b]$); see Problem 3. In this connection, see de Boor [1976]$_2$ for a discussion of μ_i satisfying $\mu_i \mathrm{B}_j = \delta_{ij}$, all j (which would imply that $Af = f$ for $f \in \$_{k,\mathbf{t}}$).

The quasi-interpolant of de Boor & Fix [1973] does not quite fit the pattern. It is of the form

$$Qg := \sum_{i=1}^{n} (\lambda_i g)\mathrm{B}_i \tag{25}$$

with λ_i the dual functional IX(54). The scheme is therefore local, it reproduces $\Pi_{<k}$ since it even reproduces all of $\$_{k,\mathbf{t}}$ on $[a \,..\, b]$. But it fails to satisfy (14) since it is not even defined on all of $C[a \,..\, b]$. Still, an analysis of the error $g - Qg$ can be carried out along the above lines, as long as g is a smooth function; see de Boor & Fix [1973].

The quasi-interpolant Q, and similar schemes discussed in Lyche & Schumaker [1975], provide a local spline approximation to a smooth function g which *simultaneously* approximates the function g and its $k-1$ derivatives to optimal order, at least for knot sequences that are not too nonuniform.

(26) Theorem. *Let Qg be the quasi-interpolant to the function g as given by (25), with $\tau_i = t_{i+k/2}$, $i = 1, \ldots, n$. Then, there exists* $\mathrm{const}_{k,j}$, $j = 0, 1, \ldots, k-1$, *so that, for all $\mathbf{t} = (t_i)_1^{n+k}$ satisfying (7) and for all functions $g \in C^{(k-1)}[a \,..\, b]$, that is, for all $k-1$ times continuously differentiable functions g on $[a \,..\, b]$, the error in the quasi-interpolant Qg for g satisfies*

$$\|D^j g - D^j Qg\| \le \mathrm{const}_{k,j}(m_{\mathbf{t}})^{(2j-k)_+}|\mathbf{t}|^{k-j-1}\omega(D^{k-1}g; |\mathbf{t}|)$$

for $j = 0, \ldots, k-1$. Here, $m_{\mathbf{t}} := \max\{\, \Delta t_r/\Delta t_s : |r-s| = 1, k \le r, s \le n \,\}$ is the **local mesh ratio** *for $\mathbf{t}$, and this ratio enters the error bound only for $j > k/2$.*

A proof can be found in de Boor & Fix [1973]. Also, see Problems 7 and 8 for removal of the mesh ratio dependence.

We mention, for completeness, the theoretically important fact that, for $k > 1$, it is possible to choose the linear functionals μ_i in our approximation scheme A so that

$$\mu_i \mathrm{B}_j = \delta_{ij}, \qquad \text{all } j,$$

while μ_i is of the form

$$\mu_i g = \sum_j \beta_{ij} g(\tau_{ij})$$

with τ_{ij} certain sites in the interval $[t_{i+1} \,..\, t_{i+k-1}]$ and $\sum_j |\beta_{ij}| \leq D_{k,\infty}$. This fact can be found in de Boor [1976]$_2$; it is equivalent to B-spline Property XI(**x**) (see XI(4)), where specific values for the constants $D_{k,\infty}$ can be found. It follows that

$$|\mu_i g| \leq D_{k,\infty} \|g\|_{[t_{i+1} \,..\, t_{i+k-1}]} \quad \text{for all } g \in C[a \,..\, b]$$

while, with this particular choice for the μ_i's in (13), we have $A\mathrm{B}_i = \mathrm{B}_i$, all i, hence our approximation scheme A reproduces ali of $\$_{k,\mathbf{t}}$. For this particular scheme, we then obtain the following specialization of (24):

$$(27) \qquad \|g - Ag\|_{[t_j \,..\, t_{j+1}]} \leq (1 + D_{k,\infty}) \operatorname{dist}(g, \Pi_{<k})_{[t_{j+2-k} \,..\, t_{j+k-1}]}.$$

Good knot placement The spline approximation schemes discussed in this chapter are *local* and so allow us to gain some insight into the effect of knot placement on the achievable accuracy of spline approximation. We have used the resulting bounds so far only to describe dist $(g, \$_{k,\mathbf{t}})$ is terms of the (global) mesh size $|\mathbf{t}| = \max_i \Delta t_i$, for example, the bound (9)

$$\operatorname{dist}(g, \$_{k,\mathbf{t}}) \leq \mathrm{const}_k \|D^k g\| |\mathbf{t}|^k$$

or the more elaborate bound in Theorem (26). But we have in (24) and (27) some specific information about how we might choose **t** so as to make the bound in (24) or (27) small.

If we combine (27) with II(16), then we obtain the bound

$$(28) \qquad \|g - Ag\|_{[t_j \,..\, t_{j+1}]} \leq \mathrm{const}_k |I_j|^k \|D^k g\|_{I_j}$$

with I_j the interval $[t_{j+2-k} \,..\, t_{j+k-1}]$ and $|I_j|$ its length. Clearly, we can influence this bound in a beneficial way in case the kth derivative of g varies greatly in size by choosing I_j relatively small in areas where $|D^k g|$ is relatively large. But exactly how this should be done is not clear.

Exact minimization of the bound in (28) as a function of $\mathrm{t} = (t_i)_1^{n+1}$ (satisfying (7)) for fixed n is actually quite difficult. If we are willing to exert that kind of effort, then we are better off attacking directly the problem of

determining $\mathbf{t}$ (satisfying (7)) for fixed n so that dist $(g, \$_{k,\mathbf{t}})$ is as small as possible. But this is not an easy problem and repays all that effort only if dist $(g, \$_{k,\mathbf{t}})$ is quite sensitive to the precise location of some or all of the knots *and* it is important to approximate the function g well using a spline with as few knots as possible. In any event, we cannot even attempt to find such optimal knots unless we know the function g well enough to evaluate $\|g - f\|$ (see Example XIV(34) for one such attempt).

In typical practical situations, the function g is only known approximately or known implicitly (for example, as the solution of some differential equation). Still, we would like to have some scheme of choosing the knot sequence $\mathbf{t}$ appropriately. We cannot hope to place each knot optimally. We can only hope to obtain an optimal knot *distribution* or density. This being so, we now take certain liberties with the placement of individual knots in order to make the analysis easier. (Results by Barrow & Smith [1979] serve to justify this simplification.) We group the knots in the open interval $(a \,..\, b)$ into groups of $k-1$ each and let each such group coalesce into a knot of multiplicity $k-1$. If the number of knots for the knot sequence we have set our hearts on is not an exact multiple of $k-1$, then we simply bring in a few more. Let

$$\zeta_2 < \cdots < \zeta_m$$

be the distinct sites among these grouped and coalesced knots in the open interval $(a \,..\, b)$ and set $\zeta_1 := a$, $\zeta_{m+1} := b$. Then, on $[a \,..\, b]$, $\$_{k,\mathbf{t}}$ now coincides with $\Pi_{<k,\zeta} \cap C[a \,..\, b] = \Pi_{<k,\zeta,\mathbf{1}}$, and (28) becomes

$$(29) \quad \|g - Ag\|_{[\zeta_j..\zeta_{j+1}]} \le \text{const}_k \|D^k g\|_{[\zeta_j..\zeta_{j+1}]} |\Delta\zeta_j|^k, \qquad j = 1, \ldots, m.$$

This suggests that we place $\zeta_2, \ldots, \zeta_m$ so as to minimize

$$(30) \qquad \max_j \|D^k g\|_{[\zeta_j..\zeta_{j+1}]} |\Delta\zeta_j|^k.$$

Since

$$s(\alpha, \beta) := \|D^k g\|_{[\alpha..\beta]} |\beta - \alpha|^k$$

is a continuous function of α and β (if $D^k g$ is continuous) and monotone, increasing in β and decreasing in α, (30) is minimized (for fixed m) when $\zeta_2, \ldots, \zeta_m$ are chosen so that

$$\|D^k g\|_{[\zeta_j..\zeta_{j+1}]} |\Delta\zeta_j|^k = \text{ constant for } j = 1, \ldots, m.$$

The exact determination of such $\zeta_2, \ldots, \zeta_m$ is a somewhat expensive task and is not justified anyway, given the approximations we have already made or are about to make, because we usually do not know the kth derivative of g. But, the task is obviously equivalent to determining $\zeta_2, \ldots, \zeta_m$ so that

$$\left(\|D^k g\|_{[\zeta_j..\zeta_{j+1}]}\right)^{1/k} \Delta\zeta_j = \text{ constant for } j = 1, \ldots, m,$$

and therefore produces asymptotically the same distribution of ζ_j's as does the problem of determining $\zeta_2, \ldots, \zeta_m$ so that

$$(31) \qquad \int_{\zeta_j}^{\zeta_{j+1}} |D^k g(x)|^{1/k} \, \mathrm{d}x = \frac{1}{m} \int_a^b |D^k g(x)|^{1/k} \, \mathrm{d}x, \quad j = 1, \ldots, m.$$

This latter problem is very easy to solve if we replace the function $|D^k g|$ by some piecewise constant approximation

$$h \sim |D^k g|.$$

For then

$$G(x) := \int_a^x (h(s))^{1/k} \, \mathrm{d}s$$

is an easily computable, continuous and monotone increasing piecewise linear function, hence so is its inverse function G^{-1} (ignoring the possibility of multivaluedness at some breaks), and the stated task amounts to nothing more than evaluating the known piecewise linear function G^{-1} at the $m-1$ sites $jG(b)/m$, $j = 1, \ldots, m-1$.

It remains to discuss how one might obtain a piecewise constant approximation h to the function $|D^k g|$. Burchard [1977:Lemma 1.5] proposes to compute first some pp approximation *of order* $k+1$ to g (using some convenient choice of breaks) and then use the absolute value of its kth derivative for the approximation h. A different proposal can be found in de Boor [1974] and consists of constructing the function h from a current pp approximation f *of order* k to g as follows.

It really makes no difference whether we construct the piecewise constant function h or the continuous piecewise linear function

$$H(x) := \int_a^x h(s) \, \mathrm{d}s.$$

We can easily obtain one from the other. But, to the extent that h approximates the function $|D^k g|$, its integral H approximates the function

$$\mathrm{Var}_{[a..x]} D^{k-1} g := \int_a^x |D^k g(s)| \, \mathrm{d}s.$$

This latter function we can compute (in principle) from the $(k-1)$st derivative of g, hence we can compute an approximation to it from an approximation to $D^{k-1} g$.

Specifically, assume that we have some pp approximation f of order k to g. Its $(k-1)$st derivative $D^{k-1} f$ is then a piecewise constant function and can be written

$$(D^{k-1} f)(x) = \alpha_1 + \sum_{j=2}^{l} \alpha_j (x - \xi_j)_+^0$$

with $\xi = (\xi_i)_1^{l+1}$ the break sequence for f. The number α_j is the jump of $D^{k-1}f$ across the break ξ_j, $j = 1, \ldots, l$. The total variation of the function $D^{k-1}f$ on the interval $[a\,..\,x]$ is therefore

$$\operatorname{Var}_{[a..x]} D^{k-1}f = \sum_{j=2}^{l} |\alpha_j|(x - \xi_j)_+^0,$$

and, taking this to be an approximation to the function $\operatorname{Var}_{[a..x]} D^{k-1}g$, we construct the second order spline function H as an approximation to $\operatorname{Var}_{[a..x]} D^{k-1}f$. Specifically, we might choose $h = H' \in \Pi_{<1,\xi}$ so that

$$(32) \qquad h(x) = \begin{cases} 2\dfrac{|\Delta\varphi_{3/2}|}{\xi_3 - \xi_1} & \text{on } [\xi_1\,..\,\xi_2], \\ \dfrac{|\Delta\varphi_{i-1/2}|}{\xi_{i+1} - \xi_{i-1}} + \dfrac{|\Delta\varphi_{i+1/2}|}{\xi_{i+2} - \xi_i} & \text{on } [\xi_i\,..\,\xi_{i+1}],\ 1 < i < l, \\ 2\dfrac{|\Delta\varphi_{l-1/2}|}{\xi_{l+1} - \xi_{l-1}} & \text{on } [\xi_l\,..\,\xi_{l+1}], \end{cases}$$

where we have used the abbreviations

$$\varphi_{i+1/2} := D^{k-1}f \qquad \text{on } [\xi_i\,..\,\xi_{i+1}], \text{ all } i.$$

This amounts to taking for h on $[\xi_i\,..\,\xi_{i+1}]$ the slope at $(\xi_{i-1} + 3\xi_i + 3\xi_{i+1} + \xi_{i+2})/8$ of the parabola that interpolates the function $\operatorname{Var}_{[a..x]} D^{k-1}f$ at the three sites $\xi_{i-1/2}$, $\xi_{i+1/2}$ and $\xi_{i+3/2}$.

The subroutine NEWNOT The algorithm just outlined is carried out in the following subprogram

```
      SUBROUTINE NEWNOT ( BREAK, COEF, L, K, BRKNEW, LNEW, COEFG )
C  RETURNS  LNEW+1  KNOTS IN  BRKNEW  WHICH ARE EQUIDISTRIBUTED ON (A..B)
C  = (BREAK(1)..BREAK(L+1)) WRTO A CERTAIN MONOTONE FCTN  G  RELATED TO
C  THE K-TH ROOT OF THE K-TH DERIVATIVE OF THE PP FUNCTION  F  WHOSE PP-
C  REPRESENTATION IS CONTAINED IN  BREAK, COEF, L, K .
C
C******  I N P U T ******
C  BREAK, COEF, L, K.....CONTAINS THE PP-REPRESENTATION OF A CERTAIN
C        FUNCTION  F  OF ORDER  K . SPECIFICALLY,
C        D**(K-1)F(X) = COEF(K,I)  FOR  BREAK(I) .LE. X .LT. BREAK(I+1)
C  LNEW.....NUMBER OF INTERVALS INTO WHICH THE INTERVAL (A..B) IS TO BE
C        SECTIONED BY THE NEW BREAKPOINT SEQUENCE  BRKNEW .
C
C******  O U T P U T  ******
C  BRKNEW.....ARRAY OF LENGTH  LNEW+1  CONTAINING THE NEW BREAKPOINT SE-
C        QUENCE
C  COEFG.....THE COEFFICIENT PART OF THE PP-REPR.  BREAK, COEFG, L, 2
C        FOR THE MONOTONE P.LINEAR FUNCTION  G  WRTO WHICH  BRKNEW  WILL
C        BE EQUIDISTRIBUTED.
C
C******  OPTICNAL  P R I N T E D  O U T P U T  ******
C  COEFG.....THE PP COEFFS OF  G  ARE PRINTED OUT IF  IPRINT  IS SET
C        .GT. 0  IN DATA STATEMENT BELOW.
C
```

```
C******  M E T H O D  ******
C     THE K-TH DERIVATIVE OF THE GIVEN PP FUNCTION  F  DOES NOT EXIST
C  (EXCEPT PERHAPS AS A LINEAR COMBINATION OF DELTA FUNCTIONS). NEVER-
C  THELESS, WE CONSTRUCT A P.CONSTANT FUNCTION  H  WITH BREAKPOINT SE-
C  QUENCE  BREAK  WHICH IS APPROXIMATELY PROPORTIONAL TO ABS(D**K(F)).
C  SPECIFICALLY, ON  (BREAK(I)..BREAK(I+1)),
C
C                ABS(JUMP AT BREAK(I) OF PC)          ABS(JUMP AT BREAK(I+1) OF PC)
C  H =  ----------------------------------  +  ------------------------------------
C         BREAK(I+1) - BREAK(I-1)                 BREAK(I+2) - BREAK(I)
C
C
C  WITH  PC  THE P.CONSTANT (K-1)ST DERIVATIVE OF  F .
C     THEN, THE P.LINEAR FUNCTION  G  IS CONSTRUCTED AS
C
C    G(X)  =  INTEGRAL OF  H(Y)**(1/K)  FOR  Y  FROM  A  TO  X
C
C  AND ITS PP COEFFS. STORED IN  COEFG .
C     THEN  BRKNEW  IS DETERMINED BY
C
C        BRKNEW(I)  =  A + G**(-1)((I-1)*STEP) , I=1,...,LNEW+1
C
C  WHERE  STEP = G(B)/LNEW  AND  (A .. B) = (BREAK(1) .. BREAK(L+1)) .
C     IN THE EVENT THAT  PC = D**(K-1)(F) IS CONSTANT IN  (A .. B)  AND
C  THEREFORE  H = 0 IDENTICALLY,  BRKNEW  IS CHOSEN UNIFORMLY SPACED.
C
      INTEGER K,L,LNEW,   I,IPRINT,J
      REAL BREAK(L+1),BRKNEW(LNEW+1),COEF(K,L),COEFG(2,L),   DIF,DIFPRV
     *                                                ,ONEOVK,STEP,STEPI
      DATA IPRINT /0/
C
      BRKNEW(1) = BREAK(1)
      BRKNEW(LNEW+1) = BREAK(L+1)
C                                  IF  G  IS CONSTANT,  BRKNEW  IS UNIFORM.
      IF (L .LE. 1)                 GO TO 90
C                           CONSTRUCT THE CONTINUOUS P.LINEAR FUNCTION  G .
      ONEOVK = 1./FLOAT(K)
      COEFG(1,1) = 0.
      DIFPRV = ABS(COEF(K,2) - COEF(K,1))/(BREAK(3)-BREAK(1))
      DO 10 I=2,L
         DIF = ABS(COEF(K,I) - COEF(K,I-1))/(BREAK(I+1) - BREAK(I-1))
         COEFG(2,I-1) = (DIF + DIFPRV)**ONEOVK
         COEFG(1,I) = COEFG(1,I-1)+COEFG(2,I-1)*(BREAK(I)-BREAK(I-1))
   10    DIFPRV = DIF
      COEFG(2,L) = (2.*DIFPRV)**ONEOVK
C                                                 STEP  =  G(B)/LNEW
      STEP = (COEFG(1,L)+COEFG(2,L)*(BREAK(L+1)-BREAK(L)))/FLOAT(LNEW)
C
      IF (IPRINT .GT. 0) PRINT 600, STEP,(I,COEFG(1,I),COEFG(2,I),I=1,L)
  600 FORMAT(7H STEP =,E16.7/(I5,2E16.5))
C                                  IF  G  IS CONSTANT,  BRKNEW  IS UNIFORM .
      IF (STEP .LE. 0.)             GO TO 90
C
C     FOR I=2,...,LNEW, CONSTRUCT  BRKNEW(I) = A + G**(-1)(STEPI),
C     WITH  STEPI = (I-1)*STEP .  THIS REQUIRES INVERSION OF THE P.LIN-
C     EAR FUNCTION  G .  FOR THIS,  J  IS FOUND SO THAT
C        G(BREAK(J)) .LE. STEPI .LE. G(BREAK(J+1))
C     AND THEN
C        BRKNEW(I)  =  BREAK(J) + (STEPI-G(BREAK(J)))/DG(BREAK(J)) .
C     THE MIDPOINT IS CHOSEN IF  DG(BREAK(J)) = 0 .
```

```
      J = 1
      DO 30 I=2,LNEW
         STEPI = FLOAT(I-1)*STEP
21          IF (J .EQ. L)                    GO TO 27
            IF (STEPI .LE. COEFG(1,J+1))GO TO 27
            J = J + 1
                                             GO TO 21
27       IF (COEFG(2,J) .EQ. 0.)             GO TO 29
            BRKNEW(I) = BREAK(J) + (STEPI - COEFG(1,J))/COEFG(2,J)
                                             GO TO 30
29          BRKNEW(I) = (BREAK(J) + BREAK(J+1))/2.
30       CONTINUE
                                          RETURN
C
C                                  IF  G  IS CONSTANT,  BRKNEW  IS UNIFORM .
      90 STEP = (BREAK(L+1) - BREAK(1))/FLOAT(LNEW)
         DO 93 I=2,LNEW
      93    BRKNEW(I) = BREAK(1) + FLOAT(I-1)*STEP
                                          RETURN
         END
```

(33) Example: A failure for NEWNOT We will give examples in later chapters showing the effectiveness of knot placement algorithms such as the one encoded in NEWNOT. So we feel safe to begin here with an example where, somewhat surprisingly, NEWNOT produces only marginal improvements, even though the function to be approximated is not uniform.

We construct our approximation to the function $g(x) := \sqrt{x+1}$ on $[-1\,..\,1]$ by cubic spline interpolation, using the routine CUBSPL of Chapter IV with the "not-a-knot" end condition. The main program below is an adaption of the program used in II(1) and IV(8). The initial construction of an interpolant on a uniform knot sequence to N data points is followed here by a cycle: the current interpolant is used in NEWNOT to construct a possibly better knot distribution and then, with these new knots, a new cubic spline interpolant (at those knots) in constructed in CUBSPL. The number of times this cycle is repeated is specified by the input parameter ITERMX.

```
CHAPTER XII, EXAMPLE 2. CUBIC SPLINE INTERPOLATION WITH GOOD KNOTS
CALLS  CUBSPL, NEWNOT
      INTEGER I,ISTEP,ITER,ITERMX,J,N,NHIGH,NLOW,NMAX,NM1
      PARAMETER (NMAX=20)
      REAL ALGERP,ALOGER,DECAY,DX,ERRMAX,C(4,NMAX),G,H,PNATX
     *    ,SCRTCH(2,NMAX),STEP,TAU(NMAX),TAUNEW(NMAX)
C     ISTEP AND STEP = FLOAT(ISTEP) SPECIFY POINT DENSITY FOR ERROR DET-
C     ERMINATION.
      DATA STEP, ISTEP /20., 20/
C                                    THE FUNCTION  G  IS TO BE INTERPOLATED .
      G(X) = SQRT(X+1.)
      DECAY = 0.
C      READ IN THE NUMBER OF ITERATIONS TO BE CARRIED OUT AND THE LOWER
C      AND UPPER LIMIT FOR THE NUMBER  N  OF DATA POINTS TO BE USED.
```

```
      READ 500,ITERMX,NLOW,NHIGH
  500 FORMAT(3I3)
      PRINT 600, ITERMX
  600 FORMAT(I4,22H CYCLES THROUGH NEWNOT//
     *       28H  N   MAX.ERROR   DECAY EXP./)
                                          LOOP OVER  N = NUMBER OF DATA POINTS .
C
      DO 40 N=NLOW,NHIGH,2
                                                KNOTS ARE INITIALLY EQUISPACED.
C
          NM1 = N-1
          H = 2./FLOAT(NM1)
          DO 10 I=1,N
     10      TAU(I) = FLOAT(I-1)*H - 1.
          ITER = 1
C               CONSTRUCT CUBIC SPLINE INTERPOLANT. THEN, ITERMX  TIMES,
C                DETERMINE NEW KNOTS FROM IT AND FIND A NEW INTERPOLANT.
     11       DO 15 I=1,N
     15          C(1,I) = G(TAU(I))
              CALL CUBSPL ( TAU, C, N, 0, 0 )
              IF (ITER .GT. ITERMX)        GO TO 19
              ITER = ITER+1
              CALL NEWNOT(TAU,C,NM1,4,TAUNEW,NM1,SCRTCH)
              DO 18 I=1,N
     18          TAU(I) = TAUNEW(I)
                                           GO TO 11

     19   CONTINUE
                                ESTIMATE MAX.INTERPOLATION ERROR ON (-1..1).
C
          ERRMAX = 0.
                                LOOP OVER POLYNOMIAL PIECES OF INTERPOLANT .
C
          DO 30 I=1,NM1
             DX = (TAU(I+1)-TAU(I))/STEP
C               INTERPOLATION ERROR IS CALCULATED AT  ISTEP  POINTS PER
C                   POLYNOMIAL PIECE .
              DO 30 J=1,ISTEP
                 H = FLOAT(J)*DX
                 PNATX = C(1,I)+H*(C(2,I)+H*(C(3,I)+H*C(4,I)/3.)/2.)
     30          ERRMAX = AMAX1(ERRMAX,ABS(G(TAU(I)+H)-PNATX))
                                                  CALCULATE DECAY EXPONENT .
C
          ALOGER = ALOG(ERRMAX)
          IF (N .GT. NLOW)  DECAY =
     *        (ALOGER - ALGERP)/ALOG(FLOAT(N)/FLOAT(N-2))
          ALGERP = ALOGER
C
     40   PRINT 640,N,ERRMAX,DECAY
    640 FORMAT(I3,E12.4,F11.2)
                                                 STOP

          END
```

0 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.1114+00	-0.69
8	0.9414-01	-0.59
10	0.8303-01	-0.56
12	0.7510-01	-0.55
14	0.6908-01	-0.54
16	0.6431-01	-0.54
18	0.6041-01	-0.53
20	0.5714-01	-0.53

3 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1461+00	0.00
6	0.1273+00	-0.34
8	0.1068+00	-0.61
10	0.9175-01	-0.68
12	0.8171-01	-0.64
14	0.7197-01	-0.82
16	0.6568-01	-0.69
18	0.6038-01	-0.71
20	0.5683-01	-0.58

The output shows the maximum interpolation error as a function of the number of data points, both for uniformly spaced data sites and when, for each N, NEWNOT was used three times to achieve a better knot distribution. For the uniformly spaced case, the decay exponent is about $-.5$, in accordance with (2) above (recall from Figure II(17) that, for the function $g(x) = \sqrt{x+1}$ on $[-1\,..\,1]$, the modulus of continuity is $\omega(g;h) = h^{1/2}$). But, even after three applications of NEWNOT, the decay exponent is still only around $-.6$ whereas I had expected it, because of Theorem (34) below, to be about $-.4$, the typical exponent for fourth order, or cubic, approximation. In fact, for all $N > 4$ tried, the approximation with the "good" knots is slightly worse than that with the uniform knots.

It is instructive to contemplate the reason for this failure (which was not apparent to me right away). NEWNOT relies essentially on the jumps in the third derivative of the cubic interpolant to gauge the size of the fourth derivative of the function g to be approximated, with a relatively large jump calling for relatively many knots to be placed near that jump. In our example, the fourth derivative of the function g grows large near -1 and it is there that one would wish knots to be placed. Yet, on closer inspection, it turns out that NEWNOT failed to do this. The reason: the "not-a-knot" end condition used in CUBSPL explicitly forces the first and the last potential jumps in the third derivative to be zero, and this makes h of (32) *vanish* between the first and the second (and between the second last and the last) data sites. Therefore NEWNOT puts all breaks well away from the ends of the interval. The best laid plans . . . .

The distance from $\$_{k,n}$ We consider now briefly just how well a function can be approximated by splines of a fixed *number* of knots if these knots are placed appropriately. This means that we are interested in the number

$$\operatorname{dist}(g, \$_{k,n})$$

where $\$_{k,n}$ denotes the collection of all splines of order k with *some* knot sequence $\mathbf{t} = (t_i)_1^{n+k}$ with $t_1 = \cdots = t_k = a$, $t_{n+1} = \cdots = t_{n+k} = b$, and with n fixed.

The above arguments that led us to the knot placement algorithm above were derived from considerations by Rice [1969], Burchard [1974], Dodson [1972] and de Boor [1973] concerning $\operatorname{dist}(g, \$_{k,n})$. These considerations produced the following theorem (among others).

(34) Theorem. *Assume that the function g is continuous on $[a\,..\,b]$ and is k times continuously differentiable at all but finitely many sites in $[a\,..\,b]$ near which $|D^k g|$ is monotone, and the kth root of $D^k g$ is integrable, that is,*

$$\int_a^b |D^k g(x)|^{1/k}\,\mathrm{d}x < \infty.$$

Then

$$\text{dist}\,(g, \$_{k,n}) \le \text{const}_k n^{-k} \left(\int_a^b |D^k g(x)|^{1/k}\,\mathrm{d}x \right)^k, \tag{35}$$

that is, the order of approximation achievable is n^{-k}. This follows from the fact that, for the choice $\zeta := (\zeta_i)_1^{m+1}$ of break sequence so that

$$\int_{\zeta_i}^{\zeta_{i+1}} |D^k g(x)|^{1/k}\,\mathrm{d}x = \frac{1}{m} \int_a^b |D^k g(x)|^{1/k}\,\mathrm{d}x, \quad i = 1, \ldots, m, \tag{31}$$

we have then

$$\text{dist}\,(g, \Pi_{<k,\zeta,1}) \le \text{const}_k m^{-k} \left(\int_a^b |D^k g(x)|^{1/k}\,\mathrm{d}x \right)^k. \tag{36}$$

Theorem III(20) concerning best broken line approximation is a special case of this theorem. We can gauge its power by comparing it with Theorem (6) which states that

$$\text{dist}\,(g, \$_{k,\mathbf{t}}) \le \text{const}_k |\mathbf{t}|^k \|D^k g\|_\infty$$

in case g has k continuous derivatives on $[a\,..\,b]$. If we choose here the knot sequence $\mathbf{t}$ uniform inside $[a\,..\,b]$ (so as to make the bound as small as possible for fixed n), that is, $t_{k+i} = a + ih$, $i = 1, \ldots, n-k$, with $h := (b-a)/(n-k+1)$, we find that

$$\text{dist}\,(g, \$_{k,\mathbf{t}}) = \mathcal{O}\left(n^{-k}\right).$$

But let the kth derivative of g become infinite anywhere in $[a\,..\,b]$, and Theorem (6) ceases to produce such rates of approximation. For the Example (33) of approximation to $g(x) = \sqrt{x+1}$ on $[a\,..\,b] = [-1\,..\,1]$, for instance, already the first derivative becomes infinite and so, for uniform knot sequence, Theorem (6) produces the estimate

$$\text{dist}\,(g, \$_{k,\mathbf{t}}) = \mathcal{O}\left(n^{-1/2}\right)$$

which is sharp as we saw in the example. Nevertheless, when we compute the kth derivative of g, we find

$$|D^k g(x)| = \frac{1}{2}\,\frac{1}{2}\,\frac{3}{2} \cdots \frac{2k-3}{2}\,(x+1)^{-(2k-1)/2}$$

and therefore

$$|D^k g(x)|^{1/k} = \text{const}_k (x+1)^{-(2k-1)/(2k)}$$

which is indeed integrable, that is,

$$\int_{-1}^{1} |D^k g(x)|^{1/k}\,\mathrm{d}x = 2k\,\mathrm{const}_k\,(x+1)^{1/(2k)}\Big|_{-1}^{1} = 2k\,\mathrm{const}_k 2^{1/(2k)}$$

which is finite. We would therefore expect to have $\mathcal{O}\left(n^{-k}\right)$-convergence to this very unsmooth function $g(x) = \sqrt{x+1}$ on $[-1\,..\,1]$ provided we choose the knots right. Theorem (34) indicates (as does the earlier discussion of the knot placement algorithm) that we should choose the knot sequence according to the break sequence $\boldsymbol{\zeta}$ for which (31) holds. This we carry out in the next example.

(37) Example: A failure for CUBSPL We continue our attempts to approximate the function $g(x) = \sqrt{x+1}$ on $[a\,..\,b] = [-1\,..\,1]$ well by means of cubic spline interpolation at knots with the not-a-knot end conditions and using N data points. But this time, we choose those N data sites explicitly so that the resulting break sequence of our interpolating cubic spline satisfies (31). Since

$$\int_{-1}^{x} |D^4 g(s)|^{1/4}\,\mathrm{d}s = \mathrm{const}(x+1)^{1/8} \quad \text{for } x \geq -1,$$

this amounts to choosing the N interpolation sites as

$$\tau_i = 2\big((i-1)/(N-1)\big)^3 - 1, \qquad i = 1,\ldots,N. \tag{38}$$

The modification of the program for Example (33) to use such data sites and to make no use of NEWNOT produces the following output

N	MAX.ERROR	DECAY EXP.
4	0.3834+03	0.00
6	0.1284+03	-2.70
8	0.4572+02	-3.59
10	0.1138+02	-6.23
12	0.0000+00	-INF
14	0.0000+00	NaN

What a disappointment!

The "INF " we can understand easily enough. For $N = 12$, we have $\tau_2 - \tau_1 = 2(1/11)^8 = 4.7 * 10^{-9} = 0$ in single precision on a UNIVAC 1110. But an error of 383 for $N = 4$ and of 11 even for $N = 10$ is disappointing when the function we wish to approximate lies between 0 and 2. It may well be (and we would need double precision calculations to verify that) that, with the good knots, the error decreases like N^{-4}, but it seems to start off so much larger than with the uniformly spaced data sites, it seems

hardly worth the effort.

In order to make certain that the program is correct, I also ran it with the following more uniform interpolation sites

$$\tau_i = 2\big((i-1)/(N-1)\big)^4 - 1, \quad \text{and} \quad \tau_i = 2\big((i-1)/(N-1)\big)^6 - 1.$$

This produced the following output.

N	MAX.ERROR	DECAY EXP.
4	0.4882+02	0.00
6	0.1179+02	-3.50
8	0.2898+01	-4.88
10	0.5748+00	-7.25
12	0.1381+00	-7.82
14	0.8346-01	-3.27
16	0.5395-01	-3.27
18	0.4629-01	-1.30
20	0.2145-01	-7.30

N	MAX.ERROR	DECAY EXP.
4	0.5001+01	0.00
6	0.7842+00	-4.57
8	0.1217+00	-6.48
10	0.5025-01	-3.96
12	0.3364-01	-2.20
14	0.2408-01	-2.17
16	0.1809-01	-2.14
18	0.1408-01	-2.13
20	0.1129-01	-2.09

Both show an improvement for larger N compared to uniform spacing, although the point of improvement comes later as the mesh becomes more nonuniform. They show a decay exponent of -2 and -3, respectively. (An analysis of the proof for Theorem (34) would lead one to expect exactly such rates in these cases.)

In order to explain the difficulty with the data sites (38), we need the following error bound (see Problem III.2). If we denote the cubic spline interpolant to the function g by $I_4 g$ (as we did in Chapter IV), then we can show that

$$\|g - I_4 g\| \le (1 + \|I_4\|)\,\mathrm{dist}\,(g, \$_4) \tag{39}$$

with $\$_4$ the collection of all cubic splines with knots $\tau_2, \ldots, \tau_{N-1}$ inside of $[-1\,..\,1]$ and

$$\|I_4\| := \max\{\, \|I_4 f\|/\|f\| : f \in C[-1\,..\,1], \|f\| \neq 0 \,\}$$

the *norm* of the interpolation process I_4. A proof of (39) goes as follows: For any $f \in \$_4$, $I_4 f = f$, therefore $g - I_4 g = (g-f) - I_4(g-f)$ for any $f \in \$_4$, hence

$$\|g - I_4 g\| \le \|g - f\| + \|I_4(g-f)\| \le \|g-f\| + \|I_4\|\,\|g-f\|.$$

Now we choose $f \in \$_4$ to make the right hand side as small as possible; this produces (39).

The particular choice of the data sites enters the bound (39) in two ways:

(i) It influences the approximation power of the function class I_4 since it determines the knot locations. In our case, we can expect dist $(g, \$_4)$ to

become smaller (for fixed N) as we come closer to the data sites (38) that are "right" for our particular g.

(ii) It influences the norm of the approximation process I_4. As the data site sequence becomes more nonuniform, it so happens that the number $\|I_4\|$ gets larger and larger. Of course, this does not *guarantee* that the interpolation error will also get larger since (39) is, after all, only a bound. But it explains why it gets larger when it does. (See the case of uniform data sites in polynomial interpolation in Example II(1).)

Connected with this is a further reason for the failure on Example (37), namely the fact that, as $\|I_4\|$ gets larger, the approximation process becomes less and less *local*. By this we mean that the error at a site may well depend on the behavior of the function g anywhere in $[a\,..\,b]$. But, as soon as our approximation scheme has this property, then it makes no sense anymore to adapt a knot sequence to the local behavior of the function. In other words, knot placement based on the local behavior of the function to be approximated makes sense only if we use an approximation process A that is more or less local, that is, for which we get, at least approximately, an estimate of the form (28).

(40) Example: Knot placement works when used with a local approximation scheme To illustrate the point just made, we replace the cubic spline interpolant at knots in Example (37) by the quasi-interpolant of Theorem (26) which we know to be a local approximant. We choose the same knot sequence used implicitly in that example, that is, we use $\mathrm{t} = (t_i)_1^{m+k}$ with $k = 4$, $m = N + 2$ and

$$t_1 = \cdots = t_4 = -1, \quad t_{4+i} = 2\big(i/(N-1)\big)^8 - 1, \quad i = 1, \ldots, N-2,$$
$$t_{m+1} = \cdots = t_{m+4} = 1.$$

The approximation Qg is then computed as

$$Qg = \sum_{i=1}^{m} (\lambda_i g)\mathrm{B}_i$$

with $(\mathrm{B}_i)_1^m$ the sequence of B-splines of order 4 for t and $\lambda_i g$, from IX(54), computed as

$$\lambda_i g = \sum_{j<4} (-)^{3-j} D^{3-j} \psi_{i,4}(\tau_i)\big(D^j g\big)(\tau_i)$$

with $\psi_{i,4}(x) := (t_{i+1} - x)(t_{i+2} - x)(t_{i+3} - x)/6$. If we choose $\tau_i = t_{i+2}$, all i, then, as already used in Example IX(58) (see IX(59)),

$$(41)\quad \lambda_i g = g(\tau_i) + \frac{1}{3}(\Delta t_{i+2} - \Delta t_{i+1})g'(t_{i+2}) - \frac{1}{3}\Delta t_{i+1}\Delta t_{i+2} g''(t_{i+2})/2.$$

For $i = 1$, this gives that

$$\lambda_1 g = g(-1) = 0.$$

For $i = 2$, this would give $\lambda_2 g = \infty$. For this reason, we use instead $\tau_2 = t_5$. This gives

$$\lambda_2 g = g(t_5) - 2\Delta t_4 g'(t_5)/3 + (\Delta t_4)^2 g''(t_5)/6.$$

```
CHAPTER XII, EXAMPLE 4. QUASI-INTERPOLANT WITH GOOD KNOTS.
CALLS BSPLPP(BSPLVB)
C
      INTEGER I,IRATE,ISTEP,J,L,M,MMK,N,NLOW,NHIGH,NM1
      REAL ALGERP,ALOGER,BCOEF(22),BREAK(20),C(4,20),DECAY,DG,DDG
     *    ,DTIP1,DTIP2,DX,ERRMAX,G,H,PNATX,SCRTCH(4,4),STEP,T(26),TAUI
C     ISTEP AND STEP = FLOAT(ISTEP) SPECIFY POINT DENSITY FOR ERROR DET-
C     ERMINATION.
      DATA STEP, ISTEP /20., 20/
C        G  IS THE FUNCTION TO BE APPROXIMATED,  DG  IS ITS FIRST, AND
C        DDG  ITS SECOND DERIVATIVE .
      G(X) = SQRT(X+1.)
      DG(X) = .5/G(X)
      DDG(X) = -.5*DG(X)/(X+1.)
      DECAY = 0.
C      READ IN THE EXPONENT  IRATE  FOR THE KNOT DISTRIBUTION AND THE
C      LOWER AND UPPER LIMIT FOR THE NUMBER  N  .
      READ 500,IRATE,NLOW,NHIGH
  500 FORMAT(3I3)
      PRINT 600
  600 FORMAT(28H  N   MAX.ERROR   DECAY EXP./)
C                                    LOOP OVER  N = DIM( SPLINE(4,T) ) - 2 .
C             N  IS CHOSEN AS THE PARAMETER IN ORDER TO AFFORD COMPAR-
C             ISON WITH EXAMPLES 2 AND 3  IN WHICH CUBIC SPLINE INTERP-
C             OLATION AT  N  DATA POINTS WAS USED .
      DO 40 N=NLOW,NHIGH,2
         NM1 = N-1
         H = 1./FLOAT(NM1)
         M = N+2
         MMK = M-4
         DO 5 I=1,4
            T(I) = -1.
    5       T(M+I) = 1.
C                INTERIOR KNOTS ARE EQUIDISTRIBUTED WITH RESPECT TO THE
C                FUNCTION  (X + 1)**(1/IRATE) .
         DO 6 I=1,MMK
    6       T(I+4) = 2.*(FLOAT(I)*H)**IRATE - 1.
C                                                 CONSTRUCT QUASI-INTERPOLANT.
C                                                      BCOEF(1) = G(-1.) = 0.
         BCOEF(1) = 0.
         DTIP2 = T(5) - T(4)
         TAUI = T(5)
C                              SPECIAL CHOICE OF  TAU(2)  TO AVOID INFINITE
C                              DERIVATIVES OF  G  AT LEFT ENDPOINT .
         BCOEF(2) = G(TAUI) - 2.*DTIP2*DG(TAUI)/3.
     *             + DTIP2**2*DDG(TAUI)/6.
         DO 15 I=3,M
            TAUI = T(I+2)
            DTIP1 = DTIP2
            DTIP2 = T(I+3) - T(I+2)
C                                          FORMULA XII(30) OF TEXT IS USED .
   15       BCOEF(I) = G(TAUI) + (DTIP2-DTIP1)*DG(TAUI)/3.
     *                - DTIP1*DTIP2*DDG(TAUI)/6.
C                                          CONVERT TO PP-REPRESENTATION .
```

```
      CALL BSPLPP(T,BCOEF,M,4,SCRTCH,BREAK,C,L)
C                              ESTIMATE MAX.INTERPOLATION ERROR ON (-1..1).
      ERRMAX = 0.
C                                                  LOOP OVER CUBIC PIECES ...
      DO 30 I=1,L
         DX = (BREAK(I+1)-BREAK(I))/STEP
C                       ERROR IS CALCULATED AT  ISTEP  POINTS PER PIECE.
         DO 30 J=1,ISTEP
            H = FLOAT(J)*DX
            PNATX = C(1,I)+H*(C(2,I)+H*(C(3,I)+H*C(4,I)/3.)/2.)
   30       ERRMAX = AMAX1(ERRMAX,ABS(G(BREAK(I)+H)-PNATX))
C                                                  CALCULATE DECAY EXPONENT .
      ALOGER = ALOG(ERRMAX)
      IF (N .GT. NLOW)  DECAY =
     *     (ALOGER - ALGERP)/ALOG(FLOAT(N)/FLOAT(N-2))
      ALGERP = ALOGER
C
   40 PRINT 640,N,ERRMAX,DECAY
  640 FORMAT(I3,E12.4,F11.2)
                                                  STOP
      END
```

The output for

$$\tau_i = 2\big((i-1)/(N-1)\big)^6 - 1 \quad \text{and} \quad \tau_i = 2\big((i-1)/(N-1)\big)^8 - 1$$

is as follows:

N	MAX.ERROR	DECAY EXP.	N	MAX.ERROR	DECAY EXP.
4	0.7240+00	0.00	4	0.4128+00	0.00
6	0.9952-01	-4.89	6	0.8938-01	-3.77
8	0.3031-01	-4.13	8	0.3258-01	-3.51
10	0.1197-01	-4.17	10	0.1535-01	-3.37
12	0.5578-02	-4.19	12	0.8372-02	-3.32
14	0.2940-02	-4.16	14	0.5060-02	-3.27
16	0.1688-02	-4.15	16	0.3281-02	-3.24
18	0.1035-02	-4.16	18	0.2856-02	-1.18
20	0.6685-03	-4.15	20	0.1289-02	-7.55

It shows clearly the $\mathcal{O}\left(N^{-4}\right)$ convergence for the correct knots (before underflow puts an end to it). Also, for the slightly less nonuniform knots, we get $\mathcal{O}\left(N^{-3}\right)$ convergence and an impressive improvement over uniform knots.

Problems

In the following three problems, let A be the approximation map given by (13).

1. Verify the assertion in the text that A is linear (if each μ_i is).

2. Prove: If $\mu_i g = \sum_{j=1}^{k} \beta_{ij} g(\tau_{ij})$ and $\tau_{ij} \in [t_{i+1} \,..\, t_{i+k-1}]$, all j, and $Ag = g$ (on $[a \,..\, b]$) for every constant function g, then $\sum_{j=1}^{k} \beta_{ij} = 1$. Conclude that, if also $k = 2$, then necessarily $\mu_i g = [t_{i+1}]g$, all i, and A is

broken line interpolation.

3. Prove: If A reproduces $\Pi_{<k}$ and, for each i, μ_i has its support in some interval $[t_j \,..\, t_{j+1}]$ contained in $[t_i \,..\, t_{i+k}]$, then A reproduces all of $\$_{k,\mathbf{t}}$.

4. Use the fact that $[\hat{x}, \hat{x}, x]g = (g'(x) - [\hat{x}, x]g)/(\hat{x} - x)$ (see I(vii)) and that $[\hat{x}, x]g = g'(\xi)$ for some ξ between $\hat{x}$ and x to show that

$$\|g - Vg\| \leq \text{const}_k \; |\mathbf{t}|\omega(g'; |\mathbf{t}|) \qquad \text{for} \quad g \in C^{(1)}[a \,..\, b].$$

5. Cubic spline interpolation (as discussed in Chapters IV and V) is not local, yet it is possible to construct interpolating cubic splines that are local. The trick is to use additional knots.

Let $(\tau_i)_1^n$ be a sequence of data sites. For each i, construct a cubic spline C_i with simple knots at the τ_j's and, possibly, also at the sites $\tau_{j+1/2} := (\tau_j + \tau_{j+1})/2$, so that (i) $C_i(\tau_j) = \delta_{ij}$, (ii) $C_i(x) = 0$ for $x \notin (\tau_{i-3} \,..\, \tau_{i+3})$, (iii) $\sum_i p(\tau_i)C_i = p$ for all cubic polynomials p (on $[a \,..\, b]$).

(Ignore the additional difficulties for $i = 1, 2$ and $n - 1, n$ if these prove too tricky.)

6. Prove the following fact of use in the analysis of finite element approximation: If $g = hf$, with $f \in \$_{k,\mathbf{t}}$ and h a continuous function, then $\text{dist}\,(g, \$_{k,\mathbf{t}}) \leq \text{const}_k \; \omega(h; |\mathbf{t}|)\|f\|$. (Hint: Use (27))).

7. Theorem (26) establishes the existence of an approximation $Qg \in \$_{k,\mathbf{t}}$ to g that simultaneously approximates all derivatives of order $< k$ of g to optimal order, but, for the higher order derivatives, the error bound involves the local mesh ratio $m_{\mathbf{t}}$. Prove the existence of an approximation $\hat{Q}g \in \$_{k,\mathbf{t}}$ to g for which

$$\|g^{(j)} - D^j\hat{Q}g\| \leq \text{const}_{k,j} \; |\mathbf{t}|^{k-j-1}\omega(g^{(k-1)}; |\mathbf{t}|), \qquad \text{for} \quad j < k,$$

that is, independently of $m_{\mathbf{t}}$. (Hint: Show that any strictly increasing sequence $t_k < \cdots < t_{n+1}$ contains a subsequence $t_k = \hat{t}_k < \cdots < \hat{t}_{\hat{n}+1} = t_{n+1}$ for which $m_{\hat{\mathbf{t}}} \leq 4$ and $|\hat{\mathbf{t}}| \leq 2|\mathbf{t}|$. For this, set $h := |\mathbf{t}|$ and, with $\hat{t}_i$ already picked, choose $\hat{t}_{i+1} = t_r$ with $h/2 \leq t_r - \hat{t}_i < 3h/2$; etc.)

8. Theorem (26) as stated applies only to splines with simple knots, as far as the approximation to higher derivatives is concerned.

Use Problem 7 to show that the conclusion of Problem 7 holds for an arbitrary knot sequence $\mathbf{t}$. (Hint: If $\mathbf{t}$ has multiple (interior) knots, approximate $\mathbf{t}$ by a sequence $\hat{\mathbf{t}}$ with simple knots; then use the fact that the error bound in Problem 7 depends only on $|\mathbf{t}|$ and not on $\mathbf{t}$ to let $\hat{\mathbf{t}}$ approach $\mathbf{t}$.)

9. Modify CUBSPL to treat the not-a-knot end condition in the alternative way outlined on p. 45, that is, without ever making the second and the second last data sites breaks. Then repeat the calculations of Example (33) with this modified CUBSPL.

XIII

Spline Interpolation;
SPLINT, SPLOPT

In this chapter, we discuss spline interpolation of arbitrary order and point out that one has to monitor the interplay between interpolation sites and knots carefully in order to produce a reasonable interpolation scheme. We discuss also a particular choice of knots for given data sites with which spline interpolation becomes the optimal recovery scheme of Michelli, Rivlin and Winograd. Finally, we mention osculatory interpolation and complete even-order spline interpolation and its best approximation properties.

Throughout this chapter, $\mathbf{t} = (t_i)_1^{n+k}$ is a nondecreasing knot sequence, with $t_i < t_{i+k}$, all i, and $(\mathrm{B}_i)_1^n$ is the corresponding sequence of B-splines of order k.

The Schoenberg-Whitney Theorem Since the sequence $(\mathrm{B}_i)_1^n$ is linearly independent, its linear span, that is, the space $\$:= \$_{k,\mathbf{t}}$, is n-dimensional. We therefore expect to accommodate n interpolation conditions. If the strictly increasing sequence $\tau = (\tau_i)_1^n$ of data sites is given somehow, then, for given function g, the spline $f := \sum_1^n \alpha_j \mathrm{B}_j$ agrees with g at τ if and only if

$$\sum_{j=1}^{n} \alpha_j \mathrm{B}_j(\tau_i) = g(\tau_i), \qquad i = 1, \ldots, n. \tag{1}$$

This is a linear system of n equations in the n-vector $\boldsymbol{\alpha} := (\alpha_i)_1^n$ of n unknowns, with coefficient matrix $\big(\mathrm{B}_j(\tau_i)\big)$, the **spline collocation matrix.** As it turns out, it is very easy to check whether this matrix is invertible, that is, whether (1) has exactly one solution for every choice of g.

(2) Theorem (Schoenberg–Whitney). *Let τ be strictly increasing and such that $a < t_i = \cdots = t_{i+r} = \tau_j < b$ implies $r < k-1$. Then the matrix $A := \big(\mathrm{B}_j(\tau_i)\big)$ of the linear system (1) is invertible if and only if*

$$\mathrm{B}_i(\tau_i) \neq 0, \qquad i = 1, \ldots, n, \tag{3}$$

that is, if and only if $t_i < \tau_i < t_{i+k}$, *all* i *(except that* $\tau_1 = t_1^+$ *and* $\tau_n = t_{n+k}^-$ *are also permitted).*

The original theorem by Schoenberg & Whitney [1953] involves truncated power functions rather than B-splines. An elementary proof of the present theorem, using nothing more than Rolle's Theorem (that is, the fact that between any two zeros of a differentiable function must lie a zero of its derivative), can be found in de Boor [1976]$_1$; see also Problem 1.

Actually, one direction is quite simple: If for some i, $t_{i+k} \le \tau_i$, then the first i columns of the matrix A have nonzero entries only in the first $i-1$ rows, hence this sequence of columns must be linearly dependent, and so, A cannot be invertible. Again, if $\tau_i \le t_i$, then columns $i, \ldots, n$ of A have nonzero entries only in rows $i+1, \ldots, n$, hence A cannot be invertible. This argument uses nothing more than the fact that both the sequence τ and the sequence of the supports of the B-splines are increasing, hence we have proved the following.

(4) **Proposition.** *If the matrix* $(\mathrm{B}_{m_j,k}(s_i) : i,j = 1, \ldots, r)$ *is invertible with both* (m_j) *and* (s_i) *increasing, then* $\mathrm{B}_{m_i,k}(s_i) \neq 0$, *all* i.

For the proof of the converse, we make use of an upper bound on the number of 'isolated' zeros a spline from \$ can have, the bound being easily provided using knot insertion. Here, we call z an **'isolated' zero** of $f \in \$_{k,\mathbf{t}}$ in case $f(z) = 0$ while

$$f^{\mathbf{t}} := \sum_j |\alpha_j| \mathrm{B}_{j,k,\mathbf{t}} \tag{5}$$

does not vanish at z. Note the use of the superscript $\mathbf{t}$ to indicate the dependence of $f^{\mathbf{t}}$ on the particular knot sequence we use to represent the spline f as $f =: \sum_j \alpha_j \mathrm{B}_j$. The importance of the function $f^{\mathbf{t}}$ in considerations of zeros of splines seems to have been first recognized by T. N. T. Goodman [1994]. For the definition of the number $\mathrm{S}^- \boldsymbol{\alpha}$ of strong sign changes in the sequence $\boldsymbol{\alpha}$, used in the next statement, see p. 138 (in Chapter XI).

(6) **Proposition.** *If* $f = \sum_j \alpha_j \mathrm{B}_{j,k,\mathbf{t}}$ *vanishes at* $\boldsymbol{\tau} = (\tau_1 < \cdots < \tau_r)$ *while* $f^{\mathbf{t}} = \sum_j |\alpha_j| \mathrm{B}_{j,k,\mathbf{t}}$ *is positive there, then* $\mathrm{S}^- \boldsymbol{\alpha} > r$ -- 1.

PROOF. Since $f^{\mathbf{t}}(\tau_i) > 0$ while $f(\tau_i) = 0$, the sequence $(\alpha_j \mathrm{B}_j(\tau_i) : \mathrm{B}_j(\tau_i) \neq 0)$ must have at least one strong sign change, hence, so must the sequence $(\alpha_j : \mathrm{B}_j(\tau_i) \neq 0)$. This gives altogether r strong sign changes in $\boldsymbol{\alpha}$ provided we can be sure that different τ_i generate different sign changes. Off-hand, this may not be so, but it can be guaranteed by inserting each of the sites $\tau_{i+1/2} = (\tau_i + \tau_{i+1})/2$, $i = 1, \ldots, r-1$, into the knot sequence k times. If $\widetilde{\mathbf{t}}$ and $\widetilde{\boldsymbol{\alpha}}$ are the resulting knot and coefficient sequence, respectively, then still $f^{\widetilde{\mathbf{t}}}(\tau_i) > 0$ (since, from $f^{\mathbf{t}}(\tau_i) > 0$ we know that τ_i is an isolated zero of f; see Problem 9), while now

$\{j : \widetilde{\mathrm{B}}_j(\tau_i) \neq 0\} \cap \{j : \widetilde{\mathrm{B}}_j(\tau_h) \neq 0\} = \emptyset$ for $i \neq h$, hence, with Lemma XI(27),

$$\mathrm{S}^- \boldsymbol{\alpha} \geq \mathrm{S}^- \widetilde{\boldsymbol{\alpha}} \geq r.$$

⊏

PROOF OF THE SCHOENBERG-WHITNEY THEOREM. We only need to prove the 'if'. Since A is square, it is sufficient to prove that $A\boldsymbol{\alpha} = \mathbf{0}$ implies $\boldsymbol{\alpha} = \mathbf{0}$. Consider $f := \sum_j \alpha_j \mathrm{B}_j$ with $A\boldsymbol{\alpha} = \mathbf{0}$. If $\boldsymbol{\alpha} \neq \mathbf{0}$ then, by the linear independence of the B-spline sequence, $f \neq 0$. Let $I =: (c..d)$ be a maximal open interval in

$$\operatorname{supp} f^{\mathrm{t}} = \bigcup_{\alpha_j \neq 0} (t_j \,..\, t_{j+k}).$$

Then necessarily $I = (t_\mu \,..\, t_{\nu+k})$ for some $1 \leq \nu \leq \mu \leq n$, as well as

$$f = f_I := \sum_{j=\nu}^{\mu} \alpha_j \mathrm{B}_j \quad \text{on } I.$$

In particular, f_I has the 'isolated' zeros $\tau_\nu < \cdots < \tau_\mu$, therefore, by Proposition (6),

$$\mathrm{S}^-(\alpha_\nu, \ldots, \alpha_\mu) \;\geq\; \mu + 1 - \nu,$$

which is nonsense. □

The Schoenberg-Whitney Theorem has been generalized in at least two directions: (i) permission of coincidences in the sequence τ corresponding to osculatory interpolation; and (ii) consideration of a subsequence $(\mathrm{B}_{m_j} : j = 1, \ldots, r)$ instead of the whole sequence (for a suitably longer knot sequence).

Consideration of repeated interpolation sites as in osculatory interpolation also raises the question to what an extent Proposition (6) still holds when r there counts the number of zeros of f with an appropriately defined *multiplicity*. See de Boor [1997] for verification that the sharpest result in this direction, namely that of T. N. T. Goodman [1994], actually uses a very natural definition of multiplicity of a zero z of $f \in \$$, namely the maximal number of 'isolated' zeros near z achievable by some $g \in \$$ in any particular neighborhood of f.

Bandedness of the spline collocation matrix Assume now that the $n \times n$ matrix $\big(\mathrm{B}_j(\tau_i)\big)$ is indeed invertible, hence $t_i < \tau_i < t_{i+k}$, all i (with $t_1^+ = \tau_1$ and $\tau_n = t_{n+k}^-$ permitted). Then, as a major benefit of using B-splines, our matrix $\big(\mathrm{B}_j(\tau_i)\big)$ must be banded, with fewer than k bands above the diagonal and fewer than k bands below the diagonal. For,

$$\mathrm{B}_j(\tau_i) \neq 0 \quad \text{if and only if} \quad t_j < \tau_i < t_{j+k}$$

hence $B_i(\tau_i) \neq 0$ and $B_j(\tau_i) \neq 0$ together implies that $t_j < \tau_i < t_{i+k}$ and $t_i < \tau_i < t_{j+k}$, that is, $|j-i| < k$. This shows that *the matrix* $\big(B_j(\tau_i)\big)$ ***has* bandwidth** *less than* k (in the sense that $B_j(\tau_i) = 0$ for $|j-i| \geq k$) in the case of interest. Of course, if the τ_i's are spaced "in tune with" the t_i's, then the matrix $\big(B_j(\tau_i)\big)$ might have as few as k bands.

Total positivity of the spline collocation matrix A second important property of the linear system (1) is the total positivity of its coefficient matrix. In order to describe this property, we need a notation for submatrices of a given matrix. For a given $m \times n$ matrix $A := (a_{ij})$ and given sequences $I = (i_1, \ldots, i_r)$ and $J := (j_1, \ldots, j_s)$, we use the abbreviation

$$A\begin{pmatrix} I \\ J \end{pmatrix} := (a_{i_p j_q})_{p=1}^{r}{}_{;\,q=1}^{s}.$$

The **matrix** A **is totally positive** if and only if all its minors are nonnegative, that is, if and only if, for $r = 1, 2, \ldots$,

$$\det A\begin{pmatrix} i_1, \ldots, i_r \\ j_1, \ldots, j_r \end{pmatrix} \geq 0 \quad \text{whenever} \quad \begin{matrix} 1 \leq i_1 < \cdots < i_r \leq m \\ 1 \leq j_1 < \cdots < j_r \leq n \end{matrix}.$$

(7) Theorem. (Karlin) *The matrix* $\big(B_j(\tau_i)\big)$ *is totally positive.*

PROOF. If $\widehat{\mathbf{t}}$ is obtained from $\mathbf{t}$ by the insertion of just one knot, and $\widehat{B}_j := B_{j,k,\widehat{\mathbf{t}}}$, all j, then

$$B_j = (1 - w_{j+1})\widehat{B}_{j+1} + w_j \widehat{B}_j,$$

with all $w_j \in [0\,..\,1]$. Since the determinant of a matrix is a *linear* function of the columns of that matrix, we have, for example,

$$\det[\cdots, B_j(x), \ldots] = (1 - w_{j+1}) \det[\cdots, \widehat{B}_{j+1}(x), \ldots] + w_j \det[\cdots, \widehat{B}_j(x), \ldots],$$

with $\cdots$ unchanged in their respective places. It follows that, for any I, J,

$$\det A(I, J) = \sum_H \gamma_H \det \widehat{A}(I, H),$$

with all the $\gamma_H \geq 0$, and the sum, offhand, over certain nondecreasing sequences, since only neighboring columns of $\widehat{A}$ participate in a column of A. However, we may omit all H that are not strictly increasing, since the corresponding determinant is trivially zero. Therefore,

$$\det A(I, J) = \sum_H \gamma_H \det \widehat{A}(I, H),$$

with the $\gamma_H \geq 0$ and all H strictly increasing.

Now insert each of the τ_i enough times so that the resulting refined knot sequence $\tilde{\mathbf{t}}$ contains each τ_i exactly $k-1$ times. By induction, we have

$$\det A(I,J) = \sum_H \gamma_H \det \tilde{A}(I,H),$$

with the $\gamma_H \geq 0$ and all H strictly increasing. However, in each row of $\tilde{A}$, there is exactly one nonzero entry, namely the entry belonging to $\tilde{B}_j$ with $\tilde{t}_j < \tau_i = \tilde{t}_{j+1}$, and that entry equals 1. Further, the sole nonzero entry in subsequent rows occurs to the right of this one. Thus $\tilde{A}(I,H)$ has a nonzero entry in every row iff $\tilde{A}(I,H)$ is the identity matrix. Consequently, $\det \tilde{A}(I,H) \geq 0$ for any H, and therefore $\det A(I,J) \geq 0$. □

A different proof of this theorem can be found in Karlin [1968:X, Thm. 4.1] as a special case of a more general result, or in de Boor $[1976]_1$.

A very simple but practically important consequence of total positivity is given in the following lemma.

(8) **Lemma.** *If C is invertible and totally positive, then its inverse is* **checkerboard**, *meaning that $C^{-1}(i,j)(-1)^{i-j} \geq 0$, all i,j.*

PROOF. By Cramer's rule,

$$C^{-1}(i,j) = (-)^{i-j} \det C(\backslash j, \backslash i)/\det C.$$

□

The subroutine SPLINT

The total positivity of the *collocation matrix*

$$(B_j(\tau_i))$$

is of interest in the present chapter in part because it is possible to show that a linear system with an (invertible) totally positive coefficient matrix may be safely solved by Gauss elimination *without* pivoting (see de Boor & Pinkus [1977]). Since our matrix $(B_j(\tau_i))$ is also banded, this allows us to solve (1) using no more storage than is required to store the $2k-1$ possibly nonzero bands of the matrix $(B_j(\tau_i))$, and also to dispense with the program complications typical for a storage-efficient banded matrix solver using row and/or column interchanges.

The following subprogram SPLINT sets up the linear system (1) for given t and τ, and then solves it using a banded matrix solver without pivoting.

```
      SUBROUTINE SPLINT ( TAU, GTAU, T, N, K, Q, BCOEF, IFLAG )
CALLS BSPLVB, BANFAC/SLV
C
C   SPLINT  PRODUCES THE B-SPLINE COEFF.S  BCOEF  OF THE SPLINE OF ORDER
C   K  WITH KNOTS  T(I), I=1,..., N + K , WHICH TAKES ON THE VALUE
C   GTAU(I) AT  TAU(I), I=1,..., N .
C
C******  I N P U T  ******
C  TAU.....ARRAY OF LENGTH  N , CONTAINING DATA POINT ABSCISSAE.
C    A S S U M P T I O N . . .  TAU  IS STRICTLY INCREASING
C  GTAU.....CORRESPONDING ARRAY OF LENGTH  N , CONTAINING DATA POINT OR-
C        DINATES
C  T.....KNOT SEQUENCE, OF LENGTH  N+K
C  N.....NUMBER OF DATA POINTS AND DIMENSION OF SPLINE SPACE  S(K,T)
C  K.....ORDER OF SPLINE
C
C******  O U T P U T  ******
C  Q.....ARRAY OF SIZE  (2*K-1)*N , CONTAINING THE TRIANGULAR FACTORIZ-
C        ATION OF THE COEFFICIENT MATRIX OF THE LINEAR SYSTEM FOR THE B-
C        COEFFICIENTS OF THE SPLINE INTERPOLANT.
C           THE B-COEFFS FOR THE INTERPOLANT OF AN ADDITIONAL DATA SET
C        (TAU(I),HTAU(I)), I=1,...,N  WITH THE SAME DATA ABSCISSAE CAN
C        BE OBTAINED WITHOUT GOING THROUGH ALL THE CALCULATIONS IN THIS
C        ROUTINE, SIMPLY BY LOADING  HTAU  INTO  BCOEF  AND THEN EXECUT-
C        ING THE    CALL BANSLV ( Q, 2*K-1, N, K-1, K-1, BCOEF )
C  BCOEF.....THE B-COEFFICIENTS OF THE INTERPOLANT, OF LENGTH  N
C  IFLAG.....AN INTEGER INDICATING SUCCESS (= 1)  OR FAILURE (= 2)
C        THE LINEAR SYSTEM TO BE SOLVED IS (THEORETICALLY) INVERTIBLE IF
C        AND ONLY IF
C              T(I) .LT. TAU(I) .LT. T(I+K),    ALL I.
C        VIOLATION OF THIS CONDITION IS CERTAIN TO LEAD TO  IFLAG = 2 .
C
C******  M E T H O D  ******
C    THE I-TH EQUATION OF THE LINEAR SYSTEM  A*BCOEF = B  FOR THE B-CO-
C  EFFS OF THE INTERPOLANT ENFORCES INTERPOLATION AT  TAU(I), I=1,...,N.
C  HENCE,  B(I) = GTAU(I), ALL I, AND  A  IS A BAND MATRIX WITH  2K-1
C   BANDS (IF IT IS INVERTIBLE).
C    THE MATRIX  A  IS GENERATED ROW BY ROW AND STORED, DIAGONAL BY DI-
C  AGONAL, IN THE  R O W S  OF THE ARRAY  Q , WITH THE MAIN DIAGONAL GO-
C  ING INTO ROW  K .  SEE COMMENTS IN THE PROGRAM BELOW.
C    THE BANDED SYSTEM IS THEN SOLVED BY A CALL TO  BANFAC (WHICH CON-
C  STRUCTS THE TRIANGULAR FACTORIZATION FOR  A  AND STORES IT AGAIN IN
C   Q ), FOLLOWED BY A CALL TO  BANSLV (WHICH THEN OBTAINS THE SOLUTION
C   BCOEF  BY SUBSTITUTION).
C    BANFAC  DOES NO PIVOTING, SINCE THE TOTAL POSITIVITY OF THE MATRIX
C  A  MAKES THIS UNNECESSARY.
C
      INTEGER IFLAG,K,N,   I,ILP1MX,J,JJ,KM1,KPKM2,LEFT,LENQ,NP1
      REAL BCOEF(N),GTAU(N),Q,T,TAU(N),   TAUI
      DIMENSION Q(2*K-1,N), T(N+K)
      NP1 = N + 1
      KM1 = K - 1
      KPKM2 = 2*KM1
      LEFT = K
C                 ZERO OUT ALL ENTRIES OF Q
      LENQ = N*(K+KM1)
      DO 5 I=1,LENQ
    5    Q(I) = 0.
C
```

```
C  ***    LOOP OVER I TO CONSTRUCT THE  N  INTERPOLATION EQUATIONS
      DO 30 I=1,N
         TAUI = TAU(I)
         ILP1MX = MINO(I+K,NP1)
C        *** FIND  LEFT  IN THE CLOSED INTERVAL (I .. I+K-1) SUCH THAT
C                T(LEFT) .LE. TAU(I) .LT. T(LEFT+1)
C        MATRIX IS SINGULAR IF THIS IS NOT POSSIBLE
         LEFT = MAXO(LEFT,I)
         IF (TAUI .LT. T(LEFT))          GO TO 998
   15       IF (TAUI .LT. T(LEFT+1))    GO TO 16
            LEFT = LEFT + 1
            IF (LEFT .LT. ILP1MX)       GO TO 15
         LEFT = LEFT - 1
         IF (TAUI .GT. T(LEFT+1))       GO TO 998
C        *** THE I-TH EQUATION ENFORCES INTERPOLATION AT TAUI, HENCE
C        A(I,J) = B(J,K,T)(TAUI), ALL J. ONLY THE  K  ENTRIES WITH  J =
C        LEFT-K+1,...,LEFT ACTUALLY MIGHT BE NONZERO. THESE  K  NUMBERS
C        ARE RETURNED, IN  BCOEF (USED FOR TEMP.STORAGE HERE), BY THE
C        FOLLOWING
   16    CALL BSPLVB ( T, K, 1, TAUI, LEFT, BCOEF )
C        WE THEREFORE WANT  BCOEF(J) = B(LEFT-K+J)(TAUI) TO GO INTO
C        A(I,LEFT-K+J), I.E., INTO  Q(I-(LEFT+J)+2*K,(LEFT+J)-K) SINCE
C        A(I+J,J)  IS TO GO INTO  Q(I,K,J), ALL I,J,  IF WE CONSIDER  Q
C        AS A TWO-DIM. ARRAY , WITH  2*K-1  ROWS (SEE COMMENTS IN
C        BANFAC). IN THE PRESENT PROGRAM, WE TREAT  Q  AS AN EQUIVALENT
C        ONE-DIMENSIONAL ARRAY (BECAUSE OF FORTRAN RESTRICTIONS ON
C        DIMENSION STATEMENTS) . WE THEREFORE WANT  BCOEF(J) TO GO INTO
C        ENTRY
C            I -(LEFT+J) + 2*K + ((LEFT+J) - K-1)*(2*K-1)
C                  =  I-LEFT+1 + (LEFT -K)*(2*K-1) + (2*K-2)*J
C        OF  Q .
         JJ = I-LEFT+1 + (LEFT-K)*(K+KM1)
         DO 30 J=1,K
            JJ = JJ+KPKM2
   30       Q(JJ) = BCOEF(J)
C
C     ***OBTAIN FACTORIZATION OF  A  , STORED AGAIN IN  Q.
      CALL BANFAC ( Q, K+KM1, N, KM1, KM1, IFLAG )
                                          GO TO (40,999), IFLAG
C     *** SOLVE  A*BCOEF = GTAU  BY BACKSUBSTITUTION
   40 DO 41 I=1,N
   41    BCOEF(I) = GTAU(I)
      CALL BANSLV ( Q, K+KM1, N, KM1, KM1, BCOEF )
                                          RETURN
  998 IFLAG = 2
  999 PRINT 699
  699 FORMAT(41H LINEAR SYSTEM IN  SPLINT  NOT INVERTIBLE)
                                          RETURN
      END
```

The program makes use of an unspecified banded matrix solver BANFAC/BANSLV to solve (1) by Gauss elimination without pivoting. For completeness, we include here a listing of one such.

```
      SUBROUTINE BANFAC ( W, NROWW, NROW, NBANDL, NBANDU, IFLAG )
C  RETURNS IN  W  THE LU-FACTORIZATION (WITHOUT PIVOTING) OF THE BANDED
C  MATRIX  A  OF ORDER  NROW  WITH  (NBANDL + 1 + NBANDU) BANDS OR DIAG-
C  ONALS IN THE WORK ARRAY  W .
C
C******  I N P U T  ******
```

```
C  W.....WORK ARRAY OF SIZE  (NROWW,NROW)  CONTAINING THE INTERESTING
C        PART OF A BANDED MATRIX  A , WITH THE DIAGONALS OR BANDS OF  A
C        STORED IN THE ROWS OF  W , WHILE COLUMNS OF  A  CORRESPOND TO
C        COLUMNS OF  W . THIS IS THE STORAGE MODE USED IN  LINPACK  AND
C        RESULTS IN EFFICIENT INNERMOST LOOPS.
C           EXPLICITLY,  A  HAS  NBANDL  BANDS BELOW THE DIAGONAL
C                              +      1      (MAIN) DIAGONAL
C                              +   NBANDU  BANDS ABOVE THE DIAGONAL
C        AND THUS, WITH    MIDDLE = NBANDU + 1,
C          A(I+J,J)  IS IN  W(I+MIDDLE,J)  FOR I=-NBANDU,...,NBANDL
C                                                J=1,...,NROW .
C        FOR EXAMPLE, THE INTERESTING ENTRIES OF A (1,2)-BANDED MATRIX
C        OF ORDER  9  WOULD APPEAR IN THE FIRST  1+1+2 = 4  ROWS OF  W
C        AS FOLLOWS.
C                              13 24 35 46 57 68 79
C                           12 23 34 45 56 67 78 89
C                        11 22 33 44 55 66 77 88 99
C                        21 32 43 54 65 76 87 98
C
C        ALL OTHER ENTRIES OF  W  NOT IDENTIFIED IN THIS WAY WITH AN EN-
C        TRY OF  A  ARE NEVER REFERENCED .
C  NROWW.....ROW DIMENSION OF THE WORK ARRAY  W .
C        MUST BE  .GE.  NBANDL + 1 + NBANDU  .
C  NBANDL.....NUMBER OF BANDS OF  A  BELOW THE MAIN DIAGONAL
C  NBANDU.....NUMBER OF BANDS OF  A  ABOVE THE MAIN DIAGONAL .
C
C******  O U T P U T  ******
C  IFLAG.....INTEGER INDICATING SUCCESS( = 1) OR FAILURE ( = 2) .
C     IF  IFLAG = 1, THEN
C  W.....CONTAINS THE LU-FACTORIZATION OF  A  INTO A UNIT LOWER TRIANGU-
C        LAR MATRIX  L  AND AN UPPER TRIANGULAR MATRIX  U (BOTH BANDED)
C        AND STORED IN CUSTOMARY FASHION OVER THE CORRESPONDING ENTRIES
C        OF  A . THIS MAKES IT POSSIBLE TO SOLVE ANY PARTICULAR LINEAR
C        SYSTEM  A*X = B  FOR  X  BY A
C             CALL BANSLV ( W, NROWW, NROW, NBANDL, NBANDU, B )
C        WITH THE SOLUTION X  CONTAINED IN  B  ON RETURN .
C     IF  IFLAG = 2, THEN
C        ONE OF  NROW-1, NBANDL,NBANDU FAILED TO BE NONNEGATIVE, OR ELSE
C        ONE OF THE POTENTIAL PIVOTS WAS FOUND TO BE ZERO INDICATING
C        THAT  A  DOES NOT HAVE AN LU-FACTORIZATION. THIS IMPLIES THAT
C        A  IS SINGULAR IN CASE IT IS TOTALLY POSITIVE .
C
C******  M E T H O D  ******
C     GAUSS ELIMINATION  W I T H O U T  PIVOTING IS USED. THE ROUTINE IS
C  INTENDED FOR USE WITH MATRICES  A  WHICH DO NOT REQUIRE ROW INTER-
C  CHANGES DURING FACTORIZATION, ESPECIALLY FOR THE  T O T A L L Y
C  P O S I T I V E  MATRICES WHICH OCCUR IN SPLINE CALCULATIONS.
C     THE ROUTINE SHOULD NOT BE USED FOR AN ARBITRARY BANDED MATRIX.
C
      INTEGER IFLAG,NBANDL,NBANDU,NROW,NROWW,   I,IPK,J,JMAX,K,KMAX
     *                                         ,MIDDLE,MIDMK,NROWM1
      REAL W(NROWW,NROW),   FACTOR,PIVOT
C
      IFLAG = 1
      MIDDLE = NBANDU + 1
C                          W(MIDDLE,.) CONTAINS THE MAIN DIAGONAL OF  A .
      NROWM1 = NROW - 1
      IF (NROWM1)                          999,900,1
    1 IF (NBANDL .GT. 0)                   GO TO 10
C                 A IS UPPER TRIANGULAR. CHECK THAT DIAGONAL IS NONZERO .
      DO 5 I=1,NROWM1
         IF (W(MIDDLE,I) .EQ. 0.)          GO TO 999
    5    CONTINUE
                                           GO TO 900
   10 IF (NBANDU .GT. 0)                   GO TO 20
```

```
C               A IS LOWER TRIANGULAR. CHECK THAT DIAGONAL IS NONZERO AND
C                  DIVIDE EACH COLUMN BY ITS DIAGONAL .
      DO 15 I=1,NROWM1
         PIVOT = W(MIDDLE,I)
         IF(PIVOT .EQ. 0.)                    GO TO 999
         JMAX = MINO(NBANDL, NROW - I)
         DO 15 J=1,JMAX
   15       W(MIDDLE+J,I) = W(MIDDLE+J,I)/PIVOT
                                              RETURN
C
C        A  IS NOT JUST A TRIANGULAR MATRIX. CONSTRUCT LU FACTORIZATION
   20 DO 50 I=1,NROWM1
C                                        W(MIDDLE,I)  IS PIVOT FOR I-TH STEP .
         PIVOT = W(MIDDLE,I)
         IF (PIVOT .EQ. 0.)                   GO TO 999
C              JMAX  IS THE NUMBER OF (NONZERO) ENTRIES IN COLUMN  I
C                  BELOW THE DIAGONAL .
         JMAX = MINO(NBANDL,NROW - I)
C           DIVIDE EACH ENTRY IN COLUMN  I  BELOW DIAGONAL BY PIVOT .
         DO 32 J=1,JMAX
   32       W(MIDDLE+J,I) = W(MIDDLE+J,I)/PIVOT
C                KMAX  IS THE NUMBER OF (NONZERO) ENTRIES IN ROW  I  TO
C                  THE RIGHT OF THE DIAGONAL .
         KMAX = MINO(NBANDU,NROW - I)
C                 SUBTRACT  A(I,I+K)*(I-TH COLUMN) FROM (I+K)-TH COLUMN
C                 (BELOW ROW  I ) .
         DO 40 K=1,KMAX
            IPK = I + K
            MIDMK = MIDDLE - K
            FACTOR = W(MIDMK,IPK)
            DO 40 J=1,JMAX
   40          W(MIDMK+J,IPK) = W(MIDMK+J,IPK) - W(MIDDLE+J,I)*FACTOR
   50    CONTINUE
C                                           CHECK THE LAST DIAGONAL ENTRY .
  900 IF (W(MIDDLE,NROW) .NE. 0.)           RETURN
  999 IFLAG = 2
                                            RETURN
      END



      SUBROUTINE BANSLV ( W, NROWW, NROW, NBANDL, NBANDU, B )
C  COMPANION ROUTINE TO  BANFAC . IT RETURNS THE SOLUTION  X  OF THE
C  LINEAR SYSTEM  A*X = B  IN PLACE OF  B , GIVEN THE LU-FACTORIZATION
C  FOR  A  IN THE WORKARRAY  W .
C
C******  I N P U T  ******
C  W, NROWW,NROW,NBANDL,NBANDU.....DESCRIBE THE LU-FACTORIZATION OF A
C        BANDED MATRIX  A  OF ORDER  NROW  AS CONSTRUCTED IN  BANFAC .
C        FOR DETAILS, SEE  BANFAC .
C  B.....RIGHT SIDE OF THE SYSTEM TO BE SOLVED .
C
C******  O U T P U T  ******
C  B.....CONTAINS THE SOLUTION  X , OF ORDER  NROW .
C
C******  M E T H O D  ******
C     (WITH  A = L*U, AS STORED IN  W,) THE UNIT LOWER TRIANGULAR SYSTEM
C  L(U*X) = B  IS SOLVED FOR  Y = U*X, AND  Y  STORED IN  B . THEN THE
C  UPPER TRIANGULAR SYSTEM  U*X = Y  IS SOLVED FOR  X  . THE CALCUL-
C  ATIONS ARE SO ARRANGED THAT THE INNERMOST LOOPS STAY WITHIN COLUMNS.
C
```

```
      INTEGER NBANDL,NBANDU,NROW,NROWW,    I,J,JMAX,MIDDLE,NROWM1
      REAL W(NROWW,NROW),B(NROW)
      MIDDLE = NBANDU + 1
      IF (NROW .EQ. 1)                          GO TO 49
      NROWM1 = NROW - 1
      IF (NBANDL .EQ. 0)                        GO TO 30
C                                    FORWARD PASS
C          FOR I=1,2,...,NROW-1, SUBTRACT  RIGHT SIDE(I)*(I-TH COLUMN
C          OF  L )  FROM RIGHT SIDE  (BELOW I-TH ROW) .
      DO 21 I=1,NROWM1
         JMAX = MINO(NBANDL, NROW-I)
         DO 21 J=1,JMAX
   21       B(I+J) = B(I+J) - B(I)*W(MIDDLE+J,I)
C                                    BACKWARD PASS
C           FOR I=NROW,NROW-1,...,1, DIVIDE RIGHT SIDE(I) BY I-TH DIAG-
C           ONAL ENTRY OF  U, THEN SUBTRACT  RIGHT SIDE(I)*(I-TH COLUMN
C           OF  U)  FROM RIGHT SIDE  (ABOVE I-TH ROW).
   30 IF (NBANDU .GT. 0)                        GO TO 40
C                                    A  IS LOWER TRIANGULAR .
      DO 31 I=1,NROW
   31    B(I) = B(I)/W(1,I)
                                               RETURN

   40 I = NROW
   41    B(I) = B(I)/W(MIDDLE,I)
         JMAX = MINO(NBANDU,I-1)
         DO 45 J=1,JMAX
   45       B(I-J) = B(I-J) - B(I)*W(MIDDLE-J,I)
         I = I - 1
         IF (I .GT. 1)                         GO TO 41
   49 B(1) = B(1)/W(MIDDLE,1)
                                               RETURN

      END
```

Readers unfamiliar with the computational (and theoretical) aspects of solving systems of linear algebraic equations are urged to consult the fine textbook on that subject by Forsythe & Moler [1968] or any textbook on numerical analysis, for example, Conte & de Boor [1980]. They will then recognize the splitting of the task of solving $C\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ into factoring C and then solving for $\mathbf{x}$ with the aid of that factorization as quite customary. In fact, the only special feature of `BANFAC/BANSLV` above compared to other banded matrix solvers available (see for example, Wilkinson & Reinsch [1971:pp. 71–92]) is the fact that no pivoting for size is used. This makes `BANFAC/BANSLV` in general *inappropriate* for band matrices other than totally positive or strictly diagonally dominant ones.

