Gary D. Knott

# Interpolating Cubic Splines

Birkhäuser
Boston • Basel • Berlin

Gary D. Knott
Civilized Software, Inc.
12109 Heritage Park Circle
Silver Spring, MD 20906
U.S.A.
www.civilized.com

**Library of Congress Cataloging-in-Publication Data**
Knott, Gary D.
Interpolating cubic splines / Gary D. Knott.
p. cm. — (Progress in computer science and applied logic ; v. 18)
Includes bibliographical references and index.
ISBN 0-8176-4100-9 (alk. paper). — ISBN 3-7643-4100-9 (alk. paper)
1. Spline theory. 2. Interpolation. I. Title. II. Series.
QA224.K66 1999
511'.42—dc21

AMS Subject Classifications: 41A15, 65D05, 65D07, 65D10, 65D17

Printed on acid-free paper.
*Birkhäuser* ®

ISBN 0-8176-4100-9 SPIN 10692469
ISBN 3-7643-4100-9

Reformatted from author's files in LaTeX 2e by TeXniques, Inc., Cambridge, MA.
Printed and bound by Hamilton Printing Company, Rensselaer, NY.
Printed in the United States of America.

9 8 7 6 5 4 3 2 1

# Contents

# Preface

A *spline* is a thin flexible strip composed of a material such as bamboo or steel that can be bent to pass through or near given points in the plane, or in 3-space in a smooth manner. Mechanical engineers and drafting specialists find such (physical) splines useful in designing and in drawing plans for a wide variety of objects, such as for hulls of boats or for the bodies of automobiles where smooth curves need to be specified. These days, physical splines are largely replaced by computer software that can compute the desired curves (with appropriate encouragment). The same mathematical ideas used for computing "spline" curves can be extended to allow us to compute "spline" surfaces.

The application of these mathematical ideas is rather widespread. Spline functions are central to computer graphics disciplines. Spline curves and surfaces are used in computer graphics renderings for both real and imaginary objects. Computer-aided-design (CAD) systems depend on algorithms for computing spline functions, and splines are used in numerical analysis and statistics. Thus the construction of movies and computer games travels side-by-side with the art of automobile design, sail construction, and architecture; and statisticians and applied mathematicians use splines as everyday computational tools, often divorced from graphic images.

The functions that have been most frequently used for the mathematical incarnation of splines are the simple univariate or bivariate polynomials, well-known to students of mathematics. Cubic polynomials hold a special place among these functions for reasons elaborated in this text. A cubic spline function is made by "joining together" various univariate or bivariate cubic polynomials, i.e., by defining the cubic spline function of interest piece by piece (*piecewise*), where each piece is given by a particular cubic polynomial. This mathematical device of constructing a function piecewise is employed to overcome the obstreperous oscillatory behavior that single polynomials often exhibit. Splines based on higher-degree polynomials are infrequently found in practical application, and their study adds only limited theoretical content to the understanding of polynomial splines beyond that which is apparent by studying cubic splines. I therefore focus on cubic polynomials in this book.

Mathematically, the study of spline functions falls under the heading of *approximation theory* wherein we study functions that approximate other functions. In some cases these other functions may be completely known, but difficult to compute, so we seek a nearby function that is easier to compute. In other cases, our knowledge may be incomplete. For example, we may only know a set of points that lie in the graph of a function of interest, and sometimes these points may be inexactly known, so that we must deal with measurement error in the given points. It is these latter situations, where only some points are known, that generally interest us here.

The study of spline functions involves an array of basic mathematical concepts from calculus and analysis, as well as numerical analysis. Spline functions and their properties thus serve as an excellent application subject for solidifying and appreciating concepts introduced in the first two or three years of college mathematics. I have included a chapter covering the basics of the differential geometry of curves, which provides a foundation for subsequent topics and which may be of independent interest particularly to computer graphics programmers.

There is a vast literature on spline functions spread across the various disciplines mentioned above and elsewhere. Much work in the last twenty years, including the material covered in most common reference texts on splines, has focused on particular representations of splines, such as B-splines and, to a lesser degree, rational splines. Much of this work has considered splines that approximate given points rather than exactly interpolate them. In contrast, the focus in this book is the under-discussed area of spline interpolation, although approximation is also covered in a few especially important contexts. My intent is to provide a reasonably broad survey of explicit formulas and algorithms for interpolating spline functions and to cover some less-commonly treated topics such as optimal smoothing splines and cardinal splines, or untreated topics such as monotonic spline interpolation and physical splines which have heretofore been confined to (relatively inaccessible) technical reports and journals. One should keep in mind, however, that any expository mathematical text necessarily presents a distillation and synthesis of the work of many people done over many years; only a few of the fruitful or unfruitful digressions explored by these many people, each of which sheds light of its own, can be presented.

This text does not use the classic structure consisting of theorems and proofs in sequence, typographically set off and visually easy to identify. Rather a discursive style is employed; I encourage you to identify and label your own theorems as you read. I believe the dividends from this investment outweigh the benefits of using a more formal structuring. Nevertheless, the discussion is carefully organized and rigor is not completely ignored. Definitions of terms occur at or just after their first use, as indicated by the use of italic type.

There are many exercises scattered throughout the text, in most cases at the places where they seem most relevant to the story. These exercises should at least be read. Many exercises have solutions, often because these are repositories of useful information, and sometimes because the result is subsequently required in the text. In these cases, reading the exercise and giving it some cursory contemplation will, I hope, make the solution clear; the exercise–solution format is a device for concisely presenting the subject material. Also, when so inclined, it helps to seek a solution yourself. Even unfruitful contemplation (in moderation) can be enjoyable and valuable. The exercises differ greatly in their levels of difficulty; in a few cases, no solutions are known, so seeking a solution is a research project. Rather than indicate the level of difficulty, I have chosen to adopt Richard Bellman's position, as stated in the seminal book *The Art of Computer Programming: Volume 1 / Fundamental Algorithms* by Donald E. Knuth.

I hope that this book will be a useful reference for numerical analysts, engineers, computer graphics programmers, and CAD specialists, as well as other mathematical scientists and students of all types. Several C programs that exhibit a few interesting computational approaches are presented. Source files for these programs are available at the Birkhäuser web site whose URL is `WWW.BIRKHAUSER.COM`. The figures in this book were mostly constructed by using the MLAB mathematical and statistical modeling program which contains a sampling of the spline computation methods discussed in this book (see `WWW.CIVILIZED.COM`).

Programmers may find this book helpful as a source of formulas for methods beyond those encoded in the given C programs. Statisticians with an interest in kernel and other forms of density estimation and in smoothing and non-parametric regression may also find this book interesting, albeit statistically incomplete. In addition to serving as a reference source and as a resource for autodidacts, this book is suitable as a text for upper level undergraduate/graduate courses, either as the primary text for a special topics course within the disciplines of numerical analysis, computer graphics, or engineering, or as an adjunct text in courses in these areas.

I wish to thank Ann Kostant and Tom Grasso at Birkhäuser and Elizabeth Loew at TEXniques, for all their help in the preparation of this book; and I wish to dedicate this book to the teachers who have inspired me, especially Robert W. Floyd and Donald E. Knuth.

Gary D. Knott
Civilized Software Inc.

# 1
# Mathematical Preliminaries

This initial chapter contains a reminder of some basic vector algebra used in the book. The focus is on a geometric view of vector spaces, and on the inner product (also known as the dot product) and vector cross product operations. Although this episodic material is no substitute for previous exposure, it may be helpful to have some basic results presented here.

## 1.1 The Pythagorean Theorem

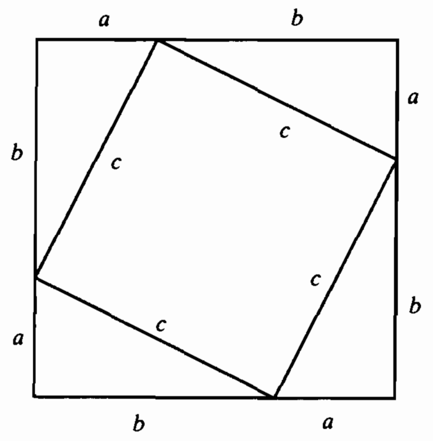
Two triangles $S$ and $T$ are *similar* if they have the same three angles at their vertices; that is, if the angles of $S$ are $\alpha$, $\beta$, and $\gamma$ in clockwise order, then the angles of $T$ are also $\alpha$, $\beta$, and $\gamma$ in clockwise order when $S$ and $T$ are similar. Similar triangles differ only in size, not shape. Note that if we did not require the angles of the triangles $S$ and $T$ to be the same in the same order (here arbitrarily chosen to be clockwise), the triangles $S$ and $T$ might be scaled reflections, i.e., scaled mirror-images of each other.

Recall that a *right* triangle is one that has a 90 degree angle at one of its vertices; the side opposite to that vertex is called the *hypotenuse* of the triangle.

Four copies of a right triangle $T$ in $\mathcal{R}^2$, which has a hypotenuse of length $c$ and whose two perpendicular sides are of length $a$ and $b$, can be placed on the four sides of a square $S_c$ of side length $c$, so as to form a larger square $S_{a+b}$ of side length $a + b$, which circumscribes the square $S_c$. Thus $area(S_{a+b}) = area(S_c) + 4 \cdot area(T)$, and so, equivalently, $(a + b)^2 = c^2 + 4(ab)/2$, which implies that $a^2 + b^2 = c^2$. This relationship among the lengths of the side of a right triangle is the *pythagorean theorem*, one of the most far-reaching theorems of mathematics, whose influence is seen from number theory to Fourier analysis. The use of a lower-case ‘p’ in the word “pythagorean” (taken from the discoverer’s name *Pythagorus*) is a traditional mark of special respect denoting ubiquity.

**Exercise 1.1:** Draw a picture of the square $S_c$ showing the four circumscribed right triangles.

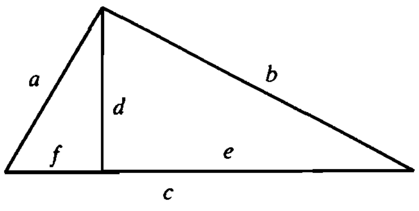
**Solution 1.1:** The requested picture is shown below.



Another demonstration of the pythagorean theorem for the right triangle $T$ is as follows. We drop a perpendicular line segment of length $d$ from the right angle vertex onto the hypotenuse of length $c$ to form two right triangles $T_1$ and $T_2$, each similar to $T$. The sides of $T_1$ are of lengths $b$, $d$, and $e$, where $b$ is the length of the hypotenuse in $T_1$. The sides of $T_2$ are of lengths $a$, $d$, and $f$, where $a$ is the length of the hypotenuse in $T_2$. These lengths satisfy $e + f = c$. Then, since these triangles are similar, we have $b/c = e/b$ and $a/c = f/a$ and so, $b^2 = ce$ and $a^2 = cf$. Thus, $a^2 + b^2 = c(f + e) = c^2$.

**Exercise 1.2:** Use a library to find another proof of the pythagorean theorem.

**Exercise 1.3:** The pythagorean theorem says that the squares con-



structed on the three sides of a right triangle are such that the area of the largest square equals the sum of the areas of the other two squares. Show that if we construct half circles on the three sides of a right triangle, then the area of the largest half circle equals the sum of the areas of the other two half circles. Can you generalize to other shapes?

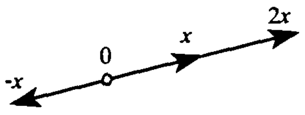
**Exercise 1.4:** Show that a triangle $T$ with sides of lengths $a$, $b$, $c$ in increasing order is a right triangle if $a^2 + b^2 = c^2$.

## 1.2 Vectors

Euclid defines a *point* as "that which has no part." Following Descartes, a numerical definition of a point is used in modern mathematics. A *point* in 2-space is an ordered pair of real numbers, and each distinct ordered pair of real numbers is a distinct point in 2-space. Similarly, a point in 3-space is an ordered triple of real numbers, and each distinct triple of real numbers is a distinct point in 3-space; in general, a point in $n$-space is an $n$-tuple of real numbers. We denote the set of real numbers by $\mathcal{R}$, and the set of ordered $n$-tuples of real numbers by $\mathcal{R}^n$, where $n$ is a positive integer. A point in $\mathcal{R}^n$ is written explicitly as $(x_1, x_2, \ldots, x_n)$, where each *component*, $x_i$, is a real number. $\mathcal{R}^n$ is called ***n*-dimensional euclidean space**, or just ***n*-space** for short. Note that, except for a subtle quibble, $\mathcal{R} = \mathcal{R}^1$. The word "scalar" is a synonym for "real number."

We may imagine mutually perpendicular $x$ and $y$ axes in 2-space and thus represent a point $p$ in $\mathcal{R}^2$ pictorially as a dot, positioned according to the first and second numerical components $p_1$ and $p_2$ of $p$, used respectively as $x$ and $y$ coordinates. A similar graphical interpretation applies in 3-space. We shall almost always interpret the components of a point as coordinate values, and then we should say we have a point in coordinatized $n$-space; but we will usually not bother to be so precise.

Points in $\mathcal{R}^n$ are also called *vectors*, and traditionally a vector $p$ in $\mathcal{R}^n$ corresponds to the directed line segment from the origin, $0 = (0, 0, \ldots, 0)$,



in coordinatized $n$-space to the point $p$ in coordinatized $n$-space. This line segment is denoted by $segment[0, p]$, and in general, we denote the line segment from a point $a$ to a point $b$ by $segment[a, b]$. Similarly, we denote the directed line, which is directed so that it passes first through the point $a$ and then through the point $b$ by $line[a, b]$. For us, vectors always start at the origin! A directed line segment starting at an arbitrary point is parallel to a particular vector starting at the origin, but the directed line segment itself is not a vector unless it starts at the origin.

Points and vectors are in one-to-one correspondence, and the two terms are used interchangeably even though they have slightly differing pictorial interpretations. The origin-based directed line segment model for vectors has the great advantage that the product of a vector and a scalar which results in another vector, and the sum of two vectors, which likewise is a vector, can be geometrically interpreted in this model.

**Exercise 1.5:** Look up the words *geometry* and *trigonometry* in a dictionary.

We shall define scalar vector products and vector sums for vectors in $\mathcal{R}^n$, but you should keep $\mathcal{R}^2$ and $\mathcal{R}^3$ uppermost in mind.

The scalar vector product of a real scalar value $\alpha$ and a vector $x \in \mathcal{R}^n$ is written $\alpha x$ (or even $x\alpha$) and is itself a vector in $\mathcal{R}^n$. When $x$ is given in coordinatized form as $(x_1, x_2, \ldots, x_n)$, $\alpha x$ is the vector $(\alpha x_1, \alpha x_2, \ldots, \alpha x_n)$, which lies in $\mathcal{R}^n$. A non-zero vector $x$ is said to specify or to have a *direction*, namely the direction from the origin 0 to $x$; and any positive multiple of $x$ has the same direction. If you stand at 0, looking at the point $x$, then you are also looking at $\alpha x$ for all $\alpha > 0$. That is, $\alpha x$ lies on the same line which connects 0 and $x$ in $\mathcal{R}^n$. If $\alpha < 0$, $\alpha x$ also lies on the line defined by the two points, 0 and $x$, but in this case $\alpha x$ is said to have the direction opposite to the direction of $x$. We may write $x/\alpha$ as short for $x(1/\alpha)$, and we write $-x$ as short for $(-1)x$. If $x, y \in \mathcal{R}^n$ and $x$ is parallel to $y$, then $x$ is a multiple of $y$, and conversely.

Note that we casually write 0 to denote the coordinate vector $(0, 0, \ldots, 0)$, and the dimension of the space holding this vector must be deduced from the context.

The sum of two $n$-space vectors $p$ and $q$ is defined to be another $n$-space vector. When $p$ and $q$ are given in coordinatized form, $p+q$ is the $n$-space vector whose components are the sums of the corresponding components of $p$ and $q$. Thus $p+q = (p_1+q_1, p_2+q_2, \ldots, p_n+q_n)$. For $n = 3$, the three points 0, $p$, and $q$ determine a plane in $\mathcal{R}^3$ except in degenerate cases, and the line segments, $segment[0, p]$ and $segment[0, q]$, may be taken as two adjacent sides of a parallelogram lying in this plane. The vector $p+q$ is the vertex of this parallelogram occurring diagonally across from 0. The vertices of this parallelogram in order are thus 0, $p$, $p+q$, $q$, and we shall denote this four-sided polygon by the notation $polygon[0, p, p+q, q]$.

**Exercise 1.6:** When do the points 0, $p$, and $q$ fail to determine a plane in $\mathcal{R}^3$ and what is the line segment, $segment[0, p+q]$, in this case?

**Exercise 1.7:** Show by a diagram that $segment[0, p+q]$ is the diagonal of the parallelogram $polygon[0, p, p+q, q]$.

**Exercise 1.8:** Show that the usual rules of real arithmetic imply that $\alpha(p+q) = \alpha p + \alpha q$, where $\alpha \in \mathcal{R}$ and $p$ and $q$ are vectors in $n$-space.

**Exercise 1.9:** Show by a diagram that $p+q = q+p$.

**Exercise 1.10:** In $\mathcal{R}^2$, how can you tell which of $polygon[0, p, p+q, q]$ and $polygon[0, q, p+q, p]$ is clockwise, i.e., has its vertices listed in clockwise order?

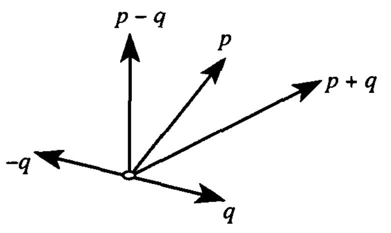
The general graphical interpretation for vector addition is as follows. To find the point $p+q$, take a line segment, $segment[p, z]$, of the same length as $q$, with its starting point at $p$, and positioned parallel to and in the same direction as $q$. Then the ending point $z$ coincides with $p+q$.

The graphical interpretation of vector subtraction is as follows. To find the point $p-q$, take the line segment, $segment[p, z]$, of the same length as $q$, with its starting point at $p$, and positioned parallel to and in the same direction as $-q$. Then the ending point $z$ coincides with $p-q$.

**Exercise 1.11:** Show that the notation $-b$ is consistent by showing that $a + (-1)b = a - b$, where $a$ and $b$ are vectors in $\mathcal{R}^n$.

**Exercise 1.12:** Show that $a - b = (a_1 - b_1, \ldots, a_n - b_n)$.

**Exercise 1.13:** Show that the midpoint of $segment[x, y]$ is the point $(x+y)/2$.



## 1.3 Subspaces and Linear Independence

The set of vectors $\mathcal{R}^n$ has the closure property that if $x, y \in \mathcal{R}^n$, then $\alpha x + \beta y \in \mathcal{R}^n$ for all scalars $\alpha$ and $\beta \in \mathcal{R}$. Thus $\mathcal{R}^n$ is closed with respect to scalar vector multiplication and vector addition. Such a closed non-empty set of vectors is called a *vector space* with respect to the field of scalars $\mathcal{R}$. There are subsets of $\mathcal{R}^n$ which are vector spaces in their own right. For example, for $n \geq 2$, the set $\{x \mid x \in \mathcal{R}^n \text{ and } x_1 = x_2\}$ is closed with respect to scalar vector multiplication and vector addition. Such non-empty subsets of $\mathcal{R}^n$ are called *subspaces* of $\mathcal{R}^n$.

The smallest subspace of $\mathcal{R}^n$ is $\{(0, 0, \dots, 0)\}$. The largest is $\mathcal{R}^n$ itself. Subspaces of $\mathcal{R}^n$ of the form $\{x \mid x = \alpha y \text{ for } \alpha \in \mathcal{R} \text{ and } y \neq 0 \text{ fixed in } \mathcal{R}^n\}$ are *lines*, and for $n \geq 2$, subspaces of $\mathcal{R}^n$ of the form $\{x \mid x = \alpha y + \beta z \text{ for } \alpha, \beta \in \mathcal{R} \text{ and } y \neq 0 \text{ and } z \neq 0 \text{ fixed in } \mathcal{R}^n \text{ with } y \neq \delta z \text{ for any } \delta \in \mathcal{R}\}$ are subspaces are *planes*. Not every line or plane is a subspace however; the point 0 must be in a line or plane which is a subspace.

**Exercise 1.14:** A *linear combination* of vectors $v_1, \dots, v_k$ is a vector $x$ which is a sum of scalar multiples of the vectors $v_1, \dots, v_k$, so that $x = \alpha_1 v_1 + \cdots + \alpha_k v_k$ where $\alpha_1, \dots, \alpha_k \in \mathcal{R}$. Show that all linear combinations of any finite set of vectors of $\mathcal{R}^n$ are members of $\mathcal{R}^n$, given that this is true for all sets of two vectors.

**Exercise 1.15:** Let $S$ be a subspace of $\mathcal{R}^n$. Show that $0 \in S$.

**Exercise 1.16:** Let $S$ be a subspace of $\mathcal{R}^n$ containing two distinct vectors. Show that $S$ contains an infinite number of distinct vectors.

**Exercise 1.17:** Let $n > 1$. Show that $\mathcal{R}^n$ has an infinite number of subspaces. What about $\mathcal{R}^1$?

There are numerous varieties of vector spaces, each with special definitions of scalar vector multiplication and vector addition, such as the set of all polynomials of degree $n$ or less, or the set of all square integrable periodic functions of period $p$, or the fundamental cycles in an abstract network (i.e., graph), in addition to the vector spaces $\mathcal{R}^1, \mathcal{R}^2, \ldots$ . The field of scalars can be chosen to be a class of values other than the reals, e.g., the complex numbers $C$ could be used; but most of the main concepts are clearly evident within the $\mathcal{R}^n$ spaces.

We shall be peripherally concerned below with the *infinite-dimensional* vector space consisting of all the real-valued functions defined on a given interval. Such functions form a vector space, since if $f$ and $g$ are such functions, then $\alpha f + \beta g$ is also such a function. When we restrict ourselves to admit only those functions that are suitably "nice," such as those functions whose squares can be integrated over the given interval, then it can be shown that *Cauchy* sequences of such functions converge to a function of this same type, and the corresponding vector space of functions is then called a *Hilbert space*. Since we generally avoid any concessions to mathematical completeness that would cause a digression from the central topics at hand, and since we shall have no need to use any of the properties of Hilbert spaces in any depth, the interesting story about Hilbert space bases and inner products, etc., will be left for independent investigation.

A given set of vectors $A \subseteq \mathcal{R}^n$ can be enlarged to obtain the smallest subspace of $\mathcal{R}^n$ containing $A$. (A set $Q$ is a *smallest* set satisfying a condition $C$ if no proper subset of $Q$ satisfies $C$.) This subspace is called the subspace *spanned* by $A$, and is denoted by $subspace(A)$. The subspace spanned by $A$ is just the set of all linear combinations of all the vectors in every finite subset of $A$. Thus $subspace(A) = \{\alpha_1 a_1 + \cdots + \alpha_k a_k \mid \alpha_1, \ldots, \alpha_k \in \mathcal{R}, a_1, \ldots, a_k \in A, 0 < k < \infty\} \cup \{(0, \ldots, 0)\}$.

**Exercise 1.18:** Show that for a given non-empty set $A \subseteq \mathcal{R}^n$, $subspace(A)$ is uniquely determined.

The intersection $P \cap Q$ of two subspaces $P$ and $Q$ of $\mathcal{R}^n$ is itself a subspace of $\mathcal{R}^n$. The union of the two subspaces is not necessarily a subspace, however the subspace spanned by $P \cup Q$ can always be formed. This subspace is denoted by $subspace(P \cup Q)$.

**Exercise 1.19:** Show that $subspace(P \cap Q) \subseteq subspace(P) \cap subspace(Q)$, and $subspace(P) \cup subspace(Q) \subseteq subspace(P \cup Q)$ where $P$ and $Q$ are arbitrary subsets of $\mathcal{R}^n$. Explain why 0 is explicitly placed in $subspace(A)$ by definition. Hint: consider the empty set as a subset of $\mathcal{R}^n$.

Every subspace $P \subseteq \mathcal{R}^n$ is spanned by some finite subset of $n$ or fewer vectors. (You can imagine that the subspace $\{0\} \subset \mathcal{R}^n$ is spanned by the $n$-vector 0). We may seek the smallest set of *non-zero* vectors which span a given subspace, $P$. This set will be called a *basis set* of the subspace $P$. All basis sets for $P$ have the same size. This number is called the *dimension* of $P$. The subspace $\{0\} \subset \mathcal{R}^n$ has dimension 0, and the empty set $\emptyset$ will be assigned to be its basis set. (You may wish to contemplate whether this assignment requires consideration of a theory of types of distinct empty sets). We start the discussion of dimension with the definition of linear independence.

The vectors $a_1, a_2, \ldots, a_k$ are called *linearly-independent* if and only if, for all scalars, $\alpha_1, \ldots, \alpha_k \in \mathcal{R}, \alpha_1 a_1 + \cdots + \alpha_k a_k = 0$ only when $\alpha_1 = \cdots = \alpha_k = 0$.

A sequence of vectors $a_1, \ldots, a_k$ are *linearly-dependent* if they are not linearly-independent, which occurs if and only if one or more of the vectors $a_j$ is a linear combination of the earlier vectors $a_1, \ldots, a_{j-1}$. Thus a set of vectors $a_1, \ldots, a_k$, which spans a subspace, can be reduced to a linearly-independent set by discarding zero or more of the vectors.

**Exercise 1.20:** Show that if the vectors $a_1, a_2, \ldots, a_k$ are linearly-independent, then one or more of these vectors $a_j$ can be written as a linear combination of the remaining vectors.

If a vector space has a finite spanning set, it is called a *finite-dimensional* vector space. If $n$ vectors span a vector space which contains $r$-linearly-independent vectors, then $n \geq r$.

## 1.4 Vector Space Bases

A *basis* of a vector space $V$ is a sequence of linearly-independent vectors which span $V$. If $B$ is a basis for $V$ consisting of $n$ vectors, then there are $n! - 1$ other bases for $V$ which have the same basis set as $B$.

If a vector space $V$ is finite-dimensional, then every basis set for $V$ is of the same size. A contradiction would arise if this were not the case. This common size is called the *dimension* of $V$ and denoted by $\dim(V)$. Note $\mathcal{R}^n$ is a finite-dimensional vector space of dimension $n$, since the sequence of vectors $\langle e_1, \ldots, e_n \rangle$, where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is a vector whose $i$-th component is 1 and whose other components are all 0, is a basis for $\mathcal{R}^n$. The sequence of vectors $\langle e_1, \ldots, e_n \rangle$ is a particularly important basis called the ***natural basis*** for $\mathcal{R}^n$.

> **Exercise 1.21:** Show that any set of $m$ vectors of an $n$-dimensional vector space which has a basis set of size $n$ with $m > n$ is linearly-dependent.

If the $k$ vectors $a_1, \ldots, a_k$ form a basis for a vector space $V$, then every vector $x$ in $V$ is uniquely expressible as a linear combination of $a_1, \ldots, a_k$. If $x = \alpha_1 a_1 + \cdots + \alpha_k a_k$, the scalar coefficients $\alpha_1, \ldots, \alpha_k$ are called the *coordinates* of $x$ with respect to the basis $\langle a_1, \ldots, a_k \rangle$. Thus $x$ corresponds to a coordinate vector $(\alpha_1, \ldots, \alpha_k)$ with respect to the basis $\langle a_1, \ldots, a_k \rangle$ in $\mathcal{R}^k$. The basis vectors themselves correspond to the coordinate vectors $e_1, \ldots, e_k$ since $a_i = 0a_1 + \cdots + 1a_i + \cdots + 0a_k$. A basis for $V$ can thus be called a *coordinate system* for $V$. Every vector in $V$ is represented by a coordinate vector $n$-tuple with respect to the chosen coordinate system basis.

When we use explicit $n$-tuple coordinate vectors, we are implicitly specifying vectors in terms of some fixed coordinate system basis. Thus, we can treat $\mathcal{R}^n$ as being coordinatized by some fixed unspecified basis of $n$ linearly-independent abstract vectors whose coordinate vectors are $e_1, \ldots, e_n$. It is usually convenient to postulate that a basis of mutually perpendicular vectors exists and to choose the coordinate system basis to be such a sequence of mutually perpendicular vectors; this basis then becomes the natural basis for $\mathcal{R}^n$. This is what we do when we draw coordinate axes for constructing graphs of objects in 2-space or 3-space. Each axis line corresponds to a coordinate system basis vector which is, in turn, a point on its axis line.

Let $basis(V)$ denote an arbitrary basis of the vector space $V$. If $S$ and $T$ are subspaces of $\mathcal{R}^n$, then $subspace(S \cup T) = subspace(basis(S) \cup basis(T))$. If $S \cap T = \{0\}$, then $subspace(S \cup T)$ is called the *direct sum* of $S$ and $T$, which is written $S \oplus T$. Moreover, when $S \cap T = \{0\}$, $basis(S) \cup basis(T) = basis(subspace(S \cup T))$ and $dim(S \oplus T) = dim(S) + dim(T)$. When $S \cap T = \{0\}$, we then call $S$ and $T$ *complementary subspaces* in $S \oplus T$. Note that use of the notation $S \oplus T$ implies that $S \cap T = \{0\}$, otherwise its use is not kosher.

**Exercise 1.22:** Let $S$ and $T$ be subspaces of $\mathcal{R}^n$. Show that if $S \cap T = \{0\}$, then $S \oplus T = subspace(S \cup T) = \{x + y \mid x \in S, y \in T\}$. Also show that $S \cap T = \{0\}$ if and only if every vector $x \in subspace(S \cup T)$ can be written $x = a + b$, where $a \in S$ and $b \in T$, in only one way; i.e., the vectors $a$ and $b$ are uniquely determined by $x$.

**Exercise 1.23:** Let $S$, $A$, and $B$ be subspaces of $\mathcal{R}^n$ with $A \cap B = \{0\}$. Show that $S \cap (A \oplus B) = (S \cap A) \oplus (S \cap B)$.

**Exercise 1.24:** Let $S$ and $T$ be arbitrary subspaces of $\mathcal{R}^n$. Show that $dim(S) + dim(T) = dim(S \cap T) + dim(subspace(S \cup T))$. This result states, for example, that the sum of the dimensions of two planes, which intersect in a line in 3-space which passes through the origin, is the dimension of the intersection line plus the dimension of 3-space; that is, $2 + 2 = 1 + 3$.

**Exercise 1.25:** Show that the set of all univariate polynomial functions of degree $n$ or less with real coefficients is a vector space of dimension $n$; do this by specifying the vector addition and scalar vector multiplication operations, and by presenting an explicit basis for this finite-dimensional vector space of functions.

We mentioned above that not every line or plane is a subspace, however, they are almost subspaces. It is useful to consider such sets of vectors; they are called *affine subspaces* or *linear manifolds* or *flats*. (The word "affine" has the same root as the word "affinity"; this root is the basis of the Latin word for "related"). In particular, if $S$ is a dimension $k$ subspace of $\mathcal{R}^n$, and $t$ is a vector in $\mathcal{R}^n$, then $\{x + t \mid x \in S\}$ is an affine subspace or flat of $\mathcal{R}^n$. The flat $\{x + t \mid x \in S\}$ is also called a *translate* of $S$ and is written $t + S$. Note that an affine subspace is not a subspace unless it contains the vector 0. The flat $A = \{x + t \mid x \in S\}$ is assigned the same dimension as the subspace $S$, so $dim(A) = k$. Lines in $\mathcal{R}^n$ are just 1-dimensional flats, and for $n \geq 2$, planes in $\mathcal{R}^n$ are 2-dimensional flats. In addition, $(n - 1)$-dimensional flats of $\mathcal{R}^n$ are called *hyperplanes*. Dimension 0 flats are points.

Let $A$ be a $k$-dimensional flat, let $t \in A$, and let $S = \{v - t \mid v \in A\}$. $S$ is the $k$-dimensional subspace corresponding to the $k$-dimensional flat $A$. Let $b_1, \ldots, b_k$ be a basis for $S$. Then every point $x$ in $A$ can be written $x = t + \beta_1 b_1 + \cdots + \beta_k b_k$ with a unique sequence of coefficients $\beta_1, \ldots, \beta_k$. But then $x = (1 - \beta_1 - \cdots - \beta_k)t + \beta_1(b_1 + t) + \cdots + \beta_k(b_1 + t)$.

Let $a_0 = t$, and $a_i = b_i + t$ for $1 \le i \le k$, and let $\beta_0 = 1 - \beta_1 - \cdots - \beta_k$. Then $x = \beta_0 a_0 + \cdots + \beta_k a_k$, where $a_0, a_1, \ldots, a_k \in A$ and $\beta_0 + \cdots + \beta_k = 1$. Thus we see that every point $x \in A$ can be written as a linear combination of $k+1$ points in $A$ with coefficients that sum to 1. The vectors $a_0, \ldots, a_k$ are called a *barycentric basis* for the flat $A$, and $(\beta_0, \ldots, \beta_k)$ are called the *barycentric coordinates* of $x$ with respect to the barycentric basis $a_0, \ldots, a_k$.

**Exercise 1.26:** Show that if $a_0, \ldots, a_k$ is a barycentric basis for the flat $A$, then the vectors $a_0, \ldots, a_k$ are linearly-dependent, but a subset of $k$ of these vectors are linearly-independent.

**Exercise 1.27:** Show that the barycentric coordinates of a point $x \in A$ with respect to the barycentric basis $a_0, \ldots, a_k$ are unique.

**Exercise 1.28:** Show that, if $a_0, \ldots, a_k$ form a barycentric basis for the flat $A$, then $(a_0 + \cdots + a_k)/(k+1)$ belongs to $A$.

It is sometimes useful to extend euclidean $n$-space by adding "ideal" points, defined by fiat as the points where parallel lines intersect, so that now parallel lines always intersect. In particular, given any family of parallel lines, we may imagine that these lines "curve back" onto themselves to all join at a single point which is not found in $\mathcal{R}^n$. Then we say that two distinct lines intersect at one point, even when they are parallel. The ideal point which represents the common intersection point of a family of parallel lines with a given direction specified by that particular line of the family is called the *point at infinity* for this direction. Note that a family of parallel lines forms a 1-dimensional flat. When we add the points at infinity to $\mathcal{R}^n$, we get a mathematical structure called *projective* $n$-space, denoted by $\mathcal{P}^n$. The points at infinity in $\mathcal{P}^n$ form *lines at infinity*, and in general, $k$-dimensional flats at infinity arise for $0 \le k < n$. Projective $n$-space is *not* a vector space, but it contains euclidean $n$-space as a subset. Euclidean geometry metamorphizes into *projective geometry* when we move from $\mathcal{R}^n$ to $\mathcal{P}^n$.

## 1.5 Euclidean Length

The *euclidean length* $|x|$ of a vector $x \in \mathcal{R}^n$ coordinatized with respect to a fixed basis of mutually perpendicular unit length vectors (which we postulate exists) is $[x_1^2 + x_2^2 + \cdots + x_n^2]^{1/2}$, and the *euclidean distance* between two vectors $x$ and $y$ is the length $|x - y|$. Note $|x| > 0$ unless $x = 0$, and for $x \neq 0$, $x/|x|$ is a vector of length 1 in the same direction as $x$. A vector of length 1 is called a *unit vector.*

**Exercise 1.29:** Show that the length of a vector $x$ in $\mathcal{R}^3$ is the same as the distance from the origin to the point $x$, obtained by applying the pythagorean theorem at most twice.

**Exercise 1.30:** Show that for $x \in \mathcal{R}^n$, $|x| = |-x|$.

**Exercise 1.31:** Show that the length of the vector $\alpha x$ is the same as $|\alpha|$ times the length of $x$, where $\alpha \in \mathcal{R}$.

**Exercise 1.32:** What is the length of the line segment $segment[a, b]$ in $\mathcal{R}^n$?

## 1.6 The Euclidean Inner Product

Given two vectors $x$ and $y$, coordinatized with respect to a fixed basis of mutually perpendicular unit vectors, the *euclidean inner product* of $x$ and $y$ is a scalar denoted by $(x, y)$. By definition $(x, y) = x_1 y_1 + \cdots + x_n y_n$. The notation $x \cdot y$ is also used to denote the inner product, which is also called the *dot product* because of this alternate notation. We may write $x \cdot y$ when $(x, y)$ might be confused with a vector.

The following facts are easily verified.

$$\begin{aligned}
(x, y) &= (y, x), \\
(\alpha x, y) &= \alpha(x, y), \\
(x + y, z) &= (x, z) + (y, z), \\
(x, x) &= |x|^2, \text{ and} \\
|x| &= 0 \text{ if and only if } x = 0.
\end{aligned}$$

Note that to compute $(x, y)$, we must express the vectors $x$ and $y$ with respect to a common basis of mutually perpendicular unit vectors. It will be seen later that $(x, y)$ is the same value, no matter which such basis is chosen.