The interplay between knots and data sites We consider now some bounds on the interpolation error. We continue to measure the size of a function g by its max-norm

$$\|g\| := \max_{a \le x \le b} |g(x)|$$

for some fixed interval $[a\,..\,b]$ that also contains all the data sites $\tau_1, \ldots, \tau_n$. Although we could be more general, we consider only knot sequences $\mathbf{t} =$

$(t_i)_1^{n+k}$ with $a = t_i = \cdots = t_k < t_{k+1} \le \cdots \le t_n < t_{n+1} = \cdots = t_{n+k} = b$, just as in Chapter XII. We assume that

$$\mathrm{B}_i(\tau_i) > 0, \qquad i = 1, \ldots, n,$$

so that (1) has exactly one solution and we denote the corresponding interpolating spline function $f = \sum_i \alpha_i \mathrm{B}_i$ by

$$Ig.$$

We derived in Chapter XII the error bound XII(39) for cubic spline interpolation at knots. The same argument establishes the following more general result.

(9) Lemma. *For every continuous function g on $[a\,..\,b]$, the interpolation error is bounded by*

$$\|g - Ig\| \le (1 + \|I\|)\,\mathrm{dist}\,(g, \$_{k,\mathbf{t}}) \tag{10}$$

with

$$\|I\| := \max\{\, \|Ig\|/\|g\| : g \in C[a\,..\,b] \setminus \{0\}\,\}.$$

Since Chapter XII already provides us with various bounds on the number $\mathrm{dist}\,(g, \$_{k,\mathbf{t}})$, this lemma then focuses attention on the *norm* $\|I\|$ of our interpolation process I.

The linear system (1) becomes singular when the condition (3) is violated. We therefore expect the number $\|I\|$ to grow large as τ_i approaches the limits t_i or t_{i+k} of its allowable range (and the knot multiplicity of this limit is $< k$).

A different cause for large $\|I\|$ is nonuniformity of τ, as we saw already in Example XII(37). To recall, we saw there an interpolation error of more than 383.4 when using cubic spline interpolation at knots at four (very nonuniformly spaced) data sites, with $g(x) = \sqrt{x+1}$ the function to be approximated in $[-1\,..\,1]$. We know from Example II(12) that the distance $\mathrm{dist}\,(g, \Pi_{<4})$ of the function $\sqrt{x+1}$ on $[-1\,..\,1]$ from cubic polynomials is $\le .03005$. Therefore, since $\Pi_{<4} \subset \$_{4,\mathbf{t}}$ whatever $\mathbf{t}$ might be, $\mathrm{dist}\,(g, \$_{4,\mathbf{t}}) \le .03005$. In consequence, $383.4 \le (1 + \|I\|).03005$, therefore

$$\|I\| > 12\,754,$$

with a corresponding loss of five decimal places in the accuracy achieved.

The next lemma shows this to be no accident.

(11) Lemma. *There exists a positive* const_k *(closely related to the number $D_{k,\infty}^{-1}$ of B-spline Property XI(x); see XI(4)) so that the norm of the interpolation process I is bounded below by*

$$\|I\| \ge \mathrm{const}_k \max_i \frac{\min\{\, t_{j+k-1} - t_j : (t_j\,..\,t_{j+k-1}) \cap (\tau_i\,..\,\tau_{i+1}) \ne \emptyset\,\}}{\Delta\tau_i}.$$

A simple proof of this lemma can be found in de Boor [1975]; see Problem 3 below.

The lemma shows that we can make $\|I\|$ arbitrarily large by letting two interpolation sites approach each other while keeping everything else fixed. It shows that we should pay careful attention to how we place data sites and knots so as to avoid making the constant $1 + \|I\|$ in (10) large, for a large $\|I\|$ would mean throwing away all that approximation power in $\$_{k,t}$.

To be sure, (10) is a bound and does not *guarantee* a relatively large interpolation error for every g whenever $\|I\|$ is large. But, as we will see below, a large $\|I\|$ makes the noise inherent in floating-point calculations an important contributor to the approximation error.

Even order interpolation at knots

For even order k,

$$k =: 2m$$

say, one can make most of the data sites also knots. One chooses $t_{k+1}, \dots, t_n$ so that

$$\begin{aligned} t_k = a \le \tau_1 < \cdots < \tau_m < t_{k+1}, \\ \tau_{m+j} = t_{k+j}, \qquad j = 1, \dots, n-k, \\ t_n < \tau_{n-m+1} < \cdots < \tau_n \le b = t_{n+1}. \end{aligned} \tag{12}$$

For $k = 4$, this corresponds to cubic spline interpolation with the "not-a-knot" end condition discussed in Chapter IV (see p. 44). We denote the resulting interpolation process by I_k.

(13) Theorem. *There exists a positive* $\text{const}_{k,M(\tau,m)}$ *which depends on* $k = 2m$ *and increases to infinity as the following* **global mesh ratio**

$$M(\tau, m) := \max_{i,j} \frac{\tau_{i+m} - \tau_i}{\tau_{j+m} - \tau_j}$$

goes to infinity, so that, for every m *times continuously differentiable function* g,

$$\|g - I_k g\| \le \text{const}_{k,M(\tau,m)} \operatorname{dist}(g^{(m)}, \$_{m,t}) |\tau|^m.$$

A proof can be adduced from de Boor [1976]$_3$.

This theorem shows that spline interpolation at knots to a smooth function g produces good results even for nonuniform data site spacing, that is, even when Lemma (11) shows that $\|I_k\|$ must be very large, and so shows that a large $\|I_k\|$ *may* have an adverse effect on the quality of the approximation $I_k g$ to g, but is not *guaranteed* to do so. But, Theorem (13) uses the smoothness of the function g and is therefore valid only to the extent that this smoothness is not destroyed in calculations by noise.

(14) Example: A large $\|I\|$ amplifies noise We choose $k = 4$, $g(x) = x^3 + x^2 + x + 1$, hence $I_4 g = g$ in exact arithmetic. We perturb the function values $g(\tau_i)$ by errors of size SIZE, pick $n = 7$, and $\tau_1 < \cdots < \tau_7$ equally spaced in $[0 \,..\, 1]$, except that $\Delta\tau_3 =: h$, initially equal to $1/6$, is allowed to grow small. The program is written to invite further experimentation.

```
CHAPTER XIII, EXAMPLE 1. A LARGE NORM AMPLIFIES NOISE.
CALLS SPLINT(BANFAC/SLV,BSPLVB),BSPLPP(BSPLVB*),PPVALU(INTERV),ROUND
C     AN INITIALLY UNIFORM DATA POINT DISTRIBUTION OF  N  POINTS IS
C  CHANGED  I T E R M X  TIMES BY MOVING THE  J C L O S E -TH DATA POINT
C  TOWARD ITS LEFT NEIGHBOR, CUTTING THE DISTANCE BETWEEN THE TWO BY A
C  FACTOR OF  R A T E  EACH TIME . TO DEMONSTRATE THE CORRESPONDING
C  INCREASE IN THE NORM OF CUBIC SPLINE INTERPOLATION AT THESE DATA
C  POINTS, THE DATA ARE TAKEN FROM A CUBIC POLYNOMIAL (WHICH WOULD BE
C  INTERPOLATED EXACTLY) BUT WITH NOISE OF SIZE  S I Z E  ADDED. THE RE-
C  SULTING NOISE OR ERROR IN THE INTERPOLANT (COMPARED TO THE CUBIC)
C  GIVES THE NORM OR NOISE AMPLIFICATION FACTOR AND IS PRINTED OUT
C  TOGETHER WITH THE DIMINISHING DISTANCE  H  BETWEEN THE TWO DATA
C  POINTS.
      INTEGER I,IFLAG,ISTEP,ITER,ITERMX,J,JCLOSE,L,N,NMAX,NM1
      PARAMETER (NMAX=200)
      REAL AMAX,BCOEF(NMAX),BREAK(NMAX),COEF(NMAX*4),DX,FLTNM1,FX
     * ,GTAU(NMAX),H,RATE,SCRTCH(NMAX*7),SIZE,STEP,T(NMAX+4),TAU(NMAX),X
C     REAL AMAX,BCOEF(200),BREAK(200),COEF(800),DX,FLTNM1,FX
C     * ,GTAU(200),H,RATE,SCRTCH(1400),SIZE,STEP,T(204),TAU(200),X
      COMMON /ROUNT/ SIZE
      DATA STEP, ISTEP / 20., 20 /
C                                          FUNCTION TO BE INTERPOLATED .
      G(X) = 1.+X*(1.+X*(1.+X))
      READ 500,N,ITERMX,JCLOSE,SIZE,RATE
  500 FORMAT(3I3/E10.3/E10.3)
      PRINT 600,SIZE
  600 FORMAT(16H SIZE OF NOISE =,E8.2//
     *       25H    H              MAX.ERROR)
C                                          START WITH UNIFORM DATA POINTS .
      NM1 = N - 1
      FLTNM1 = FLOAT(NM1)
      DO 10 I=1,N
   10    TAU(I) = FLOAT(I-1)/FLTNM1
C                    SET UP KNOT SEQUENCE FOR NOT-A-KNOT END CONDITION .
      DO 11 I=1,4
         T(I) = TAU(1)
   11    T(N+I) = TAU(N)
      DO 12 I=5,N
   12    T(I) = TAU(I-2)
C
      DO 100 ITER=1,ITERMX
         DO 21 I=1,N
   21       GTAU(I) = ROUND(G(TAU(I)))
         CALL SPLINT ( TAU, GTAU, T, N, 4, SCRTCH, BCOEF, IFLAG )
                                          GO TO (24,23),IFLAG
   23    PRINT 623
  623    FORMAT(27H SOMETHING WRONG IN SPLINT.)
                                          STOP
   24    CALL BSPLPP ( T, BCOEF, N, 4, SCRTCH, BREAK, COEF, L )
```

```
C                                          CALCULATE MAX.INTERPOLATION ERROR .
      AMAX = 0.
      DO 30 I=4,N
         DX = (BREAK(I-2) - BREAK(I-3))/STEP
         DO 25 J=2,ISTEP
            X = BREAK(I-2) - DX*FLOAT(J-1)
            FX = PPVALU(BREAK,COEF,L,4,X,0)
   25       AMAX = AMAX1(AMAX,ABS(FX-G(X)))
   30    CONTINUE
      H = TAU(JCLOSE) - TAU(JCLOSE-1)
      PRINT 630,H,AMAX
  630 FORMAT(E9.2,E15.3)
C                MOVE TAU(JCLOSE) TOWARD ITS LEFT NEIGHBOR SO AS TO CUT
C                THEIR DISTANCE BY A FACTOR OF  RATE .
      TAU(JCLOSE) = (TAU(JCLOSE) + (RATE-1.)*TAU(JCLOSE-1))/RATE
  100 CONTINUE
                                           STOP
      END
```

```
      REAL FUNCTION ROUND ( X )
C  FROM  * A PRACTICAL GUIDE TO SPLINES *  BY C. DE BOOR
CALLED IN EXAMPLE 1  OF CHAPTER XIII
      REAL X,   FLIP,SIZE
      COMMON /ROUNT/ SIZE
      DATA FLIP /-1./
      FLIP = -FLIP
      ROUND = X + FLIP*SIZE
                                           RETURN
      END
```

The input N = 7, ITERMX = 10, JCLOSE = 4, SIZE = 10^{-6} and RATE = 2. produces the following output, on the left.

SIZE OF NOISE =0.10-05

H	MAX.ERROR
0.17+00	0.143-05
0.83-01	0.250-05
0.42-01	0.691-05
0.21-01	0.159-04
0.10-01	0.353-04
0.52-02	0.672-04
0.26-02	0.146-03
0.13-02	0.259-03
0.65-03	0.570-03
0.33-03	0.111-02

SIZE OF NOISE =0.00+00

H	MAX.ERROR
0.17+00	0.238-06
0.83-01	0.238-06
0.42-01	0.477-06
0.21-01	0.477-06
0.10-01	0.143-05
0.52-02	0.477-06
0.26-02	0.954-06
0.13-02	0.204-04
0.65-03	0.763-05
0.33-03	0.157-04

The increased magnification of the initial noise is very noticeable. The magnification factor increases like $1/h$, confirming that the lower bound for $\|I\|$ in Lemma (11) is qualitatively correct. The same input, but with SIZE = 0., produces the output on the right which shows the same kind of noise increase even when starting with supposedly correct data. □

In a very simplified form, the machinery at work here is the following. If we consider the linear functional λ_h of approximate differentiation on

$C[a\,..\,b]$ given by the rule

$$\lambda_h g := \frac{g(a+h)-g(a)}{h}, \qquad \text{all } g,$$

then its size or norm is

$$\|\lambda_h\| := \max\{\,|\lambda_h g|/\|g\| : g \in C[a\,..\,b] \setminus \{0\}\,\} = 2/h.$$

In particular, the norm becomes arbitrarily large when h becomes small. This corresponds to Lemma (11). But now, for any continuously differentiable function g, $\lambda_h g \to g'(a)$ as $h \to 0$, hence for a smooth g, $|\lambda_h g|$ stays bounded even though $\|\lambda_h\| \to \infty$ as $h \to 0$. This corresponds to Theorem (13). But, finally, if the computed function values are inaccurate because of the finite precision arithmetic used, then the computed value for $\lambda_h g$ equals $\lambda_h g + \varepsilon/h$, with ε the noise or error in the computed function values. Therefore, the computed number $\lambda_h g$ may become quite large as h becomes small, that is, as the noise in the data or in the machine arithmetic becomes the dominating phenomenon. (See also Problem 7.)

Now a final observation to prevent the reader from jumping to unwarranted conclusions. A large $\|I\|$ does not come from some small $\Delta\tau_j$ *per se*. If, for example, the data sites are chosen uniformly spaced, then $\|I_4\| \le 1.7\cdots$ regardless of how small each $\Delta\tau_j$ might be!

Interpolation at knot averages If one has the freedom to choose the data sites (τ_i) for given knot sequence $\mathbf{t}$, then use of the Greville sites

$$\tau_i = t^*_{ik} := (t_{i+1} + \cdots + t_{i+k-1})/(k-1) \tag{15}$$

is to be recommended. These points are those used in Schoenberg's variation diminishing approximation XI(33). Call the resulting interpolant $I^*_k g$. Then I^*_2 is just broken line interpolation, and $\|I^*_2\| = 1$. It can be shown (Marsden [1974] (see Problem VI.1), de Boor [1975]) that

$$\|I^*_3\| \le 2, \qquad \|I^*_4\| \le 27$$

(in fact, $\|I^*_4\| \le 3$ seems a more realistic, but as yet unproven, bound). It has been conjectured that

$$\sup_{\mathbf{t}} \|I^*_k\| < \infty, \qquad \text{for any particular } k.$$

However, R.Q. Jia [1988] gives a counterexample to this, for $k = 32$. Nevertheless, since

$$\|g - I^*_k g\| \le (1 + \|I^*_k\|)\,\mathrm{dist}\,(g, \$_{k,\mathbf{t}}),$$

it is likely that, for *moderate* k (and certainly for $k \le 4$), $I^*_k g$ is about as good an approximation to g from $\$_{k,\mathbf{t}}$ as possible.

(16) Example: Cubic spline interpolation at knot averages with good knots In this example, we conclude our experimentation with the knot placement algorithm NEWNOT begun in Chapter XII; see Example XII(33)ff. We now have a cubic spline approximation process at hand which has a small norm, namely cubic spline interpolation at knot averages. We return therefore to the goal of Example XII(33), to guess a reasonable knot distribution from an approximation to the function $g(x) = \sqrt{1+x}$ in $[-1 \,..\, 1]$.

```
CHAPTER XIII, EXAMPLE 2. CUBIC SPLINE INTERPOLANT AT KNOT AVERAGES
C                        WITH GOOD KNOTS.
CALLS SPLINT(BANFAC/SLV,BSPLVB),NEWNOT,BSPLPP(BSPLVB*)
      INTEGER I,ISTEP,ITER,ITERMX,N,NHIGH,NMAX,NMK,NLOW
      PARAMETER (NMAX=20)
      REAL ALGERP,ALOGER,BCOEF(NMAX+2),BREAK(NMAX),DECAY,DX,ERRMAX
     *    ,C(4,NMAX),G,GTAU(NMAX),H,PNATX,SCRTCH(NMAX*7),STEP
     *    ,T(NMAX+6),TAU(NMAX),TNEW(NMAX)
C     REAL ALGERP,ALOGER,BCOEF(22),BREAK(20),DECAY,DX,ERRMAX,
C    *    C(4,20),G,GTAU(20),H,PNATX,SCRTCH(140),STEP,T(26),
C    *    TAU(20),TNEW(20)
C     ISTEP AND STEP = FLOAT(ISTEP) SPECIFY POINT DENSITY FOR ERROR DET-
C     ERMINATION.
      DATA STEP, ISTEP /20., 20/
C                                        THE FUNCTION  G  IS TO BE INTERPOLATED .
      G(X) = SQRT(X+1.)
      DECAY = 0.
C      READ IN THE NUMBER OF ITERATIONS TO BE CARRIED OUT AND THE LOWER
C      AND UPPER LIMIT FOR THE NUMBER  N  OF DATA POINTS TO BE USED.
      READ 500,ITERMX,NLOW,NHIGH
  500 FORMAT(3I3)
      PRINT 600, ITERMX
  600 FORMAT(I4,22H CYCLES THROUGH NEWNOT//
     *       28H  N   MAX.ERROR   DECAY EXP./)
C                                         LOOP OVER  N = NUMBER OF DATA POINTS .
      DO 40 N=NLOW,NHIGH,2
    4    NMK = N-4
         H = 2./FLOAT(NMK+1)
         DO 5 I=1,4
            T(I) = -1.
    5       T(N+I) = 1.
         IF (NMK .LT. 1)                    GO TO 10
         DO 6 I=1,NMK
    6       T(I+4) = FLOAT(I)*H - 1.
   10    ITER = 1
C                CONSTRUCT CUBIC SPLINE INTERPOLANT. THEN, ITERMX  TIMES,
C                 DETERMINE NEW KNOTS FROM IT AND FIND A NEW INTERPOLANT.
   11       DO 12 I=1,N
               TAU(I) = (T(I+1)+T(I+2)+T(I+3))/3.
   12          GTAU(I) = G(TAU(I))
            CALL SPLINT ( TAU, GTAU, T, N, 4, SCRTCH, BCOEF, IFLAG )
            CALL BSPLPP ( T, BCOEF, N, 4, SCRTCH, BREAK, C, L )
            IF (ITER .GT. ITERMX)        GO TO 19
            ITER = ITER + 1
            CALL NEWNOT ( BREAK, C, L, 4, TNEW, L, SCRTCH )
   15       DO 18 I=2,L
   18          T(3+I) = TNEW(I)
C                        ESTIMATE MAX.INTERPOLATION ERROR ON  (-1..1) .
   19    ERRMAX = 0.
                                          GO TO 11
C                                LOOP OVER POLYNOMIAL PIECES OF INTERPOLANT .
```

```
      DO 30 I=1,L
         DX = (BREAK(I+1)-BREAK(I))/STEP
C              INTERPOLATION ERROR IS CALCULATED AT  ISTEP  POINTS PER
C                  POLYNOMIAL PIECE .
         DO 30 J=1,ISTEP
            H = FLOAT(J)*DX
            PNATX = C(1,I)+H*(C(2,I)+H*(C(3,I)+H*C(4,I)/3.)/2.)
   30       ERRMAX = AMAX1(ERRMAX,ABS(G(BREAK(I)+H)-PNATX))
C                                              CALCULATE DECAY EXPONENT .
      ALOGER = ALOG(ERRMAX)
         IF (N .GT. NLOW)  DECAY =
     *        (ALOGER - ALGERP)/ALOG(FLOAT(N)/FLOAT(N-2))
         ALGERP = ALOGER
C
   40    PRINT 640,N,ERRMAX,DECAY
  640 FORMAT(I3,E12.4,F11.2)
                                                STOP
      END
```

The program allows one to specify the first and final value for the number N of data points to be used. For each choice of N, the knots are initially equally spaced and are then modified through ITERMX cycles through NEWNOT. ITERMX is specified on input.

The output for $N = 2, 4, \ldots, 20$, and for ITERMX $= 0, 3, 6$, is shown below, to the left. Both decay rate and error do indeed improve as NEWNOT is used more and more. But, even with 6 cycles through NEWNOT, we do not get the hoped for N^{-4}-convergence.

0 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.9126-01	-1.19
8	0.7070-01	-0.89
10	0.5975-01	-0.75
12	0.5270-01	-0.69
14	0.4767-01	-0.65
16	0.4385-01	-0.63
18	0.4082-01	-0.61
20	0.3834-01	-0.59

0 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.9126-01	-1.19
8	0.6436-01	-1.21
10	0.4381-01	-1.72
12	0.2896-01	-2.27
14	0.1919-01	-2.67
16	0.1278-01	-3.04
18	0.8604-02	-3.36
20	0.5865-02	-3.64

3 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.7753-01	-1.59
8	0.4044-01	-2.26
10	0.2593-01	-1.99
12	0.1822-01	-1.94
14	0.1364-01	-1.88
16	0.1065-01	-1.85
18	0.8637-02	-1.78
20	0.7155-02	-1.79

1 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.8335-01	-1.41
8	0.5004-01	-1.77
10	0.2833-01	-2.55
12	0.1647-01	-2.97
14	0.9871-02	-3.32
16	0.6103-02	-3.60
18	0.3895-02	-3.81
20	0.2568-02	-3.95

6 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.7545-01	-1.65
8	0.3413-01	-2.76
10	0.1870-01	-2.70
12	0.1139-01	-2.72
14	0.7548-02	-2.67
16	0.5329-02	-2.61
18	0.3924-02	-2.60
20	0.3009-02	-2.52

2 CYCLES THROUGH NEWNOT

N	MAX.ERROR	DECAY EXP.
4	0.1476+00	0.00
6	0.7954-01	-1.52
8	0.4185-01	-2.23
10	0.2240-01	-2.81
12	0.1259-01	-3.16
14	0.7395-02	-3.45
16	0.4526-02	-3.68
18	0.2878-02	-3.84
20	0.1891-02	-3.99

We try the following reasonable variant. Instead of starting afresh with uniform knots for each new N, we use the cubic spline approximation last computed for the preceding N to determine the initial knots for the next N. This requires the following modification of the earlier program between the DO 40 statement and the statement labeled 4.

```
CHAPTER XIII, EXAMPLE 2M. CUBIC SPLINE INTERPOLANT AT KNOT AVERAGES
C                              WITH GOOD KNOTS. MODIFIED AROUND LABEL 4.
CALLS SPLINT(BANFAC/SLV,BSPLVB),NEWNOT,BSPLPP(BSPLVB*)
      INTEGER I,ISTEP,ITER,ITERMX,N,NHIGH,NMAX,NMK,NLOW
      PARAMETER (NMAX=20)
      REAL ALGERP,ALOGER,BCOEF(NMAX+2),BREAK(NMAX),DECAY,DX,ERRMAX,
     *     C(4,NMAX),G,GTAU(NMAX),H,PNATX,SCRTCH(NMAX*7),STEP,T(NMAX+6),
     *     TAU(NMAX),TNEW(NMAX)
C     REAL ALGERP,ALOGER,BCOEF(22),BREAK(20),DECAY,DX,ERRMAX,
C    *     C(4,20),G,GTAU(20),H,PNATX,SCRTCH(140),STEP,T(26),
C    *     TAU(20),TNEW(20)
C     ISTEP AND STEP = FLOAT(ISTEP) SPECIFY POINT DENSITY FOR ERROR DET-
C     ERMINATION.
      DATA STEP, ISTEP /20., 20/
C                                    THE FUNCTION  G  IS TO BE INTERPOLATED .
      G(X) = SQRT(X+1.)
      DECAY = 0.
C      READ IN THE NUMBER OF ITERATIONS TO BE CARRIED OUT AND THE LOWER
C      AND UPPER LIMIT FOR THE NUMBER  N  OF DATA POINTS TO BE USED.
      READ 500,ITERMX,NLOW,NHIGH
  500 FORMAT(3I3)
      PRINT 600, ITERMX
  600 FORMAT(I4,22H CYCLES THROUGH NEWNOT//
     *       28H  N   MAX.ERROR   DECAY EXP./)
C                                     LOOP OVER  N = NUMBER OF DATA POINTS .
      DO 40 N=NLOW,NHIGH,2
         IF (N .EQ. NLOW)                    GO TO 4
         CALL NEWNOT ( BREAK, C, L, 4, TNEW, L+2, SCRTCH )
         L = L + 2
         T(5+L) = 1.
         T(6+L) = 1.
         ITER = 1
                                             GO TO 15

    4    NMK = N-4
         H = 2./FLOAT(NMK+1)
         DO 5 I=1,4
            T(I) = -1.
    5       T(N+I) = 1.
         IF (NMK .LT. 1)                     GO TO 10
```

```
      DO 6 I=1,NMK
    6    T(I+4) = FLOAT(I)*H - 1.
   10  ITER = 1
C                CONSTRUCT CUBIC SPLINE INTERPOLANT. THEN, ITERMX  TIMES,
C                 DETERMINE NEW KNOTS FROM IT AND FIND A NEW INTERPOLANT.
   11       DO 12 I=1,N
               TAU(I) = (T(I+1)+T(I+2)+T(I+3))/3.
   12          GTAU(I) = G(TAU(I))
            CALL SPLINT ( TAU, GTAU, T, N, 4, SCRTCH, BCOEF, IFLAG )
            CALL BSPLPP ( T, BCOEF, N, 4, SCRTCH, BREAK, C, L )
            IF (ITER .GT. ITERMX)        GO TO 19
            ITER = ITER + 1
            CALL NEWNOT ( BREAK, C, L, 4, TNEW, L, SCRTCH )
   15       DO 18 I=2,L
   18          T(3+I) = TNEW(I)
                                          GO TO 11
C                       ESTIMATE MAX.INTERPOLATION ERROR ON  (-1..1) .
   19  ERRMAX = 0.
C                                LOOP OVER POLYNOMIAL PIECES OF INTERPOLANT .
      DO 30 I=1,L
         DX = (BREAK(I+1)-BREAK(I))/STEP
C              INTERPOLATION ERROR IS CALCULATED AT  ISTEP  POINTS PER
C                  POLYNOMIAL PIECE .
         DO 30 J=1,ISTEP
            H = FLOAT(J)*DX
            PNATX = C(1,I)+H*(C(2,I)+H*(C(3,I)+H*C(4,I)/3.)/2.)
   30       ERRMAX = AMAX1(ERRMAX,ABS(G(BREAK(I)+H)-PNATX))

C                                                 CALCULATE DECAY EXPONENT .
      ALOGER = ALOG(ERRMAX)
      IF (N .GT. NLOW)  DECAY =
     *     (ALOGER - ALGERP)/ALOG(FLOAT(N)/FLOAT(N-2))
      ALGERP = ALOGER
C
   40 PRINT 640,N,ERRMAX,DECAY
  640 FORMAT(I3,E12.4,F11.2)
                                          STOP
      END
```

The resulting output is also listed above, to the right, and shows the desired $\mathcal{O}\left(N^{-4}\right)$ convergence after a few applications of NEWNOT. □

Interpolation at the Chebyshev-Demko sites While, according to R.Q. Jia [1988], use of knot averages may not guarantee a 'small' $\|I_k\|$, the analogue of the Chebyshev sites II(9) for polynomial interpolation is readily available for interpolation from $\$_{k,t}$ as S. Demko [1985] has pointed out. To be sure, use of these sites τ^c does not minimize the norm $\|I_k^c\|$ of the resulting spline interpolation map

$$I_k^c$$

over all possible choices τ of interpolation sites. But it does minimize the max-norm of the inverse A_τ^{-1} of the spline collocation matrix

$$A_\tau := (B_j(\tau_i) : i, j = 1, \ldots, n)$$

over all possible choices of τ and thereby (see Remark (23)) gives the **t-independent bound**

$$\|I_k^c\| \le D_{k,\infty}. \tag{17}$$

Here and below, the norm $\|C\|$ of a matrix C is taken as its max-norm, that is

$$\|C\| := \|C\|_\infty := \max_{\mathbf{x}} \frac{\|C\mathbf{x}\|}{\|\mathbf{x}\|} = \max_i \sum_j |C(i,j)| \tag{18}$$

with the vector norm

$$\|\mathbf{x}\| := \|\mathbf{x}\|_\infty := \max_i |x_i|.$$

In what is to follow, it will be important to recall from Theorem (7) that A_τ is totally positive, hence, by Lemma (8), the inverse of A_τ, if it exists, is necessarily checkerboard. In that case, the unique interpolant to any given g is $I_k^\tau g = \sum_j \alpha_j \mathrm{B}_{k,\tau}$ with $\boldsymbol{\alpha} = A_\tau^{-1}(g(\tau_j) : j = 1, \dots, n)$, hence

$$\|I_k^\tau g\| \le \|A_\tau^{-1}\| \|g\|.$$

Consequently,

$$\|I_k^\tau\| \le \|A_\tau^{-1}\|. \tag{19}$$

With this in mind, Demko proposes to choose as interpolation sites the sequence τ (if any) that minimizes $\|A_\tau^{-1}\|$. This turns out to be the sequence τ^c of extrema of the **Chebyshev spline** of order k for the knot sequence $\mathbf{t}$. This is, by definition, the function

$$T_{k,\mathbf{t}} := \sum_j \frac{(-)^{n-j}}{\operatorname{dist}(\mathrm{B}_j, \operatorname{span}(\mathrm{B}_i : i \ne j))} \mathrm{B}_j, \tag{20}$$

with the distance measured in the max-norm. The basis for this choice is the following lemma.

(21) Lemma. *Let* $f_* =: \sum_j \alpha_j^* \mathrm{B}_j$ *satisfy*

$$1 = (-)^{n-i} f_*(\tau_i^*) = \|f_*\|, \quad i = 1, \dots, n, \tag{22}$$

for some strictly increasing sequence τ^*. *Then*

$$(-)^{n-i}\alpha_i^* = 1/\operatorname{dist}(\mathrm{B}_i, \operatorname{span}(\mathrm{B}_j : j \ne i)), \quad i = 1, \dots, n.$$

In particular, f_* is uniquely determined by (22).

PROOF SKETCH. Let $A_* := A_{\tau^*}$ and set

$$\mu_i : \$_{k,\mathbf{t}} \to \mathbb{R} : f \mapsto \sum_j A_*^{-1}(i,j) f(\tau_j^*).$$

Then $\mu_i \mathrm{B}_j = \delta_{ij}$, hence, using the checkerboard nature of $A_{\tau^*}^{-1}$,

$$\begin{aligned}(-)^{n-i}\alpha_i^* = (-)^{n-i}\mu_i f_* &= \sum_j (-)^{n-i} A_*^{-1}(i,j)(-)^{n-j} \\ &= \sum_j |A_*^{-1}(i,j)| = \|\mu_i\|,\end{aligned}$$

while, for any linear functional μ on any normed linear space X and for any $f \in X$,

$$|\mu f| = \|\mu\| \operatorname{dist}(f, \ker \mu).$$

Hence, $1 = \operatorname{dist}(f_*, \ker \mu_i)$, while $\ker \mu_i = \operatorname{span}(\mathrm{B}_j : j \neq i)$, and so $|\alpha_i^*| \operatorname{dist}(\mathrm{B}_i : \ker \mu_i) = \operatorname{dist}(f_*, \ker \mu_i) = 1$. □

(23) Remark This shows that any $f_* \in \$_{k,\mathbf{t}}$ satisfying (22) is necessarily the Chebyshev spline $T_{k,\mathbf{t}}$, hence that, conversely, the Chebyshev spline, *if it exists*, has n local extrema, each of which is a global extremum, with neighboring extreme values having opposite signs. Also, $\boldsymbol{\alpha}^* = A_*^{-1}((-)^{n-i} : i = 1, \ldots, n)$, hence, as A_*^{-1} is checkerboard,

$$\|\boldsymbol{\alpha}^*\| = \|A_*^{-1}\|.$$

On the other hand, from B-spline Property XI(x) (see XI(4)),

$$\|\boldsymbol{\alpha}^*\| \leq D_{k,\infty}$$

since $\| \sum_j \boldsymbol{\alpha}_j^* \mathrm{B}_j \| = 1$. This, together with (19), proves (17). □

It remains to prove the *existence* of the Chebyshev spline.

(24) Proposition. *For $\tau = (\tau_1 < \cdots < \tau_n)$ with $t_i < \tau_i < \tau_{i+k}$, all i, let $f_{\mathbf{t}} =: \sum_j \boldsymbol{\alpha}_j^{\mathbf{t}}$ be the unique element of $\$_{k,\mathbf{t}}$ that satisfies $f_\tau(\tau_i) = (-)^{n-i}$, all i.*

Then, $\min_\tau \|\boldsymbol{\alpha}^\tau\|$ is taken on, at a τ for which f_τ satisfies the conditions (22) imposed on f_* in Lemma (21).

PROOF. The proof provides (not just coincidentally) an algorithm for the construction of the minimizing $\tau =: \tau^{\mathbf{c}}$. This algorithm is, essentially, the (second) Remez Algorithm for the construction of a best approximation in the uniform norm.

We start with the observation that

$$(-)^{n-i}\alpha_i^\tau > 0, \quad \text{all } i,$$

since $\alpha_i^\tau = \sum_j A_\tau^{-1}(i,j)(-)^{n-j} = (-)^{n-i}\sum_j |A_\tau^{-1}(i,j)|$, with the sum necessarily positive. Further, since f_τ strictly alternates in sign at the sequence $\tau_1, \ldots, \tau_n$ and vanishes outside $(t_1 \,..\, t_{n+k})$, it follows that f_τ has n distinct local extrema $\sigma_1 < \cdots < \sigma_n$, with

$$(-)^{n-i} f(\sigma_i) \geq 1, \quad \text{all } i.$$

This implies that $t_i < \sigma_i < t_{i+k}$, all i. Indeed, if, for example, $\sigma_i \leq t_i$ for some i, then $\mathrm{B}_\mu(\tau_\nu) = 0$ for all $\nu \leq i \leq \mu$, showing that, at $(\sigma_1, \ldots, \sigma_i)$, f_τ agrees with $\sum_{j<i} \alpha_j^\tau \mathrm{B}_j$. In particular, with Corollary XI(28), we would have $i-2 \geq \mathrm{S}^-(\alpha_1^\tau, \ldots, \alpha_{i-1}^\tau) \geq \mathrm{S}^-(\sum_{j<i} \alpha_j^\tau \mathrm{B}_j) = i-1$, which is nonsense.

Consequently, there is exactly one $f_\sigma =: \sum_j \alpha_j^\sigma \mathrm{B}_j$ with $f_\sigma(\sigma_i) = (-)^{n-i}$, all i. For any $\gamma < 1$, the difference, $f_\tau - \gamma f_\sigma$, strictly alternates in sign on $\sigma_1 < \cdots < \sigma_n$, hence we must have $\mathrm{S}^-(\boldsymbol{\alpha}^\tau - \gamma\boldsymbol{\alpha}^\sigma) = n-1$, and therefore

$$(-)^{n-i}\alpha_i^\tau \geq (-)^{n-i}\alpha_i^\sigma \geq 0, \quad \text{all } i. \tag{25}$$

Iteration of this process generates a sequence $f_m := \sum_j \alpha_j^m \mathrm{B}_j$, $m = 1, 2, \ldots$ (with $f_1 = f_\tau$, $f_2 = f_\sigma$), whose coefficients converge monotonely to some sequence $\boldsymbol{\alpha}^*$. The corresponding sequences $(\sigma_1^m < \cdots < \sigma_n^m)$ of extrema for f_m therefore also converge, necessarily to a strictly increasing sequence $\tau_1^* < \cdots < \tau_n^*$ since f_m strictly alternates in sign on $\sigma_1^m < \cdots < \sigma_n^m$.

It follows that the function $f^* := \sum_j \alpha_j^* \mathrm{B}_j$ satisfies (22) since $1 = (-)^{n-i} f_m(\sigma_i^m)$ while $\|f_m\| = \max_i |f_m(\sigma_i^m)|$ and f_* is the uniform limit of $(f_m : m = 1, 2, \ldots)$. By Lemma (21), this function is uniquely determined by the conditions (22). In particular, it is independent of the τ with which we started this process. On the other hand, by (25), $\|\boldsymbol{\alpha}^*\| \leq \|\boldsymbol{\alpha}^\tau\|$. □

A MATLAB program for the calculation of τ^c for given t can be found in an example in the SPLINE TOOLBOX (de Boor [1990]$_2$); see chebdem there. It turns out that a good choice for the τ to start off the iteration just described is the sequence of knot averages (15), and the resulting τ^c is often not all that different, so one might not even bother carrying out the iterations. The only difficulty with interpolation at knot averages is the unhappy fact that it is, in general, impossible to choose t *for given* (τ_i) so that (15) holds. For example, take $k = 3$ and $\tau = (1, 2, 5, 6)$. For this reason, we describe, finally,

Optimal interpolation We have seen now several examples of error estimates of the form

$$\|g - Sg\| \leq \text{const}_S \|g^{(k)}\| \tag{26}$$

with S some approximation scheme. Micchelli, Rivlin, & Winograd [1976] have determined, for given data sites τ, an interpolation scheme S for which const_S in (26) is as small as possible. This scheme, which we will denote by

$$\overset{\mathfrak{M}}{S} \qquad \text{(read "Cap. ess crowned")}$$

for the obvious reasons, turns out to be given by interpolation at the given data sites τ by splines of order k with its knot sequence $\mathbf{t} = (t_i)_1^{n+k}$ determined as follows. Let

$$[a\,..\,b] := [\tau_1\,..\,\tau_n],$$

choose $t_1 = \cdots = t_k = a$, $t_{n+1} = \cdots = t_{n+k} = b$, and choose the $n-k$ points $t_{k+1}, \ldots, t_n$ in $[a\,..\,b]$ as the breaks of the unique step function h for which

$$\begin{aligned} &|h(x)| = 1 \text{ for all } x \in [a\,..\,b], \text{ with } h(a^+) = 1, \\ &h \text{ has } \leq n-k \text{ sign changes in } [a\,..\,b], \\ &\int_a^b f(x)h(x)\,dx = 0 \text{ for every } f \in \$_{k,\tau}. \end{aligned} \tag{27}$$

The subprogram SPLOPT below calculates this knot sequence t for given τ, using Newton's method, starting with the initial guess

$$t_{k+i} = \tau_{ik}^* = (\tau_{i+1} + \cdots + \tau_{i+k-1})/(k-1), \qquad i = 1, \ldots, n-k. \tag{28}$$

This initial guess for the optimal knots is often very close to the optimum. This suggests the reasonable alternative of forgetting about the optimal knots altogether and using instead the knot choice (28) directly, together with the usual choice for the first and last k "boundary" knots. Such choice would certainly satisfy the conditions of the Schoenberg-Whitney Theorem.

Details concerning the program SPLOPT can be found in de Boor [1977]$_2$ from which the program is adapted.

```
      SUBROUTINE SPLOPT ( TAU, N, K, SCRTCH, T, IFLAG )
CALLS BSPLVB, BANFAC/SLV
COMPUTES THE KNOTS  T  FOR THE OPTIMAL RECOVERY SCHEME OF ORDER  K
C  FOR DATA AT  TAU(I), I=1,...,N .
C
C******  I N P U T  ******
C  TAU.....ARRAY OF LENGTH  N , CONTAINING THE INTERPOLATION POINTS.
C     A S S U M E D  TO BE NONDECREASING, WITH TAU(I).LT.TAU(I+K),ALL I.
C  N.....NUMBER OF DATA POINTS .
C  K.....ORDER OF THE OPTIMAL RECOVERY SCHEME TO BE USED .
C
C******  W O R K  A R R A Y  *****
C  SCRTCH.....ARRAY OF LENGTH  (N-K)(2K+3) + 5K + 3 . THE VARIOUS
C        CONTENTS ARE SPECIFIED IN THE TEXT BELOW .
C
C******  O U T P U T  ******
C  IFLAG.....INTEGER INDICATING SUCCESS (=1) OR FAILURE (=2) .
C     IF IFLAG = 1, THEN
C  T.....ARRAY OF LENGTH  N+K  CONTAINING THE OPTIMAL KNOTS READY FOR
C        USE IN OPTIMAL RECOVERY. SPECIFICALLY,  T(1) = ... = T(K) =
C        TAU(1)  AND  T(N+1) = ... = T(N+K) = TAU(N) , WHILE THE  N-K
C        INTERIOR KNOTS  T(K+1), ..., T(N)  ARE CALCULATED AS DESCRIBED
C        BELOW UNDER  *METHOD* .
C     IF IFLAG = 2, THEN
C        K .LT. 3, OR N .LT. K, OR A CERTAIN LINEAR SYSTEM WAS FOUND TO
C        BE SINGULAR.
C
C******  P R I N T E D  O U T P U T  ******
C  A COMMENT WILL BE PRINTED IN CASE  IFLAG = 2  OR NEWTON ITERATIONS
C  FAILED TO CONVERGE IN  N E W T M X  ITERATIONS .
C
C******  M E T H O D  ******
C     THE (INTERIOR) KNOTS  T(K+1), ..., T(N)  ARE DETERMINED BY NEWTONS
C  METHOD IN SUCH A WAY THAT THE SIGNUM FUNCTION WHICH CHANGES SIGN AT
C   T(K+1), ..., T(N)  AND NOWHERE ELSE IN  (TAU(1) .. TAU(N)) IS ORTHOGON-
C  AL TO THE SPLINE SPACE  SPLINE( K , TAU )  ON THAT INTERVAL .
C     LET  XI(J)  BE THE CURRENT GUESS FOR  T(K+J), J=1,...,N-K. THEN
C  THE NEXT NEWTON ITERATE IS OF THE FORM
C              XI(J)  +  (-)**(N-K-J)*X(J)  ,  J=1,...,N-K,
C  WITH  X  THE SOLUTION OF THE LINEAR SYSTEM
C                        C*X  =  D  .
C  HERE,  C(I,J) = B(I)(XI(J)), ALL J, WITH  B(I)  THE I-TH B-SPLINE OF
C  ORDER  K  FOR THE KNOT SEQUENCE  TAU , ALL I, AND  D  IS THE VECTOR
C  GIVEN BY  D(I) = SUM( -A(J) , J=I,...,N )*(TAU(I+K)-TAU(I))/K, ALL I,
C  WITH  A(I) = SUM ( (-)**(N-K-J)*B(I,K+1,TAU)(XI(J)) , J=1,...,N-K )
C  FOR I=1,...,N-1, AND  A(N) = -.5 .
C     (SEE CHAPTER  XIII  OF TEXT AND REFERENCES THERE FOR A DERIVATION)
C     THE FIRST GUESS FOR  T(K+J)  IS  (TAU(J+1)+...+TAU(J+K-1))/(K-1) .
C     ITERATION TERMINATES IF  MAX(ABS(X(J))) .LT. T O L  , WITH
C                  T O L  =  T O L R T E *(TAU(N)-TAU(1))/(N-K) ,
C  OR ELSE AFTER  N E W T M X  ITERATIONS , CURRENTLY,
C                  NEWTMX, TOLRTE / 10, .000001
C
      INTEGER IFLAG,K,N,   I,ID,INDEX,J,KM1,KPK,KPKM1,KPN,KP1,L,LEFT
     *,LEFTMK,LENW,LL,LLMAX,LLMIN,NA,NB,NC,ND,NEWTMX,NEWTON,NMK,NMKM1,NX
      REAL SCRTCH,T,TAU(N),   DEL,DELMAX,FLOATK,SIGN,SIGNST,SUM
     *                        ,TOL,TOLRTE,XIJ
      DIMENSION SCRTCH((N-K)*(2*K+3)+5*K+3), T(N+K)
      DATA NEWTMX,TOLRTE / 10,.000001/
      NMK = N-K
      IF (NMK)                                  1,56,2
    1 PRINT 601,N,K
  601 FORMAT(13H ARGUMENT N =,I4,29H IN  SPLOPT  IS LESS THAN K =,I3)
                                                GO TO 999
```

```
  2 IF (K .GT. 2)                          GO TO 3
    PRINT 602,K
602 FORMAT(13H ARGUMENT K =,I3,27H IN  SPLOPT  IS LESS THAN 3)
                                           GO TO 999
  3 NMKM1 = NMK - 1
    FLOATK = K
    KPK = K+K
    KP1 = K+1
    KM1 = K-1
    KPKM1 = KPK-1
    KPN = K+N
    SIGNST = -1.
    IF (NMK .GT. (NMK/2)*2)  SIGNST = 1.
C   SCRTCH(I) = TAU-EXTENDED(I), I=1,...,N+K+K
    NX = N+KPK+1
C   SCRTCH(I+NX) = XI(I),I=0,...,N-K+1
    NA = NX + NMK + 1
C   SCRTCH(I+NA) = -A(I), I=1,...,N
    ND = NA + N
C   SCRTCH(I+ND) = X(I) OR D(I), I=1,...,N-K
    NB = ND + NMK
C   SCRTCH(I+NB) = BIATX(I),I=1,...,K+1
    NC = NB + KP1
C   SCRTCH(I+(J-1)*(2K-1) + NC) = W(I,J) = C(I-K+J,J), I=J-K,...,J+K,
C                                                      J=1,...,N-K.
    LENW = KPKM1*NMK
C   EXTEND  TAU  TO A KNOT SEQUENCE AND STORE IN SCRTCH.
    DO 5 J=1,K
       SCRTCH(J) = TAU(1)
 5     SCRTCH(KPN+J) = TAU(N)
    DO 6 J=1,N
 6     SCRTCH(K+J) = TAU(J)
C   FIRST GUESS FOR  SCRTCH (.+NX)  =  XI .
    SCRTCH(NX) = TAU(1)
    SCRTCH(NMK+1+NX) = TAU(N)
    DO 10 J=1,NMK
       SUM = 0.
       DO 9 L=1,KM1
 9        SUM = SUM + TAU(J+L)
10     SCRTCH(J+NX) = SUM/FLOAT(KM1)
C   LAST ENTRY OF  SCRTCH (.+NA)  =  - A  IS ALWAYS ...
    SCRTCH(N+NA) = .5
C   START NEWTON ITERATION.
    NEWTON = 1
    TOL = TOLRTE*(TAU(N) - TAU(1))/FLOAT(NMK)
C   START NEWTON STEP
COMPUTE THE 2K-1 BANDS OF THE MATRIX C AND STORE IN SCRTCH(.+NC),
C   AND COMPUTE THE VECTOR  SCRTCH(.+NA) = -A.
20  DO 21 I=1,LENW
21     SCRTCH(I+NC) = 0.
    DO 22 I=2,N
22     SCRTCH(I-1+NA) = 0.
    SIGN = SIGNST
    LEFT = KP1
    DO 28 J=1,NMK
       XIJ = SCRTCH(J+NX)
23        IF (XIJ .LT. SCRTCH(LEFT+1))GO TO 25
          LEFT = LEFT + 1
          IF (LEFT .LT. KPN)           GO TO 23
          LEFT = LEFT - 1
25     CALL BSPLVB(SCRTCH,K,1,XIJ,LEFT,SCRTCH(1+NB))
C         THE TAU SEQUENCE IN SCRTCH IS PRECEDED BY  K  ADDITIONAL KNOTS
C         THEREFORE,  SCRTCH(LL+NB)  NOW CONTAINS  B(LEFT-2K+LL)(XIJ)
C         WHICH IS DESTINED FOR  C(LEFT-2K+LL,J), AND THEREFORE FOR
```

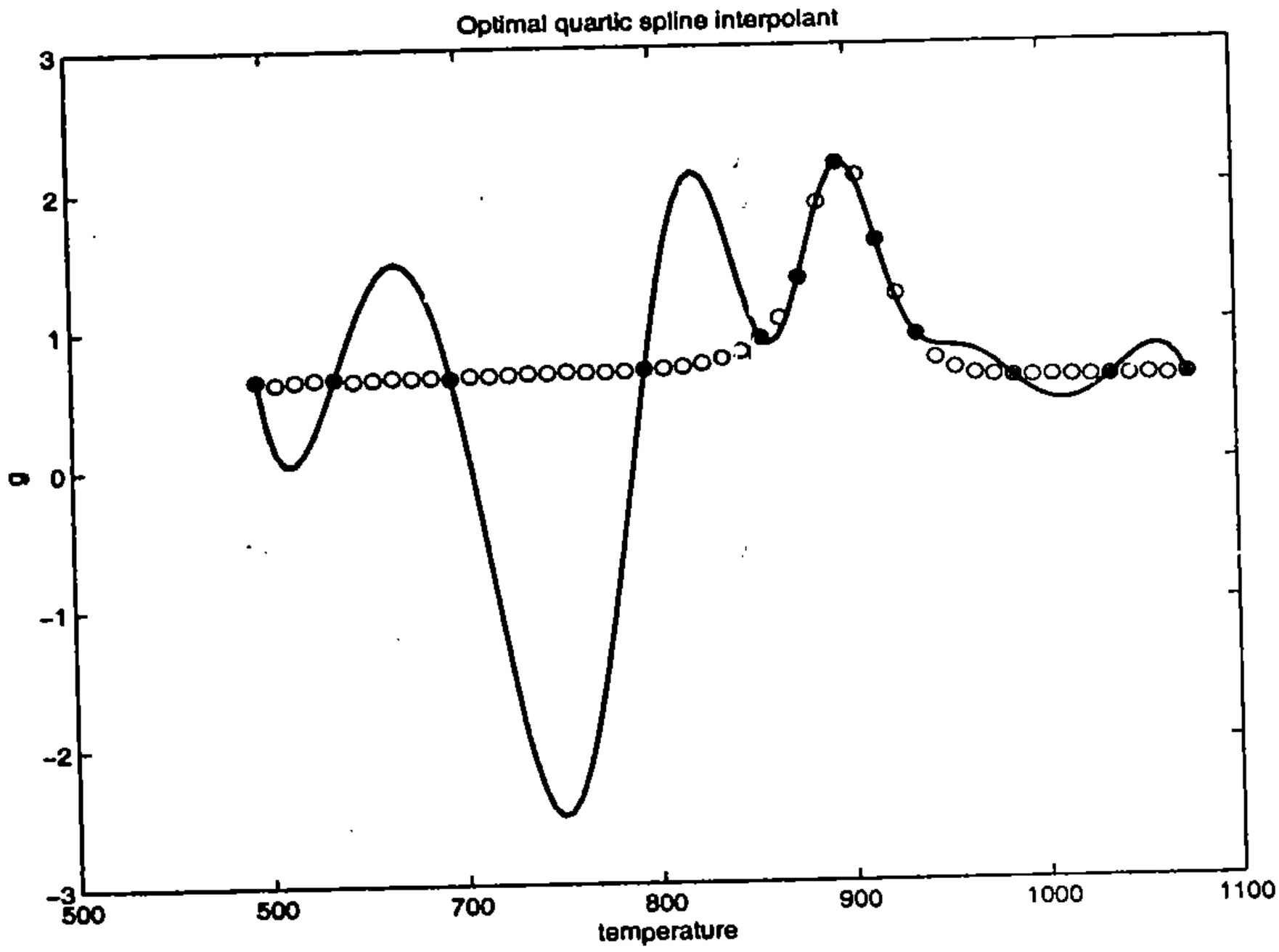
```
C         W(LEFT-K-J+LL,J)= SCRTCH(LEFT-K-J+LL + (J-1)*KPKM1 + NC)
C      SINCE WE STORE THE 2K-1 BANDS OF  C  IN THE 2K-1  R O W S  OF
C      THE WORK ARRAY W, AND  W  IN TURN IS STORED IN  S C R T C H ,
C      WITH  W(1,1) = SCRTCH(1 + NC) .
C          ALSO, C  BEING OF ORDER  N-K, WE WOULD WANT  1 .LE.
C      LEFT-2K+LL .LE. N-K  OR
C         LLMIN = 2K-LEFT  .LE.  LL  .LE.  N-LEFT+K = LLMAX .
       LEFTMK = LEFT-K
       INDEX = LEFTMK-J + (J-1)*KPKM1 + NC
       LLMIN = MAXO(1,K-LEFTMK)
       LLMAX = MINO(K,N-LEFTMK)
       DO 26 LL=LLMIN,LLMAX
    26    SCRTCH(LL+INDEX) = SCRTCH(LL+NB)
       CALL BSPLVB(SCRTCH,KP1,2,XIJ,LEFT,SCRTCH(1+NB))
       ID = MAXO(0,LEFTMK-KP1)
       LLMIN = 1 - MINO(0,LEFTMK-KP1)
       DO 27 LL=LLMIN,KP1
          ID = ID + 1
    27     SCRTCH(ID+NA) = SCRTCH(ID+NA) - SIGN*SCRTCH(LL+NB)
    28 SIGN = -SIGN
    CALL BANFAC(SCRTCH(1+NC),KPKM1,NMK,KM1,KM1,IFLAG)
                                        GO TO (45,44),IFLAG
    44 PRINT 644
   644 FORMAT(32H C IN  SPLOPT  IS NOT INVERTIBLE)
                                        RETURN
COMPUTE  SCRTCH (.+ND) =  D  FROM  SCRTCH (.+NA) = - A .
    45 I=N
    46    SCRTCH(I-1+NA) = SCRTCH(I-1+NA) + SCRTCH(I+NA)
          I = I-1
          IF (I .GT. 1)                 GO TO 46
       DO 49 I=1,NMK
    49    SCRTCH(I+ND) = SCRTCH(I+NA)*(TAU(I+K)-TAU(I))/FLOATK
COMPUTE  SCRTCH (.+ND) =  X .
       CALL BANSLV(SCRTCH(1+NC),KPKM1,NMK,KM1,KM1,SCRTCH(1+ND))
COMPUTE  SCRTCH (.+ND) = CHANGE IN  XI . MODIFY, IF NECESSARY, TO
C  PREVENT NEW  XI  FROM MOVING MORE THAN 1/3 OF THE WAY TO ITS
C  NEIGHBORS. THEN ADD TO  XI  TO OBTAIN NEW  XI  IN SCRTCH(.+NX).
       DELMAX = 0.
       SIGN = SIGNST
       DO 53 I=1,NMK
          DEL = SIGN*SCRTCH(I+ND)
          DELMAX = AMAX1(DELMAX,ABS(DEL))
          IF (DEL .GT. 0.)              GO TO 51
          DEL = AMAX1(DEL,(SCRTCH(I-1+NX)-SCRTCH(I+NX))/3.)
                                        GO TO 52
    51    DEL = AMIN1(DEL,(SCRTCH(I+1+NX)-SCRTCH(I+NX))/3.)
    52    SIGN = -SIGN
    53    SCRTCH(I+NX) = SCRTCH(I+NX) + DEL
CALL IT A DAY IN CASE CHANGE IN  XI  WAS SMALL ENOUGH OR TOO MANY
C  STEPS WERE TAKEN.
       IF (DELMAX .LT. TOL)             GO TO 54
       NEWTON = NEWTON + 1
       IF (NEWTON .LE. NEWTMX)          GO TO 20
       PRINT 653,NEWTMX
   653 FORMAT(33H NO CONVERGENCE IN  SPLOPT  AFTER,I3,14H NEWTON STEPS.)
    54 DO 55 I=1,NMK
    55    T(K+I) = SCRTCH(I+NX)
    56 DO 57 I=1,K
          T(I) = TAU(1)
    57    T(N+I) = TAU(N)
                                        RETURN
   999 IFLAG = 2
                                        RETURN
       END
```

(29) Example: "Optimal" interpolation need not be "good" We consider a function g that describes a property of titanium as a function of temperature and whose values $g(\tau_i)$ at $\tau_i = 585 + 10i$, $i = 1, \ldots, 49$, we know from a table (experimentally determined). The following subprogram provides these values (except that the second .644 should have been .664).

```
      SUBROUTINE TITAND ( TAU, GTAU, N )
C  THESE DATA REPRESENT A PROPERTY OF TITANIUM AS A FUNCTION OF
C  TEMPERATURE. THEY HAVE BEEN USED EXTENSIVELY AS AN EXAMPLE IN SPLINE
C  APPROXIMATION WITH VARIABLE KNOTS.
      INTEGER N,   I
      REAL GTAU(49),TAU(49),GTITAN(49)
      DATA GTITAN /.644,.622,.638,.649,.652,.639,.646,.657,.652,.655,
     2             .644,.663,.663,.668,.676,.676,.686,.679,.678,.683,
     3             .694,.699,.710,.730,.763,.812,.907,1.044,1.336,1.881,
     4            2.169,2.075,1.598,1.211,.916,.746,.672,.627,.615,.607
     5           ,.606,.609,.603,.601,.603,.601,.611,.601,.608/
      N = 49
      DO 10 I=1,N
         TAU(I) = 585. + 10.*FLOAT(I)
   10    GTAU(I) = GTITAN(I)
                                                  RETURN
      END
```

A plot of these data (see Figure (30)) shows g to be quite flat ($\sim .6$) except for a peak around 905 where it rises to a value of ~ 2.2. We pick 12 data points somewhat more densely distributed around the peak and use optimal interpolation.

```
CHAPTER XIII, EXAMPLE 3 . TEST OF OPTIMAL SPLINE INTERPOLATION ROUTINE
C                          ON TITANIUM HEAT DATA .
CALLS TITAND,SPLOPT(BSPLVB,BANFAC/SLV),SPLINT(*),BVALUE(INTERV)
C
C     DATA N,K /12,5/
C     LENSCR = (N-K)(2K+3)+5K+3  IS THE LENGTH OF SCRTCH REQUIRED IN
C                                SPLOPT .
      INTEGER K,LENSCR,N,NTITAN
      PARAMETER (N=12,NTITAN=49,K=5,LENSCR=(N-K)*(2*K+3)+5*K+3)
      INTEGER I,IFLAG,IPICK(N),IPICKI,LX,NMK
      REAL A(N),GTITAN(NTITAN),GTAU(NTITAN),SCRTCH(LENSCR),T(N+K),TAU(N)
     *    ,X(NTITAN)
C     INTEGER I,IFLAG,IPICK(12),IPICKI,LX,NMK
C     REAL A(12),GTITAN(49),GTAU(49),SCRTCH(119),T(17),TAU(12)
C    *    ,X(49)
      DATA IPICK /1,5,11,21,27,29,31,33,35,40,45,49/
      CALL TITAND ( X, GTITAN, LX )
      DO 10 I=1,N
         IPICKI = IPICK(I)
         TAU(I) = X(IPICKI)
   10    GTAU(I) = GTITAN(IPICKI)
      CALL SPLOPT ( TAU, N, K, SCRTCH, T, IFLAG )
      IF (IFLAG .GT. 1)                    STOP
      CALL SPLINT ( TAU, GTAU, T, N, K, SCRTCH, A, IFLAG )
      IF (IFLAG .GT. 1)                    STOP
      DO 20 I=1,LX
         GTAU(I) = BVALUE ( T, A, N, K, X(I), 0 )
   20    SCRTCH(I) = GTITAN(I) - GTAU(I)
```



(30) FIGURE. An optimal quartic spline interpolant (solid) to given data (*) at twelve points taken from a data sequence (o) of 49 points.

```
    PRINT 620,(I,X(I),GTITAN(I),GTAU(I),SCRTCH(I),I=1,LX)
620 FORMAT(41H  I, DATA POINT, DATA, INTERPOLANT, ERROR//
   2          (I3,F8.0,F10.4,F9.4,E11.3))
    NMK = N-K
    PRINT 621,(I,T(K+I),I=1,NMK)
621 FORMAT(///16H OPTIMAL KNOTS =/(I5,F15.9))
                                          STOP

    END
```

```
 OPTIMAL KNOTS =
    1  730.985412598
    2  794.413757324
    3  844.476440430
    4  880.059509277
    5  907.814086914
    6  938.000488281
    7  976.751708984
```

Instead of the full output, we give here a picture (Figure (30)) showing data and approximation. The points of interpolation are easily picked out since the approximation oscillates so badly around the flat part, having been set off by the peak. Clearly, more information about the flat part

needs to be made available to the approximation process (see Problem 5). Using a lower order spline would help, too (see Example XVI(17)). □

The optimal interpolation scheme was also derived independently by Gaffney & Powell [1976]. In fact, the term "optimal interpolation" for it is due to them; Micchelli, Rivlin & Winograd [1976] call it an "optimal recovery scheme". Gaffney & Powell came to it by a way of "envelopes", and we now discuss this idea briefly since it gives us an excuse to question the basic notion underlying the optimality aspects of spline interpolation, namely the notion that making a derivative smaller makes the function smoother.

Gaffney & Powell [1976] investigate in detail the following question posed earlier in more generality by Golomb & Weinberger [1959]:

Given that we know the values $g(\tau_1), \ldots, g(\tau_n)$ *and the bound* $\|g^{(k)}\| \le \alpha$, *what can we say about the value* $g(x)$ *of* g *at some site* x (other than $\tau_1, \ldots, \tau_n$)?

The answer turns out to be surprisingly elegant:

(31) Theorem (Gaffney & Powell [1976]). *Let*

$$F_\alpha := \{\, f : f(\tau_i) = g(\tau_i), i = 1, \ldots, n;\ \|f^{(k)}\| \le \alpha \,\}$$

be the set of all functions on $[a\,..\,b]$ *with* k *derivatives that agree with the given function* g *at* τ *and whose* k*th derivative is uniformly bounded by* α.

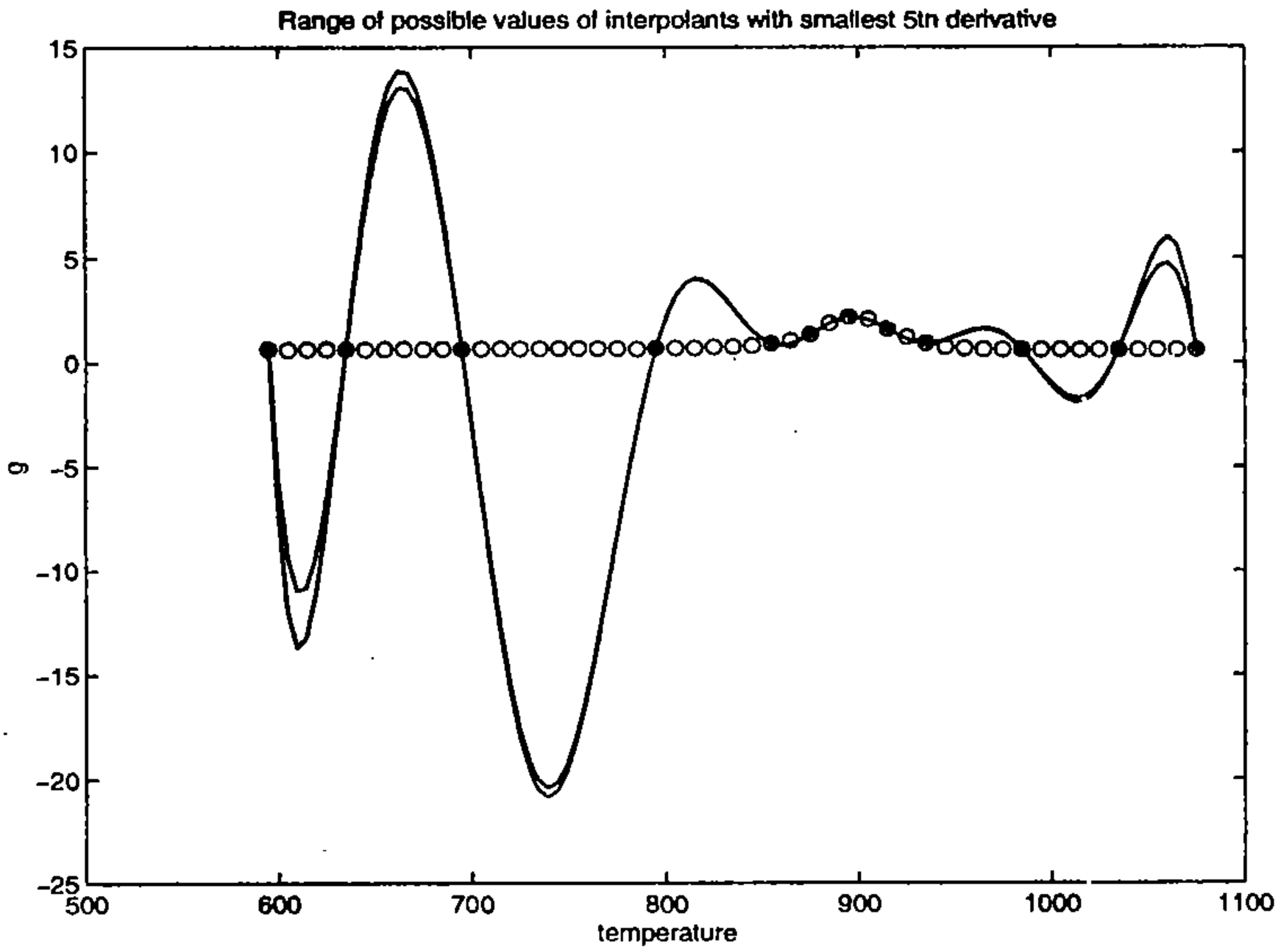
If F_α *is not empty, then it contains two function* f_1 *and* f_2 *with the property that, for any* x *and any* $f \in F_\alpha$, $f(x)$ *lies between* $f_1(x)$ *and* $f_2(x)$. *Both* f_1 *and* f_2 *are* **perfect splines** *of* **degree** k **with** $n-k$ **knots**, *that is, both are splines of order* $k+1$ *with* $n-k$ *simple (interior) knots and with absolutely constant* k*th derivative, that is,* $|(D^k f_1)(x)| = |(D^k f_2)(x)| = \alpha$, *all* x.

Gaffney & Powell come to the optimal interpolant by the following device. For each α, for which F_α is not empty, the *average* $(f_1 + f_2)/2$ of the two envelope functions f_1 and f_2 is also in F_α, and is, in a sense, its *center*. This average or center does depend on α, but, as α grows without bound, the average converges and, as it turns out, its limit is the optimal interpolant $\mathfrak{M}\mathcal{S}\, g$ to g discussed earlier.

For the promised questioning, through, we want to make α as *small* as possible. It is clear that we cannot make α arbitrarily small in general. For example, for F_α to be nonempty, we would need $k!\alpha \ge \max_i |[\tau_i, \ldots, \tau_{i+k}]g|$ since, by divided difference Property I(vii), $[\tau_i, \ldots, \tau_{i+k}]g/k! = g^{(k)}(\zeta)$ for some ζ in the interval containing $\tau_i, \ldots, \tau_{i+k}$. It can be shown (see Favard [1940], de Boor [1976]$_4$) that there is a smallest α for which F_α is not empty,

$$\alpha^* := \min\{\, \alpha : F_\alpha \neq \emptyset \,\}.$$

An $f \in F_{\alpha^*}$ is a "best interpolant" for g in the sense that, among all in-



(32) FIGURE. All "best" interpolants to the selected 12 from the given 49 data points must lie between the upper and lower envelope (solid lines) shown.

terpolants to g, its kth derivative is as small as possible. Such interpolants were first studied by Favard [1940] who developed (as explained in complete detail in de Boor [1976]$_4$) a reasonable way for picking from the possibly many best interpolants one that is best (or, better than best) among them. But, it and all other best interpolants would have to lie between the envelope for F_{α^*}. I am indebted to Patrick Gaffney for the information that the value $\alpha = .1838 \times 10^{-5}$ is very close to α^* for the data actually interpolated in Example (29). The envelope function for that value of α is shown in Figure (32) along with the Titanium Heat data from which it is taken. Farvard's best interpolant and all other "best" interpolants must lie between those strongly oscillating bounds!! Clearly, the price being paid here for a uniformly small 5th derivative (we took $k = 5$ here as in Example (29)) is a "smooth" oscillation. In effect, since a zero 5th derivative implies that the function is a polynomial of order 5, we are looking for the interpolant that is most polynomial-like, that is, we are ignoring the reasons that led us in Chapter II to *piecewise* polynomial functions.

Osculatory spline interpolation As in polynomial interpolation (see Chapter I), it is possible in spline interpolation to let some of the inter-

polation sites coalesce and so achieve osculatory or repeated interpolation, that is, a matching to value *and* some of the derivatives. To recall, r-fold interpolation at a point τ means matching of derivatives from the zero-th up to but not including the rth at τ. The condition for existence and uniqueness of an interpolant stays exactly the same as in the case of simple interpolation points.

(33) Theorem (Karlin & Ziegler [1966]). *Given the nondecreasing sequence* $\boldsymbol{\tau} = (\tau_i)_1^n$ *with* $\tau_i < \tau_{i+k}$, *all* i, *assume that*

$$t_k < \tau_{i+1} = \cdots = \tau_{i+r} = t_{j+1} = \cdots = t_{j+s} < t_{n+1} \quad \textit{implies} \quad r + s \le k.$$

Then, for any smooth function g, *there exists exactly one* $f \in \$_{k,\mathrm{t}}$ *that agrees with* g *at* τ *if and only if* $\tau_i \in \operatorname{supp} \mathrm{B}_i = (t_i \,..\, t_{i+k})$, *all* i.

A simple proof of this theorem can be found in de Boor [1976]$_1$.

It is not hard to extend the subprogram SPLINT to construct osculatory interpolants, but we do not take the time for it here. Instead, we discuss briefly a limiting case of even order spline interpolation at knots, namely **complete spline interpolation.** In this scheme, the order is even,

$$k =: 2m,$$

and the data site sequence is $a := \tau_1 = \cdots = \tau_m < \tau_{m+1} < \cdots < \tau_{n-m+1} = \cdots = \tau_n =: b$. Correspondingly, one chooses the knot sequence

$$a := t_1 = \cdots = t_k < t_{k+1} < \cdots < t_n < t_{n+1} = \cdots = t_{n+k} =: b$$

with

$$t_{k+i} = \tau_{m+i}, \qquad i = 1, \ldots, n-k \quad (= n - 2m).$$

This is the customary generalization of complete cubic spline interpolation discussed in Chapters IV and V. We take the occasion to prove here its best approximation properties.

(34) Theorem (Best approximation properties of complete spline interpolation). *Let* $\mathrm{t} = (t_i)_1^{n+k}$ *with* $k = 2m$, $a = t_1 = \cdots = t_k < t_{k+1} \le t_{k+2} \le \cdots \le t_n < t_{n+1} = \cdots = t_{n+k} = b$ *and* $t_i < t_{i+m}$ *for* $k \le i \le n+1-m$. *Denote by*

$$I_k g$$

the complete spline interpolant to g *from* $\$_{k,\mathrm{t}}$, *that is,* $I_k g$ *is the unique element in* $\$_{k,\mathrm{t}}$ *that agrees with* g *at* $\boldsymbol{\tau} = (\tau)_1^n := (t_i)_{m+1}^{n+m}$. *Then:*

(i) The m*th derivative* $D^m I_k g$ *of the interpolant is the least-squares approximation from* $\$_{m,\boldsymbol{\tau}}$ *to the* m*th derivative of* g, *that is,*

$$\textbf{(35)} \quad \int_a^b \left(D^m(g - I_k g)(x)\right)^2 \mathrm{d}x \le \int_a^b \left((D^m g - f)(x)\right)^2 \mathrm{d}x \quad \textbf{for all } f \in \$_{m,\boldsymbol{\tau}}$$

with equality if and only if $f = D^m I_k g$.

(ii) *Also, among all functions f with a piecewise continuous mth derivative that agree with the function g at τ, the particular function $I_k g$ is* **smoothest** *in the sense that it uniquely minimizes*

$$\int_a^b \left(D^m f(x)\right)^2 \, \mathrm{d}x.$$

PROOF. The proof parallels the arguments given in Chapter V for complete spline interpolation. In particular, the proof relies on the following transcription

$$[\tau_i, \ldots, \tau_{i+m}]f = \frac{m}{\tau_{i+m} - \tau_i} \int \mathrm{B}_{i,m,\tau}(s)(D^m f)(s) \, \mathrm{d}s/m! \tag{36}$$

of IX(6) using IX(5).

Apply (36) to the error

$$e := g - I_k g,$$

a function that vanishes at $\tau_i, \ldots, \tau_{i+m}$, to get that

$$\int_a^b \mathrm{B}_{i,m,\tau}(s)(D^m e)(s) \, \mathrm{d}s = (\tau_{i+m} - \tau_i)(m-1)![\tau_i, \ldots, \tau_{i+m}]e = 0, \qquad i = 1, \ldots, n.$$

This shows the mth derivative of the error to be orthogonal to the linear span of $(\mathrm{B}_{i,m,\tau})_1^n$, that is, to $\$_{m,\tau}$. Since, on the other hand,

$$D^m I_k g \in \{ D^m f : f \in \$_{k,\mathbf{t}} \} = \$_{m,\tau} \qquad (\text{on } [a \,..\, b]),$$

this proves that $D^m e$ is, in particular, orthogonal to $D^m I_k g$, and so we obtain, once again, a special instance of Pythagoras' Theorem:

$$\int_a^b \left(D^m g(x)\right)^2 \, \mathrm{d}x = \int_a^b \left(D^m I_k g(x)\right)^2 \, \mathrm{d}x + \int_a^b \left(D^m (g - I_k g)(x)\right)^2 \, \mathrm{d}x, \tag{37}$$

valid for all g with a piecewise continuous mth derivative.

Now, to prove (i), substitute $g - f$ for g in (37) and use the fact that $I_k f = f$ for $f \in \$_{k,\mathbf{t}}$. To prove (ii), use the fact that $I_k f = I_k g$ for any f that agrees with g at τ. □

These best approximation properties of complete spline interpolation are still, for many, the major justification for using splines. I have bothered to mention them here only because the proof brings out the important connection (36) between divided differences and B-splines. In preparation for the next chapter, we now indicate the following important consequence of (36).

(38) Lemma. *If* $\mathbf{t} = (t_i)_1^{n+k}$ *is in* $[a\,..\,b]$, *nondecreasing, with* $t_i < t_{i+k}$ *for all* i, *and the integrable function* g *is orthogonal to* $\$_{k,\mathbf{t}}$ *on* $[a\,..\,b]$, *that is,*

$$\int_a^b f(x)g(x)\,\mathrm{d}x = 0 \quad \text{for all} \quad f \in \$_{k,\mathbf{t}},$$

then there exists $\boldsymbol{\xi} = (\xi_i)_1^{n+1}$ *in* $[a\,..\,b]$, *strictly increasing, with* $t_i \le \xi_i \le t_{i+k-1}$ *(any equality holding only in the trivial case when* $t_i = t_{i+k-1}$*),* $i = 1, \ldots, n+1$, *so that* g *is also orthogonal to* $\$_{1,\boldsymbol{\xi}}$.

PROOF. We can always choose a polynomial p of order k so that the function

$$G(x) := p(x) + \int_a^b (x-s)_+^{k-1} g(s)\,\mathrm{d}s/(k-1)!$$

vanishes at $t_1, \ldots, t_k$. Further, by (36),

$$[t_i, \ldots, t_{i+k}]G = \int_a^b \frac{\mathrm{B}_{i,k,\mathbf{t}}(s)g(s)}{(k-1)!(t_{i+k}-t_i)}\,\mathrm{d}s,$$

hence vanishes by assumption. Consequently, G vanishes at *all* points of $\mathbf{t}$. Since G is also $k-1$ times continuously differentiable, $(k-1)$-fold application of Rolle's Theorem proves the existence of a strictly increasing sequence $\boldsymbol{\xi} = (\xi_1^{n+1})$ with $t_i \le \xi_i \le t_{i+k-1}$, all i, at which

$$D^{k-1}G = \text{const} + \int_a^b (\cdot - s)_+^0 g(s)\,\mathrm{d}s$$

vanishes, which proves the lemma. □

The following corollary is of particular interest.

(39) Corollary. *If, in addition, the function* g *is continuous, then there exists* $\boldsymbol{\tau} = (\tau_i)_1^n$ *strictly increasing, with* $t_i < \tau_i < t_{i+k}$, *all* i, *so that* $g(\tau_i) = 0$, $i = 1, \ldots, n$.

PROOF. Lemma (38) says that

$$\int_{\xi_i}^{\xi_{i+1}} g(x)\,\mathrm{d}x = 0$$

for $i = 1, \ldots, n$, hence, if g is continuous then it must vanish somewhere in each of these n disjoint intervals $(\xi_i\,..\,\xi_{i+1})$. □

This corollary proves that any least-squares approximation from $\$_{k,t}$ to some continuous function g can also be gotten by spline interpolation (apply the corollary to the error in the least-square approximation) at some *appropriate* n sites that depend on g. In other words, there isn't all that much difference between interpolation and least-squares approximation *if* one knows which sites to choose. In a way, least-squares approximation is a very reasonable way to choose the interpolation sites.

Problems

1. (Another proof of the Schoenberg-Whitney Theorem)
(i) Prove: If $B_v(\tau_v) = 0$, that is, if $\tau_v \leq t_v$ or $t_{v+k} \leq \tau_v$, for some v, then $(B_j(\tau_i))$ is not invertible. (Hint: Assume without loss that $\tau_v \leq t_v$ and show that then $B_s(\tau_r) = 0$ for $1 \leq r \leq v \leq s \leq n$; conclude that columns $v, v+1, \ldots, n$ of $(B_j(\tau_i))$ form a linearly dependent sequence.)

Now assume, conversely, that $t_v < \tau_v < t_{v+k}$, all v.
(ii) Show that it is sufficient to prove the invertibility of $(B_j(\tau_i))$ under the stronger assumption that $t_{v+1} \leq \tau_v \leq t_{v+k-1}$, all v. (Hint: Show that the invertibility of $(B_j(\tau_i))$ follows from the invertibility of the two *smaller* matrices $(B_j(\tau_i))_{i,j=1}^{v}$ and $(B_j(\tau_i))_{i,j=v+1}^{n}$ in case $\tau_v \leq t_{v+1}$ or $t_{v+k} \leq \tau_{v+1}$.)
(iii) Prove the invertibility of $(B_{j,k}(\tau_i))$ by induction on the order k, using (ii) and Rolle's Theorem.

2. (i) Show that the tridiagonal (square) matrix with general row 1, 1, 1 is not totally positive.
(ii) Use Theorem (7) to show that the tridiagonal matrix with general row 1, 4, 1 is totally positive.

3. Give a proof of Lemma (11). (Hint: Write $Ig = \sum_1^n \alpha_i B_{i,k}$ and use earlier material to verify that $[\tau_i, \tau_{i+1}]g = [\tau_i, \tau_{i+1}](Ig) = (DIg)(\xi_i)$ for some ξ_i between τ_i, τ_{i+1}, while then

$$|(DIg)(\xi_i)| = |(k-1)\sum_j \frac{\alpha_j - \alpha_{j-1}}{t_{j+k-1} - t_j} B_{j,k-1}(\xi_i)| \leq 2(k-1)\|\boldsymbol{\alpha}\|/d_i,$$

with $d_i := \min\{\, t_{j+k-1} - t_j \,:\, (t_j \,..\, t_{j+k-1}) \cap (\tau_i \,..\, \tau_{i+1}) \neq \emptyset \,\}$. Since $\|\boldsymbol{\alpha}\| \leq D_{k,\infty}\|Ig\|$, one gets from this $2(\Delta\tau_i)^{-1} = \max_g |[\tau_i, \tau_{i+1}]g|/\|g\| \leq \max_g (2(k-1)/d_i) D_{k,\infty}\|Ig\|/\|g\| = (2(k-1)D_{k,\infty}/d_i)\|I\|$, etc.)

4. Use Lemma (11) to show by an example that the choice of knots at data site averages does *not* avoid large $\|I\|$. Show also that the lower bound for $\|I\|$ in that lemma stays bounded when we choose, instead, the data sites at knot averages, that is, when $\tau_i = t_{ik}^* = (\sum_{j=1}^{k-1} t_{i+j})/(k-1)$, all i.

5. For comparison, construct the optimal interpolant of order 3 and of order 4 in Example (29). Also, construct the optimal interpolant of order 5, but bring in more data points from the flat part, especially from the two

transition regions next to the peak.

6. (i) Prove that the norm $\|I\|$ for spline interpolation at τ by $f \in \$_{k,\mathbf{t}}$ can be calculated as $\|\Lambda\|$, with $\Lambda(x) := \sum_1^n |C_i(x)|$ the **Lebesgue function** for the process, that is, C_i is the unique element in $\$_{k,\mathbf{t}}$ satisfying $C_i(\tau_j) = \delta_{ij}$, all j.
(ii) Prove that $C_i = \sum_j \beta_{ij} \mathrm{B}_j$, with (β_{ij}) the matrix inverse to the matrix $(\mathrm{B}_j(\tau_i))$.
(iii) Use Theorem (7) to show that the β_{ij} alternate in sign, that is, $(-)^{i+j}\beta_{ij} \geq 0$, all i, j.
(iv) Given that C_i changes sign only at the sites τ_j for $j \neq i$, set up a program to calculate (approximately) $\|I\| = \|\Lambda\|$. Then use it to get as large a lower bound for $\max_{\mathbf{t}} \|I_4^*\|$ as you can. For what kind of knot sequence $\mathbf{t}$, do you think, is this maximum taken on?

7. With $\lambda_h g := (g(a+h) - g(a))/h$, show that, for any particular choice of g and a and all small h, the computed value of $\lambda_h g$ is not "large" (as maintained on page 185) but, invariably, 0.

8. Rewrite SPLINT to make use of CWIDTH (see the Appendix for a listing, but change it to inhibit pivoting) since this would require roughly only *half* the storage used by BANFAC/SLV.

9. An **isolated zero** of a function f is, by definition, any point z that is the sole site in some neighborhood of z at which the function f vanishes.
Prove that any 'isolated' zero of $f \in \$_{k,\mathbf{t}}$ (as defined on p. 172) is an isolated zero of f, but that the converse does not hold.

XIV

Smoothing and Least-Squares Approximation;

SMOOTH, L2APPR

Interpolation is effective as a means of constructing spline approximation only if sufficiently accurate function values are available. Otherwise, other methods of approximation must be used. In this chapter, we discuss the computational aspects of cubic smoothing splines, and of (discrete) least squares approximation by splines of arbitrary order.

The smoothing spline of Schoenberg and Reinsch Given the approximate values $y_i = g(x_i) + \varepsilon_i$ of some supposedly smooth function g at data sites $x_1, \ldots, x_N$, and an estimate δy_i of the standard deviation in y_i, we try to recover g from these data by constructing the function $f = f_p$ that, for a given parameter $p \in (0\,..\,1)$ (to be chosen somehow), minimizes

$$(1) \qquad p \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{\delta y_i} \right)^2 + (1-p) \int_{x_1}^{x_N} \left(f^{(m)}(t)\right)^2 \mathrm{d}t$$

over all functions f with m derivatives. Minimization of (1) establishes a sort of compromise between our desire to stay close to the given data and our desire to obtain a smooth function, and the choice of p depends on which of these two conflicting goals we accord the greater importance.

The solution f_p turns out to be a spline of order $k := 2m$, with simple knots at $x_2, \ldots, x_{N-1}$, and satisfying the "natural" end conditions

$$f_p^{(j)}(x_1) = f_p^{(j)}(x_N) = 0 \qquad \text{for } \; j = m, \ldots, k-2,$$

or, for short,

$$f_p \in \$^{\mathrm{nat}}_{k,\mathbf{x}}.$$

Indeed, it is clear that, in minimizing (1), we need only consider $f \in \$^{\mathrm{nat}}_{k,\mathbf{x}}$ since, for any choice of f, substitution of its unique interpolant at x in $\$^{\mathrm{nat}}_{k,\mathbf{x}}$ for f in (1) leaves the sum unchanged while possibly decreasing the integral, by the minimum property of spline interpolation (see Problem 1).

As p approaches 0, f_p converges to the $\mathbf{L}_2$-approximant (with respect to the inner product $\langle g, h\rangle := \sum_i g(x_i)h(x_i)/(\delta y_i)^2$) from $\Pi_{<,n}$ to the data y, hence we should denote the latter by f_{0+} rather than by f_0, since (1) is only semidefinite when $p = 0$, hence has infinitely many minimizers. Nevertheless, we will also use f_0 for it, for simplicity. As p approaches 1, f_p converges to the "natural" spline interpolant to the data y in $\$_{k,\mathbf{x}}^{\mathbf{nat}}$, hence we denote the latter by f_{1-}, since (1) is only semidefinite when $p = 1$, hence has many minimizers, but may also use f_1 for it, for simplicity. Correspondingly, with

$$\mathrm{S}(f) := \sum_{i=1}^{N} \left((y_i - f(x_i))/\delta y_i\right)^2, \tag{2}$$

$\mathrm{S}(f_p)$ strictly decreases as p goes from 0^+ to 1^-, to a value of 0 at $p = 1^-$ (see Problem 2), unless $\mathrm{S}(f_p) = 0$ for all p. It is therefore possible to determine, for given S, the function f_S that minimizes $\int_a^b \left(f^{(m)}(x)\right)^2 \mathrm{d}x$ over all those f for which $\mathrm{S}(f) \le S$ simply by finding the smallest p in $[0^+ \,..\, 1^-]$ for which $\mathrm{S}(f_p) \le S$; then $f_S = f_p$.

Perhaps one should call f_p a **Whittaker spline** since the proposal to smooth data by minimizing (1) goes back to Whittaker [1923]. To be precise, Whittaker only considered uniformly spaced data sites and proposed to use the sum of squares of mth differences rather than $\int (f^{(m)}(x))^2 \mathrm{d}x$ as a measure of smoothness, but the essential idea on how to balance the two conflicting goals of accuracy and smoothness (which make up the problem of smoothing) is there. This idea was picked up again by Schoenberg [1964] who realized that the above simple solution could be given to Whittaker's smoothing problem when $\int (f^{(m)}(x))\,\mathrm{d}x$ was used instead of the mth differences. C. Reinsch [1967], apparently quite independently, proposed the same scheme and worked out the solution for $m = 2$ in explicit detail, and even provided an ALGOL program for the construction of cubic smoothing splines that is, for example, incorporated as the program ICSSCU in the IMSL Library [1977].

In Chapter XVII, we encourage the reader to construct a two-dimensional smoothing routine, and for it we need to redo Reinsch's algorithm somewhat. We present this variant here, as incorporated in the subprogram SMOOTH and its subroutines below.

Briefly, the mathematics entering the construction of the *cubic* smoothing spline is as follows. We already pointed out that f_p would have to be in $\$_{4,\mathbf{x}}^{\mathbf{nat}}$, that is, in $\Pi_{<4,\mathbf{x}} \cap C^{(2)}$ with $f_p''(x_1) = f_p''(x_N) = 0$. We can describe f_p on $[x_i \,..\, x_{i+1}]$ completely once we know f_p and f_p'' at x_i and x_{i+1} (see Problem IV.6). Set

$$a_i := f_p(x_i), \qquad c_i := f_p''(x_i)/2, \qquad i = 1, \ldots, N.$$

Then the "natural" boundary conditions and the requirement that f_p have

a continuous first derivative combine to give the linear system

$$\begin{aligned} c_1 &= 0 \\ c_{i-1}\Delta x_{i-1} + c_i 2(\Delta x_{i-1} + \Delta x_i) + c_{i+1}\Delta x_i &= \\ 3(\Delta a_i/\Delta x_i - \Delta a_{i-1}/\Delta x_{i-1}), &\quad i = 2, \ldots, N-1, \\ c_N &= 0 \end{aligned} \tag{3}$$

for the c_i's in term's of the a_i's. If we take $\mathbf{c} := (c_i)_2^{N-1}$ and $\mathbf{a} := (a_i)_1^N$ – note that $\mathbf{c}$ has length $N-2$ while $\mathbf{a}$ is an N-vector – then we can write (3) in matrix form as

$$R\mathbf{c} = 3Q^T\mathbf{a} \tag{4}$$

with R the symmetric tridiagonal matrix of order $N-2$ having the general row

$$\Delta x_{i-1}, \quad 2(\Delta x_{i-1} + \Delta x_i), \quad \Delta x_i$$

and Q^T the tridiagonal matrix of order $(N-2) \times N$ with general row

$$1/\Delta x_{i-1}, \quad -1/\Delta x_{i-1} - 1/\Delta x_i, \quad 1/\Delta x_i.$$

Next, we express (1) in terms of $\mathbf{a}$ and $\mathbf{c}$. For any straight line ℓ,

$$\int_0^h (\ell(x))^2 \, dx = (h/3)\left[(\ell(0))^2 + \ell(0)\ell(h) + (\ell(h))^2\right].$$

Therefore, at $f = f_p$, (1) has the value

$$p\sum_{i=1}^{N}\left[\frac{y_i - a_i}{\delta y_i}\right]^2 + \frac{4(1-p)}{3}\sum_{i=1}^{N-1}\Delta x_i(c_i^2 + c_i c_{i+1} + c_{i+1}^2).$$

In matrix form, this reads

$$p(\mathbf{y}-\mathbf{a})^T D^{-2}(\mathbf{y}-\mathbf{a}) + \frac{2}{3}(1-p)\mathbf{c}^T R\mathbf{c}$$

with D the diagonal matrix $\lceil \delta y_1, \ldots, \delta y_N \rfloor$. But, by (4), $\mathbf{c} = 3R^{-1}Q^T\mathbf{a}$, so that (1) can be written entirely in terms of $\mathbf{a}$ as

$$p(\mathbf{y}-\mathbf{a})^T D^{-2}(\mathbf{y}-\mathbf{a}) + 6(1-p)(R^{-1}Q^T\mathbf{a})^T R(R^{-1}Q^T\mathbf{a}). \tag{5}$$

Since both D^{-2} and $(R^{-1}Q^T)^T R(R^{-1}Q^T)$ are (symmetric and) positive definite, (5) is minimized when $\mathbf{a}$ satisfies

$$-2pD^{-2}(\mathbf{y}-\mathbf{a}) + 12(1-p)(R^{-1}Q^T)^T R(R^{-1}Q^T)\mathbf{a} = \mathbf{0}$$

or, using the fact that $R^{-1} = (R^{-1})^T$ and simplifying,

$$pD^{-2}(\mathbf{y} - \mathbf{a}) = 2(1-p)Q\mathbf{c}. \tag{6}$$

In particular,

$$\mathrm{S}(f_p) = (\mathbf{y} - \mathbf{a})^T D^{-2}(\mathbf{y} - \mathbf{a}) = \left[\frac{2(1-p)}{p}\right]^2 \|DQ\mathbf{c}\|_2^2,$$

with $\|\mathbf{b}\|_2^2 := \sum_i |b_i|^2$, which allows us to compute $\mathrm{S}(f_p)$ once we know $\mathbf{c}$.

In order to obtain a linear system for $\mathbf{c}$, we multiply both sides of (6) by $3Q^T D^2$ and then use (4) to get

$$p(3Q^T\mathbf{y} - R\mathbf{c}) = 6(1-p)Q^T D^2 Q\mathbf{c}$$

or

$$(6(1-p)Q^T D^2 Q + pR)\mathbf{c} = 3pQ^T\mathbf{y}.$$

Now let $\mathbf{u}$ be such that

$$\mathbf{c} = 3p\mathbf{u}.$$

Then, in summary,

$$(6(1-p)Q^T D^2 Q + pR)\mathbf{u} = Q^T\mathbf{y} \tag{7}$$

and

$$\mathrm{S}(f_p) = (6(1-p))^2 \|DQ\mathbf{u}\|_2^2, \tag{8}$$

while, by (6),

$$\mathbf{a} = \mathbf{y} - 6(1-p)D^2 Q\mathbf{u}. \tag{9}$$

From this, we obtain the information for the ppform of f_p as follows:

$$\begin{aligned} f_p(x_i) &= a_i \\ f_p'(x_i) &= \Delta a_i/\Delta x_i - (f_p''(x_i)/2)\Delta x_i - (f_p'''(x_i^+)/6)(\Delta x_i)^2 \\ f_p''(x_i) &= 6pu_i \\ f_p'''(x_i^+) &= (f_p''(x_{i+1}) - f_p''(x_i))/\Delta x_i \end{aligned} \tag{10}$$

The subroutine SMOOTH and its subroutines Equations (7) – (10) form the basis for the construction of f_S ($= f_p$ for appropriate p in $[0\,..\,1]$) for given S, in SMOOTH and its subroutines.

```
      REAL FUNCTION SMOOTH ( X, Y, DY, NPOINT, S, V, A )
CALLS  SETUPQ, CHOL1D
C
C  CONSTRUCTS THE CUBIC SMOOTHING SPLINE  F  TO GIVEN DATA  (X(I),Y(I)),
C  I=1,...,NPOINT, WHICH HAS AS SMALL A SECOND DERIVATIVE AS POSSIBLE
C  WHILE
C  S(F) = SUM( ((Y(I)-F(X(I)))/DY(I))**2 , I=1,...,NPOINT ) .LE. S .
C
C******  I N P U T  ******
C  X(1),...,X(NPOINT)   DATA ABSCISSAE,  A S S U M E D  TO BE STRICTLY
C        INCREASING .
C  Y(1),...,Y(NPOINT)     CORRESPONDING DATA ORDINATES .
C  DY(1),...,DY(NPOINT)     ESTIMATE OF UNCERTAINTY IN DATA,  A S S U M-
C        E D  TO BE POSITIVE .
C  NPOINT.....NUMBER OF DATA POINTS,  A S S U M E D  .GT. 1
C  S.....UPPER BOUND ON THE DISCRETE WEIGHTED MEAN SQUARE DISTANCE OF
C        THE APPROXIMATION  F  FROM THE DATA .
C
C******  W O R K  A R R A Y S  *****
C  V.....OF SIZE (NPOINT,7)
C  A.....OF SIZE (NPOINT,4)
C
C*****  O U T P U T  *****
C  A(.,1).....CONTAINS THE SEQUENCE OF SMOOTHED ORDINATES .
C  A(I,J) = F^(J-1)(X(I)), J=2,3,4, I=1,...,NPOINT-1 ,  I.E., THE
C        FIRST THREE DERIVATIVES OF THE SMOOTHING SPLINE  F  AT THE
C        LEFT END OF EACH OF THE DATA INTERVALS .
C     W A R N I N G . . .  A  WOULD HAVE TO BE TRANSPOSED BEFORE IT
C        COULD BE USED IN  PPVALU .
C
C******  M E T H O D  ******
C     THE MATRICES  Q-TRANSP*D  AND  Q-TRANSP*D**2*Q  ARE CONSTRUCTED IN
C   S E T U P Q  FROM  X  AND  DY , AS IS THE VECTOR  QTY = Q-TRANSP*Y .
C  THEN, FOR GIVEN  P , THE VECTOR  U  IS DETERMINED IN  C H O L 1 D  AS
C  THE SOLUTION OF THE LINEAR SYSTEM
C               (6(1-P)Q-TRANSP*D**2*Q + P*Q)U  = QTY  .
C  FROM  U , THE SMOOTHING SPLINE  F  (FOR THIS CHOICE OF SMOOTHING PAR-
C  AMETER  P ) IS OBTAINED IN THE SENSE THAT
C                     F(X(.))  =  Y - 6(1-P)D**2*Q*U        AND
C                   F''(X(.))  =  6*P*U                          .
C     THE SMOOTHING PARAMETER  P  IS FOUND (IF POSSIBLE) SO THAT
C                SF(P)  =  S ,
C  WITH  SF(P) = S(F) , WHERE  F  IS THE SMOOTHING SPLINE AS IT DEPENDS
C  ON  P .  IF  S = 0, THEN P = 1 . IF  SF(0) .LE. S , THEN P = 0 .
C  OTHERWISE, THE SECANT METHOD IS USED TO LOCATE AN APPROPRIATE  P  IN
C  THE OPEN INTERVAL  (0 .. 1) . HOWEVER, STRAIGHTFORWARD APPLICATION OF
C  THE SECANT METHOD, AS DONE IN THE ORIGINAL VERSION OF THIS PROGRAM,
C  CAN BE VERY SLOW AND IS INFLUENCED BY THE UNITS IN WHICH  X  AND  Y
C  ARE MEASURED, AS C. REINSCH HAS POINTED OUT. INSTEAD, ON RECOMMEND-
C  ATION FROM C. REINSCH, THE SECANT METHOD IS APPLIED TO THE FUNCTION
C           G:Q -> 1/SQRT{SFQ(Q)} - 1/SQRT{S} ,
C  WITH  SFQ(Q) := SF(Q/(1+Q)), SINCE  1/SQRT{SFQ}  IS MONOTONE INCREASING
C  AND CLOSE TO LINEAR FOR LARGER  Q . ONE STARTS AT  Q = 0  WITH A
C  NEWTON STEP, I.E.,
C                  Q_0 = 0,  Q_1 = -G(0)/G'(0)
C  WITH  G'(0) = -(1/2) SFQ(0)^{-3/2} DSFQ, WHERE DSFQ = -12*U-TRANSP*R*U ,
C  AND  U  AS OBTAINED FOR  P = 0 . ITERATION TERMINATES AS SOON AS
C   ABS(SF - S) .LE. .01*S .
C
```

```
C     LOGICAL TEST
C     PARAMETER (TEST = .TRUE.)
C     INTEGER ITERCNT
      INTEGER NPOINT,   I,NPM1
      REAL A(NPOINT,4),DY(NPOINT),S,V(NPOINT,7),X(NPOINT),Y(NPOINT)
     *     ,CHANGE,OOSS,OOSF,P,PREVSF,PREVQ,Q,SFQ,SIXP,SIX1MP,UTRU
      CALL SETUPQ(X,DY,Y,NPOINT,V,A(1,4))
      IF (S .GT. 0.)                          GO TO 20
   10 P = 1.
      CALL CHOL1D(P,V,A(1,4),NPOINT,1,A(1,3),A(1,1))
      SFQ = 0.
                                              GO TO 60
   20 P = 0.
      CALL CHOL1D(P,V,A(1,4),NPOINT,1,A(1,3),A(1,1))
      SFQ = 0.
      DO 21 I=1,NPOINT
   21    SFQ = SFQ + (A(I,1)*DY(I))**2
      SFQ = SFQ*36.
      IF (SFQ .LE. S)                         GO TO 60
      UTRU = 0.
      DO 25 I=2,NPOINT
   25    UTRU = UTRU + V(I-1,4)*(A(I-1,3)*(A(I-1,3)+A(I,3))+A(I,3)**2)
      OOSS = 1./SQRT(S)
      OOSF = 1./SQRT(SFQ)
      Q = -(OOSF-OOSS)*SFQ/(6.*UTRU*OOSF)
C  SECANT ITERATION FOR THE DETERMINATION OF P STARTS HERE.
C     ITERCNT = 0
      PREVQ = 0.
      PREVSF = OOSF
   30 CALL CHOL1D(Q/(1.+Q),V,A(1,4),NPOINT,1,A(1,3),A(1,1))
      SFQ = 0.
      DO 35 I=1,NPOINT
   35    SFQ = SFQ + (A(I,1)*DY(I))**2
      SFQ = SFQ*36./(1.+Q)**2
      IF (ABS(SFQ-S) .LE. .01*S)              GO TO 59
      OOSF = 1./SQRT(SFQ)
      CHANGE = (Q-PREVQ)/(OOSF-PREVSF)*(OOSF-OOSS)
      PREVQ = Q
      Q = Q - CHANGE
      PREVSF = OOSF
C     ITERCNT = ITERCNT + 1
                                              GO TO 30
   59 P = Q/(1.+Q)
CORRECT VALUE OF P HAS BEEN FOUND.
COMPUTE POL.COEFFICIENTS FROM  Q*U (IN A(.,1)).
   60 SMOOTH = SFQ
C     IF (TEST) THEN
C        PRINT *, 'NUMBER OF ITERATIONS = ', ITERCNT
C     END IF
      SIX1MP = 6./(1.+Q)
      DO 61 I=1,NPOINT
   61    A(I,1) = Y(I) - SIX1MP*DY(I)**2*A(I,1)
      SIXP = 6.*P
      DO 62 I=1,NPOINT
   62    A(I,3) = A(I,3)*SIXP
      NPM1 = NPOINT - 1
      DO 63 I=1,NPM1
         A(I,4) = (A(I+1,3)-A(I,3))/V(I,4)
   63    A(I,2) = (A(I+1,1)-A(I,1))/V(I,4)
     *                        - (A(I,3)+A(I,4)/3.*V(I,4))/2.*V(I,4)
                                              RETURN
      END
```

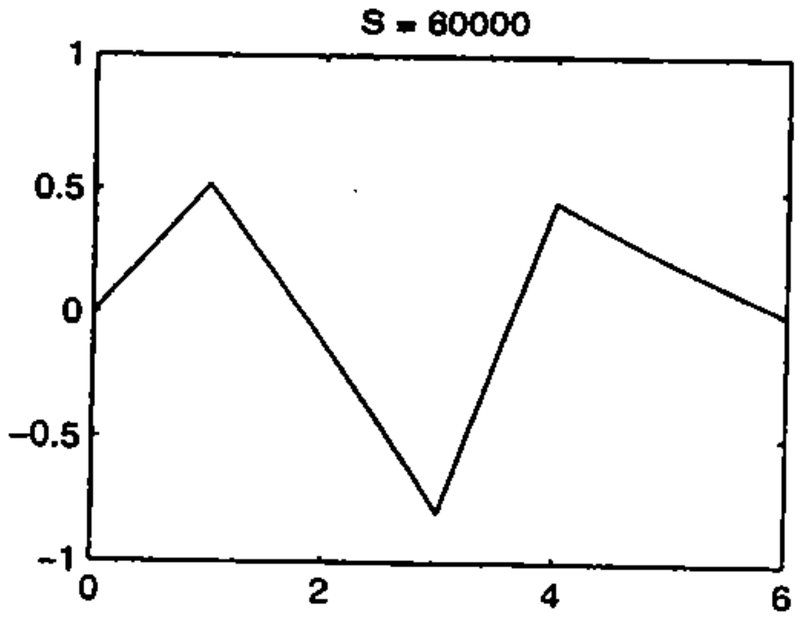
```
      SUBROUTINE SETUPQ ( X, DY, Y, NPOINT, V, QTY )
C  TO BE CALLED IN  S M O O T H
C  PUT  DELX = X(.+1) - X(.)  INTO  V(.,4),
C  PUT  THE THREE BANDS OF  Q-TRANSP*D  INTO  V(.,1-3), AND
C  PUT THE THREE BANDS OF  (D*Q)-TRANSP*(D*Q)  AT AND ABOVE THE DIAGONAL
C     INTO  V(.,5-7) .
C     HERE,  Q IS  THE TRIDIAGONAL MATRIX OF ORDER (NPOINT-2,NPOINT)
C  WITH GENERAL ROW  1/DELX(I) , -1/DELX(I) - 1/DELX(I+1) , 1/DELX(I+1)
C  AND   D  IS THE DIAGONAL MATRIX  WITH GENERAL ROW  DY(I) .
      INTEGER NPOINT,   I,NPM1
      REAL DY(NPOINT),QTY(NPOINT),V(NPOINT,7),X(NPOINT),Y(NPOINT),
     *                                                    DIFF,PREV
      NPM1 = NPOINT - 1
      V(1,4) = X(2) - X(1)
      DO 11 I=2,NPM1
         V(I,4) = X(I+1) - X(I)
         V(I,1) = DY(I-1)/V(I-1,4)
         V(I,2) = - DY(I)/V(I,4) - DY(I)/V(I-1,4)
   11    V(I,3) = DY(I+1)/V(I,4)
       V(NPOINT,1) = 0.
      DO 12 I=2,NPM1
   12    V(I,5) = V(I,1)**2 + V(I,2)**2 + V(I,3)**2
      IF (NPM1 .LT. 3)                    GO TO 14
      DO 13 I=3,NPM1
   13    V(I-1,6) = V(I-1,2)*V(I,1) + V(I-1,3)*V(I,2)
   14 V(NPM1,6) = 0.
      IF (NPM1 .LT. 4)                    GO TO 16
      DO 15 I=4,NPM1
   15    V(I-2,7) = V(I-2,3)*V(I,1)
   16 V(NPM1-1,7) = 0.
      V(NPM1,7) = 0.
CONSTRUCT  Q-TRANSP. * Y  IN  QTY.
      PREV = (Y(2) - Y(1))/V(1,4)
      DO 21 I=2,NPM1
         DIFF = (Y(I+1)-Y(I))/V(I,4)
         QTY(I) = DIFF - PREV
   21    PREV = DIFF

                                          RETURN
      END



      SUBROUTINE CHOL1D ( P, V, QTY, NPOINT, NCOL, U, QU )
C TO BE CALLED IN  S M O O T H
CONSTRUCTS THE UPPER THREE DIAGS. IN V(I,J), I=2,,NPOINT-1, J=1,3, OF
C  THE MATRIX  6*(1-P)*Q-TRANSP.*(D**2)*Q + P*R, THEN COMPUTES ITS
C  L*L-TRANSP. DECOMPOSITION AND STORES IT ALSO IN V, THEN APPLIES
C  FORWARD AND BACKSUBSTITUTION TO THE RIGHT SIDE Q-TRANSP.*Y IN  QTY
C  TO OBTAIN THE SOLUTION IN  U .
      INTEGER NCOL,NPOINT,   I,NPM1,NPM2
      REAL P,QTY(NPOINT),QU(NPOINT),U(NPOINT),V(NPOINT,7),   PREV,RATIO
     *    ,SIX1MP,TWOP
      NPM1 = NPOINT - 1
C     CONSTRUCT 6*(1-P)*Q-TRANSP.*(D**2)*Q  +  P*R
      SIX1MP = 6.*(1.-P)
      TWOP = 2.*P
      DO 2 I=2,NPM1
         V(I,1) = SIX1MP*V(I,5) + TWOP*(V(I-1,4)+V(I,4))
         V(I,2) = SIX1MP*V(I,6) + P*V(I,4)
    2    V(I,3) = SIX1MP*V(I,7)
```

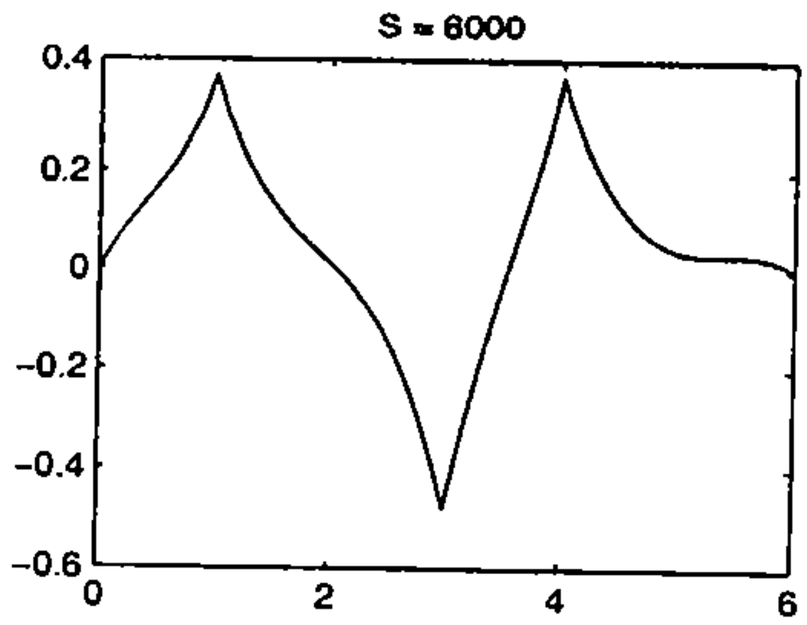
```
      NPM2 = NPOINT - 2
      IF (NPM2 .GE. 2)                    GO TO 10
      U(1) = 0.
      U(2) = QTY(2)/V(2,1)                 .
      U(3) = 0.
                                          GO TO 41
C  FACTORIZATION
   10 DO 20 I=2,NPM2
         RATIO = V(I,2)/V(I,1)
         V(I+1,1) = V(I+1,1) - RATIO*V(I,2)
         V(I+1,2) = V(I+1,2) - RATIO*V(I,3)
         V(I,2) = RATIO
         RATIO = V(I,3)/V(I,1)
         V(I+2,1) = V(I+2,1) - RATIO*V(I,3)
   20    V(I,3) = RATIO
C
C  FORWARD SUBSTITUTION
      U(1) = 0.
      V(1,3) = 0.
      U(2) = QTY(2)
      DO 30 I=2,NPM2
   30    U(I+1) = QTY(I+1) - V(I,2)*U(I) - V(I-1,3)*U(I-1)
C  BACK SUBSTITUTION
      U(NPOINT) = 0.
      U(NPM1) = U(NPM1)/V(NPM1,1)
      I = NPM2
   40    U(I) = U(I)/V(I,1)-U(I+1)*V(I,2)-U(I+2)*V(I,3)
         I = I - 1
         IF (I .GT. 1)                    GO TO 40
C  CONSTRUCT Q*U
   41 PREV = 0.
      DO 50 I=2,NPOINT
         QU(I) = (U(I) - U(I-1))/V(I-1,4)
         QU(I-1) = QU(I) - PREV
   50    PREV = QU(I)
      QU(NPOINT) = -QU(NPOINT)
                                          RETURN
      END
```

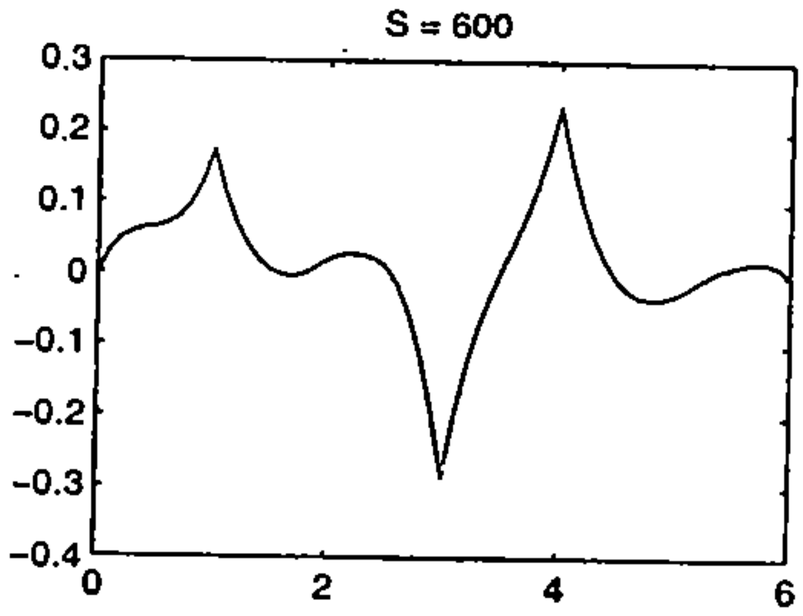
The number S has to be chosen somehow. Reinsch [1967] proposes to choose S somewhere within $\sqrt{2N}$ of N in case δy_i is a good estimate for the standard deviation of the ordinate y_i. More naively, S represents a knob that one may set or turn to achieve a satisfactory approximation to the data. Of course, if one really has no idea how to choose S, then one might as well dispense with it altogether and twiddle the smoothing parameter p directly. Else, if one can make precise what constitutes a "satisfactory" fit to the data in terms other than the function $\mathrm{S}(f)$, then one should change the program so as to find the p appropriate for such a satisfaction.

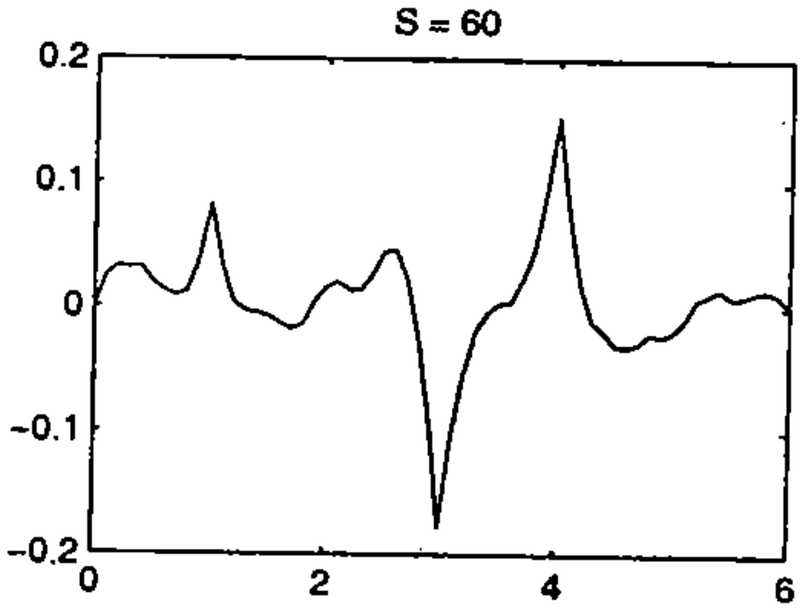
Craven & Wahba [1977] propose a much more sophisticated choice for S based on an estimate of the noise in the data obtained by a process called "cross validation".

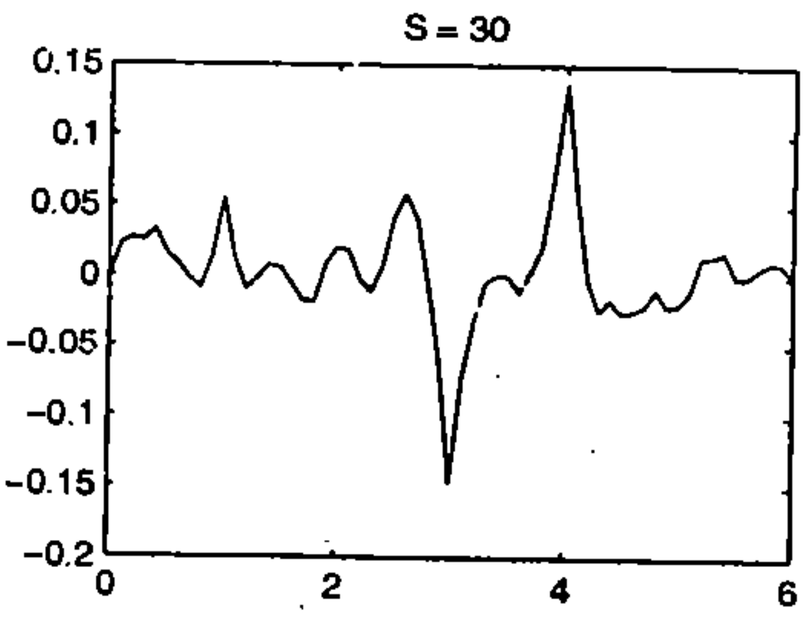
(11) Example: The cubic smoothing spline In order to gain some feel for how this smoothing technique works and just what the effect of choosing particular values for S might be, we consider now the smoothing

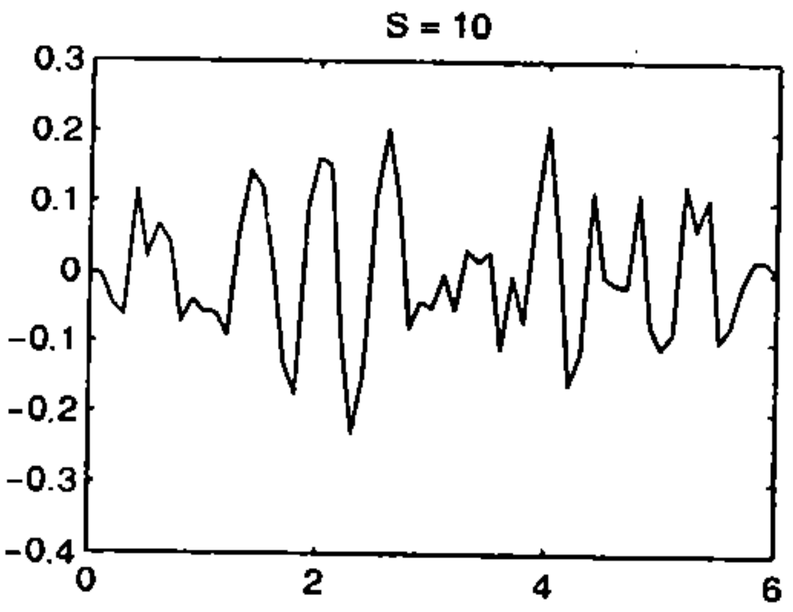


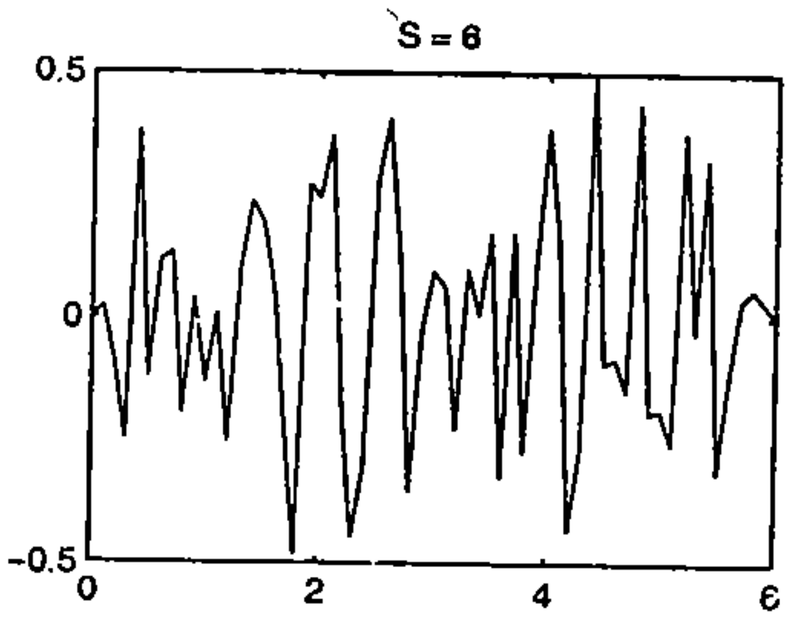


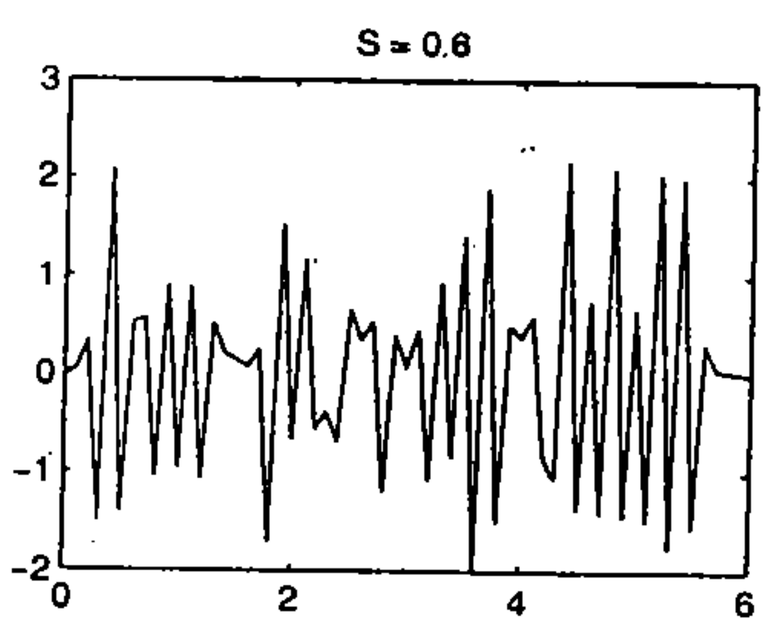












(12) FIGURE. Error in the second derivative of the smoothing spline f_S to noisy data as an approximation to the second derivative of the underlying smooth function g for various values of the smoothing parameter S. An S between 30 and 60 seems optimal.

of certain data, equally spaced on $[0\,..\,6]$ with mesh spacing $1/10$ and obtained from the B-spline of Examples X(7), X(20), and X(30) by rounding its values to a certain number of places. Precisely, with NDIGIT input, we use

$$y_i = \lfloor g(x_i) * 10^{\texttt{NDIGIT}} + .5 \rfloor / 10^{\texttt{NDIGIT}}.$$

An input of NDIGIT $= 2$, for example, would round the function values to two places after the decimal point. Correspondingly, we choose

$$\delta y_i = .5 * 10^{-\texttt{NDIGIT}}, \qquad \text{all } i.$$

In addition, various values of S are input.

Figure (12) gives the error in the second derivative of the smoothing spline f_S (as an approximation to the second derivative of the underlying noise-free function g) as a function of the smoothing parameter S. For large S, f_S is simply a straight line. The error therefore is just the second derivative g'' of the underlying function, clearly demonstrating that g is a cubic spline with simple knots. As S decreases, f_S'' becomes a better approximation to g'' (note that the plotting scale changes). Notice the spikes in the error at $x = 3$ and $x = 4$. These points are knots for g, indicating that the smoothing spline smooths out these "rough spots" in the B-spline g. But, while the error decreases until $S = 30$, there is already some roughness for $S = 60$, and this roughness increases as S decreases further, indicating that we are approximating the noise more and more. For $S = 6$ the second derivative error is already as large as it was for the simple approximation $f_S = 0$, and, for $S < .1$, the error remains pretty much as shown, of size ~ 2.5 and looking like the noise it represents.

The data for Figure (12) were supplied by the following program which also provides the printed output below in response to the input NDIGIT = 2, NS = 7, S = (600000, 60000, 6000, 600, 60, 6, .6).

```
CHAPTER XIV, EXAMPLE 1. CUBIC SMOOTHING SPLINE
CALLS  BSPLPP(BSPLVB), PPVALU(INTERV), SMOOTH(SETUPQ,CHOL1D)
C     VALUES FROM A CUBIC B-SPLINE ARE ROUNDED TO  N D I G I T  PLACES
C  AFTER THE DECIMAL POINT, THEN SMOOTHED VIA  S M O O T H  FOR
C  VARIOUS VALUES OF THE CONTROL PARAMETER  S .
      INTEGER I,IS,J,L,LSM,NDIGIT,NPOINT,NS
      REAL A(61,4),BCOEF(7),BREAK(5),COEF(4,4),COEFSM(4,60),DELY,DY(61)
     *    ,S(20),SCRTCH(427),SFP,T(11),TENTON,X(61),Y(61)
      EQUIVALENCE (SCRTCH,COEFSM)
      DATA T /4*0.,1.,3.,4.,4*6./
      DATA BCOEF /3*0.,1.,3*0./
      CALL BSPLPP(T,BCOEF,7,4,SCRTCH,BREAK,COEF,L)
      NPOINT = 61
      READ 500,NDIGIT,NS,(S(I),I=1,NS)
500   FORMAT(2I3/(E10.4))
      PRINT 600,NDIGIT
600   FORMAT(24H EXACT VALUES ROUNDED TO,I2,21H DIGITS AFTER DECIMAL
     *       ,7H POINT./)
```

```
      TENTON = 10.**NDIGIT
      DELY = .5/TENTON
      DO 10 I=1,NPOINT
         X(I) = .1*FLOAT(I-1)
         Y(I) = PPVALU(BREAK,COEF,L,4,X(I),0)
         Y(I) = FLOAT(IFIX(Y(I)*TENTON + .5))/TENTON
 10      DY(I) = DELY
      DO 15 I=1,NPOINT,5
         DO 15 J=1,4
 15         A(I,J) = PPVALU(BREAK,COEF,L,4,X(I),J-1)
      PRINT 615,(I,(A(I,J),J=1,4),I=1,NPOINT,5)
615   FORMAT(52H VALUE AND DERIVATIVES OF NOISEFREE FUNCTION AT SOME
     *       ,7H POINTS/(I4,4E15.7))
      DO 20 IS=1,NS
         SFP = SMOOTH ( X, Y, DY, NPOINT, S(IS), SCRTCH, A )
         LSM = NPOINT - 1
         DO 16 I=1,LSM
            DO 16 J=1,4
 16            COEFSM(J,I) = A(I,J)
         DO 18 I=1,NPOINT,5
            DO 18 J=1,4
 18            A(I,J) = PPVALU(X,COEFSM,LSM,4,X(I),J-1)
 20      PRINT 620, S(IS), SFP, (I,(A(I,J),J=1,4),I=1,NPOINT,5)
620   FORMAT(15H PRESCRIBED S =,E10.3,23H, S(SMOOTHING SPLINE) =,E10.3/
     *       54H VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP.
     *       ,7H POINTS/(I4,4E15.7))
                                                STOP
      END
```

```
EXACT VALUES ROUNDED TO 2 DIGITS AFTER DECIMAL POINT.
VALUE AND DERIVATIVES OF NOISEFREE FUNCTION AT SOME POINTS
  1  0.0000000+00  0.0000000+00  0.0000000+00  0.5000000+00
  6  0.1041667-01  0.6250000-01  0.2500000+00  0.5000000+00
 11  0.8333334-01  0.2500000+00  0.5000000+00 -0.6999999+00
 16  0.2562500+00  0.4125000+00  0.1500000+00 -0.6999999+00
 21  0.4666667+00  0.4000000+00 -0.1999999+00 -0.6999999+00
 26  0.6270834+00  0.2125000+00 -0.5500000+00 -0.6999999+00
 31  0.6500000+00 -0.1500000+00 -0.9000000+00  0.1300000+01
 36  0.4895834+00 -0.4375000+00 -0.2500000+00  0.1300000+01
 41  0.2666667+00 -0.4000000+00  0.4000000+00 -0.2000000+00
 46  0.1125000+00 -0.2250000+00  0.3000000+00 -0.2000000+00
 51  0.3333333-01 -0.1000000+00  0.2000000+00 -0.2000000+00
 56  0.4166633-02 -0.2499998-01  0.1500000+00 -0.2000000+00
 61  0.2980232-07  0.0000000+00  0.0000000+00 -0.2000000+00
PRESCRIBED S = 0.600+06, S(SMOOTHING SPLINE) = 0.137+06
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1  0.2949416+00 -0.1681417-01  0.0000000+00  0.0000000+00
  6  0.2865339+00 -0.1685768-01  0.0000000+00  0.0000000+00
 11  0.2781245+00 -0.1674324-01  0.0000000+00  0.0000000+00
 16  0.2697705+00 -0.1671850-01  0.0000000+00  0.0000000+00
 21  0.2614206+00 -0.1718161-01  0.0000000+00  0.0000000+00
 26  0.2530047+00 -0.1644847-01  0.0000000+00  0.0000000+00
 31  0.2450134+00 -0.1582173-01  0.0000000+00  0.0000000+00
 36  0.2371033+00 -0.1604510-01  0.0000000+00  0.0000000+00
 41  0.2291455+00 -0.1577170-01  0.0000000+00  0.0000000+00
 46  0.2212827+00 -0.1576947-01  0.0000000+00  0.0000000+00
 51  0.2134389+00 -0.1572163-01  0.0000000+00  0.0000000+00
 56  0.2056005+00 -0.1565324-01  0.0000000+00  0.0000000+00
 61  0.1977749+00 -0.1565041-01  0.0000000+00  0.0000000+00
```

```
PRESCRIBED S = 0.600+05, S(SMOOTHING SPLINE) = 0.563+05
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1  0.1010348+00  0.1354746+00  0.0000000+00 -0.3162780-02
  6  0.1686429+00  0.1344858+00 -0.5560694-02 -0.2470536-01
 11  0.2345982+00  0.1278783+00 -0.2305775-01 -0.5013588-01
 16  0.2945381+00  0.1095352+00 -0.5174045-01 -0.6411268-01
 21  0.3415664+00  0.7594109-01 -0.8298301-01 -0.5473658-01
 26  0.3681809+00  0.2872458-01 -0.1045465+00 -0.2177846-01
 31  0.3691794+00 -0.2517581-01 -0.1065227+00  0.2283935-01
 36  0.3438948+00 -0.7462048-01 -0.8767027-01  0.5539618-01
 41  0.2967495+00 -0.1114196+00 -0.5777792-01  0.6117421-01
 46  0.2350807+00 -0.1330217+00 -0.2963798-01  0.4636624-01
 51  0.1657661+00 -0.1427211+00 -0.1067606-01  0.2509925-01
 56  0.9353174-01 -0.1455003+00 -0.1883308-02  0.7798635-02
 61  0.2067093-01 -0.1457925+00  0.0000000+00  0.6470801-03
PRESCRIBED S = 0.600+04, S(SMOOTHING SPLINE) = 0.605+04
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1 -0.7716598-01  0.2021496+00  0.0000000+00  0.8301824-01
  6  0.2649403-01  0.2195854+00  0.8183185-01  0.1886732+00
 11  0.1498199+00  0.2780035+00  0.1311077+00 -0.9162558-01
 16  0.3020968+00  0.3214137+00  0.1600463-01 -0.4003466+00
 21  0.4559187+00  0.2747874+00 -0.2108194+00 -0.4661664+00
 26  0.5577974+00  0.1163412+00 -0.4057107+00 -0.2153067+00
 31  0.5622612+00 -0.9982967-01 -0.4215109+00  0.2512339+00
 36  0.4662189+00 -0.2677240+00 -0.2242713+00  0.5341890+00
 41  0.3153833+00 -0.3141730+00  0.3059394-01  0.4074625+00
 46  0.1695815+00 -0.2573432+00  0.1705836+00  0.8701154-01
 51  0.6311121-01 -0.1694264+00  0.1607977+00 -0.1504202+00
 56 -0.5010876-02 -0.1108845+00  0.6852908-01 -0.1928409+00
 61 -0.5553601-01 -0.9707810-01  0.0000000+00 -0.5974786-01
PRESCRIBED S = 0.600+03, S(SMOOTHING SPLINE) = 0.600+03
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1 -0.2963135-01  0.7783630-01  0.0000000+00  0.1881339+00
  6  0.1512813-01  0.1172208+00  0.1860341+00  0.4513534+00
 11  0.1052970+00  0.2556820+00  0.3274755+00 -0.1074531+00
 16  0.2695077+00  0.3856237+00  0.1453501+00 -0.6319802+00
 21  0.4664122+00  0.3713391+00 -0.2193756+00 -0.7647599+00
 26  0.6094426+00  0.1723370+00 -0.5597129+00 -0.4544454+00
 31  0.6190019+00 -0.1388132+00 -0.6153280+00  0.4416012+00
 36  0.4845202+00 -0.3686439+00 -0.2554220+00  0.9618705+00
 41  0.2875099+00 -0.3840763+00  0.1638901+00  0.5534083+00
 46  0.1252274+00 -0.2533469+00  0.3071828+00 -0.4325990-01
 51  0.3494206-01 -0.1145922+00  0.2279970+00 -0.2802214+00
 56  0.1165281-03 -0.3661828-01  0.8773698-01 -0.2464721+00
 61 -0.1190371-01 -0.1897625-01  0.0000000+00 -0.7557844-01
PRESCRIBED S = 0.600+02, S(SMOOTHING SPLINE) = 0.603+02
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1 -0.8416147-02  0.2586523-01  0.0000000+00  0.2517082+00
  6  0.1198798-01  0.7529248-01  0.2300497+00  0.5643039+00
 11  0.8925421-01  0.2504269+00  0.4183564+00 -0.2171975+00
 16  0.2591429+00  0.4062449+00  0.1576270+00 -0.6430817+00
 21  0.4678741+00  0.3973049+00 -0.2159979+00 -0.7444684+00
 26  0.6247162+00  0.1993473+00 -0.5940326+00 -0.7218559+00
 31  0.6391436+00 -0.1522276+00 -0.7221369+00  0.5304460+00
 36  0.4878075+00 -0.4132321+00 -0.2530465+00  0.1288554+01
 41  0.2740852+00 -0.3996694+00  0.2466919+00  0.5556253+00
 46  0.1133140+00 -0.2373865+00  0.3303083+00 -0.1859542+00
 51  0.3150384-01 -0.9931158-01  0.2192190+00 -0.2914580+00
 56  0.2944320-02 -0.2602729-01  0.9219799-01 -0.2136636+00
 61 -0.3065382-02 -0.6137666-02  0.0000000+00 -0.9167875-01
```

```
PRESCRIBED S = 0.600+01, S(SMOOTHING SPLINE) = 0.605+01
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1 -0.7999242-04 -0.3257304-02  0.0000000+00  0.2954947+00
  6  0.1196487-01  0.5915504-01  0.3677506+00 -0.1823948+01
 11  0.8113957-01  0.2499736+00  0.6289370+00 -0.2046229+01
 16  0.2597238+00  0.4002945+00 -0.4620701-01  0.7237461+00
 21  0.4704015+00  0.4034437+00 -0.4443774+00 -0.1941344+01
 26  0.6287857+00  0.2456323+00 -0.8231080+00 -0.2005794+01
 31  0.6490487+00 -0.1505089+00 -0.9929882+00  0.1608802+01
 36  0.4882288+00 -0.4450468+00 -0.4194268+00  0.6207655+01
 41  0.2690169+00 -0.4038542+00  0.1528473-01  0.2161185+01
 46  0.1116075+00 -0.2372213+00  0.3930041+00 -0.2912464+00
 51  0.2980389-01 -0.9930097-01  0.3888030+00  0.4782237+00
 56  0.2238924-02 -0.4050584-01  0.4133657+00 -0.2117302+01
 61  0.2720560-04  0.1763557-03  0.0000000+00  0.1004984+00
PRESCRIBED S = 0.600+00, S(SMOOTHING SPLINE) = 0.603+00
VALUE AND DERIVATIVES OF SMOOTHING SPLINE AT CORRESP. POINTS
  1 -0.1190162-05  0.1608201-03  0.0000000+00  0.9704943-01
  6  0.1066370-01  0.3943658-01  0.1659509+01 -0.1882163+02
 11  0.8046399-01  0.2499976+00  0.1445260+01 -0.1891479+02
 16  0.2599986+00  0.4000182+00  0.4411477-02  0.4853381-01
 21  0.4704916+00  0.3992628+00  0.4598409+00 -0.1890898+02
 26  0.6297976+00  0.2591448+00 -0.1208435+01  0.2333732+01
 31  0.6500599+00 -0.1500207+00 -0.9769868+00 -0.2431476+01
 36  0.4893048+00 -0.4598495+00 -0.1666894+01  0.3546511+02
 41  0.2700603+00 -0.3998543+00  0.2349750-01 -0.2357399+01
 46  0.1106943+00 -0.2600034+00  0.1668781+01 -0.2137029+02
 51  0.2947696-01 -0.9997959-01 -0.4671930+00  0.2131620+02
 56  0.6650722-03 -0.6054529-01  0.1655107+01 -0.1888451+02
 61  0.6715251-07 -0.4070584-05  0.0000000+00  0.5475827-02
```

This output shows the increasing roughness (see the third derivative values, in the last column) of the cubic smoothing spline as the value of S is decreased toward 0. Reinsch [1967] would pick S between 50 and 72, and the approximation for $S = 60$ seems indeed reasonably close to the original noise-free data, in value and derivatives.

There is a sign of slight trouble, though. The first derivative of the smoothing spline f_S is not as "continuous" as it could be. This is quite evident for $S = 600000$ in which case f_S should be a straight line, that is, f'_S should be constant, yet f'_S fails to be constant by an amount of $2 * 10^{-4}$. Since the calculations were carried out in better than 7 decimal digit arithmetic and the function values are of order 1, this means a loss of over three decimal digits of accuracy in these slopes. For smaller values of S, one can make this discontinuity evident by rerunning the calculations but using the routine PPVLLC instead of PPVALU, thus making the pp function f_S left-continuous (see Problem VII.2), and comparing with the output listed above. This magnification of rounding errors becomes less prominent as S is chosen smaller and could, of course, be combated easily by going to higher precision arithmetic. □

This slight trouble, though, is to me a symptom of a conceptual difficulty with the smoothing spline. The smoothing spline is chosen from an approximation family with slightly more degrees of freedom than there are data points; its (admittedly redundant) ppform consists of about $4N$ items

compared to the $2N$ original data items. These many degrees of freedom are held in check by the presence of the $\int (f(x))^2\,dx$ term in the expression to be minimized, but do appear, for instance, to cause the rounding error trouble mentioned earlier.

It strikes me as more straightforward to approximate noisy data from an approximation family with many fewer degrees of freedom than there are data points. There should be enough degrees of freedom to approximate the underlying function g well, but not enough to approximate also the faster varying noise. In this, splines are very effective as an approximating family and least-squares approximation should be considered first since it leads to a *banded and linear* system of equations to be solved.

Least-squares approximation is best approximation with respect to a norm that derives from an inner product. While much of the development below is valid for any inner product, we will discuss here only *discrete inner products*, that is, inner products of the form

$$\langle g, h\rangle := \sum_{i=1}^{N} g(\tau_i)h(\tau_i)w_i. \tag{13}$$

Here, τ is a sequence of data sites in some interval $[a\,..\,b]$, and we will assume it to be nondecreasing, for convenience. The sequence $\mathbf{w} = (w_i)_1^N$ of weights is assumed to be nonnegative. A typical choice is

$$w_i := \begin{cases} \Delta\tau_1/2, & i = 1 \\ (\Delta\tau_{i-1} + \Delta\tau_i)/2, & i = 2, \ldots, N-1 \\ \Delta\tau_{N-1}/2, & i = N \end{cases} \tag{14}$$

in which case $\langle g, h\rangle$ is a reasonable approximation to the continuous inner product

$$\int_{\tau_1}^{\tau_N} g(x)h(x)\,dx.$$

Other choices for $\mathbf{w}$ are used to include trends in the data and/or to reflect the varying accuracy of, or confidence in, the given data $g_i \sim g(\tau_i)$.

We use the customary symbol

$$\|h\|_2 := \langle h, h\rangle^{1/2}$$

for the 'norm' induced by the 'inner product' $\langle\ ,\ \rangle$. The quotes here constitute an assenting nod to the alert reader who is about to protest that $\|\cdot\|_2$ is only a *semi*norm since there are many functions h other than the 0 function for which $\|h\|_2 = 0$.

We will use the Cauchy-Schwarz-inequality

$$|\langle g, h\rangle| \le \|g\|_2 \, \|h\|_2. \tag{15}$$

Let now \$ be a finite-dimensional linear space of functions, all on $[a\,..\,b]$. We seek a *best approximation from* \$ *to* 'g' *with respect to the 'norm'* $\|\cdot\|_2$, that is, we seek an $f^* \in \$$ so that

$$\|\text{`}g\text{'} - f^*\|_2 = \min_{f \in \$} \|\text{`}g\text{'} - f\|_2.$$

The finite dimensionality of \$ guarantees that such an f^* actually exists. We have put g in quotes here because we do not know g. Instead, we intend to use the approximate value g_i whenever the value $g(\tau_i)$ is called for. In effect, 'g' is an equivalence class of functions.

(16) Lemma. *The function f^* is a best approximation from \$ to 'g' with respect to $\|\cdot\|_2$ if and only if the function f^* is in \$ and the function 'g' $- f^*$, that is, the error, is* **orthogonal** *to \$, that is,*

$$\text{for all } f \in \$, \qquad \langle f, \text{`}g\text{'} - f^*\rangle = 0. \tag{17}$$

PROOF. For any function $f \in \$$, we compute

$$\begin{aligned} \|\text{`}g\text{'} - f\|_2^2 &= \|\text{`}g\text{'} - f^* \;+\; f^* - f\|_2^2 \\ &= \|\text{`}g\text{'} - f^*\|_2^2 + 2\langle \underbrace{f^* - f}_{\in \$}, \text{`}g\text{'} - f^*\rangle + \|f^* - f\|_2^2. \end{aligned} \tag{18}$$

Therefore,

$$\text{for all } f \in \$, \quad \|\text{`}g\text{'} - f\|_2^2 = \|\text{`}g\text{'} - f^*\|_2^2 + \|f^* - f\|_2^2$$

in case (17) holds, that is, in case the error 'g' $- f^*$ is orthogonal to \$. But then, indeed,

$$\|\text{`}g\text{'} - f\|_2 \ge \|\text{`}g\text{'} - f^*\|_2, \text{ all } f \in \$, \text{ with equality iff } \|f^* - f\|_2 = 0.$$

Conversely, assume that, for some $f \in \$$, $\langle f, \text{`}g\text{'} - f^*\rangle \neq 0$. Then

$$2\langle tf, \text{`}g\text{'} - f^*\rangle = t\Big(2\langle f, \text{`}g\text{'} - f^*\rangle\Big) > t^2\|f\|_2^2 = \|tf\|_2^2$$

for all nonzero t of the same sign as $\langle f, \text{`}g\text{'} - f^*\rangle$ and sufficiently close to 0 (since t^2 goes to zero faster than t). Hence, for all such t, we get from (18) (with $-tf$ substituted for $f^* - f$, hence with f replaced by $f^* + tf \in \$$) that

$$\|\text{`}g\text{'} - (f^* + tf)\|_2^2 < \|\text{`}g\text{'} - f^*\|_2^2,$$

showing that then $f^* + tf \in \$$ is a better approximation. □

This proves, incidentally, that the difference between best approximations has 'norm' 0, therefore shows that 'g' has a *unique* best approximation in \$ if and only if $\|\cdot\|_2$ is a norm on \$, that is, if and only if the only function f in \$ for which $\|f\|_2 = 0$ is $f = 0$.