If $(x_1, x_2)$ is a unit vector in $\mathcal{R}^2$ which forms an angle of $\alpha$ radians with the $x$-axis vector $(1, 0)$, then $x_1 = \cos(\alpha)$ and $x_2 = \sin(\alpha)$. Together with an understanding of the angular direction specified by an arbitrary real number, this constitutes a geometric definition of the functions $\cos(\alpha)$ and $\sin(\alpha)$.

> **Exercise 1.33:** Prove that the sum of the angles at the three vertices of a triangle is 180°.

Consider the triangle in $\mathcal{R}^2$ with vertices at the points $(0,0)$, $(a,0)$, and $(b\cos(\alpha), \text{b}\sin(\alpha))$, where $a > 0$, $b > 0$, and $\alpha$ is the angle at the vertex $(0,0)$. The sides of this triangle are of lengths $a$, $b$, and $[a^2 + b^2 - 2ab\cos(\alpha)]^{1/2}$. Note that $0 \le \alpha \le \pi$. This follows since the length of the side opposite the angle $\alpha$ is same as the length of the vector $(a,0) - (b\cos(\alpha), \text{b}\sin(\alpha))$. This length is $|(a - b\cos(\alpha), -\text{b}\sin(\alpha))| = [a^2 + b^2 - 2ab\cos(\alpha)]^{1/2}$. This is the "law of cosines," which generalizes the pythagorean theorem to yield the length of the third side of any triangle, given the lengths of the other two sides and the angle between them.

Now consider the triangle whose vertices are the points $0, x$, and $y$. The length of the side between $x$ and $y$ is $|x - y|$, which, by the law of cosines, must equal $[|x|^2 + |y|^2 - 2|x| \cdot |y|\cos(\alpha)]^{1/2}$, where $\alpha$ is the angle between the two vectors $x$ and $y$. But $|x - y|^2 = (x - y, x - y) = (x,x) - 2(x,y) + (y,y) = |x|^2 + |y|^2 - 2(x,y)$, and so $|x|^2 + |y|^2 - 2|x| \cdot |y|\cos(\alpha) = |\text{x}|^2 + |\text{y}|^2 - 2(\text{x}, \text{y})$. Therefore $(x,y) = |x| \cdot |y|\cos(\alpha)$. Thus the euclidean inner product determines the cosine of the smallest angle between two vectors in 2-space.

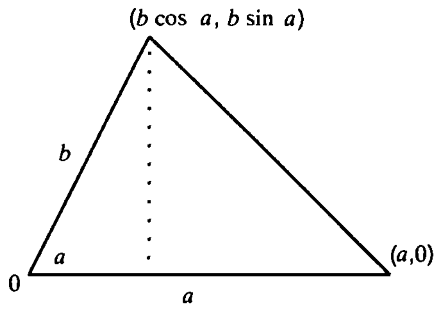
> **Exercise 1.34:** Prove that $(x,y) = |x| \cdot |y|\cos(\alpha)$ is true in 3-space.

> **Exercise 1.35:** Prove the Schwarz inequality: $|(x,y)| \le |x| \cdot |y|$.

> **Solution 1.35:** Suppose $x \ne 0$ and $y \ne 0$, and let $c > 0$ be such that $c|y| = |x|$. Then $0 \le (cy \pm x, cy \pm x) = (c|y|)^2 + |x|^2 \pm 2c(x,y)$, so $\mp 2c(x,y) \le c|y|(c|y|) + (|x|)|x| = c|y| \cdot |x| + c|y| \cdot |x| = 2c|x| \cdot |y|$, and hence $\mp(x,y) \le |x| \cdot |y|$. This proof relies only on the facts: $(x,y) = (y,x)$, $(\alpha x, y) = \alpha(x,y)$, $(x + y, z) = (x,z) + (y,z)$, $|x|^2 = (x,x)$, and $|x| = 0$ if and only if $x = 0$.

In general, for non-zero vectors $x$ and $y$ in $\mathcal{R}^n$, we define $angle(x,y)$ to be the smallest angle $\alpha$ of the two angles between $x$ and $y$ such that $0 \le \alpha \le \pi$ and $\cos(\alpha) = (\text{x}, \text{y})/(|\text{x}| \cdot |\text{y}|)$. The Schwarz inequality guarantees that $-1 \le (x,y)/(|x| \cdot |y|) \le 1$.

> **Exercise 1.36:** Prove the triangle inequality: $|x + y| \le |x| + |y|$.



Why is this called the triangle inequality? Use this to establish that $|x_1 - x_m| \leq |x_1 - x_2| + |x_2 - x_3| + \cdots + |x_{m-1} - x_m|$ for any vectors $x_1, \ldots, x_m$.

**Exercise 1.37:** Show that if $|x + y| = |x| + |y|$, then $x$ and $y$ are linearly-dependent.

**Exercise 1.38:** Show that if $u$ is a unit vector, then $u_i = \cos(\alpha_i)$, where $\alpha_i$ is the angle between $u$ and $e_i$, the $i$-th natural basis vector, and then show that $(\cos(\alpha_1))^2 + \cdots + (\cos(\alpha_n))^2 = 1$, and that the angles $\alpha_1, \ldots, \alpha_n$ satisfy $n(\arccos(1/n^{1/2})) \leq |\alpha_1| + \cdots + |\alpha_n| \leq n(\pi - \arccos(1/n^{1/2}))$. Also show that if $u_i \geq 0$ for $1 \leq i \leq n$, then $|\alpha_1| + \cdots + |\alpha_n| \leq (n-1)\pi/2$.

**Solution 1.38:** The value $|\alpha_1| + \cdots + |\alpha_n|$ is minimal when $u_1 = \cdots = u_n = 1/n^{1/2}$ and maximal when $u_1 = \cdots = u_n = -1/n^{1/2}$.

**Exercise 1.39:** Prove that if $|a| = |b|$, then $(a + b, a - b) = 0$.

**Solution 1.39:** This states that the diagonals of a rhombus with four equal sides are perpendicular.

**Exercise 1.40:** Find the unit vector which lies on the bisector of the angle between the two vectors, $a$ and $b$.

**Solution 1.40:** The bisector vector $v$ is $(a/|a| + b/|b|)/2$.

**Exercise 1.41:** Let $u \neq 0$ and $v$ be vectors in $\mathcal{R}_n$ such that $\alpha u = v$. Solve for the scalar $\alpha$ in terms of the vectors $u$ and $v$.

**Solution 1.41:** $\alpha = (v, u)/(u, u)$.

**Exercise 1.42:** Given a rectangle, $A$, whose diagonal is of length $\alpha$, and a rectangle, $B$, whose diagonal is of length $\beta$, is it true or false that $\beta > \alpha$ implies $area(B) > area(A)$?

**Solution 1.42:** It is false. But if we translate $A$ and $B$ so that the vector $a$ is the diagonal of $A$ and the vector $b$ is the diagonal of $B$, with all the coordinates of both $a$ and $b$ non-negative, then $area(B) > area(A)$ whenever $\beta > \alpha(\cos(s)\sin(s)/(\cos(t)\sin(t)))^{1/2}$, where $s = angle(a, (1,0))$ and $t = angle(b, (1,0))$.

**Exercise 1.43:** Suppose that the four points $a$, $b$, $c$, and $d$ are distinct. Show that the quadrilateral $polygon[(a+b)/2, (b+c)/2, (c+d)/2, (d+a)/2]$ is a parallelogram. What happens as $a$ approaches $b$? Show that any non-degenerate triangle circumscribes three distinct parallelograms.

A vector function $x$ of a real variable $t$ is a function that maps from $\mathcal{R}$ into $\mathcal{R}^n$, so that $x(t) \in \mathcal{R}^n$. Defining such a vector function $x$ is equivalent to specifying $n$ ordinary real-valued functions $x_1, x_2, \ldots, x_n$ such that $x_i(t) = (x(t))_i$. The derivative of a vector function $x(t) = (x_1(t), \ldots, x_n(t))$ is the vector function $x'(t)$ whose components are the derivatives $x_1'(t), \ldots, x_n'(t)$.

**Exercise 1.44:** Note that, for vector functions $u$ and $v$ of a real variable $t$, the inner product $u \cdot v$ is a scalar function of $t$. Show that the derivative $(u \cdot v)' = u' \cdot v + u \cdot v'$.

**Exercise 1.45:** Show that if $a$ and $b$ are unit vectors, then $|a|^2|b|^2 - (a,b)^2 = 0$ if and only if $a = b$ or $a = -b$.

A vector $x$ is determined by its length and its direction, and its direction is determined by the angles between the vector $x$ and each member of a set of basis vectors. Let $a_1, a_2, \ldots, a_n$ be $n$ linearly-independent vectors in $\mathcal{R}^n$ and let $\beta_1, \beta_2, \ldots, \beta_n \in \mathcal{R}$. Then the $n$ simultaneous linear equations $(a_1, x) = \beta_1, \ldots, (a_n, x) = \beta_n$ define $x$ to be an $n$-vector which forms the specific angles $\alpha_1, \alpha_2, \ldots, \alpha_n$ with the vectors $a_1, a_2, \ldots, a_n$, where $\beta_i/(|a_i| \cdot |x|) = \cos(\alpha_i)$. When $n-1$ of these angles are determined, the remaining angle is then fixed. The last equation is thus usable to determine $|x|$.

**Exercise 1.46:** Show that the triangle *triangle*$[(a+b)/2, (b+c)/2, (c+a)/2]$ is similar to *triangle*$[a, b, c]$.

In the definitions of the euclidean length of a vector and the euclidean inner product of two vectors given above, we have postulated the existence of and assumed that the underlying coordinate system basis for $\mathcal{R}^n$ is a sequence of $n$ mutually perpendicular unit vectors. This is implied by the fact that the inner product is *defined* so that $(e_i, e_j) = \delta_{ij}$, where $\delta_{ij} :=$ if $i = j$ then 1 else 0, and $e_1, \ldots, e_n$ are the coordinate vectors of the underlying basis vectors.

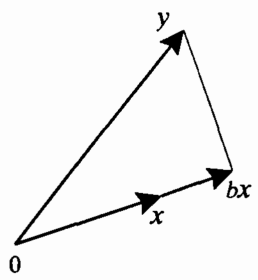
Rather than introduce the euclidean inner product $(x, y)$ and the euclidean length of a vector $|x|$ by generalizing from our notions of angle and length in $\mathcal{R}^2$ and $\mathcal{R}^3$ with a postulated coordinate basis of mutually perpendicular unit vectors, we could proceed more abstractly to show that any possible inner product operation that satisfies the properties: (1): $(x, y) = (y, x)$, (2): $(\alpha x, y) = \alpha(x, y)$, (3): $(x, y + z) = (x, y) + (x, z)$ is obtained by choosing a basis $B = \langle b_1, \ldots, b_n \rangle$ for $\mathcal{R}^n$ and specifying the inner product values $(b_i, b_j)$ for $1 \le i \le j \le n$. The length of a vector $x$ can then be defined in terms of the inner product as $(x, x)^{1/2}$ whenever $(x, x) \ge 0$, which can be shown to always be the case for certain choices of the inner product values $(b_i, b_j)$. For such choices, it turns out that there is always another basis $\langle e_1, \ldots, e_n \rangle$ for which $(e_i, e_j) = \delta_{ij}$.

**Exercise 1.47:** Show that if a length function $|x|$ is defined on $\mathcal{R}^n$ such that $|\alpha x| = |\alpha||x|$ and $|x| \ge 0$, then an associated inner product function can be defined by $(x, y) = (|x + y|^2 - |x|^2 - |y|^2)/2$.

## 1.7 Projection onto a Line

Two vectors $x$ and $y$ in $\mathcal{R}^n$, coordinatized with respect to a basis of mutually perpendicular unit vectors, are said to be *perpendicular* or *normal* or *orthogonal* to each other if and only if $(x, y) = 0$. Note this definition implies that the zero vector is normal to every vector.

Whether or not $(x, y) = 0$, the vector $y$ can always be expressed as $y = \beta x + z$ for some scalar $\beta$ and some vector $z$, where $(x, z) = 0$. This can be shown by construction: define $\beta = (x, y)/|x|^2$ and $z = y - \beta x$; then $y = \beta x + z$ and $(x, z) = 0$. The vector $\beta x$ is the component of $y$ parallel to the vector $x$, and the vector $z = y - \beta x = y - (x/|x|, y)(x/|x|)$ is the component of $y$ normal to $x$, and $(x, y - (x/|x|, y)(x/|x|)) = 0$. This



operation of decomposing a vector $y$ into a sum of mutually perpendicular component vectors is one of most fundamental and important operations in all of mathematics.

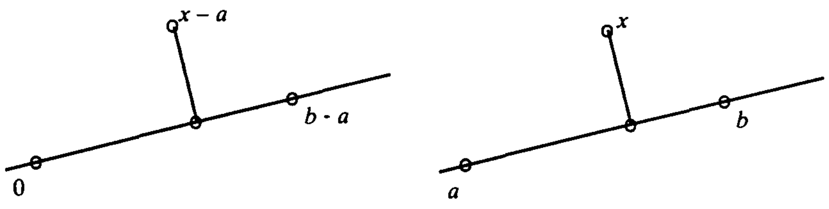
When $\beta > 0$, $\beta x$ is in the same direction as $x$, and when $\beta < 0$, $\beta x$ is in the opposite direction to $x$. The length $|\beta x|$ is $|y| \cdot |\cos(\alpha)|$ and $|z| = |y|\sin(\alpha)$ where $0 \leq \alpha \leq \pi$ is the angle between $x$ and $y$. Thus $|\beta x| = |(x, y)/|x||$. The angle $\alpha$ between $x$ and $y$ satisfies $\cos(\alpha) \geq 0$ when $\beta \geq 0$, and $\cos(\alpha) < 0$ when $\beta < 0$, and conversely. Therefore $\beta x = (|y|\cos(\alpha))(x/|x|) = (x/|x|, y)(x/|x|)$, which is the component of $y$ in the direction of $x$. The vector $\beta x = (x/|x|, y)(x/|x|)$ is called the *projection* of $y$ on $x$. Note if $x$ is a unit vector, the projection of $y$ on $x$, $\beta x$, is just $(x, y)x$.

**Exercise 1.48:** Show that $\beta = (x, y)/(x, x)$.

**Exercise 1.49:** Show that, if $x$, $y$, and $z$ are given vectors in $n$-space with $y \neq 0$ and $\alpha \in \mathcal{R}$, then $x = \alpha y + z$ implies $\alpha = ((x, y) - (z, y))/(y, y)$, but not conversely. When does a solution $\alpha$ exist such that $x = \alpha y + z$?

**Exercise 1.50:** Use trigonometry to show that in 3-space, $y$ is perpendicular to $x$ if and only if $(y, x) = 0$.

Now consider a one-dimensional subspace, $L = \{\alpha u \mid \alpha \in \mathcal{R}\}$ with $u \in \mathcal{R}^n$ such that $|u| = 1$. The subspace $L$ is a line through the origin, the projection of a point $x$ on $L$ is $(u, x)u$. The notion of projection can be obviously generalized to define the projection of a point $x$ onto an arbitrary line, $line[a, b]$. In order to compute the projection of the point $x$ onto the arbitrary line, $line[a, b]$, we may translate $n$-space by adding $-a$ to every point, compute the projection, $(x - a, (b - a)/|b - a|)(b - a)/|b - a|$, of $x - a$ on $line[0, b - a]$, and then translate back to obtain the answer



$(x-a, (b-a)/|b-a|)(b-a)/|b-a|+a$. This idea of translation, effected by adding a fixed vector to every vector of a flat in order to convert the flat into a subspace, is used frequently.

Given two line segments in 2-space, $segment[a, b]$ and $segment[c, d]$, we may wish to decide if they are parallel. This can be done by choosing a new coordinate system for 2-space by translating 2-space so that one of the segments, say $segment[a, b]$, starts at the new origin. In this new coordinate system, the question becomes: are $segment[0, b-a]$ and $segment[c-a, d-a]$ parallel? But this question can be answered by computing the projection $p$ of $c-a$ on $b-a$, the projection $q$ of $d-a$ on $b-a$, and the projection $r$ of $(c-a+d-a)/2$ on $b-a$, and checking whether $|c-a-p| = |d-a-q| = |(c-a+d-a)/2-r|$; if so, $segment[a, b]$ and $segment[c, d]$ are parallel.

**Exercise 1.51:** What are $|c-a-p|$, $|d-a-q|$, and $|(c-a+d-a)/2-r|$ in terms of $a$, $b$, $c$, and $d$?

**Exercise 1.52:** Show that $segment[a, b]$ and $segment[c, d]$ in 2-space are parallel exactly when $((b-a)/|b-a|, (d-c)/|d-c|) = \pm 1$.

**Exercise 1.53:** Show that $line[a, b] = \{ v \mid v = \alpha(b-a)+a,\ \alpha \in \mathcal{R} \} = \{ v \mid v = (1-\alpha)a + \alpha b,\ \alpha \in \mathcal{R} \}$.

**Exercise 1.54:** What is a necessary and sufficient condition for the two lines, $line[a, b]$ and $line[c, d]$ in 2-space, to be mutually perpendicular?

**Exercise 1.55:** What is the shortest vector $c$, from 0 to the line $H = \{ \alpha u + w \mid \alpha \in \mathcal{R} \}$ where $u$ and $w$ are vectors with $|u| = 1$? *Hint*: show that the vector $u$ is parallel to the line $H$.

**Solution 1.55:** $c = w - (w, u)u$. This can be shown by computing

$c = \alpha u + w$ with $\alpha$ equal to the minimizer of $|\alpha u + w|$, or alternatively, by finding $\alpha$ so that $(\alpha u + w, u) = 0$.

**Exercise 1.56:** Let $p$ be the projection of $x$ on $y$, and let $q$ be the projection of $y$ on $x$. Show that $|y| \cdot |p| = |x| \cdot |q|$. Also show that $angle(x, y) = angle(p, q)$.

**Exercise 1.57:** Let $x, y \in \mathcal{R}^n$ and define $a_0 = x$, $a_1 = y$, and $a_i$ to be the projection of $a_{i-1}$ on $a_{i-2}$ for $i > 1$. Show that $\lim_{i\to\infty} a_i = 0$ if and only if $angle(x, y) > 0$.

**Exercise 1.58:** Let $x, y \in \mathcal{R}^n$, and define the *maximal shared component* of $x$ and $y$ to be the vector $a \in \mathcal{R}^n$ such that $|a|$ is maximal subject to $x = a + b$, $y = a + c$, $angle(a, b) \leq \pi/2$ and $angle(a, c) \leq \pi/2$ with $b, c \in \mathcal{R}^n$. Find a formula for the maximal shared component of $x$ and $y$. *Hint*: consider the projection of 0 on $line[x, y]$.

**Exercise 1.59:** Suppose we specify 3 vertices $a$, $b$, and $c$ of a not-necessarily rectilinearly-oriented rectangle in the $xy$-plane. What is the length and width of this rectangle, and what is the fourth vertex in terms of $a$, $b$, and $c$?

**Exercise 1.60:** Compute the projection $h$ of a point $u$ on the line through the two points $p$ and $q$, which is $\{x \mid x = p + t(p-q), t \in \mathcal{R}\}$.

**Solution 1.60:** $h = p + (u - p, r)r$, where $r = (q - p)/|q - p|$. The distance from $u$ to the line is $|u - h|$.

**Exercise 1.61:** Given $a = (a_1, a_2) \in \mathcal{R}^2$, with $a \neq 0$, present a point $b$ such that $b \neq \alpha a$ for any real value $\alpha$.

**Solution 1.61:** $b = (-a_1 - a_2, a_1 + a_2)$.

**Exercise 1.62:** Give a computationally-effective formula for the distance from a point $x$ to a line segment $segment[a, b]$, defined by $\min_{y \in segment[a,b]} |x - y|$.

**Exercise 1.63:** Let $p$ be a point inside an equilateral triangle, $triangle$ $[a, b, c]$. Let $r$ be the projection of $p$ on $segment[a, b]$, let $s$ be the projection of $p$ on $segment[b, c]$, and let $t$ be the projection of $p$ on segment $[c, a]$. Let $d$ be the projection of $a$ on $segment[b, c]$; thus $|a - d|$

is the length of the altitude of the triangle. Show that $|r - p| + |s - p| + |t - p| = |a - d|$.

A set of linearly-independent unit vectors with the inner product of each pair of distinct vectors equal to 0 is called an *orthonormal* set of vectors. The vectors of an orthonormal set are mutually perpendicular linearly-independent unit vectors.

If $v_1, v_2, \dots, v_m$ is an orthonormal sequence of unit vectors, then $g = (v_1, y)v_1 + (v_2, y)v_2 + \cdots + (v_m, y)v_m$ is the projection of the vector $y$ on the subspace $S$, spanned by $v_1, \dots, v_m$. The vector $y - g$ is the component of $y$ normal to $S$. Various vectors may have component vectors normal to $S$ in differing directions, but if $S$ is an $n - 1$ dimensional subspace of a space of dimension $n$ (i.e., a hyperplane), the direction of a vector which is normal to $S$ is unique to within a multiple of $+1$ or $-1$.

**Exercise 1.64:** Show that, for the vectors $v_1, v_2, \dots, v_m$, $y$ and $g$ specified above, $(y - g, v_i) = 0$ for $1 \le i \le m$.

## 1.8 Planes in 3-Space

A plane $Q$ in $\mathcal{R}^3$ which passes through the origin can be defined by a non-zero normal vector $n$ as $Q = \{x \mid x \cdot n = 0\}$, and thus the translate plane $P$ denoted as $t + Q$, which is a translate of $Q$, is represented by $P = \{t + x \mid x \cdot n = 0\} = \{x \mid (x - t) \cdot n = 0\}$, where we have translated the subspace $Q$ by $t$ to obtain the flat $P$. Note $t \in P$. The plane $Q$, which contains 0, is a two-dimensional subspace of $\mathcal{R}^3$.

**Exercise 1.65:** Suppose $Q$ is a two-dimensional subspace, and suppose $t \in \mathcal{R}^3$ is non-zero. When is $t + Q$ a subspace?

Now $P = \{x \mid x \cdot n = t \cdot n\} = \{x \mid x \cdot n = k\}$ where the scalar $k = t \cdot n$. The vector $t$ is not unique; there are many translation vectors which produce $P$ from $Q$. The equation $x \cdot n = k$ is a *plane equation* for $P$. We may normalize it to obtain a plane equation for $P$ wherein $|n| = 1$ and $k \le 0$. This form is called a standardized plane equation for $P$. Note that, if $x \cdot n = k$ is any plane equation for a plane $P$, then the line $\{\beta n \mid \beta \in \mathcal{R}\}$ is normal to $P$.

**Exercise 1.66:** Explain precisely how the plane equation $x \cdot n = k$ can be normalized.

**Exercise 1.67:** Show that if $x \cdot n = k$ is any plane equation for a plane $P$, then $(k/|n|^2)n \in P$.

**Exercise 1.68:** Show that in the standardized plane equation $n \cdot x = k$ with $|n| = 1$ and $k \leq 0$ which defines a particular plane, $P = Q + t$, with $0 \in Q$, $k$ is independent of any translation vector $t$; $k$ depends only on $P$.

**Solution 1.68:** Any choice of $t$ in $P$ determines a translation which carries $Q$ into $P$, and $|t \cdot n|$ with $|n| = 1$ is the length of the projection of such a point $t$ in $P$ onto the vector $n$. This projection $(t \cdot n)n$, is the unique shortest translation vector that carries $Q$ into $P$, and it is independent of $t$. The unique translation of $Q$ along the direction normal to $Q$ which produces $P$ is determined by the vector of the form $\alpha n$ where $\alpha = k = t \cdot n$ for any vector $t$ in $P$.

Note the vector $r - s$ is parallel to $P$ for all $r$ and $s$ in $P$.

Note the $x$-axis intercept in $P$ is $k/n_1$, the $y$-axis intercept is $k/n_2$, and the $z$-axis intercept is $k/n_3$. If any of the values $n_1$, $n_2$, or $n_3$ are 0, then $P$ is parallel to the $x$-axis, $y$-axis, or $z$-axis, respectively. A plane $P$ with a standardized plane equation $n \cdot x = k$ where $|n| = 1$ and $k \leq 0$, for which $k/n_1 \neq 0, k/n_2 \neq 0$, and $k/n_3 \neq 0$ has an associated specific *cardinal* octant, which is that octant of $\mathcal{R}^3$ in which the point $kn$ lies; this is the octant $\{(x, y, z) \mid sign(x) = sign(k/n_1), sign(y) = sign(k/n_2), sign(z) = sign(k/n_3)\}$.

**Exercise 1.69:** There are 18 degenerate cases where a plane fails to have a cardinal octant. List them.

Let $n$ be a unit normal vector of the 2-dimensional subspace $Q$. Choose $\alpha$ so that the distance of the origin from the plane $P = Q + t$ is the length of the vector $\alpha n$ which is normal to $Q$ and lies in $P$. The vector $\alpha n$ is the projection of 0 onto $P$. Since $\alpha n$ lies in $P$, it satisfies $n \cdot \alpha n = k$, where $n \cdot x = k$ with $|n| = 1$ is a plane equation for $P$. Thus $\alpha = k/|n|^2 = k$. Thus $t = kn$ is the shortest translation vector which satisfies $Q + t = P$. Note that when $k < 0$, $P$ and $n$ lie on opposite sides of $Q$.

Given three non-colinear points $a$, $b$, and $c$, which determine a plane $P$, a normal vector $n$ for $P$ is obtained as $n = (a - c) \times (b - c)$ and $n \cdot x = c \cdot n$ is a plane equation for $P$, which is unique when we normalize $n$ to be a unit vector such that $c \cdot n \leq 0$. The vector $(c, n)n$ is the vector from the origin to $P$ of minimum length. Conversely, given the plane equation $n \cdot x = k$ with $|n| = 1$, we can generate a point in $P$ from an arbitrary point $y$ by projection as $y - (y, n)n + kn$.

**Exercise 1.70:** Show that if the points $a$ and $b$ lie in a plane $P$, then $\{a + \alpha(b - a) \mid \alpha \in \mathcal{R}\} \subset P$.

**Exercise 1.71:** How many numbers determine a plane?

**Solution 1.71:** Four numbers $n_1$, $n_2$, $n_3$, and $k$ determine a plane, with the dependency $n_1^2 + n_2^2 + n_3^2 = 1$. This is equivalent to three numbers and one bit. Nine numbers are needed, however, to directly specify three points. (Can you specify three points in $\mathcal{R}^3$ with fewer than nine numbers?)

**Exercise 1.72:** Find the plane $H$ which contains the line, $line[a, b]$, and which is parallel to the line, $line[p, q]$, where $line[a, b]$ and $line[p, q]$ are not themselves parallel.

**Solution 1.72:** $H$ is the plane containing the point $r$ and normal to the unique line segment, $segment[r, s]$, which is the shortest segment between $line[a, b]$ and $line[p, q]$, with $s \in line[p, q]$ and $r \in line[a, b]$. Then $H = \{x \mid (x - r, s - r) = 0\}$.

If $f : \mathcal{R}^3 \to \mathcal{R}^1$ is a continuous function such that $S = \{x \mid f(x) = 0\}$ is a surface which divides space into two disconnected parts, then $\{x \mid f(x) < 0\}$ and $\{x \mid f(x) > 0\}$ are the two parts. Thus $\{x \mid x \cdot n < k\}$ lies to one side of the plane $P = \{x \mid x \cdot n = k\}$ and $\{x \mid x \cdot n > k\}$ is on the other side.

**Exercise 1.73:** Given a standardized plane equation $n \cdot x = k$ for a plane $P$ which does not contain 0, what is the direction of $n$ with respect to $P$?

**Solution 1.73:** The cardinal octant of $P$ is the octant which contains $kn$. We know $kn \in P$ and $k < 0$, so $n$ is directed away from $P$, lying in the same side of $P$ as 0, since $n \cdot n > k$ and $n \cdot 0 > k$. The directed line segment, $segment[kn, 0]$, is in the same direction as $n$ and points from the plane $P$ towards 0.

Given the plane $Q$ holding the points 0, $u$, and $v$, and given two points, $p$ and $q$, not in $Q$, then $p$ and $q$ lie on the same side of $Q$ if and only if either they both lie on the same side as the normal vector $u \times v$, or they both lie on the opposite side from $u \times v$. Thus $p$ and $q$ lie on the same side of $Q$ if and only if $sign(p \cdot (u \times v)) = sign(q \cdot (u \times v))$.

Given two planes, $plane(0, u, v)$ and $plane(0, a, b)$, the vector $x = (u \times v) \times (a \times b)$ coincides with the line of intersection of the two planes. Let $\alpha$ be the angle between the two vectors $(a \times b)$ and $(u \times v)$. The angle between the two planes is $min(\alpha, \pi - \alpha)$.

The equation of the line of intersection $L = \{x \mid x = \alpha b + c, \alpha \in \mathcal{R}\}$ of the two planes $\{x \mid n \cdot x = \varepsilon\}$ and $\{x \mid m \cdot x = \delta\}$ given by standardized plane equations satisfies $b = n \times m$, and the vector $c$ can be chosen to satisfy $n \cdot c = \varepsilon, m \cdot c = \delta$ and $c = \beta n + \gamma m$. The reason that $c = \beta n + \gamma m$ is that $plane[0, n, m]$ intersects both the planes containing $L$, and hence intersects $L$. In fact, $L$ is normal to $plane[0, n, m]$. Thus $\beta + \gamma n \cdot m = \varepsilon$ and $\gamma + \beta n \cdot m = \delta$, and hence $\beta = (\varepsilon - \delta n \cdot m)/(1 - (n \cdot m)^2)$ and $\gamma = (\delta - \varepsilon n \cdot m)/(1 - (n \cdot m)^2)$.

**Exercise 1.74:** Given the line $L = \{x \mid x = \alpha b + c, \alpha \in \mathcal{R}\}$ and the plane $P = \{x \mid n \cdot x = k\}$, give an explicit construction for the set $L \cap P$.

**Solution 1.74:** If $(n, b) = 0$, then $L \cap P = L$. Otherwise, $L \cap P = \{[(k - (n, c))/(n, b)]b + c\}$.

**Exercise 1.75:** Given three planes $\{x \mid x \cdot a = \alpha\}$, $\{x \mid x \cdot b = \beta\}$, and $\{x \mid x \cdot c = \gamma\}$, find the single point of intersection, and indicate when it exists.

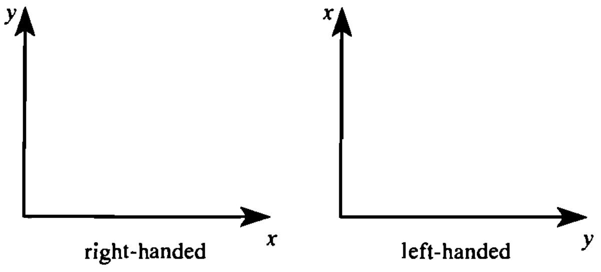
**Exercise 1.76:** Given the vertices $a_1, \dots, a_n$ and $b_1, \dots, b_m$ of two planar polygons $A$ and $B$ in 3-space, devise an algorithm which either finds a plane, $P$, that contains the origin and separates $A$ and $B$, so that all the vertices of $A$ lie on one side of $P$ and all the vertices of $B$ lie on the other side of $P$, or reports that no such plane exists.

Let $Q$ be a plane in 3-space which is a subspace and let the vector $n \neq 0$ be normal to $Q$. The 1-dimensional subspace $\{x \mid x = \alpha n$ with $\alpha \in \mathcal{R}\}$ and $Q$ are complementary subspaces, i.e., their intersection is $\{0\}$. Two planes $A$ and $B$ which are both subspaces in $\mathcal{R}^3$ and whose normal vectors are orthogonal are not complementary subspaces, however, since they share an entire line of intersection in common.

In general, in $\mathcal{R}^n$ with $n \geq 2$, a plane is a two-dimensional flat, defined by three points. An $(n-1)$-dimensional flat in $\mathcal{R}^n$ is called a *hyperplane*. A hyperplane is defined by $n$ points, or equivalently, by a translation vector and a normal vector. When $n = 3$, hyperplanes are planes.

## 1.9 Coordinate System Orientation

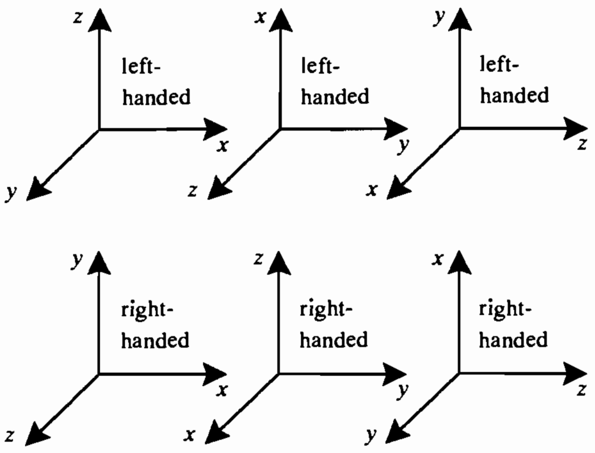
The *orientation* of a coordinate system is a property of the coordinate system indicating which of two classes (*left-handed* or *right-handed*) the coordinate system belongs to. Given an orthonormal sequence of $n$ vectors, $B = \langle b_1, \ldots, b_n \rangle$, we may coordinatize $n$-space with respect to these unit vectors listed in the arbitrarily given fixed order. The coordinate vectors of these $n$ basis vectors will be denoted by $e_1, \ldots, e_n$, where the coordinate vector $e_i$ is $(0, \ldots, 0, 1, 0, \ldots, 0)$ so that $(e_i)_j = \delta_{ij}$ with respect to the given orthonormal basis $B$. We may arbitrarily assign this basis either the left-handed or right-handed orientation. A basis, by itself, has no absolute orientation; all we can tell is if two basis sequences have the same or opposite orientation. The descriptive correspondence such as the right-hand rule by which we draw pictures representing a left or right-handed coordinate system is merely a graphical aid; it would be consistent if we were to switch these graphical description rules.



There are two possible orthonormal coordinate systems for 2-space, namely $\langle(1,0),(0,1)\rangle$ and $\langle(0,1),(1,0)\rangle$. One is right-handed and the other is left-handed.

There are six orthonormal coordinate systems for 3-space: three are left-handed and the other three are right-handed.

Each of the right-handed orthonormal coordinate systems for 3-space can be converted into one of the others by a suitable rotation operation,



and each of the left-handed orthonormal coordinate systems for 3-space are similarly interconvertable by means of suitable rotations; but there is no physically realizable rigid transformation of 3-space that can convert a right-handed coordinate system into a left-handed one, or vice-versa. In general, two orthonormal bases for $\mathcal{R}^n$ will have the same orientation if one can be transformed into the other by a series of rotations. We shall say more about this in the section on rotation transformations.

A rotation transformation of 3-space can be described as a righthand screw rotation or a lefthand screw rotation. A righthand screw rotation of a point $x$ about the non-zero vector $v$ by $\theta$ degrees results in the point $x'$, where $x'$ is found by curling the fingers of your right hand about the vector $v$ with your thumb pointing in the direction of $v$ and rotating $x$ $\theta$ degrees in the direction that your right hand fingers are curling to reach the point $x'$. This rotation of $x$ takes place in the plane $P$ containing $x$ which is perpendicular to the line, $line[0, v]$; it is observed to be a counterclockwise rotation in this plane about the point in $P$ where $line[0, v]$ intersects $P$ when $P$ is viewed with the vector $v$ directed at the observer.

A lefthand screw rotation is based on the same "rule of thumb," but using your left hand. In the case of a lefthand screw rotation of $x$, it can be seen to be a clockwise rotation of $x$ in a plane viewed with the axes of rotation vector $v$ directed toward the observer.

Just as there is no way to absolutely determine whether a coordinate system is left-handed or right-handed, there is no absolute way to distinguish between a lefthand and righthand screw rotation.

The fundamental issue has been illustrated by Richard Feynman [RFS63] as follows. Imagine that you are speaking via radio to a person in another culture who has never heard your language. You begin by teaching numeral names which you communicate by making $k$ equally spaced sounds followed by the name for $k$. Proceeding in this manner, you can (perhaps) define major parts of English. But can you define left and right? The idea that they are opposite directions or sides with respect to a center is communicable, but your communication partner may model "left" as his or her right and no discrepancy will arise which allows this to be detected.

Having called a basis left-handed, however, a righthand screw rotation of $\alpha$ radians about a vector $u$, expressed in that coordinate system, is distinguishable from the lefthand screw rotation of $\alpha$ radians about $u$. Moreover, orientation-preserving transformations which map the natural basis to a basis with the same orientation can be distinguished from an orientation-reversing transformation.

## 1.10 The Cross Product

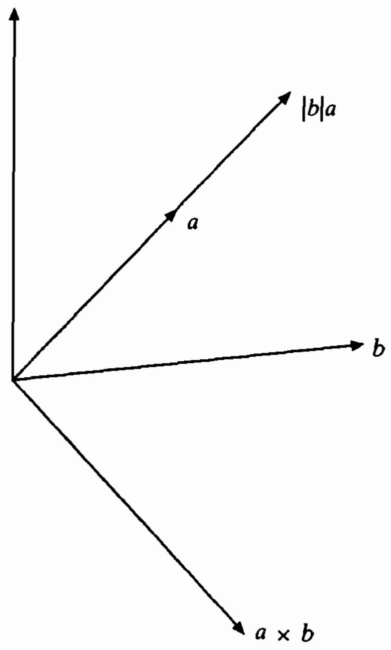
The *cross product* of two vectors $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$ in 3-space coordinatized with an orthonormal basis is $a \times b := (a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1)$. The cross product $a \times b$ is 0 if and only if $a = 0$ or $b = 0$ or $a$ is parallel to $b$, i.e., $a$ is a multiple of $b$. Also $|a \times b| = |a||b|\sin(\alpha)$ where $0 \le \alpha \le \pi$ is the angle between $a$ and $b$.

Note $a \times b = -(b \times a)$ and $(a \times b, a) = (a \times b, b) = 0$.

Suppose $a$ and $b$ are linearly-independent 3-space vectors. Then in a left-handed coordinate system for $\mathcal{R}^3$, the vectors $a$, $b$, and $a \times b$ in this order form another left-handed coordinate system for $\mathcal{R}^3$, where $b$ rotates into $a$ by means of a right-handed screw rotation of $angle(a, b)$ radians about $a \times b$. Similarly $a$, $b$, and $a \times b$ in this order form a right-handed coordinate system in a right-handed coordinate system with $a$ rotated into $b$ by a right-hand screw rotation of $angle(a, b)$ radians about $a \times b$.

**Exercise 1.77:** Derive the cross product formula from first principles by computing the vectors which are normal to both $a$ and $b$, and then choosing the nicest of these normals to be the vector $a \times b$.

**Solution 1.77:** Consider the vector, $n$, where $(n, a) = (n, b) = 0$. These two equations yield $n_1 = ((b_3a_2 - b_2a_3)/(b_2a_1 - a_2b_1))n_3$, and $n_2 = ((a_3b_1 - b_3a_1)/(b_2a_1 - a_2b_1))n_3$, and choosing $n_3 = b_2a_1 - a_2b_1$ produces the simplest result.



In a left-handed coordinate system, the cross product vector $a \times b$ can be characterized as the 90 degree righthand screw rotation of the component of $|b|a$ normal to $b$ about the vector $b$. In a right-handed coordinate system, $a \times b$ is the 90 degree righthand screw rotation of the component of $|a|b$ normal to $a$ about the vector $a$.

**Exercise 1.78:** Show that $(a + b) \times c = a \times c + b \times c$.

**Exercise 1.79:** Show that $(\alpha a) \times b = a \times (\alpha b) = \alpha(a \times b)$.

**Exercise 1.80:** Show that $(a \times b, c \times d) = (a, c)(b, d) - (a, d)(b, c)$.

**Exercise 1.81:** Show that $|a \times b| = |a||b|\sin(\alpha)$, where $\alpha$ is the angle in radians between the vectors $a$ and $b$.

**Solution 1.81:** $(a \times b, a \times b) = |a|^2|b|^2(1 - (\cos(\alpha))^2)$ from the exercise above. Now we may apply the identity $(a, b) = |a||b|\cos(\alpha)$.

**Exercise 1.82:** Show that $a \times b = 0$ if and only if $a = \alpha b$ for some real number $\alpha$.

**Exercise 1.83:** Show that $a \times (b \times c) = (a, c)b - (a, b)c$ and that $(a \times b) \times c = (a, c)b - (b, c)a$.

**Exercise 1.84:** Show that $(a, b)^2 + |a \times b|^2 = |a|^2|b|^2$.

We can extend the notion of the orientation of a basis of orthogonal vectors to apply to any linearly-independent sequence of vectors which form a basis. This is done, in essence, by having an arbitrary basis inherit the orientation of the closest orthogonal basis. This can be illustrated in 3-space with the aid of the cross product operation. In order to determine the orientation of a sequence of vectors $a, b, c$ in 3-space, it is useful to define the *vector triple product* $[a, b, c] := (a \times b) \cdot c$. (Here '$\cdot$' denotes the vector dot-product operation). We can directly check that $[a, b, c] = [b, c, a] = [c, a, b] = -[a, c, b] = -[b, a, c] = -[c, b, a]$. Also, $[a, b, c] = 0$ only when $a, b, c$ and 0 are coplanar, so for a basis $\langle a, b, c\rangle$ of $\mathcal{R}^3$, $[a, b, c] \neq 0$. Recall that the assignment of right-handedness or left-handedness to the basis with which we coordinatize $\mathcal{R}^3$ is an arbitrary choice. The resulting coordinate vectors are $e_1$, $e_2$, and $e_3$, and we can check whether any triad of independent vectors $\langle a, b, c\rangle$ is oriented identically to, or opposite from, $\langle e_1, e_2, e_3\rangle$.

Consider the triad of vectors $\langle a, b, c\rangle$ in $\mathcal{R}^3$. We can rotate 3-space to convert $a$, $b$, and $c$ to $\hat{a}$, $\hat{b}$, $\hat{c}$ such that $\hat{a} = (\hat{a}_1, 0, 0)$ with $\hat{a}_1 > 0$, and $\hat{b} = (\hat{b}_1, \hat{b}_2, 0)$ with $\hat{b}_2 > 0$, and $\hat{c} = (\hat{c}_1, \hat{c}_2, \hat{c}_3)$. Note that the triad $\langle a, b, c\rangle$ has the same orientation as $\langle e_1, e_2, e_3\rangle$ when $\hat{c}_3 > 0$ and the opposite orientation as $\langle e_1, e_2, e_3\rangle$ when $\hat{c}_3 < 0$. This means that when $\langle a, b, c\rangle$ and $\langle e_1, e_2, e_3\rangle$ have the same orientation, $\hat{c}$ lies on the same side of $plane[0, \hat{a}, \hat{b}]$ as $e_3$ lies with respect to $plane[0, e_1, e_2]$ in coordinatized 3-space.

We may show that $\langle a, b, c\rangle$ has the same orientation as that assigned to $\langle e_1, e_2, e_3\rangle$ when $[a, b, c] > 0$, and has the opposite orientation when $[a, b, c] < 0$, as follows. Let $\alpha = (a \times b, c)/(a \times b, a \times b)$. Then the projection of $c$ on $a \times b$ is $\alpha(a \times b)$ and $[a, b, c] = [a, b, \alpha(a \times b) + c - \alpha(a \times b)] = (a \times b) \cdot (\alpha(a \times b) + (a \times b) \cdot (c - (\alpha(a \times b)) = [a, b, \alpha(a \times b)] = \alpha[a, b, a \times b]$. Therefore $[a, b, c] = \beta[\hat{a}, \hat{b}, \hat{a} \times \hat{b}]$ where $\beta = (\hat{a} \times \hat{b}, \hat{c})/(\hat{a} \times \hat{b}, \hat{a} \times \hat{b})$. Note that $[x, y, x \times y] = |x \times y|^2 > 0$ whenever $x$ and $y$ are not parallel. Thus, $\text{sign}([a, b, c]) = \text{sign}(\beta) = \text{sign}((\hat{a} \times \hat{b}, \hat{c})) = \text{sign}(\hat{a}_1\hat{b}_2\hat{c}_3) = \text{sign}(\hat{c}_3)$. But then $\langle a, b, c\rangle$ has the same orientation as $\langle e_1, e_2, e_3\rangle$ when $[a, b, c] > 0$ and has the opposite orientation when $[a, b, c] < 0$.

**Exercise 1.85:** Show that

$$\operatorname{sign}([a, b, c]) = \operatorname{sign}([\alpha a, \beta b, \gamma c]) \operatorname{sign}(\alpha) \operatorname{sign}(\beta) \operatorname{sign}(\gamma).$$

Since $[a, b, a \times b] > 0$ when $a$ and $b$ are not parallel, we see that the triad of vectors $\langle a, b, a \times b\rangle$ has the same orientation as that assigned to the triad of vectors $\langle e_1, e_2, e_3\rangle$.

**Exercise 1.86:** Let $\langle a, b, c\rangle$ be a left-handed basis triad in left-handed 3-space. Which side of $line[0, a+b]$ does $c$ lie on in $plane[0, a+b, c]$ when viewed from the point $a$?

**Exercise 1.87:** Show that $[a \times b, b \times c, c \times a] = [a, b, c]^2$.

**Exercise 1.88:** Show that $a \times (b+c) + b \times (c+a) + c \times (a+b) = 0$.

**Exercise 1.89:** Show that $a \times (b \times c) + b \times (c \times a) + c \times (a \times b) = 0$.

**Exercise 1.90:** Show that $a \times b + b \times c + c \times a$ is normal to $plane[a, b, c]$.

**Exercise 1.91:** Show that in a left-handed orthonormal coordinate system, the cross product, $a \times b$, of two independent vectors, $a$ and $b$, lies on the side of the plane, $plane[0, a, b]$, such that the points 0, $a$, and $b$ in this order are arranged in clockwise order from the point of view of $a \times b$.

**Exercise 1.92:** Show that $(b \times a) \times b$ is in the same direction as the component of $a$ normal to $b$, which is $a - (a, b/|b|)b/|b|$.

**Exercise 1.93:** Show that the area of the parallelogram with vertices $0, a, b$, and $a \times b$ is $|a \times b|$.

**Exercise 1.94:** Show that, for vector functions, $u$ and $v$, of a real variable, $t$, the derivative $(u \times v)' = u' \times v + u \times v'$.

# 2 Curves

A curve in either 2-space or 3-space may be given parametrically by specifying coordinate functions. For example, a circle $x$ in the plane is defined by $x_1(t) = \cos(\mathrm{t})$ and $x_2(t) = \sin(\mathrm{t})$ for $-\pi < t \leq \pi$; the argument $t$ is called the *parameter* of the curve mapping $x$. The *graph* of $x$ is thus $\{ (x_1(t), x_2(t)) \mid x_1(t) = \cos(\mathrm{t}), \mathrm{x}_2(\mathrm{t}) = \sin(\mathrm{t}), -\pi < \mathrm{t} \leq \pi \}$. In general, a *plane curve* is a mapping from some interval $[a, b] \subseteq \mathcal{R}$ into $\mathcal{R}^2$. A *space curve* is a mapping from some interval $[a, b] \subseteq \mathcal{R}$ into $\mathcal{R}^3$. In either case, the domain of the curve mapping $[a, b]$ may be open, half-open, or closed, and may be the entire real line or may be bounded above and/or below. We thus follow modern tradition that a curve is a mapping, and *not* a point set; however, we often use language that confuses the graph of a curve with the curve mapping itself. The parametric representation of (the graph of) a space curve is not unique. The circle above can also be represented by $x_1(h) = (1 - h^2)/(1 + h^2)$ and $x_2(h) = 2h/(1 + h^2)$ for $-\infty < h \leq \infty$; this follows by introducing $\tan(t/2)$ for $h$.

**Exercise 2.1:** Show that, for $0 \leq \lambda \leq 1$, each intermediate curve $\{ (x, y) \mid \lambda(x + y - 2) + (1 - \lambda)(x^2 + y^2 - 1) = 0 \}$ lying "between" the curves, defined by $x + y = 2$ and $x^2 + y^2 = 1$ with $x \geq 0$ and $y \geq 0$, is a circular arc.

We usually assume that the component functions of a curve are continuous in the interval of interest, so that the curve itself is continuous there;

we shall disallow the case where all the component functions are constant, since this would yield a curve whose graph is a single point. If the component functions $x_1(t)$, $x_2(t)$, and $x_3(t)$ of a space curve $x$ in $\mathcal{R}^3$ are continuous periodic functions with a common period, then the curve $x$ is a *closed* curve when a sufficient range of $t$ is specified. If the three equations $x(t) = c$ have at most one solution $t$ for any vector $c$, the curve $x$ is *simple*, i.e., non-self-intersecting. If there are only a finite number of distinct vectors $c$ such that $x(t) = c$ has more than one solution $t$, and if $x(t) = c$ has a finite number of solutions $t$ for all vectors $c$, then $x$ is a *semi-simple* curve.

**Exercise 2.2:** Explain how a real-valued function of a single real argument $f : [a, b] \to \mathcal{R}$ defines a plane curve.

**Exercise 2.3:** Explain how an equation relating two real variables of the form $g(x, y) = 0$, where $g : \mathcal{R}^2 \to \mathcal{R}$, defines a plane curve.

**Solution 2.3:** Let $C = \{(x, y) \mid g(x, y) = 0\}$. When we can solve $g(x, y) = 0$ for $y$, so that we have $y = f(x)$ for some function $f$, then $(t, f(t))$ is a parametric representation of $C$. If there are several solution functions $y = f_1(x)$, $y = f_2(x)$, etc., then each corresponding parametric form $(t, f_i(t))$ generates a part of $C$.

**Exercise 2.4:** Explain how *two* equations relating three real variables, of the form $g(x, y, z) = 0$ and $h(x, y, z) = 0$, define a space curve. *Hint*: the intersection of two surfaces in $\mathcal{R}^3$ generally determines a space curve.

**Exercise 2.5:** Show that $(x_1(t), x_2(t), x_3(t)) = (a_1\cos(t) + b_1\sin(t), a_2\cos(t) + b_2\sin(t), a_3\cos(t) + b_3\sin(t))$ for $0 \le t < 2\pi$ is an ellipse embedded in a plane in 3-space, when $(a_1, a_2, a_3)$ and $(b_1, b_2, b_3)$ are linearly-independent.

**Exercise 2.6:** Does a plane curve, $x : \mathcal{R} \to \mathcal{R}^2$, exist such that $\mathcal{R}^2$ is "covered" by $x$, i.e., such that the graph of $x$ is $\mathcal{R}^2$.

## 2.1 The Tangent Curve

The direction of the line tangent to a space curve $x$ at a point $x(t)$ corresponding to the parameter value $t$ is the same as the direction of the vector

$x'(t)$ where the derivative vector $x'(t) := (x_1'(t), x_2'(t), x_3'(t))$. If, however, $x'(t) = 0$, we have no information about the tangent line (without considering $\lim_{h\to t} x'(h)$) even though it may well exist. The vector $x'(t)$ is called the *tangent vector* of $x$ at $t$. We shall usually assume that whenever we write the derivative of a function, we are implicitly postulating that the specified derivative exists. Note that if $x$ intersects itself at $x(t)$, so that $x(t) = x(t_1)$ for some value $t_1 \neq t$, then it is not precise to speak of *the* tangent line of $x$ at $x(t)$; rather we should speak of the tangent line of $x$ at $t$. Usually, we will assume that the point $x(t)$ is a *simple* point of $x$, i.e., $x(t)$ is not a self-intersection point of $x$, so that either usage makes sense. Note that $x'$ is a space curve, just as $x$ is. We often will be interested in the *unit tangent* curve associated with the space curve $x$ defined by $u(t) = x'(t)/|x'(t)|$. Note that $u$ does not exist when $= |x'| = 0$.

**Exercise 2.7:** Show that $x'(t)$ is parallel to the line tangent to $x$ at the point $x(t)$.

**Solution 2.7:** The point $(1-h)x(t)+hx(t+\delta)$ lies on the line through $x(t)$ and $x(t+\delta)$ for any choice of $h$. Thus, taking $h = 1/\delta$, we have that $x(t) + (x(t+\delta) - x(t))/\delta$ lies on $line[x(t), x(t+\delta)]$ for $\delta \neq 0$. Then, in the limit as $\delta \to 0$, we have that $x(t)+x'(t)$ lies on the tangent line to $x$ at $x(t)$.

**Exercise 2.8:** Show that the extended chain differentiation rule: $x(g(t))' = x'(g(t))g'(t)$ holds where $x : \mathcal{R} \to \mathcal{R}^3$ and $g : \mathcal{R} \to \mathcal{R}$.

**Exercise 2.9:** Let $x := (x_1(t), x_2(t))$ be a plane curve. Show that $n := (x_2'(t), -x_1'(t))$ is a normal vector of $(x_1, x_2)$ at $x(t)$, so that $line$ $[x, x+n]$ is perpendicular to the tangent line $line[x, x+x']$ in $\mathcal{R}^2$.

**Solution 2.9:** When $x$ is a plane curve, the 90° rotation of the tangent vector of $x$ at $x(t)$ is a normal vector of $x$ at $x(t)$; equivalently the inner product $((x_1', x_2'), (x_2', -x_1'))$ is 0.

**Exercise 2.10:** Let $x := (x_1(t), x_2(t))$ be a plane curve. Compute a companion offset curve $y$ such that the curve $y$ consists of those points that are $\alpha$ units perpendicularly displaced from $x$.

**Solution 2.10:** Let $u(t) = x'(t)/|x'(t)|$. Let $v(t) = (-u_2(t), u_1(t))$. Note $(u(t), v(t)) = 0$. Then the plane curve $x + \alpha v$ is the $\alpha$-west offset curve of $x$ and the plane curve $x - \alpha v$ is the $\alpha$-east offset curve of $x$. In general, these offset curves will contain loops and/or cusps in neighborhoods where the curve $x$ turns sharply.

**Exercise 2.11:** Let $x(t)$ be a point on the space curve $x$ which is closest to a given point $p$ among all the points on the trace of the curve $x$. Show that the vector $p - x(t)$ is perpendicular to the vector $x'(t)$. *Hint*: determine $t$ so as to minimize $|p - x(t)|^2$.

## 2.2 Curve Parameterization

For a space curve mapping $x(t)$ with $t \in [a, b]$, as $t$ *increases* continuously from the value $a$ to the value $b$, the uncountably infinite sequence of points $\langle x(t) \in \mathcal{R}^3 \mid a \le t \le b \rangle$ is produced. We call this sequence the *(ordered) trace* of $x$. The set of points present in the trace of $x$ coincides with the graph of $x$, but unlike the graph of $x$, a point may occur more than once in the trace of $x$ when $x$ is non-simple. We say that the curve $x$ is *parameterized* by the parameter $t$ and that $x(t)$ is a *parameterization* of the trace of $x$. There are infinitely many curves, each with an associated parameter ranging over some interval, that produce the same trace as the trace of the curve $x$; each of these curves can be considered to be a *reparameterization* of the curve $x$.

**Exercise 2.12:** Given the trace sequence and the indexing parameter interval $[a, b]$ of an unknown space curve mapping $x$, is the mapping $x$ uniquely determined?

**Solution 2.12:** No, the trace of $x$ suppresses the magnitude of the tangent vector $x'$.

A well-behaved space curve has a tangent line everywhere except at cusps. The only reason that $x'(t) = 0$ can hold is for $x$ to fail to be a so-called *regularly parameterized* curve at $t$; $x(t)$ may be interpreted as the position at time $t$ of a moving particle with velocity vector $x'(t)$; we may have an ill-behaved velocity vector, where the particle may come to rest and then reaccelerate; it may even exhibit *retrograde* motion, where the moving particle retraces part of the path that it has already traversed. (It is useful to define the *non-retrograde trace* of a space curve $x$ to be the trace of $x$ with the retrograde sections removed. If we choose a non-halting parametrization with a constant length velocity vector, or at least with a non-zero-length continuous velocity vector, however, then $x'(t) = 0$ can never occur. A *regularly parameterized* curve $x$ is a parameterization of the trace of $x$ such that $x'(t)$ exists for almost all values of $t$ in the associated domain interval and satisfies $x'(t) \neq 0$ whenever $x'(t)$ exists. The derivative vector $x'(t)$ then, of course, fails to exist at cusps in a regularly parameterized curve $x$. But for an irregular parameterization, cusps can be hidden so that $x'$ may be zero at a cusp. For example, the planar curve $x(t) = (t^3, t^2)$ has a cusp at $t = 0$ and $x'(0) = 0$.

Note that smooth component functions are no guarantee that we have a regular parameterization; the velocity vector of a curve may reverse itself smoothly, but the result is still not regular. The points on a regularly parametrized curve $x$ where $x'(t)$ exists are called *regular* points of $x$ and the points where $x'(t)$ fails to exist are called *singular* points of $x$. In addition to cusps, singular points can be points of discontinuity (if discontinuous component functions are allowed) or points where one or more of the component functions is not differentiable. We generally disregard these latter two cases and assume that they do not arise.

**Exercise 2.13:** What is a *cusp*? *Hint*: consider the piecewise defined curve $x(t) = (12t, 9t, 0)$ for $-\infty \leq t \leq 1$, and $x(t) = (4t + 8, 3t + 6, 0)$ for $1 < t \leq \infty$.

**Exercise 2.14:** If $x$ is a continuous curve whose trace does not exhibit retrograde motion, is $x$ a regularly parameterized curve?

**Solution 2.14:** Almost, but $x'(t) = 0$ is not ruled out.

**Exercise 2.15:** Carefully define the non-retrograde trace of a curve. Does the trace of a regularly parameterized curve $x$ coincide with its non-retrograde trace?

**Solution 2.15:** Not necessarily. The curve $x$ may be explicitly piecewise defined so as to exhibit retrograde motion. For example, define $x(t) = y(t)$ for $0 \leq t \leq 1$ and $x(t) = y(2 - t)$ for $1 \leq t \leq 2$, where $y$ is a regularly parameterized curve. Then $x$ is a regularly parameterized curve and $x(1)$ is a singular point of $x$. (Note the left and right derivatives of $x$ exist at 1, but are not identical).

**Exercise 2.16:** If the trace of $x$ is non-retrograde, is it necessarily free of sections of overlap?

**Exercise 2.17:** Explain when the trace of a planar curve $x$ is an isometric transformation of the graph of a real-valued function of a real variable.

**Solution 2.17:** If the trace of $x$ is non-retrograde and there exists a line $L$ through the origin in the plane of $x$ such that all the vectors $x'$ lie to the same side of $L$ in the plane of $x$, then the trace of $x$ is a rotation of the graph of a function that maps $\mathcal{R}$ to $\mathcal{R}$.

Note that a space curve $x$ which is defined so that $x'(t) = 0$ for at most finitely many values $t_1 < t_2 < \ldots < t_n$ is then regularly parameterized in between the *non-regular points* $x(t_1), \ldots, x(t_n)$. A non-regular point of a space curve may or may not be exhibited as a singular point when a completely regular parameterization having the same trace is examined.

**Exercise 2.18:** Let $x(t)$ with $a \le t \le b$ be a regularly parameterized space curve and let $r : [c, d] \to [a, b]$ be a real-valued function of a real argument. Note $\{x(r(h)) \mid c \le h \le d\} \subseteq \{x(t) \mid a \le t \le b\}$. What are conditions on $r$ which ensure that $x(r(h))$ with $c \le h \le d$ is a regularly parameterized space curve? What conditions on $r$ ensure that $\{x(r(h)) \mid c \le h \le d\} = \{x(t) \mid a \le t \le b\}$? *Hint*: let $y(h) = x(r(h))$ and consider when $y'(h) = x'(r(h))r'(h)$ can be zero.

**Solution 2.18:** If $r$ is a continuous strictly increasing function on $[c, d]$ with $r(c) = a$ and $r(d) = b$, then $y(h) := x(r(h))$ is a regularly parameterized space curve with the same trace as $x$. Indeed, the class of all such space curves obtained by choosing $r$ to be a continuous strictly increasing function on $[c, d]$ with $r(c) = a$ and $r(d) = b$ is exactly the class of all regularly parameterized space curves which are *equivalent* to $x$ in that they have the same trace. If $x$ is irregularly parameterized, however, it may be that no curve of the form $x(r(h))$ is a regular parameterization. For $x$ and $y$ to have the same graph, it is only necessary that $\{r(h) \mid h \in [c, d]\} = [a, b]$.

## 2.3 The Normal Curve

Locally, a turning space curve $x$, which we take to be regularly parameterized, bends away from its tangent line in a particular plane called the *osculating plane* at $x(t)$. The tangent line of $x$ at $x(t)$ lies in the osculating plane at $x(t)$, and the vector $x(t) + x''(t)$ also lies in this osculating plane.

This is because $x''$ is in the direction of the change of $x'$. The osculating plane at $x(t)$ also contains the so-called principal normal line to $x$ at $x(t)$. There is an entire family of lines in $\mathcal{R}^3$ that are normal to the tangent line of $x$ that passes through $x(t)$; the *principal normal line* at $x(t)$ passes through $x(t)$ and is parallel to the *principal normal vector* of $x$ at $x(t)$, which is the unit vector in the direction of the vector component of the vector $x''(t)$ normal to $x'(t)$. The principal normal vector of $x$ at $x(t)$ is denoted by $n(t)$. If $n(t) = 0$, the curve $x$ is straight at $x(t)$, with no unique osculating plane there. Such a point is called an *inflection* point of $x$.

**Exercise 2.19:** Show that the osculating plane of $x$ at $x(t)$ is $plane(x, x + x', x + x'')$.

**Exercise 2.20:** What is the plane $P(t)$ that is normal to the space curve $x$ at $x(t)$?

**Solution 2.20:** $P(t) = \{ v \in \mathcal{R}^3 \mid (v - x(t)) \cdot x'(t) = 0 \}$.

Consider $n(t)$, the principal normal vector of $x$ at $t$. When $x'' \neq 0$, $n$ is the unit vector in the direction of the component of $x''$ normal to $x'$. Thus $n = (x' \times x'') \times x'/|(x' \times x'') \times x'| = (x' \times x'') \times x'/|x' \times x''| \cdot |x'|$. When $x'' = 0$, $n$ is determined by continuity or, if necessary, arbitrarily. Note that the osculating plane of $x$ is $plane(x, x + u, x + n)$.

**Exercise 2.21:** Let $x$ be a regularly parameterized space curve. Show that the associated principal normal vector $n = [| x' \mid x'' - (x', x'')]/ \| x' \mid x'' - (x', x'') |$. (The vector $(x'', u)u - x''$ is the centrifugal force acting on the particle tracing out the curve $x$ with the velocity $x'$, where $u$ is the unit normal vector of $x$.)

Let $b = u \times n$. The vector $b(t)$ is the *binormal vector* of $x$ at $x(t)$. Note $|b| = 1$, when $x'' \neq 0$. The *normal plane* of $x$ is $plane(x, x + n, x + b)$ and the *rectifying plane* of $x$ is $plane(x, x + u, x + b)$ when $x'' \neq 0$.

At any value of $t$ in the domain of $x$, the vectors $u$, $n$, and $b$ form a left-handed coordinate system in left-handed 3-space (and a right-handed coordinate system in right-handed 3-space) called the *moving trihedral* on the curve $x$. Note $u = n \times b$, $n = b \times u$, and $b = u \times n$.

## 2.4 Envelope Curves

Consider a family $F$ of regularly-parametrized space curves continuously depending on a parameter $p$, $F := \{x_p \mid a \leq p \leq b\}$. Suppose $v(p)$ is a space curve such that each point of $v$ lies on some curve in $F$ and moreover that every curve in $F$ shares at least one point *and* associated tangent line with $v$. That is, for each $p \in [a,b]$, $v(p) = x_p(t)$ for some particular value $t$ which may depend on $p$; moreover, $v'(p)$ is also a tangent vector for the curve $x_p$ at the point $x_p(t)$, i.e., $v'(p) = \alpha x'_p(t)$ for some non-zero scalar $\alpha$. Then $v$ is called an *envelope* curve of the family $F$. Note that every regularly parametrized space curve $x$ is itself an envelope curve of the family of tangent lines of $x$.

Note that an envelope curve of the family $F$ may be determined as $v(p) = x_p(r(p))$ where $r(p)$ is a root value $t$ such that the inner product $((\partial_p x_p)(t), n_p(t)) = 0$, where $n_p(t)$ denotes the normal vector of $x_p$ at $x_p(t)$, and $(\partial_p x_p)(t)$ denotes the derivative vector of $x_p$ with respect to the parameter $p$, evaluated at $t$.

**Exercise 2.22:** What is the envelope curve of the family of principal normal lines of a space curve $x$?

**Exercise 2.23:** Does every one parameter family of regularly parameterized space curves have an envelope curve? *Hint*: consider a family of circles.

## 2.5 Arc Length Parameterization

The length of a curve is called its *arc length.* A curve is called *rectifiable* if its arc length exists and is finite between any two points of the curve.

**Exercise 2.24:** Show that the length of the rectifiable curve $x$, starting at the point $x(l)$ and ending at the point $x(h)$ is $\int_{l<t<h} |x'(t)|$.

Let $x = x(t)$ with $0 < t < a$ be a regularly parameterized space curve. The arc length of $x$ for $0 < t < h$ is given by the function $s(h) := \int_{0<t<h} |x'(t)|$. $x'$ is the tangent vector of $x$ and $s' = |x'|$. Recall that $u = x'/|x'|$ is the unit tangent vector of $x$ at $t$. Note $x' = s'u$.

It is convenient to consider a curve $x$ parameterized by arc length, so that $x(t)$ is such that $s(t) = t$. When $x$ is a regularly parameterized space curve, we can always define the arc length parameterized curve, $\hat{x}(h) = x(s^{-1}(h))$. It is also generally convenient to suppose that $x$ has no inflection points whereat $x'' = 0$. Note that $\hat{x}(s(h)) = x(h)$ and thus $x' = \hat{x}_s s'$ where the subscript $s$ denotes differentiation with respect to the arc length argument of $\hat{x}$. Of course, $\hat{x}' = \hat{x}_s$, but in this case we often use the subscript notation for emphasis.

Now, when $x$ is an arc length parameterized curve, $|x'| = s' = 1$ and $u = x'$. Also $x''$ is perpendicular to $x'$ since $x'$ is a curve on the surface of the unit sphere and $x''$ is its tangent vector which is always normal to the radius vector $x'$. Thus $(x', x'') = 0$ and $n = x''/|x''|$.

**Exercise 2.25:** Prove algebraically that $((x/|x|)', (x/|x|)) = 0$.

**Solution 2.25:** $(x/|x|)' = x'/|x| - (x, x')x/|x|^3$.

**Exercise 2.26:** Show that if $x$ is an arc length parameterized curve, then $s'(h) = |x'(h)| = 1$, and conversely.

**Solution 2.26:** First compute $s'(h) = |x'(h)|$. Then observe that, by definition, $s(h) = h$ when $x$ is an arc length parameterized curve, so that $s'(h) = 1$. Alternatively, when $x$ is merely a regularly parameterized curve, then $\hat{x}(h) = x(s^{-1}(h))$ is an arc length parameterized curve, and $|\hat{x}'(h)| = |x'(s^{-1}(h))s^{-1'}(h)| = |x'(s^{-1}(h))|s^{-1'}(h) = s'(s^{-1}(h))s^{-1'}(h) = (s(s^{-1}(h)))' = h' = 1$.

**Exercise 2.27:** If $x$ is an arc length parameterized curve, is $x'$ also an arc length parameterized curve?

## 2.6 Curvature

The way that a space curve $x(t)$ turns in space is described by a real-valued positive function $K(t)$ called the *curvature* function of $x$. The curvature of a circle is the constant function equal to the reciprocal of the radius of the circle. We shall show below that, when $x''(t) \neq 0$, there is a circle lying in the osculating plane which is tangent to $x$ at $x(t)$ and which has the same curvature as the curve at $x(t)$. This is the osculating circle at $x(t)$ whose radius is $1/K(t)$, where $K(t) = [|x'(t)|^2|x''(t)|^2 - (x'(t) \cdot x''(t))^2]^{1/2}/|x'(t)|^3$ is the curvature of the curve at $x(t)$.

Let $x(s)$ be an arc length-parameterized curve. In this case, the *curvature* function, $K(s)$, of the arc length-parameterized curve $x$ at $x(s)$ is computable as $K = |x''|$. Note $|x''| \geq 0$. The curvature $|x''|$ measures the rate of turning of the tangent vector, $x'$, and the tangent line turns most sharply at a point of maximum curvature of $x$.

We may follow Spivak [Spi75] to see that $1/K$ is the radius of the osculating circle at $x$. Suppose that $K \neq 0$ and recall that the osculating circle is tangent to the tangent line of $x$ at $x(s)$ and lies in the osculating plane,

*plane*$(x, x+x', x+x'')$, so the center $c$ of the osculating circle lies on the line, *line*$[x, x+x'']$. Thus there exists a scalar $a$ such that $c = x + ax''$. Now we may construct the osculating circle as the limit as $m \to s$ of circles defined by three necessarily non-colinear points $x(s)$, $x(s_1)$, and $x(s_2)$, with $s < s_1 < s_2 < m$. In each such circle, the distances from $x(s)$, $x(s_1)$, and $x(s_2)$ to the circle center $e$ are identical, so, if we consider the function $f_e(h) = |e - x(h)|$, there exist particular values, $h_1$ in the interval $(s, s_1)$ and $h_2$ in the interval $(s_1, s_2)$, such that $f_e'(h_1) = f_e'(h_2) = 0$. But then, similarly, there exists a value $r$ in the interval $(h_1, h_2)$ where $f_e''(r) = 0$. Thus, in the limit, as $h_1$, $h_2$, and $r$ are all squeezed together, we have $e \to c$ and $f_c'(s) = 0$ and $f_c''(s) = 0$. Now $f_c'(s) = |c - x(s)|'$ and $f_c''(s) = |c - x(s)|''$, so we have $|c - x|' = 2(x', x - c) = 0$ and $|c - x|'' = 2(x', x') + 2(x'', x - c) = 0$. The radius of the osculating circle is $|c - x|$ and since $c = x + ax''$, $|c - x|'' = 0$ yields $-a(x'', x'') = -(x', x')$. And, because $x$ is an arc length parameterized curve, $(x', x') = 1$, so $a = 1/|x''|^2$ and hence $|c - x| = 1/|x''| = 1/K$ where $K = |x''|$.

Now for the arc length parameterized curve $x$ with $K \neq 0$, the principal normal vector $n = x''/|x''|$, so $n = x''/K$, $x'' = Kn$, and the binormal vector $b = x' \times x''/K$. Note $n$ is directed "inward" so that $x + n$ lies in the osculating plane on the same side of $x$ as the osculating circle. As $x$ changes from being convex to being concave (with respect to some fixed axis vector), the osculating circle and the normal vector switch from one side of $x$ to the other in the osculating plane. The vector $Kn$ is called the *curvature vector* of $x$.

**Exercise 2.28:** Show that the center of the osculating circle of $x$ at the point $x(t)$ is $x(t) + n(t)/K(t)$.

The curvature function $K(t)$ of a space curve $x(t)$ changes its apparent algebraic form as $x$ is reparameterized. But $K$ is an *intrinsic* property of $x$ in the sense that the curvature associated with a given point in the trace of $x$ is the same value under every regular parameterization of $x$, so that the change in algebraic form is only apparent, and can always be simplified to a common expression independent of the regular parameterization being employed.

**Exercise 2.29:** Let $K(s)$ be the curvature function of the arc length parameterized curve $x$. What is the curvature function of the reparameterized curve $x(f(t))$?

**Solution 2.29:** $K(f^{-1}(t))$.

For a regularly parameterized curve $x$, The curvature $K$ at a point $x(t)$ can be computed as $K = [|x'|^2|x''|^2 - (x', x'')^2]^{1/2}/|x'|^3$. Note that for a regularly parameterized curve, $(x', x'')$ is not necessarily zero, as it is for the arc length parameterized form of $x$. This expression for $K$ follows from repeated use of the chain rule with the identity $x' = \hat{x}_s s'$, where the subscript $s$ denotes differentiation with respect to arc length, $s$.

**Exercise 2.30:** Show that the curvature $K$ for a regularly parameterized curve $x$ can be computed as $K(t) = [|x'|^2|x''|^2 - (x', x'')^2]^{1/2}/|x'|^3$.

**Solution 2.30:** $s' = |x'|$ so $\hat{x}_s = x'/|x'|$. $|\hat{x}_{ss}| = |(x'/s')'/s'| = |(x'' - (x'', x'/|x'|)x'/|x'|)/|x'||$. Note this numerator is the component of $x''$ normal to $x'$.