Assume now that $\|\cdot\|_2$ is indeed a norm on our linear space \$ and that $(\varphi_i)_1^n$ is a basis for \$. Then we know that (17) has a unique solution f^* in \$. Further, one easily verifies that (17) is equivalent to

$$\text{for} \quad i = 1, \ldots, n, \qquad \langle \varphi_i, \text{`}g\text{'} - f^* \rangle = 0, \tag{19}$$

hence this latter system of equations has a unique solution $f^* \in \$$. Also, f^* has a unique representation $\sum_1^n \alpha_i \varphi_i$ in terms of the basis $(\varphi_i)_1^n$, hence, the linear system

$$\text{for} \quad i = 1, \ldots, n, \qquad \langle \varphi_i, \text{`}g\text{'} - \sum_{j=1}^{n} \alpha_j \varphi_j \rangle = 0, \tag{20}$$

has a unique solution (α_i). These latter equations constitute the system of *normal equations*, usually written

$$\text{for} \quad i = 1, \ldots, n, \qquad \sum_{j=1}^{n} \langle \varphi_i, \varphi_j \rangle \alpha_j = \langle \varphi_i, \text{`}g\text{'} \rangle \tag{21}$$

using the linearity of the inner product $\langle\ ,\ \rangle$ in its second argument.

The determination of f^* via the normal system may or may not be a good thing. It depends on what particular basis (φ_i) for \$ we happen to have chosen. If we have somehow succeeded in choosing (φ_i) to be *orthogonal*, that is, so that

$$\langle \varphi_i, \varphi_j \rangle = 0 \qquad \text{for } i \neq j,$$

then it is quite easy to solve the normal system; the solution is then simply

$$\alpha_j = \langle \varphi_j, \text{`}g\text{'} \rangle / \langle \varphi_j, \varphi_j \rangle, \qquad j = 1, \ldots, n.$$

On the other hand, we might be unlucky and end up with a basis (φ_i) that is *badly conditioned*, that is, for which relative changes of a certain size in the coefficient vector (α_i) may result in relative changes in the function $\sum_i \alpha_i \varphi_i$ represented of widely varying sizes. A classical example is the basis $\varphi_i(x) = x^{k-i}$, $i = 1, \ldots, k$, for the linear space $\Pi_{<k}$ on the interval $[0\,..\,1]$. If the inner product

$$\langle g, h \rangle = \int_0^1 g(x)h(x)\,\mathrm{d}x$$

is used, then the coefficient matrix for the normal system (21) is the Hilbert matrix, the textbook example of an illconditioned matrix. The reader to whom these considerations are unfamiliar is urged to consult a text on the numerical solution of linear systems, for example, Forsythe & Moler [1967], or the book Lawson & Hanson [1974] on "Solving least-squares problems".

Least-squares approximation from $\$_{k,\mathrm{t}}$ For the specific linear space $\$ = \$_{k,\mathbf{t}}$ of splines, with $\mathbf{t} = (t_i)_1^{n+k}$, the B-spline basis $(\mathrm{B}_i)_1^n$ seems convenient since it is relatively well conditioned (see Cor. XI(8)), at least for moderate k. Further, the coefficient matrix for the normal system

$$\sum_{j=1}^{n} \langle \mathrm{B}_i, \mathrm{B}_j \rangle \alpha_j = \langle \mathrm{B}_i, \text{`}g\text{'} \rangle, \qquad i = 1, \ldots, n, \tag{22}$$

has bandwidth less than k in the sense that $\langle \mathrm{B}_i, \mathrm{B}_j \rangle = 0$ for $|i - j| \geq k$. This follows readily from the fact that $\mathrm{B}_i(x) = 0$ for $x \notin [t_i \,..\, t_{i+k}]$, hence $|i - j| \geq k$ implies that $\mathrm{B}_i(x)\mathrm{B}_j(x) = 0$ for all x. Since the matrix $(\langle \mathrm{B}_i, \mathrm{B}_j \rangle)$ is also symmetric, this allows one to store it by storing just its k lower diagonal bands. Further, the matrix is positive (semi)definite. We can therefore solve (22) by Gauss elimination *without pivoting*, and, since the matrix is symmetric, we can carry this out within its lower diagonal bands. This version of Gauss elimination, without pivoting and for a symmetric matrix, is called Cholesky factorization: In its squareroot-free form, the symmetric matrix A is factored into LDL^T, with L a unit lower triangular matrix and D a diagonal matrix. The triangular system $L\mathbf{y} = \mathbf{b}$ is then solved for $\mathbf{y}$, and, finally, the triangular system $DL^T\mathbf{x} = \mathbf{y}$ is solved for the solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{y}$. Further details can be found, for example, in Forsythe & Moler [1967]. These calculations are carried out in the subprograms BCHFAC/BCHSLV given below.

We must verify that our discrete 'norm' is, indeed, a norm on $\$_{k,\mathrm{t}}$ if we want to make certain that the normal system (22) has exactly one solution. Since $\|f\|_2 = 0$ implies that, for all i (with $w_i \neq 0$), $f(\tau_i) = 0$, we are asking for conditions on τ that ensure that $(f(\tau_i)) = 0$ implies that $f = \mathbf{0}$. From the Schoenberg-Whitney Theorem (Theorem XIII(2)), we have at once the following lemma.

(23) Lemma. *The 'norm'* $\|f\|_2 := \left(\sum_i w_i (f(\tau_i))^2\right)^{1/2}$ *with* $w_i > 0$, *all* i, *and* (τ_i) *nondecreasing, is a norm on* $\$_{k,\mathrm{t}}$ *if and only if, for some* $1 \leq j_1 < \cdots < j_n \leq N$,

$$t_i < \tau_{j_i} < t_{i+k}, \qquad i = 1, \ldots, n.$$

If the condition of this lemma is violated, then $f \mapsto \left(\sum_i w_i (f(\tau_i))^2\right)^{1/2}$ fails to be a norm on $\$_{k,\mathbf{t}}$ and the normal system (22) is singular. Since the normal system *always* has solutions, by Lemma (16), this implies that it has many solutions in this case, with any two solutions agreeing at all the τ_i's.

In such a situation, it has been proposed to select a "best" least-squares approximation by the requirement that it be smallest in some sense among all least-squares approximations. In effect, the singular problem is made nonsingular by the imposition of additional conditions on the solution. This process is also called "*regularization*" and is also used to rescue ill-

conditioned, or nearly singular, problems.

The approach taken in the subprogram below is somewhat different. If $\|\cdot\|_2$ fails to be a norm on $\$_{k,\mathbf{t}}$, then, the B-spline basis $(B_i)_1^n$ for $\$_{k,\mathbf{t}}$ fails to be linearly independent over τ, that is, one or more of the B_j's depends, over τ, linearly on the B_i's preceding it in the sequence. We choose a "best" least-squares approximation by forcing the coefficients corresponding to these dependent B_j's to equal zero. (See Example (26) below.)

The subroutine L2APPR (with BCHFAC/BCHSLV)

```
      SUBROUTINE L2APPR ( T, N, K, Q, DIAG, BCOEF )
C  TO BE CALLED IN MAIN PROGRAM  L 2 M A I N .
CALLS SUBPROGRAMS  BSPLVB, BCHFAC/SLV
C
CONSTRUCTS THE (WEIGHTED DISCRETE) L2-APPROXIMATION BY SPLINES OF ORDER
C  K  WITH KNOT SEQUENCE  T(1), ..., T(N+K)  TO GIVEN DATA POINTS
C  ( TAU(I), GTAU(I) ), I=1,...,NTAU. THE B-SPLINE COEFFICIENTS
C  B C O E F   OF THE APPROXIMATING SPLINE ARE DETERMINED FROM THE
C  NORMAL EQUATIONS USING CHOLESKY'S METHOD.
C
C******  I N P U T  ******
C  T(1), ..., T(N+K)  THE KNOT SEQUENCE
C  N.....THE DIMENSION OF THE SPACE OF SPLINES OF ORDER K WITH KNOTS T.
C  K.....THE ORDER
C
C  W A R N I N G  . . .  THE RESTRICTION   K .LE. KMAX (= 20)   IS IMPO-
C       SED BY THE ARBITRARY DIMENSION STATEMENT FOR  BIATX  BELOW, BUT
C       IS  N O W H E R E  C H E C K E D  FOR.
C
C******  W O R K  A R R A Y S  ******
C  Q....A WORK ARRAY OF SIZE (AT LEAST) K*N. ITS FIRST  K  ROWS ARE USED
C      FOR THE  K  LOWER DIAGONALS OF THE GRAMIAN MATRIX  C .
C  DIAG.....A WORK ARRAY OF LENGTH  N  USED IN BCHFAC .
C
C******  I N P U T  VIA  C O M M O N  /DATA/  ******
C  NTAU.....NUMBER OF DATA POINTS
C  (TAU(I),GTAU(I)), I=1,...,NTAU     ARE THE  NTAU  DATA POINTS TO BE
C       FITTED .
C  WEIGHT(I), I=1,...,NTAU    ARE THE CORRESPONDING WEIGHTS .
C
C******  O U T P U T  ******
C  BCOEF(1), ..., BCOEF(N)  THE B-SPLINE COEFFS. OF THE L2-APPR.
C
C******  M E T H O D  ******
C  THE B-SPLINE COEFFICIENTS OF THE L2-APPR. ARE DETERMINED AS THE SOL-
C  UTION OF THE NORMAL EQUATIONS
C     SUM ( (B(I),B(J))*BCOEF(J) : J=1,...,N)  = (B(I),G),
C                                                 I = 1, ..., N .
C  HERE,  B(I)  DENOTES THE I-TH B-SPLINE,  G  DENOTES THE FUNCTION TO
C  BE APPROXIMATED, AND THE  I N N E R  P R O D U C T  OF TWO FUNCT-
C  IONS  F  AND  G  IS GIVEN BY
C      (F,G)  :=  SUM ( F(TAU(I))*G(TAU(I))*WEIGHT(I) : I=1,...,NTAU) .
C  THE ARRAYS  T A U  AND  W E I G H T  ARE GIVEN IN COMMON BLOCK
C   D A T A , AS IS THE ARRAY  G T A U  CONTAINING THE SEQUENCE
C  G(TAU(I)), I=1,...,NTAU.
C  THE RELEVANT FUNCTION VALUES OF THE B-SPLINES  B(I), I=1,...,N, ARE
C  SUPPLIED BY THE SUBPROGRAM  B S P L V B .
C     THE COEFF.MATRIX  C , WITH
C           C(I,J)  :=  (B(I), B(J)), I,J=1,...,N,
```

```
C  OF THE NORMAL EQUATIONS IS SYMMETRIC AND (2*K-1)-BANDED, THEREFORE
C  CAN BE SPECIFIED BY GIVING ITS K BANDS AT OR BELOW THE DIAGONAL. FOR
C  I=1,...,N,  WE STORE
C   (B(I),B(J))  =  C(I,J)  IN  Q(I-J+1,J), J=I,...,MINO(I+K-1,N)
C  AND THE RIGHT SIDE
C   (B(I), G )  IN  BCOEF(I) .
C  SINCE B-SPLINE VALUES ARE MOST EFFICIENTLY GENERATED BY FINDING SIM-
C  ULTANEOUSLY THE VALUE OF  E V E R Y  NONZERO B-SPLINE AT ONE POINT,
C  THE ENTRIES OF  C  (I.E., OF  Q ), ARE GENERATED BY COMPUTING, FOR
C  EACH LL, ALL THE TERMS INVOLVING  TAU(LL)  SIMULTANEOUSLY AND ADDING
C  THEM TO ALL RELEVANT ENTRIES.
      INTEGER K,N,   I,J,JJ,KMAX,LEFT,LEFTMK,LL,MM,NTAU,NTMAX
      PARAMETER (KMAX=20,NTMAX=200)
      REAL BCOEF(N),DIAG(N),Q(K,N),T(1),  BIATX(KMAX),DW,GTAU,TAU,WEIGHT
      DIMENSION T(N+K)
      COMMON / DATA / NTAU, TAU(NTMAX),GTAU(NTMAX),WEIGHT(NTMAX)
      DO 7 J=1,N
         BCOEF(J) = 0.
         DO 7 I=1,K
    7       Q(I,J) = 0.
      LEFT = K
      LEFTMK = 0
      DO 20 LL=1,NTAU
C                    LOCATE  L E F T  S.T. TAU(LL) IN (T(LEFT),T(LEFT+1))
   10       IF (LEFT .EQ. N)              GO TO 15
            IF (TAU(LL) .LT. T(LEFT+1)) GO TO 15
            LEFT = LEFT+1
            LEFTMK = LEFTMK + 1
                                          GO TO 10
   15    CALL BSPLVB ( T, K, 1, TAU(LL), LEFT, BIATX )
C        BIATX(MM) CONTAINS THE VALUE OF B(LEFT-K+MM) AT TAU(LL).
C        HENCE, WITH  DW := BIATX(MM)*WEIGHT(LL), THE NUMBER DW*GTAU(LL)
C        IS A SUMMAND IN THE INNER PRODUCT
C           (B(LEFT-K+MM), G)  WHICH GOES INTO  BCOEF(LEFT-K+MM)
C        AND THE NUMBER BIATX(JJ)*DW IS A SUMMAND IN THE INNER PRODUCT
C           (B(LEFT-K+JJ), B(LEFT-K+MM)), INTO  Q(JJ-MM+1,LEFT-K+MM)
C        SINCE  (LEFT-K+JJ) - (LEFT-K+MM) + 1  =  JJ - MM + 1 .
         DO 20 MM=1,K
            DW = BIATX(MM)*WEIGHT(LL)
            J = LEFTMK + MM
            BCOEF(J) = DW*GTAU(LL) + BCOEF(J)
            I = 1
            DO 20 JJ=MM,K
               Q(I,J) = BIATX(JJ)*DW + Q(I,J)
   20          I = I + 1
C
C           CONSTRUCT CHOLESKY FACTORIZATION FOR  C  IN  Q , THEN USE
C           IT TO SOLVE THE NORMAL EQUATIONS
C                  C*X  =  BCOEF
C           FOR  X , AND STORE  X  IN  BCOEF .
      CALL BCHFAC ( Q, K, N, DIAG )
      CALL BCHSLV ( Q, K, N, BCOEF )
                                       RETURN
      END
```

```
      SUBROUTINE BCHFAC ( W, NBANDS, NROW, DIAG )
CONSTRUCTS CHOLESKY FACTORIZATION
C                        C  =  L * D * L-TRANSPOSE
C  WITH L UNIT LOWER TRIANGULAR AND D DIAGONAL, FOR GIVEN MATRIX C OF
C  ORDER  N R O W , IN CASE  C  IS (SYMMETRIC) POSITIVE SEMIDEFINITE
C  AND  B A N D E D , HAVING  N B A N D S  DIAGONALS AT AND BELOW THE
C  MAIN DIAGONAL.
C
C******  I N P U T  ******
C  NROW.....IS THE ORDER OF THE MATRIX  C .
C  NBANDS.....INDICATES ITS BANDWIDTH, I.E.,
C           C(I,J) = 0 FOR I-J .GE. NBANDS .
C  W.....WORKARRAY OF SIZE (NBANDS,NROW)  CONTAINING THE  NBANDS  DIAGO-
C        NALS IN ITS ROWS, WITH THE MAIN DIAGONAL IN ROW  1 . PRECISELY,
C        W(I,J)  CONTAINS  C(I+J-1,J), I=1,...,NBANDS, J=1,...,NROW.
C          FOR EXAMPLE, THE INTERESTING ENTRIES OF A SEVEN DIAGONAL SYM-
C        METRIC MATRIX  C  OF ORDER  9  WOULD BE STORED IN  W  AS
C
C                        11 22 33 44 55 66 77 88 99
C                        21 32 43 54 65 76 87 98
C                        31 42 53 64 75 86 97
C                        41 52 63 74 85 96
C
C        ALL OTHER ENTRIES OF  W  NOT IDENTIFIED IN THIS WAY WITH AN EN-
C        TRY OF  C  ARE NEVER REFERENCED .
C  DIAG.....IS A WORK ARRAY OF LENGTH  NROW .
C
C******  O U T P U T  ******
C  W.....CONTAINS THE CHOLESKY FACTORIZATION  C = L*D*L-TRANSP, WITH
C        W(1,I) CONTAINING  1/D(I,I)
C        AND  W(I,J)  CONTAINING  L(I-1+J,J), I=2,...,NBANDS.
C
C******  M E T H O D  ******
C   GAUSS ELIMINATION, ADAPTED TO THE SYMMETRY AND BANDEDNESS OF  C , IS
C   USED .
C     NEAR ZERO PIVOTS ARE HANDLED IN A SPECIAL WAY. THE DIAGONAL ELE-
C  MENT C(N,N) = W(1,N) IS SAVED INITIALLY IN  DIAG(N), ALL N. AT THE N-
C  TH ELIMINATION STEP, THE CURRENT PIVOT ELEMENT, VIZ.  W(1,N), IS COM-
C  PARED WITH ITS ORIGINAL VALUE, DIAG(N). IF, AS THE RESULT OF PRIOR
C  ELIMINATION STEPS, THIS ELEMENT HAS BEEN REDUCED BY ABOUT A WORD
C  LENGTH, (I.E., IF W(1,N)+DIAG(N) .LE. DIAG(N)), THEN THE PIVOT IS DE-
C  CLARED TO BE ZERO, AND THE ENTIRE N-TH ROW IS DECLARED TO BE LINEARLY
C  DEPENDENT ON THE PRECEDING ROWS. THIS HAS THE EFFECT OF PRODUCING
C   X(N) = 0  WHEN SOLVING  C*X = B  FOR  X, REGARDLESS OF  B. JUSTIFIC-
C  ATION FOR THIS IS AS FOLLOWS. IN CONTEMPLATED APPLICATIONS OF THIS
C  PROGRAM, THE GIVEN EQUATIONS ARE THE NORMAL EQUATIONS FOR SOME LEAST-
C  SQUARES APPROXIMATION PROBLEM, DIAG(N) = C(N,N) GIVES THE NORM-SQUARE
C  OF THE N-TH BASIS FUNCTION, AND, AT THIS POINT,  W(1,N)  CONTAINS THE
C  NORM-SQUARE OF THE ERROR IN THE LEAST-SQUARES APPROXIMATION TO THE N-
C  TH BASIS FUNCTION BY LINEAR COMBINATIONS OF THE FIRST N-1 . HAVING
C  W(1,N)+DIAG(N) .LE. DIAG(N) SIGNIFIES THAT THE N-TH FUNCTION IS LIN-
C  EARLY DEPENDENT TO MACHINE ACCURACY ON THE FIRST N-1 FUNCTIONS, THERE
C  FORE CAN SAFELY BE LEFT OUT FROM THE BASIS OF APPROXIMATING FUNCTIONS
C     THE SOLUTION OF A LINEAR SYSTEM
C                        C*X = B
C  IS EFFECTED BY THE SUCCESSION OF THE FOLLOWING  T W O  CALLS:
C    CALL BCHFAC ( W, NBANDS, NROW, DIAG )        , TO GET FACTORIZATION
C    CALL BCHSLV ( W, NBANDS, NROW, B )           , TO SOLVE FOR X.
C
      INTEGER NBANDS,NROW,   I,IMAX,J,JMAX,N
      REAL W(NBANDS,NROW),DIAG(NROW),   RATIO
      IF (NROW .GT. 1)                  GO TO 9
      IF (W(1,1) .GT. 0.) W(1,1) = 1./W(1,1)
                                        RETURN
C                                        STORE DIAGONAL OF  C  IN  DIAG.
```

```
    9 DO 10 N=1,NROW
   10    DIAG(N) = W(1,N)
C                                                    FACTORIZATION .
      DO 20 N=1,NROW
         IF (W(1,N)+DIAG(N) .GT. DIAG(N)) GO TO 15
         DO 14 J=1,NBANDS
   14       W(J,N) = 0.
                                              GO TO 20
   15    W(1,N) = 1./W(1,N)
         IMAX = MINO(NBANDS-1,NROW - N)
         IF (IMAX .LT. 1)                   GO TO 20
         JMAX = IMAX
         DO 18 I=1,IMAX
            RATIO = W(I+1,N)*W(1,N)
            DO 17 J=1,JMAX
   17          W(J,N+I) = W(J,N+I) - W(J+I,N)*RATIO
            JMAX = JMAX - 1
   18       W(I+1,N) = RATIO
   20 CONTINUE
                                              RETURN
      END



      SUBROUTINE BCHSLV ( W, NBANDS, NROW, B )
C  FROM  * A PRACTICAL GUIDE TO SPLINES *  BY C. DE BOOR
C  SOLVES THE LINEAR SYSTEM     C*X = B   OF ORDER  N R O W  FOR  X
C  PROVIDED  W  CONTAINS THE CHOLESKY FACTORIZATION FOR THE BANDED (SYM-
C  METRIC) POSITIVE DEFINITE MATRIX  C  AS CONSTRUCTED IN THE SUBROUTINE
C    B C H F A C  (QUO VIDE).
C
C******  I N P U T  ******
C  NROW.....IS THE ORDER OF THE MATRIX  C .
C  NBANDS.....INDICATES THE BANDWIDTH OF  C .
C  W.....CONTAINS THE CHOLESKY FACTORIZATION FOR  C , AS OUTPUT FROM
C        SUBROUTINE BCHFAC  (QUO VIDE).
C  B.....THE VECTOR OF LENGTH  N R O W  CONTAINING THE RIGHT SIDE.
C
C******  O U T P U T  ******
C  B.....THE VECTOR OF LENGTH  N R O W  CONTAINING THE SOLUTION.
C
C******  M E T H O D  ******
C  WITH THE FACTORIZATION  C = L*D*L-TRANSPOSE  AVAILABLE, WHERE  L  IS
C  UNIT LOWER TRIANGULAR AND  D  IS DIAGONAL, THE TRIANGULAR SYSTEM
C  L*Y = B  IS SOLVED FOR  Y (FORWARD SUBSTITUTION), Y IS STORED IN  B,
C  THE VECTOR  D**(-1)*Y IS COMPUTED AND STORED IN  B, THEN THE TRIANG-
C  ULAR SYSTEM  L-TRANSPOSE*X = D**(-1)*Y IS SOLVED FOR  X (BACKSUBSTIT-
C  UTION).
      INTEGER NBANDS,NROW,   J,JMAX,N,NBNDM1
      REAL W(NBANDS,NROW),B(NROW)
      IF (NROW .GT. 1)                  GO TO 21
      B(1) = B(1)*W(1,1)
                                        RETURN
```

```
C
C                  FORWARD SUBSTITUTION. SOLVE L*Y = B FOR Y, STORE IN B.
   21 NBNDM1 = NBANDS - 1
      DO 30 N=1,NROW
         JMAX = MINO(NBNDM1,NROW-N)
         IF (JMAX .LT. 1)                    GO TO 30
         DO 25 J=1,JMAX
   25       B(J+N) = B(J+N) - W(J+1,N)*B(N)
   30    CONTINUE
C
C     BACKSUBSTITUTION. SOLVE L-TRANSP.X = D**(-1)*Y  FOR X, STORE IN B.
      N = NROW
   39    B(N) = B(N)*W(1,N)
         JMAX = MINO(NBNDM1,NROW-N)
         IF (JMAX .LT. 1)                    GO TO 40
         DO 35 J=1,JMAX
   35       B(N) = B(N) - W(J+1,N)*B(J+N)
   40    N = N-1
         IF (N .GT. 0)                       GO TO 39
                                             RETURN
      END
```

L2MAIN and its subroutines For the calculation of least-squares spline approximation, for example in the examples to follow, we need to read in the data, construct the appropriate knot sequence from a given break sequence, evaluate the approximation, and calculate and print out various measures of the error. We therefore need a program to coordinate all these activities, including the calculation of a series of approximations with varying number and location of knots. All this is carried out in subprograms **SETDAT**, **L2KNTS** and **L2ERR**, and coordinated in the main program L2MAIN, listed below.

```
C  MAIN PROGRAM FOR LEAST-SQUARES APPROXIMATION BY SPLINES
CALLS SETDAT,L2KNTS,L2APPR(BSPLVB,BCHFAC,BCHSLV),BSPLPP(BSPLVB*)
C     ,L2ERR(PPVALU(INTERV)),PPVALU*,NEWNOT
C
C  THE PROGRAM, THOUGH OSTENSIBLY WRITTEN FOR L2-APPROXIMATION, IS TYP-
C  ICAL FOR PROGRAMS CONSTRUCTING A PP APPROXIMATION TO A FUNCTION GI-
C  VEN IN SOME SENSE. THE SUBPROGRAM  L 2 A P P R , FOR INSTANCE, COULD
C  EASILY BE REPLACED BY ONE CARRYING OUT INTERPOLATION OR SOME OTHER
C  FORM OF APPROXIMATION.
C
C******  I N P U T  ******
C  IS EXPECTED IN  S E T D A T  (QUO VIDE), SPECIFYING BOTH THE DATA TO
C  BE APPROXIMATED AND THE ORDER AND BREAKPOINT SEQUENCE OF THE PP AP-
C  PROXIMATING FUNCTION TO BE USED. FURTHER,  S E T D A T  IS EXPECTED
C  TO  T E R M I N A T E  THE RUN (FOR LACK OF FURTHER INPUT OR BECAUSE
C   I C O U N T  HAS REACHED A CRITICAL VALUE).
C     THE NUMBER  N T I M E S  IS READ IN IN THE MAIN PROGRAM. IT SPECI
C  FIES THE NUMBER OF PASSES THROUGH THE KNOT IMPROVEMENT ALGORITHM IN
C  N E W N C T  TO BE MADE. ALSO,  A D D B R K  IS READ IN TO SPECIFY
C  THAT, ON THE AVERAGE, ADDBRK KNOTS ARE TO BE ADDED PER PASS THROUGH
C  NEWNOT. FOR EXAMPLE,  ADDBRK = .34  WOULD CAUSE A KNOT TO BE ADDED
C  EVERY THIRD PASS (AS LONG AS  NTIMES .LT. 50).
C
```

```
C******  P R I N T E D  O U T P U T  ******
C  IS GOVERNED BY THE THREE PRINT CONTROL HOLLERITH STRINGS
C  P R B C O  = 'ON'  GIVES PRINTOUT OF B-SPLINE COEFFS. OF APPROXIM.
C  P R P C O  = 'ON'  GIVES PRINTOUT OF PP REPR. OF APPROXIMATION.
C  P R F U N  = 'ON'  GIVES PRINTOUT OF APPROXIMATION AND ERROR AT
C                     EVERY DATA POINT.
C  THE ORDER  K , THE NUMBER OF PIECES  L, AND THE INTERIOR BREAKPOINTS
C  ARE ALWAYS PRINTED OUT AS ARE (IN L2ERR) THE MEAN, MEAN SQUARE, AND
C  MAXIMUM ERRORS IN THE APPROXIMATION.
C
      INTEGER I,ICOUNT,II,J,K,L,LBEGIN,LNEW,LL,LPKMAX,LTKMAX,N,NT,NTAU
     *        ,NTIMES,NTMAX,ON,PRBCO,PRFUN,PRPCO
      PARAMETER (LPKMAX=100,NTMAX=200,LTKMAX=2000)
      REAL ADDBRK,BCOEF(LPKMAX),BREAK,COEF,GTAU,Q(LTKMAX),SCRTCH(NTMAX)
     *    ,T(NTMAX),TAU,TOTALW,WEIGHT
      COMMON / DATA / NTAU, TAU(NTMAX),GTAU(NTMAX),WEIGHT(NTMAX),TOTALW
      COMMON /APPROX/ BREAK(LPKMAX),COEF(LTKMAX),L,K
      DATA ON /'ON'/
C
      ICOUNT = 0
C        I C O U N T  PROVIDES COMMUNICATION WITH THE DATA-INPUT-AND-
C     TERMINATION ROUTINE  S E T D A T . IT IS INITIALIZED TO  0  TO
C     SIGNAL TO SETDAT WHEN IT IS BEING CALLED FOR THE FIRST TIME. AFTER
C     THAT, IT IS UP TO SETDAT TO USE ICOUNT FOR KEEPING TRACK OF THE
C     PASSES THROUGH SETDAT .
C
C     INFORMATION ABOUT THE FUNCTION TO BE APPROXIMATED AND ORDER AND
C     BREAKPOINT SEQUENCE OF THE APPROXIMATING PP FUNCTIONS IS GATHERED
C     BY A
    1 CALL SETDAT(ICOUNT)
C
C     BREAKPOINTS ARE TRANSLATED INTO KNOTS, AND THE NUMBER  N  OF
C     B-SPLINES TO BE USED IS OBTAINED BY A
      CALL L2KNTS ( BREAK, L, K, T, N )
C
C     THE INTEGER  N T I M E S  AND THE REAL  A D D B R K  ARE REQUESTED
C     AS WELL AS THE PRINT CONTROLS  P R B C O ,  P R P C O  AND
C     P R F U N .  NTIMES  PASSES  ARE MADE THROUGH THE SUBROUTINE NEW-
C     NOT, WITH AN INCREASE OF  ADDBRK  KNOTS FOR EVERY PASS .
      PRINT 600
  600 FORMAT(' NTIMES,ADDBRK , PRBCO,PRPCO,PRFUN =? (I3,F10.5/3A2)')
      READ 500,NTIMES,ADDBRK,PRBCO,PRPCO,PRFUN
  500 FORMAT(I3,F10.5/3A2)
C
      LBEGIN = L
      NT = 0
C        THE B-SPLINE COEFFS.  B C O E F  OF THE L2-APPROX. ARE OBTAIN-
C        ED BY A
   10    CALL L2APPR ( T, N, K, Q, SCRTCH, BCOEF )
         IF (PRBCO .EQ. ON)  PRINT 609, (BCOEF(I),I=1,N)
  609    FORMAT(//' B-SPLINE COEFFICIENTS'/(4E20.10))
C
C        CONVERT THE B-REPR. OF THE APPROXIMATION TO PP REPR.
         CALL BSPLPP ( T, BCOEF, N, K, Q, BREAK, COEF, L )
         PRINT 610, K, L, (BREAK(LL),LL=2,L)
  610    FORMAT(//' APPROXIMATION BY SPLINES OF ORDER',I3,' ON ',
     *          I3,' INTERVALS. BREAKPOINTS -'/(4E20.10))
         IF (PRPCO .NE. ON)              GO TO 15
         PRINT 611
  611    FORMAT(/' PP-REPRESENTATION FOR APPROXIMATION')
         DO 12 I=1,L
            II = (I-1)*K
   12       PRINT 613,BREAK(I),(COEF(II+J),J=1,K)
  613    FORMAT(F9.3,4E20.10/(11X,4E20.10))
C
```

```
C       COMPUTE AND PRINT OUT VARIOUS ERROR NORMS BY A
   15   CALL L2ERR ( PRFUN, SCRTCH, Q )
C
CCC     IF NEWNOT HAS BEEN APPLIED LESS THAN  N T I M E S  TIMES, TRY
CCC     IT AGAIN TO OBTAIN, FROM THE CURRENT APPROX. A POSSIBLY IMPROV-
CCC     ED SEQUENCE OF BREAKPOINTS WITH  ADDBRK  MORE BREAKPOINTS (ON
CCC     THE AVERAGE) THAN THE CURRENT APPROXIMATION HAS.
CCC        IF ONLY AN INCREASE IN BREAKPOINTS IS WANTED, WITHOUT THE
CCC     ADJUSTMENT THAT NEWNOT PROVIDES, A FAKE NEWNOT ROUTINE COULD BE
CCC     USED HERE WHICH MERELY RETURNS THE BREAKPOINTS FOR  L N E W
C       EQUAL INTERVALS .
        IF (NT .GE. NTIMES)                GO TO 1
        LNEW = LBEGIN + FLOAT(NT)*ADDBRK
        CALL NEWNOT (BREAK, COEF, L, K, SCRTCH, LNEW, T )
        CALL L2KNTS ( SCRTCH, LNEW, K, T, N )
        NT = NT + 1
                                           GO TO 10

      END



      SUBROUTINE L2KNTS ( BREAK, L, K, T, N )
C  TO BE CALLED IN MAIN PROGRAM  L 2 M A I N .
CONVERTS THE BREAKPOINT SEQUENCE  B R E A K  INTO A CORRESPONDING KNOT
C  SEQUENCE  T  TO ALLOW THE REPR. OF A PP FUNCTION OF ORDER  K  WITH
C  K-2 CONTINUOUS DERIVATIVES AS A SPLINE OF ORDER  K  WITH KNOT
C  SEQUENCE  T . THIS MEANS THAT
C  T(1), ..., T(N+K) =  BREAK(1) K TIMES, THEN BREAK(I), I=2,...,L, EACH
C                       ONCE, THEN BREAK(L+1) K TIMES .
C
C  THEREFORE,  N = K-1 + L.
C
C******  I N P U T  ******
C  K      THE ORDER
C  L      THE NUMBER OF POLYNOMIAL PIECES
C  BREAK(1), ...,BREAK(L+1)  THE BREAKPOINT SEQUENCE
C
C******  O U T P U T  ******
C  T(1),...,T(N+K)   THE KNOT SEQUENCE
C  N      THE DIMENSION OF THE CORRESP. SPLINE SPACE OF ORDER  K .
C
      INTEGER K,L,N,   I,KM1
      REAL BREAK(1),T(1)
C     DIMENSION BREAK(L+1),T(N+K)
      KM1 = K - 1
      DO 5 I=1,KM1
    5    T(I) = BREAK(1)
      DO 6 I=1,L
    6    T(KM1+I) = BREAK(I)
      N = KM1 + L
      DO 7 I=1,K
    7    T(N+I) = BREAK(L+1)

      END
```

RETURN

```
      SUBROUTINE L2ERR ( PRFUN , FTAU , ERROR )
C  THIS ROUTINE IS TO BE CALLED IN THE MAIN PROGRAM  L 2 M A I N .
CALLS SUBPROGRAM  PPVALU(INTERV)
C  THIS SUBROUTINE COMPUTES VARIOUS ERRORS OF THE CURRENT L2-APPROXI-
C  MATION , WHOSE PP-REPR. IS CONTAINED IN COMMON BLOCK  APPROX  ,
C  TO THE GIVEN DATA CONTAINED IN COMMON BLOCK  DATA . IT PRINTS OUT
C  THE AVERAGE ERROR  E R R L 1 , THE L2-ERROR  E R R L 2,  AND THE
C  MAXIMUM ERROR  E R R M A X .
C
C******  I N P U T  ******
C  PRFUN  A HOLLERITH STRING.  IF PRFUN = 'ON', THE ROUTINE PRINTS OUT
C         THE VALUE OF THE APPROXIMATION AS WELL AS ITS ERROR AT
C         EVERY DATA POINT.
C
C******  O U T P U T  ******
C  FTAU(1), ..., FTAU(NTAU),  WITH  FTAU(I)  THE APPROXIMATION  F AT
C         TAU(I), ALL I.
C  ERROR(1), ..., ERROR(NTAU),  WITH  ERROR(I) = SCALE*(G - F)
C         AT TAU(I), ALL I. HERE,  S C A L E  EQUALS  1. IN CASE
C         PRFUN .NE. 'ON' , OR THE ABS.ERROR IS GREATER THAN 100 SOME-
C         WHERE. OTHERWISE, S C A L E  IS SUCH THAT THE MAXIMUM OF
C         ABS(ERROR))  OVER ALL  I  LIES BETWEEN  10  AND  100. THIS
C         MAKES THE PRINTED OUTPUT MORE ILLUSTRATIVE.
C
      INTEGER PRFUN,   IE,K,L,LL,LPKMAX,LTKMAX,NTAU,NTMAX,ON
      REAL FTAU(1),ERROR(1),  BREAK,COEF,ERR,ERRMAX,ERRL1,ERRL2
     *                       ,GTAU,SCALE,TAU,TOTALW,WEIGHT
      DIMENSION FTAU(NTAU),ERROR(NTAU)
      PARAMETER (LPKMAX=100,NTMAX=200,LTKMAX=2000)
      COMMON / DATA / NTAU, TAU(NTMAX),GTAU(NTMAX),WEIGHT(NTMAX),TOTALW
      COMMON /APPROX/ BREAK(LPKMAX),COEF(LTKMAX),L,K
      DATA ON /'ON'/
      ERRL1 = 0.
      ERRL2 = 0.
      ERRMAX = 0.
      DO 10 LL=1,NTAU
         FTAU(LL) = PPVALU (BREAK, COEF, L, K, TAU(LL), 0 )
         ERROR(LL) = GTAU(LL) - FTAU(LL)
         ERR = ABS(ERROR(LL))
         IF (ERRMAX .LT. ERR)   ERRMAX = ERR
         ERRL1 = ERRL1 + ERR*WEIGHT(LL)
 10      ERRL2 = ERRL2 + ERR**2*WEIGHT(LL)
      ERRL1 = ERRL1/TOTALW
      ERRL2 = SQRT(ERRL2/TOTALW)
      PRINT 615,ERRL2,ERRL1,ERRMAX
  615 FORMAT(///' LEAST SQUARE ERROR =',E20.6/
     1           ' AVERAGE ERROR      =',E20.6/
     2           ' MAXIMUM ERROR      =',E20.6//)
      IF (PRFUN .NE. ON)                    RETURN
C     **  SCALE ERROR CURVE AND PRINT  **
      IE = 0
      SCALE = 1.
      IF (ERRMAX .GE. 10.)                  GO TO 18
      DO 17 IE=1,9
         SCALE = SCALE*10.
         IF (ERRMAX*SCALE .GE. 10.)         GO TO 18
 17      CONTINUE
 18   DO 19 LL=1,NTAU
 19      ERROR(LL) = ERROR(LL)*SCALE
      PRINT 620,IE,(LL,TAU(LL),FTAU(LL),ERROR(LL),LL=1,NTAU)
  620 FORMAT(///14X,'APPROXIMATION AND SCALED ERROR CURVE'/7X,
     1'DATA POINT',7X,'APPROXIMATION',3X,'DEVIATION X 10**',I1/
     2(I4,F16.8,F16.8,F17.6))
                                            RETURN
      END
```

The use of L2APPR We begin with a discussion of an important diagnostic aid, namely the number of sign changes in the error

$$e := \text{`}g\text{'} - f^*$$

of the least-squares approximation f^* from $\$_{k,\mathbf{t}}$. We first prove that, as in continuous least-squares approximation (see the end of preceding chapter) from an n-dimensional spline space, the error must have at least n sign changes.

By Lemma (16), we know that the error e must be orthogonal to $\$_{k,\mathbf{t}}$, that is, we must have

$$\langle f, e\rangle = 0 \qquad \text{for all } f \in \$_{k,\mathbf{t}}.$$

This implies that the (rectangular) matrix

$$A := (\mathrm{B}_i(\tau_j))_{i=1}^{n};{}_{j=1}^{N}$$

maps the N-vector $\mathbf{e} := (e(\tau_j)w_j)_1^N$ to zero. The matrix A is totally positive by Theorem XIII(7), hence, by a strengthening of Proposition XIII(6),

$$\operatorname{rank} A \le \mathrm{S}^+\mathbf{e}, \tag{24}$$

unless $\mathbf{e}$ is identically zero. Here, $\mathrm{S}^+\mathbf{e}$ denoted the number of *weak* sign changes in the sequence $\mathbf{e}$, that is, the maximal number of sign changes achievable by an appropriate assignment of signs to the zero entries of $\mathbf{e}$. For a proof of (24), see, for example, Karlin [1968:Theorem V.2.2].

By taking account of the bandedness of the matrix A, one proves

(25) Lemma. *Assume that $\|\cdot\|_2$ is a norm on $\$_{k,\mathbf{t}}$ (that is, $\operatorname{rank} A = n$), and that the error $e = \text{`}g\text{'} - f^*$ in the least squares approximation to g from $\$_{k,\mathbf{t}}$ is not identically zero on τ. Then, the sequence $(e(\tau_j))_1^N$ has at least n (weak) sign changes, with the ith of these n sign changes occurring "in the support" of B_i, $i = 1, \ldots, n$.*

(26) Example: An approximation with fewer sign changes in the error than perhaps expected We calculate the discrete least-squares approximation in the specific case $g(x) = x^2 + 1$ and $\$_{k,\mathbf{t}} = \Pi_{<2,\boldsymbol{\xi}} \cap C[0..1]$ with

$$\begin{aligned}\boldsymbol{\xi} &= ((i-1)/6 : i = 1, \ldots, 7)\\ \boldsymbol{\tau} &= (0, 1/2, 3/4, 7/8, \ldots, 1 - 2^{-8}, 1),\\ w_i &= 1, \quad \text{all } i.\end{aligned}$$

An appropriate subroutine SETDAT, to be used in L2MAIN, is then

```
      SUBROUTINE SETDAT(ICOUNT)
C  TO BE CALLED IN MAIN PROGRAM  L 2 M A I N .
C     THIS ROUTINE IS SET UP TO PROVIDE THE SPECIFIC DATA FOR EXAMPLE 2
C     IN CHAPTER XIV. FOR A GENERAL PURPOSE L2-APPROXIMATION PROGRAM, IT
C     WOULD HAVE TO BE REPLACED BY A SUBROUTINE READING IN
C           NTAU, TAU(I), GTAU(I), I=1,...,NTAU
C     AND READING IN OR SETTING
C            K, L, BREAK(I),I=1,...,L+1,  AND WEIGHT(I),I=1,...,NTAU,
C           AS WELL AS  TOTALW = SUM( WEIGHT(I) , I=1,...,NTAU).
C    I C O U N T  IS EQUAL TO ZERO WHEN SETDAT IS CALLED IN  L 2 M A I N
C     FOR THE FIRST TIME. AFTER THAT, IT IS UP TO SETDAT TO USE ICOUNT
C     FOR KEEPING TRACK OF THE PASSES THROUGH SETDAT . THIS IS IMPORTANT
C     SINCE L2MAIN RELIES ON SETDAT FOR  T E R M I N A T I O N .
      INTEGER ICOUNT,  I,K,L,LPKMAX,LP1,LTKMAX,NTAU,NTAUM1,NTMAX
      REAL BREAK,COEF,GTAU,STEP,TAU,TOTALW,WEIGHT
      PARAMETER (LPKMAX=100,NTMAX=200,LTKMAX=2000)
      COMMON / DATA / NTAU, TAU(NTMAX),GTAU(NTMAX),WEIGHT(NTMAX),TOTALW
      COMMON /APPROX/ BREAK(LPKMAX),COEF(LTKMAX),L,K
      IF (ICOUNT .GT. 0)                    STOP
      ICOUNT = ICOUNT + 1
      NTAU = 10
      NTAUM1 = NTAU-1
      DO 8 I=1,NTAUM1
    8    TAU(I) = 1. - .5**(I-1)
      TAU(NTAU) = 1.
      DO 9 I=1,NTAU
    9    GTAU(I) = TAU(I)**2  + 1.
      DO 10 I=1,NTAU
   10    WEIGHT(I) = 1.
      TOTALW = NTAU
      L = 6
      LP1 = L+1
      STEP = 1./FLOAT(L)
      K = 2
      DO 11 I=1,LP1
   11    BREAK(I) = (I-1)*STEP
                                            RETURN
      END
```

Here is (input and) output from L2MAIN and its subroutines for this particular choice of SETDAT.

```
 NTIMES,ADDBRK , PRBCO,PRPCO,PRFUN =? (I3,F10.5/3A2)
  0  0.
ONONON


 B-SPLINE COEFFICIENTS
    0.1000000000+01    0.0000000000+00    0.0000000000+00    0.1250000000+01
    0.1439158797+01    0.1685841441+01    0.1998644829+01


 APPROXIMATION BY SPLINES OF ORDER  2 ON   6 INTERVALS. BREAKPOINTS -
    0.1666666716+00    0.3333333433+00    0.5000000000+00    0.6666666865+00
    0.8333333731+00
```

```
PP-REPRESENTATION FOR APPROXIMATION
    0.000     0.1000000000+01    -0.6000000000+01
    0.167     0.0000000000+00     0.0000000000+00
    0.333     0.0000000000+00     0.7500000477+01
    0.500     0.1250000000+01     0.1134952664+01
    0.667     0.1439158797+01     0.1480095744+01
    0.833     0.1685841441+01     0.1876820803+01

LEAST SQUARE ERROR =          0.116994-02
AVERAGE ERROR      =          0.856149-03
MAXIMUM ERROR      =          0.243723-02

                APPROXIMATION AND SCALED ERROR CURVE
      DATA POINT          APPROXIMATION    DEVIATION X 10**4
 1    0.00000000          1.00000000            0.000000
 2    0.50000000          1.25000000            0.000000
 3    0.75000000          1.56250012           -0.001192
 4    0.87500000          1.76404226           15.827417
 5    0.93750000          1.88134348          -24.372339
 6    0.96875000          1.93999422          -15.176535
 7    0.98437500          1.96931958           -3.254414
 8    0.99218750          1.98398221            4.538298
 9    0.99609375          1.99131346            8.893013
10    1.00000000          1.99864483           13.551712
```

Our spline space in this example is of dimension $n = 7$ $(= l + k - 1)$. We would therefore expect at least seven sign changes in the error. In fact, we get only five, even if we take the first two entries in the error vector to be zero (as they would be in exact arithmetic) and so obtain three weak sign changes in the first four entries of the error vector. A closer look at breaks and data sites shows that B_2 and B_3 in the B-spline sequence for $\mathbf{t}$ vanish on τ, that is, are linearly dependent on the others in a trivial way. Thus, $\$_{k,\mathbf{t}}$ is only of dimension 5 when restricted to τ, and (24) only promises five sign changes under these circumstances. □

This example also illustrates that the subprogram BCHFAC/BCHSLV is equipped to deal with such singularity of the normal system, but also shows that you may not like the result (which, in this case, is identical with the so-called "best" least-squares approximation).

Lemma (25) provides a simple check that a program for least squares spline approximation works correctly. It also points out that the least-squares spline approximant could have been gotten, in principle, by interpolation, though not usually at some of the given data sites. This shows that least-squares approximation is perhaps not the method of choice for the approximation to a smooth and exactly known function, since the construction of an interpolant, as described in Chapter XIII, is considerably cheaper and just as effective *provided* one has a reasonable way for choosing

appropriate interpolation sites.

Least-squares approximation is very suitable for the recovery of a smooth function from noisy information. In this case, one thinks of the data g_i as being of the form

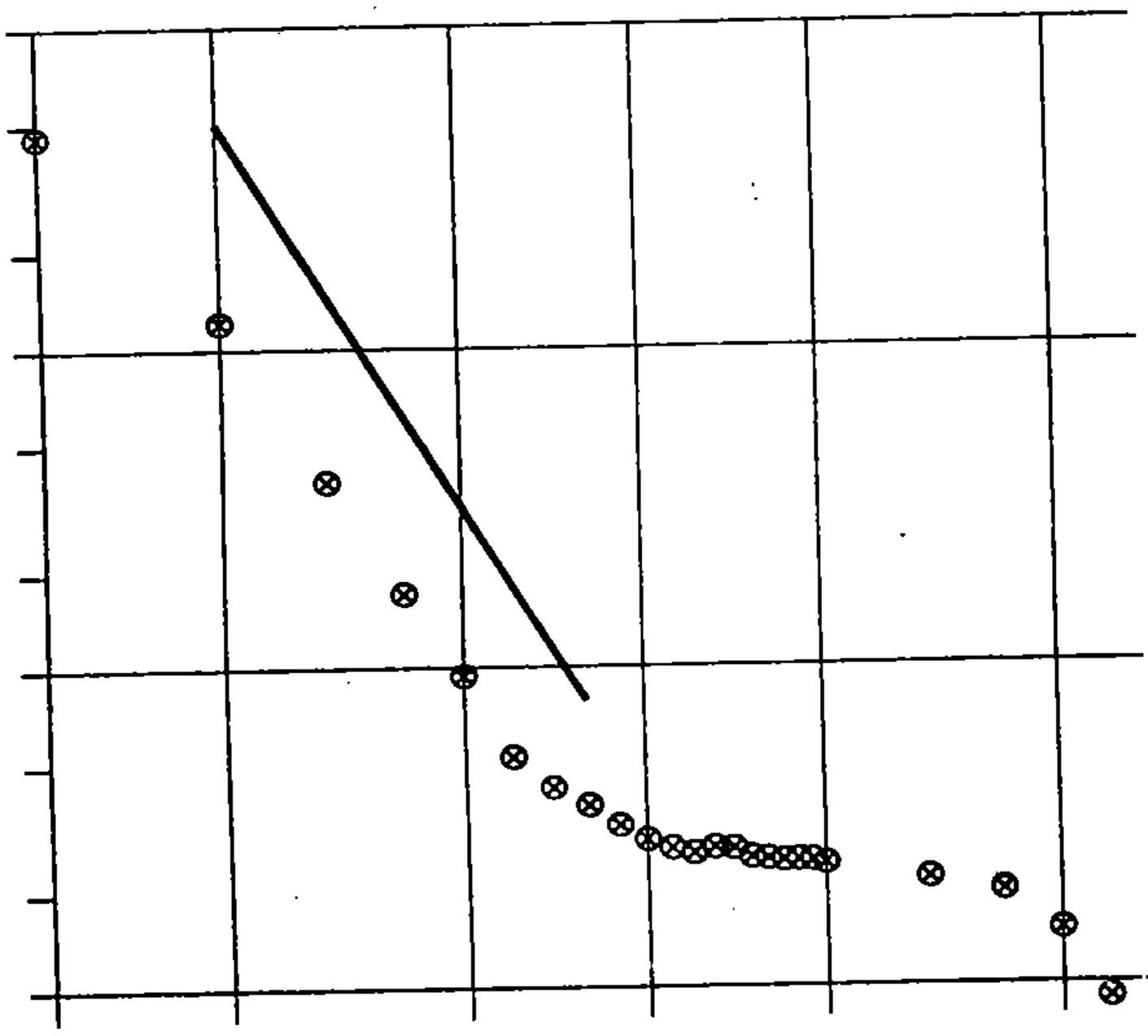
$$g_i = g(\tau_i) + \delta_i,$$

with g some "smooth", or "slowly varying", function to be recovered and the sequence (δ_i) quite "rough", or "fast varying", distributed around 0, but of smaller magnitude than g. The trick is to approximate from a function class \$ that is flexible enough to approximate the underlying information g while still orthogonal to the noise, that is, unable to follow oscillations in the sequence (δ_i).

In this situation, one would expect the error vector of a successful fit to look like "noise", that is, to have many more sign changes than the approximating class has degrees of freedom. One would also expect the error not to change much in size when increasing the available degrees of freedom slightly. In fact, one would expect the error to decrease in size initially as one approximates from subspaces with an ever increasing dimension until one reaches a point at which the error consists essentially of noise. Of course, going *much* beyond this point will eventually lead to zero error since, with sufficiently many degrees of freedom, one can simply interpolate. These considerations are illustrated in the next example.

(27) Example: The noise plateau in the error We construct the least-squares approximation to rounded values of the function $g(x) = e^x$ at 65 sites in [0..3] from parabolic splines with uniformly spaced simple knots. We start with no interior knots and then increase the number of knots by 1 each time, using a fake subroutine NEWNOT which merely outputs the new breaks uniformly spaced. Here is an appropriate subroutine SETDAT.

```
      SUBROUTINE SETDAT(ICOUNT)
C  TO BE CALLED IN MAIN PROGRAM  L 2 M A I N .
C     THIS ROUTINE IS SET UP TO PROVIDE THE SPECIFIC DATA FOR EXAMPLE 3
C     IN CHAPTER XIV.
C
      INTEGER ICOUNT,  I,K,L,LPKMAX,LTKMAX,NTAU,NTMAX
      REAL BREAK,COEF,GTAU,STEP,TAU,TOTALW,WEIGHT
      PARAMETER (LPKMAX=100,NTMAX=200,LTKMAX=2000)
      COMMON / DATA / NTAU, TAU(NTMAX),GTAU(NTMAX),WEIGHT(NTMAX),TOTALW
      COMMON /APPROX/ BREAK(LPKMAX),COEF(LTKMAX),L,K
      ROUND(X) = FLOAT(IFIX(X*100.))/100.
      IF (ICOUNT .GT. 0)                    STOP
      ICOUNT = ICOUNT + 1
      NTAU = 65
      STEP = 3./FLOAT(NTAU-1)
      DO 10 I=1,NTAU
         TAU(I) = (I-1)*STEP
         GTAU(I) = ROUND(EXP(TAU(I)))
   10    WEIGHT(I) = 1.
```



(28) FIGURE. As the number of intervals (or polynomial pieces) increases, the least-square error decreases until it reaches a plateau whose height is determined by the noise level in the data.

```
TOTALW = NTAU
L = 1
BREAK(1) = TAU(1)
BREAK(2) = TAU(NTAU)
K = 3
                                        RETURN
END
```

The output for this example is summarized in Figure (28) where the logarithm of the least-square error is plotted versus the logarithm of the number n of intervals. According to Theorem XII(6),

$$\operatorname{dist}(g, \$_{3,\mathbf{t}}) \sim \mathcal{O}\left(|t|^3\right).$$

We would therefore expect the error to decay initially like $\mathcal{O}(n^{-3})$. We have drawn in, therefore, for comparison, a straight line with slope -3. Note that the error reaches a plateau around $n = 10$, of height $\sim .0025$. This corresponds nicely to the fact that we rounded the exact values to 2 digits after the decimal point. Note also that we eventually have enough degrees of freedom to approximate the noise as well; there is real improvement again around $n = 50$ while, for $n \geq 65$, the error is $2 * 10^{-7}$, that is, roundoff. □

(29) Example: Once more the Titanium Heat data In this example, we use the "optimal" knots derived in Example XIII(29) for interpolation to some of the data points of the Titanium Heat Data. But this time, we construct a least-squares approximation to *all* the data. We also increase the number of knots repeatedly, each time by 2, using NEWNOT to obtain a good distribution for them.

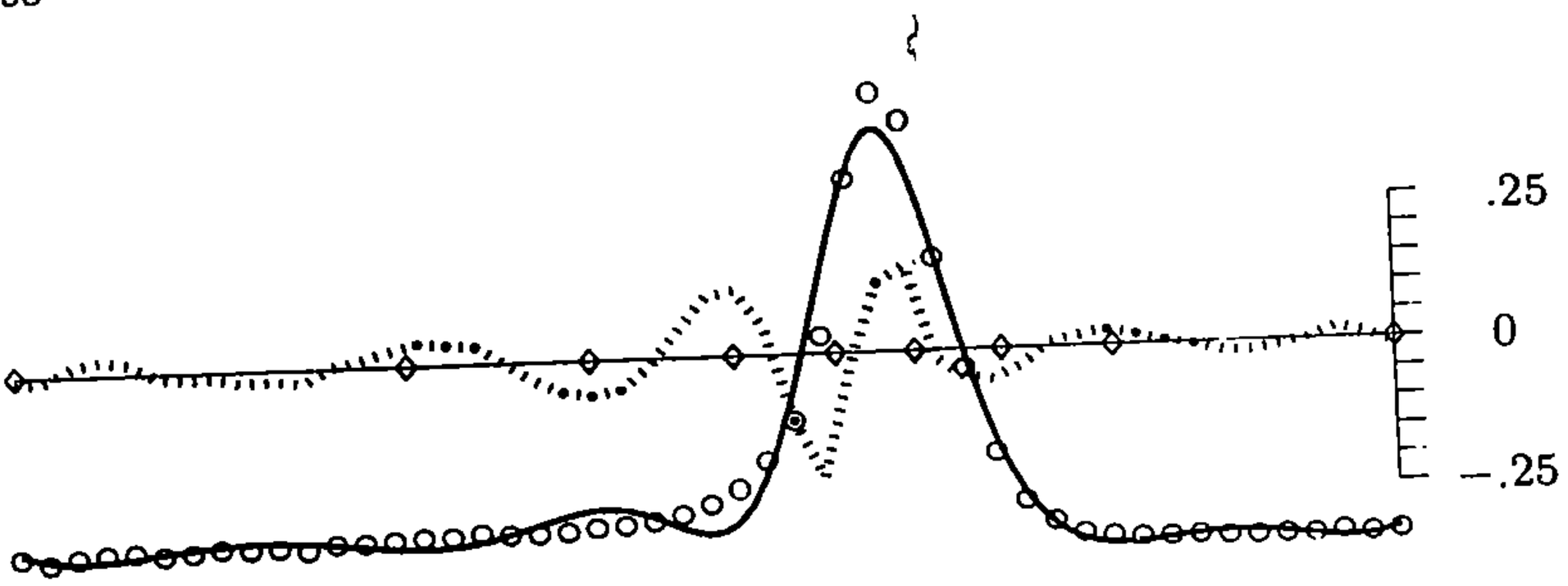
Here is an appropriate version of SETDAT.

```
      SUBROUTINE SETDAT(ICOUNT)
C  TO BE CALLED IN MAIN PROGRAM  L 2 M A I N .
C        CALLS TITAND
C     THIS ROUTINE IS SET UP TO PROVIDE THE SPECIFIC DATA FOR EXAMPLE 4
C     IN CHAPTER XIV.
      INTEGER ICOUNT,  I,K,L,LPKMAX,LTKMAX,N,NTAU,NTMAX
      REAL BREAK,BRKPIC(9),COEF,GTAU,TAU,TOTALW,WEIGHT
      PARAMETER (LPKMAX=100,NTMAX=200,LTKMAX=2000)
      COMMON / DATA / NTAU, TAU(NTMAX),GTAU(NTMAX),WEIGHT(NTMAX),TOTALW
      COMMON /APPROX/ BREAK(LPKMAX),COEF(LTKMAX),L,K
      DATA BRKPIC/595.,730.985,794.414,844.476,880.06,907.814,
     *            938.001,976.752,1075./, N/9/
      IF (ICOUNT .GT. 0)                    STOP
      ICOUNT = ICOUNT + 1
      CALL TITAND ( TAU, GTAU, NTAU )
      DO 10 I=1,NTAU
   10    WEIGHT(I) = 1.
      TOTALW = NTAU
      L = N-1
      K = 5
      DO 11 I=1,N
   11    BREAK(I) = BRKPIC(I)
                                            RETURN
      END
```

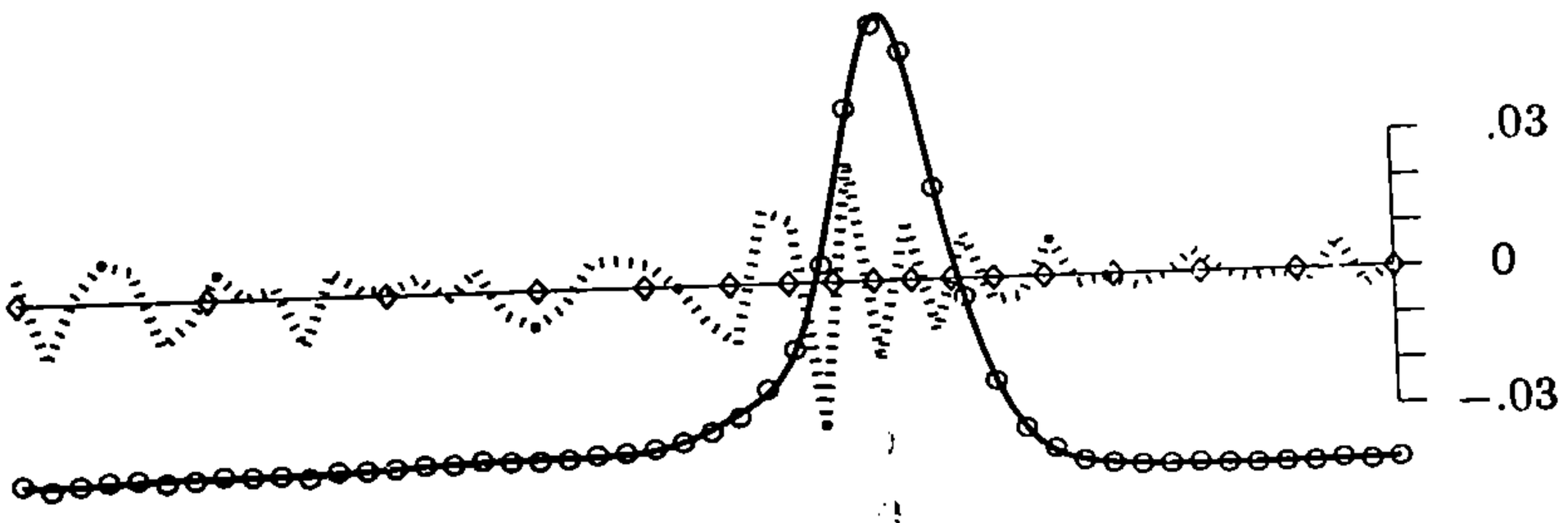
Instead of a printout, we show data and approximation in a picture, with an enlarged plot of the error (dashed) superimposed. The error is plotted in the scale indicated on the right. The breaks used are also indicated, as diamonds on the zero line for the error. The approximation (see Figure (30)) is considerably better than that achieved in Chapter XIII by interpolation (see Figure XIII(30)). But, the error curve still has quite regular behavior and the plot of the approximation itself also indicates that it does not as yet contain all the information in the data. In particular, the error has 12 sign changes, just equal to the dimension of the approximating spline space.

After four passes through NEWNOT, we have twice as many intervals, and approximation and error now look quite satisfactory; see Figure (31). In particular, the error now has 28 sign changes while the dimension of the approximating spline space is only 20. It seems reasonable that a less smooth approximation, for example, a spline of order 3 or 4 or a spline of order 5 but only $C^{(1)}$ or $C^{(2)}$, would have been more effective for these data (see Problem 7).

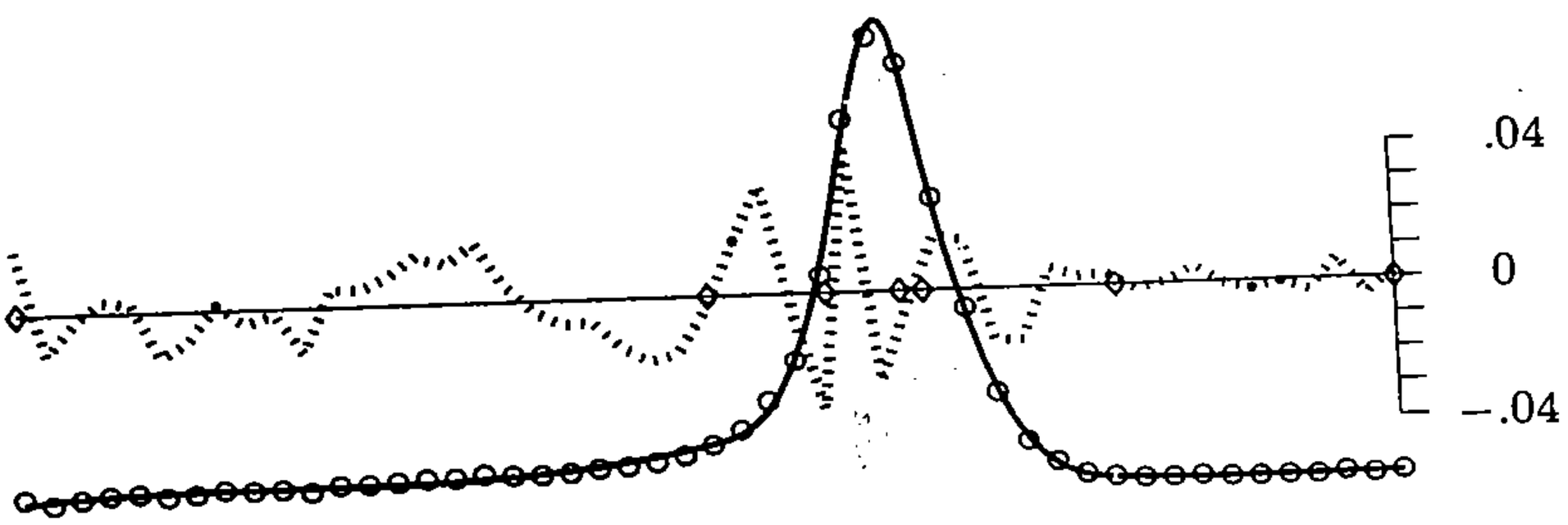
We alert the reader to Chapter 5 of Hayes [1970], written by M.J.D. Powell, for a different approach to least-squares (cubic) spline approximation (see Problem 8).



(30) FIGURE. Least-squares approximation (solid) to data (o) by quartic splines with seven interior knots (◇), with the error also shown along with its scale.



(31) FIGURE. Same as previous figure, except that approximation has fifteen interior knots, chosen by NEWNOT.



(32) FIGURE. Least-squares approximation (solid) to data (o) by cubic splines with five free (interior) knots (◇); also the error, in its own scale as shown on the right.

□

Least-squares approximation by splines with variable knots We bring up once more the topic of best approximation by splines with *variable* knots, this time in the specific context of least-squares approximation. This means that we consider least-squares approximation from the function class

$$(33) \qquad \$_{k,n} := \bigcup\{ f \in \$_{k,\mathbf{t}} : t_1 = \cdots = t_k = a, t_{n+1} = \cdots = t_{n+k} = b \}.$$

There are many difficulties in constructing best approximations from this class, all stemming from the fact that this class is *nonlinear*, that is, the sum of two functions in $\$_{k,n}$, is, in general, not in $\$_{k,n}$. This makes it essentially impossible to characterize a best approximation, that is, to give computationally useful criteria by which a best approximation can be recognized and distinguished from other approximations. For the linear case, we did this earlier by showing that a best approximation must satisfy the normal equations and vice versa. It is also possible to give the equivalent of "normal equations" in the nonlinear case. But, while every best approximation must satisfy these, not every solution of these equations is a best approximation. Connected with this is the fact that there may be many best approximations to a given function from $\$_{k,n}$. (If a ball merely touches a *plane*, it does so at exactly one point. But a ball may touch a *curved surface* at many points.)

The upshot of all this is that it is possible only to check whether a proposed approximation is best *locally*. This means that one can try, numerically, to improve a proposed approximation by "small" changes in the parameters describing it.

A program for calculating such locally best approximations by cubic splines with variable knots can be found in de Boor & Rice [1968]. A version of the program is available as code ICSVKU in the IMSL library [1977]. Jupp [1978] has discussed a peculiar difficulty that arises in the numerical construction of least-squares approximations from $\$_{k,n}$ and has proposed an effective remedy.

Already Chapter XII contains a discussion of the merit of finding best knots. To reiterate, use of a code for finding a (locally) best approximation from $\$_{k,n}$ is expensive. It is warranted only when precise placement of some knots is essential for the quality of the approximation (for example, when the data exhibit some discontinuity) *and* an approximation with as few parameters as possible is wanted. Otherwise, an approximation with two or three times a many well chosen knots is much cheaper to obtain and, usually, just as effective.

(34) Example: Least-square approximation to the Titanium Heat data from $\$_{4,9}$ In de Boor & Rice [1968], a (locally) best approximation was obtained for the data in Examples XII(40) and (29) by splines of order 4 with five simple (interior) knots. Subsequent investigations by Jupp [1978]

showed this to be very nearly a best approximation.

Figure (32) shows the approximation and the error. The approximation is about as good as the one we obtained in Example (29) with quartic splines with 15 interior knots. Note, though, the regular behavior of the error in the first knot interval. Two or three more knots would take care of it. □

Problems

1. Modify the proof of Theorem XIII(34) in order to establish the following facts about existence and optimality of the "natural" spline interpolant:
(i) Using the definition of $\$^{\mathbf{nat}}_{2m,\mathbf{x}}$ given in the text, show that $f \in \$^{\mathbf{nat}}_{2m,\mathbf{x}}$ if and only if $D^m f \in \$_{m,\mathbf{x}}$ but orthogonal to $\Pi_{<m}$, that is, $\int_{x_1}^{x_N} D^m f(x) q(x)\,dx = 0$ for all $q \in \Pi_{<m}$.
(ii) If $N \geq m$, then there exists, for given g, exactly one $f \in \$^{\mathbf{nat}}_{2m,\mathbf{x}}$ that agrees with g at $\mathbf{x} = (x_i)_1^N$. (Hint: Show that the homogeneous system $\int \mathrm{B}_{i,m,\mathbf{x}}(x)\varphi(x)\,dx = 0$, $i = 1, \ldots, N-m$, has only the trivial solution $\varphi = 0$ in $\$_{m,\mathbf{x}}$, for example, by Lemma XIII(38) and its corollary and the Schoenberg-Whitney theorem, or, by the linear independence of $(\mathrm{B}_{i,m,\mathbf{x}})$ and Lemma (16). Conclude that the linear system

$$\int \mathrm{B}_{i,m,\mathbf{x}}(x)\varphi(x)\,dx/m! = \frac{1}{m}(x_{i+m} - x_i)[x_i, \ldots, x_{i+m}]g, \quad i = 1, \ldots, N-m,$$

has a solution $\varphi \in \$_{m,\mathbf{x}}$, and then construct $f \in \$^{\mathbf{nat}}_{2m,\mathbf{x}}$ as $f = p + \int (\cdot - s)_+^{m-1} \varphi(s)\,ds/(m-1)!$ for an appropriate $p \in \Pi_{<m}$; etc.)
(iii) Conclude from (ii) that $\int \left(f^{(m)}(s)\right)^2 ds \leq \int \left(h^{(m)}\right)^2 ds$ *for any smooth function* h *that agrees with* g at $\mathbf{x}$, *with equality if and only if* $h^{(m)} = f^{(m)}$.
(iv) From (ii), develop a computational scheme for the construction of the "natural" interpolant f.

2. Prove that $S(f_p)$ (with S given by (2)) strictly decreases as p goes from 0^+ to 1^-, in case $S(f_0^+) \neq 0$. (Hint: Set $F_q(f) := qS(f) + (1-q)T(f)$, with $T(f) := \int (f^{(m)}(x))^2\,dx$. Then $F : p \mapsto F_p(f_p)$ is the pointwise minimum of the straight lines $q \mapsto F_q(f)$, hence strictly concave. This makes $S(f_p)$, as the value at $q = 1$ of the tangent, $q \mapsto F_q(f_p)$, to F at p, strictly decrease as p increases (and its value at $q = 0$, i.e., $T(f_p)$, strictly increase).

3. Study the rounding error effects in the smoothing spline calculations by rerunning the calculations, but using PPVLLC instead of PPVALU and comparing the first derivative values obtained with those of Example (11).

4. Prove the Cauchy-Schwarz-Inequality (15). (Hint: If $\|g\| = \|h\| = 0$, then $0 \leq \|g \pm h\|^2$ implies $(g, h) = 0$. Otherwise, without loss, $\|h\| \neq 0$, and

then $0 \le \|g - \frac{\langle g,h\rangle}{\langle h,h\rangle} h\|^2$ does the job.)

5. One way to enforce interpolation to some of the data during least-squares approximation is to make the weights corresponding to those data points very large compared to the rest. Investigate the effects of a few large weights on the accuracy of least-squares construction as carried out in L2APPR and its subroutines.

6. For $\mathbf{x} \in \mathbb{R}^n$, let $\mathbf{x}^* := ((-)^i x_i)_1^n$. Prove that $S^-\mathbf{x} + S^+(\mathbf{x}^*) = n - 1$.

7. Approximate the Titanium Heat data by (i) parabolic and cubic splines with simple knots (perhaps using the optimal knots for that order and for the 12 data points used in Example XIII(29)) and (ii) by $C^{(1)}$-quartics (that is, all interior knots are triple; this would require some editing of L2KNTS), and compare with Example (29) and Example XIII(29).

8. (*A test for noise*). According to M.J.D. Powell (see Hayes [1970: Chapter 5]), one expects the inequality

$$\sum_{i=p+1}^{q} r_{i-1} r_i \le \sum_{i=p}^{q} r_i^2 \frac{\sqrt{q-p}}{q-p+1}$$

to hold in case the error $r_i := y_i - f(x_i)$, $i = p, \ldots, q$, in the approximation f to the given data points (x_i, y_i) is essentially noise.

Modify L2MAIN to include a termination criterion based on this test. Then try it, for example, on the data in Example (27) or (11).

9. Write a program that realizes the following variable knot least squares cubic spline approximation algorithm.

1 The interval $[a\,..\,b]$ containing all the abscissae of the given data points is subdivided initially into one or more subintervals; call their collection M. Also, there is a collection M^*, initially empty.

2 While M is not empty, carry out the following two steps:

2.1 Pick some $I \in M$, and calculate a discrete least-squares approximation $p_I \in \Pi_{<4}$ to the data given in I.

2.2 If

$$\sum_{x_{i-1}, x_i \in I} r_{i-1} r_i \le \sum_{x_i \in I} r_i^2 / \sqrt{n_I}$$

(with $r_i := y_i - p_I(x_i)$, and $n_I :=$ number of x_i in I), then remove I from M and put it into M^*. Otherwise, cut I in two and put both pieces back into M.

3 At this point (is it certain that this point will always be reached?), M^* contains a partition of the original interval. If $\boldsymbol{\xi}$ is the break sequence giving this partition, then we are now certain that the data can be approximated well from $\Pi_{<4,\boldsymbol{\xi}}$. But, since we want a smooth approximated, we now "pull apart knots", that is, we approximate the data from

$\$_{4,\xi}$, with $\mathbf{t} := (a, a, a, a, \xi_{1.75}, \xi_{1.75}, \xi_{2.25}, \xi_{2.25}, \xi_{2.75}, \xi_{2.75}, \ldots, \xi_{l-.25}, \xi_{l-.25}, \xi_{l+.25}, \xi_{l+.25}, b, b, b, b)$, and $\xi_{i+\alpha} := (1-\alpha)\xi_i + \alpha\xi_{i+1}$, all $\alpha \in [0\,..\,1]$.

For "one-pass algorithms" that establish appropriate breaks by going through the data from left to right, see, for example, K. Ichida, F. Yoshimoto, and T. Kiyoni [1977].

10. The standard treatment of the smoothing spline concerns the minimization, over f, of the expression $G_\rho(f) := S(f) + \rho T(f)$, with $T(f) := \int (f^{(m)}(x))^2\,dx$, and with the *smoothing parameter* ρ chosen in $[0^+\,..\,\infty]$. Establish the map $p \mapsto \rho(p)$ for which f_p is the minimum for $G_{\rho(p)}$.

11. A useful extension of the smoothing spline construct can be obtained by considering the weighted roughness measure

$$T_\lambda(f) := \int_{x_1}^{x_n} \lambda(t)(D^m f(t))^2\,dt,$$

with λ some nonnegative function. It can be shown that, for $T = T_\lambda$, $\lambda D^m f_p$ (rather than $D^m f_p$) is in $\$^{\mathbf{nat}}_{2m,\mathbf{x}}$.