**Exercise 2.31:** Show that $K = |x' \times x''|/|x'|^3$ for a regularly parameterized curve $x$.

**Exercise 2.32:** Show that $x'' = s''u + s'^2 Kn$ for a regularly parameterized space curve $x$.

**Solution 2.32:** $x' = s'u = s'\hat{x}_s$, so $x'' = s'u's' + s''u$, and $u' = Kn$, thus $x'' = s''u + s'^2 Kn$.

## 2.7 The Frenet Equations

Let us consider the moving trihedral unit vectors $u$, $n$, and $b$, as functions of arc length, $s$. Since $|u| = |n| = |b| = 1$, $u(s)$, $n(s)$, and $b(s)$ are all curves on the surface of the unit sphere, and so $(u', u) = (n', n) = (b', b) = 0$, and also $u = n \times b$, $n = b \times u$, and $b = u \times n$. We may compute $u'$, $n'$, and $b'$ as follows. First $u' = x''$, so $u' = Kn$. Now $b' = (u \times n)' = u \times n' + u' \times n = u \times n'$ since $u' \times n = Kn \times n = 0$. Thus $(b', u) = 0$ and $(b', n') = 0$. But $(b', u) = 0$ and $(b', b) = 0$ implies that $b'$ is parallel to $n$, so $b' = Tn$ for some scalar function $T(s)$. Following DoCarmo [DoC76], the function $T(s)$ is called the *torsion* function of the arc length parameterized curve $x$ at $x(s)$. $|b'| = |T|$ measures the rate of turning of the osculating plane, just as $|u'| = |K| = K$ measures the rate of turning of the normal plane. Note at an inflection point where $x'' = 0$, we have $K = 0$.

Finally $n' = (b \times u)' = b \times u' + b' \times u = b \times (Kn) + (Tn) \times u = K(-u) + T(-b)$. We thus have nine differential equations called the Frenet equations:

$$\begin{aligned} u' &= Kn \\ n' &= -Ku - Tb \\ b' &= Tn. \end{aligned}$$

These equations, with given initial conditions: $u(0) = u_0$, $n(0) = n_0$, $b(0) = u_0 \times n_0$, determine an arc length parameterized space curve intrinsically in terms of the scalar curvature and torsion functions. We can include the three additional differential equations $x' = u$ with $x(0) = x_0$ to explicitly obtain the arc length parameterized curve $x$.

**Exercise 2.33:** Is the torsion $T(s)$ of an arc length parameterized curve $x$ necessarily a positive function?

If $K = 0, x$ is a straight line. If $T = 0, x$ is a planar curve, and if $T = 0$ and $K$ is a positive constant, $x$ is a circular arc. If $T$ is a constant value, $x$ is a curve spiraling around a central curve, $y$, whose torsion is 0, and $x$ is an involute of $y$. If $K/T$ is constant then $x$ is a helix.

**Exercise 2.34:** Show that if $K = 0$, then the torsion function $T$ does not influence the straight line space curve defined by the Frenet equations. What does $T$ influence in this case?

**Solution 2.34:** When $K = 0$, the equations $n' = -Tb$ and $b' = Tn$ determine how the translated normal and binormal vectors $x + n$ and $x + b$ rotate in the normal plane as we move along the straight line space curve $x$.

**Exercise 2.35:** Show that if we start with $u(0) = u_0$, $n(0) = n_0$ and $b(0) = b_0$ such that $\langle u_0, n_0, b_0 \rangle$ form an an orthonormal basis of $\mathcal{R}^3$, then $u(s)$, $n(s)$, and $b(s)$ as determined by the Frenet equations always form an orthonormal basis for every value of $s$.

**Solution 2.35:** (J. Eastham Jr.) Let $V$ row $1 = u$, $V$ row $2 = n$, and $V$ row $3 = b$. Thus $V(s)$ is a $3 \times 3$ matrix with $V(0)$ having the orthonormal rows $u_0$, $n_0$ and $b_0$. Let $H = \begin{bmatrix} 0 & K & 0 \\ -K & 0 & -T \\ 0 & T & 0 \end{bmatrix}$. Thus $H(s)$ is a skew-symmetric matrix such that $H^{\mathrm{T}} = -H$. The nine Frenet equations may now be written in terms of the matrix derivative $V'$ as

$V' = HV$. Note that $V(0)^{\mathrm{T}}V(0) = I = V(0)V(0)^{\mathrm{T}}$. We wish to show that $V$ evolves so that $V(s)V(s)^{\mathrm{T}} = I$, that is so that the rows of $V(s)$ are orthonormal for all $s \geq 0$. But the rates of change $(V^{\mathrm{T}}V)' = V^{\mathrm{T}'}V + V^{\mathrm{T}}V' = V^{\mathrm{T}}H^{\mathrm{T}}V + V^{\mathrm{T}}HV = V^{\mathrm{T}}(H^{\mathrm{T}} + H)V = 0$. Thus, $V$ evolves via the Frenet equations so that $V(s)^{\mathrm{T}}V(s) = V(0)^{\mathrm{T}}V(0) = I$, and thus $V(s)V(s)^{\mathrm{T}} = I$ as well.

The torsion $T$ can be computed as $T(t) = -(x' \times x'') \cdot x'''/|x' \times x''|^2$ for a regularly parameterized curve $x$ at $x(t)$. When $x$ is an arc length parametrized curve, $T = -(x' \times x'') \cdot x'''/K^2$.

**Exercise 2.36:** Let $x(t) = a + bt + ct^2$, where $a, b, c \in \mathcal{R}^3$. Compute the curvature $K(t)$ of $x$ and draw a graph of $K(t)$ vs. $t$ for some interesting choice of $a$,$b$, and $c$. Also compute the torsion $T(t)$ of $x$.

**Exercise 2.37:** Show that $K = (u', n)/|x'| > 0$ and $T = (n', b)/|x'|$ for a regularly parameterized curve $x$ for which $x'' \neq 0$.

**Exercise 2.38:** Show that the center of the *osculating sphere* of the arc length parameterized space curve $x$ is $x + n/K + bK'/(K^2T)$. *Hint*: define the osculating sphere by the equation $| x - c |^2 = \alpha$. Differentiate three times to obtain three equations, and then solve for $(c_1, c_2, c_3)$.

## 2.8 Involutes and Evolutes

Any curve $x$ that cuts the tangent lines of a curve $c$ orthogonally ( i.e., such that $(x', c') = 0$) is an *involute curve* of $c$, and if $x$ is an involute of $c$, then $c$ is an *evolute curve* of $x$. In general, a curve has an infinite number of evolutes and involutes. Several curves may have a common evolute curve or a common involute curve, whence they are all involutes or evolutes of their common evolute or common involute.

We may follow Wardle [War65] to construct explicit formulas for the involutes and evolutes of a space curve $x$. Let the space curve $i$ be an involute of the arc length parameterized curve $x$. Then each tangent line of $x$ is cut orthogonally by $i$, so $i(s) = x(s) + \alpha(s)x'(s)$ for some function $\alpha$ which may variously assume a positive or negative value depending on where $i$ cuts the tangent line $line[x, x + x']$. Note $s$ denotes the arc length parameter for $x$; $i$ is *not* generally arc length parameterized by the arc length parameter of $x$. Thus, $i' = x' + \alpha'x' + \alpha x''$. But $x'' = Kn$, so $i' = x' + \alpha'x' + \alpha Kn$. And $(i', x') = 0$, so, since $(x', x') = 1$ and

$(x', n) = 0$, we have $1 + \alpha' = 0$. Hence, $\alpha = -s + \delta$, where $\delta$ is a constant of integration. Thus $i = x + (\delta - s)x'$ is the general form for an involute of $x$; different choices of $\delta$ result in different involutes. Note again that the involute $i$ is parameterized with the arc length parameter of $x$; the curve $x(s) + (\delta - s)x'(s)$ is *not* generally arc length parameterized, nor even necessarily regularly parameterized.

Now suppose that $e$ is an evolute of the arc length parameterized space curve $x$, so that $x$ is an involute of $e$. Then $(x'(s), e'(s)) = 0$ and the point $e(s)$ that corresponds to the point $x(s)$ on $x$ must lie in the normal plane of $x$ taken at the point $x(s)$. Note $s$ denotes the arc length parameter for $x$; $e$ is *not* generally arc length parameterized. The normal plane of $x$ is $plane[x, x + n, x + b]$, so $e = x + \lambda n + \mu b$ for particular functions $\lambda(s)$ and $\mu(s)$. Also, the line segment $segment[e, x]$ coincides with the tangent line of $e$, so the tangent vector $e'$ has the same direction as the vector $x - e$, and thus $e' = \rho(\lambda n + \mu b)$ for some particular function $\rho$.

Now we differentiate $e$ to write $e' = x' + \lambda' n + \lambda n' + \mu' b + \mu b'$, and using the Frenet equations for $x$ yields $e' = x' + \lambda' n - \lambda(Ku + Tb) + \mu' b + \mu T n = (1 + \lambda K)u + (\lambda' + \mu T)n + (\mu' - \lambda T)b$. But since $x' = u$ and $(e', x') = 0$, we have $1 - \lambda K = 0$. Also, since $e' = \rho(\lambda n + \mu b)$, we have $\lambda' + \mu T = \rho\lambda$ and $\mu' - \lambda T = \rho\mu$.

But then $\lambda = 1/K$, and $\mu(\lambda' + \mu T) = \lambda(\mu' - \lambda T)$. This latter equation is equivalent to $\lambda\mu' - \mu\lambda' = (\lambda^2 + \mu^2)T$. We recall that $(\mu/\lambda)' = (\lambda\mu' - \mu\lambda')/\lambda^2$, so $(\mu/\lambda)' = T(\lambda^2 + \mu^2)/\lambda^2$. Thus $(\mu/\lambda)'/(1 + (\mu/\lambda)^2) = T$. Let $f(s) := \mu(s)/\lambda(s) = K(s)\mu(s)$. Then $f'/(1 + f^2) = T$ and $\int[f'/(1 + f^2)]ds = \int[1/(1 + f^2)]df = \tan^{-1}(f) + \beta$, where $\beta$ is a constant of integration. But then $\beta$ can be chosen so that $\tan^{-1}(K\mu) + \beta = \int T ds$, where the indefinite integral $\int T ds$ here denotes a function $F(s)$ such that $F' = T$. Now $\mu = \tan(F - \beta)/K$, and thus an evolute curve of $x$ has the form $e = x + [n + \tan(F - \beta)b]/K$; for each choice of the constant $\beta$, we obtain a different evolute of $x$. Note that $K$ is the curvature function of $x$; and the argument of $e$ is *not* the arc length parameter of $e$; rather it is the arc length parameter of $x$. Also note that the arc length parameterized curve $x$ has associated evolute curves defined only at points where $K \neq 0$ (and $n \neq 0$.)

When $x$ is a planar space curve, with the associated torsion function 0, choosing $\beta = 0$ yields the evolute curve $c := x + n/K$. This evolute space curve $c$ is formed by the locus of the centers of the osculating circles of $x$, and is called the *central evolute curve* of the planar space curve $x$. Note that the central evolute of the planar space curve $x$ is the envelope of the family of normal lines of $x$; the study of this envelope curve is the genesis of the more general notions of evolute and involute curves for an arbitrary space curve.

**Exercise 2.39:** Is every evolute curve of a planar space curve $x$ planar?

**Exercise 2.40:** Compute the central evolute of the ellipse given by $x(t) = (a \cdot \sin(t), b \cdot \cos(t))$.

**Exercise 2.41:** Show that the envelope curve $c$ of the family of lines normal to a planar space curve $x$ is the central evolute curve of $x$ and show that $c' = n$, i.e., the tangent vector of $c$ is the normal vector of $x$.

## 2.9 Helices

A $\Gamma$-*cylinder* surface is created by extruding or sweeping a simple planar closed curve $\Gamma$ along a line parallel to a central axis line. A $\Gamma$-*helix* is obtained by wrapping a wire about a $\Gamma$-cylinder with a uniform pitch angle. Thus a helix is a curve with the property that each of its tangent vectors is at a fixed angle to a particular fixed direction, which is the direction of extrusion of the corresponding cylinder. Now we shall drop the requirement that the generating planar curve $\Gamma$ is simple, or even closed; so, for example, a $\Gamma$-cylinder may be a plane or other variety of surface beyond those immediately considered to be cylinders. Indeed, a cylinder can now be described as a parallel ruled surface, and an associated helix is a curve that cuts these rule-lines at a fixed angle.

**Exercise 2.42:** Show that a regularly parameterized space curve $x$ is a helix if and only if $K/T$ is constant.

**Solution 2.42:** If $x$ is a helix, then $(u, d) = \cos(\alpha)$, where $u$ is the unit tangent vector of $x$ and where $d$ is a unit vector parallel to the central axis of the associated cylinder. Then $(u, d)' = (Kn, d) = 0$, so $d$ and $n$ are perpendicular, and hence $d = \delta u + \gamma b$ for some scalars, $\delta$ and $\gamma$, such that $\delta^2 + \gamma^2 = 1$. Therefore $\cos(\alpha) = (\mathrm{u}, \mathrm{d}) = \delta$. But differentiating $(n, d) = 0$ yields $(n', d) = (-Ku - Tb, d) = 0$, so $0 = (-Ku - Tb, \cos(\alpha)\mathrm{u} \pm \sin(\alpha)\mathrm{b}) = -\mathrm{K}\cos(\alpha) \pm \mathrm{T}\sin(\alpha)$, and thus $K/T = \pm\tan(\alpha)$, which is constant.

Conversely, if $K/T = \tan(\alpha)$, for $\alpha$ fixed, then $K\cos(\alpha) = \mathrm{T}\sin(\alpha)$ and $K\cos(\alpha)\mathrm{n} = \cos(\alpha)\mathrm{u}'$ and $T\sin(\alpha)\mathrm{n} = \sin(\alpha)\mathrm{b}'$, so $\cos(\alpha)\mathrm{u}' -$

sin($\alpha$)b$'$ = 0, and integrating this equation yields cos($\alpha$)u − sin($\alpha$)b = d where $d$ is a constant unit vector. Now $(u, d) = \cos(\alpha)$, so $u$ forms a constant angle, $\alpha$, with a fixed direction, $d$, and hence $x$ is a helix. Note this solution shows how the direction of the central axis can be computed, given the helix $x$, and it also exhibits the fact that each principal normal vector of a helix is orthogonal to the central axis of the associated cylinder.

**Exercise 2.43:** Show that a regularly parameterized space curve $x$ is a helix if and only if the binormal vector $b$ of $x$ always makes a fixed angle $\alpha$ with a particular direction $d$.

**Exercise 2.44:** Why do we require that the generating curve $\Gamma$ of a cylinder be planar?

## 2.10 Signed Curvature

The *signed z-directional curvature* of an arc length parameterized curve $x$ is defined as the rate of change of the angle $\theta_z(s)$ formed between the tangent vector $u(s)$ and the $xy$-plane. This is the derivative of $\theta_z$ with respect to arc length $s$, which is $\theta_z' = (|p|'(u,p) - |p|(u,p)')/(|p| \cdot |u \times p|)$ where $p = (u_1, u_2, 0)$. $\theta_z'$ is positive when the curve $x$ is turning up and negative when $x$ is turning down with respect to the $z$-axis direction. The notion of a $z$-directional curvature can be generalized to refer to an arbitrary direction.

**Exercise 2.45:** Show that, in the case where the curve $x$ is a plane curve defined by a twice-differentiable function $f$ as $x(t) = (t, f(t), 0)$, the signed $(0, 1, 0)$-directional curvature of $x$ is the *functional* signed curvature of the function $f$ defined as $f''/(1 + (f')^2)^{3/2}$.

**Exercise 2.46:** Let $x(t) = (x_1(t), x_2(t))$ be a regularly parameterized plane curve. Show that the curvature of $x$ is $K = | x_1'x_2'' - x_2'x_1'' | / ((x_1')^2 + (x_2')^2)^{3/2}$ and show that at the point $x(t)$, $x$ is turning left (so that $x$ is concave to its left and convex toward its right at $x(t)$) when $x_1'(t)x_2''(t) - x_2'(t)x_1''(t) > 0$, and $x$ is turning right when $x_1'(t)x_2''(t) - x_2'(t)x_1''(t) < 0$.

**Exercise 2.47:** Show that the natural unsigned curvature $K$ is a directional curvature, but with respect to the dynamically varying normal vector $n$. The space curve $x$ is always turning up with respect to the direction of $n$, which is consistent with $K \geq 0$.

**Exercise 2.48:** Show that the torsion $T$ is the dynamic directional curvature with respect to the varying binormal vector $b$.

**Solution 2.48:** $T(s) = \lim_{h \to 0}[angle(u(s+h), plane(0, u(s), n(s))) - angle(u(s), plane(0, u(s), n(s)))]/h$. Also, $angle(u(s), plane(0, u(s), n(s))) = 0$.

## 2.11 Inflection Points

Recall that a point $x(t)$ is an inflection point of the arc length parameterized space curve $x$ when $x''(t) = 0$. The point $x(t)$ is a *simple inflection point* of the arc length parameterized space curve $x$ when there exists a direction specified by a unit vector $d$ such that the signed $d$-directional curvature $\theta_d'(s)$ changes sign at $s = t$, i.e., there exists a value $\epsilon > 0$ such that $\theta_d'(t-\delta)\theta_d'(t+\delta) < 0$ for $0 < \delta < \epsilon$. Alternately, $x(t)$ is a simple inflection point of $x$ if the curve $x + x''$ "cuts across" the curve $x$ at the single point $x(t)$ so that $x''(t) = 0$ and the tangent lines $\{x(t) + \alpha x'(t) \mid \alpha \in \mathcal{R}\}$ and $\{x(t)+\beta(x'(t)+x'''(t)) \mid \beta \in \mathcal{R}\}$ intersect at the single point $x(t)$, in which case $x'(t)$ and $x'''(t)$ are linearly independent vectors. Inflection points are places where the curve is linear, at least momentarily, i.e., coincides in the direction with its tangent line. In general, inflection points can be classified as *crossing inflection points* and *touching inflection points*. The inflection point $x(t)$ is a crossing inflection point if the curve $x$ *crosses* its tangent line $\{x(t) + \alpha x'(t) \mid \alpha \in \mathcal{R}\}$ in the associated osculating plane. Otherwise the inflection point $x(t)$ is a touching inflection point. Note that a simple inflection point is a crossing inflection point.

**Exercise 2.49:** Show that in the case where the curve $x$ is a plane curve defined by an infinitely-differentiable function $f$ as $x(r) = (r, f(r), 0)$, the point $x(t)$ is a crossing inflection point when $f'$ is locally maximal or minimal at $t$; i.e., $f''(t) = 0$ and the next-higher derivative of $f$ which is non-zero at $t$ is of odd order, so that there exists an odd integer $h > 2$ such that $f^{(k)}(t) = 0$ for $k = 2, \dots, h-1$ and $f^{(h)}(t) \neq 0$.

**Exercise 2.50:** Show that a plane curve $x$ does not possess a simple inflection point if the side of the directed line defined by the vector $x'(t)$ in which the point $x''(t)$ lies is the same for all values of $t$. Show that a quadratic space curve has no inflection points.

**Exercise 2.51:** If $x(t)$ is a vector function of the scalar $t$, then the vector $x$ has an instantaneous axis passing through the origin, about which the vector $x$ is rotating at the parameter value $t$. This axis is orthogonal to $x$ and $x'$, and we shall call the vector $a = (x \times x')/(x, x)$, which coincides with the instantaneous rotation axis of $x$, the *spin vector* of $x$. The length of the spin vector $a$ is the angular rate of turn of $x$ about $a$. Show that, if $x$ is a regularly parameterized space curve, then the spin vector of $u$ is $b$, the spin vector of $b$ is $-u$, and the spin vector of $n$ is $Kb - Tu$. The vector $Kb - Tu$ is called the *Darboux vector* of $x$. Show also that $u'$, $n'$, and $b'$ are all orthogonal to $Kb - Tu$.

The points of the space curve, $x$, where $K' = 0$, are the points of sharpest turning. Such points are called *vertices* of $x$. Each point of a circle is a vertex, a parabolic space curve has only one vertex, and any simple planar closed curve has at least four vertices. At a vertex where $K \neq 0$, the curve is at an extreme point in the direction $-n$, opposite from the principal normal vector.

**Exercise 2.52:** (J. E. Kiefer) Find a non-planar simple closed curve which has fewer than four vertices.

**Solution 2.52:** Consider a cycloid of the form $\phi(t) = t$ and $r(t) = 1 + 2\cos(t)$ in polar coordinates with $0 \le t < 2\pi$ and the interpretation that a polar point $(r, \phi)$ with $r < 0$ denotes the opposite point $(-r, \phi + \pi)$. This is not a simple curve. The curvature is least at $t = 0$ and greatest at $t = \pi$ and monotonic in between, so there are only two vertices. Now lift the cycloid curve into 3-space, slightly separating the two branches of the curve at the origin, so that it becomes a simple closed curve in 3-space; it still has just two vertices. In cartesian coordinates, we have the space curve $(r(t) \cdot \cos(\phi(t)), r(t) \cdot \sin(\phi(t)), .1 \cdot \sin(t/2))$, where $r(t) = |1+2\cos(t)|$ and $\phi(t) = t + \pi(\cos(t) < -1/2)$, with $0 \le t < 2\pi$.

**Exercise 2.53:** Show that if a space curve $x$ can be projected onto a plane so that the resulting projection of $x$ is a line segment, then $x$ is a planar curve.

**Exercise 2.54:** Show that if a closed curve in 3-space is simple in every projection onto a plane, except one, then it is itself a simple planar closed curve.

**Exercise 2.55:** Given a curve $x(t)$ with $a \leq t \leq b$, propose an algorithm to compute the points of $x$ closest to a given point $p$. *Hint*: consider $|p - x(t)|^2$.

**Exercise 2.56:** Given the component functions: $x_1(t), x_2(t)$, and $x_3(t)$ for a space curve, $x$, with the parameter $t$ specified to be in the range $t_0 \leq t \leq t_1$, and given the values $r_1, r_2, \ldots, r_n$, where $t_0 \leq r_1 \leq r_2 \leq \ldots \leq r_n \leq t_1$, write programs to do the following.

1. Draw a graph of $x$ and the osculating circles of $x$ at the points $x(r_1), x(r_2), \ldots, x(r_n)$.
2. Draw a graph of $x$ and circles of radius $h$, which are centered at $x(r_1), x(r_2), \ldots, x(r_n)$, and which lie in the normal planes of $x$ at these points.
3. Draw a graph of $x$ with directed branches (line segments) of length $l$ at $x(r_1), x(r_2), \ldots, x(r_n)$, where the branches lie in the corresponding osculating planes and form random angles between $\alpha_0$ and $\alpha_1$ radians with the normal vectors.

# 3
# Surfaces

A surface in $\mathcal{R}^3$ can be defined in several ways in terms of functions of several variables. For a surface $S$, we may have a real-valued function $f : \mathcal{R}^3 \to \mathcal{R}$, such that $S = \{(x, y, z) \mid f(x, y, z) = 0\}$. Alternatively, we may have a vector-valued function $p : Q \subseteq \mathcal{R}^2 \to \mathcal{R}^3$, such that $S = \{p(u, v) \mid (u, v) \in Q \subseteq \mathcal{R}^2\}$. The component functions of $p$ are identified separately as $p_1(u, v)$, $p_2(u, v)$, and $p_3(u, v)$. We thus have the surface $S$ defined parametrically in terms of three separate coordinate functions of two independent parameters, $u$ and $v$, as $S = \{(p_1(u, v), p_2(u, v), p_3(u, v)) \mid (u, v) \in Q \subseteq \mathcal{R}^2\}$, where the coordinate functions $p_1$, $p_2$, and $p_3$ map from $Q \subseteq \mathcal{R}^2$ into $\mathcal{R}$. The variables $u$ and $v$ are the coordinates of a *parameter* point in $Q \subseteq \mathcal{R}^2$, and we can imagine the surface $S$ is formed by distorting the region $Q$. The variables $u$ and $v$ can also be considered to be coordinate values of a point on the surface $S$ located with respect to a curvilinear coordinate grid inscribed on the surface. This coordinate grid is composed of the space curves $p(u, v)$ with $u$ or $v$ fixed and the remaining variable $v$ or $u$ changing in order to trace out a grid curve on $S$. When $S$ is a so-called *single-valued* surface, then there is a real-valued function $z$, such that $S = \{(x, y, z(x, y)) \mid (x, y) \in U \subseteq \mathcal{R}^2\}$, where $z : U \subseteq \mathcal{R}^2 \to \mathcal{R}$. By *single-valued*, we mean that a line parallel to the $z$-axis intersects the surface at most once. This latter case is a trivial form of parametric representation with $p_1(u, v) = u$, $p_2(u, v) = v$, and $p_3(u, v) = z(u, v)$. Generally, we shall suppose that all the partial derivatives of the functions we consider exist, and that all mixed partials are invariant with respect to the order of differentiation.

For example, let $a(t)$ and $b(t)$ be two space curves with $0 \leq t \leq h$. Let $p(t, \lambda) = (1-\lambda)a(t)+\lambda b(t)$. Then $S = \{ p(t, \lambda) \mid (t, \lambda) \in [0, h] \times [0, 1] \}$ is a parametrically represented homotopic mixture surface defined by $a$ and $b$. Note we can obtain other homtopic mixture surfaces by replacing $\lambda$ by any adequately smooth monotonically-increasing function $m(\lambda)$ for which $m(0) = 0$ and $m(1) = 1$.

With some caveats, we can convert between the surface representations introduced above. Given the scalar-valued function $z(x, y)$ and the point set $U$, we have $p(u, v) = (u, v, z(u, v))$ and $Q = U$. Given the vector-valued function $p(u, v)$ and $Q$, we have $f(x, y, z) = \mathit{if}\,(x, y) \in Q\ \mathit{then}\ p_3(x, y) - z\ \mathit{else}\ 1$. Given $f(x, y, z)$, we have $p(x, y) = (x, y, root_z f(x, y, z))$, where $root_z E(z)$ denotes a value $r$ such that $E(r) = 0$, and $Q$ is the set of $(x, y)$-points for which $root_z f(x, y, z)$ exists. Given the vector-valued function $p(u, v)$ and the point set $Q$, we may, in principle, solve for $u$ and $v$ in terms of variables $x$ and $y$ by solving $p_1(u, v) = x$ and $p_2(u, v) = y$, to obtain $u = \tilde{u}(x, y)$ and $v = \tilde{v}(x, y)$ for certain functions $\tilde{u}$ and $\tilde{v}$. When such solution functions exist uniquely, then the corresponding surface $S$ is the single-valued surface $\{(x, y, z(x, y)) \mid (x, y) \in U \subseteq \mathcal{R}^2\}$, where $z(x, y) = p_3(\tilde{u}(x, y), \tilde{v}(x, y))$ and $U = \{(x, y) \mid x = p_1(u, v), y = p_2(u, v), (u, v) \in Q \subseteq \mathcal{R}^2\}$.

**Exercise 3.1:** Let $r(t)$ be a space curve, let $n(t)$ denote the unit normal vector space curve associated with $r$, and let $b(t)$ denote the unit binormal vector space curve associated with $r$. Consider the function $p : \mathcal{R}^2 \to \mathcal{R}^3$ defined as $p(s, t) = r(t) + d(\cos(s)n(t) - \sin(s)b(t))$, where $d$ is a constant. Describe the surface $\{p(s, t) | 0 \leq s < 2\pi,\ t \in [\alpha, \beta] \subseteq \mathcal{R}\}$.

## 3.1 The Gradient of a Function

Consider a differentiable function $z(q_1, \ldots, q_n) = z(q) \in \mathcal{R}$. The function $z$ maps $\mathcal{R}^n$ into $\mathcal{R}$ and the graph of $z$ is a surface in $\mathcal{R}^{n+1}$. Define the *gradient* of $z$ at $q$ as the vector $(\nabla z)(q) = (\partial_1 z(q), \ldots, \partial_n z(q)) \in \mathcal{R}^n$, where $\partial_i z$ denotes $\partial z / \partial q_i$. The symbol $\nabla$ denotes a functional suggestively defined by $\nabla = (\partial/\partial q_1, \ldots, \partial/\partial q_n)$. The operator $\nabla$ necessarily has a binding priority such that $\nabla z(q) = (\nabla z)(q)$. Note that when the domain of $z$ is $\mathcal{R}^1$, the gradient $\nabla z$ is just the 1-element vector consisting of the derivative $z'$.

Let $x(s)$ be an arc length parameterized curve in $\mathcal{R}^n$ and let the function $z : \mathcal{R}^n \to \mathcal{R}$ be given. Then $z(x(s))$ is a real-valued function of a single real variable, $s$, and $(z(x))' = \nabla z(x) \cdot x'$ by the chain rule. This is the rate of change of $z$ at $x(s)$ in the direction $x'(s)$, so, in general, the $v$-directional derivative of $z$ at $q$ is appropriately defined to be $\nabla z(q) \cdot v$, where $|v| = 1$. This is just the length of the component vector of $\nabla z(q)$ in the direction $v$. Thus the direction in $\mathcal{R}^n$ that has the maximum rate of positive change of $z$ at $q$ is just $\nabla z(q)$, since $\nabla z(q) \cdot (\nabla z(q)/|\nabla z(q)|) = |\nabla z(q)| = max_{|v|=1} \nabla z(q) \cdot v$. Similarly, the direction in $\mathcal{R}^n$ that has the maximum rate of negative change of $z$ at $q$ is just $-\nabla z(q)$. The vector $\nabla z(q)$ is the steepest ascent direction in $\mathcal{R}^n$ and the vector $-\nabla z(q)$ is the steepest descent direction in $\mathcal{R}^n$.

**Exercise 3.2:** What is the direction of the minimum rate of (positive or negative) change of $z$?

**Exercise 3.3:** What is the meaning of the statement $\nabla z(q) = 0$?

**Solution 3.3:** If $\nabla z(q) = 0$, the surface $S = \{a\&z(a) \mid a \in \mathcal{R}^n\}$ in $\mathcal{R}^{n+1}$ is flat at the point $q$. Thus the point $q\&z(q)$ is a local minimum, a local maximum, or a saddle point of the surface $S$ embedded in $\mathcal{R}^{n+1}$ defined by the function $z$, with respect to the direction $e_{n+1}$. (The vector-valued binary operator & denotes concatenation: $a\&b = [a \quad b]$.)

Consider the surface, $S = \{q \mid f(q) = 0\} \subset \mathcal{R}^3$, and consider a regularly parametrized space curve $x$ embedded in $S$. Then $f(x(s)) = 0$ for all $s$, so $\nabla f(x) \cdot x' = 0$ and therefore $\nabla f(q)$ is normal to the tangent vector at $q$ of every regular space curve embedded in $S$ and passing through $q$. Thus $\nabla f(q)$ is parallel to the line normal to $S$ at the point $q \in S$. When $\nabla f(q) = 0$, the point $q \in S$ is a *singular* point of $S$; there is an embedded curve in $S$ which has a cusp or other singularity at $q$.

Let the surface $S$ be variously represented by $\{q \mid f(q) = 0\}$ or $\{q \mid q = p(u, v), (u, v) \in Q \subseteq \mathcal{R}^2\}$ or $\{q \mid q = (x, y, z(x, y)), (x, y) \in U\}$. A downward directed normal vector of $S$ at the point $q \in S$ is $\nabla z(q_1, q_2)\&(-1)$ when $S$ is defined as $(x, y, z(x, y))$, so $\nabla z(q_1, q_2)\&(-1)$ is a multiple of $\nabla f(q)$. Also, the vectors $\nabla f(q)$ and $p_u(u, v) \times p_v(u, v)$, where $p(u, v) = q$, are both normal to $S$ at $q$, so they are both multiples of each other. Here $p_u$ denotes the derivative $\frac{\partial p}{\partial u}$ and $p_v$ denotes $\frac{\partial p}{\partial v}$. In fact, if $a(t)$ and $b(t)$ are any two regularly parameterized simple space curves embedded in $S$ which pass through the point $q$, then the tangents of $a$ and

$b$ at $q$ are tangents of the surface $S$, perpendicular to the corresponding surface normal at $q$, so that $\nabla f(q)$ and $a'(t_0) \times b'(t_1)$ are multiples of each other, where $a(t_0) = b(t_1) = q$. The special case $a(u) = p(u, v)$ and $b(v) = p(u, v)$ was used above in constructing the normal vector $p_u(u, v) \times p_v(u, v)$. (In the case where $a'(t_0)$ and $b'(t_1)$ are parallel, 0 is the multiple of $\nabla f(q)$ that equals $a'(t_0) \times b'(t_1)$). If $p_u(u, v) \times p_v(u, v) = 0$, the point $q = p(u, v)$ is a singular point of $S$.

Consider a curve $C$ in the $xy$-plane defined implicitly by $C := \{ (x, y) \mid g(x, y) = 0 \}$. Then $C$ is the intersection of the surface $S := \{ (x, y, z) \mid g(x, y) = z \}$ and the $xy$-plane. A normal vector $n$ to the curve $C$ at a point $p \in C$ can be obtained as the projection of a normal of $S$ at the point $p\&0$ into the $xy$-plane; thus $n = \nabla g(p)$. The tangent line of $C$ at the point $p$ is thus $\{ v + p \mid \nabla g(p) \cdot v = 0 \} \subset \mathcal{R}^2$.

**Exercise 3.4:** What is the plane that is tangent to the surface $S = \{q \mid f(q) = 0\}$ at $p \in S$?

**Solution 3.4:** $\{ v \in \mathcal{R}^3 \mid (v - p) \cdot \nabla f(p) = 0 \}$.

## 3.2 The Tangent Space and Normal Vector

It is convenient to restrict our attention to parametrically defined surfaces so that we may define a surface as a mapping from a domain $Q \subseteq \mathcal{R}^2$ into $\mathcal{R}^3$. Now given the surface $S = \{p(u, v) \in \mathcal{R}^3 \mid (u, v) \in Q \subseteq \mathcal{R}^2\}$, we may define the unit normal vector of $S$ at the point $(u, v) \in Q$ corresponding to the point $p(u, v) \in S$ as $N_p := (p_u \times p_v)/ \mid p_u \times p_v \mid$. Note the direction of $N_p$ is independent of the "shape" of $S$ at $p(u, v)$; the direction of $p_v \times p_u$ is the same as the direction of $-N_p$.

The *tangent space*, $T_p$, of the surface $S = \{p(u, v) \in \mathcal{R}^3 \mid (u, v) \in Q \subseteq \mathcal{R}^2\}$ at the point $(u, v) \in Q$ corresponding to the point $p(u, v) \in S$ is $plane[0, p_u(u, v), p_v(u, v)] = subspace[p_u(u, v), p_v(u, v)] \subset \mathcal{R}^3$. The *tangent plane* of $S$ at the point $p(u, v)$ is the flat $T_p + p$. More generally, the tangent space $T_p = \{a \mid a \cdot N_p = 0\}$.

A surface $S$ viewed in the neighborhood of a point $p$ in the surface $S$ is not necessarily convex with respect to any direction; there may be space curves embedded in $S$ and passing through $p$ which have principal normal vectors at $p$ directed to either side of $S - p$. This occurs for a saddle-shaped surface for example. There is thus no "natural" direction for a principal normal vector of the surface analogous to the case for a space curve. Not only may the normals of the embedded space curves vary in direction, but their curvatures at $p$ may likewise form a spectrum of values. Part of the differential geometry of surfaces consists of defining useful notions of curvature for a surface at a point; unlike space curves, a surface will generally not have a single number characterizing its curvature at a point. Again, a saddle-shaped surface exhibits the difficulty. Curvature of a surface $S$ at a point $p$ can be defined in terms of the curvatures of the embedded *section* space curves which are those planar space curves that lie in planes that contain the normal line of $S$ at $p$. An alternate equivalent approach is to define curvature in terms of the rates of change of the normal vector of $S$ at $p$ in various directions.

**Exercise 3.5:** Describe a way to determine if a surface $S$ is convex at a point $p$.

**Solution 3.5:** Consider every space curve $x_\theta$ embedded in $S$ that passes through the point $p$ (it suffices to consider only planar space curves.) Let $n_\theta$ denote the principal normal vector of $x_\theta$ at $p$. If the projection of $n_\theta$ on the surface normal vector $N_p$ lies on the same side of the tangent space hyperplane $T_p$ of $S$ at $p$ as all the other such projections, then $S$ is convex at $p$ in the direction $(n_\theta, N_p)N_p$.

## 3.3 Derivatives

Let us consider the function $F : \mathcal{R}^n \to \mathcal{R}^m$. The *derivative* of $F$ at $x \in \mathcal{R}^n$ is the linear transformation $F'(x)$ described by the $n \times m$ matrix

$$[F'(x)] := \begin{bmatrix} \partial_1 F_1(x) & \partial_1 F_2(x) & \dots & \partial_1 F_m(x) \\ \partial_2 F_1(x) & \partial_2 F_2(x) & \dots & \partial_2 F_m(x) \\ \cdot & \cdot & \dots & \cdot \\ \partial_n F_1(x) & \cdot & \dots & \partial_n F_m(x) \end{bmatrix},$$ where $\partial_i F_j$ denotes

the partial derivative function $\frac{\partial F_j}{\partial x_i}$. Note that $[F'(x)]$ col $j = (\nabla F_j(x))^{\mathrm{T}}$.

Note that, like $F$, $F'(x)$ maps $\mathcal{R}^n$ to $\mathcal{R}^m$. For $n = 1$ and $m = 3$, $F$ is a parametric space curve whose tangent vector $F'$ is $[\partial_1 F_1\ \partial_1 F_2\ \partial_1 F_3]$; the linear transformation $F'(x)$ maps values in $\mathcal{R}^1$ into the one-dimensional subspace parallel to the tangent line of $F$ at $x$. For $n = 2$ and $m = 3$, $F$ is a parametric surface whose tangent space at the $\mathcal{R}^2$-point $x$ corresponding to the $\mathcal{R}^3$-point $F(x)$ is $rowspace([F'(x)])$. In general, the *tangent space* of $F : \mathcal{R}^n \to \mathcal{R}^m$ at $x \in \mathcal{R}^n$ is defined as $T_{F(x)} := rowspace([F'(x)]) \subseteq \mathcal{R}^m$, and the *normal space* of $F$ at $x$ is the orthogonal complement subspace

$T^{\perp}_{F(x)} = nullspace([F'(x)]^{\mathrm{T}}) = \{y \in \mathcal{R}^m \mid y[F'(x)]^{\mathrm{T}} = 0\} \subseteq \mathcal{R}^m$.

Since $rowspace([F'(x)]) = range(F'(x)) \subseteq \mathcal{R}^m$ is the tangent space of $F$ at $x$, the matrix $[F'(x)]$ is the matrix of the linear transformation $F'(x)$ with respect to the canonical bases of $\mathcal{R}^n$ and $\mathcal{R}^m$ that maps $\mathcal{R}^n$ onto the tangent space of $F$ at $x$, i.e., $y[F'(x)] \in T_{F(x)} \subseteq \mathcal{R}^m$ for $y \in \mathcal{R}^n$. Note that we take vectors to be row vectors, so linear transformations are applied by means of matrix-multiplication on the right. The matrix $[F'(x)]$ representing the derivative of $F$ is called the *Jacobian matrix* of $F$ at $x$. Note that for $m = 1$, $[F'(x)] = (\nabla F(x))^{\mathrm{T}}$.

**Exercise 3.6:** Note that $dim(T_{F(x)}) = 1$ when $F : \mathcal{R}^1 \to \mathcal{R}^3$ is a regularly parameterized space curve, and $dim(T_{F(x)}) = 2$ when $F : \mathcal{R}^2 \to \mathcal{R}^3$ is a non-degenerate surface. What can you say about $dim(T_{F(x)})$ in general? When does $T_{F(x)} = \{0\}$? When does $T_{F(x)} = \mathcal{R}^m$?

**Exercise 3.7:** Let $F : \mathcal{R}^n \to \mathcal{R}^m$ be defined by $F(x) = xA$ where $A$ is an $n \times m$ constant real matrix. Show that $[F'(x)] = A$. What is $F''(x)$? Since $[F'(x)]$ is a matrix, why isn't $F''(x)$ described by an $n \times m$ zero matrix for *all* functions $F$?

**Solution 3.7:** For $F(x) = xA$, $F''(x)$ is represented by the $n \times m$ zero matrix $O_{n \times m}$, but in general $F''(x)$ is represented by a triply-subscripted array of the functions $\partial_i \partial_j F_k$. Although $F'(x)$ is a linear transformation, it is a *different* linear transformation for each distinct vector $x$, and the argument vector of the linear transformation $F'(x)$ is *not* $x$; rather it is a distinct $n$-vector, say $y$. As a function of $y$, the function defined by $y[F'(x)]$ can be differentiated with respect to $y$ to yield the matrix $[F'(x)]$, but this is *not* $F''(x)$.

**Exercise 3.8:** Let $F : \mathcal{R}^1 \to \mathcal{R}^m$ be defined by $F_j(x) = y_1 f_{1j}(x) + \ldots + y_n f_{nj}(x)$ for $1 \le j \le m$, where $f_{ij} : \mathcal{R} \to \mathcal{R}$ is a real-valued differentiable function of the single real argument $x$ for $1 \le i \le n$ and $1 \le j \le m$. Note that $F(x) = y[f_{ij}(x)]$ where $y = (y_1, \ldots, y_n)$ and $[f_{ij}(x)]$ denotes the $n \times m$ matrix whose $(i, j)$th element is $f_{ij}(x)$ . Show that $[F'(x)] = y[f'_{ij}(x)]$. What is $F''(x)$?

**Exercise 3.9:** Let $F : \mathcal{R}^n \to \mathcal{R}^m$ and $G : \mathcal{R}^k \to \mathcal{R}^n$ be given functions. Then the $G, F$ composition function $F \circ G : \mathcal{R}^k \to \mathcal{R}^m$ is defined by $(F \circ G)(x) = F(G(x))$. Show that the matrix $[(F \circ G)'(x)]$ equals the matrix product $[G'(x)][F'(G(x))]$.

**Exercise 3.10:** What is the line in $\mathcal{R}^3$ which lies in the tangent plane of the surface $S$ defined by the function $z : \mathcal{R}^2 \to \mathcal{R}$ at the point $q\&z(q) \in S \subset \mathcal{R}^3$ and which passes through the point $q\&z(q)$ in the direction of steepest ascent?

**Solution 3.10:** Let $p(q) = (q_1, q_2, z(q_1, q_2))$, so $S = \{p(q) \mid q \in Q \subseteq \mathcal{R}^2\}$. Then the line of steepest ascent in $\mathcal{R}^3$ is obtained by translating the image under the linear transformation $[p']$ of the subspace defined as the multiples of the vector $\nabla z$. Thus, we have $line[p(q), p(q)+ (\nabla z(q))[p'(q)]]$.

# Function and Space Curve Interpolation

The first general problem we wish to consider below is how to construct an interpolatory space curve which passes, in order, through given points $p_1, p_2, \ldots, p_n$ in $\mathcal{R}^2$ or $\mathcal{R}^3$, perhaps along given associated directions $m_1, m_2, \ldots, m_n$, with $|m_i| \neq 0$ for $i = 1, 2, \ldots, n$. Indeed, we could elaborate the interpretation of the direction vectors, $m_1, \ldots, m_n$, so that $m_i = 0$ would be taken to specify a sharp corner or *cusp* at $p_i$.

We may select one of the infinite number of admissible interpolatory curves by requiring that various additional constraints such as a fixed arc length or mimimal curvature or $k$-fold continuous differentiability be honored.

It is often desirable that the solution be independent of the choice of orthogonal coordinate system, so that the shape of the interpolatory curve is similar to that obtained after any sequence of translations, rotations, and uniform scale transformations of the data. Thus we may desire our interpolation process to be invariant under a euclidean rigid motion transformation.

Sometimes constraints on curvature must be honored in order to satisfy requirements such as the requirement of a convex interpolation function for convex data. Moreover, it may be desirable that certain curves like straight lines or circles be exactly recovered when points of such curves are interpolated.

Another property of interest is linear combinability. When direction vectors are not given, an interpolatory curve algorithm is linearly combinable if the sum of the curve $C_1(t)$ obtained for the points $p_1, \ldots, p_n$, and the curve $C_2(t)$ obtained for the points $q_1, \ldots, q_n$, is equal to the curve produced for the points $p_1+q_1, \ldots, p_n+q_n$, where the range of the parameter $t$ is the same for $C_1$ and $C_2$. If $C_1(t)$ and $C_2(t)$ are expressible as weighted averages of the points being interpolated, with the same weight expressions in both cases, then $C_1(t) + C_2(t)$ is also such a weighted average of the points $p_1+q_1, \ldots, p_n+q_n$, and $C_1(t)+C_2(t)$ interpolates these points.

**Exercise 4.1:** Devise reasonable conditions for an interpolatory curve algorithm to be linearly combinable when direction vectors $m_1, \ldots, m_n$ are given, as well as points $p_1, \ldots, p_n$.

**Exercise 4.2:** Show that if $C_1(t)$ and $C_2(t)$ both interpolate the points $p_1, \ldots, p_n$ with $0 \le t \le 1$ where $C_1(0) = C_2(0) = p_1$ and $C_1(1) = C_2(1) = p_n$, then $\alpha C_1(t) + (1-\alpha)C_2(t)$ also interpolates the points $p_1, \ldots, p_n$, where $\alpha \in \mathcal{R}$.

We will consider the problem of interpolating between two points with associated directions below. Curves that are solutions to this problem can be the basis for a piecewise solution to the $n$-point problem by joining such two-point interpolatory curves. Note that even in the two-point case, and even in two dimensions, the general interpolatory curve problem is not the same as the more specialized problem of finding a suitable single-valued function $y(x)$, whose graph passes through two given points with given slopes.

**Exercise 4.3:** When does the two-dimensional two-point problem with directions reduce to a single-valued function interpolation problem?

**Solution 4.3:** Place the point $p_1$ at the origin and place the point $p_2$ on the positive $x$-axis. The direction vectors $m_1$ and $m_2$ are now fixed by the requirement that this transformation be a rigid motion. If both the resulting $m_1$ vector and the resulting $m_2$ vector lie in the positive $x$ half-plane *or* if the resulting $m_1$ vector and the resulting $m_2$ vector both lie in the negative $x$ half-plane, then there exists a single-valued continuous function which interpolates the two transformed points with tangent vectors in the given transformed directions. (Remember that the vectors $m_1$ and $m_2$ are just points in $\mathcal{R}^2$.) The essential point here is that tangent vectors (i.e., directions) are a more flexible constraint mechanism than slope values on determining curve shape.

# 5
# 2D-Function Interpolation

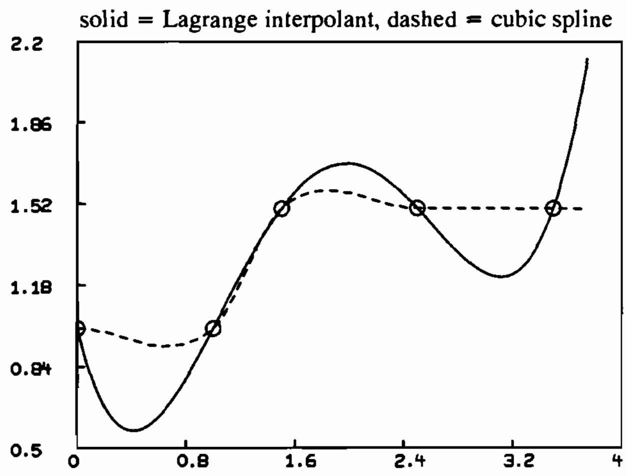
A *2D-function* is a function whose domain and range are both included in $\mathcal{R}$, and whose graph thus lies in $\mathcal{R}^2$. The initial interpolation problem we will consider is that of 2D-functional interpolation: given points of the graph of an otherwise unknown 2D-function $g$, we are interested in constructing another 2D-function which interpolates the given points and which serves as an estimate of the function $g$.

## 5.1 Lagrange Interpolating Polynomials

The Lagrange interpolating polynomial of degree $n$ or less for the $n+1$ $\mathcal{R}^2$-points $(x_0, y_0), \ldots, (x_n, y_n)$ with $x_0 < x_1 < \cdots < x_n$ is

$$\begin{aligned} y(x) &= q_0(x)y_0/v_0 + \cdots + q_n(x)y_n/v_n, \quad \text{where} \\ q_i(x) &:= \prod_{\substack{0\le j\le n \\ j\ne i}} (x - x_j) \quad \text{and} \\ v_i &:= q_i(x_i). \end{aligned}$$

Note that $v_i \neq 0$ as long as $x_0 < x_1 < \cdots < x_n$ holds. It is easy to check that the polynomial $y$ satisfies the $n+1$ equations $y(x_i) = y_i$ for $i = 0, 1, \ldots, n$. No other polynomial of degree $n$ or less exactly passes through the given points $(x_0, y_0), \ldots, (x_n, y_n)$.



**Exercise 5.1:** Let the Lagrange interpolating polynomial for the points $(x_0, y_0), \ldots, (x_n, y_n)$ with $x_0 < x_1 < \cdots < x_n$ be defined in terms of its coefficients $c_0, \ldots, c_n$ as $y(x) = c_0 + c_1 x + \cdots + c_n x^n$. Show that the coefficients $c_0, \ldots, c_n$ satisfy the linear system

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix},$$

and show that these $n + 1$ linear equations have a unique solution.

The figure above shows that the Lagrange interpolating polynomial for $n + 1$ points may 'oscillate' through the points, that is, it can markedly deviate from the piecewise-linear interpolatory curve for the $n + 1$ points. Thus it may be unsuitable for use in many cases. (Compare the Lagrange interpolating polynomial with the dashed-line cubic spline curve.)

**Exercise 5.2:** Compute the quadratic polynomial, $y(x)$ passing through three points $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$ with $x_1 < x_2 < x_3$.

**Solution 5.2:** $y(x) = y_1 + b(x - x_1) + c(x - x_1)^2$, where

$$\begin{aligned} b &= \frac{(x_3 - x_1)^2(y_2 - y_1) - (x_2 - x_1)^2(y_3 - y_1)}{(x_2 - x_1)(x_3 - x_1)(x_3 - x_2)}, \quad \text{and} \\ c &= \frac{(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)}{(x_2 - x_1)(x_3 - x_1)(x_3 - x_2)}. \end{aligned}$$

**Exercise 5.3:** Define $z_{ik}(x) \in \mathcal{R}^2$ for $k = 0, 1, \ldots, n$, and $i = 0, 1, \ldots, n - k$ as follows.

$$\begin{aligned} z_{i0}(x) &:= (x_i, y_i) \qquad \text{for } i = 0, 1, \ldots, n, \text{ and} \\ z_{ik}(x) &:= \frac{x_{i+k} - x}{x_{i+k} - x_i} z_{i,k-1}(x) + \frac{x - x_i}{x_{i+k} - x_i} z_{i+1,k-1}(x) \end{aligned}$$

for $k = 1, 2, \ldots, n$ and $i = 0, 1, \ldots, n - k$.

Show that $z_{0n}(x) = y(x)$, the Lagrange interpolating polynomial for $(x_0, y_0), \ldots, (x_n, y_n)$. This recursive construction of $y(x)$ is called Aitken's algorithm.

## 5.2 Whittaker's Interpolation Formula

Another exact method of interest for 2D-functional interpolation is Whittaker's interpolation formula. For an odd number of data points $(x_0, y_0)$, $\ldots$, $(x_n, y_n)$, with equally spaced abscissa values, $x_i = x_0 + ip/(n + 1)$, Whittaker's interpolation formula is: $y(x_0 + x) = \sum_{0 \le j \le n} y_j \sin(\pi(\text{px} - \text{j}))/(\pi(\text{px} - \text{j}))$, where $p = (x_n - x_0)(n + 1)/n$, and we take $\sin(0)/0 = 1$. The resulting interpolating function $y$ is the unique continuous band-limited periodic function of period $p$ which exactly satisfies $y(x_i) = y_i$ for $0 \le i \le n$, and which has no spectral components with frequencies outside the band $[-n/(2p), n/(2p)]$. Whittaker's interpolation formula is a consequence of computing the inverse discrete Fourier transform of a square wave of width $n/p$ and expressing its convolution with the discrete function being interpolated. (See [DM72].)

## 5.3 Cubic Splines for 2D-Function Interpolation

Given the points $(x_1, y_1), \ldots, (x_n, y_n)$, with $x_1 < x_2 < \ldots < x_n$ and associated slopes $m_1, m_2, \ldots, m_n$, we may use a piecewise cubic polynomial curve to interpolate the $n$ given points with the specified slopes. Such an interpolatory curve, which is made up of $n-1$ cubic polynomial segments joined together with common slopes at the data points, is called a *Hermite cubic spline*. (In mechanical drawing, a *spline* is a thin flexible strip that can be bent to interpolate a given set of points). When, as here, the slopes are explicitly specified initially, we have a *local* cubic spline, in contrast to a *global* cubic spline where slopes are not specified, but are instead computed from imposed continuity conditions. Note that the class of cubic polynomials includes the quadratic and linear polynomials as special cases. The main focus of this book will be centered on cubic splines and various generalizations thereof.

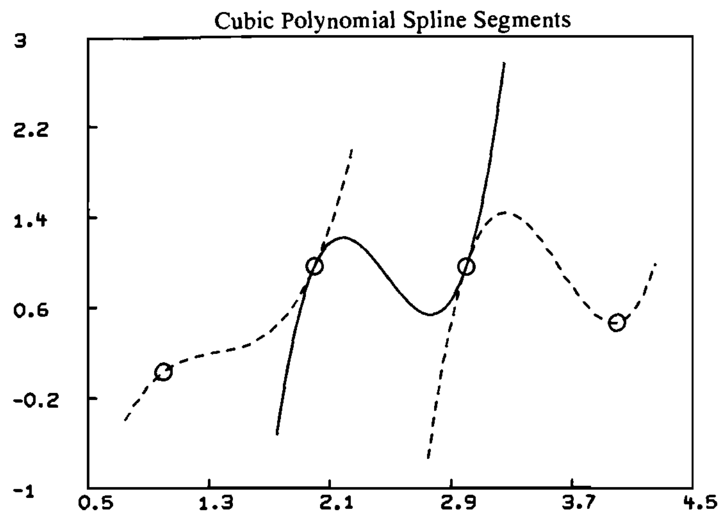
We are interested in piecewise-defined low degree curves because we have seen that a single high degree polynomial that does not oscillate unacceptably can be difficult to construct; putting many low degree polynomials together is a device to avoid the potential bad behavior of high degree polynomials. Cubic polynomial segment curves are a natural choice, since a cubic polynomial $u(x)$ has four coefficients that can (almost always) be uniquely determined to obtain a curve that connects two given points $(x_1, y_1)$ and $(x_2, y_2)$ with specified slopes $m_1$ and $m_2$ at these points. This is clear because we have four equations thus specified: $u(x_1) = y_1$, $u'(x_1) = m_1$, $u(x_2) = y_2$, and $u'(x_2) = m_2$.

**Exercise 5.4:** When does a cubic polynomial $u(x)$ that connects two given points $(x_1, y_1)$ and $(x_2, y_2)$ with specified slopes $m_1$ and $m_2$ at these points fail to exist? When such a cubic polynomial does exist, when is it unique?

The $i$th cubic polynomial $u_i(x)$ which connects $(x_i, y_i)$ with slope $m_i$ to $(x_{i+1}, y_{i+1})$ with slope $m_{i+1}$ as $x$ ranges from $x_i$ to $x_{i+1}$ is

$$\begin{aligned}
u_i(x) &= a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad \text{where} \\
a_i &= y_i, \\
b_i &= m_i, \\
c_i &= \frac{3(y_{i+1} - y_i)/(x_{i+1} - x_i) - 2m_i - m_{i+1}}{(x_{i+1} - x_i)}, \quad \text{and} \\
d_i &= \frac{m_i + m_{i+1} - 2(y_{i+1} - y_i)/(x_{i+1} - x_i)}{(x_{i+1} - x_i)^2}.
\end{aligned}$$

**Exercise 5.5:** Show that $u_{i-1}(x_i) = u_i(x_i) = y_i$ and $u'_{i-1}(x_i) = u'_i(x_i) = m_i$ for $2 \le i \le n-1$.



**Exercise 5.6:** Describe the graphs of cubic polynomial functions $f(t) = a + bt + ct^2 + dt^3$. *Hint*: consider $\lim_{t\uparrow\infty} f(t)$, $\lim_{t\downarrow-\infty} f(t)$, and those points where the slope of $f(t)$ is 0.

Note that $u_i(x)$ can be written as

$$\begin{aligned} u_i(x) &= \left(1 - \frac{3(x-x_i)^2}{(x_{i+1}-x_i)^2} + \frac{2(x-x_i)^3}{(x_{i+1}-x_i)^3}\right) y_i \\ &+ \left((x-x_i) - \frac{2(x-x_i)^2}{(x_{i+1}-x_i)} + \frac{(x-x_i)^3}{(x_{i+1}-x_i)^2}\right) m_i \\ &+ \left(\frac{3(x-x_i)^2}{(x_{i+1}-x_i)^2} - \frac{2(x-x_i)^3}{(x_{i+1}-x_i)^3}\right) y_{i+1} \\ &- \left(\frac{(x-x_i)^2}{(x_{i+1}-x_i)} - \frac{(x-x_i)^3}{(x_{i+1}-x_i)^2}\right) m_{i+1}. \end{aligned}$$

Thus $u_i(x)$ is a linear combination with the four coefficient values $y_i$, $m_i$, $y_{i+1}$ and $m_{i+1}$ multiplying four corresponding expressions that *do not* contain these values. This fact will be seen in the sequel to have far-reaching consequences; in particular, this means that $u_i(x)$ is an element in a vector space of functions that contains the four functions of $x$ appearing as factors in the sum above.

**Exercise 5.7:** Show that the Hermite cubic spline function piecewise-defined in terms of the cubic polynomials $u_1, \ldots, u_{n-1}$ satisfies the fourth order differential equation $y'''' = 0$, except possibly at the join points $(x_2, y_2), \ldots, (x_{n-1}, y_{n-1})$.

## 5.4 Estimating Slopes

Often we are given only the points $(x_1, y_1), \ldots, (x_n, y_n)$, and we must estimate the slopes $m_1, \ldots, m_n$. Define the line slopes, $s_j$, as $s_j := (y_{j+1} - y_j)/(x_{j+1} - x_j)$, with special choices for $s_{-1}, s_0, s_n$, and $s_{n+1}$, perhaps based on linear or quadratic extrapolation, as well as special choices for $x_0$, $y_0$, $x_{n+1}$, and $y_{n+1}$ so that our formulas for $m_1$ and $m_n$ will always make sense. Akima [Aki70] suggests that the interpolation slopes $m_1, \ldots, m_n$ can then be chosen based on the line slopes within five point groups as

$$m_i = \begin{cases} (s_{i-1} + s_i)/2 & \text{if } s_{i+1} = s_i \text{ and } s_{i-1} = s_{i-2}, \\ \dfrac{|s_{i+1} - s_i| s_{i-1} + |s_{i-1} - s_{i-2}| s_i}{|s_{i+1} - s_i| + |s_{i-1} - s_{i-2}|} & \text{otherwise.} \end{cases}$$

We can also try the following slope estimators for a Hermite cubic spline function in the $xy$-plane.

1. $m_i = (s_{i-1} + s_i)/2$.
2. $m_i = \tan((\text{atan}(\text{s}_{\text{i}-1}) + \text{atan}(\text{s}_\text{i}))/2)$.
3. $m_i = ((x_{i+1} - x_i)s_{i-1} + (x_i - x_{i-1})s_i)/(x_{i+1} - x_{i-1})$.
4. $m_i = (|p_{i+1} - p_i| s_{i-1} + |p_i - p_{i-1}| s_i)/|p_{i+1} - p_{i-1}|$, where $p_j = (x_j, y_j)$ for $0 \le j \le n + 1$.
5. $m_i = (y_{i+1} - y_{i-1})/(x_{i+1} - x_{i-1})$.

Another way to estimate the values $m_2, \ldots, m_{n-1}$ is to determine the unique quadratics $q_i(x)$ for $1 \le i \le n - 2$ where $q_i$ interpolates the points $(x_i, y_i)$, $(x_{i+1}, y_{i+1})$, and $(x_{i+2}, y_{i+2})$, and then choose $m_i = q'_{i-1}(x_i)$ for $2 \le i \le n - 1$. Also, we may choose $m_1 = q'_1(x_1)$ and $m_n = q'_{n-2}(x_n)$.

**Exercise 5.8:** Give the explicit formula for the quadratic interpolation function $q_{i-1}(x)$ in terms of $x_{i-1}, x_i, x_{i+1}, y_{i-1}, y_i$, and $y_{i+1}$.

**Exercise 5.9:** (L. Schumaker [Sch83]) Show that, given the points $(x_1, y_1), \ldots, (x_1, y_n)$, with $x_1 < x_2 < \cdots < x_n$, there are always choices for the slopes $m_1, \ldots, m_n$, such that there exist quadratic polynomials $q_1, \ldots, q_{n-1}$ that satisfy $q_i(x_i) = y_i$, $q_i(x_{i+1}) = y_{i+1}$, and $q_i'(x_i) = m_i$ and $q_i'(x_{i+1}) = m_{i+1}$ for $1 \le i \le n-1$. *Hint*: impose the conditions $m_i + m_{i+1} = 2s_i$.

## 5.5 Monotone 2D Cubic Spline Functions

Suppose we are given monotonic data points $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_1 < x_2 < \cdots < x_n$, so that $y_1 \le y_2 \le \cdots \le y_n$ or $y_1 \ge y_2 \ge \cdots \ge y_n$. We want to devise a slope estimation method which ensures that the corresponding Hermite cubic spline function is similarly monotonic on $[x_1, x_n]$. Let us consider the monotone increasing case where $x_1 < x_2 < \cdots < x_n$ and $y_1 \le y_2 \le \cdots \le y_n$. Let $s_i = (y_{i+1} - y_i)/(x_{i+1} - x_i) \ge 0$ for $1 \le i \le n-1$. Define $s_n := s_{n-1}$, and $s_{n+1} := s_n$. We have $s_i \ge 0$ for $1 \le i \le n+1$. The local Hermite cubic spline segment $u_i$ connecting $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ is monotonically increasing on $[x_i, x_{i+1}]$ exactly when $u_i'$ is non-negative on $[x_i, x_{i+1}]$, that is, when

$$m_i + 2[3s_i - 2m_i - m_{i+1}] \cdot \left(\frac{x - x_i}{x_{i+1} - x_i}\right)$$
$$+ 3[m_i + m_{i+1} - 2s_i] \cdot \left(\frac{x - x_i}{x_{i+1} - x_i}\right)^2 \ge 0$$

for $x_i \le x \le x_{i+1}$.

This inequality is satisfied when $m_i \ge 0, m_{i+1} \ge 0$, and $m_i + m_{i+1} \le 2s_i$. (When $m_i \ge 0$ and $m_{i+1} \ge 0$, choosing $m_i + m_{i+1} - 2s_i \le 0$ forces the cubic spline curve segment connecting $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ to be a non-decreasing segment of a cubic polynomial that grows toward $-\infty$ or at least remains bounded as its argument approaches $\infty$. Such a cubic polynomial has only one such section occurring between its two local extreme points. Imposing the constraints $m_i \ge 0$, $m_{i+1} \ge 0$, and $m_i + m_{i+1} \le 2s_i$ thus ensures non-decreasing monotonicity).

Thus the local cubic spline segments $u_1, \ldots, u_{n-1}$ are all individually monotonic when the slopes $m_i, \ldots, m_n$ are chosen so that $m_i \ge 0$, $\ldots, m_n \ge 0$, and $m_i + m_{i+1} \le 2s_i$ for $1 \le i \le n-1$. One way to choose $m_1, \ldots, m_n$ is to take $m_1 = s_1$ and then take $m_i = \min((s_i + s_{i-1})/2, 2s_i, 2s_{i-1} - m_{i-1})$ for $2 \le i \le n$. When this algorithm results in

$m_i \ge 0$ for $1 \le i \le n$, the resulting cubic spline function is monotonic on $[x_1, x_n]$. When we can enforce the additional conditions that $m_i \ge m_{i+1}$ when $s_i \ge s_{i+1}$ and $m_i \le m_{i+1}$ when $s_i \le s_{i+1}$, then the local convexity and concavity of the data will be preserved.

**Exercise 5.10:** Devise a slope estimation method that ensures that local minima and local maxima in the data are also local minima and local maxima in the corresponding Hermite cubic spline, so that there will be no undershoot or overshoot at such points.

**Solution 5.10:** Use slope 0 at local minima and local maxima.

**Exercise 5.11:** In the case where $m_i > 0$ and $m_{i+1} < 0$, what is $[\max_{x_i<x<x_{i+1}} u_i(x)] - \max(y_i, y_{i+1})$? That is, how much overshoot occurs in the segment $u_i$?

**Exercise 5.12:** How should we choose the slope values $m_i$ and $m_{i+1}$ so as to make the value $v_i := \max_{x_i \le x \le x_{i+1}} |u_i'(x)|$ as small as possible? How should we choose $m_1, \ldots, m_n$ so as to make $\max_{1 \le i \le n-1} v_i$ as small as possible?

**Exercise 5.13:** A function $f$ is *convex-upward* on $[x_1, x_n]$ if $f''$ increases monotonically on $[x_1, x_n]$. Suppose the data points $(x_1, y_1), \ldots, (x_n, y_n)$ have line-slopes that satisfy $s_1 \le s_2 \le \ldots \le s_{n-1}$. Show that the Hermite cubic spline that interpolates the points $(x_1, y_1), \ldots, (x_n, y_n)$ is convex-upward on $[x_1, x_n]$ whenever $m_1, \ldots, m_n$ satisfy the relations $2m_{i-1} + m_i \le 3s_{i-1}$ and $m_{i-1} + 2m_i \ge 3s_{i-1}$ for $2 \le i \le n$.

**Exercise 5.14:** (F. Fritsch and R. Carlson [FC80]) Define $\lambda_i := m_i/s_i$ and $\mu_i := m_{i+1}/s_i$. Show that the cubic spline segment function $u_i$ connecting $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$, with the respective slopes $m_i$ and $m_{i+1}$, is monotonic if and only if $(\lambda_i, \mu_i)$ lies in the *monotonicity region* $M$ defined by $\{(\lambda, \mu) \mid \lambda \ge 0, \mu \ge 0, 2\lambda + \mu \le 3\} \cup \{(\lambda, \mu) \mid \lambda \ge 0, \mu \ge 0, \lambda + 2\mu \le 3\} \cup \{(\lambda, \mu) \mid \lambda^2 + \lambda\mu + \mu^2 - 6(\lambda + \mu) + 9 \le 0\}$; $M$ is the union of two triangular regions and an ellipsoidal region. Make a graph of the region $M$ and show that $\{(\lambda, \mu) \mid \lambda \ge 0, \mu \ge 0, \lambda + \mu \le 2\} \subset M$ and that $\{(\lambda, \mu) \mid 0 \le \lambda \le 3, 0 \le \mu \le 3\} \subset M$.

**Solution 5.14:** Suppose $s_i \ne 0$ and $\lambda_i > 0$ and $\mu_i > 0$. Then $u_i$ is monotonic on $[x_i, x_{i+1}]$ exactly when $u_i'$ has no roots in $[x_i, x_{i+1}]$;

this is the case when $f(z) := \lambda_i + 2(3 - 2\lambda_i - \mu_i)z + 3(\lambda_i + \mu_i - 2)z^2$ has no roots in (0, 1). (Note that $u_i'((x_{i+1} - x_i)z + x_i) = f(z)s_i$.) Now, $f(0) = \lambda_i > 0$ and $f(1) = \mu_i > 0$. When $\lambda_i + \mu_i \le 2$, $f$ is convex-downward on [0, 1] and does not extend below the line $line[(0, \lambda_i), (1, \mu_i)]$. When $\lambda_i + \mu_i > 2$, $f$ has a local minimum at $z_0 := -\frac{1}{3}(3 - 2\lambda_i - \mu_i)/(\lambda_i + \mu_i - 2)$, and $f$ might have a root in (0, 1) if $z_0 \in (0, 1)$. But $z_0 \in (0, 1)$ only if $2\lambda_i + \mu_i > 3$ and $\lambda_i + 2\mu_i > 3$. When $z_0 \in (0, 1)$, $f$ has a root in (0, 1) only if $f(z_0) < 0$; but $f(z_0) = -\frac{1}{3}(\lambda_i^2 + \lambda_i\mu_i + \mu_i^2 - 6\lambda_i - 6\mu_i + 9)/(\lambda_i + \mu_i - 2)$, and thus $f(z_0) < 0$ exactly when $\lambda_i^2 + \lambda_i\mu_i + \mu_i^2 - 6\lambda_i - 6\mu_i + 9 > 0$. Note that the conditions $(\lambda_i, \mu_i) \in [0, 3] \times [0, 3]$ for $1 \le i < n$ are equivalent to the conditions $m_i < 3\min(s_{i-1}, s_i)$ for $1 \le i \le n$ with $s_0 := s_1$ and $s_n := s_{n-1}$.

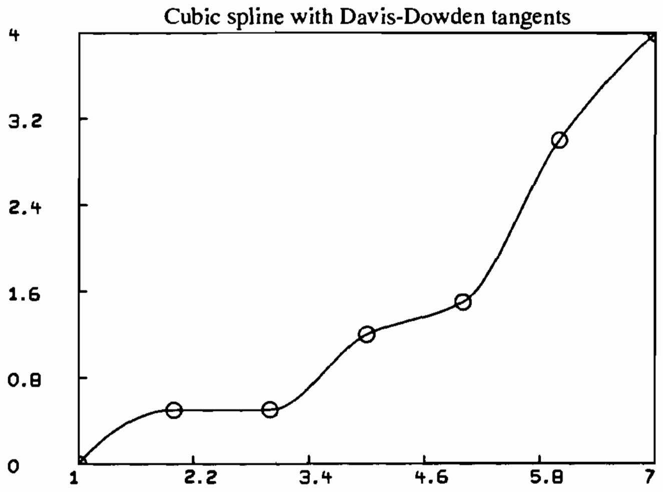
Davis and Dowden [DD87] proposed a slope estimation scheme suitable for most monotone data. Their scheme uses ratios of simple slope estimates. The Davis-Dowden slope estimation scheme defines $m_i = s_{i-1}(s_i/[(y_{i+1} - y_{i-1})/(x_{i+1} - x_{i-1})])$ for $1 < i < n$, with special choices for $m_1$ and $m_n$ such as $m_1 = s_1$ and $m_n = s_{n-1}$. When $y_1, y_2$, and $y_3$ are monotonic, then $m_1 = s_1^2/[(y_3 - y_1)/(x_3 - x_1)]$ is recommended, and similarly, $m_n = s_{n-1}^2/[(y_n - y_{n-2})/(x_n - x_{n-2})]$ is recommended when $y_{n-2}, y_{n-1}$, and $y_n$ are monotonic. Whenever $s_{i-1} \cdot s_i < 0$, the special choice $m_i = 0$ can be imposed.

**Exercise 5.15:** What happens when the Davis-Dowden scheme is used for choosing $m_i$ when $s_{i-1} \cdot s_i < 0$?

**Exercise 5.16:** Show that the cubic spline interpolation function for the points $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_1 < x_2 < \ldots < x_n$ and $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$ with slopes estimated via the Davis-Dowden scheme is monotonic on $[x_2, x_{n-1}]$ when $(x_{i+1} - x_{i-1})/(x_i - x_{i-1}) < 3$ and $(x_{i+2} - x_i)/(x_{i+2} - x_{i+1}) < 3$ for $2 \le i \le n - 2$.

**Exercise 5.17:** Propose a slope estimation scheme that produces a monotonic cubic spline interpolant for *any* given monotone data.

**Solution 5.17:** (F. Fritsch and J. Butland [FB84]) Define $s_0 := s_1$ and $s_n := s_{n-1}$ and define $s_i^{min} := \min(s_{i-1}, s_i)$ and $s_i^{max} := \max(s_{i-1}, s_i)$. Then choose $m_i = 3s_i^{min}s_i^{max}/(s_i^{max} + 2s_i^{min})$ for $1 \le i \le n$.



Hyman [Hym83] has proposed a scheme for enforcing monotonicity in a cubic spline interpolant for monotonic sequences of data points which works by "filtering" a given sequence of slope values $h_1, \ldots, h_n$ associated with the data points $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_1 < x_2 < \ldots < x_n$. Let the additional slope value $s_0$ be chosen arbitrarily to correspond to the slope at an invented point occurring prior to $(x_1, y_1)$, and let the additional slope value $s_n$ be chosen arbitrarily to correspond to the slope at the point $(x_n, y_n)$. Hyman chooses $m_i = \text{sign}(h_i) \cdot \min(|h_i|, 3\min(|s_{i-1}|, |s_i|))$ for $1 \leq i \leq n$ so that his scheme will produce a monotonic cubic spline interpolant for monotonic data provided that $\text{sign}(h_i) = \text{sign}(s_i)$ whenever $s_{i-1} \cdot s_i > 0$.

## 5.6 Error in 2D Cubic Spline Interpolation Functions

In general, a Hermite cubic spline approximation to a 2D-function $f$ based on given discrete points from the graph of $f$ (and associated known or estimated slopes) is a good approximation if $f$ doesn't "surprise" us in-between the given points by oscillating rapidly or becoming very large or very small. We cannot know, without additional information, that $f$ is well-behaved, but we can use the derivatives of $f$ as a measure of how much $f$ can rise or fall in an interval, and it is thus possible to state how good a

Hermite cubic spline approximation is with an error bound involving the derivatives of $f$. If slopes are correctly given, the error bounds we can obtain involve only the second and higher derivatives of $f$.