(a) Show that such f_p is piecewise polynomial in case λ is the reciprocal of a pp function.

(b) Work out the changes needed in SMOOTH in order to handle the special case when λ is piecewise constant, with breaks only at the x_i.

(See Kulkarni and Laurent [1991] and Bos and Salkauskas [1992] for good discussions of such **weighted splines.)**

XV

The Numerical Solution of an Ordinary Differential Equation by Collocation;

BSPLVD, COLLOC

The entire chapter is, in effect, just one example. We show how to use B-splines in the numerical solution of an ordinary differential equation by collocation. In the process, we construct one more basic routine, this one for the efficient simultaneous evaluation of all the B-splines *and* some of their derivatives at a site.

Until the late 1960's, the area called "numerical solutions of differential equations" was thought to be completely disjoint from the area called "approximation theory". But an outburst of mathematical and computational work on Galerkin's method and other projection methods changed all that. This outburst was caused by the realization that old standby techniques like the Rayleigh-Ritz method or Galerkin's method could be made quite effective if one were to give up on using polynomials or other analytic functions as trial functions and used *piecewise* polynomials instead. In a way, finite difference methods had derived their earlier superiority from the same source, that is, by being based on piecewise, or *local*, polynomial approximation.

There is no hope, in a book such as this, to give a fair description of the computational aspects of the **finite element method**, as this conjunction of projection methods and piecewise polynomial trial functions has come to be called. Instead, I want to discuss the numerical solution of an ODE from an approximation theoretic point of view, that is, as an interpolation process, not unlike the interpolation processes described in the preceding three chapters.

Mathematical background The mathematical background for this chapter, as found in de Boor & Swartz [1973], is as follows.

We intend to approximate a function g on $[a\,..\,b]$ that is given to us only *implicitly*, as a solution of the differential equation

$$(1) \qquad (D^m g)(x) = F(x; g(x), \ldots, (D^{m-1}g)(x)), \qquad \text{for } x \in [a\,..\,b]$$

with side conditions (for example, boundary conditions)

$$\beta_i g = c_i, \qquad i = 1, \dots, m. \tag{2}$$

Here, $F = F(x; z_0, \dots, z_{m-1})$ is a real-valued function on $\mathbb{R}^{m+1}$, and we will assume it to be "sufficiently smooth". Further, the side conditions are of the form

$$\sum_{j=1}^{m} w_{ij}(D^{j-1}g)(x_i) =: \beta_i g = c_i, \qquad i = 1, \dots, m, \tag{3}$$

for some constants w_{ij} and some sites $a \le x_1 \le \cdots \le x_m \le b$.

These side conditions are linear. Since we are eventually going to linearize (1) in the computations, we could have made these conditions nonlinear as well (see Wittenbrink [1973] and Problem 3).

Since (1) is nonlinear, in general, it may happen that (1)–(2) has many solutions. All we require in that case is that there be a neighborhood around the specific solution g we have in mind that contains no other solution. Our hope is to start our iterative scheme (necessary since (1) is nonlinear) within this neighborhood and then have it converge to this particular solution.

We attempt to approximate g by pp functions, using collocation. This means that we determine a pp function f so that it exactly satisfies the differential equation at certain sites, the collocation sites. In a way, these are sites of interpolation, but, in contrast to interpolation as discussed earlier, we do not match function values or derivative values at such sites but, rather, certain combinations of function and derivative values.

Specifically, with $\boldsymbol{\xi} = (\xi_i)_1^{l+1}$ given breaks (with $\xi_1 = a$, $\xi_{l+1} = b$, say), we look for $f \in \Pi_{<k+m,\boldsymbol{\xi}} \cap C^{(m-1)}$ for which

$$\begin{aligned} (D^m f)(\tau_i) &= F(\tau_i; f(\tau_i), \dots, (D^{m-1}f)(\tau_i)), \quad \text{for } i = 1, \dots, kl \\ \beta_i f &= c_i, \qquad i = 1, \dots, m. \end{aligned} \tag{4}$$

Here, we choose the **collocation sites** $(\tau_i)_1^{kl}$ k per subinterval, and distributed the same in each subinterval; that is, with

$$-1 \le \varrho_1 < \varrho_2 < \cdots < \varrho_k \le 1$$

picked somehow, we set

$$\tau_{(i-1)k+v} := [\xi_{i+1} + \xi_i + \varrho_v(\xi_{i+1} - \xi_i)]/2, \quad v = 1, \dots, k; \quad i = 1, \dots, l.$$

The program below leaves the choice of $\rho = (\varrho_i)_1^k$ to a subprogram COLPNT. The particular specimen of COLPNT given below picks ρ as the zeros of the kth Legendre polynomial, that is, as the sites that are used in the standard Gauss quadrature rule. The reason for this can be found in the following theorem.

(5) Theorem. *Assume that the function F in (1) is sufficiently smooth in a neighborhood of the curve*

$$[a\,..\,b] \to \mathbb{R}^{m+1} : x \mapsto (x, g(x), \ldots, D^{m-1}g(x)).$$

Assume further that the collocation pattern $\rho = (\varrho_i)_1^k$ in the standard interval $[-1\,..\,1]$ has been so chosen that

$$\int_{-1}^{1} q(x) \prod_{i=1}^{k} (x - \varrho_i)\,\mathrm{d}x = 0$$

for every $q \in \Pi_{<s}$. Then the solution f (if it exists) near g of the approximate problem (4) satisfies

$$(6) \qquad \|D^i g - D^i f\| \le \text{const}\ |\xi|^{k+\min(s,m-i)}, \qquad i = 0, \ldots, m.$$

At the breaks, the approximation is of even higher order, namely

$$(7) \quad |D^i(g-f)(\xi_j)| \le \text{const}\ |\xi|^{k+s}, \qquad j = 1, \ldots, l+1; \quad i = 0, \ldots, m-1.$$

Here, const *depends on F, g, k, but does not depend on ξ.*

Since $D^i f$ is pp of order $m + k - i$, the bound (6) shows that $D^i f$ approximates $D^i g$ to optimal order, according to Chapter XII, provided $s \ge m-i$. But the bound (7) shows that the approximation is of better than optimal order at the specific sites $\xi_1, \ldots, \xi_{l+1}$ (if $s > m$). This phenomenon is somewhat dramatically called **superconvergence**. To talk of better than optimal convergence may sound paradoxical, but isn't. Our statement that the order $\mathcal{O}\left(|\xi|^{m+k}\right)$ is best possible for $\operatorname{dist}(g, \Pi_{<m+k,\xi})$ refers to the overall size or norm of the error. This does not deny the possibility that we obtain a much better approximation at certain sites. After all, we can obtain *zero* error at certain sites simply by interpolating at these sites.

Note that the choice of ρ as the zeros of the kth degree Legendre polynomial allows us to take $s = k$ in the theorem. Thus at the breaks, we obtain

$$(8) \qquad |D^i(g-f)(\xi_i)| = o\left(|\xi|^{2k}\right) \qquad \text{for } j = 1, \ldots, l+1.$$

Since the overall approximation is then only of order $\mathcal{O}\left(|\xi|^{m+k}\right)$, one might as well forget about the computed approximation and construct instead a new overall approximation by some local interpolation to f from $\Pi_{<2k}$ matching f m-fold at nearby ξ_i's.

The approximate problem (4) is nonlinear (in general) and therefore requires some iterative scheme for its solution. It is shown in de Boor & Swartz [1973] that (4) can be solved by Newton's method starting with a

sufficiently close initial guess f_0 (provided F is sufficiently smooth and $|\xi|$ is sufficiently small). This means that (4) has a solution

$$f = \lim_{r\to\infty} f_r,$$

with f_{r+1} the solution $y \in \Pi_{<k+m,\xi} \cap C^{(m-1)}$ of the *linear* problem

$$(D^m y)(\tau_i) + \sum_{j<m} v_j(\tau_i)(D^j y)(\tau_i) = h(\tau_i), \quad i = 1, \ldots, kl,$$
$$\beta_i y = c_i, \qquad i = 1, \ldots, m, \tag{9}$$

where

$$v_j(x) := \left(\frac{\partial F}{\partial z_j}\right)(x; f_r(x), \ldots, (D^{m-1} f_r)(x)), \quad j = 0, \ldots, m-1, \tag{10}$$

and

$$h(x) := F(x; f_r(x), \ldots, (D^{m-1} f_r)(x)) + \sum_{j<m} v_j(x)(D^j f_r)(x). \tag{11}$$

The almost block diagonal character of the system of collocation equations; EQBLOK, PUTIT We express the unknown function y in (9) as a linear combination of appropriate B-splines. Explicitly, let $\mathbf{t} := (t_i)_1^{n+k+m}$ be the nondecreasing sequence (generated in the subprogram KNOTS below from ξ, m, and k) that contains each of ξ_1 and ξ_{l+1} $k+m$ times, and each interior break $\xi_2, \ldots, \xi_l$ k times. Then, $n = kl + m$, and

$$\Pi_{<k+m,\xi} \cap C^{(m-1)} = \$_{k+m,\mathbf{t}} \qquad (\text{on } [\xi_1 \,..\, \xi_{l+1}]).$$

The solution y of (9) can therefore be written in the form

$$y = \sum_{j=1}^{n} \alpha_j B_{j,k+m,\mathbf{t}},$$

that is, we can determine y by determining its B-coefficient vector α. This gives the linear system

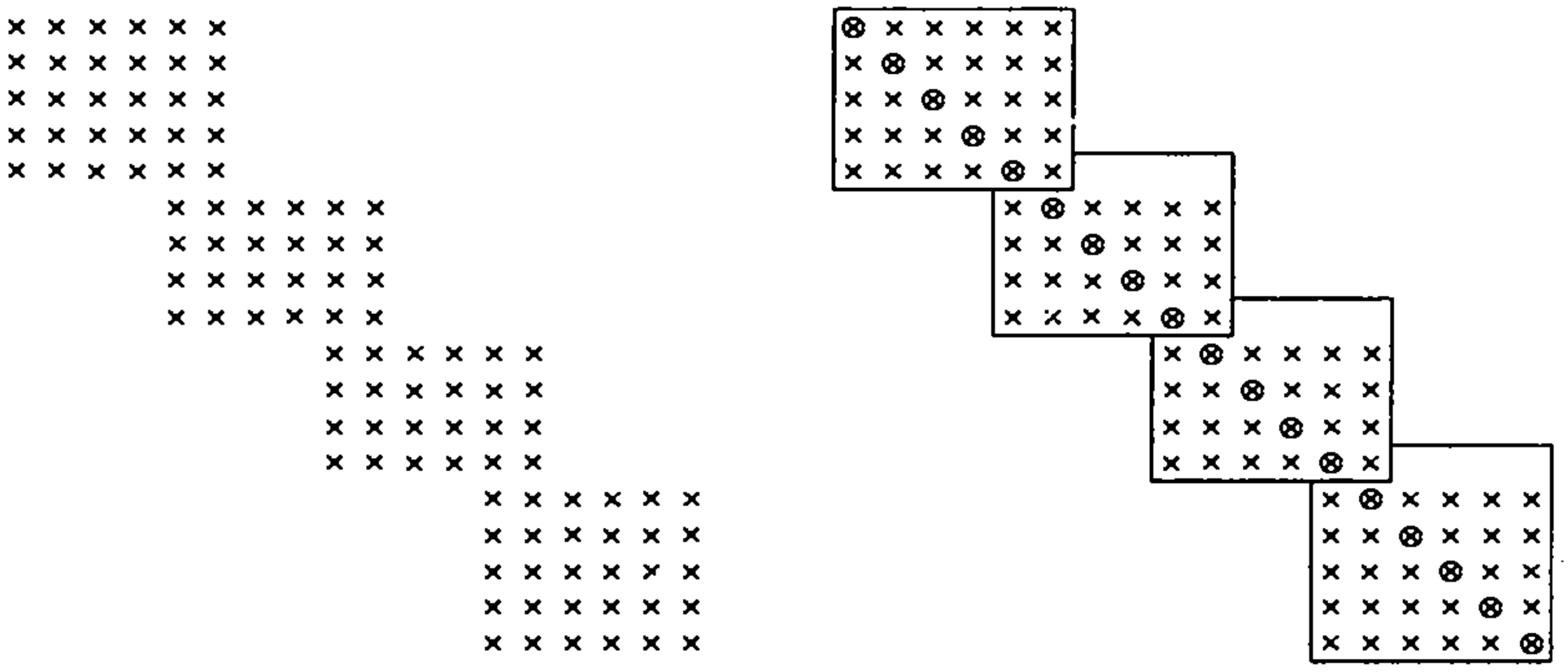
$$\begin{aligned} \textstyle\sum_{j=1}^{n} (LB_j)(\tau_i)\alpha_j &= h(\tau_i), \quad i = 1, \ldots, kl, \\ \textstyle\sum_{j=1}^{n} (\beta_i B_j)\alpha_j &= c_i \quad , \quad i = 1, \ldots, m, \end{aligned} \tag{12}$$

with the abbreviation

$$Ly := D^m y + \sum_{j<m} v_j\, D^j y$$

and v_j given explicitly by (10), h by (11), and β_i by (3).

If the side conditions are appropriately distributed among the collocation equations, then (12) is an *almost block diagonal* system. For instance, if we are dealing with a second-order two-point boundary-value problem and approximate by pp functions of order 6 with 4 pieces, then, indicating only the possibly nonzero entries, the (matrix of the) linear system (12) has the form given in the left half of Figure (13).



(13) FIGURE. Matrix for collocation equations for $(m, l, k) = (2, 4, 4)$ (left), and in its almost block diagonal form (right).

The ith block corresponds to the ith polynomial piece, that is, to the interval $(\xi_i \,..\, \xi_{i+1})$. The columns of the ith block correspond to the $k + m$ B-splines that have this interval in their support, and the rows of the ith block are made up of the k collocation equations at sites in the interval $(\xi_i \,..\, \xi_{i+1})$, together with the (properly interspersed) side conditions (if any) at sites in this interval.

Consider now Gauss elimination with partial pivoting applied to the linear system (12). After k elimination steps in the first block, an interchange might bring in an equation from the next block. We therefore take all equations not yet used as pivot equation (in our example above, there will be just one such equation) and adjoin them to the second block. This makes sense since these equations now have nonzero entries only in the columns that belong to the second block.

We then continue the elimination process. The next k steps will take place entirely within the second block. After that, an interchange might bring in an equation from the next block. So we take all equations in the second block not yet used as pivot equation (in our example above, there will be again just one such equation) and adjoin them to the third block, etc.

We see that the total space required for each block is found by taking

the block formed by the $k+m$ columns and the rows associated with an interval and extending this block toward the top until its left corner element is an element of the (main) diagonal of the whole matrix. This is shown in the right half of Figure (13) for the above example. The additional rows at the top of each block need not be set initially, they are filled during the elimination process.

In an Appendix, we give a listing of a package of programs, called SOLVE-BLOK, for the efficient solution of such almost block diagonal linear systems. Here, we now give the subprogram EQBLOK and its subroutine PUTIT, for putting together blocks and right sides for the linear system (12). The details about the specific differential equation to be solved are left here to a subroutine DIFEQU, an example of which is given below in Example (15).

```
      SUBROUTINE EQBLOK ( T, N, KPM,  WORK1, WORK2,
     *                    BLOKS, LENBLK, INTEGS, NBLOKS,  B )
CALLS PUTIT(DIFEQU,BSPLVD(BSPLVB))
C  TO BE CALLED IN  C O L L O C
C
C******  I N P U T  ******
C  T    THE KNOT SEQUENCE, OF LENGTH N+KPM
C  N    THE DIMENSION OF THE APPROXIMATING SPLINE SPACE, I.E., THE ORDER
C       OF THE LINEAR SYSTEM TO BE CONSTRUCTED.
C  KPM = K+M, THE ORDER OF THE APPROXIMATING SPLINE
C  LENBLK   THE MAXIMUM LENGTH OF THE ARRAY  BLOKS  AS ALLOWED BY THE
C           DIMENSION STATEMENT IN  COLLOC .
C
C******  W O R K   A R E A S  ******
C  WORK1    USED IN  PUTIT, OF SIZE (KPM,KPM)
C  WORK2    USED IN  PUTIT, OF SIZE (KPM,M+1)
C
C******  O U T P U T  ******
C  BLOKS    THE COEFFICIENT MATRIX OF THE LINEAR SYSTEM, STORED IN AL-
C           MOST BLOCK DIAGONAL FORM, OF SIZE
C              KPM*SUM(INTEGS(1,I) , I=1,...,NBLOKS)
C  INTEGS   AN INTEGER ARRAY, OF SIZE (3,NBLOKS), DESCRIBING THE BLOCK
C           STRUCTURE.
C           INTEGS(1,I)  =  NUMBER OF ROWS IN BLOCK  I
C           INTEGS(2,I)  =  NUMBER OF COLUMNS IN BLOCK  I
C           INTEGS(3,I)  =  NUMBER OF ELIMINATION STEPS WHICH CAN BE
C                        CARRIED OUT IN BLOCK  I  BEFORE PIVOTING MIGHT
C                        BRING IN AN EQUATION FROM THE NEXT BLOCK.
C  NBLOKS   NUMBER OF BLOCKS, EQUALS NUMBER OF POLYNOMIAL PIECES
C  B   THE RIGHT SIDE OF THE LINEAR SYSTEM, STORED CORRESPONDING TO THE
C      ALMOST BLOCK DIAGONAL FORM, OF SIZE SUM(INTEGS(1,I) , I=1,...,
C      NBLOKS).
C
C
```

```
C******  M E T H O D  ******
C  EACH BREAKPOINT INTERVAL GIVES RISE TO A BLOCK IN THE LINEAR SYSTEM.
C  THIS BLOCK IS DETERMINED BY THE  K  COLLOC.EQUATIONS IN THE INTERVAL
C  WITH THE SIDE CONDITIONS (IF ANY) IN THE INTERVAL INTERSPERSED AP-
C  PROPRIATELY, AND INVOLVES THE  KPM  B-SPLINES HAVING THE INTERVAL IN
C  THEIR SUPPORT. CORRESPONDINGLY, SUCH A BLOCK HAS  NROW = K + ISIDEL
C  ROWS, WITH  ISIDEL = NUMBER OF SIDE CONDITIONS IN THIS AND THE PREV-
C  IOUS INTERVALS, AND  NCOL = KPM  COLUMNS.
C     FURTHER, BECAUSE THE INTERIOR KNOTS HAVE MULTIPLICITY  K, WE CAN
C  CARRY OUT (IN SLVBLK)  K  ELIMINATION STEPS IN A BLOCK BEFORE PIVOT-
C  ING MIGHT INVOLVE AN EQUATION FROM THE NEXT BLOCK. IN THE LAST BLOCK,
C  OF COURSE, ALL KPM ELIMINATION STEPS WILL BE CARRIED OUT (IN SLVBLK).
C
C  SEE THE DETAILED COMMENTS IN THE SOLVEBLOK PACKAGE FOR FURTHER IN-
C  FORMATION ABOUT THE ALMOST BLOCK DIAGONAL FORM USED HERE.
      INTEGER INTEGS(3,1),KPM,LENBLK,N,NBLOKS,   I,INDEX,INDEXB,ISIDE
     *                                    ,ISIDEL,ITERMX,K,LEFT,M,NROW
      REAL B(1),BLOKS(1),T(N+K),WORK1(1),WORK2(1),   RHO,XSIDE
      COMMON /SIDE/ M, ISIDE, XSIDE(10)
      COMMON /OTHER/ ITERMX,K,RHO(19)
      INDEX = 1
      INDEXB = 1
      I = 0
      ISIDE = 1
      DO 20 LEFT=KPM,N,K
         I = I+1
C        DETERMINE INTEGS(.,I)
         INTEGS(2,I) = KPM
         IF (LEFT .LT. N)                  GO TO 14
         INTEGS(3,I) = KPM
         ISIDEL = M
                                           GO TO 16
   14    INTEGS(3,I) = K
C        AT THIS POINT,  ISIDE-1  GIVES THE NUMBER OF SIDE CONDITIONS
C        INCORPORATED SO FAR. ADDING TO THIS THE SIDE CONDITIONS IN THE
C        CURRENT INTERVAL GIVES THE NUMBER  ISIDEL .
         ISIDEL = ISIDE-1
   15    IF (ISIDEL .EQ. M)                GO TO 16
         IF (XSIDE(ISIDEL+1) .GE. T(LEFT+1))
     *                                     GO TO 16
         ISIDEL = ISIDEL+1
                                           GO TO 15
   16    NROW = K + ISIDEL
         INTEGS(1,I) = NROW
C        THE DETAILED EQUATIONS FOR THIS BLOCK ARE GENERATED AND PUT
C        TOGETHER IN  P U T I T .
         IF (LENBLK .LT. INDEX+NROW*KPM-1)GO TO 999
         CALL PUTIT(T,KPM,LEFT,WORK1,WORK2,BLOKS(INDEX),NROW,B(INDEXB))
         INDEX = INDEX + NROW*KPM
   20    INDEXB = INDEXB + NROW
      NBLOKS = I
                                           RETURN
  999 PRINT 699,LENBLK
  699 FORMAT(11H **********/23H THE ASSIGNED DIMENSION,I5
     *       ,38H FOR  BLOKS  IN  COLLOC  IS TOO SMALL.)
                                           STOP
      END
```

```
      SUBROUTINE PUTIT ( T, KPM, LEFT, SCRTCH, DBIATX, Q, NROW, B )
CALLS  BSPLVD(BSPLVB),DIFEQU(*)
C  TO BE CALLED BY  E Q B L O K .
C
C  PUTS TOGETHER ONE BLOCK OF THE COLLOCATION EQUATION SYSTEM
C
C******  I N P U T  ******
C  T     KNOT SEQUENCE, OF SIZE  LEFT+KPM (AT LEAST)
C  KPM   ORDER OF SPLINE
C  LEFT  INTEGER INDICATING INTERVAL OF INTEREST, VIZ THE INTERVAL
C           (T(LEFT)..T(LEFT+1))
C  NROW  NUMBER OF ROWS IN BLOCK TO BE PUT TOGETHER
C
C******  W O R K  A R E A  ******
C  SCRTCH   USED IN BSPLVD, OF SIZE (KPM,KPM)
C  DBIATX   USED TO CONTAIN DERIVATIVES OF B-SPLINES, OF SIZE (KPM,M+1)
C           WITH DBIATX(J,I+1) CONTAINING THE I-TH DERIVATIVE OF THE
C           J-TH B-SPLINE OF INTEREST
C
C******  O U T P U T  ******
C  Q  THE BLOCK, OF SIZE (NROW,KPM)
C  B  THE CORRESPONDING PIECE OF THE RIGHT SIDE, OF SIZE (NROW)
C
C******  M E T H O D  ******
C  THE  K  COLLOCATION EQUATIONS FOR THE INTERVAL  (T(LEFT)..T(LEFT+1))
C  ARE CONSTRUCTED WITH THE AID OF THE SUBROUTINE  D I F E Q U ( 2, .,
C  . ) AND INTERSPERSED (IN ORDER) WITH THE SIDE CONDITIONS (IF ANY) IN
C  THIS INTERVAL, USING  D I F E Q U ( 3, ., . )  FOR THE INFORMATION.
C     THE BLOCK  Q  HAS  KPM  COLUMNS, CORRESPONDING TO THE  KPM  B-
C  SPLINES OF ORDER  KPM  WHICH HAVE THE INTERVAL (T(LEFT)..T(LEFT+1))
C  IN THEIR SUPPORT. THE BLOCK'S DIAGONAL IS PART OF THE DIAGONAL OF THE
C  TOTAL SYSTEM. THE FIRST EQUATION IN THIS BLOCK NOT OVERLAPPED BY THE
C  PRECEDING BLOCK IS THEREFORE EQUATION  L O W R O W , WITH LOWROW = 1+
C  NUMBER OF SIDE CONDITIONS IN PRECEDING INTERVALS (OR BLOCKS).
C
      INTEGER KPM,LEFT,NROW,    I,IROW,ISIDE,ITERMX,J,K,LL,LOWROW,M,MODE
     *                         ,MP1
      REAL B(1),DBIATX(KPM,1),Q(NROW,KPM),SCRTCH(1),T(1),    DX,RHO,SUM
     *                                                  ,V(20),XM,XSIDE,XX
      COMMON /SIDE/ M, ISIDE, XSIDE(10)
      COMMON /OTHER/ ITERMX,K,RHO(19)
      MP1 = M+1
      DO 10 J=1,KPM
         DO 10 I=1,NROW
10          Q(I,J) = 0.
      XM = (T(LEFT+1)+T(LEFT))/2.
      DX = (T(LEFT+1)-T(LEFT))/2.
C
      LL = 1
      LOWROW = ISIDE
      DO 30 IROW=LOWROW,NROW
         IF (LL .GT. K)                    GO TO 22
         MODE = 2
C        NEXT COLLOCATION POINT IS ...
         XX = XM + DX*RHO(LL)
         LL = LL + 1
C        THE CORRESP.COLLOCATION EQUATION IS NEXT UNLESS THE NEXT SIDE
CC       CONDITION OCCURS AT A POINT AT, OR TO THE LEFT OF, THE NEXT
C        COLLOCATION POINT.
         IF (ISIDE .GT. M)                 GO TO 24
         IF (XSIDE(ISIDE) .GT. XX)         GO TO 24
         LL = LL - 1
22       MODE = 3
         XX = XSIDE(ISIDE)
24       CALL DIFEQU ( MODE, XX, V )
```

```
C         THE NEXT EQUATION, A COLLOCATION EQUATION (MODE = 2) OR A SIDE
C         CONDITION (MODE = 3), READS
C    (*)   (V(M+1)*D**M + V(M)*D**(M-1) +...+ V(1)*D**0)F(XX) = V(M+2)
C         IN TERMS OF THE INFO SUPPLIED BY  DIFEQU . THE CORRESPONDING
C         EQUATION FOR THE B-COEFFS OF  F  THEREFORE HAS THE LEFT SIDE OF
C         (*), EVALUATED AT EACH OF THE  KPM  B-SPLINES HAVING  XX  IN
C         THEIR SUPPORT, AS ITS  KPM  POSSIBLY NONZERO COEFFICIENTS.
          CALL BSPLVD ( T, KPM, XX, LEFT, SCRTCH, DBIATX, MP1 )
          DO 26 J=1,KPM
             SUM = 0.
             DO 25 I=1,MP1
25              SUM = V(I)*DBIATX(J,I) + SUM
26           Q(IROW,J) = SUM
30        B(IROW) = V(M+2)
                                                    RETURN
       END
```

The subroutine BSPLVD Construction of the linear system (12) for the B-coefficients of a Newton iterate requires knowledge of the values and the first m derivatives of all the B-splines on t of order $k+m$ at all the collocation sites as well as at all side condition sites. The subroutine PUTIT above relies for this information on a subroutine BSPLVD that we now discuss.

We already know how to generate the values of the k B-splines of order k at a given site in the (closed) support of these B-splines. There are two possible ways (at least) for generating their derivatives as well. One way is to work with the recurrence relation for the derivatives of a B-spline, as has been advocated by Cox (see ref. 2 of Butterfield [1976]). One can obtain these recurrence relations by substituting a j for the k in each of the exponents in IX(17), that is, by applying Leibniz' formula I(iv) for the kth divided difference of a product to the particular product

$$(t-x)_+^{j-1} = (t-x)(t-x)_+^{j-2}$$

(see Problem X.5). Since

$$\frac{D^j \mathrm{B}_{i,k}(x)}{(t_{i+k}-t_i)} = (k-1)\cdots(k-j)(-)^j[t_i,\ldots,t_{i+k}](\cdot-x)_+^{k-j-1},$$

we then obtain that

$$\frac{k-j-1}{k-1} D^j \mathrm{B}_{i,k}(x) = \frac{x-t_i}{t_{i+k-1}-t_i} D^j \mathrm{B}_{i,k-1}(x) + \frac{t_{i+k}-x}{t_{i+k}-t_{i+1}} D^j \mathrm{B}_{i+1,k-1}(x). \tag{14}$$

Another way is to consider a B-spline to be a spline with a particularly

simple B-coefficient sequence and use X(15) to express the jth derivative of the spline as a linear combination of B-splines of order $k-j$. We have chosen this latter approach in the following subroutine BSPLVD. Support for this choice can be found in a rounding error analysis of these two (and other) alternatives as carried out by Butterfield [1976].

```
      SUBROUTINE BSPLVD ( T, K, X, LEFT, A, DBIATX, NDERIV )
CALLS BSPLVB
CALCULATES VALUE AND DERIV.S OF ALL B-SPLINES WHICH DO NOT VANISH AT X
C
C******  I N P U T  ******
C  T      THE KNOT ARRAY, OF LENGTH LEFT+K (AT LEAST)
C  K      THE ORDER OF THE B-SPLINES TO BE EVALUATED
C  X      THE POINT AT WHICH THESE VALUES ARE SOUGHT
C  LEFT   AN INTEGER INDICATING THE LEFT ENDPOINT OF THE INTERVAL OF
C         INTEREST. THE  K  B-SPLINES WHOSE SUPPORT CONTAINS THE INTERVAL
C                (T(LEFT)..T(LEFT+1))
C         ARE TO BE CONSIDERED.
C  A S S U M P T I O N  - - -  IT IS ASSUMED THAT
C                T(LEFT) .LT. T(LEFT+1)
C         DIVISION BY ZERO WILL RESULT OTHERWISE (IN  B S P L V B ).
C         ALSO, THE OUTPUT IS AS ADVERTISED ONLY IF
C                T(LEFT) .LE. X .LE. T(LEFT+1) .
C  NDERIV   AN INTEGER INDICATING THAT VALUES OF B-SPLINES AND THEIR
C         DERIVATIVES UP TO BUT NOT INCLUDING THE  NDERIV-TH  ARE ASKED
C         FOR. ( NDERIV  IS REPLACED INTERNALLY BY THE INTEGER  M H I G H
C         IN  (1,K)  CLOSEST TO IT.)
C
C******  W O R K   A R E A  ******
C  A      AN ARRAY OF ORDER (K,K), TO CONTAIN B-COEFF.S OF THE DERIVAT-
C         IVES OF A CERTAIN ORDER OF THE  K  B-SPLINES OF INTEREST.
C
C******  O U T P U T  ******
C  DBIATX   AN ARRAY OF ORDER (K,NDERIV). ITS ENTRY  (I,M)  CONTAINS
C         VALUE OF  (M-1)ST  DERIVATIVE OF  (LEFT-K+I)-TH  B-SPLINE OF
C         ORDER  K  FOR KNOT SEQUENCE  T , I=1,...,K, M=1,...,NDERIV.
C
C******  M E T H O D  ******
C  VALUES AT  X  OF ALL THE RELEVANT B-SPLINES OF ORDER K,K-1,...,
C  K+1-NDERIV  ARE GENERATED VIA  BSPLVB  AND STORED TEMPORARILY IN
C  DBIATX .  THEN, THE B-COEFFS OF THE REQUIRED DERIVATIVES OF THE B-
C  SPLINES OF INTEREST ARE GENERATED BY DIFFERENCING, EACH FROM THE PRE-
C  CEDING ONE OF LOWER ORDER, AND COMBINED WITH THE VALUES OF B-SPLINES
C  OF CORRESPONDING ORDER IN  DBIATX  TO PRODUCE THE DESIRED VALUES .
C
      INTEGER K,LEFT,NDERIV,   I,IDERIV,IL,J,JLOW,JP1MID,KP1,KP1MM
     *                        ,LDUMMY,M,MHIGH
      REAL A(K,K),DBIATX(K,NDERIV),T(1),X,   FACTOR,FKP1MM,SUM
      MHIGH = MAXO(MINO(NDERIV,K),1)
C     MHIGH IS USUALLY EQUAL TO NDERIV.
      KP1 = K+1
      CALL BSPLVB(T,KP1-MHIGH,1,X,LEFT,DBIATX)
      IF (MHIGH .EQ. 1)                          GO TO 99
C     THE FIRST COLUMN OF  DBIATX  ALWAYS CONTAINS THE B-SPLINE VALUES
C     FOR THE CURRENT ORDER. THESE ARE STORED IN COLUMN K+1-CURRENT
C     ORDER  BEFORE  BSPLVB  IS CALLED TO PUT VALUES FOR THE NEXT
C     HIGHER ORDER ON TOP OF IT.
      IDERIV = MHIGH
```

```
      DO 15 M=2,MHIGH
         JP1MID = 1
         DO 11 J=IDERIV,K
            DBIATX(J,IDERIV) = DBIATX(JP1MID,1)
   11       JP1MID = JP1MID + 1
         IDERIV = IDERIV - 1
         CALL BSPLVB(T,KP1-IDERIV,2,X,LEFT,DBIATX)
   15    CONTINUE
C
C     AT THIS POINT,  B(LEFT-K+I, K+1-J)(X) IS IN  DBIATX(I,J) FOR
C     I=J,...,K AND J=1,...,MHIGH ('=' NDERIV). IN PARTICULAR, THE
C     FIRST COLUMN OF  DBIATX  IS ALREADY IN FINAL FORM. TO OBTAIN COR-
C     RESPONDING DERIVATIVES OF B-SPLINES IN SUBSEQUENT COLUMNS, GENE-
C     RATE THEIR B-REPR. BY DIFFERENCING, THEN EVALUATE AT  X.
C
      JLOW = 1
      DO 20 I=1,K
         DO 19 J=JLOW,K
   19       A(J,I) = 0.
         JLOW = I
   20    A(I,I) = 1.
C     AT THIS POINT, A(.,J) CONTAINS THE B-COEFFS FOR THE J-TH OF THE
C     K  B-SPLINES OF INTEREST HERE.
C
      DO 40 M=2,MHIGH
         KP1MM = KP1 - M
         FKP1MM = FLOAT(KP1MM)
         IL = LEFT
         I = K
C
C        FOR J=1,...,K, CONSTRUCT B-COEFFS OF  (M-1)ST  DERIVATIVE OF
C        B-SPLINES FROM THOSE FOR PRECEDING DERIVATIVE BY DIFFERENCING
C        AND STORE AGAIN IN  A(.,J) . THE FACT THAT  A(I,J) = 0  FOR
C        I .LT. J  IS USED.
         DO 25 LDUMMY=1,KP1MM
            FACTOR = FKP1MM/(T(IL+KP1MM) - T(IL))
C           THE ASSUMPTION THAT T(LEFT).LT.T(LEFT+1) MAKES DENOMINATOR
C           IN  FACTOR  NONZERO.
            DO 24 J=1,I
   24          A(I,J) = (A(I,J) - A(I-1,J))*FACTOR
            IL = IL - 1
   25       I = I - 1
C
C       FOR I=1,...,K, COMBINE B-COEFFS A(.,I) WITH B-SPLINE VALUES
C       STORED IN DBIATX(.,M) TO GET VALUE OF  (M-1)ST  DERIVATIVE OF
C       I-TH B-SPLINE (OF INTEREST HERE) AT  X , AND STORE IN
C       DBIATX(I,M). STORAGE OF THIS VALUE OVER THE VALUE OF A B-SPLINE
C       OF ORDER M THERE IS SAFE SINCE THE REMAINING B-SPLINE DERIVAT-
C       IVES OF THE SAME ORDER DO NOT USE THIS VALUE DUE TO THE FACT
C       THAT  A(J,I) = 0  FOR J .LT. I .
        DO 40 I=1,K
           SUM = 0.
           JLOW = MAXO(I,M)
           DO 35 J=JLOW,K
   35         SUM = A(J,I)*DBIATX(J,M) + SUM
   40      DBIATX(I,M) = SUM
   99                                    RETURN
      END
```

COLLOC and its subroutines The following subroutine COLLOC oversees the iterative solution of (4) by Newton's method, that is, via (9). It

relies on KNOTS to convert break sequences to knot sequences and uses COLPNT to supply the Gauss-Legendre sites as collocation sites ρ for the standard interval. It uses EQBLOK with PUTIT to put together the linear system (12), and uses the package SOLVEBLOK, listed in an Appendix, to solve (12). It also calls on NEWNOT (of Chapter XII) for possible improvement of the chosen break sequence. A subroutine DIFEQU must be supplied for each specific ODE to be solved.

```
      SUBROUTINE COLLOC(ALEFT,ARIGHT,LBEGIN,IORDER,NTIMES,ADDBRK,RELERR)
CHAPTER XV, EXAMPLE. SOLUTION OF AN ODE BY COLLOCATION.
CALLS COLPNT, DIFEQU(PPVALU(INTERV)), KNOTS, EQBLOK(PUTIT(DIFEQU*,
C     BSPLVD(BSPLVB)))), SLVBLK(VARIOUS SUBPROGRAMS), BSPLPP(BSPLVB*),
C     NEWNOT
C
C****** I N P U T ******
C  ALEFT, ARIGHT  ENDPOINTS OF INTERVAL OF APPROXIMATION
C  LBEGIN   INITIAL NUMBER OF POLYNOMIAL PIECES IN THE APPROXIMATION.
C           A UNIFORM BREAKPOINT SEQUENCE IS CHOSEN.
C  IORDER   ORDER OF POLYNOMIAL PIECES IN THE APPROXIMATION
C  NTIMES   NUMBER OF PASSES THROUGH  N E W N O T  TO BE MADE
C  ADDBRK   THE NUMBER (POSSIBLY FRACTIONAL) OF BREAKS TO BE ADDED PER
C           PASS THROUGH NEWNOT. E.G., IF ADDBRK = .33334, THEN A BREAK-
C           POINT WILL BE ADDED AT EVERY THIRD PASS THROUGH NEWNOT.
C  RELERR   A TOLERANCE. NEWTON ITERATION IS STOPPED IF THE DIFFERENCE
C           BETWEEN THE B-COEFFS OF TWO SUCCESSIVE ITERATES IS NO MORE
C           THAN  RELERR*(ABSOL.LARGEST B-COEFFICIENT).
C
C****** P R I N T E D  O U T P U T ******
C     CONSISTS OF THE PP-REPRESENTATION OF THE APPROXIMATE SOLUTION,
C     AND OF THE ERROR AT SELECTED POINTS.
C
C****** M E T H O D ******
C  THE M-TH ORDER ORDINARY DIFFERENTIAL EQUATION WITH  M  SIDE CONDIT-
C  IONS, TO BE SPECIFIED IN SUBROUTINE  D I F E Q U , IS SOLVED APPROX-
C  IMATELY BY COLLOCATION.
C  THE APPROXIMATION  F  TO THE SOLUTION  G  IS PP OF ORDER  K+M  WITH
C  L  PIECES AND  M-1 CONTINUOUS DERIVATIVES.  F  IS DETERMINED BY THE
C  REQUIREMENT THAT IT SATISFY THE D.E. AT  K  POINTS PER INTERVAL (TO
C  BE SPECIFIED IN  C O L P N T ) AND THE  M  SIDE CONDITIONS.
C     THIS USUALLY NONLINEAR SYSTEM OF EQUATIONS FOR  F  IS SOLVED BY
C  NEWTON'S METHOD. THE RESULTING LINEAR SYSTEM FOR THE B-COEFFS OF AN
C  ITERATE IS CONSTRUCTED APPROPRIATELY IN  E Q B L O K  AND THEN SOLVED
C  IN  S L V B L K , A PROGRAM DESIGNED TO SOLVE  A L M O S T  B L O C K
C  D I A G O N A L  LINEAR SYSTEMS EFFICIENTLY.
C     THERE IS AN OPPORTUNITY TO ATTEMPT IMPROVEMENT OF THE BREAKPOINT
C  SEQUENCE (BOTH IN NUMBER AND LOCATION) THROUGH USE OF  N E W N O T .
C
      INTEGER NPIECE
      PARAMETER (NPIECE=100)
      INTEGER IORDER,LBEGIN,NTIMES,   I,IFLAG,II,INTEGS(3,NPIECE),ISIDE
     *                    ,ITER,ITERMX,K,KMAX,KPM,L,LENBLK,LNEW,M,N,NBLOKS
     *                    ,NDIM,NCOEF,NNCOEF,NPIECE,NT
      PARAMETER (NDIM=200,KMAX=20,NCOEF=NPIECE*KMAX,LENBLK=NCOEF)
      REAL ADDBRK,ALEFT,ARIGHT,RELERR,   A(NDIM),AMAX,ASAVE(NDIM)
     *     ,B(NDIM),BLOKS(LENBLK),BREAK,COEF,DX,ERR,RHO,T(NDIM)
     *     ,TEMPL(LENBLK),TEMPS(NDIM),XSIDE
      EQUIVALENCE (BLOKS,TEMPL)
      COMMON /APPROX/ BREAK(NPIECE), COEF(NCOEF), L,KPM
      COMMON /SIDE/ M, ISIDE, XSIDE(10)
      COMMON /OTHER/ ITERMX,K,RHO(KMAX-1)
C
```

```
      KPM = IORDER
      IF (LBEGIN*KPM .GT. NCOEF)        GO TO 999
C  *** SET THE VARIOUS PARAMETERS CONCERNING THE PARTICULAR DIF.EQU.
C     INCLUDING A FIRST APPROX. IN CASE THE DE IS TO BE SOLVED BY
C     ITERATION ( ITERMX .GT. 0) .
      CALL DIFEQU (1, TEMPS(1), TEMPS )
C  *** OBTAIN THE  K  COLLOCATION POINTS FOR THE STANDARD INTERVAL.
      K = KPM - M
      CALL COLPNT ( K, RHO )
C  *** THE FOLLOWING FIVE STATEMENTS COULD BE REPLACED BY A READ IN OR-
C     DER TO OBTAIN A SPECIFIC (NONUNIFORM) SPACING OF THE BREAKPNTS.
      DX = (ARIGHT - ALEFT)/FLOAT(LBEGIN)
      TEMPS(1) = ALEFT
      DO 4 I=2,LBEGIN
    4    TEMPS(I) = TEMPS(I-1) + DX
      TEMPS(LBEGIN+1) = ARIGHT
C  *** GENERATE, IN KNOTS, THE REQUIRED KNOTS T(1),...,T(N+KPM).
      CALL KNOTS ( TEMPS, LBEGIN, KPM, T, N )
      NT = 1
C  *** GENERATE THE ALMOST BLOCK DIAGONAL COEFFICIENT MATRIX  BLOKS  AND
C     RIGHT SIDE  B  FROM COLLOCATION EQUATIONS AND SIDE CONDITIONS.
C     THEN SOLVE VIA  SLVBLK , OBTAINING THE B-REPRESENTATION OF THE AP-
C     PROXIMATION IN  T , A ,  N  , KPM  .
   10    CALL EQBLOK(T,N,KPM,TEMPS,A,BLOKS,LENBLK,INTEGS,NBLOKS,B)
         CALL SLVBLK(BLOKS,INTEGS,NBLOKS,B,TEMPS,A,IFLAG)
         ITER = 1
         IF (ITERMX .LE. 1)                GO TO 30
C  *** SAVE B-SPLINE COEFF. OF CURRENT APPROX. IN  ASAVE , THEN GET NEW
C     APPROX. AND COMPARE WITH OLD. IF COEFF. ARE MORE THAN  RELERR
C     APART (RELATIVELY) OR IF NO. OF ITERATIONS IS LESS THAN  ITERMX ,
C     CONTINUE ITERATING.
   20       CALL BSPLPP(T,A,N,KPM,TEMPL,BREAK,COEF,L)
            DO 25 I=1,N
   25          ASAVE(I) = A(I)
            CALL EQBLOK(T,N,KPM,TEMPS,A,BLOKS,LENBLK,INTEGS,NBLOKS,B)
            CALL SLVBLK(BLOKS,INTEGS,NBLOKS,B,TEMPS,A,IFLAG)
            ERR = 0.
            AMAX = 0.
            DO 26 I=1,N
               AMAX = AMAX1(AMAX,ABS(A(I)))
   26          ERR = AMAX1(ERR,ABS(A(I)-ASAVE(I)))
            IF (ERR .LE. RELERR*AMAX)    GO TO 30
            ITER = ITER+1
            IF (ITER .LT. ITERMX)        GO TO 20
C  *** ITERATION (IF ANY) COMPLETED. PRINT OUT APPROX. BASED ON CURRENT
C     BREAKPOINT SEQUENCE, THEN TRY TO IMPROVE THE SEQUENCE.
   30    PRINT 630,KPM,L,N,(BREAK(I),I=2,L)
  630    FORMAT(47H APPROXIMATION FROM A SPACE OF SPLINES OF ORDER,I3
     *          ,4H ON ,I3,11H INTERVALS,/13H OF DIMENSION,I4
     *          ,16H.  BREAKPOINTS -/(5E20.10))
         IF (ITERMX .GT. 0)  PRINT 635,ITER,ITERMX
  635    FORMAT(6H AFTER,I3,3H OF,I3,20H ALLOWED ITERATIONS,)
         CALL BSPLPP(T,A,N,KPM,TEMPL,BREAK,COEF,L)
         PRINT 637
  637    FORMAT(46H THE PP REPRESENTATION OF THE APPROXIMATION IS)
         DO 38 I=1,L
            II = (I-1)*KPM
   38       PRINT 638, BREAK(I),(COEF(II+J),J=1,KPM)
  638    FORMAT(F9.3,E13.6,10E11.3)
```

```
C  *** THE FOLLOWING CALL IS PROVIDED HERE FOR POSSIBLE FURTHER ANALYSIS
C     OF THE APPROXIMATION SPECIFIC TO THE PROBLEM BEING SOLVED.
C     IT IS, OF COURSE, EASILY OMITTED.
         CALL DIFEQU ( 4, TEMPS(1), TEMPS )
C
         IF (NT .GT. NTIMES)                    RETURN
C  *** FROM THE PP-REP. OF THE CURRENT APPROX., OBTAIN IN  NEWNOT  A NEW
C     (AND POSSIBLY BETTER) SEQUENCE OF BREAKPOINTS, ADDING (ON THE AVE-
C     RAGE)  A D D B R K  BREAKPOINTS PER PASS THROUGH NEWNOT.
         LNEW = LBEGIN + IFIX(FLOAT(NT)*ADDBRK)
         IF (LNEW*KPM .GT. NCOEF)               GO TO 999
         CALL NEWNOT(BREAK,COEF,L,KPM,TEMPS,LNEW,TEMPL)
         CALL KNOTS(TEMPS,LNEW,KPM,T,N)
         NT = NT + 1
                                                GO TO 10
  999 NNCOEF = NCOEF
      PRINT 699,NNCOEF
  699 FORMAT(11H **********/23H THE ASSIGNED DIMENSION,I5
     *       ,25H FOR  COEF  IS TOO SMALL.)
                                                RETURN
      END



      SUBROUTINE KNOTS ( BREAK, L, KPM, T, N )
C  TO BE CALLED IN  C O L L O C .
C  CONSTRUCTS FROM THE GIVEN BREAKPOINT SEQUENCE  B R E A K  THE KNOT
C  SEQUENCE  T  SO THAT
C  SPLINE(K+M,T) = PP(K+M,BREAK) WITH  M-1  CONTINUOUS DERIVATIVES .
C  THIS MEANS THAT
C  T(1),...,T(N+KPM)  =  BREAK(1) KPM TIMES, THEN BREAK(2),...,
C        BREAK(L) EACH  K  TIMES, THEN, FINALLY, BREAK(L+1) KPM TIMES
C
C******  I N P U T  ******
C  BREAK(1),...,BREAK(L+1)  BREAKPOINT SEQUENCE
C  L  NUMBER OF INTERVALS OR PIECES
C  KPM   = K + M, ORDER OF THE PP FUNCTION OR SPLINE
C
C******  O U T P U T  ******
C  T(1),...,T(N+KPM)  THE KNOT SEQUENCE.
C  N     = L*K + M  =  DIMENSION OF  SPLINE(K+M,T).
C
      INTEGER L,KPM,N,   ISIDE,J,JJ,JJJ,K,LL,M
      REAL BREAK(1),T(1),   XSIDE
      COMMON /SIDE/ M,ISIDE,XSIDE(10)
      K = KPM - M
      N = L*K + M
      JJ = N + KPM
      JJJ = L + 1
      DO 11 LL=1,KPM
         T(JJ) = BREAK(JJJ)
   11    JJ = JJ - 1
      DO 12 J=1,L
      JJJ = JJJ - 1
         DO 12 LL=1,K
            T(JJ) = BREAK(JJJ)
   12       JJ = JJ - 1
      DO 13 LL=1,KPM
   13    T(LL) = BREAK(1)
                                                RETURN
      END
```

```
      SUBROUTINE COLPNT(K,RHO)
C  THE  K COLLOCATION POINTS FOR THE STANDARD INTERVAL (-1..1) ARE SUP-
C  PLIED HERE AS THE ZEROS OF THE LEGENDRE POLYNOMIAL OF DEGREE K ,
C  PROVIDED  K .LE. 8 . OTHERWISE, UNIFORMLY SPACED POINTS ARE GIVEN.
      INTEGER K,  J
      REAL RHO(K),  FKM102
      IF (K .GT. 8)                          GO TO 99
                                             GO TO (10,20,30,40,50,60,70,80),K
   10 RHO(1) = 0.
                                             RETURN
   20 RHO(2) = .57735 02691 89626 D0
      RHO(1) = - RHO(2)
                                             RETURN
   30 RHO(3) = .77459 66692 41483 D0
      RHO(1) = - RHO(3)
      RHO(2) = 0.
                                             RETURN
   40 RHO(3) = .33998 10435 84856 D0
      RHO(2) = - RHO(3)
      RHO(4) = .86113 63115 94053 D0
      RHO(1) = - RHO(4)
                                             RETURN
   50 RHO(4) = .53846 93101 05683 D0
      RHO(2) = - RHO(4)
      RHO(5) = .90617 98459 38664 D0
      RHO(1) = - RHO(5)
      RHO(3) = 0.
                                             RETURN
   60 RHO(4) = .23861 91860 83197 D0
      RHO(3) = - RHO(4)
      RHO(5) = .66120 93864 66265 D0
      RHO(2) = - RHO(5)
      RHO(6) = .93246 95142 03152 D0
      RHO(1) = - RHO(6)
                                             RETURN
   70 RHO(5) = .40584 51513 77397 D0
      RHO(3) = - RHO(5)
      RHO(6) = .74153 11855 99394 D0
      RHO(2) = - RHO(6)
      RHO(7) = .94910 79123 42759 D0
      RHO(1) = - RHO(7)
      RHO(4) = 0.
                                             RETURN
   80 RHO(5) = .18343 46424 95650 D0
      RHO(4) = - RHO(5)
      RHO(6) = .52553 24099 16329 D0
      RHO(3) = - RHO(6)
      RHO(7) = .79666 64774 13627 D0
      RHO(2) = - RHO(7)
      RHO(8) = .96028 98564 97536 D0
      RHO(1) = - RHO(8)
                                             RETURN
C  IF K .GT. 8, USE EQUISPACED POINTS, BUT PRINT WARNING
   99 PRINT 699,K
  699 FORMAT(11H **********/
     *       49H EQUISPACED COLLOCATION POINTS ARE USED SINCE K =,I2,
     *       19H IS GREATER THAN 8.)
      FKM102 = FLOAT(K-1)/2.
      DO 100 J=1,K
  100    RHO(J) = FLOAT(J-1)/FKM102 - 1.
                                                RETURN
      END
```

(15) Example: A second order nonlinear two-point boundary-value problem with a boundary layer This example is taken from Dodson [1972] who took it from Carrier [1970]. The function to be approximated is specified by

$$\varepsilon g''(x) + [g(x)]^2 = 1 \quad \text{on} \quad [0\,..\,1]$$
$$g'(0) = g(1) = 0.$$

For small ε, the solution g is given approximately by

$$g(x) \approx 12[\varepsilon_+(x)/(1+\varepsilon_+(x))^2 + \varepsilon_-(x)/(1+\varepsilon_-(x))^2] - 1$$

with

$$\varepsilon_\pm(x) := (\sqrt{2}+\sqrt{3})^2 \exp\left((1 \pm x)\sqrt{2/\varepsilon}\right)$$

which cannot be evaluated on a UNIVAC 1110 in single precision (roughly 8 decimal places) without underflow in case ε is much smaller than $10^{-2}/2$.

The linear problem (9) for the determination of $y = f_{r+1}$ from the preceding iterate f_r becomes

$$\varepsilon y''(\tau_i) + v_0(\tau_i)y(\tau_i) = h(\tau_i), \qquad i = 1, \ldots, kl,$$
$$y'(0) = y(1) = 0,$$

with

$$v_0(x) = 2f_r(x), \qquad h(x) = [f_r(x)]^2 + 1.$$

The following subroutine DIFEQU incorporates this information in a form expected in COLLOC and PUTIT. It also contains the above formula for g.

```
      SUBROUTINE DIFEQU ( MODE, XX, V )
CALLS  PPVALU(INTERV)
C  TO BE CALLED BY   C O L L O C ,  P U T I T
C  INFORMATION ABOUT THE DIFFERENTIAL EQUATION IS DISPENSED FROM HERE
C
C******  I N P U T  ******
C  MODE  AN INTEGER INDICATING THE TASK TO BE PERFORMED.
C        = 1   INITIALIZATION
C        = 2   EVALUATE  DE  AT  XX
C        = 3   SPECIFY THE NEXT SIDE CONDITION
C        = 4   ANALYZE THE APPROXIMATION
C  XX A POINT AT WHICH INFORMATION IS WANTED
C
C******  O U T P U T  ******
C  V  DEPENDS ON THE  MODE  . SEE COMMENTS BELOW
C
      INTEGER MODE,   I,ISIDE,ITERMX,K,KMAX,KPM,L,M,NCOEF,NPIECE
      PARAMETER (NPIECE=100, KMAX=20, NCOEF=NPIECE*KMAX)
      REAL V(KMAX),XX,   BREAK,COEF,EPS,EP1,EP2,ERROR,FACTOR,RHO,SOLUTN
     *                  ,S20VEP,UN,X,XSIDE
      COMMON /APPROX/ BREAK(NPIECE),COEF(NCOEF),L,KPM
      COMMON /SIDE/ M,ISIDE,XSIDE(10)
      COMMON /OTHER/ ITERMX,K,RHO(KMAX-1)
      SAVE EPS,FACTOR,S20VEP
C
```

```
C  THIS SAMPLE OF  DIFEQU  IS FOR THE EXAMPLE IN CHAPTER XV. IT IS A
C  NONLINEAR SECOND ORDER TWO POINT BOUNDARY VALUE PROBLEM.
C
                                        GO TO (10,20,30,40),MODE
C  INITIALIZE EVERYTHING
C  I.E. SET THE ORDER  M  OF THE DIF.EQU., THE NONDECREASING SEQUENCE
C  XSIDE(I),I=1,...,M, OF POINTS AT WHICH SIDE COND.S ARE GIVEN AND
C  ANYTHING ELSE NECESSARY.
   10 M = 2
      XSIDE(1) = 0.
      XSIDE(2) = 1.
C  *** PRINT OUT HEADING
      PRINT 499
  499 FORMAT(' CARRIER,S NONLINEAR PERTURB. PROBLEM')
      EPS = .5E-2.
      PRINT 610, EPS
  610 FORMAT(' EPS ',E20.10)
C  *** SET CONSTANTS USED IN FORMULA FOR SOLUTION BELOW.
      FACTOR = (SQRT(2.) + SQRT(3.))**2
      S2OVEP = SQRT(2./EPS)
C  *** INITIAL GUESS FOR NEWTON ITERATION. UN(X) = X*X - 1.
      L = 1
      BREAK(1) = 0.
      DO 16 I=1,KPM
   16    COEF(I) = 0.
      COEF(1) = -1.
      COEF(3) = 2.
      ITERMX = 10
                                        RETURN
C
C  PROVIDE VALUE OF LEFT SIDE COEFF.S AND RIGHT SIDE AT  XX .
C  SPECIFICALLY, AT  XX  THE DIF.EQU. READS
C        V(M+1)D**M + V(M)D**(M-1) + ... + V(1)D**0  =  V(M+2)
C  IN TERMS OF THE QUANTITIES V(I),I=1,...,M+2, TO BE COMPUTED HERE.
   20 CONTINUE
      V(3) = EPS
      V(2) = 0.
      UN = PPVALU(BREAK,COEF,L,KPM,XX,0)
      V(1) = 2.*UN
      V(4) = UN**2 + 1.
                                        RETURN
C
C  PROVIDE THE  M  SIDE CONDITIONS. THESE CONDITIONS ARE OF THE FORM
C        V(M+1)D**M + V(M)D**(M-1) + ... + V(1)D**0  =  V(M+2)
C  IN TERMS OF THE QUANTITIES V(I),I=1,...,M+2, TO BE SPECIFIED HERE.
C  NOTE THAT V(M+1) = 0  FOR CUSTOMARY SIDE CONDITIONS.
   30 V(M+1) = 0.
                                        GO TO (31,32,39),ISIDE

   31 V(2) = 1.
      V(1) = 0.
      V(4) = 0.
                                        GO TO 38

   32 V(2) = 0.
      V(1) = 1.
      V(4) = 0.
   38 ISIDE = ISIDE + 1
   39                                   RETURN
C
C  CALCULATE THE ERROR NEAR THE BOUNDARY LAYER AT  1.
   40 CONTINUE
      PRINT 640
  640 FORMAT(' X, G(X)  AND  G(X)-F(X)  AT SELECTED POINTS')
      X = .75
```

```
      DO 41 I=1,9
         EP1 = EXP(S20VEP*(1.-X))*FACTOR
         EP2 = EXP(S20VEP*(1.+X))*FACTOR
         SOLUTN = 12./(1.+EP1)**2*EP1 + 12./(1.+EP2)**2*EP2 - 1.
         ERROR = SOLUTN - PPVALU(BREAK,COEF,L,KPM,X,0)
         PRINT 641,X,SOLUTN,ERROR
  641    FORMAT(3E20.10)
   41    X = X + .03125
                                                  RETURN
      END
```

The following driver sets the calculations into motion and produces the output listed below.

```
C  MAIN PROGRAM FOR EXAMPLE IN CHAPTER XV.
C  SOLUTION OF A SECOND ORDER NONLINEAR TWO POINT BOUNDARY VALUE
C  PROBLEM  ON [0 .. 1] , BY COLLOCATION WITH PP FUNCTIONS HAVING
C  PIECES OF ORDER  6 .  2  PASSES THROUGH NEWNOT ARE TO BE MADE,
C  WITHOUT ANY KNOTS BEING ADDED. NEWTON ITERATION IS TO BE STOPPED
C  WHEN TWO ITERATES AGREE TO  6  DECIMAL PLACES.
       CALL COLLOC(0.,1.,4,6,2,0.,1.E-6)
                                                  STOP
       END
```

The error printout shows the improvement near the boundary layer brought about by the repositioning of the breaks through NEWNOT. Not only is the size of the error reduced, but the number of sign changes in the error is increased from 2 to 5. Note also the decrease in the number of Newton steps required. This is due to the fact that the program always uses the most recent approximation to the solution as the current guess f_r even when the next approximation is to have a different knot sequence. Finally, note the detailed picture of the approximate solution provided by the ppform: the sharp change of f near 1 is quite evident.

```
CARRIER,S NONLINEAR PERTURB. PROBLEM
EPS     0.4999999888-02
APPROXIMATION FROM A SPACE OF SPLINES OF ORDER  6 ON   4 INTERVALS,
OF DIMENSION  18.  BREAKPOINTS -
   0.2500000000+00    0.5000000000+00    0.7500000000+00
AFTER  5 OF 10 ALLOWED ITERATIONS,
THE PP REPRESENTATION OF THE APPROXIMATION IS
   0.000-0.100000+01  0.000+00 -0.381-04  0.252-02 -0.659-01  0.692+00
   0.250-0.100000+01  0.101-04 -0.486-03  0.477-01 -0.132+01  0.195+02
   0.500-0.999944+00  0.112-02 -0.629-01  0.676+01 -0.188+03  0.280+04
   0.750-0.991799+00  0.163+00  0.452+01 -0.139+02  0.215+04  0.609+05
```

```
X, G(X)  AND  G(X)-F(X)  AT SELECTED POINTS
   0.7500000000+00   -0.9918430448+00   -0.4374980927-04
   0.7812500000+00   -0.9847788215+00   -0.3274083138-03
   0.8125000000+00   -0.9716256857+00   -0.1661181450-03
   0.8437500000+00   -0.9472073317+00    0.6934404373-03
   0.8750000000+00   -0.9021227360+00    0.1047730446-02
   0.9062500000+00   -0.8197249174+00   -0.5564689636-03
   0.9375000000+00   -0.6719498634+00   -0.3465056419-02
   0.9687500000+00   -0.4159959555+00   -0.3410220146-02
   0.1000000000+01    0.0000000000+00    0.0000000000+00
APPROXIMATION FROM A SPACE OF SPLINES OF ORDER  6 ON   4 INTERVALS,
OF DIMENSION  18.  BREAKPOINTS -
   0.4417714477+00    0.6529041529+00    0.8314099312+00
AFTER  2 OF 10 ALLOWED ITERATIONS,
THE PP REPRESENTATION OF THE APPROXIMATION IS
   0.000-0.100000+01  0.000+00 -0.293-03  0.130-01 -0.220+00  0.149+01
   0.442-0.999983+00  0.339-03 -0.198-02  0.938+00 -0.274+02  0.550+03
   0.653-0.998827+00  0.234-01  0.261+00  0.315+02 -0.820+03  0.242+05
   0.831-0.958661+00  0.821+00  0.199+02 -0.969+02  0.235+05 -0.156+06
X, G(X)  AND  G(X)-F(X)  AT SELECTED POINTS
   0.7500000000+00   -0.9918430448+00   -0.3850460052-04
   0.7812500000+00   -0.9847788215+00    0.1472234726-04
   0.8125000000+00   -0.9716256857+00    0.1883506775-04
   0.8437500000+00   -0.9472073317+00   -0.1868605614-03
   0.8750000000+00   -0.9021227360+00   -0.1836419106-03
   0.9062500000+00   -0.8197249174+00    0.7538199425-03
   0.9375000000+00   -0.6719498634+00    0.9310245514-04
   0.9687500000+00   -0.4159959555+00   -0.1260221004-02
   0.1000000000+01    0.0000000000+00    0.0000000000+00
APPROXIMATION FROM A SPACE OF SPLINES OF ORDER  6 ON   4 INTERVALS,
OF DIMENSION  18.  BREAKPOINTS -
   0.4450708628+00    0.6789402366+00    0.8465198278+00
AFTER  1 OF 10 ALLOWED ITERATIONS,
THE PP REPRESENTATION OF THE APPROXIMATION IS
   0.000-0.100000+01  0.670-06 -0.331-03  0.138-01 -0.231+00  0.155+01
   0.445-0.999981+00  0.365-03 -0.108-01  0.161+01 -0.461+02  0.770+03
   0.679-0.998027+00  0.394-01  0.555+00  0.425+02 -0.990+03  0.351+05
   0.847-0.944220+00  0.111+01  0.253+02 -0.248+02  0.291+05 -0.254+06
X, G(X)  AND  G(X)-F(X)  AT SELECTED POINTS
   0.7500000000+00   -0.9918430448+00   -0.3337860107-04
   0.7812500000+00   -0.9847788215+00   -0.2211332321-04
   0.8125000000+00   -0.9716256857+00    0.3129243851-04
   0.8437500000+00   -0.9472073317+00   -0.6973743439-05
   0.8750000000+00   -0.9021227360+00   -0.3032088280-03
   0.9062500000+00   -0.8197249174+00    0.4141330719-03
   0.9375000000+00   -0.6719498634+00    0.3328919411-03
   0.9687500000+00   -0.4159959555+00   -0.8399486542-03
   0.1000000000+01    0.0000000000+00    0.0000000000+00
```

□

Problems

1. Estimate the work involved in setting up the collocation equations (that is, the labor done in EQBLOK and PUTIT) and compare to the work of solving the resulting equations (in SLVBLK, that is, by Gauss elimination with attention being paid to the special structure).

2. (i) If ξ is uniform, then the generation of the collocation equations can be made more efficient by calling BSPLVB only in the first interval.

(ii) Even if ξ is not uniform, certain savings can be effected in EQBLOK and PUTIT due to the fact that every B-spline involves at most two ξ-intervals, and some have their support on only one ξ-interval.

3. Discuss the proper treatment of nonlinear side conditions. Specifically, if (2) is replaced by

$$G_i(x_i; g(x_i), \ldots, (D^{m-1}g)(x_i)) =: \beta_i g = c_i, \qquad i = 1, \ldots, m,$$

what would be the appropriate explicit formulation of (9) (assuming that G_i is a smooth function of its arguments)? (Hint: How was (9) obtained from (4)?)

4. Identify the above collocation method with implicit Runge-Kutta methods, in case $\beta_i g = g^{(i-1)}(a)$, $i = 1, \ldots, m$.

5. Let $m = 2$, $k = 4$. Prove that the polynomial $p \in \Pi_{<8}$ that agrees with the collocation approximation f to g at $(\xi_{i-1}, \xi_{i-1}, \xi_i, \xi_i, \xi_{i+1}, \xi_{i+1}, \xi_{i+2}, \xi_{i+2})$ is an $\mathcal{O}(|\xi|^8)$-approximation to g on $[\xi_{i-1} \,..\, \xi_{i+2}]$.

6. Set up the relevant equations for solving the integral equation

$$g(x) + \int_a^b K(x,y)g(y)\,dy = h(x) \quad \text{on} \quad [a\,..\,b]$$

of the second kind for g by collocation from $\Pi_{<k,\xi}$.

7. What would the relevant equations be for solving (1) by Galerkin's method using $\Pi_{<m+k,\xi} \cap C^{(m-1)}$?

8. Rework COLLOC and its subroutines to make use instead of the subroutine CWIDTH given in the appendix which requires less storage than does SLVBLK.

XVI

Taut Splines, Periodic Splines, Cardinal Splines and the Approximation of Curves;

TAUTSP

In this chapter, we discuss various details concerning the approximation of curves, leading us eventually to a side issue, namely the approximation by periodic splines with a uniform knot sequence. But, with the exception of taut splines, or, splines in tension, all is amplification of earlier material.

We will deal with *planar* curves, that is, with point sets in the (x, y)-plane of the form

$$C_c := \{\, (c_x(s), c_y(s)) : a \le s \le b \,\},$$

for some interval $[a\,..\,b]$ and some (usually continuous) function pair c_x, c_y.

The typical computational problem concerns the construction of a curve C_c that contains a prescribed point sequence $(P_i)_1^N$ in the plane in that order. Two aspects of this problem give difficulty. One is the proper choice for the parametrization. The second problem is lack of data. We will discuss each of these problems in turn, but begin with the second problem.

Lack of data This is an intuitive and imprecise term that is meant to describe the following situation. A curve or function has been approximated by some means, making use of the data provided. It fits the data, yet the intended user of the approximation protests that it doesn't look right, or, that she would have drawn it differently. In effect, the data provided were not sufficient to convey to the approximation scheme the user's wishes.

Of course, such user complaints are at times as unreasonable as they sound and can then be dealt with simply by requesting more data in the area where the approximation was felt to be wrong. But, in at least one situation, there is justification for complaint, and this is in the case of "extraneous" inflection points. This we discuss now in some detail for the case of interpolation to a function rather than a curve.

"Extraneous" inflection points We are given data $(\tau_i, g(\tau_i))_1^n$ to be interpolated. Let f be a smooth interpolant to these data. We know from I(vii) that, for each i, there is a site η_i in the open interval $(\tau_{i-1} \,..\, \tau_{i+1})$ so that

$$[\tau_{i-1}, \tau_i, \tau_{i+1}]g = f''(\eta_i)/2.$$

Therefore, corresponding to each sign change in the sequence $(\delta_i)_2^{n-1}$, with

$$\delta_i := [\tau_i, \tau_{i+1}]g - [\tau_i, \tau_{i-1}]g,$$

there must be a sign change in the second derivative of f, that is, an inflection point of f. We call an inflection point of the interpolant f in the open interval $(\tau_i \,..\, \tau_{i+1})$ **extraneous** if $\delta_i\delta_{i+1} > 0$.

In effect, we insist that the interpolant preserve convexity/concavity of the data in the following sense: *if the broken line interpolant to the data is convex (concave) on the interval* $(\tau_{r-1} \,..\, \tau_{s+1})$, *then a "good" interpolant should be convex (concave) on the interval* $[\tau_r \,..\, \tau_s]$. This prescription allows only one interpolant on $[\tau_{i-1} \,..\, \tau_{i+1}]$ in case $\delta_i = 0$, namely a straight line.

Spline in tension Schweikert [1966] was the first to deal with the problem of extraneous inflection points. He proposed the *tensioned* spline as a remedy. In constructing this spline, he was guided by the physical notion of the (graph of the) spline as an elastic band that goes through rings at the interpolation points in the plane and that is pulled until all extraneous kinks have been straightened out. In the limit, that is, with a very strong pull, one obtains the broken line interpolant.

In mathematical terms, Schweikert's tensioned interpolant has two continuous derivatives and satisfies, between interpolation sites, the differential equation

$$(D^2 - p^2)\,D^2 y = 0,$$

with p the tension parameter. For $p = 0$, this means that the interpolant is piecewise cubic, that is, just the customary cubic spline interpolant described in Chapter IV. But, for $p > 0$, each piece of the interpolant is a linear combination of the four functions

$$1, \quad x, \quad e^{px}, \quad e^{-px}.$$

Algorithms for the construction of such a tensioned spline have been published in Fortran by A. Cline [1974]. Späth (see, for example, his book Späth [1974]) has generalized this to allow the tension parameter p to vary from interval to interval, thus making the procedure more responsive to the local behavior of the data. In fact, Späth deals with more general interpolants that, on the interval $[\tau_i \,..\, \tau_{i+1}]$, have the form

$$f(x) = A_i u + B_i v + C_i \varphi_i(u) + D_i \varphi_i(v) \tag{1}$$

with

$$u := u(x) := (x - \tau_i)/\Delta\tau_i, \qquad v := 1 - u,$$

and φ_i a function on $[0\,..\,1]$ with $\varphi_i(0) = \varphi_i(1) = 0$, and $\varphi_i''(0)^2 \neq \varphi_i''(1)^2$. For $\varphi_i(s) = s^3 - s$, all s, one recovers the cubic spline. The piecewise tensioned spline fits into this scheme with the choice

$$\varphi_i(s) = \sinh(p_i s) - s\sinh(p_i),$$

with p_i the tension parameter for the ith interval.

A different scheme has been proposed by R.V. Soanes, Jr., [1976], who uses an interpolant that, on the interval $[\tau_i\,..\,\tau_{i+1}]$, has the form

$$f(x) = A_i u + B_i v + C_i u^{m_i} + D_i v^{n_i},$$

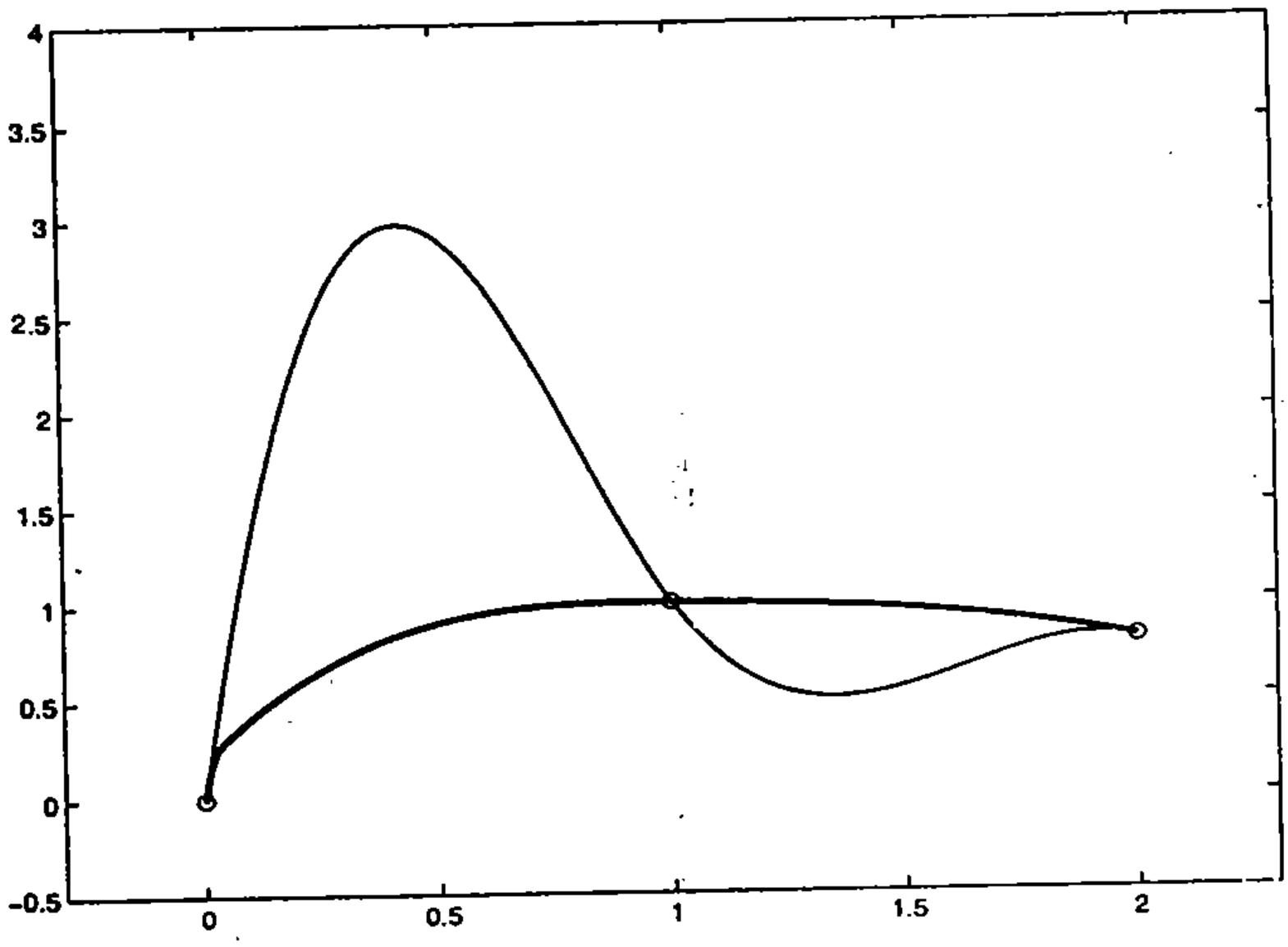
with the possibly fractional exponents m_i, n_i to be chosen appropriately. The choice $m_i = n_i = 3$ reduces this scheme to cubic spline interpolation. While the Schweikert–Späth tensioned spline has its tension parameters chosen interval for interval, Soanes chooses the *ratio* m_{i-1}/n_i data point for data point.