Let $f$ denote the function being approximated by the cubic polynomial segments $u_1, \ldots, u_{n-1}$, so that $f(x_i) = y_i$ for $1 \leq i \leq n$. If our slope choices are correct, so that $m_i = f'(x_i)$, then

$$\max_{x \in [x_i, x_{i+1}]} |f(x) - u_i(x)| \leq \frac{1}{384}(x_{i+1} - x_i)^4 \max_{x \in [x_i, x_{i+1}]} |f''''(x)|.$$

More generally, if the slope choices $m_i$ are not correct, then

$$\max_{x \in [x_i, x_{i+1}]} |f(x) - u_i(x)| \leq \tfrac{1}{384}(x_{i+1} - x_i)^4 \max_{x \in [x_i, x_{i+1}]}[|f''''(x)|] + |x_{i+1} - x_i| \max(|m_i - f'(x_i)|, |m_{i+1} - f'(x_{i+1})|).$$

See DeBoor [DeB78] for a discussion of these inequalities.

**Exercise 5.18:** Show that if $x_1, \ldots, x_n$ are equally spaced values with $x_{i+1} - x_i = (x_n - x_1)/(n-1)$, then $\max_{x \in [x_1, x_n]} |f(x) - u(x)| \leq kn^{-4}$ for some value $k$, where $u(x) = u_i(x)$ on $[x_i, x_{i+1}]$, and where $k$ depends upon $f$ and $x_1$ and $x_n$ and $m_1, \ldots, m_n$, but not upon $n$.

# 6 Λ-Spline Curves With Range Dimension *d*

All forms of splines we shall consider for 2D-functions and 3D-space curves will be Λ-splines as now introduced. Let $\Lambda$ be a class of vector-valued functions which map $\mathcal{R}$ to $\mathcal{R}^d$, such that no two functions in $\Lambda$ are identical on $[r_0, r_{n-1}]$. Given the real values $r_0 \le r_1 \le \cdots \le r_{n-1}$, the parametric function $s(t)$ defined on the interval $[r_0, r_{n-1}]$ is called a $\Lambda$-*spline of join order k with respect to* $r_0, r_1, \ldots, r_{n-1}$ if $s(t) = g_i(t - r_{i-1})$ for $r_{i-1} \le t \le r_i$ and $1 \le i \le n-1$, where each function $g_i$ is a function in $\Lambda$, and if $s$ is $C^k$ continuous at $r_1, \ldots, r_{n-2}$, i.e., $s$ is continuous at $r_1, \ldots, r_{n-2}$ and $s$ has at least $k$ successive derivatives that are continuous at $r_1, \ldots, r_{n-2}$. The $\Lambda$-spline function $s$ is also said to be $C^k$-*joined* at the points $s(r_1), \ldots, s(r_{n-2})$, called the *join points* of $s$.

Let $C^k[r_0, r_{n-1}]$ denote the class of functions which are continuous and have at least $k$ successive continuous derivatives on $[r_0, r_{n-1}]$. (Functions which are continuous and have at least $k$ successive continuous derivatives on the entire real line are often called $C^k$-*functions*). If $\Lambda \subseteq C^k[r_0, r_{n-1}]$ and $s$ is a $\Lambda$-spline of join order $k$ with respect to $r_0, \ldots, r_{n-1}$, then $s \in C^k[r_0, r_{n-1}]$, i.e., the $\Lambda$-spline function $s$ is continuous and has at least $k$ successive continuous derivatives everywhere in the interval $[r_0, r_{n-1}]$. The function $s$ is defined piecewise in terms of the segment functions $g_1, \ldots, g_{n-1}$. The $\Lambda$-spline $s(t)$ is of range dimension $d$ when the functions in $\Lambda$ are mappings from $\mathcal{R}$ to $\mathcal{R}^d$.

**Exercise 6.1:** Show that if $s : \mathcal{R} \to \mathcal{R}^d$ is a $C^k$ function, then so are all the component functions $s_1, \ldots, s_d$, and conversely.

The class of Λ-splines of join order $k$ and range dimension $d$ with respect to $r_0, \ldots, r_{n-1}$ is denoted by $\Lambda^*_{(k)}$, with $d$ and $r_0, \ldots, r_{n-1}$ understood. The particular parameter values $r_0, \ldots, r_{n-1}$ are often called *knot values*. Note that range dimension 1 Λ-splines are real-valued functions, range dimension 2 Λ-splines are plane curves, and range dimension 3 Λ-splines are space curves.

When we take Λ to be the class of cubic polynomial functions that map $\mathcal{R}$ to $\mathcal{R}$ and we define $r_j = x_{j+1}$ where $x_1, \ldots, x_n$ are the abscissa values of the points to be interpolated, then the 2D-function-interpolating Hermite cubic splines described above are the class $\Lambda^*_{(1)}$ of Λ-splines of join order 1 and range dimension 1 with respect to the knot values $r_0, r_1, \ldots, r_{n-1}$.

When Λ is a vector space of dimension $m$, $\Lambda^*_{(k)}$ is also a vector space, and when the knot values $r_0, r_1, \ldots, r_{n-1}$ are distinct values, $dim(\Lambda^*_{(k)}) = (n-2)(m-dk-d)+m$, since this corresponds to $n-1$ Λ-curve segments each with $m$ degrees of freedom subject to $d(k+1)$ continuity constraints at each of the $n-2$ interior knot values.

**Exercise 6.2:** What is the dimension of the vector space of join order 1, range dimension 1 Hermite cubic splines with respect to the distinct knot values $r_0, r_1, \ldots, r_{n-1}$?

**Solution 6.2:** $2n$.

# 7
# Cubic Polynomial Space Curve Splines

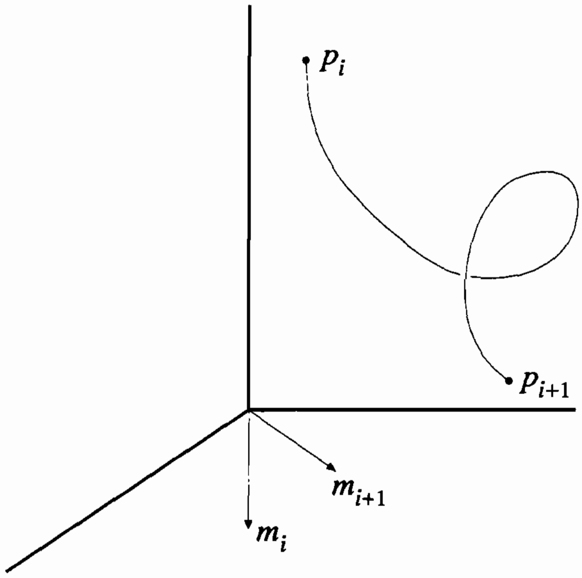
We may construct interpolating cubic splines for points in 3-space by using a parametric representation for such splines. Given the sequence of points $p_1, p_2, \ldots, p_n$ in 3-space, together with the associated 3-space tangent vectors $m_1, m_2, \ldots, m_n$, and given the positive real parameter limit values $t_1, t_2, \ldots, t_{n-1}$, we may interpolate between the points $p_i$ and $p_{i+1}$ with the Hermite cubic polynomial space curve $x_i$, defined parametrically as $x_i(t) = (x_{i1}(t), x_{i2}(t), x_{i3}(t))$ for $0 \le t \le t_i$ and $1 \le i \le n-1$, where

$$\begin{aligned} x_i(t) &= a_i + b_i t + c_i t^2 + d_i t^3 \text{ and} \\ x_i(0) &= p_i, \\ x_i(t_i) &= p_{i+1}, \\ x_i'(0) &= m_i, \text{ and} \\ x_i'(t_i) &= m_{i+1}. \end{aligned}$$

Thus the 3-tuple vector coefficients are:

$$\begin{aligned} a_i &= p_i, \\ b_i &= m_i, \\ c_i &= 3(p_{i+1} - p_i)/t_i^2 - (2m_i + m_{i+1})/t_i, \text{ and} \\ d_i &= 2(p_i - p_{i+1})/t_i^3 + (m_i + m_{i+1})/t_i^2. \end{aligned}$$

Note that each of the cubic polynomial component functions $x_{i1}$, $x_{i2}$, and $x_{i3}$ are Hermite cubic splines; for $j = 1, 2, 3$, the cubic polynomial



segment component function $x_{ij}(t)$ with $0 \le t \le t_i$ joins the $\mathcal{R}^2$-point $(0, p_{ij})$ with slope $m_{ij}$ to the $\mathcal{R}^2$-point $(t_i, p_{i+1,j})$ with slope $m_{i+1,j}$.

We may relax the conditions that the parameter limit values $t_1, \dots, t_{n-1}$ be positive when we desire. The space curve $x(t)$, made up by the piecewise joining of the segment curves $x_1, \dots, x_{n-1}$, is the *Hermite cubic spline space curve* that interpolates the points $p_1, \dots, p_n$ with tangent vectors $m_1, \dots, m_n$ and the argument-range parameter limit values $t_1, \dots, t_{n-1}$. The points $p_2 = x_1(t_1) = x_2(0), \dots, p_{n-1} = x_{n-2}(t_{n-2}) = x_{n-1}(0)$ are the interior *join points* of the Hermite cubic spline $x(t)$.

Let $H$ be the set of functions corresponding to cubic polynomial curves in 3-space with a scalar parameter $t$ and all possible coefficient vectors. $H$ is a vector space of dimension 12. The Hermite cubic polynomial interpolating space curve splines defined above are members of $H^*_{(1)}$, where $H^*_{(1)}$ is the class of join order 1, range dimension 3 $H$-splines with respect to the knot values $r_0, r_1, \dots, r_{n-1}$ where $r_0 = 0$ and $r_j = t_1 + \dots + t_j$ for $1 \le j < n$. These particular splines also depend upon the additional vector parameters, $p_1, \dots, p_n$ and $m_1, \dots, m_n$ chosen freely from $\mathcal{R}^3$. When $n > 1$ and the values $t_1, \dots, t_{n-1}$ are constrained to be positive, $H^*_{(1)}$ is a vector space of dimension $6n$.

For a cubic spline $x \in H^*_{(1)}$, made up of the piecewise joining of the cubic polynomial segments $x_i$, for $i = 1, 2, \dots, n-1$, we may define $r_j = \sum_{1 \le k \le j} t_k$, with $r_0 = 0$, and then define the truncated segment functions $X_i(t) =$ if $r_{i-1} \le t \le r_i$ then $x_i(t - r_{i-1})$ else 0. Then $x(t) =$

$\sum_{1 \le i < n} X_i(t)$ for $r_0 \le t \le r_{n-1}$; stated more directly, $x(t) = x_i(t - r_{i-1})$ for $t \in [r_{i-1}, r_i]$. Often we choose to define $x$ outside the interval $[r_0, r_{n-1}]$ by defining $x(t) = x_1(t)$ for $t < r_0$ and $x(t) = x_{n-1}(t)$ for $t > r_{n-1}$.

**Exercise 7.1:** Is a cubic polynomial space curve regularly parameterized?

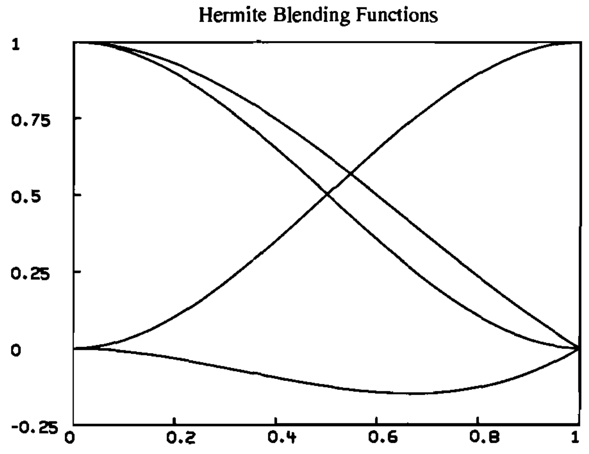
**Solution 7.1:** Not necessarily. Consider $((t-1)^3, (t-1)^2, 0)$ for $0 \le t \le 2$. However, at most three non-regular points may occur.

A cubic polynomial space curve segment has more defining parameters and thus more freedom than a 2D cubic polynomial function segment. The 2D function $u_i$ given above, which interpolates between $(\tilde{x}_i, \tilde{y}_i)$ with slope $\tilde{m}_i$ and $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ with slope $\tilde{m}_{i+1}$, is duplicated by the cubic polynomial space curve segment $x_i$ with $p_i = (\tilde{x}_i, \tilde{y}_i, 0)$, $p_{i+1} = (\tilde{x}_{i+1}, \tilde{y}_{i+1}, 0)$, $m_i = (1, \tilde{m}_i, 0)$, $m_{i+1} = (1, \tilde{m}_{i+1}, 0)$, and $t_i = \tilde{x}_{i+1} - \tilde{x}_i$. Note $x_i$ is in fact a planar curve. However there are infinitely many other cubic polynomial planar space curve segments that interpolate between $(\tilde{x}_i, \tilde{y}_i)$ with slope $\tilde{m}_i$ and $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ with slope $\tilde{m}_{i+1}$.

**Exercise 7.2:** Give several other cubic polynomial space curve segments that interpolate between $(\tilde{x}_i, \tilde{y}_i)$ with slope $\tilde{m}_i$ and $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ with slope $\tilde{m}_{i+1}$.

**Exercise 7.3:** An *equidegree* cubic polynomial space curve $h$ is one whose three scalar component polynomials have the same degree, i.e. $h$ satisfies $degree(h_1) = degree(h_2) = degree(h_3)$. Show that "in the large," an equidegree cubic polynomial space curve "looks like" a line or a ray. More precisely, show that for an equidegree cubic polynomial space curve $h(t) = a + bt + ct^2 + dt^3$ with $a, b, c, d \in \mathcal{R}^3$, there is an associated unit vector $u$ such that $h(t)$ asymptotically approaches the line $\{\alpha u \mid \alpha \in \mathcal{R}\}$ as $t \to \infty$ and as $t \to -\infty$. Also show that every unit vector $u$ has three corresponding equidegree cubic polynomial space curves which asymptotically approach the line defined by $u$. *Hint*: Consider the combinations of the cases $d = 0$, $c = 0$, and $b = 0$. What about non-equidegree cubic polynomial space curves?

Sometimes we may want to extrapolate forward or backwards outside the curve between $p_1$ and $p_n$. We may use the cubic polynomial space curve $x_1(t)$ with $t < 0$ to extrapolate points before $p_1$, and we may use the cubic polynomial space curve $x_{n-1}(t)$ with $t > t_{n-1}$ to extrapolate



points after $p_n$; however, these curves sometimes bend sharply "outside" the intervals they are selected for. Alternatively, we may use local linear or quadratic extrapolation beyond the end points. It is sometimes useful for fine control to add two additional points with associated tangent vectors at each end and use the new induced cubic polynomial segment curves to effect the extrapolation calculation which is now converted to an interpolation calculation.

The Hermite cubic polynomial segment curve, $x_i$, may be written as

$$\begin{aligned} x_i(t) &= y_i(t/t_i) \quad \text{where} \\ y_i(s) &= (1-3s^2+2s^3)p_i + (3s^2-2s^3)p_{i+1} \\ &\qquad + (s-2s^2+s^3)t_i m_i + (s^3-s^2)t_i m_{i+1}, \end{aligned}$$

so that $\{x_i(t) \mid t \text{ between } 0 \text{ and } t_i\} = \{y_i(s) \mid 0 \le s \le 1\}$. In addition to being an effective round-off error reducing formula for computing $x_i(t)$, this shows that $x_i(t)$ is a linear combination of the vectors $p_i$, $p_{i+1}$, $t_i m_i$, and $t_i m_{i+1}$; the functions of $s$ that appear as the scalar coefficients in this linear combination are called the *Hermite blending functions.*

**Exercise 7.4:** The orthogonal projection of a Hermite cubic spline space curve onto a plane produces a planar curve. Is this curve isometric to a plane Hermite cubic spline?

**Exercise 7.5:** Show that $x_i'(0) = m_i$ and $x_i'(t_i) = m_{i+1}$.

Note that a curve in the $xy$-plane can be embedded in the complex plane with the correspondence $(x(t), y(t)) \to x(t) + iy(t)$ and described with the complex-valued function of a real argument: $z(t) = x(t) + iy(t)$. Thus we may express the cubic spline segment curve in the plane which interpolates between the point $a$ with tangent vector $p$ and the point $b$ with tangent vector $q$ as a complex-valued function $z(t)$ of a real argument $t$ with $0 \le t \le 1$. Let us write $a$, $p$, $b$, and $q$ as complex numbers; then $z(t) = (1 - 3t^2 + 2t^3)a + (3t^2 - 2t^3)b + t(1-t)^2 p + t^2(t-1)q$. If we wish, we may write $a$, $b$, $p$, or $q$ in polar form to explicitly exhibit their magnitudes and polar angles. For example, if $p = re^{i\theta}$ and $q = se^{i\phi}$, then $z(t) = (1 - 3t^2 + 2t^3)a + (3t^2 - 2t^3)b + t(1-t)^2 re^{i\theta} + t^2(t-1)se^{i\phi}$.

**Exercise 7.6:** Show that $\{e^{i\psi}z(t) \mid 0 \le t \le 1\}$ is the rotation of the curve segment $\{z(t) \mid 0 \le t \le 1\}$ about the origin in the complex plane by the angle $\psi$.

## 7.1 Choosing the Segment Parameter Limits

The arc length of the interpolating curve segment $x_i$ between $p_i$ and $p_{i+1}$ is bounded below by the value $|p_{i+1} - p_i|$ and is directly related to the values $|m_i|$, $|m_{i+1}|$, and $|t_i|$, although the exact relationship is complicated. As a heuristic, we may choose each parameter limit value $t_i$, for $1 \le i \le n-1$, as the desired arc length of the corresponding interpolating curve segment between $p_i$ and $p_{i+1}$, or as an approximation thereof, such as $t_i = |p_{i+1} - p_i|$ or $t_i = 1$. Another choice is to take $t_i = |p_{i+1} - p_i|^{1/2}$; this is the so-called centripetal parametrization proposed by E. Lee [Lee89]. An alternate way to choose the values $t_1, \ldots, t_{n-1}$ is via the *maximum norm* distance; take $t_i = |\max(p_{i+1.1} - p_{i.1}, p_{i+1.2} - p_{i.2}, p_{i+1.3} - p_{i.3})|$. That is, $t_i$ is the largest of the distances between $p_{i+1}$ and $p_i$ in each coordinate direction. A variation on this idea is to use the sum of the distances between $p_{i+1}$ and $p_i$ in each coordinate direction as the value for $t_i$.

**Exercise 7.7:** If $t_i$ is used in the definition of the segment curve $x_i$ to approximate the arc length of $x_i$ between $p_i$ and $p_{i+1}$, isn't it necessary that $t_i \ge |p_{i+1} - p_i|$?

**Solution 7.7:** No, because the coefficients of the terms in $x_i$ take the magnitude of $t_i$ into account.

**Exercise 7.8:** What happens if $p_i = p_{i+1}$? Consider both the special cases $m_{i+1} = m_i$ and $m_{i+1} = -m_i$.

**Exercise 7.9:** What is the trace of $x_i(t)$ for $p_i = p_{i+1}, m_i = m_{i+1}$ and $t_i = 1$? Can a planar cubic spline segment form a figure-eight shaped curve?

**Exercise 7.10:** What is $x_i'(0)$ in the case when $m_i = 0$? *Hint*: look at $x_i''(0)$. What happens when $m_{i+1} = 0$?

**Exercise 7.11:** Graph the cubic spline for $p_1 = p_2 = \ldots = p_n = 0$ with $t_1 = \ldots = t_{n-1} = 1$, with various choices for the tangent vectors $m_1, \ldots, m_n$.

**Exercise 7.12:** Experiment with choices of $t_i$. What happens if one or more of the $t_i$-values are negative? What happens when $t_i$ is small? What happens if $t_i = 0$ when we use the cancellation convention that $0/0 = 1$? What if we take $0/0 = 0$?

**Solution 7.12:** When $t_i = 0, x_i(0) = p_{i+1}$, with the convention $0/0 = 1$. Thus a discontinuity is introduced when $t_i = 0$ and $p_{i+1} \neq p_i$. When $t_i = 0$ and $p_{i+1} = p_i$, but $m_i \neq m_{i+1}$, we generally obtain a cusp at $p_i$.

Another way to choose $t_1, \ldots, t_{n-1}$, proposed by Yamaguchi [Yam88], is to base the choice of $t_i$ on the lengths of suitable circular arcs that locally fit the data points. Let $a_i$ be the arc length of the circular arc between $p_{i-2}$ and $p_{i-1}$ on the circumference of the circle $C_i$ that is defined by the three points $p_{i-2}, p_{i-1}$, and $p_i$. Let $b_i$ be the arc length of the circular arc between $p_{i-1}$ and $p_i$ on the circle $C_i$. Then we can define $t_1 = a_3$, $t_{n-1} = b_n$, and $t_j = (a_{j+2} + b_{j+1})/2$ for $2 \leq j \leq n-2$. This same idea can be carried out using parabolic arcs, but the required arc length calculations are more difficult.

**Exercise 7.13:** Devise a method to compute $t_i$ so that $t_i$ is the true arc length of $x_i(t)$ between $p_i$ and $p_{i+1}$.

**Solution 7.13:** Initially estimate $t_i^{(0)} = |p_{i+1} - p_i|$. Then for $k = 1, 2, \ldots$, until convergence, compute $t_i^{(k+1)} = \int_{0 \leq t \leq t_i^{(k)}} |x_i'(t; t_i^{(k)})|$, where $x_i(t; t_i^{(k)})$ is the Hermite cubic polynomial space curve based on $p_i$, $p_{i+1}$, $m_i$, $m_{i+1}$, and $t_i^{(k)}$. Alternatively, use Newton's method to solve for $t_i$ in the equation $t_i = \int_{0 \leq t \leq t_i} |x_i'(t; t_i)|$.

**Exercise 7.14:** What is the relationship between $t_i$ and the arc length $g_i(t_i) = \int_{0 \leq t \leq t_i} |x_i'(t)|$.

**Solution 7.14:** The function $g_i$ is concave. When $t_i$ is small, $x_i$ is nearly a straight line, so in general, $g_i(t_i) > t_i$ when $t_i$ is small enough. As $t_i$ increases, a point is reached where $g_i(t_i) = t_i$, and beyond this point $g_i(t_i)$ grows slower than $t_i$. The cubic spline segment $x_i$ is never an arc length parameterization, even when $g_i(t_i) = t_i$, unless $x_i$ is a straight line.

**Exercise 7.15:** Show that, when $t_i = 1$ for $1 \le i < n$, then

$$\begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_i \\ p_{i+1} \\ m_i \\ m_{i+1} \end{bmatrix}.$$

Recall that the Hermite cubic polynomial segment curve, $x_i$, may be written in terms of Hermite blending functions as

$$\begin{aligned} x_i(t) &= y_i(t/t_i) \quad \text{where} \\ y_i(s) &= (1-3s^2+2s^3)p_i + (3s^2-2s^3)p_{i+1} \\ &\quad + (s-2s^2+s^3)t_i m_i + (s^3-s^2)t_i m_{i+1}, \end{aligned}$$

so that $\{x_i(t) \mid t \text{ between } 0 \text{ and } t_i\} = \{y_i(s) \mid 0 \le s \le 1\}$.

Let $C(p_i, p_{i+1}, m_i, m_{i+1}, t_i) = \{x_i(t) \mid t \text{ between } 0 \text{ and } t_i\}$, with $t_i$ positive or negative. Then, by referring to the Hermite blending function representation of $x_i$, we see that

$$C(p_i, p_{i+1}, m_i, m_{i+1}, t_i) = C(p_i, p_{i+1}, t_i m_i, t_i m_{i+1}, 1).$$

Thus choosing the parameter limit value $t_i$ amounts to selecting a scalar multiplier for the tangent vectors $m_i$ and $m_{i+1}$; therefore for the single cubic segment $x_i$, the limit value $t_i$ may be fixed, say at $t_i = 1$, with no loss of generality, since $m_i$ and $m_{i+1}$ may be compensatorially scaled. This also shows that choosing $t_i < 0$ amounts to reversing the directions of $m_i$ and $m_{i+1}$. Nevertheless, the freedom to choose $t_1, \ldots, t_{n-1}$ is valuable since it, in effect, allows the spline segments $x_i$ and $x_{i+1}$, which join at $p_{i+1}$ to have separately controllable arc lengths or flatness, while still having tangent geometric continuity at the join point $p_{i+1}$. (A curve $x$ is *tangent geometrically continuous* at a point $x(t)$ if the left entry and right exit tangent vectors $x'_-(t) = \lim_{\delta\downarrow 0}(x(t) - x(t-\delta))/\delta$ and $x'_+(t) = \lim_{\delta\downarrow 0}(x(t+\delta) - x(t))/\delta$ have the same direction, but not necessarily the same magnitude. The curve $x$ is *tangent algebraically continuous* at $x(t)$ if $x'_-(t) = x'_+(t)$). Because of the effect of the parameter limit value $t_i$ on the segment $x_i$, $t_i$ is sometimes called the *tension* of $x_i$. (Perhaps $t_i$ should be called the inverse tension or slackness of $x_i$, since the smaller $t_i$ becomes, the more taut the curve segment $x_i$ becomes).

**Exercise 7.16:** Directly verify that $C(p_{i+1}, p_i, -m_{i+1}, -m_i, t_i) = C(p_i, p_{i+1}, m_i, m_{i+1}, t_i)$.

**Exercise 7.17:** Show that $C(x_i(t), p_{i+1}, x_i'(t)/t_i, m_{i+1}, t_i) \subseteq C(p_i, p_{i+1}, m_i, m_{i+1}, t_i)$ for $0 < t < t_i$.

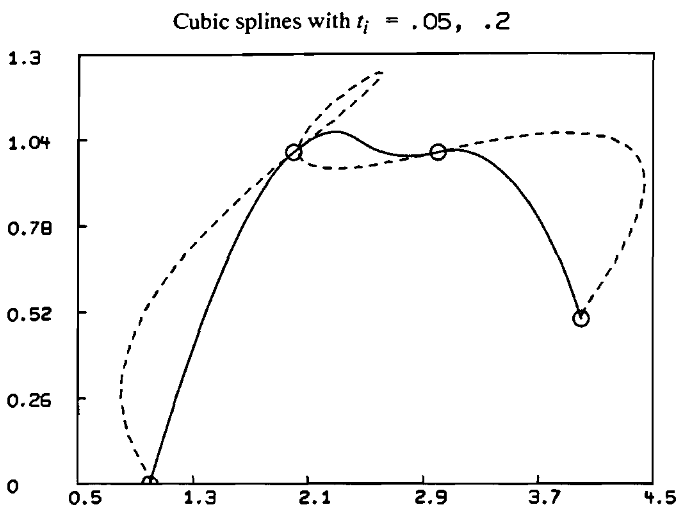
**Exercise 7.18:** Let $x_i(t)$ be the Hermite cubic polynomial segment curve for the points $p_i$ and $p_{i+1}$ with the tangent vectors $m_i$ and $m_{i+1}$ with $0 \le t \le t_i$. Let $\overline{x}_i(t)$ be the segment curve for the points $\overline{p}_i$ and $\overline{p}_{i+1}$ with the tangent vectors $\overline{m}_i$ and $\overline{m}_{i+1}$ and with the same parameter range $0 \le t \le t_i$. Show that $x_i(t) + \overline{x}_i(t)$ is the segment curve for the points $p_i + \overline{p}_i$ and $p_{i+1} + \overline{p}_{i+1}$ with the tangent vectors $m_i + \overline{m}_i$ and $m_{i+1} + \overline{m}_{i+1}$ with $0 \le t \le t_i$, and thus show that Hermite cubic spline interpolation is a linearly combinable interpolation scheme.

**Exercise 7.19:** Study the closed curve $g$ whose graph is given by the set $C(p_i, p_{i+1}, m_i, m_{i+1}, 1) \cup C(p_{i+1}, p_i, m_{i+1}, m_i, 1)$ in both the 2-space and 3-space cases. Characterize the convex hull of the graph of $g$.

**Exercise 7.20:** Show that if $x$ is a regularly parameterized tangent geometrically continuous curve, then at each point $x(t)$ with $0 \le t \le 1$, there exists a function $h(t)$ such that the reparametrization of $x$ : $\hat{x}(s) := x(h(s))$ is tangent algebraically continuous. *Hint*: $h$ may be chosen to yield an arc length parametrization.

**Exercise 7.21:** How can we generate a sequence of parameter values $0 = u_0, u_1, \ldots, u_{k-1}, u_k = t_i$ which result in approximately equally-spaced points $x_i(0), x_i(u_1), \ldots, x_i(u_{k-1}), x_i(t_i)$ on the space curve segment $x_i$?

**Exercise 7.22:** What is $x_i(t)$ in the case where $m_i = \alpha(p_{i+1} - p_i)$ and $m_{i+1} = \beta(p_{i+1} - p_i)$?



**Solution 7.22:** $x_i(t) = p_i + v(t)m$ where $m = p_{i+1} - p_i$ and $v(t) = \alpha t + (3 - (2\alpha + \beta)t_i)t^2/t_i^2 + ((\alpha + \beta)t_i - 2)t^3/t_i^3$. When $\alpha = \beta = 1$, $v(t) = t$.

**Exercise 7.23:** Under what conditions can $x_i'(t) = 0$ occur? Give examples of values for $p_i$, $p_{i+1}$, $m_i$, $m_{i+1}$, $t_i$ and $t$ for which $x_i'(t) = 0$.

## 7.2 Estimating Tangent Vectors

We may choose the tangent vectors $m_1, \ldots, m_n$ for a cubic spline space curve interpolating the points $p_1, \ldots, p_n$ using a local definition such as $m_i = (u_i + u_{i-1})/2$, where $u_j = (p_{j+1} - p_j)/|p_{j+1} - p_j|$ with special choices for $u_0$ and $u_n$. Each vector $u_j$, $j = 1, \ldots, n$, is the unit tangent vector parallel to the line segment between $p_j$ and $p_{j+1}$ directed from $p_j$ to $p_{j+1}$. Generally, we will want to scale the tangent vectors $m_1, \ldots, m_n$ to have what we deem to be suitable magnitudes, either explicitly or implicitly by our choices of the parameter limit values $t_1, \ldots, t_{n-1}$. We want to control the magnitudes of our tangent vectors in order to control the arc lengths of the individual spline segments, which should generally be longer as the distances between the interpolated points increase.

For the special case of a planar curve where $p_i = (x_i, y_i, 0)$, Akima [Aki70] suggests: $m_i = |p_{i+1} - p_i|(r_i, v_i, 0)/|(r_i, v_i, 0)|$, where $r_i =$

$h_i a_{i-1} + g_i a_i$, $v_i = h_i b_{i-1} + g_i b_i$, $h_i = |a_i b_{i+1} - a_{i+1} b_i|$, $g_i = |a_{i-2} b_{i-1} - a_{i-1} b_{i-2}|$, and $a_i = x_{i+1} - x_i$, $b_i = y_{i+1} - y_i$ with special choices of $x_{-1}$, $x_0$, $x_{n+1}$, $y_{-1}$, $y_0$, and $y_{n+1}$.

We can also try the following tangent estimators for a Hermite cubic space curve spline, where $u_j := (p_{j+1} - p_j)/|p_{j+1} - p_j|$. In all cases, special rules must be imposed to specify $m_1$ and $m_n$.

1. For plane curves in the $xy$-plane, we may take $m_i = (\cos\theta_i, \sin\theta_i)$, where $\theta_i = (atan2(p_{i,2} - p_{i-1,2}, p_{i,1} - p_{i-1,1}) + atan2(p_{i+1,2} - p_{i,2}, p_{i+1,1} - p_{i,1}))/2$. Recall that $atan2(y, x)$ is the angle by which the vector $|(x, y)|(1, 0)$ must be rotated counterclockwise to coincide with the vector $(x, y)$ in the $xy$-plane. Such angle-based tangent estimators can be extended to be used for space curves by separately mapping each three points $p_{i-1}$, $p_i$, $p_{i+1}$ to and from the $xy$-plane with a rigid motion.

2. $m_i = |p_{i+1} - p_i|(|p_{i+1} - p_i|u_{i-1} + |p_i - p_{i-1}|u_i)/|p_{i+1} - p_{i-1}|$.

3. $m_i = |p_{i+1} - p_i|u_{i-1} + |p_i - p_{i-1}|u_i$.

4. $m_i = |p_{i+1} - p_i|(p_{i+1} - p_{i-1})/|p_{i+1} - p_{i-1}|$.

5. $m_i = (p_{i+1} - p_{i-1})/2$. This is often called the *Catmull-Rom* or the *chordal* tangent estimator.

6. $m_i = (1 - c_i)u_{i-1} + c_i u_i$ where $c_i = |u_{i+1} - u_{i-2}|/(|u_{i-1} - u_{i-2}| + |u_{i+1} - u_i|)$ when $|u_{i+1} - u_i| + |u_{i-1} - u_{i-2}| > 0$ and $c_i = 1$ otherwise. Take $u_{-1} = u_1$ and $u_0 = u_1$, and take $u_{n+1} = u_n$ to define $m_i$ for $1 \le i \le n$. This is based on Akima's tangent estimator for planar curves, given above.

The options (1) and (6) given above specify unit length vectors, and will, in many situations, need to be scaled; some approaches for computing suitable scale values are discussed below.

**Exercise 7.24:** Show that, when $m_i$ and $m_{i+1}$ are defined by (3) or (5) above, then the spline segment $x_i(t)$ becomes $\alpha x_i(t)$ when $p_i$ and $p_{i+1}$ are replaced by $\alpha p_i$ and $\alpha p_{i+1}$.

**Exercise 7.25:** Consider the following scheme for computing an estimate for $m_1$. Let $u = (p_2 - p_1)/|p_2 - p_1|$, and let $v = (p_3 - p_1)/2$. Now define $m_1$ in terms of the euclidean inner product as $m_1 = 2(v, u)u - v$.

Explain the idea of this estimation formula, and state the similar formula for estimating $m_n$.

**Solution 7.25:** $m_n = 2(b,a)a - b$, where $a = (p_{n-1} - p_n)/|p_{n-1} - p_n|$ and $b = (p_{n-2} - p_n)/2$.

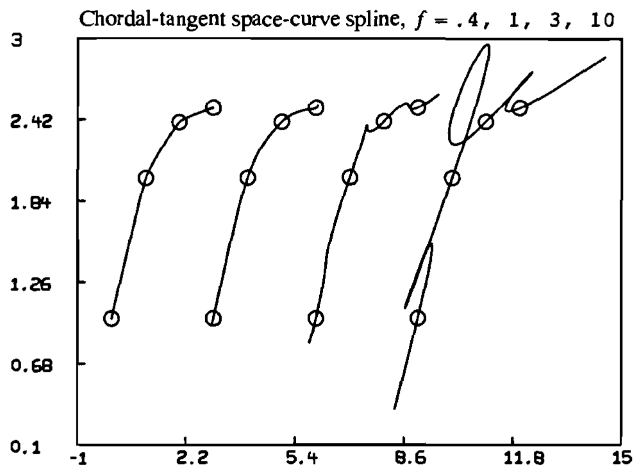
Note that any slope estimation scheme for 2D functional $(x, y)$ data can be employed in estimating tangent vectors for 3D $(x, y, z)$ data by using the 2D slope estimation method for the $x$, $y$, and $z$ component data separately with a common sequence of parameter limit values determined by any means desired taken as the (independent) argument values. Also, any 2D curve interpolation scheme can be applied to univariate functional data. In this latter case, however, it is not guaranteed that the resulting curve will be the graph of a function. In the case of functional data, it is often appropriate to transform the tangent vectors $m_1, \ldots, m_n$ by scaling $m_i$ by $1/m_{i,1}$, so that our tangent vectors all have 1 as their first component.

**Exercise 7.26:** What do you think of the following procedure to extend a 2D tangent estimation method to 3D data points $p_1, \ldots, p_n$? Take the mean of the three sequences of tangent vectors obtained by estimating the 2D tangent vectors in the $xy$-plane for the $(p_{i1}, p_{i2})$ data (promoted to 3-space), and the estimated 2D tangent vectors in the $xz$-plane for the $(p_{i1}, p_{i3})$ data (promoted to 3-space), and the estimated 2D tangent vectors in the $yz$-plane for the $(p_{i2}, p_{i3})$ data (promoted to 3-space).

**Exercise 7.27:** If $m_i = 0$, does the Hermite cubic spline curve $x$ have a well-defined tangent vector at the point $p_i$?

**Exercise 7.28:** Show that the Hermite cubic spline for $p_1, p_2, \ldots, p_n$ and $m_1 = m_2 = \ldots = m_n = 0$ is the piecewise linear interpolant connecting $p_1, p_2, \ldots, p_n$ in that order. What happens when an isolated tangent vector, $m_i$, is 0, with $m_{i-1} \neq 0$ and $m_{i+1} \neq 0$?

The shape of the Hermite cubic polynomial segment curve $x_i$ depends upon the parameter limit value $t_i$ and upon the directions and the magnitudes of the tangent vectors $m_i$ and $m_{i+1}$. Let $u_1, u_2, \ldots, u_n$ be $n$ unit vectors, and let us choose $m_i = f\gamma_i u_i$ for $1 \leq i \leq n$ as the tangent vectors used to determine the space curve polynomials $x_1, \ldots, x_{n-1}$. The scalar $f$ is a non-negative real number called the overall tension or flatness parameter, and $\gamma_1, \ldots, \gamma_n$ are additional non-negative local tension control



variables to be chosen as desired. The values $f\gamma_1, \ldots, f\gamma_n$ determine the magnitudes of the vectors $m_1, \ldots, m_n$, and the unit vectors $u_1, \ldots, u_n$ determine the directions of $m_1, \ldots, m_n$. A general Hermite cubic space curve spline is determined by $p_1, \ldots, p_n$, $u_1, \ldots, u_n$, $\gamma_1, \ldots, \gamma_n$, $t_1, \ldots, t_{n-1}$, and $f$. For a fixed value of $f$, the greater $\gamma_i$ is, the more the curve $x_i$ hugs the ray, $ray[p_i, p_i + u_i]$, and in the same way, the greater $\gamma_{i+1}$ is, the more $x_i$ hugs the ray, $ray[p_{i+1}, p_{i+1} + u_{i+1}]$. The arc length of $x_i$ between $p_i$ and $p_{i+1}$ increases as $|\gamma_i \gamma_{i+1}|$ increases. When $\gamma_i$ and $\gamma_{i+1}$ are both large enough, the curve $x_i$ loops for suitable directions $u_i$ and $u_{i+1}$. If $u_i$, $u_{i+1}$, $p_i$ and $p_{i+1}$ are coplanar, then $x_i$ is planar, and any loop entails the self-intersection of $x_i$. It is also possible for $x_i$ to have a solitary cusp as a kind of degenerate loop. A rough heuristic for choosing the magnitudes of $m_1, \ldots, m_n$ is to aim for $|m_i| + |m_{i+1}| \approx 2|p_{i+1} - p_i|$ with $t_i = 1$.

The choice of the values $t_1, t_2, \ldots, t_{n-1}$ and $\gamma_1, \ldots, \gamma_n$, and $f$ can drastically affect the spline curve that is generated for interpolating given points $p_1, \ldots, p_n$ with the unit tangent vectors $u_1, \ldots, u_n$. If some of these values are too large, you will see baroque shapes, while if some are too small, the tension of the corresponding parts of the resulting spline will be too severe. Also, the "extrapolated" parts of the curve can behave in unexpected ways. Consider the above spline curves using the same tangent estimation method, and varying only the value of the flatness scale factor $f$.

**Exercise 7.29:** Determine a way to predict when a planar Hermite cubic spline segment will form a loop.

**Exercise 7.30:** How can we introduce tension multipliers for 2D functional interpolation using degree 3 Hermite polynomials?

**Solution 7.30:** Use 2D space-curve interpolation with tension multipliers to obtain the parametric functions $x(t)$, $y(t)$. The tension multipliers must be chosen so that a function graph is obtained; there can be no loops or retrogression. Then, given an $x$-value $x_0$, find the corresponding $y$-value by first solving $x(t_0) = x_0$ for the value $t_0$, and then use this value $t_0$ to compute $y(t_0)$. This entails computing a root of a cubic polynomial.

**Exercise 7.31:** Show that if $x_i$ is the Hermite cubic polynomial segment curve defined by $p_i$, $p_{i+1}$, $m_i$, $m_{i+1}$, and $t_i$, then $\alpha x_i + q$ is the Hermite cubic polynomial segment curve defined by $\alpha p_i + q, \alpha p_{i+1} + q$, $\alpha m_i, \alpha m_{i+1}$, and $t_i$.

**Exercise 7.32:** (J. Chou [CP92]) Given the cubic polynomial space curve $x(t) = (1-3t^2+2t^3)p_1+(3t^2-2t^3)p_2+(t-2t^2+t^3)\alpha u_1+(t^3-t^2)\beta u_2$ with $|u_1| = 1, |u_2| = 1, x(0) = p_1, x'(0) = \alpha u_1, x(1) = p_2$, and $x'(1) = \beta u_2$, and given a point $q \in \mathcal{R}^3$, devise a way to choose the scalars $v$, $\alpha$ and $\beta$ such that $x(v) = q$ and $0 \le v \le 1$ when this is possible. Also state the conditions for which no such values $v$, $\alpha$ and $\beta$ exist.

**Exercise 7.33:** Let $Q$ be a $3 \times 3$ rotation matrix. Show that if $x_i$ is the Hermite cubic polynomial segment curve for $p_i$, $p_{i+1}$, $m_i$, $m_{i+1}$, and $t_i$, then $x_i Q$ is the Hermite cubic polynomial segment curve for $p_i Q$, $p_{i+1} Q, m_i Q, m_{i+1} Q$, and $t_i$.

**Exercise 7.34:** Compute the unit normal vector $v(t)$ for the Hermite cubic polynomial segment curve $x_i$ at the point $x_i(t)$.

**Exercise 7.35:** Suppose we are given two points $p_1$ and $p_2$ to be connected by a cubic spline segment curve $x(t)$ for $0 \le t \le 1$. Instead of having the tangent vectors $m_1$ and $m_2$, we are given the normal vectors $n_1$ and $n_2$ such that, when $x$ is correctly determined, $(n_1, x'(0)) = 0$ and $(n_2, x'(1)) = 0$. Devise a satisfying way to estimate $m_1$ and $m_2$ so that the cubic spline segment curve based on $p_1$, $p_2$, $m_1$, and $m_2$ has $(n_1, m_1) = (n_2, m_2) = 0$.

**Solution 7.35:** (G. Nielson [Nie74]) Determine $m_1$ by taking $m_1 + p_1 =$ the intersection of the line $line(n_1+p_1, p_2)$ with the tangent plane $p_1+\{p \mid (n_1, p) = 0\}$. Similarly, determine $m_2$ by taking $m_2+p_2 =$ the intersection of $line(n_2 + p_2, p_1)$ with the plane $p_2 + \{p \mid (n_2, p) = 0\}$.

**Exercise 7.36:** Given the cubic spline segment $x_i$, explain how to compute the smallest axis-aligned box $B$ containing the curve trace $C(p_i, p_{i+1}, m_i, m_{i+1}, t_i) = \{x_i(t) \mid 0 \le t \le t_i\}$. Also explain how to compute the smallest sphere $S$ containing $\{x_i(t) \mid 0 \le t \le t_i\}$.

**Exercise 7.37:** Compute the curvature function, $K(t)$, and the torsion function $T(t)$ of the Hermite cubic polynomial space curve segment $x_i(t)$, excluding any non-regular points.

**Exercise 7.38:** Let $t_1, \ldots, t_{n-1}$ be non-negative real numbers. Define $r_i := t_1+t_2+\cdots+t_i$ for $0 \le i \le n-1$, and show that if $p_i = z(r_{i-1})$ and $m_i = z'(r_{i-1})$ for $1 \le i \le n$, where $z(t) = a + bt + ct^2 + dt^3$, then the Hermite cubic spline interpolation function, $x$, based on $t_1, t_2, \ldots, t_{n-1}$, $p_1, p_2, \ldots, p_n$ and $m_1, \ldots m_n$ is identically $z$.

## 7.3 Bézier Polynomials

Cubic spline segments are often introduced via the so-called Bézier form [B́74]. The *Bézier* cubic polynomial space curve for the *control points* $p_1$, $p_2$, $p_3$, and $p_4$ is just the Hermite cubic interpolating polynomial space curve for the points $p_1$ and $p_4$ with the corresponding tangent vectors $3(p_2 - p_1)$ and $3(p_4 - p_3)$ whose parameter $t$ runs from 0 to 1 to generate the curve between $p_1$ and $p_4$. Thus the Bézier cubic for the control points $p_i$, $p_i + m_i/3$, $p_{i+1} - m_{i+1}/3$, and $p_{i+1}$ is the same as the Hermite cubic polynomial $x_i$ for the points $p_i$ and $p_{i+1}$ with the associated tangent vectors $m_i$ and $m_{i+1}$ whose parameter $t$ runs from 0 to 1.

**Exercise 7.39:** Show that the osculating plane of the Hermite cubic polynomial space curve segment $x_i(t)$, with $0 \le t \le 1$, at $x_i(0)$ is the flat $p_i + plane(0, m_i/3, p_{i+1} - p_i - m_{i+1}/3)$. Can $m_i/3$ be replaced by $m_i$? Can $m_{i+1}/3$ be replaced by $m_{i+1}$? What is the osculating plane of $x_i$ at $x_i(1)$?

The Bézier cubic polynomial form can be written using the Bernstein basis polynomials as blending functions, where the $i$th degree-$k$ Bernstein basis polynomial denoted by $N_{ki}$ is $N_{ki}(t) := \binom{k}{i} t^i (1-t)^{k-i}$. We have

$$x_i(t) = p_i N_{30}(t) + (p_i + m_i/3) N_{31}(t) + (p_{i+1} - m_{i+1}/3) N_{32}(t) + p_{i+1} N_{33}(t).$$

**Exercise 7.40:** What is the relationship between Bernstein polynomials and the binomial theorem?

**Exercise 7.41:** Let $b_0^0 = p_i$, $b_1^0 = p_i + m_i/3$, $b_2^0 = p_{i+1} - m_{i+i}/3$, $b_3^0 = p_{i+1}$. Define $b_i^k(t) := (1-t)b_i^{k-1}(t) + tb_{i+1}^{k-1}(t)$ for $k = 1, 2, 3$ and $i = 0, 1, \ldots, 3-k$.

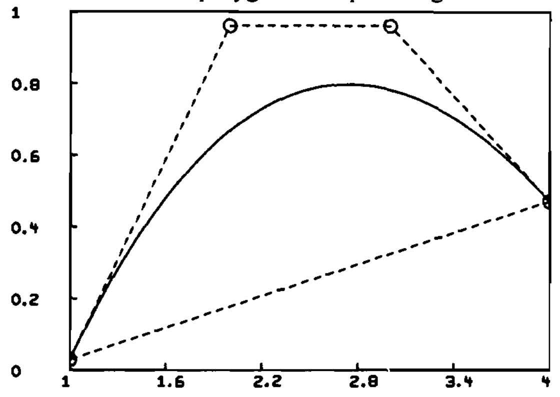
Show that $b_0^3(t) = x_i(t)$, where $x_i$ is the Hermite cubic polynomial for the points $p_i$ and $p_{i+1}$ with the associated tangent vectors $m_i$ and $m_{i+1}$ whose parameter $t$ ranges from 0 to 1. This construction of $x_i(t)$ by repeated convex combination linear interpolation is called de Casteljau's algorithm. Computing $x_i(t)$ with de Casteljau's algorithm is numerically more stable (introduces less round-off error) than computing $x_i(t)$ directly (with Horner's rule). In practice this difference is rarely troublesome.

**Exercise 7.42:** Show that $\{x_i(t) \mid 0 \le t \le 1\}$ is contained in the volume in $\mathcal{R}^3$ whose boundary is $convexhull(\{p_i, p_i + m_i/3, p_{i+1}, p_{i+1} - m_{i+1}/3\})$. The polytope (or polygon when $x_i$ is a plane curve) $convexhull(\{p_i, p_i + m_i/3, p_{i+1}, p_{i+1} - m_{i+1}/3\})$ is called the *Bézier polytope* (or the *Bézier polygon* when $x_i$ lies in $\mathcal{R}^2$) of the segment curve $x_i$. *Hint*: use knowledge of $N_{30}$, $N_{31}$, $N_{32}$, and $N_{33}$ to show that $x_i(t)$ is a convex combination of $p_i$, $p_i + m_i/3$, $p_{i+1}$, and $p_{i+1} - m_{i+1}/3$.

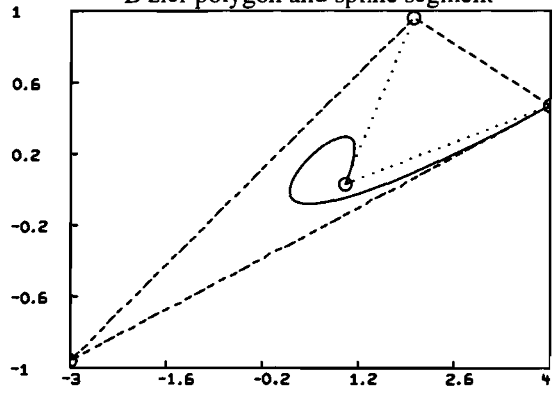
Note that the Bézier cubic space curve for the control points $p_1$, $p_2$, $p_3$, and $p_4$ connects $p_1$ and $p_4$ while generally only passing *close* to the points $p_2$ and $p_3$; thus the Bézier cubic is not an interpolating curve. Rather it may be considered a *smoothing* curve. The Bézier degree $n$ smoothing polynomial $b_0^n(t)$ for the points $p_1, \ldots, p_{n+1}$ is

$$b_0^n(t) := \sum_{i=0}^{n} p_{i+1} \binom{n}{i} t^i (1-t)^{n-i} \quad \text{for} \quad 0 \le t \le 1.$$

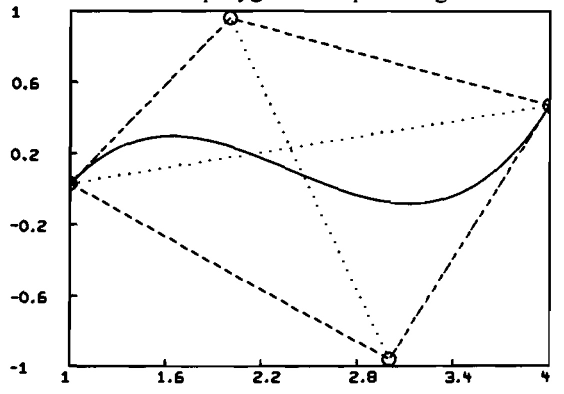
As $t$ ranges from 0 to 1, $b_0^n(t)$ joins the points $p_1$ and $p_{n+1}$ and passes "near" the points $p_2, \ldots, p_n$. Indeed, if $p_{i+1} = (\frac{i}{n}, f(\frac{i}{n}))$ for some rectifiable continuous function $f$, then $b_0^n(t) \to f(t)$ as $n \to \infty$ for $t \in [0, 1]$, B zier polygon and spline segment



B zier polygon and spline segment



B zier polygon and spline segment



and the function $b_0^n$ is a kind of "kernel estimator" for the function $f$, where $\binom{n}{i-1}t^{i-1}(1-t)^{n-i}$ for $i = 1, \ldots, n+1$ are the kernel functions with the argument $t$ and the parameter $n$.

# 8

# Double Tangent Cubic Splines

We may construct a more general class of cubic space curve splines called *double tangent splines* by introducing *two* tangent vectors at each point $p_i$. Let $m_1, \ldots, m_n$ be the *exit* tangent vectors at $p_1, \ldots, p_n$, and let $l_1, \ldots, l_n$ be the *entry* tangent vectors at $p_1, \ldots, p_n$. We intend that our double tangent spline will asymptotically *enter* the point $p_i$ along the line parallel to the vector $l_i$ and *exit* from the point $p_i$ along the line parallel to the vector $m_i$. Now we define the double tangent cubic spline space curve segment $x_i$ which defines the spline between $p_i$ and $p_{i+1}$ as follows:

$$\begin{aligned} x_i(t) &= a_i + b_i t + c_i t^2 + d_i t^3, \text{ where} \\ a_i &= p_i, \\ b_i &= m_i, \\ c_i &= 3(p_{i+1} - p_i)/t_i^2 - (2m_i + l_{i+1})/t_i, \\ d_i &= 2(p_i - p_{i+1})/t_i^3 + (m_i + l_{i+1})/t_i^2. \end{aligned}$$

The curve $x_i$ is the local Hermite cubic polynomial curve segment such that $x_i(0) = p_i$, $x_i(t_i) = p_{i+1}$, $x_i'(0) = m_i$, and $x_i'(t_i) = l_{i+1}$. Then the associated double tangent cubic spline is defined as $x(t) = x_i(t - r_{i-1})$ for $t \in [r_{i-1}, r_i)$ with $r_0 := 0$ and $r_i := t_1 + \ldots + t_i$.

If the vectors $l_i$ and $m_i$ are not identically directed, $x_{i-1}$ joins $x_i$ at $p_i$ with a cusp at $p_i$. Even when $l_i$ is a multiple of $m_i$, $x_{i-1}$ does not join $x_i$

at $p_i$ with an algebraic continuous tangent vector unless $l_i = m_i$, thus we may have tangent vector geometric continuity, i.e., directional continuity, without having tangent vector algebraic continuity. When $l_i = m_i$ for $1 < i < n$, the segments $x_1, \ldots, x_{n-1}$ form a continuously differentiable Hermite cubic spline curve (often called a $C^1$ cubic spline curve). Note $l_1$ and $m_n$ are unused in defining the double tangent spline on $[r_0, r_{n-1})$.

Thus it is possible to introduce a discontinuity in the derivative of a double tangent cubic spline interpolating space curve at any desired data points by suitably choosing the two sequences of tangent vectors: $m_1, \ldots, m_n$, and $l_1, \ldots, l_n$. The vectors $l_i$ and $m_i$ may be chosen to be unequal in direction, or length, or both, and different effects can be achieved in each case.

Reducing or increasing the magnitude of the tangent vector $m_i$ affects both the segment curve $x_{i-1}$ and the segment curve $x_i$ in a single tangent cubic spline, while changing the parameter limit value $t_i$ affects how both $m_i$ and $m_{i+1}$ govern the shape of the segment curve $x_i$. Double tangent splines are often useful because a double tangent spline allows us to overcome these coupling constraints and control the tension and shape of the individual segment curves independently. In particular, we can specify a spline curve which enters the point $p_i$ along the line $\{p_i + \alpha l_i \mid \alpha \in \mathcal{R}\}$ and exits along the line $\{p_i + \beta m_i \mid \beta \in \mathcal{R}\}$, but which "hugs" these lines to differing degrees depending upon the magnitudes of the vectors $l_i$ and $m_i$.

**Exercise 8.1:** How should the double tangent spline $x$ be extended so as to be defined on $(-\infty, r_0)$ and on $[r_{n-1}, \infty)$?

## 8.1 Kochanek-Bartels Tangents

Kochanek and Bartels [KB84] have proposed that the exit and entry tangent vectors $m_i$ and $l_i$ at the point $p_i$ be chosen as:

$$\begin{aligned} m_i &= (1-\alpha_i)(1+\beta_i)(1-\gamma_i)(p_i - p_{i-1})/2 \\ &\qquad + (1-\alpha_i)(1-\beta_i)(1+\gamma_i)(p_{i+1} - p_i)/2, \text{ and} \\ l_i &= (1-\alpha_i)(1+\beta_i)(1+\gamma_i)(p_i - p_{i-1})/2 \\ &\qquad + (1-\alpha_i)(1-\beta_i)(1-\gamma_i)(p_{i+1} - p_i)/2 \end{aligned}$$

for $2 \le i \le n-1$, with special choices for $m_1$ and $l_n$. The scalar values $\alpha_i$, $\beta_i$, and $\gamma_i$ may be chosen separately for each point $p_i$. The value $\alpha_i$

affects the *tension* of the segment curves $x_i$ and $x_{i+1}$, near $p_i$; the value of $\beta_i$ affects the *bias* near $p_i$ which is the amount the curve segments $x_i$ and $x_{i+1}$ shift asymmetrically near $p_i$; and the value of $\gamma_i$ affects the tangent vector *continuity* of the segment curves $x_i$ and $x_{i+1}$ joining at $p_i$. Tension near $p_i$ is increased when $\alpha_i$ approaches 1 and decreased as $\alpha_i$ decreases toward $-1$; $\alpha_i = 0$ corresponds to neutral tension. Bias near $p_i$ is neutral when $\beta_i = 0$, shifted to the entry side of $p_i$ when $\beta_i$ approaches $-1$ and shifted to the exit side when $\beta_i$ approaches 1. Finally the spline curve has an increasingly-pronounced sharp corner (i.e., a discontinous tangent) toward one side at $p_i$ when $\gamma_i$ approaches $-1$ and toward the other side as $\gamma_i$ approaches 1; $\gamma_i = 0$ corresponds to neutral tangent continuity.

**Exercise 8.2:** Experiment with various choices of $m_1, \ldots, m_{n-1}$ and $l_2, \ldots, l_n$. *Hint*: it suffices to consider just two adjacent spline segments with a common join point.

## 8.2 Fletcher-McAllister Tangent Magnitudes

Fletcher and McAllister [FM86] have suggested a useful heuristic for choosing the magnitudes of the exit tangents $m_1, \ldots, m_n$ and the entry tangents $l_1, \ldots, l_n$ for planar geometrically continuous double tangent cubic splines which we generalize for $p_1, \ldots, p_n \in \mathcal{R}^3$, as follows. Let $u_1, \ldots, u_n$ be any desired choice of unit vectors which we take to specify tangent *directions* at $p_1, \ldots, p_n$. For example, we may choose $u_i = (p_{i+1} - p_{i-1})/|p_{i+1} - p_{i-1}|$ for $1 < i < n$ with $u_1 := (p_2 - p_1)/|p_2 - p_1|$ and $u_n := (p_n - p_{n-1})/|p_n - p_{n-1}|$. Let $f$ denote a uniform scale factor used to control the overall tension imposed on our double tangent cubic space curve spline.

Now for $i = 1, \ldots, n-1$, we may determine the entry and exit tangent vectors $m_i$ and $l_{i+1}$ associated with the segment curve $x_i$ by applying the following procedure.

1. $a \leftarrow u_i$; $\quad b \leftarrow u_{i+1}$; $\quad v \leftarrow p_{i+1} - p_i$

2. If $a$ and $b$ are linearly dependent, and $a$ and $v$ are linearly independent, replace $b$ by its reflection about $v$ in $subspace(a, v)$, i.e., $b \leftarrow 2[(b, v)/(v, v)]v - b$.

3. If $a$ and $b$ are still linearly dependent, set $\alpha_i \leftarrow |v|/2$ and set $\beta_{i+1} \leftarrow |v|/2$, and go to step 10.

4. Let $T$ denote the $3 \times 3$ matrix for the projection onto $subspace(a, b)$ and let $w = vT$. If $w = 0$, set $\alpha_i \leftarrow |v|/2$ and set $\beta_{i+1} \leftarrow |v|/2$, and go to step 10.

5. Let $R$ denote the $3 \times 3$ rotation matrix that rotates $plane(0, a, b)$ onto the $xy$-plane such that $wR = (|w|, 0, 0)$. $a \leftarrow aR$; $b \leftarrow bR$; $w \leftarrow wR$.

6. If $a_1 < 0$, set $a \leftarrow -a$.

7. If $b_1 < 0$, set $b \leftarrow -b$.

8. If $(a_2)(b_2) > 0$, set $a_2 \leftarrow -a_2$.

9. Let $\alpha_i$ and $\beta_{i+1}$ be scalar values such that $\alpha_i a + \beta_{i+1} b = w$.

10. $m_i \leftarrow f\alpha_i u_i$; $l_{i+1} \leftarrow f\beta_{i+1} u_{i+1}$, and stop.

The scalars $\beta_1$ and $\alpha_n$ and the vectors $m_n$ and $l_1$ may be chosen arbitrarily; for example, we may choose $m_n = |l_n|u_n$ and $l_1 = |m_1|u_1$.

**Exercise 8.3:** Experiment with the above algorithm for $n = 2$ and $n = 3$ with $t_1 = t_2 = 1$. What happens when $u_1 = (p_2 - p_1)/|p_2 - p_1|$?

**Solution 8.3:** In general, if $u_i$ or $u_{i+1}$ are multiples of $p_{i+1} - p_i$, the spline segment $x_i$ reduces to the straight line segment joining $p_i$ and $p_{i+1}$. For example, if $u_i = (p_{i+1} - p_i)/|p_{i+1} - p_i|$ and $u_{i+1}$ is chosen to be linearly-independent of $p_{i+1} - p_i$, then $x_i$ and $x_{i+1}$ have join order 0 at $p_{i+1}$, and there will be a cusp at $p_{i+1}$ (unless $p_{i+2}$ and $u_{i+2}$ are chosen exactly so that the traces of $x_i$ and $x_{i+1}$ lie on the same straight line.) If such cusps are not wanted, the Fletcher-McAllister double tangent estimation scheme may be modified by establishing a lower bound for the exit and entry scale factors $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$.

Suppose $p_1, \ldots, p_n$ and $u_1, \ldots, u_n$ all lie in the $xy$-plane; choose the parameter limit values $t_1 = t_2 = \ldots = t_{n-1} = 1$. Now suppose $\alpha_i \beta_{i+1} > 0$ such that $\alpha_i u_i + \beta_{i+1} u_{i+1} = p_{i+1} - p_i$. Fletcher and McAllister have shown that, in this situation, the planar segment curve $x_i$ has no inflection points between $x_i(0)$ and $x_i(1)$ when the tension scale factor $f$ satisfies $f \leq 3$. They demonstrated that the segment curve $x_i(t)$ between $x_i(0)$ and $x_i(1)$ posesses no inflection point as follows. Recall that when $a$ and $b$ are points in the $xy$-plane, the point $a$ lies to the west of the directed line defined by the vector $b$ when $a_1 b_2 - a_2 b_1 > 0$ and $a$ lies to the east of the directed line defined by the vector $b$ when $a_1b_2 - a_2b_1 < 0$. Thus, if $[x_i'(t)]_1[x_i''(t)]_2 - [x_i'(t)]_2[x_i''(t)]_1$ never changes sign for $0 < t < 1$, then $x_i$ does not possess an inflection point between $x_i(0)$ and $x_i(1)$

Let $q_i := p_{i+1}-p_i$. Then $x_i'(t) = m_i+2t[3q_i-(2m_i+l_{i+1})]+3t^2[-2q_i+m_i + l_{i+1}]$ and $x_i''(t) = 2[3q_i - (2m_i + l_{i+1})] + 6t[-2q_i + m_i + l_{i+1}]$. Also we have $m_i = f\alpha_i u_i$, $l_{i+1} = f\beta_{i+1}u_{i+1}$, and $\alpha_i u_i + \beta_{i+1}u_{i+1} = q_i$, so $x_i'(t) = (1 - 2t)f\alpha_i u_i + [2t(3 - f) + 3t^2(f - 2)]q_i$ and $x_i''(t) = [6t(f - 2) - 2f + 6]q_i - 2f\alpha_i u_i$.

Thus $[x_i'(t)]_1[x_i''(t)]_2-[x_i'(t)]_2[x_i''(t)]_1 = [(2-f)(3t^2-3t+1)+1]2f\alpha_i (q_{i2}u_{i1} - q_{i1}u_{i2})$, and this expression changes sign as $t$ ranges over $[0, 1]$ only when $3 < f < 6$, since the maximum value of $3t^2 - 3t + 1$ for $t \in [0, 1]$ is 1 which occurs when $t = 0$ or $t = 1$; and $(2-f)\cdot 1+1 \geq 0$ for $f \leq 3$. Note when $f = 3$, the planar segment curve $x_i(t)$ has 0 curvature when $t = 0$ and when $t = 1$. For $3 < f < 6$, the curve $x_i$ has two inflection points between $x_i(0)$ and $x_i(1)$. When $f = 6$, $x_i$ has a cusp at $t = .5$, and when $f > 6$, $x_i$ forms a loop and has no inflection points.

Note that when $\alpha_i\beta_{i+1} > 0$ such that $\alpha_i u_i + \beta_{i+1}u_{i+1} = p_{i+1} - p_i$ and $0 \leq f \leq 3$, the triangle with the vertices $p_i$, $p_{i+1}$, and $p_i + \alpha_i u_i$ covers the Bézier polygon of the segment curve $x_i$. Also note that when $p_1, \ldots, p_n$ are $\mathcal{R}^2$-points of the graph of a convex-upward function, and $u_1, \ldots, u_n$ are the corresponding unit tangent vectors, the corresponding Fletcher-McAllister double tangent spline with $f < 3$ will also be a convex-upward function.

**Exercise 8.4:** (G. Y. Fletcher and D. McAllister [FM86]) Suppose $p_1, \ldots, p_n$ and $u_1, \ldots, u_n$ all lie in the $xy$-plane and choose the parameter limit values $t_1 = t_2 = \ldots = t_{n-1} = 1$. Now suppose $\alpha_i\beta_{i+1} > 0$ such that $\alpha_i u_i + \beta_{i+1}u_{i+1} = p_{i+1} - p_i$. Show that when $f = 2$, each component of the segment curve $x_i$ is a quadratic function.

**Exercise 8.5:** (G. Y. Fletcher and D. McAllister [FM86]) Suppose $p_1, \ldots, p_n$ and $u_1, \ldots, u_n$ all lie in the $xy$-plane and choose the parameter limit values $t_1 = t_2 = \ldots = t_{n-1} = 1$. Now suppose $\alpha_i\beta_{i+1} < 0$ such that $\alpha_i u_i + \beta_{i+1}u_{i+1} = p_{i+1} - p_i$. Show that the intersection point of the segment curve $x_i$ and the line segment $segment(p_i, p_{i+1})$ is $x_i(.5)$. Also show that $x_i(.5)$ is the single inflection point of $x_i$ between $x_i(0)$ and $x_i(1)$ when the tension scale factor $f$ satisfies $f \leq 6$.

**Exercise 8.6:** Study the following method for extending the Fletcher-McAllister double tangent estimation procedure for 2D data points and tangent-directions to apply to space curves. For the interpolation points $p_1, \ldots, p_n \in \mathcal{R}^3$, with associated parameter limit values $t_1, \ldots, t_{n-1}$ and chosen initial tangent directions given by unit vectors $u_1, \ldots, u_n$, define the three auxillary sets of plane interpolation points and associated unit tangent direction vectors constructed by neglecting, in turn, the first, second, and third components of the points $p_1, \ldots, p_n$ and the vectors $u_1, \ldots, u_n$ with the common parameter limit values $t_1, \ldots, t_{n-1}$. Compute the Fletcher-McAllister $\alpha_i$ and $\beta_i$ coefficients for the entry and exit tangent vectors at each plane interpolation point for each of the three auxillary sets of component data. Now take the averages of these three $\alpha$-values and three $\beta$-values associated with each point $p_i$ to obtain the final $\alpha_i$ and $\beta_i$ coefficients that will then be used to determine the entry and exit tangent vectors for the original 3-space interpolation points $p_1, \ldots, p_n$.

# 9
# Global Cubic Space Curve Splines

As we have seen, a cubic spline that interpolates given points $p_1, \ldots, p_n$ is determined by choosing associated tangent vectors $m_1, \ldots, m_n$, together with particular parameter limit values $t_1, \ldots, t_{n-1}$. Instead of determining each tangent vector $m_i$ locally in terms of $p_i$ and its immediate neighbors, we may choose the tangent vectors $m_1, \ldots, m_n$ in a global manner by requiring second derivative continuity of the curve segments $x_1, x_2, \ldots, x_{n-1}$ at the data points $p_2, \ldots, p_{n-1}$, together with any of a variety of special end conditions. Thus we may require that $x_i''(t_i) = x_{i+1}''(0)$, $x_i'(t_i) = x_{i+1}'(0)$, and $x_i(t_i) = x_{i+1}(0) = p_{i+1}$, together with some particular choice of end conditions such as $x_1''(0) = 0$ and $x_{n-1}''(t_{n-1}) = 0$.

One reason we may want to compute a global cubic spline is that its derivative is a curve with a continuous derivative (i.e., a global cubic spine is a $C^2$ curve), unlike the second derivative of a local cubic spline, which will in general have discontinuities. The derivative of a $C^1$ curve will usually have cusps at the join points; such cusps are uncommon for $C^2$ curves.

Possible global spline end conditions that produce the various types of global cubic splines listed below include:

1. A *clamped* spline: $x_1'(0) = u$ and $x_{n-1}'(t_{n-1}) = v$, where $u$ and $v$ are specified vectors.

2. A *natural* spline: $x_1''(0) = 0$ and $x_{n-1}''(t_{n-1}) = 0$.

3. A *quadratic end condition* spline: $x_1''(0) = x_2''(0)$ and $x_{n-1}''(t_{n-1}) = x_{n-2}''(t_{n-2})$. Here $x_1$ and $x_{n-1}$ reduce to quadratic curves.

4. A *cyclic* spline: $x_1'(0) = x_{n-1}'(t_{n-1})$ and $x_1''(0) = x_{n-1}''(t_{n-1})$. A cyclic spline is suitable for closed cycles or periodic curves.

5. An *anti-cyclic* spline: $x_1'(0) = -x_{n-1}'(t_{n-1})$ and $x_1''(0) = -x_{n-1}''(t_{n-1})$. When $p_1 = p_n$ an anti-cyclic spline produces a closed cycle with a cusp.

6. A *third derivative-constrained* spline:
$x_1'''(t_1) = x_2'''(0)$ and $x_{n-2}'''(t_{n-1}) = x_{n-1}'''(0)$. This is called the "not-a-knot" condition because the cubic segments $x_1$ and $x_2$ are adjacent parts of the same cubic curve, i.e., they agree in the values of their first three derivatives at their common point $p_2$, so that $p_2$ can be discarded from the data, and, with this cubic spline interpolant between $p_1$ and $p_3$, the point $p_2$ will be reconstructed. The same situation occurs for $p_{n-1}$; the cubic segments $x_{n-2}$ and $x_{n-1}$ are adjacent parts of the same cubic curve.

**Exercise 9.1:** How can we demand that the entering and exiting second derivative vectors at the points $p_2, \ldots, p_{n-1}$ be equal, when the entire spline is determined by just the first derivative vectors $m_1, \ldots, m_n$?

**Solution 9.1:** The conditions $x_i'(t_i) = x_{i+1}'(0)$ only specify that $x_i'(t_i) = m_i$ and $x_{i+1}'(0) = m_i$, and this is not enough to obtain $m_i$ itself; thus further conditions, such as $x_i''(t_i) = x_{i+1}''(0)$ are required.

The second derivative continuity conditions $x_i''(t_i) = x_{i+1}''(0)$, with $x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$ as defined above, yields $2c_i + 6d_i t_i = 2c_{i+1}$ for $1 \le i \le n-2$, and substituting for $c_i$, $d_i$, and $c_{i+1}$ produces $6(p_{i+1} - p_i)/t_i^2 - 2(2m_i + m_{i+1})/t_i + 12(p_i - p_{i+1})/t_i^2 + 6(m_i + m_{i+1})/t_i = 6(p_{i+2} - p_{i+1})/t_{i+1}^2 - 2(2m_{i+1} + m_{i+2})/t_{i+1}$. This set of $n-2$ linear vector equations specifies that the vectors $m_1, \ldots, m_n$ must satisfy the following vector equations involving vector-scalar products, where the scalars are elements of upper-triangular band width 3 matrices; such a matrix has the property that removing the first and last columns results in a diagonally-dominant tridiagonal matrix.

$$\begin{bmatrix} t_2 & 2(t_2+t_1) & t_1 & & & \\ & t_3 & 2(t_3+t_2) & t_2 & & \\ & & & \ddots & & \\ & & & t_{n-1} & 2(t_{n-1}+t_{n-2}) & t_{n-2} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix}$$

$$= 3 \cdot \begin{bmatrix} -\frac{t_2}{t_1} & \frac{t_2}{t_1} - \frac{t_1}{t_2} & \frac{t_1}{t_2} & & & \\ & -\frac{t_3}{t_2} & \frac{t_3}{t_2} - \frac{t_2}{t_3} & \frac{t_2}{t_3} & & \\ & & & \ddots & & \\ & & & -\frac{t_{n-1}}{t_{n-2}} & \frac{t_{n-1}}{t_{n-2}} - \frac{t_{n-2}}{t_{n-1}} & \frac{t_{n-2}}{t_{n-1}} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}.$$

Define the $(n-2) \times n$ matrix $C$ as:

$$C := 3 \cdot \begin{bmatrix} -\frac{t_2}{t_1} & \frac{t_2}{t_1} - \frac{t_1}{t_2} & \frac{t_1}{t_2} & & & \\ & -\frac{t_3}{t_2} & \frac{t_3}{t_2} - \frac{t_2}{t_3} & \frac{t_2}{t_3} & & \\ & & & \ddots & & \\ & & & -\frac{t_{n-1}}{t_{n-2}} & \frac{t_{n-1}}{t_{n-2}} - \frac{t_{n-2}}{t_{n-1}} & \frac{t_{n-2}}{t_{n-1}} \end{bmatrix}.$$

Define the $(n-2) \times n$ matrix $D$ as:

$$D := \begin{bmatrix} t_2 & 2(t_2+t_1) & t_1 & & & \\ & t_3 & 2(t_3+t_2) & t_2 & & \\ & & & \ddots & & \\ & & & t_{n-1} & 2(t_{n-1}+t_{n-2}) & t_{n-2} \end{bmatrix}.$$

In matrix notation, we have $Dm = Cp$, where $m$ and $p$ denote $n \times 3$ matrices whose rows are the vectors $m_1, \ldots, m_n$ and $p_1, \ldots, p_n$, respectively.