The only real objection to these and other schemes is that they use *exponential functions* instead of polynomials, thereby making evaluation of the interpolant more expensive and, on some minicomputers, even impractical. One wonders, therefore, whether the same effect, namely preservation of convexity/concavity in the data, could not be had while sticking to cubic splines as interpolants. As it turns out, this is indeed possible. The trick is to use a cubic spline with *additional knots* placed so that the interpolant can make sharp bends where required without breaking into oscillations as a consequence.

(2) Example: Coping with a large endslope We are given the following data $f(0) = 0$, $f'(0) = +\infty$, $f(1) = f(2) = 1$, $f'(2) = -1/2$, and are to construct a concave interpolant.

If we really insist on an infinite slope at 0, there is no way of solving this problem with polynomial splines. But, if this is merely a design curve, then a "very large" slope, say $f'(0) = 20$, will do just as well. But now complete cubic spline interpolation to these (modified) data produces an interpolant with two inflection points; see Figure (3).

Quite simply, the cubic polynomial in the first interval cannot bend fast enough. We help it along by adding just one knot, very close to the site of difficulty. In effect, we supply "missing data". Specifically, we give the additional data point (.04, .03). The new interpolant is also shown in Figure (3); it is concave. The transition from the large endslope to the mild behavior in the rest of the interval could easily be made less abrupt by supplying more data, for example by taking points from a spline *curve* that fits the original data.



(3) FIGURE. Spline interpolation to a large endslope without (thin) and with (thick) an additional knot. □

A taut cubic spline It took a little experimentation in Example (2) to find the appropriate additional data point. Typically, one chooses the abscissa and then experiments with the function value until the curve is satisfactory. In the following, we describe a more systematic way of introducing additional knots where required.

We begin with the following special case. We are to interpolate to the data $f(0) = 0$, $f'(0) = s_0$, $f(1) = 0$, $f'(1) = s_1$, by a cubic polynomial. From IV(3)–(4), we get at once that

$$f(x) = s_0 x - (2s_0 + s_1)x^2 + (s_0 + s_1)x^3. \tag{4}$$

We have $[0,0,1]f = -s_0$, $[0,1,1]f = s_1$, hence an inflection point in $(0\,..\,1)$ would be extraneous in case $s_0 s_1 < 0$. On the other hand,

$$f''(0) = -2(2s_0 + s_1), \quad f''(1) = 2(s_0 + s_1).$$

Thus, f has no (extraneous) inflection points in $(0\,..\,1)$ if and only if $(2s_0 + s_1)(s_0 + 2s_1) \le 0$. We can write this condition also as $(2(s_0 - s_1) + 3s_1)((s_0 - s_1) + 3s_1) \le 0$, or, dividing through by $(s_0 - s_1)^2$ and using the abbreviation $z := s_1/(s_1 - s_0)$, as $(2 - 3z)(1 - 3z) \le 0$.

It follows that the cubic interpolant in this case reproduces the convexity/concavity of the data only if

$$1/3 \leq z := s_1/(s_1 - s_0) \leq 2/3.$$

For values of z outside this range, we propose to change the interpolating function as follows.

Consider first the case $z > 2/3$. Then the interpolant is required to bend near 1 more sharply than a monotone cubic can. We therefore replace the one basis function for $\Pi_{<4}$ among the four: x, $1-x$, x^3, $(1-x)^3$, that bends most sharply near 1, namely x^3, by a cubic spline φ with one knot that bends more sharply near 1 than does x^3. For the sake of uniformity, we also insist that $\varphi(0) = \varphi'(0) = \varphi''(0) = 0$, $\varphi(1) = 1$, and $\varphi'' \geq 0$ on $[0\,..\,1]$. This ensures, among other things, that the interpolant has an inflection point in $[0\,..\,1]$ if and only if the coefficients of φ and $(1-x)^3$ are of opposite sign, as we will see in a moment. The resulting function φ is given by

$$\varphi(x) := \varphi(x; z) := \alpha x^3 + (1-\alpha)\left(\frac{x-\zeta}{1-\zeta}\right)_+^3$$

with $\zeta = \zeta(z)$ the additional knot and $\alpha = \alpha(z)$ a number in $[0\,..\,1]$. We take both ζ and α to be continuous monotone functions of z, with $\alpha(2/3) = 1$ in order to connect continuously with the usual cubic when $z \leq 2/3$, and with $\zeta(1^-) = 1$.

In terms of this new basis, the interpolant now reads

$$f(x) = Ax + B(1-x) + C\varphi(x; z) + D(1-x)^3$$

where

$$-A = C = (s_0 + 2s_1)/(3(2p-1))$$
$$-B = D = -((3p-1)s_0 + s_1)/(3(2p-1))$$

and

$$p := \varphi'(1; z)/3 = \alpha + (1-\alpha)/(1-\zeta).$$

It follows that $f''(x) = 6(C(\alpha x + (1-\alpha)(x-\zeta)_+/(1-\zeta)^3) + D(1-x))$. This is of one sign in $[0\,..\,1]$ if and only if C and D are of the same sign. The condition that f have no extraneous inflection point in $(0\,..\,1)$ (in case $s_0 s_1 < 0$) thus becomes

$$((3p-1)s_0 + s_1)(s_0 + 2s_1) \leq 0.$$

Slightly rearranging, dividing through by $(s_0 - s_1)^2$, and using again $z := s_1/(s_1 - s_0)$ produces the equivalent condition that

$$((3p-1) - 3pz)(1-3z) \leq 0.$$

The second factor is negative since we are considering the case $z > 2/3$. Preservation of convexity/concavity of the modified interpolant therefore demands (even under the weaker assumption $z \geq 1/3$) that $z \leq (3p - 1)/(3p)$, that is,

$$3p \geq 1/(1-z).$$

In terms of α, this condition reads $\alpha \leq [1 - (1-\zeta)/(3(1-z))]/\zeta$. Thus, choosing α to be as large as possible, we get the formula

$$\alpha(z) = \left(1 - \frac{1-\zeta}{3(1-z)}\right) \Big/ \zeta.$$

Note that $\alpha(2/3) = 1$ regardless of ζ. This also shows that we need

$$3(1-z) \geq 1 - \zeta$$

to make certain that $\alpha \geq 0$. This condition is satisfied for the choice $1 - \zeta(z) = \gamma(1-z)$ as long as $\gamma \leq 3$.

The case $z < 1/3$ is handled analogously, with the basis function $(1-x)^3$ replaced by $\varphi(1-x; 1-z)$ and φ determined as in the case $z > 2/3$.

We can combine these various cases conveniently into one by considering the approximation to be of the form

$$f(x) = Ax + B(1-x) + C\varphi(x; z) + D\varphi(1-x; 1-z) \tag{5}$$

with

$$\begin{aligned} \varphi(x; z) &:= \alpha x^3 + (1-\alpha)\left(\frac{x-\zeta}{1-\zeta}\right)_+^3 \\ \alpha(z) &:= (1 - \gamma/3)/\zeta \\ \zeta(z) &:= 1 - \gamma \min\{\, 1-z, 1/3 \,\} \end{aligned} \tag{6}$$

for some $\gamma \in [0\,..\,3]$. As γ increases, both α and ζ decrease for fixed z. The overall effect is one of increased "roundness" for increased γ.

The more general case of interpolation to data $f(0) = f_0$, $f'(0) = s_0$, $f(1) = f_1$, $f'(1) = s_1$ is reduced to the present one by subtracting first the straight line $f_0 + (f_1 - f_0)x$. This subtraction does not affect questions of convexity and concavity, and changes the quantities s_0 and s_1 used above into $s_0 - (f_1 - f_0)$ and $s_1 - (f_1 - f_0)$. If, even more generally, interpolation to data $f(\tau_j) = f_j$, $f'(\tau_j) = s_j$, $j = i, i+1$ is considered, then, instead of s_0 and s_1, the relevant quantities are

$$s_i - [\tau_i, \tau_{i+1}]f \qquad \text{and} \qquad s_{i+1} - [\tau_i, \tau_{i+1}]f.$$

We make use of these considerations in the construction of a "taut" cubic spline interpolant to given data $((\tau_i, g(\tau_i)))_1^N$ by choosing the basis in each

interval $[\tau_i \,..\, \tau_{i+1}]$ in the above manner, with the role of s_i played by the slope $[\tau_{i-1}, \tau_i]f$ and the role of s_{i+1} played by $[\tau_{i+1}, \tau_{i+2}]f$. This means that the interpolant has the form

$$A_i + B_i u + C_i \varphi(u; z) + D_i \varphi(1-u; 1-z) \tag{7}$$

on $[\tau_i \,..\, \tau_{i+1}]$, with

$$u(x) := (x - \tau_i)/\Delta\tau_i \tag{8}$$

$$z = z_i := \begin{cases} \delta_{i+1}/(\delta_i + \delta_{i+1}) & \text{if } \delta_i\delta_{i+1} \geq 0, \quad \delta_i + \delta_{i+1} \neq 0 \\ 1/2 & \text{otherwise} \end{cases} \tag{9}$$

where

$$\delta_j := [\tau_j, \tau_{j+1}]f - [\tau_j, \tau_{j-1}]f, \qquad \text{all } j,$$

as before. We set $z = 1/2$ in case $i = 1$ or $i = N-1$.

The particular choice (9) of z makes the interpolant discontinuous as a function of the data. This means that a small change in the data that changes the sign of some $\delta_i\delta_{i+1}$ may change the interpolant drastically. For this reason, the program below also allows the modified choice

$$z = z_i := \begin{cases} |\delta_{i+1}|/(|\delta_i| + |\delta_{i+1}|) & \text{if } |\delta_i| + |\delta_{i+1}| > 0 \\ 1/2 & \text{otherwise} \end{cases} \tag{10}$$

which removes an obvious source of discontinuity.

In terms of the quantities f_j, f''_j, $j = i, i+1$, and the number $h := \Delta\tau_i$, the interpolant has the coefficients

$$\begin{aligned} &A_i = f_i - D_i, \qquad B_i = h[\tau_i, \tau_{i+1}]f - (C_i - D_i) \\ &C_i = h^2 f''_{i+1}/\varphi''(1; z), \qquad D_i = h^2 f''_i/\varphi''(1; 1-z). \end{aligned} \tag{11}$$

Note that $\varphi''(1; z) = 6(\alpha + (1-\alpha)/(1-\zeta)^2)$, hence

$$1/\varphi''(1; z) = \frac{1}{6}(1-\zeta)^2/\big(\alpha(1-\zeta)^2 + (1-\alpha)\big). \tag{12}$$

We determine the vector $(f''_i)_1^N$ so that the resulting interpolant has two continuous derivatives. This gives, at each interior data site τ_i, the equation $f'(\tau_i^-) = f'(\tau_i^+)$, or

(13)

$$\begin{aligned} \frac{\Delta\tau_{i-1}}{\varphi''(1; 1-z_{i-1})} f''_{i-1} + \left(\Delta\tau_{i-1} \frac{\varphi'(1; z_{i-1}) - 1}{\varphi''(1; z_{i-1})} + \frac{\varphi'(1; 1-z_i) - 1}{\varphi''(1; 1-z_i)} \Delta\tau_i\right) f''_i \\ + \frac{\Delta\tau_i}{\varphi''(1; z_i)} f''_{i+1} = \delta_i, \qquad i = 2, \ldots, N-1. \end{aligned}$$

Note that

$$\frac{\varphi'(1;z)-1}{\varphi''(1;z)} = \frac{1-\zeta}{6}\,\frac{(3\alpha-1)(1-\zeta)+3(1-\alpha)}{\alpha(1-\zeta)^2+(1-\alpha)}. \tag{14}$$

Evaluation of the coefficients in (13) in the manner of (12) and (14) ensures accuracy as $\zeta \to 1$.

If no additional knots are introduced, that is, if $\alpha = 1$ throughout, then (12) and (14) show that (13) reduces to

$$(\Delta\tau_{i-1}/6)f''_{i-1} + (\Delta\tau_{i-1}/3 + \Delta\tau_i/3)f''_i + (\Delta\tau_i/6)f''_{i+1} = \delta_i,$$

the equation for cubic spline interpolation familiar from Problem IV.6.

We need two more equations to obtain N equations in the N unknowns $(f''_i)_1^N$. Of the several choices of boundary conditions discussed in Chapter IV, we consider here only the "not-a-knot" condition. This means that we adjoin the two equations

$$\operatorname{jump}_{\tau_2} f''' = 0 = \operatorname{jump}_{\tau_{N-1}} f'''.$$

More explicitly, by (7) and (11), the first of these reads

$$(f''_2 - f''_1)/\Delta\tau_1 = \left(\frac{\varphi'''(0;z_2)}{\varphi''(1;z_2)}\,f''_3 - \frac{\varphi'''(1;1-z_2)}{\varphi''(1;1-z_2)}\,f''_2\right)/\Delta\tau_2,$$

and the second looks similar. Note that

$$\frac{\varphi''(1;z)}{\varphi'''(1;z)} = (1-\zeta)\frac{\alpha(1-\zeta)^2+(1-\alpha)}{\alpha(1-\zeta)^3+(1-\alpha)}, \tag{15}$$

so that the equation is in computational difficulty if $z_2 = 0$. For this reason, we write the boundary equation in the form

$$\begin{aligned}\frac{\varphi''(1;1-z_2)}{\varphi'''(1;1-z_2)}\Delta\tau_2(f''_2 - f''_1) = \\ \left(\frac{\varphi''(1;1-z_2)}{\varphi'''(1;1-z_2)}\,\frac{\varphi'''(0;z_2)}{\varphi''(1;z_2)}\,f''_3 - f''_2\right)\Delta\tau_1,\end{aligned} \tag{16}$$

with an analogous version for the equation $\operatorname{jump}_{\tau_{N-1}} f''' = 0$. Finally, a detailed analysis shows that these equations should be the second, respectively the second last, in the total system in order to avoid the necessity of pivoting.

```
      SUBROUTINE TAUTSP ( TAU, GTAU, NTAU, GAMMA, S,
     *                    BREAK, COEF, L, K, IFLAG )
CONSTRUCTS CUBIC SPLINE INTERPOLANT TO GIVEN DATA
C         TAU(I), GTAU(I), I=1,...,NTAU.
C  IF  GAMMA .GT. 0., ADDITIONAL KNOTS ARE INTRODUCED WHERE NEEDED TO
C  MAKE THE INTERPOLANT MORE FLEXIBLE LOCALLY. THIS AVOIDS EXTRANEOUS
C  INFLECTION POINTS TYPICAL OF CUBIC SPLINE INTERPOLATION AT KNOTS TO
C  RAPIDLY CHANGING DATA.
C
C  PARAMETERS
C            INPUT
C  TAU       SEQUENCE OF DATA POINTS. MUST BE STRICTLY INCREASING.
C  GTAU      CORRESPONDING SEQUENCE OF FUNCTION VALUES.
C  NTAU      NUMBER OF DATA POINTS. MUST BE AT LEAST  4 .
C  GAMMA     INDICATES WHETHER ADDITIONAL FLEXIBILITY IS DESIRED.
C           = 0., NO ADDITIONAL KNOTS
C           IN (0.,3.), UNDER CERTAIN CONDITIONS ON THE GIVEN DATA AT
C                 POINTS I-1, I, I+1, AND I+2, A KNOT IS ADDED IN THE
C                 I-TH INTERVAL, I=2,...,NTAU-2. SEE DESCRIPTION OF METH-
C                 OD BELOW. THE INTERPOLANT GETS ROUNDED WITH INCREASING
C                 GAMMA. A VALUE OF  2.5  FOR GAMMA IS TYPICAL.
C           IN (3.,6.), SAME , EXCEPT THAT KNOTS MIGHT ALSO BE ADDED IN
C                 INTERVALS IN WHICH AN INFLECTION POINT WOULD BE PERMIT-
C                 TED.  A VALUE OF  5.5  FOR GAMMA IS TYPICAL.
C             OUTPUT
C  BREAK, COEF, L, K  GIVE THE PP-REPRESENTATION OF THE INTERPOLANT.
C           SPECIFICALLY, FOR BREAK(I) .LE. X .LE. BREAK(I+1), THE
C         INTERPOLANT HAS THE FORM
C  F(X) = COEF(1,I) +DX(COEF(2,I) +(DX/2)(COEF(3,I) +(DX/3)COEF(4,I)))
C        WITH  DX = X - BREAK(I) AND I=1,...,L .
C  IFLAG    = 1, OK
C           = 2, INPUT WAS INCORRECT. A PRINTOUT SPECIFYING THE MISTAKE
C             WAS MADE.
C             WORKSPACE
C  S      IS REQUIRED, OF SIZE (NTAU,6). THE INDIVIDUAL COLUMNS OF THIS
C         ARRAY CONTAIN THE FOLLOWING QUANTITIES MENTIONED IN THE WRITE-
C         UP AND BELOW.
C     S(.,1) = DTAU = TAU(.+1) - TAU
C     S(.,2) = DIAG = DIAGONAL IN LINEAR SYSTEM
C     S(.,3) = U = UPPER DIAGONAL IN LINEAR SYSTEM
C     S(.,4) = R = RIGHT SIDE FOR LINEAR SYSTEM (INITIALLY)
C            = FSECND = SOLUTION OF LINEAR SYSTEM , NAMELY THE SECOND
C                       DERIVATIVES OF INTERPOLANT AT  TAU
C     S(.,5) = Z = INDICATOR OF ADDITIONAL KNOTS
C     S(.,6) = 1/HSECND(1,X) WITH X = Z OR = 1-Z. SEE BELOW.
C
C  ------  M E T H O D  ------
C  ON THE I-TH INTERVAL, (TAU(I)..TAU(I+1)), THE INTERPOLANT IS OF THE
C  FORM
C  (*)  F(U(X)) = A + B*U + C*H(U,Z) + D*H(1-U,1-Z) ,
C  WITH  U = U(X) = (X - TAU(I))/DTAU(I). HERE,
C       Z = Z(I) = ADDG(I+1)/(ADDG(I) + ADDG(I+1))
C  (= .5, IN CASE THE DENOMINATOR VANISHES). WITH
C       ADDG(J) = ABS(DDG(J)), DDG(J) = DG(J+1) - DG(J),
C       DG(J) = DIVDIF(J) = (GTAU(J+1) - GTAU(J))/DTAU(J)
C  AND
C       H(U,Z) = ALPHA*U**3 + (1 - ALPHA)*(MAX(((U-ZETA)/(1-ZETA)),0)**3
C  WITH
C       ALPHA(Z) = (1-GAMMA/3)/ZETA
C       ZETA(Z) = 1 - GAMMA*MIN((1 - Z), 1/3)
C  THUS, FOR 1/3 .LE. Z .LE. 2/3,  F  IS JUST A CUBIC POLYNOMIAL ON
C  THE INTERVAL I. OTHERWISE, IT HAS ONE ADDITIONAL KNOT, AT
C          TAU(I) + ZETA*DTAU(I) .
C  AS  Z  APPROACHES  1, H(.,Z) HAS AN INCREASINGLY SHARP BEND  NEAR 1,
C  THUS ALLOWING  F  TO TURN RAPIDLY NEAR THE ADDITIONAL KNOT.
C     IN TERMS OF F(J) = GTAU(J) AND
C       FSECND(J) = 2.DERIVATIVE OF  F  AT  TAU(J),
```

```
C  THE COEFFICIENTS FOR (*) ARE GIVEN AS
C       A = F(I) - D
C       B = (F(I+1) - F(I)) - (C - D)
C       C = FSECND(I+1)*DTAU(I)**2/HSECND(1,Z)
C       D = FSECND(I)*DTAU(I)**2/HSECND(1,1-Z)
C  HENCE CAN BE COMPUTED ONCE FSECND(I),I=1,...,NTAU, IS FIXED.
C   F  IS AUTOMATICALLY CONTINUOUS AND HAS A CONTINUOUS SECOND DERIVAT-
C  IVE (EXCEPT WHEN Z = 0 OR 1 FOR SOME I). WE DETERMINE FSCND(.) FROM
C  THE REQUIREMENT THAT ALSO THE FIRST DERIVATIVE OF  F  BE CONTIN-
C  UOUS. IN ADDITION, WE REQUIRE THAT THE THIRD DERIVATIVE BE CONTINUOUS
C  ACROSS  TAU(2) AND ACROSS  TAU(NTAU-1) . THIS LEADS TO A STRICTLY
C  DIAGONALLY DOMINANT TRIDIAGONAL LINEAR SYSTEM FOR THE FSECND(I)
C  WHICH WE SOLVE BY GAUSS ELIMINATION WITHOUT PIVOTING.
C
      INTEGER IFLAG,K,L,NTAU,    I,METHOD,NTAUM1
      REAL BREAK(1),COEF(4,1),GAMMA,GTAU(NTAU),S(NTAU,6),TAU(NTAU)
     *     ,ALPHA,C,D,DEL,DENOM,DIVDIF,ENTRY,ENTRY3,FACTOR,FACTR2,GAM
     *     ,ONEMG3,ONEMZT,RATIO,SIXTH,TEMP,X,Z,ZETA,ZT2
      ALPH(X) = AMIN1(1.,ONEMG3/X)
C
C  THERE MUST BE AT LEAST  4  INTERPOLATION POINTS.
      IF (NTAU .GE. 4)                          GO TO 5
      PRINT 600,NTAU
  600 FORMAT(8H NTAU = ,I4,20H. SHOULD BE .GE. 4 .)
                                                GO TO 999

C
CONSTRUCT DELTA TAU AND FIRST AND SECOND (DIVIDED) DIFFERENCES OF DATA
C
    5 NTAUM1 = NTAU - 1
      DO 6 I=1,NTAUM1
         S(I,1) = TAU(I+1) - TAU(I)
         IF (S(I,1) .GT. 0.)                    GO TO 6
         PRINT 610,I,TAU(I),TAU(I+1)
  610    FORMAT(7H POINT ,I3,13H AND THE NEXT,2E15.6,15H ARE DISORDERED)
                                                GO TO 999
    6    S(I+1,4) = (GTAU(I+1)-GTAU(I))/S(I,1)
      DO 7 I=2,NTAUM1
    7    S(I,4) = S(I+1,4) - S(I,4)
C
CONSTRUCT SYSTEM OF EQUATIONS FOR SECOND DERIVATIVES AT  TAU. AT EACH
C  INTERIOR DATA POINT, THERE IS ONE CONTINUITY EQUATION, AT THE FIRST
C  AND THE LAST INTERIOR DATA POINT THERE IS AN ADDITIONAL ONE FOR A
C  TOTAL OF  NTAU  EQUATIONS IN  NTAU  UNKNOWNS.
C
      I = 2
      S(2,2) = S(1,1)/3.
      SIXTH = 1./6.
      METHOD = 2
      GAM = GAMMA
      IF (GAM .LE. 0.)    METHOD = 1
      IF ( GAM .LE. 3.)                         GO TO 9
      METHOD = 3
      GAM = GAM - 3.
    9 ONEMG3 = 1. - GAM/3.
                  ------ LOOP OVER I ------
C
   10 CONTINUE
C           CONSTRUCT Z(I) AND ZETA(I)
      Z = .5
                                                GO TO (19,11,12),METHOD
   11 IF (S(I,4)*S(I+1,4) .LT. 0.)              GO TO 19
   12 TEMP = ABS(S(I+1,4))
      DENOM = ABS(S(I,4)) + TEMP
      IF (DENOM .EQ. 0.)                        GO TO 19
      Z = TEMP/DENOM
      IF (ABS(Z - .5) .LE. SIXTH)  Z = .5
   19 S(I,5) = Z
```

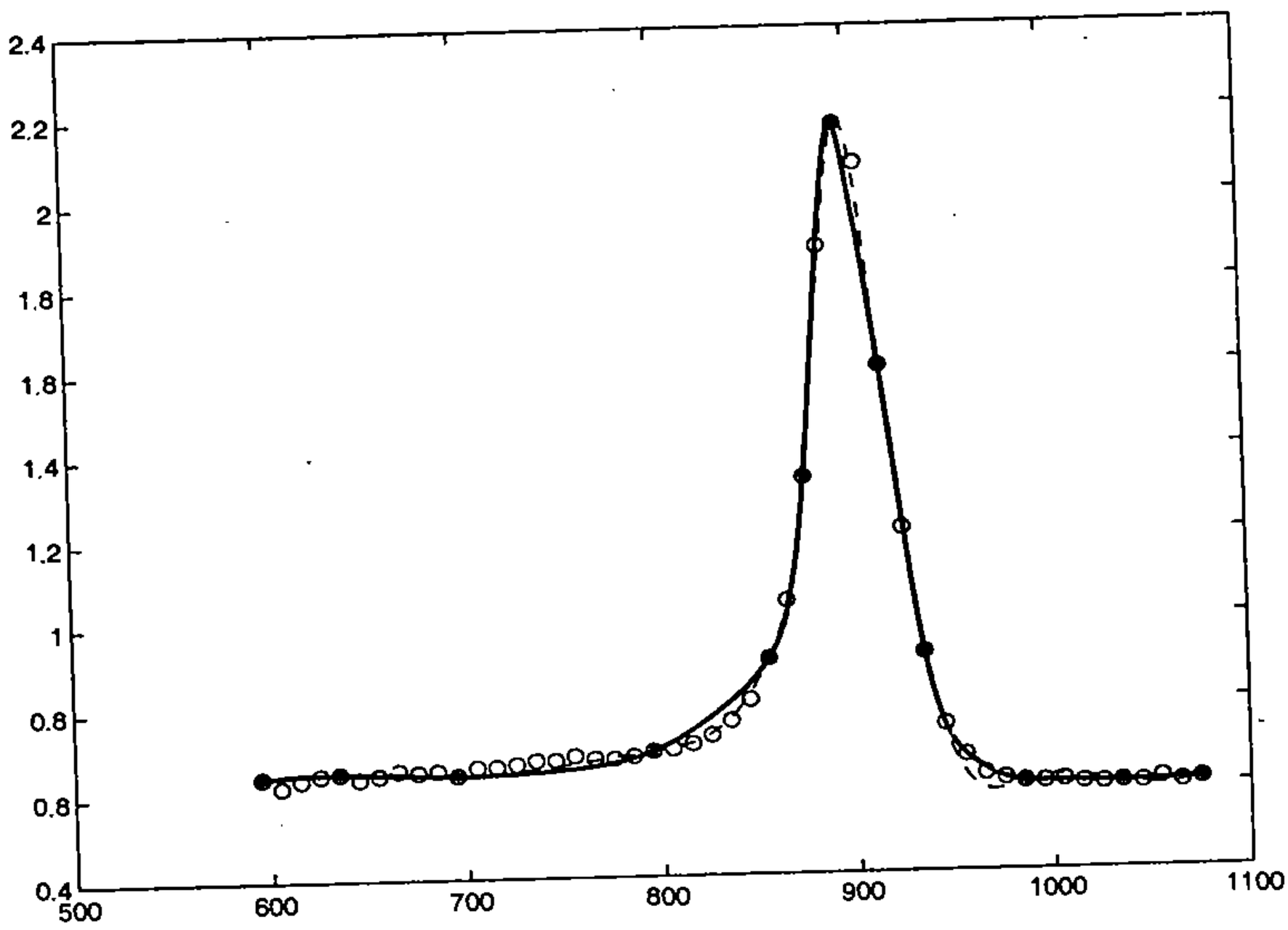
```
C    ******SET UP PART OF THE I-TH EQUATION WHICH DEPENDS ON
C          THE I-TH INTERVAL
      IF (Z - .5)                                21,22,23
   21 ZETA = GAM*Z
      ONEMZT = 1. - ZETA
      ZT2 = ZETA**2
      ALPHA = ALPH(ONEMZT)
      FACTOR = ZETA/(ALPHA*(ZT2-1.) + 1.)
      S(I,6) = ZETA*FACTOR/6.
      S(I,2) = S(I,2) + S(I,1)*((1.-ALPHA*ONEMZT)*FACTOR/2. - S(I,6))
C     IF Z = 0 AND THE PREVIOUS Z = 1, THEN D(I) = 0. SINCE THEN
C     ALSO U(I-1) = L(I+1) = 0, ITS VALUE DOES NOT MATTER. RESET
C     D(I) = 1 TO INSURE NONZERO PIVOT IN ELIMINATION.
      IF (S(I,2) .LE. 0.) S(I,2) = 1.
      S(I,3) = S(I,1)/6.
                                                 GO TO 25
   22 S(I,2) = S(I,2) + S(I,1)/3.
      S(I,3) = S(I,1)/6.
                                                 GO TO 25
   23 ONEMZT = GAM*(1. - Z)
      ZETA = 1. - ONEMZT
      ALPHA = ALPH(ZETA)
      FACTOR = ONEMZT/(1. - ALPHA*ZETA*(1.+ONEMZT))
      S(I,6) = ONEMZT*FACTOR/6.
      S(I,2) = S(I,2) + S(I,1)/3.
      S(I,3) = S(I,6)*S(I,1)
   25 IF (I .GT. 2)                              GO TO 30
      S(1,5) = .5
C    ******THE FIRST TWO EQUATIONS ENFORCE CONTINUITY OF THE FIRST AND OF
C          THE THIRD DERIVATIVE ACROSS TAU(2).
      S(1,2) = S(1,1)/6.
      S(1,3) = S(2,2)
      ENTRY3 = S(2,3)
      IF (Z - .5)                               26,27,28
   26 FACTR2 = ZETA*(ALPHA*(ZT2-1.) + 1.)/(ALPHA*(ZETA*ZT2-1.)+1.)
      RATIO = FACTR2*S(2,1)/S(1,2)
      S(2,2) = FACTR2*S(2,1) + S(1,1)
      S(2,3) = -FACTR2*S(1,1)
                                                 GO TO 29
   27 RATIO = S(2,1)/S(1,2)
      S(2,2) = S(2,1) + S(1,1)
      S(2,3) = -S(1,1)
                                                 GO TO 29
   28 RATIO = S(2,1)/S(1,2)
      S(2,2) = S(2,1) + S(1,1)
      S(2,3) = -S(1,1)*6.*ALPHA*S(2,6)
C       AT THIS POINT, THE FIRST TWO EQUATIONS READ
C             DIAG(1)*X1 + U(1)*X2 + ENTRY3*X3 = R(2)
C       -RATIO*DIAG(1)*X1 + DIAG(2)*X2 + U(2)*X3 = 0.
C       ELIMINATE FIRST UNKNOWN FROM SECOND EQUATION
   29 S(2,2) = RATIO*S(1,3) + S(2,2)
      S(2,3) = RATIO*ENTRY3 + S(2,3)
      S(1,4) = S(2,4)
      S(2,4) = RATIO*S(1,4)
                                                 GO TO 35
   30 CONTINUE
C    ******THE I-TH EQUATION ENFORCES CONTINUITY OF THE FIRST DERIVATIVE
C          ACROSS TAU(I). IT HAS BEEN SET UP IN STATEMENTS 35 UP TO 40
C          AND 21 UP TO 25 AND READS NOW
C           -RATIO*DIAG(I-1)*XI-1 + DIAG(I)*XI + U(I)*XI+1 = R(I) .
C          ELIMINATE (I-1)ST UNKNOWN FROM THIS EQUATION
      S(I,2) = RATIO*S(I-1,3) + S(I,2)
      S(I,4) = RATIO*S(I-1,4) + S(I,4)
C
```

```
C  ******SET UP THE PART OF THE NEXT EQUATION WHICH DEPENDS ON THE
C         I-TH INTERVAL.
   35 IF (Z - .5)                                    36,37,38
   36 RATIO = -S(I,6)*S(I,1)/S(I,2)
      S(I+1,2) = S(I,1)/3.
                                                     GO TO 40
   37 RATIO = -(S(I,1)/6.)/S(I,2)
      S(I+1,2) = S(I,1)/3.
                                                     GO TO 40
   38 RATIO = -(S(I,1)/6.)/S(I,2)
      S(I+1,2) = S(I,1)*((1.-ZETA*ALPHA)*FACTOR/2. - S(I,6))
C          ------  END OF I LOOP ------
   40 I = I+1
      IF (I .LT. NTAUM1)                             GO TO 10
      S(I,5) = .5
C
C          ------  LAST TWO EQUATIONS  ------
C  THE LAST TWO EQUATIONS ENFORCE CONTINUITY OF THIRD DERIVATIVE AND
C  OF FIRST DERIVATIVE ACROSS  TAU(NTAU-1).
      ENTRY = RATIO*S(I-1,3) + S(I,2) + S(I,1)/3.
      S(I+1,2) = S(I,1)/6.
      S(I+1,4) = RATIO*S(I-1,4) + S(I,4)
      IF (Z - .5)                                    41,42,43
   41 RATIO = S(I,1)*6.*S(I-1,6)*ALPHA/S(I-1,2)
      S(I,2) = RATIO*S(I-1,3) + S(I,1) + S(I-1,1)
      S(I,3) = -S(I-1,1)
                                                     GO TO 45
   42 RATIO = S(I,1)/S(I-1,2)
      S(I,2) = RATIO*S(I-1,3) + S(I,1) + S(I-1,1)
      S(I,3) = -S(I-1,1)
                                                     GO TO 45
   43 FACTR2 = ONEMZT*(ALPHA*(ONEMZT**2-1.)+1.)/
     *                (ALPHA*(ONEMZT**3-1.)+1.)
      RATIO = FACTR2*S(I,1)/S(I-1,2)
      S(I,2) = RATIO*S(I-1,3) + FACTR2*S(I-1,1) + S(I,1)
      S(I,3) = -FACTR2*S(I-1,1)
C     AT THIS POINT, THE LAST TWO EQUATIONS READ
C              DIAG(I)*XI + U(I)*XI+1 = R(I)
C       -RATIO*DIAG(I)*XI + DIAG(I+1)*XI+1 = R(I+1)
C     ELIMINATE XI FROM LAST EQUATION
   45 S(I,4) = RATIO*S(I-1,4)
      RATIO = -ENTRY/S(I,2)
      S(I+1,2) = RATIO*S(I,3) + S(I+1,2)
      S(I+1,4) = RATIO*S(I,4) + S(I+1,4)
C
C          ------ BACK SUBSTITUTION ------
C
      S(NTAU,4) = S(NTAU,4)/S(NTAU,2)
   50    S(I,4) = (S(I,4) - S(I,3)*S(I+1,4))/S(I,2)
         I = I - 1
         IF (I .GT. 1)                               GO TO 50
      S(1,4) = (S(1,4)-S(1,3)*S(2,4)-ENTRY3*S(3,4))/S(1,2)
C
C          ------ CONSTRUCT POLYNOMIAL PIECES ------
C
      BREAK(1) = TAU(1)
      L = 1
      DO 70 I=1,NTAUM1
         COEF(1,L) = GTAU(I)
         COEF(3,L) = S(I,4)
         DIVDIF = (GTAU(I+1)-GTAU(I))/S(I,1)
         Z = S(I,5)
         IF (Z - .5)                                 61,62,63
   61    IF (Z .EQ. 0.)                              GO TO 65
```

```
      ZETA = GAM*Z
      ONEMZT = 1. - ZETA
      C = S(I+1,4)/6.
      D = S(I,4)*S(I,6)
      L = L + 1
      DEL = ZETA*S(I,1)
      BREAK(L) = TAU(I) + DEL
      ZT2 = ZETA**2
      ALPHA = ALPH(ONEMZT)
      FACTOR = ONEMZT**2*ALPHA
      COEF(1,L) = GTAU(I) + DIVDIF*DEL
     *          + S(I,1)**2*(D*ONEMZT*(FACTOR-1.)+C*ZETA*(ZT2-1.))
      COEF(2,L) = DIVDIF + S(I,1)*(D*(1.-3.*FACTOR)+C*(3.*ZT2-1.))
      COEF(3,L) = 6.*(D*ALPHA*ONEMZT + C*ZETA)
      COEF(4,L) = 6.*(C - D*ALPHA)/S(I,1)
      COEF(4,L-1) = COEF(4,L) - 6.*D*(1.-ALPHA)/(DEL*ZT2)
      COEF(2,L-1) = COEF(2,L) - DEL*(COEF(3,L) -(DEL/2.)*COEF(4,L-1))
                                          GO TO 68
   62 COEF(2,L) = DIVDIF - S(I,1)*(2.*S(I,4) + S(I+1,4))/6.
      COEF(4,L) = (S(I+1,4)-S(I,4))/S(I,1)
                                          GO TO 68
   63 ONEMZT = GAM*(1. - Z)
      IF (ONEMZT .EQ. 0.)                 GO TO 65
      ZETA = 1. - ONEMZT
      ALPHA = ALPH(ZETA)
      C = S(I+1,4)*S(I,6)
      D = S(I,4)/6.
      DEL = ZETA*S(I,1)
      BREAK(L+1) = TAU(I) + DEL
      COEF(2,L) = DIVDIF - S(I,1)*(2.*D + C)
      COEF(4,L) = 6.*(C*ALPHA - D)/S(I,1)
      L = L + 1
      COEF(4,L) = COEF(4,L-1) + 6.*(1.-ALPHA)*C/(S(I,1)*ONEMZT**3)
      COEF(3,L) = COEF(3,L-1) + DEL*COEF(4,L-1)
      COEF(2,L) = COEF(2,L-1)+DEL*(COEF(3,L-1)+(DEL/2.)*COEF(4,L-1))
      COEF(1,L) = COEF(1,L-1)+DEL*(COEF(2,L-1)+(DEL/2.)*(COEF(3,L-1)
     *                    +(DEL/3.)*COEF(4,L-1)))
                                          GO TO 68
   65 COEF(2,L) = DIVDIF
      COEF(3,L) = 0.
      COEF(4,L) = 0.
   68 L = L + 1
   70 BREAK(L) = TAU(I+1)
    L = L - 1
    K = 4
    IFLAG = 1
                                          RETURN
  999 IFLAG = 2
                                          RETURN
    END
```

END

(17) Example: Taut cubic spline interpolation to Titanium Heat data As an example, we picked the Titanium Heat Data which caused the optimal quartic interpolant in Example XIII(29) to oscillate so wildly. First, we interpolate by a cubic spline to the same data used in Example XIII(29), using the "not-a-knot" end condition and obtain, once again, some oscillations (see Figure (18)). Then, we also construct the taut spline with $\gamma = 2.5$.



(18) FIGURE. The ordinary (dashed) and taut (solid) cubic spline interpolant to some points of the Titanium Heat Data.

The program for this example differs little from that in Example XIII(29). We simply call on TAUTSP, once with GAMMA = 0. and once with GAMMA = 2.5, instead of using SPLINT and BSPLPP. □

Further details about the taut spline can be found in de Boor [1980]. We note that McAllister & Roulier [1978] also use carefully placed additional knots to interpolate correctly to data that are both monotone and convex. We also alert the reader to the paper by G. Nielson [1974] who discusses a surprising alternative to splines in tension. He develops a cubic spline in tension that has only one continuous derivative but produces a curve with continuous tangent *and* continuous curvature when applied to the two components of a curve.

With this, we are back to the discussion of curve approximation, and do now take up the first problem mentioned in the beginning of the chapter.

Proper choice of parametrization A curve may, of course, be parametrized in many ways, that is, for a given curve C_c, there are infinitely many continuous function pairs (d_x, d_y) on $[a\,..\,b]$ for which $C_c = C_d$. We could enlarge the set of possible parametrizations further by allowing the interval $[a\,..\,b]$ to change as well. Hence, we know of the interpolation pair (c_x, c_y) that it should take on certain values $P_i = (x_i, y_i)$, $i = 1, \ldots, N$, but

we have to choose somehow the sites $(s_i)_1^N$ in the parameter interval (and, indeed, the parameter interval $[a\,..\,b]$ itself) at which we would like to have

$$(c_x(s_i), c_y(s_i)) = (x_i, y_i).$$

(19) Example: Choice of parametrization is important We treat (the graph of) the function $f(x) = (x-.3)^2$ as the curve C_c with $c_x(s) = s$, $c_y(s) = f(s)$, and take from it the data points $P_i = c(s_i)$, $i = 1, \ldots, 8$, with

$$(s_i)_1^8 = (0., .1, .2, .3, .301, .4, .5, .6).$$

We choose (i) the parametrization used in the definition of the curve. But we also use a straightforward parametrization that might be, and indeed is frequently, used when one has no idea from what curve the data have come from, namely (ii) the "uniform" parametrization $P_i = c(i)$, $i = 1, \ldots, N = 8$.

In the program below, note that only BANSLV is used and not the entire SPLINT when it comes to interpolating the second component.

```
CHAPTER XVI, EXAMPLE 3. TWO PARAMETRIZATIONS OF SOME DATA
CALLS  SPLINT(BSPLVB,BANFAC/SLV),BSPLPP(BSPLVB*),BANSLV*,PPVALU(INTERV)
C      PARAMETER K=4,KPKM1=7, N=8,NPK=12, NPIECE=6,NPOINT=21
C      INTEGER I,ICOUNT,IFLAG,KP1,L
C      REAL BCOEF(N),BREAK(NPIECE),DS,Q(N,KPKM1),S(N),SCRTCH(K,K)
C     *     ,SS,T(NPK),X(N),XCOEF(K,NPIECE),XX(NPOINT),Y(N)
C     *     ,YCOEF(K,NPIECE),YY(NPOINT)
       INTEGER I,ICOUNT,IFLAG,K,KPKM1,KP1,L,N,NPOINT
       DATA K,KPKM1,N,NPOINT /4,7,8,21/
       REAL BCOEF(8),BREAK(6),DS,Q(8,7),S(8),SCRTCH(4,4)
      *     ,SS,T(12),X(8),XCOEF(4,6),XX(21),Y(8)
      *     ,YCOEF(4,6),YY(21)
       DATA X /0.,.1,.2,.3,.301,.4,.5,.6/
C  *** COMPUTE Y-COMPONENT AND SET 'NATURAL' PARAMETRIZATION
       DO 1 I=1,N
          Y(I) = (X(I)-.3)**2
    1     S(I) = X(I)
       PRINT 601
  601 FORMAT(26H 'NATURAL' PARAMETRIZATION/6X,1HX,11X,1HY)
       ICOUNT = 1
C  *** CONVERT DATA ABSCISSAE TO KNOTS. NOTE THAT SECOND AND SECOND
C      LAST DATA ABSCISSAE ARE NOT KNOTS.
    5  DO 6 I=1,K
          T(I) = S(1)
    6     T(N+I) = S(N)
       KP1 = K+1
       DO 7 I=KP1,N
    7     T(I) = S(I+2-K)
C  *** INTERPOLATE TO X-COMPONENT
       CALL SPLINT(S,X,T,N,K,Q,BCOEF,IFLAG)
       CALL BSPLPP(T,BCOEF,N,K,SCRTCH,BREAK,XCOEF,L)
```

```
C  *** INTERPOLATE TO Y-COMPONENT. SINCE DATA ABSCISSAE AND KNOTS ARE
C      THE SAME FOR BOTH COMPONENTS, WE ONLY NEED TO USE BACKSUBSTITUTION
       DO 10 I=1,N
   10     BCOEF(I) = Y(I)
       CALL BANSLV(Q,KPKM1,N,K-1,K-1,BCOEF)
       CALL BSPLPP(T,BCOEF,N,K,SCRTCH,BREAK,YCOEF,L)
C  *** EVALUATE CURVE AT SOME POINTS NEAR THE POTENTIAL TROUBLE SPOT,
C      THE FOURTH AND FIFTH DATA POINTS.
       SS = S(3)
       DS = (S(6)-S(3))/FLOAT(NPOINT-1)
       DO 20 I=1,NPOINT
          XX(I) = PPVALU(BREAK,XCOEF,L,K,SS,0)
          YY(I) = PPVALU(BREAK,YCOEF,L,K,SS,0)
    20    SS = SS + DS
       PRINT 620,(XX(I),YY(I),I=1,NPOINT)
   620 FORMAT(2F12.7)
       IF (ICOUNT .GE. 2)                          STOP
C  *** NOW REPEAT THE WHOLE PROCESS WITH UNIFORM PARAMETRIZATION
       ICOUNT = ICOUNT + 1
       DO 30 I=1,N
    30    S(I) = FLOAT(I)
       PRINT 630
   630 FORMAT(/26H 'UNIFORM' PARAMETRIZATION/6X,1HX,11X,1HY)
                                                   GO TO 5
       END
```

The data points are approximately uniformly spaced, $(x_i, y_i) = (i/10, ((i-3)/10)^2)$, except that an additional point $(.301, 10^{-6})$ is inserted. Instead of the printed output, we show in Figure (22) the piece of the resulting curves, between the points $(.2, .01)$ and $(.4, .01)$. For the parametrization (i), we expect and get the original curve since both components are polynomials of order 4. For the "uniform" parametrization, we get a kink or loop in the resulting interpolating curve (solid). □

One can, of course, argue that we are only given the data points $(P_i)_1^8$ and nothing else and therefore one way of filling in the curve is as good as any other. True, but if you don't like little kinks in the curve, then you'd better pay attention to the parametrization.

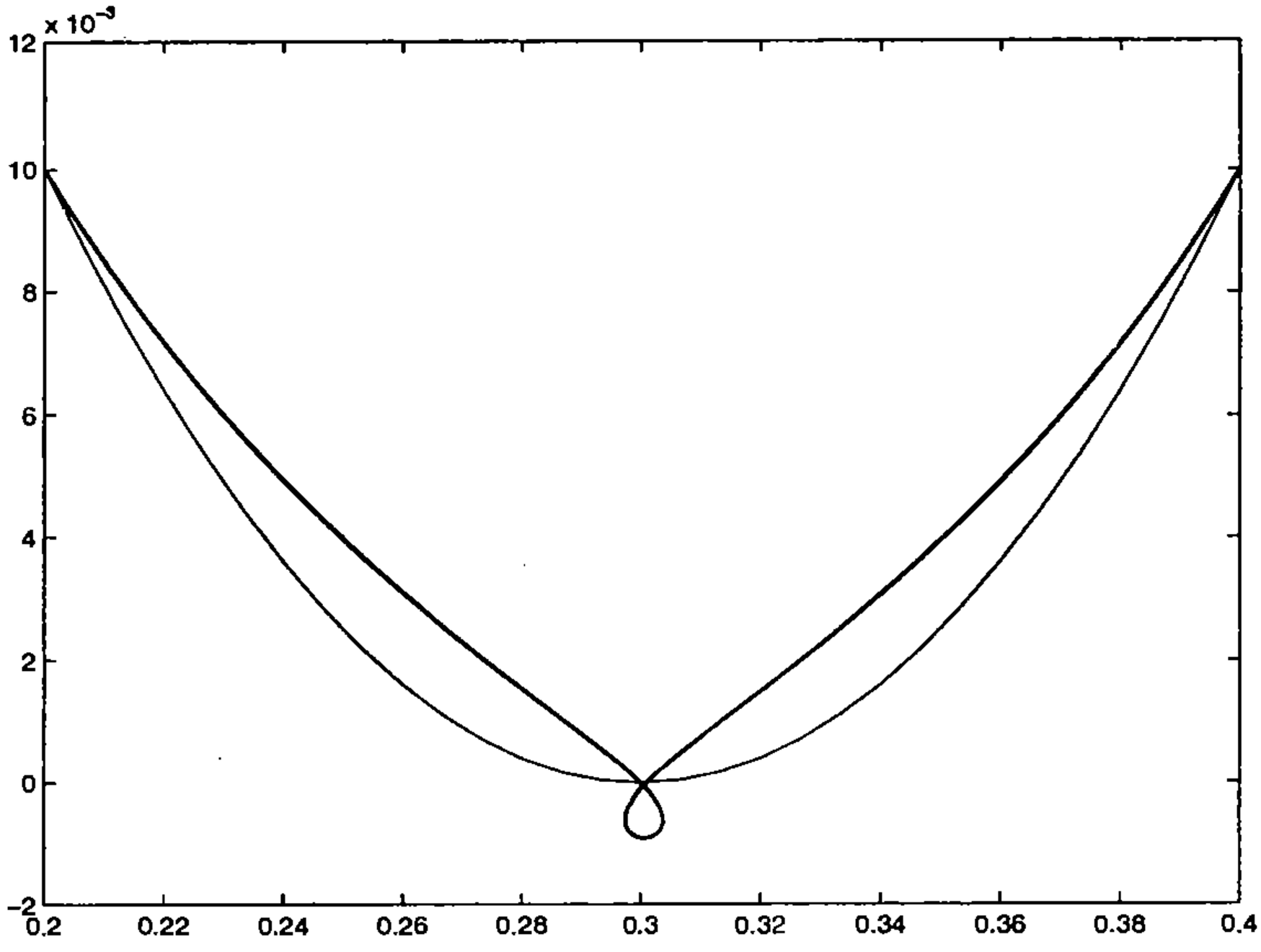
Experience has shown that any parametrization that approximates the *arc length* of the resulting curve is suitable. Although considerable effort and much iteration can be put into a program for making the parametrization exactly equal to the arc length of the resulting curve, it usually suffices to choose

$$(20) \qquad s_1 = 0, \quad s_{i+1} = s_i + \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2}, \qquad i = 1, \ldots, N-1.$$

If the resulting sequence **s** of parameter values is not very uniform, then it is better to use Eugene Lee's [1989] **centripetal** scheme

$$(21) \quad s_1 = 0, \quad s_{i+1} = s_i + \left((\Delta x_i)^2 + (\Delta y_i)^2\right)^{1/4}, \qquad i = 1, \ldots, N-1,$$

as is done in **cscvn** of the SPLINE TOOLBOX (de Boor [1990]$_2$).



(22) FIGURE. Part of a curve (thick) obtained by interpolation to data taken from a parabola (thin) and approximately uniformly spaced (with $\Delta x_i = .1$) except for the pair $(.3, 0)$ and $(.301, 10^{-6})$, but nevertheless uniformly parametrized.

The approximation of a curve In the construction of an approximation to a curve, one is not limited to interpolation processes. Any process of approximation to functions can be applied to the two components of a curve.

This brings up the question of how the approximation might depend on the particular coordinate system in which we chose to measure the curve. Suppose we apply the approximation scheme A that produces the approximation $f = Ag$ from certain information about the function g. Our approximation to the curve $C_c = (c_x, c_y)$ then becomes the curve $C_{Ac} = (Ac_x, Ac_y)$. What if we had used the (u, v)-coordinate system instead, with the transformation $B : (x, y) \mapsto (u, v)$ *affine*, that is,

$$\begin{array}{rcccccc} u = & u_0 & + & b_{ux}x & + & b_{uy}y \\ v = & v_0 & + & b_{vx}x & + & b_{vy}y \end{array}, \qquad \text{all } (x, y)$$

for some fixed point (u_0, v_0) and some invertible matrix $\left[\begin{smallmatrix} b_{ux} & b_{uy} \\ b_{vx} & b_{vy} \end{smallmatrix}\right]$? Ideally, we would like our approximation process for curves to be *invariant* under

such changes of coordinates since the choice of a particular coordinate system is often arbitrary and a curve exists, in some sense, independently from the coordinate system that we happen to use to draw it.

(23) Lemma. *If the approximation process is linear, then the approximation curve C_{Ac} to C_c is invariant under an affine change of coordinates B. Explicitly, then $C_{A(Bc)} = BC_{Ac}$.*

PROOF. We are to prove that it does not matter whether we first go over to the new coordinate system and then approximate the two components or first approximate the two components of a curve and then go over to the new coordinates. In the former case, we look at C_c in the new coordinates, as the curve C_d with $d = Bc$, that is,

$$\begin{aligned} d_u &= u_0 + b_{ux}c_x + b_{uy}c_y, \\ d_v &= v_0 + b_{vx}c_x + b_{vy}c_y, \end{aligned}$$

and then get the approximation C_{Ad}. But then,

$$\begin{aligned} A(d_u) &= u_0 + b_{ux}A(c_x) + b_{uy}A(c_y), \\ A(d_v) &= v_0 + b_{vx}A(c_x) + b_{vy}A(c_y), \end{aligned}$$

by the linearity of A which shows that $C_{Ad} = B(C_{Ac})$. □

Spline interpolation and discrete Least-squares approximation, as discussed in Chapters XIII and XIV, are linear approximation processes, as are the many local schemes discussed in Chapter XII. But taut spline interpolation discussed above is not linear since the location of the additional knots depends nonlinearly on the given data. Taut spline interpolation is homogeneous, that is, $A(\alpha g) = \alpha(Ag)$ for any scalar α. Therefore, a curve approximation scheme based on taut interpolation is, at least, invariant under *scaling*, that is, under coordinate transformations of the form $(x, y) \mapsto (b_u x, b_v y)$.

Nonlinear splines

Any discussion of spline *curves* requires at least a mention of **nonlinear** spline interpolation in which one attempts to solve the original problem of spline interpolation exactly, by the construction of a curve with *minimum curvature* that contains the given points in the given order. Such a discussion was initiated by Even Mehlum in Mehlum [1964], and further elucidated in Mehlum [1974], and has also been taken up by others, e.g., Malcolm [1977], Brunnett [1992], Linner [1996].

In the present context of *planar* curves and in the present notation, the mathematical facts are as follows. Let C_c be a planar curve parametrized by arclength. This means that $C_c' = (c_x', c_y')$ has unit Euclidean length everywhere, that is,

$$|C_c'(s)| := \sqrt{(c_x')(s)^2 + (c_y')(s)^2} = 1, \quad s \in [0\,..\,L],$$

with L the total length of the curve. Then its **(normal) curvature** at the curve point $C_c(s)$ is given by

$$\kappa(s) := |C_c''(s)|, \quad s \in [0\,..\,L],$$

that is, equal to the reciprocal of the radius of the circle that has second-order contact with the curve at that point.

Non-linear spline interpolation seeks to determine such a curve that is **smoothest** in the sense that its strain energy, that is, the integral

$$\int_0^L \kappa(s)^2\,\mathrm{d}s, \tag{24}$$

is as small as possible among all curves that pass through the given sequence $P_1, \ldots, P_N$ of points in $\mathbb{R}^2$ in that given order, starting at P_1 and ending at P_N, and satisfy an additional condition, like taking on a prescribed tangent, at these two endpoints.

A first difficulty with non-linear spline interpolation concerns the length of the curve, that is, the choice of L. If we leave L as one of the parameters to be determined by minimization of (24), then, as already pointed out in Birkhoff & de Boor [1965], it is possible to make (24) as small as one pleases, by the simple device of allowing the curve to form huge near-circular loops between successive points. This device works because the curve given by the rule

$$C(s) := \mathrm{o} + R(\cos(s/R), \sin(s/R)), \quad s \in [0\,..\,2\pi R],$$

has constant curvature, $1/R$, (evident from the fact that it is just a circle of radius R), and length $L = 2\pi R$, hence (24) evaluates to $2\pi R(1/R)^2 = 2\pi/R$, and this goes to zero as $R \to \infty$. One therefore looks, more precisely, for interpolating curves that minimize (24) either for a fixed length, L, or with L bounded by some given constant. In the latter case, one then looks, more precisely, for 'critical' values for L.

Between each pair P_i, P_{i+1}, a minimizing C_c can be shown to have to satisfy the differential equation

$$-C_c''' + \lambda C_c' = \mathbf{a},$$

for some constant λ and some constant vector $\mathbf{a}$, and this leads to an explicit description of that segment of C_c in terms of elliptic integrals that depend on λ and $\mathbf{a}$ and on constants of integration, and requires the determination of all these constants from the requirement that the curve segment connect P_i with P_{i+1} in such a way that a smooth curve results when these various segments are concatenated.

The above-cited literature details the resulting difficulties and how they might be overcome in various ways, for example by approximating, as does Mehlum, the exact solution by a sequence of properly chosen circular arcs. With this, we are back to the topic of approximating a given curve.

Periodic splines If the curve C_c to be approximated is *closed*, then its component functions c_x, c_y are periodic and it makes sense to approximate them by periodic splines. *Periodic* splines of order k on some interval $[a\,..\,b]$ differ from other such splines only in that they satisfy the periodic boundary conditions

$$f^{(j)}(a^+) = f^{(j)}(b^-), \qquad j = 0, \ldots, k-1.$$

In contrast to other boundary conditions discussed earlier (in Chapter IV and XIII), periodic boundary conditions are not separated. The typical linear system for the construction of periodic splines therefore fails to be banded; the first few unknowns will also appear in the last few equations. Still, it is possible to modify programs such as BANFAC/SLV in XII and BCHFAC/SLV in XIV appropriately to cope with this fact efficiently. (See also the discussion of periodic splines with uniform knot sequences below.)

Periodicity can be built directly into the B-spline basis. If we need a basis for

$$\mathring{\mathrm{S}} := \{\, f \in \Pi_{<k,\xi,\nu} : D^j f(\xi_1^+) = D^j f(\xi_{l+1}^-),\ j = 0, \ldots, \nu_1 - 1 \,\}$$

for some positive integer ν_1, then we would modify the construction of the knot sequence, as given in IX(50), as follows. We choose ξ_1 and ξ_{l+1} to be knots of multiplicity $k - \nu_1$ only and obtain the remaining end knots by a periodic extension of the interior knots. Precisely, we would set

$$t_{\nu_1+1} = \cdots = t_k = \xi_1$$

and would then determine the remaining end knots by

$$t_i = t_{n-\nu_1+i} - (\xi_{l+1} - \xi_1), \qquad \text{all } i.$$

In this way, we obtain a basis for $\Pi_{<k,\xi,\nu}$ in which the $(\xi_{l+1} - \xi_1)$-periodicity is mirrored by the $(n - \nu_1)$-periodicity in the B-spline coefficients:

$$f = \sum \alpha_i \mathrm{B}_i \in \mathring{\mathrm{S}} \quad \Longleftrightarrow \quad \alpha_i = \alpha_{n-\nu_1+i}, \quad i = 1, \ldots, \nu_1.$$

In many situations, though, periodicity can be handled to sufficient accuracy merely by extending the data periodically and approximating it on an interval somewhat larger than the interval of periodicity. This is obviously true in case the approximation process is *local*. See Chapter XII for some examples. But this is also true in case the approximation process is **essentially local**. This imprecise term is meant to describe the situation where the effect on the approximation Ag of a change in the function g on some "small" interval $[\alpha\,..\,\beta]$ becomes less and less noticeable as one moves away from the interval $[\alpha\,..\,\beta]$. We discussed an instance in Chapter VI and brought out the importance of this notion in connection with knot

placement algorithms in Chapter XII. A typical example is cubic spline interpolation at knots to data at uniformly spaced sites $\tau_i = \tau_0 + ih$, all i. If one writes the interpolant in Lagrange form,

$$Ag = \sum g(\tau_i)C_i,$$

then the function C_i is a cubic spline with $C_i(\tau_j) = \delta_{ij}$, all j, and it tells us explicitly how a change in the datum $g(\tau_i)$ affects the approximation Ag. It can be shown (see Birkhoff & de Boor [1964]) that, in this case, $|C_i(x)| \leq \lambda^{-j}$ for $x \notin [\tau_{i-j} \,..\, \tau_{i+j}]$, with $\lambda = 2 + \sqrt{3} = 3.732\ldots$ (see Problem VI.6).

Cardinal splines It often happens that the approximation problem to be solved can be handled well by splines with equally spaced knots. Such splines have been called **cardinal splines** by Schoenberg, and many facts about them have been gathered in Schoenberg's [1973] beautiful monograph. Because of their simple knot structure, such splines can be studied in much more precise detail and there are many computational advantages connected with their use.

The advantages of cardinal splines over others are typified by the fact that, for cardinal splines, there is essentially only *one* B-spline of a given order. All others of the same order are (dilated) translates of this one. Explicitly, let Q_k denote the B-spline of order k with knots $0, 1, \ldots, k$, that is,

$$Q_k(x) := k[0, \ldots, k](\cdot - x)_+^{k-1}. \tag{25}$$

Then, for the knot sequence $\mathbf{t} = (t_0 + ih)_{i=-\infty}^{\infty}$,

$$\mathrm{B}_{i,k,\mathbf{t}}(x) = Q_k((x - t_i)/h), \qquad \text{all } i. \tag{26}$$

Among the consequences of this fact are the following:
(i) Conversion from B-form to ppform is much cheaper than in the general case. If $f = \sum \alpha_i Q_k((\cdot - t_i)/h)$, then

$$D^r f(t_s) = \sum \alpha_i D^r Q_k((t_s - t_i)/h)/h^r,$$

hence

$$h^r D^r f(t_s) = \sum_{i=0}^{k-1} \alpha_{s-i} D^r Q_k(i), \qquad r = 0, \ldots, k-1.$$

The matrix $(D^r Q_k(i))_{i,r=0}^{k-1}$ is constructed once, and then applied to the appropriate section of the B-coefficient vector $\boldsymbol{\alpha}$ to produce the polynomial coefficients.

(27) Example: Conversion to ppform is cheaper when knots are uniform Computer aided design people are using Schoenberg's variation diminishing spline approximation XI(33), typically with cubic splines, and with a uniformly spaced data sequence. If the resulting function

$$f = \sum_i g(t_i)\mathrm{B}_{i,4}$$

is to be evaluated extensively (as it would if it were to be graphed), then it pays to convert to ppform. If the spacing of the data sites is $t_i = t_0 + ih$, all i, then

$$\begin{bmatrix} f(t_s) \\ hf'(t_s) \\ h^2 f''(t_s) \\ h^3 f'''(t_s) \end{bmatrix} = \begin{bmatrix} 1/6 & 2/3 & 1/6 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1 & -2 & 1 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} g(t_{s-3}) \\ g(t_{s-2}) \\ g(t_{s-1}) \\ g(t_s) \end{bmatrix}.$$

□

For this and other values of k, the matrix $(D^r Q_k(i))_{i,r=0}^{k-1}$ can be obtained directly from BSPLVD. If $\mathtt{T}(i) = i$, $i = 1, \ldots, 2k$, then

```
CALL BSPLVD (T, k, T(k), k, SCRTCH, DBIATX, k)
```

produces the array DBIATX of order (k, k), with

$$\mathtt{DBIATX}(i+1, k-r) = D^r Q_k(i), \qquad i = r = 0, \ldots, k-1.$$

(ii) In typical approximation problems, the coefficient matrix is (almost) Toeplitz. If not only the knots, but also the interpolation conditions are uniform, then the linear system for the B-coefficients of the approximation is a **Toeplitz matrix**, that is, is of the form

$$(c_{i-j})_{i,j}$$

for some fixed sequence (c_i). It is possible to give explicit formulæ for the inverse of such a matrix, but this has no practical application. The chief benefit in computations lies in the fact that such matrices are easy to generate. Also, the matrices are usually only almost Toeplitz since the end conditions usually spoil things.

(28) Example: Complete cubic spline interpolation at uniformly spaced sites $t_4, \ldots, t_{n+1}$ has as its linear system for the B-coefficients of the interpolant the following:

$$\begin{array}{rcl} -\alpha_1 + \alpha_3 & = & 2hg'(t_4) \\ \alpha_{i-1} + 4\alpha_i + \alpha_{i+1} & = & 6g(t_{i+2}), \qquad i = 2, \ldots, n-1, \\ -\alpha_{n-2} + \alpha_n & = & 2hg'(t_{n+1}) \end{array}$$

assuming that $t_i = t_0 + ih$, all i (see Problem IV.5). The matrix is therefore Toeplitz except for the first and last row.

In Schoenberg's [1973] investigations of cardinal splines over a biinfinite knot sequence, the various matrices that occur are biinfinite Toeplitz matrices. The fact that their effect can be easily analyzed with the aid of the Fourier transform is at the basis of Schoenberg's successful and elegant analysis. It is possible to recapture the spirit of such analysis in the practically important case of

Periodic splines on uniform meshes We take a biinfinite uniform knot sequence $\mathbf{t} = (t_i)$ with $t_i = t_0 + ih$, all i, and

$$nh = 1,$$

and consider all 1-periodic splines of order k with knot sequence $\mathbf{t}$. We denote their collection by

$$\mathring{\mathbb{S}}.$$

If $f \in \mathbb{S}_{k,t}$, then $f \in \mathring{\mathbb{S}}$ if and only if its B-coefficient sequence is n-periodic, that is,

$$f = \sum_i \alpha_i \mathrm{B}_{i,k,\mathbf{t}} \qquad \text{with } \alpha_{i+n} = \alpha_i, \text{ all } i.$$

This shows $\mathring{\mathbb{S}}$ to be a linear space of dimension n. □

For a given 1-periodic function g, we construct an approximation Ag from $\mathring{\mathbb{S}}$ by the requirement that

$$\mu_i Ag = \mu_i g, \qquad \text{all } i, \tag{29}$$

with

$$\mu_i g = \mu g(\cdot + ih), \qquad \text{all } i. \tag{30}$$

For instance, if we choose $\mu g = g(a)$ for some fixed site a, then $\mu_i g = g(a+ih)$, and our approximation process is nothing more than interpolation at the n equispaced sites $a + h, a + 2h, \ldots, a + nh$. Again, least-squares approximation to a 1-periodic function g from $\mathring{\mathbb{S}}$ fits into this pattern since the approximation Ag is then characterized by the fact (see, e.g., Lemma XIV(16)) that

$$\int \mathrm{B}_i(x)(Ag)(x)\,\mathrm{d}x = \int \mathrm{B}_i(x)g(x)\,\mathrm{d}x, \qquad i = 1, \ldots, n,$$

and this is of the form (29) with

$$\mu g = \int \mathrm{B}_0(x) g(x)\,\mathrm{d}x,$$

because of the uniform knot sequence.

The corresponding linear system for the B-coefficients α of $Ag \in \mathring{\mathrm{S}}$ has the form

$$\sum_j (\mu_i \mathrm{B}_j)\alpha_j = \mu_i g, \qquad \text{all } i.$$

But, because of (30) and since $\mathrm{B}_j(x) = \mathrm{B}_0(x - jh)$, we have

$$\mu_i \mathrm{B}_j = \mu \mathrm{B}_j(\cdot + ih) = \mu \mathrm{B}_0(\cdot + (i-j)h) =: b_{i-j}.$$

The linear system for the coefficients therefore reads

$$\sum_j b_{i-j}\alpha_j = \mu_i g, \qquad \text{all } i. \tag{31}$$

We now bring the fact into play that $Ag \in \mathring{\mathrm{S}}$, that is, $\alpha_i = \alpha_{i+n}$, all i. Then (31) reduces to the nth order linear system

$$\sum_{j=1}^{n} c_{ij}\alpha_j = \mu_i g, \qquad i = 1, \ldots, n, \tag{32}$$

with

$$c_{ij} := \sum_r b_{i-j-rn}, \qquad i, j = 1, \ldots, n. \tag{33}$$

In the typical situation, μ has its support in the open interval $(t_0 \,..\, t_k)$ and then $b_i = \mu_i \mathrm{B}_0 = \mu \mathrm{B}_0(\cdot + ih) = 0$ unless $(t_0 \,..\, t_k) \cap (t_i \,..\, t_{i+k}) \neq \emptyset$, that is, unless $|i| < k$. If then also $2k - 1 \leq n$, then each of the sums in (33) has at most one nonzero entry.

The coefficient matrix $C := (\mathrm{c}_{ij})$ in (32) is a **circulant**, that is,

$$c_{ij} = c_{rs} \quad \text{if } i - j = r - s (\mathrm{mod}\ n).$$

This means that the entries of the ith row of C are obtained from the $(i-1)$st row by shifting all entries one to the right and putting the last entry of row $i-1$, forced out now, back as the first entry of row i; hence the name "circulant".

The theory of circulant matrices parallels that of Toeplitz matrices (which is not too surprising in view of (33)). Consider the vector

$$\mathbf{v}^{(r)} := (\omega^{rs})_{s=1}^{n}, \quad \text{with } \omega := \mathrm{e}^{2\pi \mathrm{i}/n} \text{ (recall that } \mathrm{i} := \sqrt{-1}).$$

Then

$$(C\mathbf{v}^{(r)})_j = \sum_s c_{js}\omega^{rs} = p(\omega^r)\omega^{rj} = p(\omega^r)(\mathbf{v}^{(r)})_j$$

with

$$p(x) := \sum_{j=1}^{n} c_{1j}x^{j-1}$$

the associated polynomial for C. This shows the basis $(\mathbf{v}^{(r)})_{r=1}^n$ for $\mathbb{C}^n$ (as a vector space over the complex scalars, $\mathbb{C}$) to consist entirely of eigenvectors for C, with corresponding eigenvalue sequence $(p(\omega^r))_{r=1}^n$. In other words, if we go over to the basis $(\mathbf{v}^{(r)})_1^n$ for $\mathbb{C}^n$, **then** *every* circulant goes over to a diagonal matrix, with its diagonal entries equal to the values of its associated polynomial at the nth roots of unity.

In particular, if C is invertible, that is, if $p(\omega^r) \neq 0$ for $r = 1, \ldots, n$, then (see Problem 5) its inverse is also a circulant, and if q is the associated polynomial of its inverse, then $q(\omega^r) = 1/p(\omega^r)$, $r = 1, \ldots, n$.

This suggests the following way of solving (32):

(i) Express the right side $\boldsymbol{\beta} := (\mu_i g)_1^n$ in terms of the basis $(\mathbf{v}^{(r)})_1^n$, that is, convert $\boldsymbol{\beta}$ into $V^H\boldsymbol{\beta}/n$, with $V := (v_s^{(r)})_{s,r=1}^n$ and V^H the conjugate transpose of V.

(ii) "Divide" by $D_p := [p(\omega^1), \ldots, p(\omega^n)]$, that is, construct the vector $D_p^{-1}V^H\boldsymbol{\beta}/n$.

(iii) Convert back to standard coordinates, that is, premultiply by $V = (V^H/n)^{-1}$, to get the solution

$$\boldsymbol{\alpha} = VD_p^{-1}V^H(\mu_i g)/n \tag{34}$$

of (32).

Note that step (i) corresponds to constructing a discrete Fourier transform of the vector $\boldsymbol{\beta}$, and step (iii) corresponds to constructing its inverse. For appropriate values of n, the Fast Fourier Transform (see for example, Van Loan [1992]) is therefore applicable in this process.

(35) Example: Periodic spline interpolation to uniformly spaced data and harmonic analysis We choose $\mu g = g(a)$, with $a := t_0 + kh/2$. Our approximation process then becomes periodic spline interpolation, either at the knots if k is even, or halfway between the knots if k is odd. For $n \geq 2k-1$, we can write $p(\omega^r)$ in the form

$$p(\omega^r) = \sum_{j\in\mathbb{Z}} \mathrm{B}_0(a - jh)(\omega^r)^j,$$

making use of the fact that $\omega^{n-r} = 1/\omega^r$. But since $\mathrm{B}_0(t_0 + yh) = Q_k(y)$ (see (26)) and Q_k is even around $k/2$, we get the formula

$$p(\omega^r) = \sum_{|j|<k/2} Q_k(k/2 - j)\cos(2\pi rj/n), \qquad r = 1, \ldots, n.$$

These cosine polynomials already play a basic role in Schoenberg's fundamental paper Schoenberg [1946].

Since $\mu_i g = g(a + jh)$, all j, we compute explicitly in step (i) the customary approximation to the discrete Fourier transform for the 1-periodic function g. For, then

$$V^H \beta/n = \left(\frac{1}{n} \sum_{j=1}^{n} g(a + jh) e^{2\pi i r j/n} \right)_{r=1}^{n}.$$

In this connection, it is interesting to note that the first substantial application of spline functions was carried out by Eagle [1928] in his work on numerical harmonic analysis. He recognized, as others had before him, that the number

$$\hat{c}_r(g) := \frac{1}{n} \sum_{j=1}^{n} g(jh) e^{-2\pi i r j/n}$$

is a poor approximation to the Fourier coefficient

$$c_r(g) := \int_0^1 g(x) e^{-2\pi i r x} \, dx$$

for $r \geq n$. For instance, $\hat{c}_{n+r}(g) = \hat{c}_r(g)$ for all r, while, for a 1-periodic function g with k continuous derivatives, $c_r(g) = \mathcal{O}\left(r^{-k}\right)$ as r becomes large. Eagle proposed as a remedy to approximate $c_r(g)$ by $c_r(Ag)$ instead, with Ag the periodic spline interpolant of the present example. (Eagle called these functions "lath functions".) He found that $c_r(Ag)$ could be obtained from $\hat{c}_r(g)$ in the simple form

$$c_r(Ag) = \tau_r \hat{c}_r(g), \tag{36}$$

with the so-called **attenuation factor** τ_r given by

$$\tau_r := \begin{cases} 1 / \sum_{\upsilon=-\infty}^{\infty} (r/(r + \upsilon n))^k & , \text{for } r \neq 0 (\text{mod } n), \\ 0 & , \text{for } r = 0 (\text{mod } n). \end{cases} \tag{37}$$

The fact that the discrete Fourier transform of the interpolating spline is obtainable so simply from the "cheap" approximation $\hat{c}_r$ is due to the symmetry of the situation. We can write the interpolant in Lagrange form,

$$Ag =: \sum_{j=1}^{n} g(a + jh) C_j.$$

But the "Lagrange" splines C_j are translates of each other, therefore

$$Ag = \sum_{j=1}^{n} g(a + jh) C(\cdot - jh)$$

for the fixed function $C \in \mathring{\mathrm{S}}$ with $C(a+jh) = \delta_{0j}$, all j. Consequently,

$$\begin{aligned} \mathrm{c}_r(Ag) &= \sum_j g(a+jh) \int_0^1 C(x-jh)\mathrm{e}^{-2\pi \mathrm{i} r x}\,\mathrm{d}x \\ &= \sum_j g(a+jh)\mathrm{e}^{-2\pi \mathrm{i} r j h} \int_0^1 C(x-jh)\mathrm{e}^{-2\pi \mathrm{i} r (x-jh)}\,\mathrm{d}x \\ &= n\hat{c}_r(g) \int_0^1 C(x)\mathrm{e}^{-2\pi \mathrm{i} r x}\,\mathrm{d}x. \end{aligned}$$

This shows that

$$\tau_r = n \int_0^1 C(x)\mathrm{e}^{-2\pi \mathrm{i} r x}\,\mathrm{d}x.$$

For further information about attenuation factors (and a proof of (36)) see the detailed paper by W. Gautschi [1972]$_1$. For another substantial early paper on splines, also on discrete harmonic analysis, see Quade & Collatz [1938].