In order to solve for the vectors $m_1, \ldots, m_n$, we must add two more suitable vector equations. For the global clamped spline, with $x_1'(0) = u$ and $x_{n-1}'(t_{n-1}) = v$, we obtain the equations: $b_1 = u$ and $b_{n-1} + 2c_{n-1}t_{n-1} + 3d_{n-1}t_{n-1}^3 = v$, which reduce to $m_1 = u$ and $m_n = v$.

**Exercise 9.2:** What are the additional equations for the cases of a global natural spline, a global quadratic end condition spline, a global cyclic spline, a global anti-cyclic spline, and a global third derivative-constrained spline as defined above?

**Solution 9.2:** For a global natural spline, we have $2m_1 + m_2 = 3(p_2 - p_1)/t_1$ and $m_{n-1} + 2m_n = 3(p_n - p_{n-1})/t_{n-1}$. For a global quadratic end condition spline, we have: $(2m_2 + m_3)/t_2 - (2m_1 + m_2)/t_2 = 3(p_3 - p_2)/t_2^2 - 3(p_2 - p_1)/t_1^2$, and $m_{n-2}/t_{n-2} - (2m_{n-1} + m_n)/t_{n-1} = -3(p_{n-1} - p_{n-2})/t_{n-2}^2 - 3(p_n - p_{n-1})/t_{n-1}^2$. For a global cyclic spline, we have: $m_1 - m_n = 0$ and $2(t_1 + t_{n-1})m_1 + t_{n-1}m_2 + t_1 m_{n-2} = 3(p_2 - p_1)t_{n-1}/t_1 - 3(p_{n-1} - p_n)t_1/t_{n-1}$. For a global anticyclic spline, we have $m_1 + m_n = 0$ and $2(t_1 + t_{n-1})m_1 + t_{n-1}m_2 - t_1 m_{n-2} = 3(p_2 - p_1)t_{n-1}/t_1 + 3(p_{n-1} - p_n)t_1/t_{n-1}$. For a global third derivative-constrained spline, we have $2(p_1 - p_2)/t_1^3 + (m_1 + m_2)/t_1^2 = 2(p_2 - p_3)/t_2^3 + (m_2 + m_3)/t_2^2$, and $2(p_{n-2} - p_{n-1})/t_{n-1}^3 + (m_{n-2} + m_{n-1})/t_{n-1}^2 = 2(p_{n-1} - p_n)/t_n^3 + (m_{n-1} + m_n)/t_n^2$.

A procedure for computing $m_1, m_2, \ldots, m_n$ for a clamped global cubic spline for the points $p_1, p_2, \ldots, p_n$ and the associated parameter limit values $t_1, t_2, \ldots, t_{n-1}$, with $m_1 = u$ and $m_n = v$, is given below.

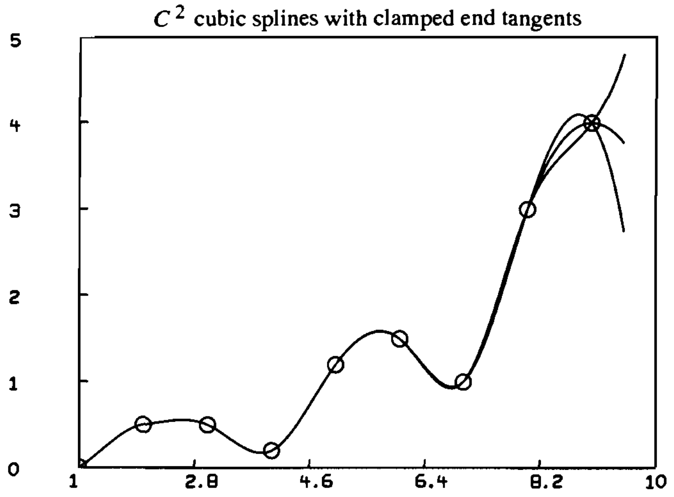
```
{ $r_1 \leftarrow 1$; $t_0 \leftarrow 0$; $m_1 \leftarrow u$; $m_n \leftarrow v$;
  for $i \leftarrow 2 : (n-1)$ do
    [$s \leftarrow t_i/t_{i-1}$;
    $m_i \leftarrow 3(-sp_{i-1} + (s - 1/s)p_i + p_{i+1}/s) - (t_i/r_{i-1})m_{i-1}$;
    $r_i \leftarrow 2(t_i + t_{i-1}) - (t_i/r_{i-1})t_{i-2}$ ];
  for $i \leftarrow (n-1) : 2$ do $m_i \leftarrow (m_i - m_{i+1}t_{i-1})/r_i$ }
```

The first loop reduces the tridiagonal system of equations for $m_1, \ldots, m_n$ to upper-triangular form by successively subtracting multiples of equation $i - 1$ from equation $i$. During the first loop, the array $m$ holds the successively-modified righthand-side values. The second loop computes the desired solution by successive back substitution.

**Exercise 9.3:** Can you reprogram the above algorithm to avoid using the array $r$ and/or avoid using two loops?

**Exercise 9.4:** Suggest several ways of choosing $m_1 = u$ and $m_n = v$ for the case of a clamped global cubic spline. *Hint*: consider linear and quadratic interpolation.

Tridiagonal linear systems of equations arise frequently in various contexts related to cubic splines. We can solve a general $n \times n$ tridiagonal nonsingular system of linear equations $Az = b$ via Gaussian elimination as follows. Since $A$ is tridiagonal, $A_{ij} = 0$ for $|i - j| > 1$ for $i = 1, 2, \ldots, n-1$. We subtract an appropriate multiple of equation $i$ from equation $i + 1$ in order to make the element $A_{i+1,j} = 0$; the final result is an upper triangu-



lar bidiagonal matrix such that no zeros appear on its main diagonal. This transformed system of equations can now be easily solved by using $z_n$ to compute $z_{n-1}$, and so on, until $z_1$ is determined; this process is called *back substitution*. It is generally convenient to arrange that each element of the main diagonal in the transformed coefficient matrix be 1.

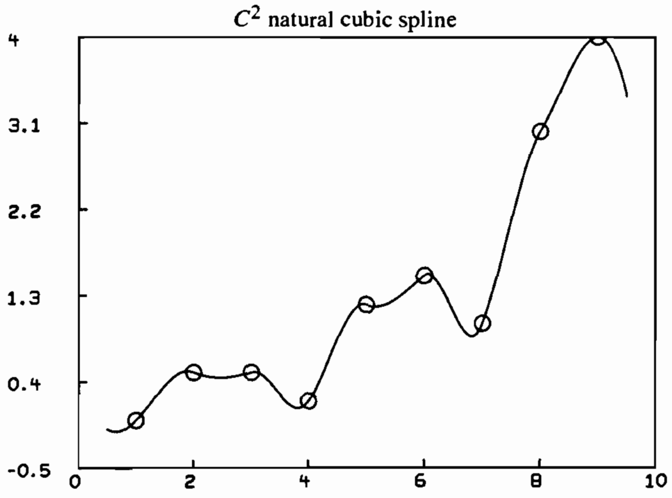
This transformation process can be described as:

```
{ for $i \leftarrow 1 : (n-1)$ do
    [$A$ row $i \leftarrow (A$ row $i)/A_{ii}$;
     $b_i \leftarrow b_i/A_{ii}$;
     $A$ row $(i+1) \leftarrow (A$ row $(i+1)) - A_{i+1,i}(A$ row $i)$;
     $b_{i+1} \leftarrow b_{i+1} - A_{i+1,i}b_i$];
  $A$ row $n \leftarrow (A$ row $n)/A_{nn}$;   $b_n \leftarrow b_n/A_{nn}$ }
```

Then $z_1, \dots, z_n$ are computed as $z_n = b_n$ and $z_i = b_i - A_{i,i+1}z_{i+1}$ for $1 \le i < n$.

An explicit program for computing $z_1, \dots, z_n$ in $Az = b$, where $A$ is an $n \times n$ non-singular tridiagonal matrix is given below.

```
{ $d \leftarrow A_{11}$; $z_1 \leftarrow b_1$;
  for $i \leftarrow 1 : (n-1)$ do
    [$c_i \leftarrow A_{i,i+1}/d$; $z_i \leftarrow z_i/d$;
     $d \leftarrow A_{i+1,i+1} - A_{i+1,i}c_i$;  $z_{i+1} \leftarrow b_{i+1} - A_{i+1,i}z_i$]
  $z_n \leftarrow z_n/d$;
  for $i \leftarrow (n-1) : 1$ do $z_i \leftarrow z_i - c_i z_{i+1}$ }
```



**Exercise 9.5:** Show that the program given earlier for computing the vectors $m_1, \ldots, m_n$ for a clamped global cubic spline is a correctly modified instance of the just-given general program for solving a tri-diagonal non-singular system.

**Exercise 9.6:** Why do tridiagonal linear systems arise in contexts related to cubic splines?

**Exercise 9.7:** State conditions for an $n \times n$ tridiagonal matrix $A$ to be non-singular, and show that when $A$ is non-singular, the general program given above never involves a divide-by-zero operation.

**Exercise 9.8:** Give a program for computing the tangent vectors $m_1, \ldots, m_n$ for the natural global cubic spline that interpolates the points $p_i, \ldots, p_n$.

**Solution 9.8:**

```
{$d \leftarrow 2; m_1 \leftarrow 3(p_2 - p_1)/t_1; t_0 \leftarrow 1; t_n \leftarrow 1;$
 for $i \leftarrow 1 : (n-1)$ do
   [$c_i \leftarrow t_{i-1}/d;\ m_i \leftarrow m_i/d;$
    $d \leftarrow 2(t_{i+1} + t_i) - t_{i+1}c_i;$
    if $(i < n-1)$ then $j \leftarrow 2$ else $j \leftarrow 1;$
    $s \leftarrow t_{i+1}/t_i;\ m_{i+1} \leftarrow 3(-sp_i + (s - 1/s)p_{i+1} + p_{i+j}/s) - t_{i+1}m_i$];
 $m_n \leftarrow m_n/(2 - c_{n-1});$
 for $i \leftarrow (n-1) : 1$ do $m_i \leftarrow m_i - c_i m_{i+1}$ }
```

**Exercise 9.9:** Let $x$ be the natural global cubic spline for the points $p_1, \dots, p_n$ with the parameter limit values $t_1, \dots, t_n$ and let $y$ be the local cubic spline for the same points $p_1, \dots, p_n$ with the same parameter values $t_1, \dots, t_n$ and the tangent vectors $m_i = p_{i+1} - p_{i-1}$ for $2 \le i \le n-1$ and $m_1 = p_2 - p_1$ and $m_n = p_n - p_{n-1}$. What is $\max_{0 \le i \le n-1} \max_{0 \le t \le t_i} |x_i(t) - y_i(t)|$? That is, how different can a natural global cubic spline and a chordal tangent local cubic spline be? *Hint*: consider $p_1 = (0,0,0)$, $p_2 = (1,0,0)$, $p_3 = (2,0,0)$, $p_4 = (3,v,0)$, $p_5 = (4,0,0,)$, $p_6 = (5,0,0)$, and $p_7 = (6,0,0)$, and consider the situation where $v \to \infty$.

**Exercise 9.10:** Compute the natural global cubic spline that interpolates the two points $p_1$ and $p_2$.

**Exercise 9.11:** Compute the natural global cubic spline that interpolates the points $p_1 = (0,0,0)$, $p_2 = (2,0,0)$, and $p_3 = (1,0,0)$ in this order, with $t_1 = 2$ and $t_1 = 1$.

**Exercise 9.12:** Compute the natural global cubic spline that interpolates the points $p_1 = (0,1)$, $p_2 = (1,1.5)$, $p_3 = (2,2)$, and $p_3 = (3,2.5)$ in this order, with $t_1 = .333333$, $t_2 = .333333$ and $t_3 = .333333$. Also compute the natural global cubic projection functional spline that interpolates the points $(0,0)$, $(.333333,1)$, $(.666666,2)$, and $(1,3)$ in this order, and the natural global cubic projection functional spline that interpolates the points $(0,1)$, $(.333333,1.5)$, $(.666666,2)$, and $(1,2.5)$ in this order. Graph these curves on the interval $-.2$ through 1.2. Compare the results you get with the corresponding splines using explicit tangent vectors, constant for each curve, that specifies the slope of the straight line data being interpolated.

**Exercise 9.13:** Explain how to estimate the slope values that define the natural global cubic spline *function* which interpolates the points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ in this order, with $x_1 < x_2 < \dots < x_n$.

**Exercise 9.14:** Can you always choose the values $t_1, \dots, t_{n-1}, x_1'(0)$, and $x_{n-1}'(t_{n-1})$ so that the graph of the resulting global $C^2$ cubic spline is monotonically-increasing in the planar case where the data points $p_1, \dots, p_n$ satisfy $p_{11} < p_{21} < \dots < p_{n2}$, and $p_{12} < p_{22} < \dots < p_{n2}$, and $p_{13} = p_{23} = \dots = p_{n3} = 0$?

Note the vector equations $Dm = Cp$ given above, augmented with suitable end condition equations, say for $p_1$ and $p_n$, can be used to solve for the points $p_1, \ldots, p_n$ in terms of the tangent vectors $m_1, \ldots, m_n$ and the $t_1, \ldots, t_{n-1}$ values. Applied iteratively, this leads to a method for solving a system of first-order ordinary differential equations with boundary value conditions. We use estimated solution points given for each given differential equation defined function to compute the corresponding solution function slopes $s_1, \ldots, s_n$, and then construct the tangent vectors $m_1, \ldots, m_n$ as $m_i = (1, s_i, 0)$. Then, for each differential equation, we solve for new points using the vector equations above. Then we reiterate this entire process until we converge.

## 9.1 Second Derivatives of Global Cubic Splines

**Exercise 9.15:** Fix $x_1''(0) = z_1$ and $x_{n-1}''(t_{n-1}) = z_n$. Impose the conditions $x_i''(t_i) = x_{i+1}''(0)$ for $1 \le i \le n-2$. Compute the tangent vectors $m_1, \ldots, m_n$ for the global cubic spline $x$ that interpolates the 3-space points $p_1, \ldots, p_n$ with the specified second derivative vector values $z_1$ and $z_n$ at the ends.

**Solution 9.15:** $x_1''(0) = 2c_1$ and $x_{n-1}''(t_{n-1}) = 2c_{n-1} + 6d_{n-1}t_{n-1}$ where $x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$. Thus we may append the vector equations

$$\begin{aligned} z_1 &= 6(p_2 - p_1)/t_1^2 - 2(2m_1 + m_2)/t_1 \qquad \text{and} \\ z_n &= -6(p_n - p_{n-1})/t_{n-1}^2 + 2(m_{n-1} + 2m_n)/t_{n-1} \end{aligned}$$

to the $n-2$ vector equations given above. The resulting $n$ vector equations define the tangent vectors $m_1, \ldots, m_n$ of the global cubic spline for the points $p_1, \ldots, p_n$ with specified second derivative vector values $z_1$ and $z_n$ at the ends in terms of $z_1$, $z_n$, and $p_1, \ldots, p_n$.

**Exercise 9.16:** The space curve $x''$ associated with a cubic spline $x$ as defined above is a piecewise linear curve consisting of $n-1$ line segments continuously joined end-to-end so as to connect $n$ points $z_1, \ldots, z_n$. What are the points $z_1, \ldots, z_n$ in terms of the points $p_1, \ldots, p_n$, the tangent vectors $m_1, \ldots, m_n$, and the scalars $t_1, \ldots, t_{n-1}$? Does a global cubic spline curve $x$ always exist such that $x_i(0) = p_i$ and $x_i(t_i) = p_{i+1}$ for $1 \le i \le n-1$, and $x_i''(t_i) = x_{i+1}''(0)$ for $1 \le i \le n-2$, and $x_i''(0) = z_i$ for $1 \le i \le n-1$ and $x_{n-1}''(t_{n-1}) = z_n$ where $p_1, \dots, p_n, t_1, \dots, t_{n-1}$, and $z_1, \dots, z_n$ are specified?

**Solution 9.16:** $z_i = 6(p_{i+1} - p_i)/t_i^2 - 2(2m_i + m_{i+1})/t_i$ for $1 \le i \le n-1$ and $z_n = -6(p_n - p_{n-1})/t_{n-1}^2 + 2(m_{n-1} + 2m_n)/t_{n-1}$. Given $t_1, \dots, t_{n-1}$, these $n$ vector equations can be solved for $m_1, \dots, m_n$ in terms of $z_1, \dots, z_n$ and $p_1, \dots, p_n$ in order to determine the corresponding cubic spline function; this cubic spline is a global spline if the associated second derivative relations hold. (These equations also yield $n-1$ independent vector equations that relate $p_1, \dots, p_n$ to $m_1, \dots, m_n$ and $z_1, \dots, z_n$.)

Define the $n \times n$ rank $n-1$ matrix $A :=$
$$\left[\begin{array}{cccccc} \frac{-6}{t_1^2} & \frac{6}{t_1^2} & & & & \\ & \frac{-6}{t_2^2} & \frac{6}{t_2^2} & & & \\ & & \ddots & & & \\ & & & & \frac{-6}{t_{n-1}^2} & \frac{6}{t_{n-1}^2} \\ & & & & \frac{6}{t_{n-1}^2} & \frac{-6}{t_{n-1}^2} \end{array}\right].$$

Define the $n \times n$ matrix $B :=$
$$\left[\begin{array}{cccccc} \frac{-4}{t_1} & \frac{-2}{t_1} & & & & \\ & \frac{-4}{t_2} & \frac{-2}{t_2} & & & \\ & & \ddots & & & \\ & & & & \frac{-4}{t_{n-1}} & \frac{-2}{t_{n-1}} \\ & & & & \frac{2}{t_{n-1}} & \frac{4}{t_{n-1}} \end{array}\right].$$

Then, in matrix terms, we have $z = Ap + Bm$.
For a global join order 2 cubic spline, recall that $Dm = Cp$ where $D$ and $C$ are $(n-2) \times n$ matrices defined above. Thus a global cubic spline exists for given points $p_1, \dots, p_n$ and given second-derivative vectors $z_1, \dots, z_n$ whenever the $n-2$ vector equations $DB^{-1}z = (C + DB^{-1}A)p$ are consistent; and in this case, the tangent vectors $m_1, \dots, m_n$ are given by $m = B^{-1}(z - Ap)$.

Let $r_0 = 0$, and $r_j := t_1 + \dots + t_j$ for $1 \le j \le n-1$. Then the piecewise-linear curve $x''(t)$, with $x''(r_j) = z_{j+1}$ for $j = 0, \dots, n-1$, is defined on $[r_{i-1}, r_i]$ by

$$x''(t) = z_i(r_i - t)/(r_i - r_{i-1}) + z_{i+1}(t - r_{i-1})/(r_i - r_{i-1}),$$

for $i = 1, \dots, n-1$.

Let

$$l_j(t) = \begin{cases} 0 & \text{for } t < r_{j-2} \\ (t - r_{j-2})/t_{j-1} & \text{for } r_{j-2} \le t < r_{j-1} \\ (r_j - t)/t_j & \text{for } r_{j-1} \le t < r_j \\ 0 & \text{for } r_j \le t \end{cases}$$

and take $r_{-1} := -1$, $r_n := r_{n-1} + 1$, $t_0 = 1$, and $t_n = 1$, so that the *tent functions* $l_j$ are defined for $j = 1, \ldots, n$. (Why is $l_j$ called a 'tent function'?) Then

$$x''(t) = z_1 l_1(t) + z_2 l_2(t) + \ldots + z_n l_n(t).$$

**Exercise 9.17:** Let $H^*_{(2)}$ be the set of join order 2, range dimension 3 global cubic splines with respect to $r_0, r_1, \ldots, r_{n-1}$ where $r_0 = 0$ and $r_j = t_1 + \cdots + t_j$. Show that $H^*_{(2)}$ is a vector space of dimension $3n + 6$.

**Exercise 9.18:** Deduce a system of linear vector equations which relate $z_1, z_2, \ldots, z_n$ and $p_1, p_2, \ldots, p_n$ for a global natural cubic spline where $z_1 = z_n = 0$.

**Solution 9.18:** Append the natural global spline end condition equations $2m_1 + m_2 = 3(p_2 - p_1)/t_1$ and $m_{n-1} + 2m_n = 3(p_n - p_{n-1})/t_{n-1}$ to the linear vector system $Dm = Cp$ given above to obtain the $n \times n$ linear vector system $\overline{D}m = \overline{C}p$. Then substitute $\overline{D}^{-1}\overline{C}p$ for $m$ in the $n \times n$ linear vector system $z = Ap + Bm$ to obtain $z = (A + B\overline{D}^{-1}\overline{C})p$. We may also obtain $z = (A\overline{C}^{-1}D + B)m$ by substituting $\overline{C}^{-1}Dm$ for $p$.

Following Lancaster and Šalkauskas [Lv86], we may obtain several useful relationships from the identity

$$x''(t) = z_j(r_j - t)/t_j + z_{j+1}(t - r_{j-1})/t_j$$

with $r_{j-1} \le t \le r_j$ for $j = 1, \ldots, n - 1$.

By integrating twice we have an equation for the global spline $x$ with $r_{j-1} \le t \le r_j$:

$$x(t) = \frac{z_j}{6}\frac{(r_j - t)^3}{t_j} + \frac{z_{j+1}}{6}\frac{(t - r_{j-1})^3}{t_j} + at + b,$$

where $a$ and $b$ are 3-tuples whose components are constants of integration. We may use the vector equations $x(r_{j-1}) = p_j$ and $x(r_j) = p_{j+1}$

to determine $a$ and $b$; these equations are: $z_j t_j^2/6 + ar_{j-1} + b = p_j$ and $z_{j+1}t_j^2/6 + ar_j + b = p_{j+1}$.

Thus,

$$at + b = \frac{1}{6}(6p_j - z_j t_j^2)(r_j - t)/t_j + \frac{1}{6}(6p_{j+1} - z_{j+1}t_j^2)(t - r_{j-1})/t_j.$$

Now, we may use the continuity of $x'$ at the knot values $r_j$ via the equations $\lim_{\epsilon\downarrow 0} x'(r_j - \epsilon) - x'(r_j + \epsilon) = 0$ to obtain equations relating $z_1, \ldots, z_n$ to $p_1, \ldots, p_n$.

Differentiating $x$ yields

$$\begin{aligned} x'(t) = {} & -\frac{z_j}{2}\frac{(r_j - t)^2}{t_j} + \frac{z_{j+1}}{2}\frac{(t - r_{j-1})^2}{t_j} \\ & - \frac{1}{6}(6p_j - z_j t_j^2)/t_j + \frac{1}{6}(6p_{j+1} - z_{j+1}t_j^2)/t_j \end{aligned}$$

for $r_{j-1} \le t \le r_j$.

Thus, using this equation for $x'$ in the two adjacent intervals $[r_{j-1}, r_j]$ and $[r_j, r_{j+1}]$ to compute $x'(r_j)$, we have

$$\begin{aligned} & \frac{1}{2}z_j t_j - \frac{1}{6t_j}((6p_j - z_j t_j^2) + (6p_{j+1} - z_{j+1}t_j^2)) \\ & = -\frac{1}{2}z_{j+1}\frac{(r_{j+1} - r_j)^2}{t_{j+1}} - \frac{1}{6t_{j+1}}((6p_{j+1} - z_{j+1}t_{j+1}^2) \\ & \quad - (6p_{j+2} - z_{j+2}t_{j+1}^2)) \end{aligned}$$

or

$$\frac{1}{6}t_j z_j + \frac{1}{3}(t_j + t_{j+1})z_{j+1} + \frac{1}{6}t_{j+1}z_{j+2} = \frac{1}{t_{j+1}}(p_{j+2} - p_{j+1}) - \frac{1}{t_j}(p_{j+1} - p_j)$$

for $j = 1, \ldots, n-2$.

These $n-2$ identities form the $(n-2) \times n$ vector system:

$$\begin{bmatrix} \frac{t_1}{6} & \frac{(t_1+t_2)}{3} & \frac{t_2}{6} & & & \\ & \frac{t_2}{6} & \frac{(t_2+t_3)}{3} & \frac{t_3}{6} & & \\ & & & \ddots & & \\ & & & \frac{t_{n-2}}{6} & \frac{(t_{n-2}+t_{n-1})}{3} & \frac{t_{n-1}}{6} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{t_1} & -\left(\frac{1}{t_1} + \frac{1}{t_2}\right) & \frac{1}{t_2} & & \\ & \frac{1}{t_2} & -\left(\frac{1}{t_2} + \frac{1}{t_3}\right) & \frac{1}{t_3} & \\ & & & \ddots & \\ & & \frac{1}{t_{n-2}} & -\left(\frac{1}{t_{n-2}} + \frac{1}{t_{n-1}}\right) & \frac{1}{t_{n-1}} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}.$$

$$\text{Let } G = \begin{bmatrix} \frac{t_1}{6} & \frac{(t_1+t_2)}{3} & \frac{t_2}{6} & & & \\ & \frac{t_2}{6} & \frac{(t_2+t_3)}{3} & \frac{t_3}{6} & & \\ & & & \ddots & & \\ & & & \frac{t_{n-2}}{6} & \frac{(t_{n-2}+t_{n-1})}{3} & \frac{t_{n-1}}{6} \end{bmatrix}.$$

$$\text{Let } H = \begin{bmatrix} \frac{1}{t_1} & -\left(\frac{1}{t_1}+\frac{1}{t_2}\right) & \frac{1}{t_2} & & & \\ & \frac{1}{t_2} & -\left(\frac{1}{t_2}+\frac{1}{t_3}\right) & \frac{1}{t_3} & & \\ & & & \ddots & & \\ & & & \frac{1}{t_{n-2}} & -\left(\frac{1}{t_{n-2}}+\frac{1}{t_{n-1}}\right) & \frac{1}{t_{n-1}} \end{bmatrix}.$$

Then, we have the $n-2$ vector equations: $Gz = Hp$.

**Exercise 9.19:** Let $x(t)$ be the natural global cubic spline for the points $p_1, \ldots, p_n$ with $t_1 = t_2 = \ldots = t_{n-1} = 1$. Show that $\int_0^t |x''(r)|^2\,dr = \min_g \int_0^t |g''(r)|^2\,dr$, where $g$ ranges over all possible local cubic splines for the points $p_1, \ldots, p_n$ with each component parametrized as a piecewise cubic polynomial. Thus the approximate energy integral $\int_0^t |g''(r)|^2\,dr$ is minimal for $g = x$.

## 9.2 Third Derivatives of Global Cubic Splines

Let $(t)_+ := \max(0, t)$. The function $(t)_+$ is called the *(negative) truncation* of $t$. A global cubic spline has an interesting representation in terms of cubes of truncated linear expressions. The basic reason that $C^2$ splines can be usefully described with powers of the function $(t)_+$ is that the constant function 0 on $[-\infty, 0]$ joins the power function $t^k$ on $[0, \infty]$ at 0 with join order $k-1$, and $((t)_+)^k$ is just a name for this piecewise combination.

**Exercise 9.20:** Let $f(t) = [(t)_+]^k$ with $k > 0$. Show that $f$ is a $C^{k-1}$ function, i.e., $f$ is continuous and $f$ has $k-1$ continuous derivatives.

Let $r_0 = 0$, and let $r_j = t_1 + \ldots + t_j$ for $1 \le j \le n-1$. As usual, we let $x_i$ denote the $i$th cubic polynomial space curve segment function of the global cubic spline $x$ where $x_i$ connects the points $p_i$ and $p_{i+1}$ for $i = 1, \ldots, n-1$. Let $\bar{x}_i(t) = x_i(t - r_{i-1})$, for $i = 1, \ldots, n-1$. Then the global cubic spline $x$ coincides with $\bar{x}_i$ on $[r_{i-1}, r_i]$.

Let $\Delta_i(t) = \bar{x}_{i+1}(t) - \bar{x}_i(t)$ for $i = 1, \dots, n-2$. Then $\Delta_i(r_i) = 0$, $\Delta_i'(r_i) = 0$, and $\Delta_i''(r_i) = 0$, and hence $r_i$ is a root of multiplicity 3 of each component, $\Delta_{i1}(t)$, $\Delta_{i2}(t)$, and $\Delta_{i3}(t)$, of the cubic 3-tuple. Therefore $\Delta_i(t) = e_i(t - r_i)^3$ for some constant 3-tuple $e_i \in \mathcal{R}^3$. Thus $\bar{x}_{i+1}(t) = \bar{x}_i(t) + e_i(t - r_i)^3$, and $r_0 = 0$ and $\bar{x}_1 = x_1$, so

$$\bar{x}_{i+1}(t) = x_1(t) + e_1(t - r_1)^3 + \cdots + e_i(t - r_i)^3.$$

Now since $(t - r_j)_+ = 0$ for $t \le r_j$, we can write

$$\bar{x}_{i+1}(t) = x_1(t) + e_1(t - r_1)_+^3 + \cdots + e_{n-2}(t - r_{n-2})_+^3$$

for $r_i \le t \le r_{i+1}$ and for $i = 0, \dots, n-2$; and moreover, without restriction,

$$x(t) = x_1(t) + e_1(t - r_1)_+^3 + \cdots + e_{n-2}(t - r_{n-2})_+^3.$$

**Exercise 9.21:** Show that $x(t) = x_1(t) + e_1(t - r_1)_+^3 + \cdots + e_{n-2}(t - r_{n-2})_+^3 + e_{n-1}(t - r_{n-1})_+^3$ on $[r_0, r_{n-1}]$ for any choice of $e_{n-1} \in \mathcal{R}^3$.

Recall that the coefficient vectors $a_i, b_i, c_i$, and $d_i$ are defined so that $x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$. Then $x_1''(t) = 2c_1 + 6d_1 t$. Since $r_0 = 0$, we can define $e_0 := d_1$ so that $x_1''(t) = 6e_0(t - r_0) + 2c_1$. Then we can write

$$x''(t) = 2c_1 + \sum_{j=0}^{n-1} 6e_j(t - r_j)_+.$$

This is in contrast to the expression obtained before:

$$x''(t) = \frac{z_j(r_j - t)}{r_j - r_{j-1}} + \frac{z_{j+1}(t - r_{j-1})}{r_j - r_{j-1}}$$

for $t \in [r_{j-1}, r_j]$ and $j = 1, \dots, n-1$, where $x''(r_{i-1}) = z_i$.

But then $x'''(t) = \sum_{j=0}^{n-1} 6e_j(\text{if } t < r_j \text{ then } 0 \text{ else } 1)$, so $x'''(r_j + \varepsilon) - x'''(r_j - \varepsilon) \to 6e_j$ as $\varepsilon \downarrow 0$. Also let us define $x'''(r_j) := \lim_{\varepsilon \downarrow 0} x'''(r_j + \varepsilon)$. Then, by differentiating $x''(t)$ above, $x'''(t) = (z_{j+1} - z_j)/(r_j - r_{j-1})$ on $[r_{j-1}, r_j)$. Therefore, if we take $r_{-1} := -1$, $r_n := r_{n-1} + 1$, and define $z_0$ such that $6d_1 = (z_2 - z_1)/r_1 + (z_1 - z_0)/r_{-1}$, and define $z_{n+1}$ such that $(z_{n+1} - z_n)/(r_n - r_{n-1}) - (z_n - z_{n-1})/(r_{n-1} - r_{n-2}) = 6e_{n-1}$, we have

$$6e_j = \frac{z_{j+2} - z_{j+1}}{r_{j+1} - r_j} - \frac{z_{j+1} - z_j}{r_j - r_{j-1}} \qquad \text{for} \quad j = 0, 1, \dots, n-1.$$

Let $w_j := (z_{j+1} - z_j)/(r_j - r_{j-1})$ for $j = 0, \ldots, n$. Thus $w_j = x'''(t)$ for $t \in [r_{j-1}, r_j)$, and $e_i = (w_{i+1} - w_i)/6$ for $i = 0, \ldots, n-1$. Recall that every component function of $x'''$ is a piecewise-constant step function and every component function of $x''$ is a piecewise-linear continuous function.

**Exercise 9.22:** Can a global cubic spline space curve have a cusp?

**Solution 9.22:** Yes. For particular choices of $p_1, \ldots, p_n$, a cusp may occur in a segment curve $x_i$ at a join point $p_i$ or $p_{i+1}$, or at at most two non-join points between $p_i$ and $p_{i+1}$; the value of $x'$ at such a cusp point is 0. The $C^2$ continuity of $x$ does *not* imply that $x$ is regularly parameterized. Such a cusp may usually be removed by reducing the values of $t_{i-1}, t_i$, and $t_{i+1}$.

## 9.3 A Variational Characterization of Natural Splines

We may define the approximate energy or smoothness functional $J(f) := \int_{r_0}^{r_{n-1}} |f''(t)|^2 \, dt$. Here the space curve $f$ must be a member of the class $F$ of 3-tuples of twice differentiable functions which correspond to space curves that interpolate the points $p_1, \ldots, p_n$ so that $f(r_{i-1}) = p_i$ for $i = 1, \ldots, n$. (And the class F must be restricted to hold only functions $f$ for which $J(f)$ exists.) Following Schoenberg [Sch46] [Sch67] [Sch64b] [Sch64a] [Yam88], we can show that the natural global cubic spline $x$ that interpolates $p_1, \ldots, p_n$ with $x''(r_0) = z_1 = x''(r_{n-1}) = z_n = 0$ minimizes $J$. The value $J(x)$ is an approximation of the potential energy of the "bent" curve $x$, where $x'$ is approximated by 1. This tells us that a natural global cubic spline nearly (but not exactly) minimizes the energy stored therein as it is "bent" to interpolate the given points $p_1, \ldots, p_n$, when the amount of bending is modest. Later we will discuss in more detail how $J(x)$ approximates the actual potential energy due to bending.

Let $f = (f_1, f_2, f_3) \in F$. Write $f(t) = x(t) + [f(t) - x(t)]$. Then, in terms of the vector dot product operation $(\cdot)$, we have

$$J(f) = \int_{r_0}^{r_{n-1}} \left\{ |x''(t)|^2 + |[f''(t) - x''(t)]|^2 + 2x''(t) \cdot [f''(t) - x''(t)] \right\} dt$$

$$= \int_{r_0}^{r_{n-1}} |x''(t)|^2 \, dt + \int_{r_0}^{r_{n-1}} |[f''(t) - x''(t)]|^2 \, dt$$

$$+ 2 \int_{r_0}^{r_{n-1}} x''(t) \cdot [f''(t) - x''(t)] \, dt.$$

And $\int_{r_0}^{r_{n-1}} x''(t)\cdot[f''(t)-x''(t)]\,dt = 0$, so $f$ must equal $x$ in order for $J(f)$ to be minimal.

Let $g = f - x$. The value $\int_{r_0}^{r_{n-1}} x''\cdot g''\,dt$ can be seen to be zero as follows. Using integration by parts, we have $\int_{r_0}^{r_{n-1}} x''\cdot g''\,dt = [g'(t)\cdot x''(t)]_{t=r_0}^{t=r_{n-1}} - \int_{r_0}^{r_{n-1}} x'''\cdot g'\,dt$. And $x''(r_0) = x''(r_{n-1}) = 0$, so $\int_{r_0}^{r_{n-1}} x''\cdot g''\,dt = -\int_{r_0}^{r_{n-1}} x'''\cdot g'\,dt$.

Recall that for $j = 0,\ldots,n$, $x'''(t) = w_j = (z_{j+1}-z_j)/(r_j - r_{j-1})$ for $t\in[r_{j-1},r_j)$ with $r_{-1} = -1$ and $r_n = r_{n-1}+1$. Also $e_i = (w_{i+1}-w_i)/6$ for $0\le i\le n-1$. We have $z_1 = z_n = 0$. Here we may take $z_0 = 0$ and $z_{n+1} = 0$ so that $w_0 = 0$ and $w_n = 0$. Then

$$\begin{aligned}
-\int_{r_0}^{r_{n-1}} x'''\cdot g'\,dt &= -\sum_{i=0}^{n-2}\int_{r_i}^{r_{i+1}} w_{i+1}\cdot g'\,dt\\
&= -\sum_{i=0}^{n-2} w_{i+1}\cdot[g(t)]_{t=r_i}^{t=r_{i+1}}\\
&= -\sum_{i=0}^{n-2} w_{i+1}\cdot[g(r_{i+1})-g(r_i)]\\
&= \sum_{i=0}^{n-1} g(r_i)\cdot[w_{i+1}-w_i]\\
&= \sum_{i=0}^{n-1} 6e_i\cdot g(r_i).
\end{aligned}$$

But $g(r_i) = f(r_i) - x(r_i) = 0$