A final word. To be sure, I do not advocate using the discrete Fourier transform (fast or otherwise) merely to construct an approximant such as the periodic spline interpolant of Example (35). Aside from the restrictions on n placed by the FFT, the bandedness of the linear system to be solved makes it cheaper to solve the linear system directly. But, for theoretical investigations, for the derivation of asymptotic error expansions and for practical harmonic analysis, the connection between periodic splines on a uniform knot sequence and circulants, hence with the discrete Fourier transform, is of prime importance.

Problems

1. Use TAUTSP to interpolate the convex data

0	1	2	3	4	5
0	0	.001	10.	21.	33.

Can you make the interpolant convex through proper choice of γ?

2. Use TAUTSP to construct an interpolant f to the cubic polynomial $g(x) = x(x^2-1)$ at -3, -2, -1, 0, 1, 2, 3. How badly does TAUTSP fail to reproduce g? Is $f \in C^{(2)}$? Pay special attention to the interval $[-1\,..\,1]$. Show numerically that, nevertheless, $\|g-f\| = \mathcal{O}\left(n^{-4}\right)$ if we interpolate g on $[-3\,..\,3]$ at $n+1$ equally spaced sites using TAUTSP.

3. Use spline interpolation to $g(x) = \sin x$ over $[-\pi\,..\,3\pi]$ on 21 sites (using, for example, the not-a-knot end condition). Then check to what an extent the interpolant on $[0\,..\,2\pi]$ is 2π-periodic.

4. Supply numerical evidence for the fact that least-squares approxima-

tion by splines is essentially local.

5. Prove that the nth order matrix A is a circulant if and only if $A\mathbf{v}^{(r)} = \alpha_r \mathbf{v}^{(r)}$, $r = 1, \ldots, n$, for some vector $\boldsymbol{\alpha} = (\alpha_r)_1^n$ and with $\mathbf{v}^{(r)} := (\omega^{rj})_{j=1}^n$, $\omega := \mathrm{e}^{2\pi \mathrm{i}/n}$. Conclude that the inverse of a circulant is a circulant. (Hint: Show that VDV^H is a circulant in case D is diagonal.)

6. Prove that $\int Q_k(x + \frac{k}{2}) \mathrm{e}^{-\mathrm{i}xt}\, \mathrm{d}x = (\frac{\sin t/2}{t/2})^k$. (Hint: Use IX(6).)

7. Prove that $\int Q_r(x-y) Q_s(y)\, \mathrm{d}y = Q_{r+s}(x)$. Conclude that the equations for least-squares approximation by splines of order k and those for interpolation at knots by splines of order $2k$ have the same coefficient matrix in case of uniform knots.

XVII

Surface Approximation by Tensor Products

In this chapter, we take a slightly more abstract view of linear approximation methods in order to show how to generalize them to functions of several variables by the tensor product construction.

Applicability of tensor product methods is limited, but when their use can be justified, then these methods should be used since they are extremely efficient compared to other surface approximation techniques.

An abstract linear interpolation scheme Most of the linear approximation schemes discussed in the preceding chapters (such as interpolation, least-squares approximation, even collocation to a linear differential equation) fit into the following abstract framework. For the given function g, we attempt to construct an approximation Pg in the form $\sum_{j=1}^{n} \alpha_j f_j$, with $f_1, \ldots, f_n$ certain fixed functions, on the same domain X on which g is defined. We construct the specific approximation by *interpolation with respect to linear functionals*, that is, by the requirement that

$$\lambda_i Pg = \lambda_i g, \qquad i = 1, \ldots, n, \tag{1}$$

for certain fixed *linear functionals* $\lambda_1, \ldots, \lambda_n$, the **interpolation functionals**.

For instance, we might choose $f_i = \mathrm{B}_{i,k,\mathbf{t}}$, $i = 1, \ldots, n$ (to stick to the spirit of this particular book) and choose $\lambda_i g = g(\tau_i)$, $i = 1, \ldots, n$ for some $\tau_1 < \cdots < \tau_n$. This is ordinary interpolation. Or, we might take $\lambda_i g = [\tau_1, \ldots, \tau_i]g$, $i = 1, \ldots, n$, allowing for osculatory interpolation. Again, we might use $\lambda_i g = \int_{\tau_i}^{\tau_{i+1}} g(x)\,\mathrm{d}x$, as in the area matching approximation scheme in Chapter VIII. The reader should be able to supply many more examples (see Problems 2–4).

The approximation problem does not depend on the *individual* functions $f_1, \ldots, f_n$ nor on the individual interpolation functionals $\lambda_1, \ldots, \lambda_n$, but only on the linear span

$$F := \operatorname{span}(f_j)_1^n := \{ \sum_{j=1}^{n} \alpha_j f_j : \boldsymbol{\alpha} \in \mathbb{R}^n \}$$

of the f_j's and the linear span

$$\Lambda := \mathrm{span}(\lambda_i)_1^n := \{ \sum_{i=1}^{n} \alpha_i \lambda_i : \boldsymbol{\alpha} \in \mathbb{R}^n \}$$

of the interpolation functionals. This is quite clear for the f_j's since we require the approximation quite explicitly to be of the form $\sum \alpha_j f_j$, that is, to be in F, a statement that makes no reference to the individual f_j's any more. But, also

$$\lambda_i g = \lambda_i h, \quad i = 1, \ldots, n \quad \text{if and only if}$$

$$\sum_i \alpha_i \lambda_i g = \sum_i \alpha_i \lambda_i h \quad \text{for all } \boldsymbol{\alpha} \in \mathbb{R}^n.$$

We can therefore phrase our **linear interpolation problem** or **LIP** as follows: *to find, for given g, an $f \in F$ so that*

$$\lambda f = \lambda g \qquad \textit{for all} \quad \lambda \in \Lambda.$$

We say that the LIP given by F and Λ is **correct** if it has exactly one solution for every $g \in U$. Here, U is some nebulous linear space of functions, all defined on the same domain X, and U contains F. Linearity, to recall, requires that the sum of any two functions in U and any scalar multiple of a function in U be also in U. In the preceding chapters, we have dealt with the linear space $U = C(X)$ of functions continuous on the set $X = [a\,..\,b]$. We have also considered the linear space $U = C^{(k)}(X)$ of all functions on $X = [a\,..\,b]$ with k continuous derivatives on the interval $[a\,..\,b]$. In a moment, we will deal with the linear space

$$C^{(k,k)}(X \times Y)$$

of all functions f of two variables on some rectangle $X \times Y = [a..b] \times [c..d]$ that have all partial derivatives $\partial^{i+j} f / \partial x^i \partial y^j$ with $i, j \leq k$ continuous.

Here is the basic lemma concerning correctness and numerical solution of the LIP.

Lemma(2). *Let $(f_j)_1^m$ be a basis for F and let $(\lambda_i)_1^n$ be a basis for Λ. Then:*

(i) The LIP given by F and Λ is correct if and only if the **Gramian matrix**

$$A := (\lambda_i f_j)_{i=1;\,j=1}^{n;\,m} \tag{3}$$

is invertible. In particular, $n = m$ is a necessary (but not a sufficient) condition for the correctness of the LIP.

(ii) If the LIP given by F and Λ is correct, then the interpolant $Pg \in F$ can be constructed as

$$Pg = \sum_{j=1}^{n} \alpha_j f_j, \qquad \text{with } \boldsymbol{\alpha} = A^{-1}(\lambda_i g). \tag{4}$$

PROOF. Since $(\lambda_i)_1^n$ is a basis for Λ, we have that $f = \sum_j \alpha_j f_j$ is a solution of the LIP for g if and only if

$$\lambda_i(\sum_j \alpha_j f_j) = \lambda_i g, \qquad i = 1, \ldots, n,$$

that is, if and only if $A\alpha = (\lambda_i g)_1^n$. Invertibility of the Gramian matrix A therefore guarantees exactly one solution vector $\boldsymbol{\alpha}$, hence exactly one solution $f \in F$, and also provides the formula (4).

Assume, for the converse, that A is not invertible. If $m \geq n$, this means the existence of a nonzero vector $\boldsymbol{\alpha}$ so that $A\boldsymbol{\alpha} = \mathbf{0}$. But then, since $(f_j)_1^m$ is a *basis*, it follows that the function $f := \sum_j \alpha_j f_j$ in F is different from the zero function, yet interpolates to the zero function, that is, the LIP with $g = 0$ has more than one solution. If $m \leq n$, then there exists a nonzero vector $\boldsymbol{\alpha}$ so that $\boldsymbol{\alpha}^T A = \mathbf{0}^T$. But then, $(\lambda_i)_1^n$ being a basis for Λ, $\lambda := \sum_i \alpha_i \lambda_i$ vanishes at every f_j, hence we have $\lambda f = 0$ for every $f \in F$. On the other hand, λ being nonzero, there must be some function $g \in U$ for which $\lambda g \neq 0$. The LIP for this function g then fails to have any solutions since $\lambda g \neq 0 = \lambda f$ for all choices $f \in F$. □

Note that all the action takes place in the interplay between F and Λ, that is, the correctness of the LIP depends only on F and Λ and does not depend on the particular function space U in which F lies and to whose elements g we wish to interpolate. Once we have established the LIP to be correct, then we only need to have a way of evaluating the interpolation functionals $\lambda_1, \ldots, \lambda_n$ on a function g in order to interpolate it from F.

Tensor product of two linear spaces of functions The tensor product of two (or more) algebraic structures is a well understood construct in abstract algebra. But since we only need a few notions concerning the tensor product of two linear spaces of functions, we give now a short discussion of this special case in order to spare the reader an excursion into an abstract algebra text.

Let U be a linear space of functions, all defined on some set X into the reals, and let V be, similarly, a linear space of functions defined on some set Y into $\mathbb{R}$. For each $u \in U$ and $v \in V$, the rule

$$w(x, y) := u(x)v(y), \qquad (x, y) \in X \times Y,$$

defines a function on $X \times Y$ called the **tensor product of u with v** and denoted by

$$u \otimes v.$$

Further, the set of all finite linear combinations of functions on $X \times Y$ of the form $u \otimes v$ for some $u \in U$ and some $v \in V$ is called the **tensor product of U with V** and is denoted by $U \otimes V$. Thus,

$$U \otimes V := \{ \sum_{i=1}^{n} \alpha_i (u_i \otimes v_i) : \alpha_i \in \mathbb{R}, u_i \in U, v_i \in V, \quad i = 1, \ldots, n; \text{ some } n \},$$

and $U \otimes V$ is a linear space (of functions on $X \times Y$).

A simple and important example is furnished by polynomials in two variables: Taking $U = \Pi_{<h}$, the linear space of polynomials of order h, as functions on $X = \mathbb{R}$, and similarly $V = \Pi_{<k}$ as functions on $Y = \mathbb{R}$, we easily recognize $U \otimes V$ as the linear space $\Pi_{<h,k}$ of all polynomials in two variables of degree $< h$ in the first variable and of degree $< k$ in the second variable, considered as functions on the plane $X \times Y = \mathbb{R}^2$. A second simple example arises with the choice $U = \mathbb{R}^m$, the linear space of real m-vectors considered as functions on $X := \{1, 2, \ldots, m\}$, and, similarly, $V = \mathbb{R}^n$ considered as the linear space of functions on $Y := \{1, 2, \ldots, n\}$. In this case, $U \otimes V$ is the linear space of all $m \times n$ matrices, considered as functions on $X \times Y = \{ (i, j) : i = 1, \ldots, m; j = 1, \ldots, n \}$.

One verifies that the tensor product is bilinear, that is, the map

$$U \times V \to U \otimes V : (u, v) \mapsto u \otimes v$$

is linear in each argument:

$$(\alpha_1 u_1 + \alpha_2 u_2) \otimes v = \alpha_1 (u_1 \otimes v) + \alpha_2 (u_2 \otimes v),$$
$$u \otimes (\beta_1 v_1 + \beta_2 v_2) = \beta_1 (u \otimes v_1) + \beta_2 (u \otimes v_2),$$

In particular,

$$U \otimes V = \{ \sum_i u_i \otimes v_i : u_i \in U, v_i \in V, \text{all } i \}$$

which saves a little writing.

Let now λ and μ be linear functionals on U and V, respectively. One defines $\lambda \otimes \mu$ by the rule

$$(\lambda \otimes \mu)(\sum_i u_i \otimes v_i) := \sum_i (\lambda u_i)(\mu v_i), \quad \text{all } \sum_i u_i \otimes v_i. \tag{5}$$

Clearly, if $\lambda \otimes \mu$ is a map on $U \otimes V$ satisfying (5), then $\lambda \otimes \mu$ is a linear functional on $U \otimes V$. But, (5) requires some discussion before we can accept

it as defining a map on $U \otimes V$. For, (5) makes use of the particular form of $\sum_i u_i \otimes v_i$, that is, of the particular u_i's and v_i's, to define $\lambda \otimes \mu$ on the function $\sum_i u_i \otimes v_i$. On the other hand, an element $w \in U \otimes V$ may be written in many different ways. If, for example, $w = u \otimes v$, and $u = u_1 + u_2$, while $v = 3v_1$, then we can write w as

$$u \otimes v \text{ or } u_1 \otimes v + u_2 \otimes v \text{ or even as } (3u_1) \otimes v_1 + u_2 \otimes v.$$

Correspondingly, the rule (5) would give

$$(\lambda u)(\mu v) \text{ or } (\lambda u_1)(\mu v) + (\lambda u_2)(\mu v) \text{ or } (3\lambda u_1)(\mu v_1) + (\lambda u_2)(\mu v)$$

(among others) for "the" value of $(\lambda \otimes \mu)w$.

The doubts just raised can be dispelled as follows. If w is any function on $X \times Y$, and y is a particular point in Y, then

$$w_y(x) := w(x, y), \qquad \text{all } x \in X,$$

defines a function w_y on X, the **y-section** of w. If, in particular,

$$w = \sum_i u_i \otimes v_i, \qquad \text{for some } (u_i, v_i)\text{'s in } U \times V,$$

then, by the definition of $u_i \otimes v_i$, we can compute $w_y(x)$ as $w_y(x) = \sum_i u_i(x)v_i(y)$; that is,

$$w_y = \sum_i v_i(y)u_i.$$

This shows that $w_y \in U$, hence allows us to compute the number λw_y, and to compute it as

$$\lambda w_y = \sum_i v_i(y)(\lambda u_i), \tag{6}$$

using the linearity of λ, Let now w_λ be the **λ-section of** w, that is, the function on Y defined by

$$w_\lambda(y) := \lambda w_y, \qquad \text{all } y \in Y.$$

The notation is correct, w_λ depends only on w and λ and not on the particular u_i's and v_i's used to represent w since w_y depends only on w. On the other hand, w_λ can be computed by (6) as

$$w_\lambda = \sum_i (\lambda u_i)v_i \qquad \text{whenever } w = \sum_i u_i \otimes v_i.$$

This shows that w_λ is an element of V, hence allows us to compute the number μw_λ, and to compute it as

$$\mu w_\lambda = \mu(\sum_i (\lambda u_i)v_i) = \sum_i (\lambda u_i)(\mu v_i).$$

This shows that the number $\sum_i (\lambda u_i)(\mu v_i)$ depends only on λ, μ and the function $w = \sum_i u_i \otimes v_i$ and not on the particular constituents (u_i), (v_i) of its representation.

We conclude that, *for every linear functional λ on U and every linear functional μ on V, (5) defines a linear functional on $U \otimes V$, and this linear functional satisfies*

$$(\lambda \otimes \mu)w = \lambda w_\mu = \mu w_\lambda, \quad \text{for all } w \in U \otimes V. \tag{7}$$

Here, w_μ is the μ-section of w, that is, $w_\mu(x)$ is the result of applying μ to $w(x,y)$ as a function of y for each fixed $x \in X$.

In particular, with (λ_i) a sequence of linear functionals dual to a given basis (u_i) for U (meaning that the Gramian, $(\lambda_i u_j)$ is the identity matrix) and also (μ_h) dual to a given basis (v_k) for V, it follows that the Gramian $((\lambda_i \otimes \mu_h)(u_j \otimes v_k))$ is the identity matrix and, in particular, $(u_i \le v_k)$ is a basis for $U \otimes V$.

To give a simple example, let $U = V = C^{(k)}(\mathbb{R})$, and let

$$\lambda = \delta_a^{(r)}, \qquad \mu = \delta_b^{(s)}$$

for some a and b, and some integers $r, s \le k$. By this we mean that

$$\lambda u = u^{(r)}(a), \quad u \in U, \text{ and } \mu v = v^{(s)}(b), \quad \text{all } v \in V.$$

Then $U \otimes V$ is contained in $C^{(k,k)}(\mathbb{R}^2)$, the space of bivariate functions with all derivatives of order $\le (k,k)$ continuous. Further, on $U \otimes V$, $\lambda \otimes \mu$ agrees with the linear functional

$$\nu := \delta_{a,b}^{(r,s)} : w \mapsto (\partial^{r+s} w/(\partial x^r \partial y^s))(a,b)$$

since, for every $u, v \in C^{(k)}(\mathbb{R})$,

$$\nu(u \otimes v) = (\partial^{r+s}/(\partial x^r \partial y^s))u(x)v(y)|_{\substack{x=a \\ y=b}} = u^{(r)}(a)v^{(s)}(b) = (\lambda u)(\mu v).$$

With μ changed to

$$\mu v := \int_b^c v(y)\,dy, \qquad \text{all } v \in C^{(k)},$$

$\lambda \otimes \mu$ agrees with the linear functional ν given by the rule

$$\nu w = \int_b^c (\partial^r/\partial x^r)w(a,y)\,dy, \qquad \text{all } w \in C(k,k).$$

(8) Example: Evaluation of a tensor product spline. We take $U = \$_{h,\mathbf{s}}$, $V = \$_{k,\mathbf{t}}$ and wish to evaluate the tensor product spline $w \in U \otimes V$ at the point (a, b). Since $w = \sum_{i,j} \alpha_{ij} B_{ij}$ for some coefficient matrix (α_{ij}) and for

$$B_{ij}(x, y) := B_{i,h,\mathbf{s}}(x) B_{j,k,t}(y),$$

we can accomplish this by factoring out appropriately. Explicitly,

$$w(a, b) = \sum_i (\sum_j \alpha_{ij} B_{j,k,\mathbf{t}}(b)) B_{i,h,\mathbf{s}}(a) = \lambda w_\mu$$

or

$$w(a, b) = \sum_j (\sum_i \alpha_{ij} B_{i,h,\mathbf{s}}(a)) B_{j,k,\mathbf{t}}(b) = \mu w_\lambda,$$

with $\lambda := [a]$, $\mu := [b]$.

The following program fragment carries out this evaluation with the aid of the function subprogram BVALUE, using the second formula because of the way arrays are stored in Fortran. We assume the B-coefficients α_{ij} to be in an array ALPHA, and use the same notation as in the one–variable case except that the letter X or Y is added to indicate the variables. Thus, the knot sequence on the x–variable is TX(1), ..., TX(NX+KX), that for the y–variable is TY(1), ..., TY(NY+KY).

```
   CALL INTERV ( TY, NY, B, LEFTY, MFLAG )
   VALUE = 0.
   IF ( MFLAG .NE. 0)          GO TO 11
   DO 10 J=1,KY
10    BCOEF(J) = BVALUE ( TX, ALPHA(1,LEFTY-KY+J), NX, KY, A, 0)
   VALUE = BVALUE (TY(LEFTY-KY+1), BCOEF, KY, KY, B, 0)
11 ...
```

□

The tensor product of two linear interpolation schemes We are now ready to consider the product of two univariate interpolation schemes. The basic facts are contained in the following theorem.

(9) **Theorem.** *Suppose that the Gramian matrix $A := (\lambda_i f_j)$ for the sequence $f_1, \ldots, f_m$ in U and the sequence $\lambda_1, \ldots, \lambda_m$ of linear functionals on U is invertible, so that the LIP given by*

$$F := \operatorname{span}(f_i)_1^m \quad \text{and} \quad \Lambda := \operatorname{span}(\lambda_i)_1^m$$

is correct. Similarly, assume that $B := (\mu_i g_j)$ is invertible, with $g_1, \ldots, g_n$ in V and $\mu_1, \ldots, \mu_n$ linear functionals on V, and set

$$G := \operatorname{span}(g_i)_1^n \quad \text{and} \quad \mathrm{M} := \operatorname{span}(\mu_i)_1^n.$$

Finally, assume that (ν_{ij}) *is a matrix (or double sequence) of linear functionals on some linear space* W *containing* $U \otimes V$ *so that*

$$\nu_{ij}(u \otimes v) = (\lambda_i u)(\mu_j v) \quad \text{for all} \quad i, j \quad \text{and all} \quad (u, v) \in U \times V.$$

Then:

(i) $(f_i \otimes g_j)$ *is a basis for* $F \otimes G$, *hence*

$$\dim F \otimes G = (\dim F)(\dim G) = mn;$$

(ii) the LIP on W *given by* $F \otimes G$ *and* $\operatorname{span}(\nu_{ij})$ *is correct;*

(iii) for given $w \in W$, *the interpolant* Rw *can be computed as*

$$Rw = \sum_{i,j} \Gamma(i,j) f_i \otimes g_j \tag{10}$$

with

$$\Gamma := \Gamma_w := A^{-1} L_w (B^T)^{-1} \tag{11}$$

and

$$L_w(i,j) := \nu_{ij} w; \quad \text{all} \quad i, j. \tag{12}$$

Remark. Here and below, we write $D(i,j)$ rather than D_{ij} or d_{ij} for the (i,j)th entry of the matrix D. □

PROOF OF THE THEOREM. If $w \in F \otimes G$, then

$$w = \sum_{i,j} \Gamma(i,j) f_i \otimes g_j \tag{13}$$

for some matrix Γ. But then

$$\begin{aligned} L_w(r,s) = (\lambda_r \otimes \mu_s) w &= \sum_{i,j} \Gamma(i,j) (\lambda_r f_i) (\mu_s g_j) \\ &= \sum_{i,j} \Gamma(i,j) A(r,i) B(s,j) \\ &(A \Gamma B^T)(r,s), \quad \text{all} \quad r, s. \end{aligned}$$

This shows that

$$L_w = A \Gamma B^T.$$

Since both A and B are invertible by assumption, and since L_w does not depend on the particular representation (13) for w but only on w, this implies the uniqueness of the expansion (13) for w, therefore showing (i).

It follows further that, for a given matrix L and a given $w \in F \otimes G$, we have $L_w = L$ if and only if the coefficient matrix Γ for w with respect to the basis $(f_i \otimes g_j)$ of $F \otimes G$ satisfies

$$\Gamma = A^{-1} L (B^T)^{-1},$$

proving (ii) and (iii). □

The calculation of a tensor product interpolant We come now to the heart of the matter, the computational savings in the construction of the interpolant Rw to be had by using (11) instead of solving the linear system

$$\sum_{i,j}(\lambda_r f_i)\,(\mu_s g_j)\Gamma(i,j) = L_w(r,s), \quad \text{all} \quad r,s, \tag{14}$$

directly for the coefficient matrix Γ. Equations (14) constitute a linear system of mn equations for the mn unknowns. Its straightforward solution by Gauss elimination therefore would require $\mathcal{O}\left((mn)^3\right)$ operations (see, for example, Forsythe & Moler [1967] for the facts concerning the numerical solution of a linear system). Equation (11) offers us the alternative of forming the product of an $m \times m$, an $m \times n$, and an $n \times n$ matrix, for a total of $\mathcal{O}\left(m^2n + mn^2\right)$ operations.

Of course, we also have to produce the matrices A^{-1} and B^{-1}. The straightforward thing would be to construct these matrices explicitly, but that would be wasteful and less accurate compared to the following procedure. We can obtain the jth column of the matrix product $A^{-1}L$ by applying A^{-1} to the jth column L_j of L, that is, by computing $\mathbf{x}^{(j)} := A^{-1}L_j$. But such an operation corresponds to solving the linear system

$$A\mathbf{x}^{(j)} = L_j$$

for $\mathbf{x}^{(j)}$, which is precisely the operation we have to carry out when constructing the interpolant $\sum_i x_i^{(j)} f_i$ to the data $(\lambda_i g)_1^m = L_j$. In the preceding chapters, we have followed established procedure and solved such a linear system $A\mathbf{x} = \mathbf{b}$ by constructing a triangular factorization $A_\ell A_u = A$ for A and then computing the solution $\mathbf{x}$ in two stages, by forward substitution, getting $\mathbf{y} := A_\ell^{-1}\mathbf{b}$, followed by back substitution, getting $\mathbf{x} = A_u^{-1}\mathbf{y} = A_u^{-1}(A_\ell^{-1}\mathbf{b}) = A^{-1}\mathbf{b}$. The computational cost for this is $\mathcal{O}\left(m^3\right)$ operations for the factorization, and $\mathcal{O}\left(m^2n\right)$ for applying the substitution process to the n right sides $L_1, \ldots, L_n$. This gives a total cost of

$$\mathcal{O}\left(m^3 + m^2n + mn^2 + n^3\right)$$

for computing the coefficients Γ via (11), compared to $\mathcal{O}\left((mn)^3\right)$ for the direct attack.

The savings are even more significant if (as is typical for the interpolation schemes in this book) A and B are *band* matrices, a fact difficult to exploit in a direct attack on (14) whatever the actual ordering of Γ into a vector might be.

Schematically, the construction of Γ in a program would proceed as follows. Assume that

INTERA (A, $\mathbf{b}$, m, Q, $\mathbf{x}$)

carries out the first univariate interpolation process (with INTERB constructed similarly for the second), using the data vector $\mathbf{b}$ of length m and returning the coefficient vector $\mathbf{x}$, together with the factorization for A in Q. Assume further that

BSOLVE (Q, $\mathbf{b}$, m, $\mathbf{x}$,)

produces from the factorization for A in Q and from the right side $\mathbf{b}$ the solution vector $\mathbf{x} = A^{-1}\mathbf{b}$. The following program sketch then would produce Γ from L_w.

```
   CALL INTERA ( A, L_w(·,1), m, Q,Γ(·,1) )
   DO 10 j = 2,n
10    CALL BSOLVE ( Q, L_w(·,j), m, Γ(·,j) )
   CALL INTERB ( B, Γ(1,·),n Q, Γ(1,·) )
   DO 20 i = 2,m
20    CALL BSOLVE ( Q, Γ(i,·), n, Γ(i,·) )
```

Here, $\Gamma(\cdot, j)$ denotes the jth column of Γ taken as a vector, and, similarly, $\Gamma(i, \cdot)$ denotes the ith row of Γ taken as a vector. But, while the Fortran specification `GAMMA(1,J)`, at a place in an argument list where a vector is expected, does indeed refer to the jth column of the matrix `GAMMA` as a vector, the specification `GAMMA(I,1)` does not perform the same service for the ith row of `GAMMA`. One can, of course, adjust to this by introducing an additional argument `ISTEP` to deal with a vector whose entries are stored `ISTEP` apart in memory. But, rather than play such games, which would have further ramifications in subroutines called by `INTERA`, we propose here the following alternative procedure:

Modify the interpolation routine `INTERA` *slightly so that it works on all the* n *right sides* $L_w(\cdot, 1)$, ..., $L_w(\cdot, n)$ *in the same call, but stores the resulting* n *columns of* $A^{-1}L_w$ *in the* n *rows of some work array* `WORK` *to be output.*

This would change our model routine into something like

`INTERA` (A, L_w, m, n, Q, `WORK`).

The whole process of constructing Γ could then be accomplished by just two calls,

`CALL INTERA` (A, L_w, m, n, Q, `WORK`)

`CALL INTERB` (B, `WORK`, N, m, Q, Γ).

In particular, `INTERA` and `INTERB` could be the *same* routine if interpolation in x and in y involves the same interpolation scheme. Also, note that Γ could be stored over L_w in practice. For a proof, see de Boor [1979].

(15) Example: Tensor product spline interpolation We consider the following specific situation in Theorem (9). We have

$$F = \$_{h,\mathbf{s}} \quad \text{with} \quad \mathbf{s} = (s_i)_1^{m+h}, \quad \text{and} \quad \lambda_i = [\sigma_i], \quad i = 1, \ldots, m$$

with $\sigma_1 < \cdots < \sigma_m$. By the Schoenberg-Whitney Theorem XIII(2), this problem is correct if and only if $s_i < \sigma_i < s_{i+h}$, all i, a condition we assume now. Also, we take

$$X = [s_1 \,..\, s_{m+h}], \quad \text{and} \quad U = C(X),$$

the linear space of continuous functions on the interval X.

Similarly, we take

$$G = \$_{k,\mathbf{t}} \quad \text{with} \quad \mathbf{t} = (t_i)_1^{n+k}, \quad \text{and} \quad \mu_i = [\tau_i], \quad i = 1, \ldots, n,$$

with τ strictly increasing and $t_i < \tau_i < t_{i+k}$, all i. Also,

$$Y = [t_1 \,..\, t_{n+k}], \quad \text{and} \quad V = C(Y).$$

Then, $U \otimes V$ is contained in $W := C(X \times Y)$, the linear space of continuous functions on the rectangle $X \times Y$, and $\lambda_i \otimes \mu_j$ coincides (on $U \otimes V$) with the linear functional

$$\nu_{ij} : w \mapsto w(\sigma_i, \tau_j)$$

of evaluation at the point $(\sigma_i, \tau_j) \in X \times Y$.

The theorem therefore gives us, for given $w \in C(X \times Y)$, exactly one spline function $Rw \in F \otimes G$ that agrees with w at the points (σ_i, τ_j), $i = 1, \ldots, m$; $j = 1, \ldots, n$ of the given rectangular mesh. Further, this interpolant can be written in the form

$$Rw = \sum_{i,j} \Gamma(i,j) B_{i,h,\mathbf{s}} \otimes B_{j,k,\mathbf{t}},$$

with

$$\Gamma = (B_{j,h,\mathbf{s}}(\sigma_i))^{-1} (w(\sigma_i, \tau_j)) \, (B_{i,k,\mathbf{t}}(\tau_j))^{-1}.$$

Here, each of the three matrices on the right has row index i and column index j. Once Γ is computed, the interpolant could be evaluated as in Example (8).

To carry out the calculation of Γ, we follow the above discussion and produce an expanded version of SPLINT of Chapter XIII. We call it SPLI2D. It differs from SPLINT only in that it accepts and works on several data vectors at the same time and, consequently, puts out several sets of B-coefficients. We have marked the essential changes with stars in columns 73–76. Additional editorial changes (principally changes in comments and

use of a work array instead of the temporary use of BCOEF) are marked by dashes in columns 73–76, to keep the record straight.

```
      SUBROUTINE SPLI2D ( TAU, GTAU, T, N, K, M, WORK, Q, BCOEF, IFLAG )****
CALLS BSPLVB, BANFAC/SLV
C  THIS IS AN EXTENDED VERSION OF  SPLINT , FOR THE USE IN TENSOR PROD- ----
C  UCT INTERPOLATION.                                                  ----
C
C   SPLI2D  PRODUCES THE B-SPLINE COEFF.S  BCOEF(J,.)  OF THE SPLINE OF ----
C   ORDER  K  WITH KNOTS  T (I), I=1,..., N + K , WHICH TAKES ON THE    ----
C   VALUE  GTAU (I,J)  AT  TAU (I), I=1,..., N , J=1,..., M .          ----
C
C******  I N P U T  ******
C  TAU.....ARRAY OF LENGTH  N , CONTAINING DATA POINT ABSCISSAE.
C    A S S U M P T I O N . . .  TAU  IS STRICTLY INCREASING
C  GTAU(.,J)..CORRESPONDING ARRAY OF LENGTH  N , CONTAINING DATA POINT ----
C        ORDINATES, J=1,...,M                                          ----
C  T.....KNOT SEQUENCE, OF LENGTH  N+K
C  N.....NUMBER OF DATA POINTS AND DIMENSION OF SPLINE SPACE  S(K,T)
C  K.....ORDER OF SPLINE
C  M.....NUMBER OF DATA SETS                                                ****
C
C******  W O R K   A R E A  ******                                        ****
C  WORK  A VECTOR OF LENGTH  N                                           ****
C
C******  O U T P U T  ******
C  Q.....ARRAY OF SIZE  (2*K-1)*N , CONTAINING THE TRIANGULAR FACTORIZ-
C        ATION OF THE COEFFICIENT MATRIX OF THE LINEAR SYSTEM FOR THE B-
C        COEFFICIENTS OF THE SPLINE INTERPOLANT.
C           THE B-COEFFS FOR THE INTERPOLANT OF AN ADDITIONAL DATA SET
C        (TAU(I),HTAU(I)), I=1,...,N  WITH THE SAME DATA ABSCISSAE CAN
C        BE OBTAINED WITHOUT GOING THROUGH ALL THE CALCULATIONS IN THIS
C        ROUTINE, SIMPLY BY LOADING  HTAU  INTO  BCOEF  AND THEN EXECUT-
C        ING THE    CALL BANSLV ( Q, 2*K-1, N, K-1, K-1, BCOEF )
C  BCOEF.....THE B-COEFFICIENTS OF THE INTERPOLANT, OF LENGTH  N
C  IFLAG.....AN INTEGER INDICATING SUCCESS (= 1)  OR FAILURE (= 2)
C        THE LINEAR SYSTEM TO BE SOLVED IS (THEORETICALLY) INVERTIBLE IF
C        AND ONLY IF
C              T(I) .LT. TAU(I) .LT. TAU(I+K),    ALL I.
C        VIOLATION OF THIS CONDITION IS CERTAIN TO LEAD TO  IFLAG = 2 .
C
C******  M E T H O D  ******
C     THE I-TH EQUATION OF THE LINEAR SYSTEM  A*BCOEF = B  FOR THE B-CO-
C  EFFS OF THE INTERPOLANT ENFORCES INTERPOLATION AT  TAU(I), I=1,...,N.
C  HENCE,  B(I) = GTAU(I), ALL I, AND  A  IS A BAND MATRIX WITH  2K-1
C   BANDS (IF IT IS INVERTIBLE).
C     THE MATRIX  A  IS GENERATED ROW BY ROW AND STORED, DIAGONAL BY DI-
C  AGONAL, IN THE  R O W S  OF THE ARRAY  Q , WITH THE MAIN DIAGONAL GO-
C  ING INTO ROW  K .  SEE COMMENTS IN THE PROGRAM BELOW.
C     THE BANDED SYSTEM IS THEN SOLVED BY A CALL TO  BANFAC (WHICH CON-
C  STRUCTS THE TRIANGULAR FACTORIZATION FOR  A  AND STORES IT AGAIN IN
C   Q ), FOLLOWED BY A CALL TO  BANSLV (WHICH THEN OBTAINS THE SOLUTION
C   BCOEF  BY SUBSTITUTION).
C     BANFAC  DOES NO PIVOTING, SINCE THE TOTAL POSITIVITY OF THE MATRIX
C  A  MAKES THIS UNNECESSARY.
C
      INTEGER IFLAG,K,M,N,   I,ILP1MX,J,JJ,KM1,KPKM2,LEFT,LENQ,NP1       ****
      REAL BCOEF(M,N),GTAU(N,M),Q(2*K-1,N),T(N+K),TAU(N),WORK(N),   TAUI****
      NP1 = N + 1
      KM1 = K - 1
      KPKM2 = 2*KM1
      LEFT = K
```

```
C                    ZERO OUT ALL ENTRIES OF Q
      LENQ = N*(K+KM1)
      DO 5 I=1,LENQ
    5    Q(I) = 0.
C
C  ***   LOOP OVER I TO CONSTRUCT THE  N  INTERPOLATION EQUATIONS
      DO 30 I=1,N
         TAUI = TAU(I)
         ILP1MX = MINO(I+K,NP1)
C        *** FIND  LEFT  IN THE CLOSED INTERVAL [I..I+K-1] SUCH THAT
C                T(LEFT) .LE. TAU(I) .LT. T(LEFT+1)
C        MATRIX IS SINGULAR IF THIS IS NOT POSSIBLE
         LEFT = MAXO(LEFT,I)
         IF (TAUI .LT. T(LEFT))          GO TO 998
   15       IF (TAUI .LT. T(LEFT+1))     GO TO 16
            LEFT = LEFT + 1
            IF (LEFT .LT. ILP1MX)        GO TO 15
         LEFT = LEFT - 1
         IF (TAUI .GT. T(LEFT+1))        GO TO 998
C        *** THE I-TH EQUATION ENFORCES INTERPOLATION AT TAUI, HENCE
C        A(I,J) = B(J,K,T)(TAUI), ALL J. ONLY THE  K  ENTRIES WITH  J =
C        LEFT-K+1,...,LEFT ACTUALLY MIGHT BE NONZERO. THESE  K  NUMBERS
C        ARE RETURNED, IN  WORK  (USED FOR TEMP.STORAGE HERE), BY THE   ----
C        FOLLOWING
   16    CALL BSPLVB ( T, K, 1, TAUI, LEFT, WORK )                      ----
C        WE THEREFORE WANT  WORK(J) = B(LEFT -K+J)(TAUI) TO GO INTO     ----
C        A(I,LEFT-K+J), I.E., INTO  Q(I-(LEFT+J)+2*K,(LEFT+J)-K) SINCE
C        A(I+J,J)  IS TO GO INTO  Q(I+K,J), ALL I,J,  IF WE CONSIDER  Q
C        AS A TWO-DIM. ARRAY , WITH  2*K-1  ROWS (SEE COMMENTS IN
C        BANFAC). IN THE PRESENT PROGRAM, WE TREAT  Q  AS AN EQUIVALENT
C        ONE-DIMENSIONAL ARRAY (BECAUSE OF FORTRAN RESTRICTIONS ON
C        DIMENSION STATEMENTS) . WE THEREFORE WANT  WORK(J) TO GO INTO  ----
C        ENTRY
C            I -(LEFT+J) + 2*K + ((LEFT+J) - K-1)*(2*K-1)
C                  =  I-LEFT+1 + (LEFT -K)*(2*K-1) + (2*K-2)*J
C        OF  Q .
         JJ = I-LEFT+1 + (LEFT-K)*(K+KM1)
         DO 30 J=1,K
            JJ = JJ+KPKM2
   30       Q(JJ) = WORK(J)                                             ----
C
C     ***OBTAIN FACTORIZATION OF  A  , STORED AGAIN IN  Q.
      CALL BANFAC ( Q, K+KM1, N, KM1, KM1, IFLAG )
                                       GO TO (40,999), IFLAG
C     *** SOLVE  A*BCOEF = GTAU  BY BACKSUBSTITUTION
   40 DO 50 J=1,M                                                       ****
         DO 41 I=1,N                                                    ----
   41       WORK(I) = GTAU(I,J)                                         ****
         CALL BANSLV ( Q, K+KM1, N, KM1, KM1, WORK )                    ----
         DO 50 I=1,N                                                    ****
   50       BCOEF(J,I) = WORK(I)                                        ****
                                       RETURN
  998 IFLAG = 2
  999 PRINT 699
  699 FORMAT(41H LINEAR SYSTEM IN  SPLINT  NOT INVERTIBLE)
                                       RETURN
      END
```

We make use of SPLI2D in the following simple example of interpolation to a tensor product spline by tensor product splines of order (3,4).

```
CHAPTER XVII, EXAMPLE 2. BIVARIATE SPLINE INTERPOLATION
CALLS  SPLI2D(BSPLVB,BANFAC/SLV),INTERV,BVALUE(BSPLVB*,INTERV*)
      INTEGER I,IFLAG,J,JJ,KX,KY,LEFTY,MFLAG,NX,NY
      PARAMETER (NX=7,KX=3,NY=6,KY=4)
      REAL BCOEF(NX,NY),TAUX(NX),TAUY(NY),TX(NX+KX),TY(NY+KY)
     *          ,WORK1(NX,NY),WORK2(NX),WORK3(NX*NY)
C     INTEGER I,IFLAG,J,JJ,KP1,KX,KY,LEFTY,MFLAG,NX,NY
C     DATA NX,KX,NY,KY /7,3,6,4/
C     REAL BCOEF(7,6),TAUX(7),TAUY(6),TX(10),TY(10)
C    *          ,WORK1(7,6),WORK2(7),WORK3(42)
      G(X,Y) = AMAX1(X-3.5,0.)**2 + AMAX1(Y-3.,0.)**3
C *** SET UP DATA POINTS AND KNOTS
C     IN X, INTERPOLATE BETWEEN KNOTS BY PARABOLIC SPLINES, USING
C     NOT-A-KNOT END CONDITION
      DO 10 I=1,NX
   10    TAUX(I) = FLOAT(I)
      DO 11 I=1,KX
         TX(I) = TAUX(1)
   11    TX(NX+I) = TAUX(NX)
      KP1 = KX+1
      DO 12 I=KP1,NX
   12    TX(I) = (TAUX(I-KX+1) + TAUX(I-KX+2))/2.
C     IN Y, INTERPOLATE AT KNOTS BY CUBIC SPLINES, USING NOT-A-KNOT
C     END CONDITION
      DO 20 I=1,NY
   20    TAUY(I) = FLOAT(I)
      DO 21 I=1,KY
         TY(I) = TAUY(1)
   21    TY(NY+I) = TAUY(NY)
      KP1 = KY+1
      DO 22 I=KP1,NY
   22    TY(I) = TAUY(I-KY+2)
C  *** GENERATE AND PRINT OUT FUNCTION VALUES
      PRINT 620,(TAUY(I),I=1,NY)
  620 FORMAT(' GIVEN DATA'//6F12.1)
      DO 32 I=1,NX
         DO 31 J=1,NY
   31       BCOEF(I,J) = G(TAUX(I),TAUY(J))
   32    PRINT 632,TAUX(I),(BCOEF(I,J),J=1,NY)
  632 FORMAT(F5.1,6E12.5)
C
C  *** CONSTRUCT B-COEFFICIENTS OF INTERPOLANT
C
      CALL SPLI2D(TAUX,BCOEF,TX,NX,KX,NY,WORK2,WORK3,WORK1,IFLAG)
      CALL SPLI2D(TAUY,WORK1,TY,NY,KY,NX,WORK2,WORK3,BCOEF,IFLAG)
C
C  *** EVALUATE INTERPOLATION ERROR AT MESH POINTS AND PRINT OUT
      PRINT 640,(TAUY(J),J=1,NY)
  640 FORMAT(//' INTERPOLATION ERROR'//6F12.1)
      DO 45 J=1,NY
         CALL INTERV(TY,NY+1,TAUY(J),LEFTY,MFLAG)
         DO 45 I=1,NX
            DO 41 JJ=1,KY
   41          WORK2(JJ)=BVALUE(TX,BCOEF(1,LEFTY-KY+JJ),NX,KX,TAUX(I),0)
   45       WORK1(I,J) = G(TAUX(I),TAUY(J)) -
     *                   BVALUE(TY(LEFTY-KY+1),WORK2,KY,KY,TAUY(J),0)
      DO 46 I=1,NX
   46    PRINT 632,TAUX(I),(WORK1(I,J),J=1,NY)
                                                  STOP
      END
```

```
INTERPOLATION ERROR

        1.0          2.0          3.0          4.0          5.0          6.0
 1.0 0.00000+00 0.00000+00-0.11842-14-0.11921-06 0.00000+00 0.00000+00
 2.0 0.00000+00-0.44409-15-0.47370-14 0.00000+00 0.00000+00 0.00000+00
 3.0 0.00000+00 0.44409-15-0.11842-14-0.11921-06 0.00000+00 0.00000+00
 4.0 0.00000+00 0.29802-07 0.00000+00 0.00000+00 0.95367-06 0.00000+00
 5.0 0.00000+00 0.00000+00 0.00000+00 0.00000+00 0.19073-05 0.00000+00
 6.0 0.00000+00 0.95367-06 0.47684-06 0.14305-05 0.19073-05 0.00000+00
 7.0 0.00000+00 0.19073-05 0.95367-06 0.00000+00 0.00000+00 0.00000+00
```

The output shows the error to be acceptably small (for 27 binary digit floating-point arithmetic) except in the last row and the last column. Here, we are led into trouble once again by our use of a right continuous BVALUE where we should be using a left continuous one when evaluating at the right most knot. Else, we should take the last k knots greater than the last interpolation site, or we could first convert to ppform as is done later in Example (16). □

The ppform of a tensor product spline is, not unexpectedly, the tensor product of the ppform for univariate splines. Specifically, assume that the subprogram BSPLPP produces the break sequence $(\zeta_i)_1^{\ell+1}$ for $f \in \$_{h,\mathbf{s}}$. It then also produces the numbers $c_{ij}(f) := f^{(i-1)}(\zeta_j^+)$, $i = 1, \ldots, h$, as the polynomial coefficients of f on the jth interval $(\zeta_j \,..\, \zeta_{j+1})$, $j = 1, \ldots, \ell$. Correspondingly, if the subprogram BSLPP produces the break sequence $(\xi_i)_1^{\hat{\ell}+1}$ for $g \in \$_{k,\mathbf{t}}$, then it also provides the numbers $\hat{c}_{ij}(g) := g^{(i-1)}(\xi_j^+)$, $i = 1, \ldots, k$, $j = 1, \ldots, \hat{\ell}$.

The tensor product of this information, for $w \in \$_{h,\mathbf{s}} \otimes \$_{k,\mathbf{t}}$, consists of

(i) the integers ℓ, h, and $\hat{\ell}$, k, giving the number of pieces and the order for each variable;

(ii) the two sequences $(\zeta_i)_1^{\ell+1}$ and $(\xi_i)_1^{\hat{\ell}+1}$ of breaks in each variable, thus providing a rectangular grid;

(iii) the four-dimensional array

$$c_{i,p;j,q}(w) := D_x^{i-1} D_y^{j-1} w(\zeta_p^+, \xi_q^+), \quad \begin{matrix} i = 1, \ldots, h; & p = 1, \ldots, \ell \\ j = 1, \ldots, k; & q = 1, \ldots, \hat{\ell} \end{matrix} .$$

This is, indeed, a tensor product, since $c_{i,p;j,q}(f \otimes g) = c_{i,p}(f)\hat{c}_{j,q}(g)$ for every $f \otimes g \in \$_{h,\mathbf{s}} \otimes \$_{k,\mathbf{t}}$.

The evaluation of a tensor product spline from its ppform The typical operations one might perform on a tensor product w are linear and, since they usually lead to one more or more numbers, they then must consist of evaluating one or more linear functionals on the tensor product w.

If such a linear functional ν can be written as a tensor product, $\nu = \lambda \otimes \mu$, then we can evaluate $\nu(w)$ by forming the tensor product of the codes for the evaluation of λ and μ. More explicitly, in order to evaluate the linear functional λ on some function f, we must have available a representation for f, say $f = \sum \alpha_i f_i$, and we must have a way of calculating λf_i for all i. We then compute λf by $\sum \alpha_i (\lambda f_i)$. Similarly, we calculate μg in the form $\sum \beta_j (\mu g_j)$. If now $w \in F \otimes G$ (with $F := \text{span}(f_i)$ and $G := \text{span}(g_j)$), then it has a representation $w = \sum \alpha_{ij} f_i \otimes g_j$. Therefore, if $\nu = \lambda \otimes \mu$ on $F \otimes G$, then we have

$$\nu w = \sum_{i,j} \alpha_{ij} (\lambda f_i)(\mu g_j) = \sum_j (\sum_i \alpha_{ij} \lambda f_i) \mu g_j .$$

In effect, for each j, we apply our scheme for evaluating λf from the coefficients (α_i) for f to the specific coefficient sequence $\alpha_{1j}, \alpha_{2j}, \ldots$, that is, to the jth column of the coefficient matrix (α_{ij}) for w, thus producing a number β_j. Then, we apply our scheme for producing μg from the coefficients (β_j) for g to this particular coefficient sequence.

This procedure was illustrated in Example (8). We now give the evaluation from the ppform as a second illustration.

We know that

PPVALU (BREAK, COEF, L, K, a, j)

returns the number $(D^j f)(a)$ if the ppform for f is in BREAK, COEF, L, K. Therefore, we can calculate the number $D_x^r D_y^s w(a,b)$ from the ppform BREAKX, BREAKY, COEF, LX, KX, LY, KY for w as follows (assuming that COEF$(i,p,j,q) = c_{i,p;j,q}(w)$, all i,p,j,q).

```
      CALL INTERV ( BREAKY, LY, a, LEFTY, MFLAG )
      DO 10 J=1,KY
   10    COEFY(J) = PPVALU(BREAKX,COEF(1,1,J,LEFTY),LX,KX,a,r)
      VALUE = PPVALU(BREAKY(LEFTY),COEFY,1,KY,b,s)
```

Note the similarity with the program sketch in Example (8). It is clear that it would be more efficient to work out an expanded version of PPVALU to combine the various calls to PPVALU in the DO loop into one. It is also clear that the big calculation in the DO loop need not be repeated if we only change b, as long as b remains in the same break interval, that is, as long as LEFTY remains unchanged.

This brings up the point that evaluation of a tensor product at many points is done most efficiently when the sites lie in lines parallel to the coordinate axes, and, because of Fortran, preferably on lines parallel to the y–axis. To pursue this point a bit further, in such a circumstance, it does not pay to produce the storage consuming ppform. Rather, one gets the univariate B-form along that line, converts it to ppform via BSPLPP and then uses PPVALU.

Conversion from B-form to ppform is another example of evaluating tensor product linear functionals on a tensor product since it involves the calculation of the numbers $(D_x^i D_y^j w)(\zeta_p^+, \xi_q^+)$ from the B-form for w. We generate the corresponding univariate information $(D^i f)(\zeta_p^+)$, $i = 0, \ldots, h-1$; $p = 1, \ldots, \ell$, in one fell swoop, through a

CALL BSPLPP (T, BCOEF, N, K, WORK, BREAK, COEF, L),

much as we generate *all* the B-coefficients of the interpolant from the data vector by *one* call to SPLINT. We therefore take here the tack of enlarging BSPLPP appropriately, to a routine

BSPP2D (T, BCOEF, N, K, M, SCRTCH, BREAK, COEF, L)

which returns in COEF(i,.,.) the pp–coefficients for the function whose B-coefficients are in BCOEF(.,i), $i = 1, \ldots, M$. In this, we follow entirely the suggestion made earlier concerning the formation of the tensor product of two interpolation schemes.

With such a routine BSPP2D handy, the conversion from the B-form in TX, TY, BCOEF, NX, NY, KX, KY to the ppform BREAKX, BREAKY, COEF, LX, LY, KX, KY is accomplished by just two calls to this routine, namely by

```
CALL BSPP2D (TX,BCOEF,NX,KX,NY,WORK1,BREAKX,WORK2,LX)
CAL.L BSPP2D (TY,WORK2,NY,KY,NX*KX,WORK1,BREAKY,COEF,LY)
```

Here is such a routine BSPP2D. We have again added stars and dashed in columns 73–76 of those statements modified in the original BSPLPP to accommodate a whole matrix of B-spline coefficients as input rather than just one vector.

```
      SUBROUTINE BSPP2D ( T, BCOEF, N, K, M, SCRTCH, BREAK, COEF)        ****
CALLS  BSPLVB
C  THIS IS AN EXTENDED VERSION OF  BSPLPP  FOR USE WITH TENSOR PRODUCTS ----
C
CONVERTS THE B-REPRESENTATION  T, BCOEF(.,J), N, K  OF SOME SPLINE INTO ----
C  ITS PP-REPRESENTATION  BREAK, COEF(J,.,.), L, K , J=1, ..., M  .      ----
C
C******  I N P U T  ******
C  T.....KNOT SEQUENCE, OF LENGTH  N+K
C  BCOEF(.,J) B-SPLINE COEFFICIENT SEQUENCE, OF LENGTH  N ,J=1,...,M     ----
C  N.....LENGTH OF  BCOEF  AND  DIMENSION OF SPLINE SPACE  SPLINE(K,T)
C  K.....ORDER OF THE SPLINE
C
C  W A R N I N G  . . .  THE RESTRICTION   K .LE. KMAX (= 20)   IS IMPO-
C        SED BY THE ARBITRARY DIMENSION STATEMENT FOR  BIATX  BELOW, BUT
C        IS  N O W H E R E  C H E C K E D  FOR.
C
C  M     NUMBER OF DATA SETS                                             ****
C
C******  W O R K  A R E A  ******
C  SCRTCH   OF SIZE  (K,K,M), NEEDED TO CONTAIN BCOEFFS OF A PIECE OF   ****
C        THE SPLINE AND ITS  K-1  DERIVATIVES   FOR EACH OF THE M SETS  ----
```

```
C
C******  O U T P U T  ******
C  BREAK.....BREAKPOINT SEQUENCE, OF LENGTH  L+1, CONTAINS (IN INCREAS-
C         ING ORDER) THE DISTINCT POINTS IN THE SEQUENCE  T(K),...,T(N+1)
C  COEF(MM,.,.)  ARRAY OF SIZE (K,N), WITH  COEF(MM,I,J) = (I-1)ST DER- ****
C         IVATIVE OF  MM-TH  SPLINE AT BREAK(J) FROM THE RIGHT, MM=1,.,M ****
C
C******  M E T H O D  ******
C     FOR EACH BREAKPOINT INTERVAL, THE  K  RELEVANT B-COEFFS OF THE
C  SPLINE ARE FOUND AND THEN DIFFERENCED REPEATEDLY TO GET THE B-COEFFS
C  OF ALL THE DERIVATIVES OF THE SPLINE ON THAT INTERVAL. THE SPLINE AND
C  ITS FIRST  K-1  DERIVATIVES ARE THEN EVALUATED AT THE LEFT END POINT
C  OF THAT INTERVAL, USING  BSPLVB  REPEATEDLY TO OBTAIN THE VALUES OF
C  ALL B-SPLINES OF THE APPROPRIATE ORDER AT THAT POINT.
C
      INTEGER K,M,N,    I,J,JP1,KMAX,KMJ,LEFT,LSOFAR,MM                    ****
      PARAMETER (KMAX = 20)
      REAL BCOEF(N,M),BREAK(N+2-K),COEF(M,K,N+1-K),T(N+K)                  ****
     *                            ,SCRTCH(K,K,M),BIATX(KMAX),DIFF,FKMJ,SUM
      LSOFAR = 0
      BREAK(1) = T(K)
      DO 50 LEFT=K,N
C                                    FIND THE NEXT NONTRIVIAL KNOT INTERVAL.
         IF (T(LEFT+1) .EQ. T(LEFT))    GO TO 50
         LSOFAR = LSOFAR + 1
         BREAK(LSOFAR+1) = T(LEFT+1)
         IF (K .GT. 1)                  GO TO 9
         DO 5 MM=1,M                                                       ****
    5       COEF(MM,1,LSOFAR) = BCOEF(LEFT,MM)                             ****
                                        GO TO 50
C        STORE THE K B-SPLINE COEFF.S RELEVANT TO CURRENT KNOT INTERVAL
C                                   IN  SCRTCH(.,1) .
    9    DO 10 I=1,K
         DO 10 MM=1,M                                                      ****
   10       SCRTCH(I,1,MM) = BCOEF(LEFT-K+I,MM)                           ****
C
C        FOR J=1,...,K-1, COMPUTE THE  K-J  B-SPLINE COEFF.S RELEVANT TO
C        CURRENT KNOT INTERVAL FOR THE J-TH DERIVATIVE BY DIFFERENCING
C        THOSE FOR THE (J-1)ST DERIVATIVE, AND STORE IN SCRTCH(.,J+1) .
         DO 20 JP1=2,K
            J = JP1 - 1
            KMJ = K - J
            FKMJ = FLOAT(KMJ)
            DO 20 I=1,KMJ
               DIFF = (T(LEFT+I) - T(LEFT+I - KMJ))/FKMJ                   ----
               IF (DIFF .LE. 0.)          GO TO 20                         ----
               DO 15 MM=1,M                                                ****
   15             SCRTCH(I,JP1,MM) =                                      ****
     *             (SCRTCH(I+1,J,MM) - SCRTCH(I,J,MM))/DIFF               ****
   20          CONTINUE
C
C        FOR  J = 0, ..., K-1, FIND THE VALUES AT  T(LEFT)  OF THE  J+1
C        B-SPLINES OF ORDER  J+1  WHOSE SUPPORT CONTAINS THE CURRENT
C        KNOT INTERVAL FROM THOSE OF ORDER  J  (IN  BIATX ), THEN COMB-
C        INE WITH THE B-SPLINE COEFF.S (IN SCRTCH(.,K-J) ) FOUND EARLIER
C        TO COMPUTE THE (K-J-1)ST DERIVATIVE AT  T(LEFT)  OF THE GIVEN
C        SPLINE.
         CALL BSPLVB ( T, 1, 1, T(LEFT), LEFT, BIATX )
         DO 25 MM=1,M                                                      ****
   25       COEF(MM,K,LSOFAR) = SCRTCH(1,K,MM)                            ****
```

```
C
      DO 30 JP1=2,K
         CALL BSPLVB ( T, JP1, 2, T(LEFT), LEFT, BIATX )
         KMJ = K+1 - JP1
         DO 30 MM=1,M                                                    ****
            SUM = 0.                                                     ----
            DO 28 I=1,JP1                                                ----
28             SUM = BIATX(I)*SCRTCH(I,KMJ,MM) + SUM                     ****
30          COEF(MM,KMJ,LSOFAR) = SUM                                    ****
50    CONTINUE
                                          RETURN
   END
```

(16) Example: Tensor product spline interpolation (continued) We add to our program for tensor product spline interpolation in Example (15) a few additional lines to convert the B-form of the interpolant to its ppform before evaluation. This requires

(i) additional storage, that is,

```
      REAL      BREAKX(LX+1),BREAKY(LY+1),COEF(KX,LX,KY,LY)
     *         ,WORK4(KX,KX,NY),WORK5(NY,KX,LX),WORK6(KY,KY,NX*KX)
```

with LX = 5, LY = 3 for the parameter setting given in Example (15);

(ii) insertion of the two calls

```
      CALL BSPP2D(TX,BCOEF,NX,KX,NY,WORK4,BREAKX,WORK5)
      CALL BSPP2D(TY,WORK5,NY,KY,LX*KX,WORK6,BREAKY,COEF)
```

after the calls to SPLI2D, in order to convert to ppform;

(iii) replacement of the evaluation statements involving BVALUE

```
      DO 45 J=1,NY
         CALL INTERV(TY,NY+1,TAUY(J),LEFTY,MFLAG)
         DO 45 I=1,NX
            DO 41 JJ=1,KY
41             WORK2(JJ)=BVALUE(TX,BCOEF(1,LEFTY-KY+JJ),NX,KX,TAUX(I),0)
45          WORK1(I,J) = G(TAUX(I),TAUY(J)) -
     *                   BVALUE(TY(LEFTY-KY+1),WORK2,KY,KY,TAUY(J),0)
```

by the corresponding statements using PPVALU:

```
      DO 45 J=1,NY
         CALL INTERV(BREAKY,LY,TAUY(J),LEFTY,MFLAG)
         DO 45 I=1,NX
            DO 41 JJ=1,KY
41           WORK2(JJ)=PPVALU(BREAKX,COEF(1,1,JJ,LEFTY),LX,KX,TAUX(I),0)
45          WORK1(I,J) = G(TAUX(I),TAUY(J)) -
     *      PPVALU(BREAKY(LEFTY),WORK2,1,KY,TAUY(J),0)
```

We do not list again the entire program here, but do list the matrix of interpolation errors produced by it. This time, the error is suitably small everywhere.

```
INTERPOLATION ERROR

          1.0           2.0           3.0           4.0           5.0           6.0
1.0 0.00000+00 0.71054-14 0.00000+00 0.00000+00 0.95367-06 0.38147-05
2.0 0.00000+00 0.00000+00 0.17764-14 0.59605-07 0.95367-06 0.57220-05
3.0 0.00000+00-0.71054-14 0.28866-14-0.23842-06 0.00000+00 0.00000+00
4.0 0.00000+00 0.14901-07 0.00000+00-0.11921-06 0.00000+00 0.57220-05
5.0 0.00000+00 0.23842-06 0.00000+00 0.23842-06 0.47684-05 0.22888-04
6.0 0.00000+00 0.47684-06 0.00000+00 0.47684-06 0.19073-05 0.38147-05
7.0 0.00000+00 0.19073-05 0.00000+00-0.95367-06 0.38147-05 0.15259-04
```

□

Limitations of tensor product approximation and alternatives Approximation by tensor products is limited to situations where it makes sense (or, at least does no harm) to have *preferred directions* in the approximant. By this we mean that, for example, a function f from $\$_{h,\mathbf{s}} \otimes \$_{k,\mathbf{t}}$ is a pp function of order h along lines parallel to the x–axis, and a pp function of order k along lines parallel to the y–axis, while, along any other lines, it is a pp function of order $h + k - 1$.

Again, if the function g to be approximated has a difficult feature, a peak say, running along a line parallel to the x–axis, then, by proper refinement of the mesh in the y–variable, we can adjust to it efficiently. But should this feature run along a diagonal line, then, in order to approximate well to it, we must have a fine mesh everywhere in x *and* y. This means that we end up having a fine grid even in regions where the function to be approximated is quite smooth. This is, at the very least, a waste in degrees of freedom and therefore in storage and computing time.

Another severe limitation of tensor product methods is the requirement that the information about the function g to be approximated be in tensor product form, that is, in the form $(\lambda_i \otimes \mu_j)g$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, for some univariate linear functionals (λ_i) and (μ_j). Thus, there is no advantage to be gained from using tensor products directly over other function classes when this requirement is not satisfied. E.g., interpolation to data given at some points in the plane cannot be handled directly by tensor product methods unless these sites happen to form the set of mesh points of a rectangular grid. Still, the computational advantage of using tensor product methods is so large that it pays to bring a problem into this form. for example, by local methods, if this is possible.

Alternatives to tensor product methods use the finite elements, that is, approximating functions that are piecewise polynomials over triangular subdivisions, or quadrilateral subdivisions. There is no hope of giving an adequate description of these techniques in a few pages. We refer the

interested reader to the survey articles by L.L. Schumaker [1976], and R.E.Barnhill [1977], and to two specific, useful examples, in the paper by C. Lawson [1977] and the paper by M.J.D. Powell [1974].

Problems

1. Assume that the LIP on U given by F and Λ is correct. Show that the resulting map P given on U by the rule

$$\lambda Pg = \lambda g, \quad \text{all} \quad \lambda \in \Lambda, \quad \text{and} \quad Pg \in F$$

for $g \in U$ is a linear projector onto F, i.e., a linear idempotent map with range $\operatorname{ran} P = F$.
Show that, conversely, for any linear projector P on U with $\operatorname{ran} P = F$, there exists a linear subspace Λ of linear functionals on U so that Pg is the solution, for given $g \in U$, of the LIP given by F and Λ. (Hint: If $(f_i)_1^n$ is a basis for F, then every $f \in F$ can be written uniquely as $f = \sum_i \alpha_i f_i$. Prove that then $\mu_i : f \mapsto \alpha_i$ is a linear functional on F and consider $\lambda_i := \mu_i P$.)

2. Let I_k denote interpolation at $\tau = (\tau_i)_1^n$ by elements of $\$_{k,\mathbf{t}}$, with $\mathbf{t} = (t_i)_1^{n+k}$ strictly increasing, and $t_i < \tau_i < t_{i+k}$, all i. Also, let $0 < j < k$. Prove that the map $f^{(j)} \mapsto (I_k f)^{(j)}$ is a linear projector with range $\$_{k-j,\mathbf{t}}$ and give a description of the interpolation conditions for the corresponding interpolation scheme (on $U = C(\mathbb{R})$, say). (Hint: Look at the proof of Theorem XIII(34).)

3. Identify least–squares approximation as a linear interpolation scheme.

4. Take the differential equation XV(1)–(2) to be *linear*,

$$(D^m g)(x) = \sum_{i=0}^{m-1} p_i(x)\,(D^i g)(x) \quad \text{for} \quad x \in [a\,..\,b]$$
$$\beta_i g = c_i, \quad i = 1, \ldots, m.$$

Then identify the collocation approximation f in $\$_{m+k,\mathbf{t}}$ to the solution g as an interpolant to f, and describe the interpolation conditions.

5. Write an expanded version
`PPVA2D ( BREAKX, BREAKY, COEF, LX, LY, KX, KY, X, Y, JDX, JDY)`
of `PPVALU`.

6. Construct an expanded version of `L2APPR`. This makes it necessary to make the data arguments (rather than having them come into `L2APPR` via `BLOCK COMMON`), but then should allow the construction of a least–squares bivariate spline approximant to data on a rectangular grid by just two calls to this expanded routine. (Consider an expanded `BCHSLV` for this purpose.)

7. Produce an expanded version of SMOOTH for the smoothing of data on a rectangular grid.

8. Demonstrate that a spline interpolant in r variables can be constructed to data on a hyperrectangular grid by just r calls to the routine SPLI2D given in the text and try it for $r = 3$. (See de Boor [1979] if you have difficulties.)

Postscript on Things Not Covered

This is a brief discussion of some items that a book about splines might well have been expected to cover but that this one did not.

At least a third of the existing spline literature is concerned with the fact that splines are solutions to interesting *variational problems*, yet, with the exception of Theorem XIII(34) and a stab at optimal interpolation, this variational aspect of splines is not mentioned in this book. This reflects my judgement that most of this material is not practical (as yet). The same holds for the related topic of best approximation of linear functionals, especially *optimal quadrature*, which, starting with Sard's work as summarized in Sard [1963], and the work of Golomb & Weinberger [1959], has produced much elegant and, at times, hard mathematics. Again, the many *generalized* splines that arise naturally if one pursues the variational aspect of splines have found no place in this book. An exception should have been made for piecewise exponentials and, perhaps, piecewise rationals. We refer the reader to Späth [1974] and to Schaback [1976], and also to Lyche & Winther [1979] for a recurrence relation for trigonometric B-splines.

The decision to leave out the above mentioned material was made easy by the fact that L.L. Schumaker is in the process of completing a thorough monograph on splines in which the variational approach to splines is fully documented.

It seems somewhat harder to defend the omission of **best approximation** in norms other than those derived from an inner product. There is, for example, a fully developed theory of best approximation by splines with fixed and with free knots in the uniform norm $\|f\| := \max\{|f(x)| : a \le x \le b\}$ on some interval $[a\,..\,b]$; see Rice [1967], Schumaker [1968], Schumaker [1969], and most recent developments concerning the calculation of best uniform spline approximants can be found in Cromme [1976]. But, aside from the fact that I have no practical experience with the construction of best spline approximants (in norms other than $\mathbf{L_2}$), I have not discussed such matters in this book because I am convinced that, for most practical purposes, spline approximants obtained by interpolation (in the general sense described in Chapter XVII) are sufficient. In short, the additional

accuracy gained in a best spline approximation over an available good one is usually not worth the considerable effort required to construct it.

I have no defense but weariness for the failure to provide a fully detailed discussion of **nonlinear splines** (see pp. 280f) that arise when one takes the problem of modeling the draftman's spline seriously. (Cubic spline interpolation has gotten a lot of publicity out of the claim that it models the configuration of a thin elastic beam constrained to go through certain points in the plane. In fact, it does model such a spline only for small slopes.)

I also regret not having included a discussion of **constrained** approximation. This is an area filled with mathematical and computational difficulties. Not only is it hard to give a mathematical description of what people consider "nice" in a fit, but those properties easily recognized as "nice" such as monotonicity or convexity are not all that easy to control.

The book covers only univariate spline approximation, although the easy generalization to multivariate problems by tensor products is given a chapter. This is to me the most painful omission since **multivariate approximation** is, at present, an exiting area of research, and practical results, particularly from the finite element method and from Computer Aided Design, are already available. On the other hand, it is hard to find a multivariate definition of "spline" that fully corresponds to the univariate one adopted in this book. Tensor products of univariate splines, box splines, and thin-plate or, more generally, D^m-splines are three very successful, but very different, multivariate spline concepts.

The book contains only one instance of the use of splines in the solution of functional equations, namely the solution of an ODE by collocation. It should be emphasized that splines have, by their presence, contributed greatly to the resurrection of various variational methods for the solution of ordinary and partial differential equations, such as the (Rayleigh–Ritz–)Galerkin method, the least–squares method and other residual reduction methods; see, for example, Varga [1971].

Fortran programs

Fortran programs All the Fortran programs in this book (see the list below) are presently (1998) available via `netlib` as the package pppack which also contains the example programs that are not explicitly listed here. To get started, send email to `netlib@research.bell-labs.com`, with the body of the letter just the two words `send index`. These Fortran programs also served as the starting point for the SPLINE TOOLBOX (de Boor [1990]$_2$), a supplement to MathWorks' MATLAB, and version 3 of that toolbox contains two GUIs that make experimentation with the basic spline approximation algorithms very easy.

I do not claim, though, that these Fortran programs are production software. In bringing them to their present stage, I have taken care to test them, to provide them with comments (much more extensively than I had planned originally) and to make them reasonably efficient. I have not made them foolproof, though. Also, while their present versions do make use of various FORTRAN 77 features, no advantage is taken of FORTRAN 90 features.

I have tried to keep the argument lists short and consistent. They follow the pattern

input, work area(s), output

and, within each of these groups, arrays precede the integers (if any) specifying their dimensions. In addition, certain variables retain their meaning from program to program. for example, K is always the order of some pp function while L is always the number of polynomial pieces that make up a particular pp function.

List of Fortran programs Here is a list of Fortran programs connected with this book. The list is organized as follows:
name of program, page reference, list of programs calling it, followed by the name of a corresponding command, if any, in the SPLINE TOOLBOX (de Boor [1990]$_2$), followed by a brief statement of the purpose of the program. (Names in parentheses refer to programs not listed in this book.)

BANFAC/SLV, 177, SPLINT, SPLI2D, SPLOPT, XVIEX3
solution of a banded linear system without row interchanges
BCHFAC/SLV, 226/7, L2APPR,
solution of a banded system with positive definite matrix
BSPLPP, 118, COLLOC, L2MAIN, XEX3, XIIEX4, XIIIEX1, XIIIEX2, XIVEX1, XVIEX3; fn2fm(f,'pp')
conversion from B-form to ppform
BSPLVB, 111, BSPLPP, BSPLVD, BSPP2D, L2APPR, SPLINT, SPLI2D, SPLOPT, XEX1, XEX2; spcol
generates value of all B-splines at a site common to their support
BSPLVD, 252, PUTIT, (XVIEX4); spcol
generates values and derivatives of all B-splines at a site common to their support
BSPP2D, 307, (XVIIEX3); fn2fm(f,'pp')
extended version of BSPLPP for multivariate work
(BVALLC), 128,
left continuous version of BVALUE
BVALUE, 121, XEX4, XIIIEX3,(XVIIEX1), XVIIEX2, (XVIIEX3); fnval
generates value of spline or of one of its derivatives from B-form
(CHBINT), 28,
constructs the Chebyshev form of the polynomial interpolating at the Chebyshev sites
(CHBVAL), 16,
evaluates a polynomial at a site from its Chebyshev form
CHOL1D, 213, SMOOTH
subroutine used in SMOOTH
COLLOC, 254, XVEXAMPLE
main routine for the solution of an ODE by collocation
COLPNT, 257, COLLOC
provides the collocation site pattern for COLLOC
CUBSPL, 46, (VIIIEX1), XIIEX2, (XIIEX3), (Problem XII.9); spline, csapi, csape
constructs cubic spline interpolant to given data and for various end conditions
CWIDTH, 326, (Problem XIII.8), (Problem XV.8)
solves an almost block diagonal linear system with constant block width
DIFEQU, 258, COLLOC, PUTIT
specifies differential equation and side conditions for COLLOC
EQBLOK, 248, COLLOC
sets up collocation equations for COLLOC
INTERV, 74, BVALUE, PPVALU, XEX1, XVIIEX2; sorted
locates a site within an increasing sequence of sites
(INTRVL), 77,
version of INTERV suitable for left-continuous pp functions
KNOTS, 256, COLLOC
generates the knot sequence for the approximant in COLLOC
L2APPR, 224, L2MAIN; spap2
constructs least-squares approximation to given data

Listing of SOLVEBLOK package Listed below is an extensively commented version of the SOLVEBLOK package which appeared in C. de Boor & R. Weiss, "SOLVEBLOK: a package for solving almost block diagonal linear systems, with applications to spline approximation and the numerical solution of ordinary differential equations", *ACM Trans. Math. Software* 6, (1980), 80–87.

This package is needed in Chapter XV.

```
      SUBROUTINE SLVBLK ( BLOKS, INTEGS, NBLOKS, B, IPIVOT, X, IFLAG )
C     THIS PROGRAM SOLVES  THE  LINEAR SYSTEM  A*X = B  WHERE A IS AN
C  ALMOST BLOCK DIAGONAL MATRIX.  SUCH ALMOST BLOCK DIAGONAL MATRICES
C  ARISE NATURALLY IN PIECEWISE POLYNOMIAL INTERPOLATION OR APPROX-
C  IMATION AND IN FINITE ELEMENT METHODS FOR TWO-POINT BOUNDARY VALUE
C  PROBLEMS.  THE PLU FACTORIZATION METHOD IS IMPLEMENTED HERE TO TAKE
C  ADVANTAGE OF THE SPECIAL STRUCTURE OF SUCH SYSTEMS FOR SAVINGS IN
C  COMPUTING TIME AND STORAGE REQUIREMENTS.
C
C                    PARAMETERS
C  BLOKS    A ONE-DIMENSIONAL ARRAY, OF LENGTH
C                 SUM( INTEGS(1,I)*INTEGS(2,I) ; I = 1,NBLOKS )
C           ON INPUT, CONTAINS THE BLOCKS OF THE ALMOST BLOCK DIAGONAL
C           MATRIX  A  .  THE ARRAY INTEGS (SEE BELOW AND THE EXAMPLE)
C           DESCRIBES THE BLOCK STRUCTURE.
C           ON OUTPUT, CONTAINS CORRESPONDINGLY THE PLU FACTORIZATION
C           OF  A  (IF IFLAG .NE. 0).  CERTAIN OF THE ENTRIES INTO BLOKS
C           ARE ARBITRARY (WHERE THE BLOCKS OVERLAP).
C  INTEGS   INTEGER ARRAY DESCRIPTION OF THE BLOCK STRUCTURE OF  A .
C             INTEGS(1,I) = NO. OF ROWS OF BLOCK I          =  NROW
C             INTEGS(2,I) = NO. OF COLUMNS OF BLOCK I       =  NCOL
C             INTEGS(3,I) = NO. OF ELIM. STEPS IN BLOCK I =  LAST
C                       I  = 1,2,...,NBLOKS
C           THE LINEAR SYSTEM IS OF ORDER
C                 N  =  SUM ( INTEGS(3,I) , I=1,...,NBLOKS ),
C           BUT THE TOTAL NUMBER OF ROWS IN THE BLOCKS IS
C               NBROWS = SUM( INTEGS(1,I) ; I = 1,...,NBLOKS)
C  NBLOKS   NUMBER OF BLOCKS
C  B        RIGHT SIDE OF THE LINEAR SYSTEM, ARRAY OF LENGTH NBROWS.
C           CERTAIN OF THE ENTRIES ARE ARBITRARY, CORRESPONDING TO
C           ROWS OF THE BLOCKS WHICH OVERLAP (SEE BLOCK STRUCTURE AND
C           THE EXAMPLE BELOW).
C  IPIVOT   ON OUTPUT, INTEGER ARRAY CONTAINING THE PIVOTING SEQUENCE
C           USED. LENGTH IS NBROWS
C  X        ON OUTPUT, CONTAINS THE COMPUTED SOLUTION (IF IFLAG .NE. 0)
C           LENGTH IS N.
C  IFLAG    ON OUTPUT, INTEGER
C             = (-1)**(NO. OF INTERCHANGES DURING FACTORIZATION)
C                   IF  A  IS INVERTIBLE
C             = 0   IF  A  IS SINGULAR
C
C                    AUXILIARY PROGRAMS
C  FCBLOK (BLOKS,INTEGS,NBLOKS,IPIVOT,SCRTCH,IFLAG)  FACTORS THE MATRIX
C           A , AND IS USED FOR THIS PURPOSE IN SLVBLK. ITS ARGUMENTS
C           ARE AS IN SLVBLK, EXCEPT FOR
C               SCRTCH = A WORK ARRAY OF LENGTH MAX(INTEGS(1,I)).
C
C  SBBLOK (BLOKS,INTEGS,NBLOKS,IPIVOT,B,X)  SOLVES THE SYSTEM A*X = B
C           ONCE  A  IS FACTORED. THIS IS DONE AUTOMATICALLY BY SLVBLK
C           FOR ONE RIGHT SIDE B, BUT SUBSEQUENT SOLUTIONS MAY BE
C           OBTAINED FOR ADDITIONAL B-VECTORS. THE ARGUMENTS ARE ALL
C           AS IN SLVBLK.
C
C  DTBLOK (BLOKS,INTEGS,NBLOKS,IPIVOT,IFLAG,DETSGN,DETLOG) COMPUTES THE
C           DETERMINANT OF  A  ONCE SLVBLK OR FCBLOK HAS DONE THE FACT-
C           ORIZATION.THE FIRST FIVE ARGUMENTS ARE AS IN SLVBLK.
C               DETSGN  = SIGN OF THE DETERMINANT
C               DETLOG  = NATURAL LOG OF THE DETERMINANT
C
```

```
C                ------ BLOCK STRUCTURE OF  A  ------.
C  THE NBLOKS BLOCKS ARE STORED CONSECUTIVELY IN THE ARRAY  BLOKS .
C  THE FIRST BLOCK HAS ITS (1,1)-ENTRY AT BLOKS(1), AND, IF THE I-TH
C  BLOCK HAS ITS (1,1)-ENTRY AT BLOKS(INDEX(I)), THEN
C          INDEX(I+1) = INDEX(I)  +  NROW(I)*NCOL(I) .
C    THE BLOCKS ARE PIECED TOGETHER TO GIVE THE INTERESTING PART OF  A
C  AS FOLLOWS.  FOR I = 1,2,...,NBLOKS-1, THE (1,1)-ENTRY OF THE NEXT
C  BLOCK (THE (I+1)ST BLOCK ) CORRESPONDS TO THE (LAST+1,LAST+1)-ENTRY
C  OF THE CURRENT I-TH BLOCK.  RECALL LAST = INTEGS(3,I) AND NOTE THAT
C  THIS MEANS THAT
C       A. EVERY BLOCK STARTS ON THE DIAGONAL OF  A .
C       B. THE BLOCKS OVERLAP (USUALLY). THE ROWS OF THE (I+1)ST BLOCK
C          WHICH ARE OVERLAPPED BY THE I-TH BLOCK MAY BE ARBITRARILY DE-
C          FINED INITIALLY. THEY ARE OVERWRITTEN DURING ELIMINATION.
C    THE RIGHT SIDE FOR THE EQUATIONS IN THE I-TH BLOCK ARE STORED COR-
C  RESPONDINGLY AS THE LAST ENTRIES OF A PIECE OF  B  OF LENGTH  NROW
C  (= INTEGS(1,I)) AND FOLLOWING IMMEDIATELY IN  B  THE CORRESPONDING
C  PIECE FOR THE RIGHT SIDE OF THE PRECEDING BLOCK, WITH THE RIGHT SIDE
C  FOR THE FIRST BLOCK STARTING AT  B(1) . IN THIS, THE RIGHT SIDE FOR
C  AN EQUATION NEED ONLY BE SPECIFIED ONCE ON INPUT, IN THE FIRST BLOCK
C  IN WHICH THE EQUATION APPEARS.
C
C                ------ EXAMPLE AND TEST DRIVER ------
C    THE TEST DRIVER FOR THIS PACKAGE CONTAINS AN EXAMPLE, A LINEAR
C  SYSTEM OF ORDER 11, WHOSE NONZERO ENTRIES ARE INDICATED IN THE FOL-
C  LOWING SCHEMA BY THEIR ROW AND COLUMN INDEX MODULO 10. NEXT TO IT
C  ARE THE CONTENTS OF THE  INTEGS  ARRRAY WHEN THE MATRIX IS TAKEN TO
C  BE ALMOST BLOCK DIAGONAL WITH  NBLOKS = 5, AND BELOW IT ARE THE FIVE
C  BLOCKS.
C
C                          NROW1 = 3, NCOL1 = 4
C            11 12 13 14
C            21 22 23 24    NROW2 = 3, NCOL2 = 3
C            31 32 33 34
C  LAST1 = 2      43 44 45
C                 53 54 55              NROW3 = 3, NCOL3 = 4
C         LAST2 = 3          66 67 68 69    NROW4 = 3, NCOL4 = 4
C                            76 77 78 79       NROW5 = 4, NCOL5 = 4
C                            86 87 88 89
C                  LAST3 = 1   97 98 99 90
C                     LAST4 = 1   08 09 00 01
C                                 18 19 10 11
C                        LAST5 = 4
C
C          ACTUAL INPUT TO BLOKS SHOWN BY ROWS OF BLOCKS OF  A .
C       (THE ** ITEMS ARE ARBITRARY, THIS STORAGE IS USED BY SLVBLK)
C
C  11 12 13 14  / ** ** **  / 66 67 68 69  / ** ** ** **  / ** ** ** **
C  21 22 23 24 /  43 44 45 /  76 77 78 79 /  ** ** ** ** /  ** ** ** **
C  31 32 33 34/   53 54 55/   86 87 88 89/   97 98 99 90/   08 09 00 01
C                                                         18 19 10 11
C
C
C  INDEX = 1       INDEX = 13  INDEX = 22      INDEX = 34      INDEX = 46
C
C          ACTUAL RIGHT SIDE VALUES WITH ** FOR ARBITRARY VALUES
C  B1 B2 B3 ** B4 B5 B6 B7 B8 ** ** B9 ** ** B10 B11
C
C  (IT WOULD HAVE BEEN MORE EFFICIENT TO COMBINE BLOCK 3 WITH BLOCK 4)
C
```

```
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1),IFLAG
      REAL BLOKS(1),B(1),X(1)
C     IN THE CALL TO FCBLOK,  X  IS USED FOR TEMPORARY STORAGE.
      CALL FCBLOK(BLOKS,INTEGS,NBLOKS,IPIVOT,X,IFLAG)
      IF (IFLAG .EQ. 0)                          RETURN
      CALL SBBLOK(BLOKS,INTEGS,NBLOKS,IPIVOT,B,X)
                                                 RETURN
      END
      SUBROUTINE FCBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, SCRTCH, IFLAG )
CALLS SUBROUTINES  F A C T R B  AND  S H I F T B .
C
C   F C B L O K  SUPERVISES THE PLU FACTORIZATION WITH PIVOTING OF
C  SCALED ROWS OF THE ALMOST BLOCK DIAGONAL MATRIX STORED IN THE ARRAYS
C   B L O K S  AND  I N T E G S .
C
C   FACTRB = SUBPROCRAM WHICH CARRIES OUT STEPS 1,...,LAST OF GAUSS
C            ELIMINATION (WITH PIVOTING) FOR AN INDIVIDUAL BLOCK.
C   SHIFTB = SUBPROGRAM WHICH SHIFTS THE REMAINING ROWS TO THE TOP OF
C            THE NEXT BLOCK
C
C PARAMETERS
C    BLOKS   AN ARRAY THAT INITIALLY CONTAINS THE ALMOST BLOCK DIAGONAL
C            MATRIX  A  TO BE FACTORED, AND ON RETURN CONTAINS THE COM-
C            PUTED FACTORIZATION OF  A .
C    INTEGS  AN INTEGER ARRAY DESCRIBING THE BLOCK STRUCTURE OF  A .
C    NBLOKS  THE NUMBER OF BLOCKS IN  A .
C    IPIVOT  AN INTEGER ARRAY OF DIMENSION  SUM (INTEGS(1,N) ; N=1,
C            ...,NBLOKS) WHICH, ON RETURN, CONTAINS THE PIVOTING STRA-
C            TEGY USED.
C    SCRTCH  WORK AREA REQUIRED, OF LENGTH  MAX (INTEGS(1,N) ; N=1,
C            ...,NBLOKS).
C    IFLAG   OUTPUT PARAMETER;
C            = 0  IN CASE MATRIX WAS FOUND TO BE SINGULAR.
C            OTHERWISE,
C            = (-1)**(NUMBER OF ROW INTERCHANGES DURING FACTORIZATION)
C
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1),IFLAG, I,INDEX,INDEXB,INDEXN,
     *        LAST,NCOL,NROW
      REAL BLOKS(1),SCRTCH(1)
      IFLAG = 1
      INDEXB = 1
      INDEXN = 1
      I = 1
C                          LOOP OVER THE BLOCKS.  I  IS LOOP INDEX
   10   INDEX = INDEXN
        NROW = INTEGS(1,I)
        NCOL = INTEGS(2,I)
        LAST = INTEGS(3,I)
C       CARRY OUT ELIMINATION ON THE I-TH BLOCK UNTIL NEXT BLOCK
C       ENTERS, I.E., FOR COLUMNS 1,...,LAST  OF I-TH BLOCK.
        CALL FACTRB(BLOKS(INDEX),IPIVOT(INDEXB),SCRTCH,NROW,NCOL,LAST,
     *              IFLAG)
C        CHECK FOR HAVING REACHED A SINGULAR BLOCK OR THE LAST BLOCK
        IF (IFLAG .EQ. 0 .OR. I .EQ. NBLOKS)
     *                                        RETURN
        I = I+1
        INDEXN = NROW*NCOL + INDEX
C             PUT THE REST OF THE I-TH BLOCK ONTO THE NEXT BLOCK
        CALL SHIFTB(BLOKS(INDEX),IPIVOT(INDEXB),NROW,NCOL,LAST,
     *              BLOKS(INDEXN),INTEGS(1,I),INTEGS(2,I))
        INDEXB = INDEXB + NROW
                                              GO TO 10
      END
```

```
      SUBROUTINE FACTRB ( W, IPIVOT, D, NROW, NCOL, LAST, IFLAG )
C  ADAPTED FROM P.132 OF 'ELEMENT.NUMER.ANALYSIS' BY CONTE-DE BOOR
C
C  CONSTRUCTS A PARTIAL PLU FACTORIZATION, CORRESPONDING TO STEPS 1,...,
C   L A S T   IN GAUSS ELIMINATION, FOR THE MATRIX  W  OF ORDER
C   ( N R O W ,  N C O L ), USING PIVOTING OF SCALED ROWS.
C
C  PARAMETERS
C    W       CONTAINS THE (NROW,NCOL) MATRIX TO BE PARTIALLY FACTORED
C            ON INPUT, AND THE PARTIAL FACTORIZATION ON OUTPUT.
C    IPIVOT  AN INTEGER ARRAY OF LENGTH NROW CONTAINING A RECORD OF THE
C            PIVOTING STRATEGY USED; ROW IPIVOT(I) IS USED DURING THE
C            I-TH ELIMINATION STEP, I=1,...,LAST.
C    D       A WORK ARRAY OF LENGTH NROW USED TO STORE ROW SIZES
C            TEMPORARILY.
C    NROW    NUMBER OF ROWS OF W.
C    NCOL    NUMBER OF COLUMNS OF W.
C    LAST    NUMBER OF ELIMINATION STEPS TO BE CARRIED OUT.
C    IFLAG   ON OUTPUT, EQUALS IFLAG ON INPUT TIMES (-1)**(NUMBER OF
C            ROW INTERCHANGES DURING THE FACTORIZATION PROCESS), IN
C            CASE NO ZERO PIVOT WAS ENCOUNTERED.
C            OTHERWISE, IFLAG = 0 ON OUTPUT.
C
      INTEGER IPIVOT(NROW),NCOL,LAST,IFLAG, I,IPIVI,IPIVK,J,K,KP1
      REAL W(NROW,NCOL),D(NROW), AWIKDI,COLMAX,RATIO,ROWMAX
C  INITIALIZE IPIVOT, D
      DO 10 I=1,NROW
         IPIVOT(I) = I
         ROWMAX = 0.
         DO 9 J=1,NCOL
    9       ROWMAX = AMAX1(ROWMAX, ABS(W(I,J)))
         IF (ROWMAX .EQ. 0.)              GO TO 999
   10    D(I) = ROWMAX
C GAUSS ELIMINATION WITH PIVOTING OF SCALED ROWS, LOOP OVER K=1,..,LAST
      K = 1
C        AS PIVOT ROW FOR K-TH STEP, PICK AMONG THE ROWS NOT YET USED,
C        I.E., FROM ROWS IPIVOT(K),...,IPIVOT(NROW), THE ONE WHOSE K-TH
C        ENTRY (COMPARED TO THE ROW SIZE) IS LARGEST. THEN, IF THIS ROW
C        DOES NOT TURN OUT TO BE ROW IPIVOT(K), REDEFINE IPIVOT(K) AP-
C        PROPRIATELY AND RECORD THIS INTERCHANGE BY CHANGING THE SIGN
C        OF  I F L A G .
   11    IPIVK = IPIVOT(K)
         IF (K .EQ. NROW)                 GO TO 21
         J = K
         KP1 = K+1
         COLMAX = ABS(W(IPIVK,K))/D(IPIVK)
C              FIND THE (RELATIVELY) LARGEST PIVOT
         DO 15 I=KP1,NROW
            IPIVI = IPIVOT(I)
            AWIKDI = ABS(W(IPIVI,K))/D(IPIVI)
            IF (AWIKDI .LE. COLMAX)       GO TO 15
               COLMAX = AWIKDI
               J = I
   15       CONTINUE
         IF (J .EQ. K)                    GO TO 16
         IPIVK = IPIVOT(J)
         IPIVOT(J) = IPIVOT(K)
         IPIVOT(K) = IPIVK
         IFLAG = -IFLAG
   16    CONTINUE
```

```
C       IF PIVOT ELEMENT IS TOO SMALL IN ABSOLUTE VALUE, DECLARE
C       MATRIX TO BE NONINVERTIBLE AND QUIT.
        IF (ABS(W(IPIVK,K))+D(IPIVK) .LE. D(IPIVK))
     *                                    GO TO 999
C       OTHERWISE, SUBTRACT THE APPROPRIATE MULTIPLE OF THE PIVOT
C       ROW FROM REMAINING ROWS, I.E., THE ROWS IPIVOT(K+1),...,
C       IPIVOT(NROW), TO MAKE K-TH ENTRY ZERO. SAVE THE MULTIPLIER IN
C       ITS PLACE.
        DO 20 I=KP1,NROW
           IPIVI = IPIVOT(I)
           W(IPIVI,K) = W(IPIVI,K)/W(IPIVK,K)
           RATIO = -W(IPIVI,K)
           DO 20 J=KP1,NCOL
  20          W(IPIVI,J) = RATIO*W(IPIVK,J) + W(IPIVI,J)
        K = KP1
C       CHECK FOR HAVING REACHED THE NEXT BLOCK.
        IF (K .LE. LAST)                  GO TO 11
                                          RETURN
C     IF  LAST  .EQ. NROW , CHECK NOW THAT PIVOT ELEMENT IN LAST ROW
C     IS NONZERO.
  21  IF( ABS(W(IPIVK,NROW))+D(IPIVK) .GT. D(IPIVK) )
     *                                    RETURN
C                    SINGULARITY FLAG SET
  999 IFLAG = 0
                                          RETURN
      END
      SUBROUTINE SHIFTB ( AI, IPIVOT, NROWI, NCOLI, LAST,
     *                    AI1, NROWI1, NCOLI1 )
C  SHIFTS THE ROWS IN CURRENT BLOCK, AI, NOT USED AS PIVOT ROWS, IF
C  ANY, I.E., ROWS IPIVOT(LAST+1),...,IPIVOT(NROWI), ONTO THE FIRST
C  MMAX = NROW-LAST ROWS OF THE NEXT BLOCK, AI1, WITH COLUMN LAST+J OF
C  AI  GOING TO COLUMN J , J=1,...,JMAX=NCOLI-LAST. THE REMAINING COL-
C  UMNS OF THESE ROWS OF AI1 ARE ZEROED OUT.
C
C                                PICTURE
C
C       ORIGINAL SITUATION AFTER          RESULTS IN A NEW BLOCK I+1
C       LAST = 2 COLUMNS HAVE BEEN        CREATED AND READY TO BE
C       DONE IN FACTRB (ASSUMING NO       FACTORED BY NEXT FACTRB CALL.
C       INTERCHANGES OF ROWS)
C                  1
C            X  X 1X  X  X          X  X  X  X  X
C                  1
C            0  X 1X  X  X          0  X  X  X  X
C  BLOCK I         1                    --------------
C  NROWI = 4 0  0 1X  X  X          0  0 1X  X  X  0  01
C  NCOLI = 5       1                      1              1
C  LAST = 2  0  0 1X  X  X          0  0 1X  X  X  0  01
C  ------------------------------         1              1   NEW
C                  1X  X  X  X  X         1X  X  X  X  X1  BLOCK
C                  1                      1              1   I+1
C  BLOCK I+1       1X  X  X  X  X         1X  X  X  X  X1
C  NROWI1= 5       1                      1              1
C  NCOLI1= 5       1X  X  X  X  X         1X  X  X  X  X1
C  ------------------------------         1-------------1
C                  1
C
      INTEGER IPIVOT(NROWI),LAST, IP,J,JMAX,JMAXP1,M,MMAX
      REAL AI(NROWI,NCOLI),AI1(NROWI1,NCOLI1)
      MMAX = NROWI - LAST
      JMAX = NCOLI - LAST
      IF (MMAX .LT. 1 .OR. JMAX .LT. 1) RETURN
```

```
C                PUT THE REMAINDER OF BLOCK I INTO AI1
      DO 10 M=1,MMAX
         IP = IPIVOT(LAST+M)
         DO 10 J=1,JMAX
   10       AI1(M,J) = AI(IP,LAST+J)
      IF (JMAX .EQ. NCOLI1)                RETURN
C                ZERO OUT THE UPPER RIGHT CORNER OF AI1
      JMAXP1 = JMAX + 1
      DO 20 J=JMAXP1,NCOLI1
         DO 20 M=1,MMAX
   20       AI1(M,J) = 0.
                                              RETURN
      END
      SUBROUTINE SBBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, B, X )
CALLS SUBROUTINES  S U B F O R  AND  S U B B A K .
C
C  SUPERVISES THE SOLUTION (BY FORWARD AND BACKWARD SUBSTITUTION) OF
C  THE LINEAR SYSTEM  A*X = B  FOR X, WITH THE PLU FACTORIZATION OF  A
C  ALREADY GENERATED IN  F C B L O K .  INDIVIDUAL BLOCKS OF EQUATIONS
C  ARE SOLVED VIA  S U B F O R  AND  S U B B A K .
C
C PARAMETERS
C    BLOKS, INTEGS, NBLOKS, IPIVOT    ARE AS ON RETURN FROM FCBLOK.
C    B       THE RIGHT SIDE, STORED CORRESPONDING TO THE STORAGE OF
C            THE EQUATIONS. SEE COMMENTS IN  S L V B L K  FOR DETAILS.
C    X       SOLUTION VECTOR
C
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1), I,INDEX,INDEXB,INDEXX,J,LAST,
     *        NBP1,NCOL,NROW
      REAL BLOKS(1),B(1),X(1)
C
C      FORWARD SUBSTITUTION PASS
C
      INDEX = 1
      INDEXB = 1
      INDEXX = 1
      DO 20 I=1,NBLOKS
         NROW = INTEGS(1,I)
         LAST = INTEGS(3,I)
         CALL SUBFOR(BLOKS(INDEX),IPIVOT(INDEXB),NROW,LAST,B(INDEXB),
     *               X(INDEXX))
         INDEX = NROW*INTEGS(2,I) + INDEX
         INDEXB = INDEXB + NROW
   20    INDEXX = INDEXX + LAST
C
C     BACK SUBSTITUTION PASS
C
      NBP1 = NBLOKS + 1
      DO 30 J=1,NBLOKS
         I = NBP1 - J
         NROW = INTEGS(1,I)
         NCOL = INTEGS(2,I)
         LAST = INTEGS(3,I)
         INDEX = INDEX - NROW*NCOL
         INDEXB = INDEXB - NROW
         INDEXX = INDEXX - LAST
   30    CALL SUBBAK(BLOKS(INDEX),IPIVOT(INDEXB),NROW,NCOL,LAST,
     *               X(INDEXX))
                                              RETURN
      END
```

```
      SUBROUTINE SUBFOR ( W, IPIVOT, NROW, LAST, B, X )
C  CARRIES OUT THE FORWARD PASS OF SUBSTITUTION FOR THE CURRENT BLOCK,
C  I.E., THE ACTION ON THE RIGHT SIDE CORRESPONDING TO THE ELIMINATION
C  CARRIED OUT IN  F A C T R B  FOR THIS BLOCK.
C     AT THE END, X(J) CONTAINS THE RIGHT SIDE OF THE TRANSFORMED
C  IPIVOT(J)-TH EQUATION IN THIS BLOCK, J=1,...,NROW. THEN, SINCE
C  FOR I=1,...,NROW-LAST, B(NROW+I) IS GOING TO BE USED AS THE RIGHT
C  SIDE OF EQUATION  I  IN THE NEXT BLOCK (SHIFTED OVER THERE FROM
C  THIS BLOCK DURING FACTORIZATION), IT IS SET EQUAL TO X(LAST+I) HERE.
C
C PARAMETERS
C    W, IPIVOT, NROW, LAST  ARE AS ON RETURN FROM FACTRB.
C    B(J)   IS EXPECTED TO CONTAIN, ON INPUT, THE RIGHT SIDE OF J-TH
C           EQUATION FOR THIS BLOCK, J=1,...,NROW.
C    B(NROW+J)   CONTAINS, ON OUTPUT, THE APPROPRIATELY MODIFIED RIGHT
C           SIDE FOR EQUATION J IN NEXT BLOCK, J=1,...,NROW-LAST.
C    X(J)   CONTAINS, ON OUTPUT, THE APPROPRIATELY MODIFIED RIGHT
C           SIDE OF EQUATION IPIVOT(J) IN THIS BLOCK, J=1,...,LAST (AND
C           EVEN FOR J=LAST+1,...,NROW).
C
      INTEGER IPIVOT(NROW), IP,JMAX,K
C     DIMENSION B(NROW + NROW-LAST)
      REAL W(NROW,LAST),B(1),X(NROW)
      IP = IPIVOT(1)
      X(1) = B(IP)
      IF (NROW .EQ. 1)                    GO TO 99
      DO 15 K=2,NROW
         IP = IPIVOT(K)
         JMAX = AMINO(K-1,LAST)
         SUM = 0.
         DO 14 J=1,JMAX
   14       SUM = W(IP,J)*X(J) + SUM
   15    X(K) = B(IP) - SUM
C
C     TRANSFER MODIFIED RIGHT SIDES OF EQUATIONS IPIVOT(LAST+1),...,
C     IPIVOT(NROW) TO NEXT BLOCK.
      NROWML = NROW - LAST
      IF (NROWML .EQ. 0)                  GO TO 99
      LASTP1 = LAST+1
      DO 25 K=LASTP1,NROW
   25    B(NROWML+K) = X(K)
   99                                     RETURN
      END
      SUBROUTINE SUBBAK ( W, IPIVOT, NROW, NCOL, LAST, X )
C  CARRIES OUT BACKSUBSTITUTION FOR CURRENT BLOCK.
C
C PARAMETERS
C    W, IPIVOT, NROW, NCOL, LAST  ARE AS ON RETURN FROM FACTRB.
C    X(1),...,X(NCOL)  CONTAINS, ON INPUT, THE RIGHT SIDE FOR THE
C           EQUATIONS IN THIS BLOCK AFTER BACKSUBSTITUTION HAS BEEN
C           CARRIED UP TO BUT NOT INCLUDING EQUATION IPIVOT(LAST).
C           MEANS THAT X(J) CONTAINS THE RIGHT SIDE OF EQUATION IPI-
C           VOT(J) AS MODIFIED DURING ELIMINATION, J=1,...,LAST, WHILE
C           FOR J .GT. LAST, X(J) IS ALREADY A COMPONENT OF THE SOLUT-
C           ION VECTOR.
C    X(1),...,X(NCOL) CONTAINS, ON OUTPUT, THE COMPONENTS OF THE SOLUT-
C           ION CORRESPONDING TO THE PRESENT BLOCK.
C
      INTEGER IPIVOT(NROW),LAST,  IP,J,K,KP1
      REAL W(NROW,NCOL),X(NCOL), SUM
      K = LAST
      IP = IPIVOT(K)
      SUM = 0.
      IF (K .EQ. NCOL)                    GO TO 4
      KP1 = K+1
    2    DO 3 J=KP1,NCOL
```

```
3        SUM = W(IP,J)*X(J) + SUM
4     X(K) = (X(K) - SUM)/W(IP,K)
      IF (K .EQ. 1)                          RETURN
      KP1 = K
      K = K-1
      IP = IPIVOT(K)
      SUM = 0.
                                             GO TO 2
      END
      SUBROUTINE DTBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, IFLAG,
     *                    DETSGN, DETLOG )
C  COMPUTES THE DETERMINANT OF AN ALMOST BLOCK DIAGONAL MATRIX WHOSE
C  PLU FACTORIZATION HAS BEEN OBTAINED PREVIOUSLY IN FCBLOK.
C  *** THE LOGARITHM OF THE DETERMINANT IS COMPUTED INSTEAD OF THE
C  DETERMINANT ITSELF TO AVOID THE DANGER OF OVERFLOW OR UNDERFLOW
C  INHERENT IN THIS CALCULATION.
C
C PARAMETERS
C    BLOKS, INTEGS, NBLOKS, IPIVOT, IFLAG  ARE AS ON RETURN FROM FCBLOK.
C             IN PARTICULAR, IFLAG = (-1)**(NUMBER OF INTERCHANGES DUR-
C             ING FACTORIZATION) IF SUCCESSFUL, OTHERWISE IFLAG = 0.
C    DETSGN  ON OUTPUT, CONTAINS THE SIGN OF THE DETERMINANT.
C    DETLOG  ON OUTPUT, CONTAINS THE NATURAL LOGARITHM OF THE DETERMI-
C             NANT IF DETERMINANT IS NOT ZERO. OTHERWISE CONTAINS 0.
C
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1),IFLAG, I,INDEXP,IP,K,LAST
      REAL BLOKS(1),DETSGN,DETLOG
C
      DETSGN = IFLAG
      DETLOG = 0.
      IF (IFLAG .EQ. 0)                      RETURN
      INDEX = 0
      INDEXP = 0
      DO 2 I=1,NBLOKS
         NROW = INTEGS(1,I)
         LAST = INTEGS(3,I)
         DO 1 K=1,LAST
            IP = INDEX + NROW*(K-1) + IPIVOT(INDEXP+K)
            DETLOG = DETLOG + ALOG(ABS(BLOKS(IP)))
1           DETSGN = DETSGN*SIGN(1.,BLOKS(IP))
         INDEX = NROW*INTEGS(2,I) + INDEX
2        INDEXP = INDEXP + NROW
                                             RETURN
      END
```

Here is a subroutine for solving linear systems involving B-splines which uses less storage than does SLVBLK, but does not save the factorization.

```
      SUBROUTINE CWIDTH ( W,B,NEQU,NCOLS,INTEGS,NBLOKS, D, X,IFLAG )
C  THIS PROGRAM IS A VARIATION OF THE THEME IN THE ALGORITHM BANDET1
C  BY MARTIN AND WILKINSON (NUMER.MATH. 9(1976)279-307). IT SOLVES
C  THE LINEAR SYSTEM
C                              A*X = B
C  OF NEQU EQUATIONS IN CASE A IS ALMOST BLOCK DIAGONAL WITH ALL
C  BLOCKS HAVING NCOLS COLUMNS USING NO MORE STORAGE THAN IT TAKES TO
C  STORE THE INTERESTING PART OF A . SUCH SYSTEMS OCCUR IN THE DETERM-
C  INATION OF THE B-SPLINE COEFFICIENTS OF A SPLINE APPROXIMATION.
C
C                              PARAMETERS
C  W     ON INPUT, A TWO-DIMENSIONAL ARRAY OF SIZE (NEQU,NCOLS) CONTAIN-
```

```
C         ING THE INTERESTING PART OF THE ALMOST BLOCK DIAGONAL COEFFICI-
C         ENT MATRIX  A (SEE DESCRIPTION AND EXAMPLE BELOW). THE ARRAY
C         INTEGS  DESCRIBES THE STORAGE SCHEME.
C         ON OUTPUT, W  CONTAINS THE UPPER TRIANGULAR FACTOR  U  OF THE
C         LU FACTORIZATION OF A POSSIBLY PERMUTED VERSION OF  A . IN PAR-
C         TICULAR, THE DETERMINANT OF  A  COULD NOW BE FOUND AS
C             IFLAG*W(1,1)*W(2,1)* ... * W(NEQU,1)  .
C  B      ON INPUT, THE RIGHT SIDE OF THE LINEAR SYSTEM, OF LENGTH  NEQU.
C         THE CONTENTS OF  B  ARE CHANGED DURING EXECUTION.
C  NEQU   NUMBER OF EQUATIONS IN SYSTEM
C  NCOLS  BLOCK WIDTH, I.E., NUMBER OF COLUMNS IN EACH BLOCK.
C  INTEGS  INTEGER ARRAY, OF SIZE (2,NEQU), DESCRIBING THE BLOCK STRUCT-
C         URE OF  A .
C            INTEGS(1,I) = NO. OF ROWS IN BLOCK I                  =  NROW
C            INTEGS(2,I) = NO. OF ELIMINATION STEPS IN BLOCK I
C                        = OVERHANG OVER NEXT BLOCK                =  LAST
C  NBLOKS  NUMBER OF BLOCKS
C  D      WORK ARRAY, TO CONTAIN ROW SIZES . IF STORAGE IS SCARCE, THE
C         ARRAY  X  COULD BE USED IN THE CALLING SEQUENCE FOR  D .
C  X      ON OUTPUT, CONTAINS COMPUTED SOLUTION (IF IFLAG .NE. 0), OF
C         LENGTH  NEQU .
C  IFLAG   ON OUTPUT, INTEGER
C          = (-1)**(NO.OF INTERCHANGES DURING ELIMINATION)
C                IF  A  IS INVERTIBLE
C          =  0   IF  A  IS SINGULAR
C
C         ------  BLOCK STRUCTURE OF  A  ------
C     THE INTERESTING PART OF  A  IS TAKEN TO CONSIST OF  NBLOKS  CON-
C  SECUTIVE BLOCKS, WITH THE I-TH BLOCK MADE UP OF  NROWI = INTEGS(1,I)
C  CONSECUTIVE ROWS AND  NCOLS  CONSECUTIVE COLUMNS OF  A , AND WITH
C  THE FIRST  LASTI = INTEGS(2,I) COLUMNS TO THE LEFT OF THE NEXT BLOCK.
C  THESE BLOCKS ARE STORED CONSECUTIVELY IN THE WORKARRAY  W .
C     FOR EXAMPLE, HERE IS AN 11TH ORDER MATRIX AND ITS ARRANGEMENT IN
C  THE WORKARRAY  W . (THE INTERESTING ENTRIES OF  A  ARE INDICATED BY
C  THEIR ROW AND COLUMN INDEX MODULO 10.)
C
C                    ---   A   ---                              ---   W   ---
C
C                       NROW1=3
C           11 12 13 14                                         11 12 13 14
C           21 22 23 24                                         21 22 23 24
C           31 32 33 34       NROW2=2                           31 32 33 34
C   LAST1=2       43 44 45 46                                   43 44 45 46
C                 53 54 55 56          NROW3=3                  53 54 55 56
C         LAST2=3          66 67 68 69                         66 67 68 69
C                          76 77 78 79                         76 77 78 79
C                          86 87 88 89    NROW4=1              86 87 88 89
C                  LAST3=1   97 98 99 90    NROW5=2           97 98 99 90
C                     LAST4=1   08 09 00 01                    08 09 00 01
C                               18 19 10 11                    18 19 10 11
C                        LAST5=4
C
C  FOR THIS INTERPRETATION OF  A  AS AN ALMOST BLOCK DIAGONAL MATRIX,
C  WE HAVE  NBLOKS = 5 , AND THE INTEGS ARRAY IS
C
C                              I=  1   2   3   4   5
C                        K=
C  INTEGS(K,I) =          1       3   2   3   1   2
C                         2       2   3   1   1   4
C
C          --------  METHOD  --------
C     GAUSS ELIMINATION WITH SCALED PARTIAL PIVOTING IS USED, BUT MULT-
C  IPLIERS ARE  N O T  S A V E D  IN ORDER TO SAVE STORAGE. RATHER, THE
C  RIGHT SIDE IS OPERATED ON DURING ELIMINATION.
C     THE TWO PARAMETERS
```

```
C                    I P V T E Q   AND  L A S T E Q
C  ARE USED TO KEEP TRACK OF THE ACTION.  IPVTEQ IS THE INDEX OF THE
C  VARIABLE TO BE ELIMINATED NEXT, FROM EQUATIONS  IPVTEQ+1,...,LASTEQ,
C  USING EQUATION  IPVTEQ (POSSIBLY AFTER AN INTERCHANGE) AS THE PIVOT
C  EQUATION. THE ENTRIES IN THE PIVOT COLUMN ARE  A L W A Y S  IN COLUMN
C  1 OF  W . THIS IS ACCOMPLISHED BY PUTTING THE ENTRIES IN ROWS
C  IPVTEQ+1,...,LASTEQ  REVISED BY THE ELIMINATION OF THE  IPVTEQ-TH
C  VARIABLE ONE TO THE LEFT IN  W . IN THIS WAY, THE COLUMNS OF THE
C  EQUATIONS IN A GIVEN BLOCK (AS STORED IN  W ) WILL BE ALIGNED WITH
C  THOSE OF THE NEXT BLOCK AT THE MOMENT WHEN THESE NEXT EQUATIONS BE-
C  COME INVOLVED IN THE ELIMINATION PROCESS.
C     THUS, FOR THE ABOVE EXAMPLE, THE FIRST ELIMINATION STEPS PROCEED
C  AS FOLLOWS.
C
C  *11 12 13 14     11 12 13 14     11 12 13 14     11 12 13 14
C  *21 22 23 24    *22 23 24        22 23 24        22 23 24
C  *31 32 33 34    *32 33 34       *33 34           33 34
C   43 44 45 46     43 44 45 46    *43 44 45 46    *44 45 46          ETC.
C   53 54 55 56     53 54 55 56    *53 54 55 56    *54 55 56
C   66 67 68 69     66 67 68 69     66 67 68 69     66 67 68 69
C        .               .               .               .
C
C     IN ALL OTHER RESPECTS, THE PROCEDURE IS STANDARD, INCLUDING THE
C  SCALED PARTIAL PIVOTING.
C
      INTEGER IFLAG,INTEGS(2,NBLOKS),NCOLS,NEQU,   I,II,ICOUNT,IPVTEQ
     *                       ,ISTAR,J,LASTCL,LASTEQ,LASTI,NEXTEQ,NROWAD
      REAL B(NEQU),D(NEQU),W(NEQU,NCOLS),X(NEQU),   AWI1OD,COLMAX
     *                                             ,ROWMAX,TEMP
      IFLAG = 1
      IPVTEQ = 0
      LASTEQ = 0
C                                   THE I-LOOP RUNS OVER THE BLOCKS
      DO 50 I=1,NBLOKS
C
C        THE EQUATIONS FOR THE CURRENT BLOCK ARE ADDED TO THOSE CURRENT-
C        LY INVOLVED IN THE ELIMINATION PROCESS, BY INCREASING  LASTEQ
C        BY  INTEGS(1,I) AFTER THE ROWSIZE OF THESE EQUATIONS HAS BEEN
C        RECORDED IN THE ARRAY  D .
C
         NROWAD = INTEGS(1,I)
         DO 10 ICOUNT=1,NROWAD
            NEXTEQ = LASTEQ + ICOUNT
            ROWMAX = 0.
            DO 5 J=1,NCOLS
    5          ROWMAX = AMAX1(ROWMAX,ABS(W(NEXTEQ,J)))
            IF (ROWMAX .EQ. 0.)          GO TO 999
   10       D(NEXTEQ) = ROWMAX
         LASTEQ = LASTEQ + NROWAD
C
C        THERE WILL BE  LASTI = INTEGS(2,I)  ELIMINATION STEPS BEFORE
C        THE EQUATIONS IN THE NEXT BLOCK BECOME INVOLVED. FURTHER,
C        L A S T C L  RECORDS THE NUMBER OF COLUMNS INVOLVED IN THE CUR-
C        RENT ELIMINATION STEP. IT STARTS EQUAL TO  NCOLS  WHEN A BLOCK
C        FIRST BECOMES INVOLVED AND THEN DROPS BY ONE AFTER EACH ELIM-
C        INATION STEP.
C
         LASTCL = NCOLS
         LASTI = INTEGS(2,I)
         DO 30 ICOUNT=1,LASTI
            IPVTEQ = IPVTEQ + 1
            IF (IPVTEQ .LT. LASTEQ)     GO TO 11
            IF ( ABS(W(IPVTEQ,1))+D(IPVTEQ) .GT. D(IPVTEQ) )
     *                                  GO TO 50
                                        GO TO 999

C
```

```
C         DETERMINE THE SMALLEST  I S T A R  IN  (IPVTEQ,LASTEQ)  FOR
C         WHICH  ABS(W(ISTAR,1))/D(ISTAR)  IS AS LARGE AS POSSIBLE, AND
C         INTERCHANGE EQUATIONS  IPVTEQ  AND  ISTAR  IN CASE  IPVTEQ
C         .LT. ISTAR .
C
   11        COLMAX = ABS(W(IPVTEQ,1))/D(IPVTEQ)
             ISTAR = IPVTEQ
             IPVTP1 = IPVTEQ + 1
             DO 13 II=IPVTP1,LASTEQ
                AWI10D = ABS(W(II,1))/D(II)
                IF (AWI10D .LE. COLMAX)  GO TO 13
                COLMAX = AWI10D
                ISTAR = II
   13           CONTINUE
             IF ( ABS(W(ISTAR,1))+D(ISTAR) .EQ. D(ISTAR) )
               *                                 GO TO 999
             IF (ISTAR .EQ. IPVTEQ)              GO TO 16
             IFLAG = -IFLAG
             TEMP = D(ISTAR)
             D(ISTAR) = D(IPVTEQ)
             D(IPVTEQ) = TEMP
             TEMP = B(ISTAR)
             B(ISTAR) = B(IPVTEQ)
             B(IPVTEQ) = TEMP
             DO 14 J=1,LASTCL
                TEMP = W(ISTAR,J)
                W(ISTAR,J) = W(IPVTEQ,J)
   14           W(IPVTEQ,J) = TEMP
C
C         SUBTRACT THE APPROPRIATE MULTIPLE OF EQUATION  IPVTEQ  FROM
C         EQUATIONS  IPVTEQ+1,...,LASTEQ TO MAKE THE COEFFICIENT OF THE
C         IPVTEQ-TH UNKNOWN (PRESENTLY IN COLUMN 1 OF  W ) ZERO, BUT
C         STORE THE NEW COEFFICIENTS IN  W  ONE TO THE LEFT FROM THE OLD.
C
   16        DO 20 II=IPVTP1,LASTEQ
                RATIO = W(II,1)/W(IPVTEQ,1)
                DO 18 J=2,LASTCL
   18              W(II,J-1) = W(II,J) - RATIO*W(IPVTEQ,J)
                W(II,LASTCL) = 0.
   20           B(II) = B(II) - RATIO*B(IPVTEQ)
   30        LASTCL = LASTCL - 1
   50     CONTINUE
C
C  AT THIS POINT,  W  AND  B  CONTAIN AN UPPER TRIANGULAR LINEAR SYSTEM
C  EQUIVALENT TO THE ORIGINAL ONE, WITH  W(I,J) CONTAINING ENTRY
C  (I, I-1+J ) OF THE COEFFICIENT MATRIX. SOLVE THIS SYSTEM BY BACKSUB-
C  STITUTION, TAKING INTO ACCOUNT ITS BLOCK STRUCTURE.
C
C                                 I-LOOP OVER THE BLOCKS, IN REVERSE ORDER
      I = NBLOKS
   59    LASTI = INTEGS(2,I)
         JMAX = NCOLS - LASTI
         DO 70 ICOUNT=1,LASTI
            SUM = 0.
            IF (JMAX .EQ. 0)                 GO TO 61
            DO 60 J=1,JMAX
   60          SUM = SUM + X(IPVTEQ+J)*W(IPVTEQ,J+1)
   61       X(IPVTEQ) = (B(IPVTEQ)-SUM)/W(IPVTEQ,1)
            JMAX = JMAX + 1
   70       IPVTEQ = IPVTEQ - 1
         I = I - 1
         IF (I .GT. 0)                       GO TO 59
                                             RETURN
  999 IFLAG = 0
                                             RETURN
      END
```

Bibliography

* J. H. AHLBERG, E. N. NILSON, AND J. L. WALSH [1967], *The Theory of Splines and Their Applications*, Academic Press (New York).

H. AKIMA [1970], "A new method of interpolation and smooth curve fitting based on local procedures", *J. Assoc. Comput. Mach.* **17**, 589–602; *p. 42.*

R. E. BARNHILL [1977], "Representation and approximation of surfaces", in *Mathematical Software III* (J. R. Rice, ed), Academic Press (New York), 68–119; *p. 311.*

* R. E. BARNHILL AND R. F. RIESENFELD (eds.) [1974], *Computer Aided Geometric Design*, Academic Press (New York); *p. 141.*

D. L. BARROW AND P. W. SMITH [1978], "Asymptotic properties of best $L_2[0,1]$ approximation by splines with variable knots", *Quart. Appl. Math.* **36**, 293–304; *p. 157.*

G. BIRKHOFF AND C. DE BOOR [1964], "Error bounds for spline interpolation", *J. Math. Mech.* **13**, 827–835; *p. 282.*

G. BIRKHOFF AND C. R. DE BOOR [1965], "Piecewise polynomial interpolation and approximation", in *Approximation of Functions* (H. L. Garabedian, ed), Elsevier (New York), 164–190; *p. 281.*

WOLFGANG BOEHM [1980], "Inserting new knots into B-spline curves", *Computer-Aided Design* **12(4)**, 199–201; *p. 135, 137, 138.*

* K. BÖHMER [1974], *Spline-Funktionen*, Teubner (Stuttgart).

* K. BÖHMER, G. MEINARDUS, AND W. SCHEMPP [1974], *Tagung über Spline-Funktionen*, Bibliographisches Institut (Mannheim).

L. I. BONEVA, D. G. KENDALL, AND I. STEFANOV [1971], "Spline transformations: Three new diagnostic aids for the statistical data-analyst", *J. of the Royal Statistical Soc. Series B* **33**, 1–70; *p. 50.*

C. DE BOOR [1966], "The method of projections as applied to the numerical solution of two point boundary value problems using cubic splines", dissertation, Univ. Michigan; *p. 44.*

C. DE BOOR [1968], "On uniform approximation by splines", *J. Approx. Theory* **1**, 219–235; *p. 132, 154.*

C. DE BOOR [1971], "Subroutine package for calculating with B-splines", Techn.Rep. LA-4728-MS, Los Alamos Sci.Lab, Los Alamos NM; *p. 109, 121.*

C. DE BOOR [1972], "On calculating with *B*-splines", *J. Approx. Theory* **6**, 50–62; *p. 124, 138.*

C. DE BOOR [1973], "Good approximation by splines with variable knots", in *Spline Functions and Approximation Theory, ISNM 21* (A. Meir and A. Sharma, eds), Birkhäuser Verlag (Basel), 57–72; *p. 36, 173.*

C. DE BOOR [1974], "Good approximation by splines with variable knots. II", in *Numerical Solution of Differential Equations* (G. A. Watson, ed), Springer (Berlin), 12–20; *p. 158.*

C. DE BOOR [1975], "On bounding spline interpolation", *J. Approx. Theory* **14**, 191–203; *p. 182, 185.*

C. DE BOOR [1976]$_1$, "Total positivity of the spline collocation matrix", *Indiana Univ. Math. J.* **25**, 541–551; *p. 138, 172, 175, 201.*

C. DE BOOR [1976]$_2$, "On local linear functionals which vanish at all *B*-splines but one", in *Theory of Approximation with Applications* (A. G. Law and N. B. Sahney, eds), Academic Press (New York), 120–145; *p. 132, 154, 156.*

C. DE BOOR [1976]$_3$, "A bound on the L_∞-norm of L_2-approximation by splines in terms of a global mesh ratio", *Math. Comp.* **30(136)**, 765–771; *p. 182.*

C. DE BOOR [1976]$_4$, "On 'best' interpolation", *J. Approx. Theory* **16**, 28–42; *p. 199.*

C. DE BOOR [1977]$_1$, "Package for calculating with B-splines", *SIAM J. Numer. Anal.* **14**, 441–472; *p. 121.*

C. DE BOOR [1977]$_2$, "Computational aspects of optimal recovery", in *Optimal Estimation in Approximation Theory* (C. Micchelli and T. Rivlin, eds), Plenum (New York), 69–91; *p. 193.*

C. DE BOOR [1979], "Efficient computer manipulation of tensor products", *ACM Trans. Math. Software* **5**, 173–182. *Corrigenda:* 525; *p. 299, 312.*

C. DE BOOR [1980], "A taut cubic spline", to appear (ms); *p. 276.*

C. DE BOOR [1990]$_1$, "The exact condition of the B-spline basis may be hard to determine", *J. Approx. Theory* **60**, 344–359; *p. 132.*
rhl$_2$

* C. DE BOOR [1990]$_1$, *Spline Toolbox (for use with MATLAB)*, The MathWorks, Inc. (21 Eliot St., South Natick MA 01760); *p. 1, 71, 75, 96, 100, 101, 192, 278, 315.*

C. DE BOOR [1993], "B(asic)-spline basics", in *Fundamental Developments of Computer-Aided Geometric Modeling* (Les Piegl, ed), Academic Press (London), 27–49; *p. 88.*

C. DE BOOR [1997], "The multiplicity of a spline zero", *Annals of Numerical Mathematics* 4, 229–238; *p. 173.*

C. DE BOOR AND G. J. FIX [1973], "Spline approximation by quasi-interpolants", *J. Approx. Theory* **8**, 19–45; *p. 102, 155.*

C. DE BOOR AND K. HÖLLIG [1987], "*B*-splines without divided differences", in *Geometric Modeling: Algorithms and New Trends* (G. E. Farin, ed), SIAM Publications (Philadelphia), 21–27; *p. 88.*

C. DE BOOR AND A. PINKUS [1977], "Backward error analysis for totally positive linear systems", *Numer. Math.* **27**, 485–490; *p. 175.*

C. DE BOOR AND J. R. RICE [1968], "Least squares cubic spline approximation I. Fixed knots/II. Variable knots", Comp.Sci.Dpt. TR 20/21, Purdue University (available at `ftp.cs.wisc.edu/Approx`) ; *p. 239.*

C. DE BOOR AND B. SWARTZ [1973], "Collocation at Gaussian points", *SIAM J. Numer. Anal.* **10**, 582–606; *p. 243, 245.*

L. BOS AND K. SALKAUSKAS [1992], "Weighted splines based on piecewise polynomial weight functions", in *Curve and Surface Design* (H. Hagen, ed), SIAM Publications, SIAM (Philadelphia PA), 87–98; *p. 242.*

GUIDO BRUNNETT [1992], "Properties of minimal-energy splines", in *Curve and Surface Design* (H. Hagen, ed), SIAM Publications, SIAM (Philadelphia PA), 3–22; *p. 280.*

L. BRUTMAN [1978], "On the Lebesgue function for polynomial interpolation", *SIAM J. Numer. Anal.* **15**, 694–704; *p. 21.*

L. BRUTMAN [1997], "Lebesgue functions for polynomial interpolation – a survey", *Annals of Numerical Mathematics* **4**, 111–127; *p. 20.*

O. BUNEMAN [1973], "On-line spline fitting", Stanford University Institute for Plasma Research Rpt. 507 (Palo Alto CA); *p. 67.*

H. G. BURCHARD [1974], "Splines (with optimal knots) are better", *Appl. Anal.* 3, 309–319; *p. 163.*

H. BURCHARD [1977], "On the degree of convergence of piecewise polynomial approximation on optimal meshes", *Trans. Amer. Math. Soc.* **234**, 531–559; *p. 158.*

K. R. BUTTERFIELD [1976], "The computation of all the derivatives of a B-spline basis", *J. Inst. Math. Applics.* **17**, 15–25; *p. 251, 252.*

C. CARASSO AND P. J. LAURENT [1969], "On the numerical construction and the practical use of interpolating Spline Functions", in *Info. Processing 68, Vol. 1* (xxx, ed), North Holland (Amsterdam), 86–89; *p. 81.*

G. F. CARRIER [1970], "Singular perturbation theory and geophysics", *SIAM Review* **12**, 175–193; *p. 258.*

P. DE CASTELJAU [1963], "Courbes et Surfaces à Pôles", André Citroën Automobiles SA, Paris; *p. 144.*

A. K. CLINE [1974], "Scalar- and planar-valued curve fitting in one and two dimensions using splines under tension", *Comm. ACM* **17**, 218–223 ; *p. 264.*

* S. CONTE AND C. DE BOOR [1980], *Elementary Numerical Analysis, 3rd Edition*, xii + 428p, McGraw-Hill (New York); *p. 1, 6, 12, 16, 180.*

M. G. COX [1972], "The numerical evaluation of B-splines", *J. Inst. Math. Applics.* **10**, 134–149; *p. 109.*

PETER CRAVEN AND GRACE WAHBA [1979], "Smoothing noisy data with spline functions estimating the correct degree of smoothing by the method of generalized cross validation", *Numer. Math.* **31**, 377–403; *p. 214.*

LUDWIG J. CROMME [1976], "Eine Klasse von Verfahren zur Ermittlung bester nichtlinearer Tschebyscheff-Approximation", *Numer. Math.* **25**, 447–459; *p. 313.*

H. B. CURRY AND I. J. SCHOENBERG [1947], "On spline distributions and their limits: The Polya distribution functions", *Bull. Amer. Math. Soc.* **53**, 1114; *p. 87.*

H. B. CURRY AND I. J. SCHOENBERG [1966], "On Pólya frequency functions IV: the fundamental spline functions and their limits", *J. Analyse Math.* **17**, 71–107; *p. 98.*

S. DEMKO [1985], "On the existence of interpolating projections onto spline spaces", *J. Approx. Theory* **43**, 151–156; *p. 189.*

R. DEVORE AND F. RICHARDS [1973], "The degree of approximation by Chebyshevian splines", *Trans. Amer. Math. Soc.* **181**, 401–418; *p. 149.*

D. S. DODSON [1972], "Optimal order approximation by polynomial spline functions", dissertation, Purdue Univ.; *p. 163, 258.*

A. EAGLE [1928], "On the relations between Fourier constants of a periodic function and the coefficients determined by harmonic analysis", *Philos. Mag.* **5**, 113–132; *p. 288.*

J. FAVARD [1940], "Sur l'interpolation", *J. Math. Pures Appl.* **19**, 281–306; *p. 199, 200.*

* GEORGE E. FORSYTHE AND CLEVE B. MOLER [1967], *Computer Methods of Linear Algebraic Systems*, Prentice-Hall (Englewood Cliffs NJ); *p. 34, 43, 180, 222, 223, 299.*

P. W. GAFFNEY AND M. J. D. POWELL [1982], "Optimal interpolation", in *Numerical Analysis Dundee, 1981* (G. A. Watson, ed), Springer Lecture Notes 912 (Berlin), 90–99; *p. 199.*

W. GAUTSCHI $[1972]_1$, "Attenuation factors in practical Fourier analysis", *Numer. Math.* **18**, 373–400; *p. 282.*

W. GAUTSCHI [1972]$_2$, "The condition of orthogonal polynomials", *Math. Comp.* **26**, 923–924; *p. 14.*

* M. GOLOMB [1962], *Lectures on Theory of Approximation*, Applied Math. Division, Argonne National Laboratory (Argonne IL).

M. GOLOMB AND H. F. WEINBERGER [1959], "Optimal approximation and error bounds", in *On Numerical Approximation* (R. E. Langer, ed), U. Wis. Press (Madison), 117–190; *p. 199, 313.*

T. N. T. GOODMAN [1994], "New bounds on the zeros of spline functions", *J. Approx. Theory* **76(1)**, 123–130; *p. 172, 173.*

* T. N. E. GREVILLE (ed.) [1969], *Theory and Applications of Spline Functions*, Academic Press (New York).

C. A. HALL [1968], "On error bounds for spline interpolation", *J. Approx. Theory* **1**, 209–218; *p. 55, 56.*

C. A. HALL AND W. W. MEYER [1976], "Optimal error bounds for cubic spline interpolation", *J. Approx. Theory* **16**, 105–122; *p. 55, 56.*

* J. G. HAYES (ed.) [1970], *Numerical Approximation to Functions and Data*, Athlone Press (London); *p. 45, 237, 241.*

K. ICHIDA, F. YOSHIMOTO, AND T. KIYOMO [1977], "Curve fitting by a one-pass method with a piecewise cubic polynomial", *ACM Trans. Math. Software* **3**, 164–174; *p. 241.*

IMSL, International Mathematical and Statistical Libraries, Inc, Computer Subroutines Libraries in Mathematics and Statistics, April 1977, Sixth Floor, GNB Building, 7500 Bellaire Blvd, Houston TX 77036, (713)772–1927. ; *p. 42, 208, 239.*

* E. ISAACSON AND H. B. KELLER [1966], *Analysis of Numerical Methods*, Wiley (New York); *p. 1, 6.*

RONG-QING JIA [1988], "Spline interpolation at knot averages", *Constr. Approx.* **4**, 1–7; *p. 185, 189.*

D. L. JUPP [1978], "Approximation to data by splines with free knots", *SIAM J. Numer. Anal.* **15**, 328–343; *p. 239.*

* S. KARLIN [1968], *Total Positivity*, Stanford Univ. Press (Stanford); *p. 139, 175, 232.*

* S. KARLIN AND W. J. STUDDEN [1966], *Tschebycheff Systems: With Applications in Analysis and Statistics*, Interscience (New York).

S. KARLIN AND Z. ZIEGLER [1966], "Chebyshevian spline functions", *SIAM J. Numer. Anal.* **3**, 514–543; *p. 201.*

R. KULKARNI AND P. J. LAURENT [1991], "Q-splines", *Numer. Algorithms* **1**, 45–73; *p. 242.*

J. M. LANE AND R. F. RIESENFELD [1983], "A geometric proof for the variation diminishing property of B-spline approximation", *J. Approx. Theory* **37**, 1–4; *p. 139.*

* P. J. LAURENT [1972], *Approximation et Optimisation*, Hermann (Paris).

C. L. LAWSON [1977], "Software for C^1 surface interpolation", in *Mathematical Software III* (J. R. Rice, ed), Academic Press (New York), 161–194; *p. 311.*

* C. L. LAWSON AND R. J. HANSON [1974], *Solving Least Squares Problems*, Prentice-Hall (Englewood Cliffs NJ); *p. 222.*

E. T. Y. LEE [1989], "Choosing nodes in parametric curve interpolation", *Computer-Aided Design* **21**, 363–370; *p. 278.*

ANDERS LINNÉR [1996], "Unified representations of nonlinear splines", *J. Approx. Theory* **84(3)**, 315-0-350; *p. 280.*

T. LYCHE AND L. L. SCHUMAKER [1975], "Local spline approximation methods", *J. Approx. Theory* **15**, 294–325; *p. 155.*

T. LYCHE AND R. WINTHER [1979], "A stable recurrence relation for trigonometric B-splines", *J. Approx. Theory* **25**, 266–279; *p. 313.*

MICHAEL A. MALCOLM [1977], "On the computation of nonlinear spline functions", *SIAM J. Numer. Anal.* **14**, 254–282; *p. 280, 314.*

M. J. MARSDEN [1972], "On uniform spline approximation", *J. Approx. Theory* **6**, 249–253; *p. 147.*

M. J. MARSDEN [1974], "Quadratic spline interplation", *Bull. Amer. Math. Soc.* **80**, 903–906; *p. 64, 185.*

DAVID F. MCALLISTER AND JOHN A. ROULIER [1978], "Interpolation by convex quadratic splines", *Math. Comp.* **32(144)**, 1154–1162 ; *p. 276.*

E. MEHLUM [1964], "A curve-fitting method based on a variational criterion", *BIT* **4**, 213–223; *p. 280, 314.*

E. MEHLUM [1974], "Non-linear splines", in *Computer Aided Geometric Design* (R. E. Barnhill and R. F. Riesenfeld, eds), Academic Press (New York), 173–207; *p. 280, 314.*

CHARLES A. MICCHELLI, T. J. RIVLIN, AND S. WINOGRAD [1976], "The optimal recovery of smooth functions", *Numer. Math.* **26**, 191–200; *p. 193, 199.*

M. J. MUNTEANU AND L. L. SCHUMAKER [1973], "On a method of Carasso and Laurent for constructing interpolating splines", *Math. Comp.* **27(122)**, 317–325; *p. 81.*

G. M. NIELSON [1974], "Some piecewise polynomial alternatives to splines under tension", in *Computer Aided Geometric Design* (R. E. Barnhill and R. F. Riesenfeld, eds), Academic Press (New York), 209–235; *p. 276.*

* G. NÜRNBERGER [1989], *Approximation by Spline Functions*, Springer-Verlag (Berlin).

M. J. D. POWELL [1967], "On the maximum errors of polynomial approximations defined by interpolation and least squares criteria", *Computer J.* 9, 404–407; *p. 20.*

M. J. D. POWELL [1974], "Piecewise quadratic surface fitting for contour plotting", in *Software for Numerical Mathematics* (D. J. Evans, ed), Academic Press (London), 253–271; *p. 311.*

* P. M. PRENTER [1975], *Splines and Variational Methods*, Wiley (New York).

W. QUADE AND L. COLLATZ [1938], "Zur Interpolationstheorie der reellen periodischen Funktionen", *Sitzungsber. der Preuss. Akad. Wiss., Phys. Math.* **30**, 383–429; *p. 289.*

L. RAMSHAW [1989], "Blossoms are polar forms", *Comput. Aided Geom. Design* **6(4)**, 323–358; *p. 144.*

C. H. REINSCH [1967], "Smoothing by spline functions", *Numer. Math.* 10, 177–183; *p. 208, 214, 219.*

JOHN R. RICE [1967], "Characterization of Tchebycheff approximations by splines", *SIAM J. Numer. Anal.* 4, 557–565 ; *p. 313.*

J. R. RICE [1969], "On the degree of convergence of nonlinear spline approximation", in *Approximation with Special Emphasis on Spline Functions* (I. J. Schoenberg, ed), Academic Press (New York), 349–365; *p. 163.*

R. RIESENFELD [1973], "Applications of B-spline approximation to geometric problems of computer-aided design", dissertation, Syracuse Univ; *p. 141.*

* T. J. RIVLIN [1969], *An Introduction to the Approximation of Functions*, Blaisdell (Walton, Mass); *p. 20, 26, 27, 89, 132.*

* THEODORE J. RIVLIN [1974], *The Chebyshev Polynomials*, John Wiley and Sons (New York); *p. 14, 28.*

P. L. J. VAN ROOIJ AND F. SCHURER [1974], "A Bibliography on Spline Functions", in *Spline-Funktionen* (K. Böhmer, G. Meinardus, and W. Schempp, eds), Bibliographisches Institut (Mannheim), 315–415.

* A. SARD [1963], *Linear Approximation*, Math. Survey 9, AMS (Providence RI); *p. 313.*

* A. SARD AND S. WEINTRAUB [1971], *A Book of Splines*, Wiley (New York).

R. SCHABACK [1976], "Calculation of best approximations by rational splines", in *Approximation Theory, II* (G. G. Lorentz, C. K. Chui, and L. L. Schumaker, eds), Academic Press (New York), 533–539; *p. 313.*

K. SCHERER AND A. YU. SHADRIN [1999], "New upper bound for the B-spline basis condition number. II. A proof of de Boor's 2^k-conjecture", *J. Approx. Theory* **99(2)**, 217–229; *p. 132.*

I. J. SCHOENBERG [1946], "Contributions to the problem of approximation of equidistant data by analytic functions, Part A: On the problem of smoothing or graduation, a first class of analytic approximation formulas", *Quart. Appl. Math.* 4, 45–99, 112–141; *p. 54, 105, 288.*

I. J. SCHOENBERG [1964], "Spline functions and the problem of graduation", *Proc. Amer. Math. Soc.* **52**, 947–950; *p. 208.*

I. J. SCHOENBERG [1967], "On spline functions", in *Inequalities I* (O. Shisha, ed), Academic Press (New York), 255–291; *p. 139.*

* I. J. SCHOENBERG (ed.) [1969], *Approximation with Special Emphasis on Spline Functions*, Academic Press (New York).

* I. J. SCHOENBERG [1973], *Cardinal Spline Interpolation*, CBMS, SIAM (Philadelphia); *p. 65, 283, 285.*

I. J. SCHOENBERG AND A. WHITNEY [1953], "On Pólya frequency functions. III. The positivity of translation determinants with an application to the interpolation problem by spline curves", *Trans. Amer. Math. Soc.* 74, 246–259; *p. 172.*

* M. H. SCHULTZ [1973], *Spline Analysis*, Prentice–Hall (Englewood Cliffs, NJ).

L. L. SCHUMAKER [1968]$_1$, "Uniform approximation by Tchebycheffian spline functions, II. Free knots", *SIAM J. Numer. Anal.* **5**, 647–656; *p. 313.*

L. L. SCHUMAKER [1969]$_2$, "Uniform approximation by Tchebycheffian spline functions", *J. Math. Mech.* **18**, 369–377 ; *p. 313.*

L. L. SCHUMAKER [1976], "Fitting surfaces to scattered data", in *Approximation Theory, II* (G. G. Lorentz, C. K. Chui, and L. L. Schumaker, eds), Academic Press (New York), 203–268; *p. 311.*

* L. L. SCHUMAKER [1981], *Spline Functions: Basic Theory*, Wiley (New York).

D. G. SCHWEIKERT [1966], "An interpolation curve using a spline in tension", *J. Math. Phys.* **45**, 312–317; *p. 264.*

A. SHARMA AND A. MEIR [1966], "Degree of approximation of spline interpolation", *J. Math. Mech.* **15(5)**, 759–767; *p. 35.*

ROYCE W., JR. SOANES [1976], "VP-splines, an extension of twice differentiable interpolation", in *Proc. 1976 Army Numerical Anulysis and Computers Conf., ARO Report 76-3* (xxx, ed), U.S. Army Res. Office, P.O. Box 12211 (Research Triangle Park NC), 141–152; *p. 265.*

* H. SPÄTH [1974], *Spline Algorithms for Curves and Surfaces*, translated by W. D. Hoskins and H. W. Sager, Utilitas Math. (Winnipeg); *p. 264, 313.*

* H. SPÄTH [1990], *Eindimensionale Spline–Algorithmen*, Oldenbourg (München).

* H. SPÄTH [1991], *Zweidimensionale Spline–Interpolations–Algorithmen*, Oldenbourg (München).

* S. B. STECHKIN AND YU. SUBBOTIN [1976], *Splines in Numerical Mathematics*, Izd. Nauka (Moscow).

Y. N. SUBBOTIN [1967], "On piecewise-polynomial (spline) interpolation", *Mat. Zametki* **1**, 63–70; *p. 61, 64.*

B. K. SWARTZ AND R. VARGA [1972], "Error bounds for spline and *L*-spline interpolation", *J. Approx. Theory* **6**, 6–49; *p. 44.*

* CHARLES VAN LOAN [1992], *Computational Frameworks for the Fast Fourier Transform*, SIAM (Philadelphia); *p. 287.*

* R. S. VARGA [1971], *Functional Analysis and Approximation Theory in Numerical Analysis*, CBMS Vol. 3, SIAM (Philadelphia PA); *p. 314.*

E. T. WHITTAKER [1923], "On a new method of graduation", *Proc. Edinburgh Math. Soc.* **41**, 63–75; *p. 208.*

* J. H. WILKINSON AND C. REINSCH [1971], *Linear Algebra*, Springer Verlag (New York); *p. 180.*

K. A. WITTENBRINK [1973], "High order projection methods of moment- and collocation-type for nonlinear boundary value problems", *Computing* **11**, 255–274; *p. 244.*

Note: A reasonably complete (and searchable) list of publications concerning splines is available at `http://www.cs.wisc.edu/~deboor/bib`.

Index

Applied Mathematical Sciences

(continued from page ii)

60. *Ghil/Childress:* Topics in Geophysical Dynamics: Atmospheric Dynamics, Dynamo Theory and Climate Dynamics.
61. *Sattinger/Weaver:* Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics.
62. *LaSalle:* The Stability and Control of Discrete Processes.
63. *Grasman:* Asymptotic Methods of Relaxation Oscillations and Applications.
64. *Hsu:* Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems.
65. *Rand/Armbruster:* Perturbation Methods, Bifurcation Theory and Computer Algebra.
66. *Hlavácek/Haslinger/Necasl/Lovísek:* Solution of Variational Inequalities in Mechanics.
67. *Cercignani:* The Boltzmann Equation and Its Applications.
68. *Temam:* Infinite-Dimensional Dynamical Systems in Mechanics and Physics, 2nd ed.
69. *Golubitsky/Stewart/Schaeffer:* Singularities and Groups in Bifurcation Theory, Vol. II.
70. *Constantin/Foias/Nicolaenko/Temam:* Integral Manifolds and Inertial Manifolds for Dissipative Partial Differential Equations.
71. *Catlin:* Estimation, Control, and the Discrete Kalman Filter.
72. *Lochak/Meunier:* Multiphase Averaging for Classical Systems.
73. *Wiggins:* Global Bifurcations and Chaos.
74. *Mawhin/Willem:* Critical Point Theory and Hamiltonian Systems.
75. *Abraham/Marsden/Ratiu:* Manifolds, Tensor Analysis, and Applications, 2nd ed.
76. *Lagerstrom:* Matched Asymptotic Expansions: Ideas and Techniques.
77. *Aldous:* Probability Approximations via the Poisson Clumping Heuristic.
78. *Dacorogna:* Direct Methods in the Calculus of Variations.
79. *Hernández-Lerma:* Adaptive Markov Processes.
80. *Lawden:* Elliptic Functions and Applications.
81. *Bluman/Kumei:* Symmetries and Differential Equations.
82. *Kress:* Linear Integral Equations, 2nd ed.
83. *Bebernes/Eberly:* Mathematical Problems from Combustion Theory.
84. *Joseph:* Fluid Dynamics of Viscoelastic Fluids.
85. *Yang:* Wave Packets and Their Bifurcations in Geophysical Fluid Dynamics.
86. *Dendrinos/Sonis:* Chaos and Socio-Spatial Dynamics.
87. *Weder:* Spectral and Scattering Theory for Wave Propagation in Perturbed Stratified Media.
88. *Bogaevski/Povzner:* Algebraic Methods in Nonlinear Perturbation Theory.
89. *O'Malley:* Singular Perturbation Methods for Ordinary Differential Equations.
90. *Meyer/Hall:* Introduction to Hamiltonian Dynamical Systems and the N-body Problem.
91. *Straughan:* The Energy Method, Stability, and Nonlinear Convection.
92. *Naber:* The Geometry of Minkowski Spacetime.
93. *Colton/Kress:* Inverse Acoustic and Electromagnetic Scattering Theory, 2nd ed.
94. *Hoppensteadt:* Analysis and Simulation of Chaotic Systems, 2nd ed.
95. *Hackbusch:* Iterative Solution of Large Sparse Systems of Equations.
96. *Marchioro/Pulvirenti:* Mathematical Theory of Incompressible Nonviscous Fluids.
97. *Lasota/Mackey:* Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics, 2nd ed.
98. *de Boor/Höllig/Riemenschneider:* Box Splines.
99. *Hale/Lunel:* Introduction to Functional Differential Equations.
100. *Sirovich (ed):* Trends and Perspectives in Applied Mathematics.
101. *Nusse/Yorke:* Dynamics: Numerical Explorations, 2nd ed.
102. *Chossat/Iooss:* The Couette-Taylor Problem.
103. *Chorin:* Vorticity and Turbulence.
104. *Farkas:* Periodic Motions.
105. *Wiggins:* Normally Hyperbolic Invariant Manifolds in Dynamical Systems.
106. *Cercignani/Illner/Pulvirenti:* The Mathematical Theory of Dilute Gases.
107. *Antman:* Nonlinear Problems of Elasticity.
108. *Zeidler:* Applied Functional Analysis: Applications to Mathematical Physics.
109. *Zeidler:* Applied Functional Analysis: Main Principles and Their Applications.
110. *Diekmann/van Gils/Verduyn Lunel/Walther:* Delay Equations: Functional-, Complex-, and Nonlinear Analysis.
111. *Visintin:* Differential Models of Hysteresis.
112. *Kuznetsov:* Elements of Applied Bifurcation Theory, 2nd ed.
113. *Hislop/Sigal:* Introduction to Spectral Theory: With Applications to Schrödinger Operators.
114. *Kevorkian/Cole:* Multiple Scale and Singular Perturbation Methods.
115. *Taylor:* Partial Differential Equations I, Basic Theory.
116. *Taylor:* Partial Differential Equations II, Qualitative Studies of Linear Equations.

(continued on next page)

Applied Mathematical Sciences

(continued from previous page)

117. *Taylor:* Partial Differential Equations III, Nonlinear Equations.
118. *Godlewski/Raviart:* Numerical Approximation of Hyperbolic Systems of Conservation Laws.
119. *Wu:* Theory and Applications of Partial Functional Differential Equations.
120. *Kirsch:* An Introduction to the Mathematical Theory of Inverse Problems.
121. *Brokate/Sprekels:* Hysteresis and Phase Transitions.
122. *Gliklikh:* Global Analysis in Mathematical Physics: Geometric and Stochastic Methods.
123. *Le/Schmitt:* Global Bifurcation in Variational Inequalities: Applications to Obstacle and Unilateral Problems.
124. *Polak:* Optimization: Algorithms and Consistent Approximations.
125. *Arnold/Khesin:* Topological Methods in Hydrodynamics.
126. *Hoppensteadt/Izhikevich:* Weakly Connected Neural Networks.
127. *Isakov:* Inverse Problems for Partial Differential Equations.
128. *Li/Wiggins:* Invariant Manifolds and Fibrations for Perturbed Nonlinear Schrödinger Equations.
129. *Müller:* Analysis of Spherical Symmetries in Euclidean Spaces.
130. *Feintuch:* Robust Control Theory in Hilbert Space.
131. *Ericksen:* Introduction to the Thermodynamics of Solids, Revised ed.
132. *Ihlenburg:* Finite Element Analysis of Acoustic Scattering.
133. *Vorovich:* Nonlinear Theory of Shallow Shells.
134. *Vein/Dale:* Determinants and Their Applications in Mathematical Physics.
135. *Drew/Passman:* Theory of Multicomponent Fluids.
136. *Cioranescu/Saint Jean Paulin:* Homogenization of Reticulated Structures.
137. *Gurtin:* Configurational Forces as Basic Concepts of Continuum Physics.
138. *Haller:* Chaos Near Resonance.
139. *Sulem/Sulem:* The Nonlinear Schrödinger Equation: Self-Focusing and Wave Collapse.
140. *Cherkaev:* Variational Methods for Structural Optimization.
141. *Naber:* Topology, Geometry, and Gauge Fields: Interactions.
142. *Schmid/Henningson:* Stability and Transition in Shear Flows.
143. *Sell/You:* Dynamics of Evolutionary Equations.
144. *Nédélec:* Acoustic and Electromagnetic Equations: Integral Representations for Harmonic Problems.
145. *Newton:* The *N*-Vortex Problem: Analytical Techniques.
146. *Allaire*: Shape Optimization by the Homogenization Method.
147. *Aubert/Kornprobst:* Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations.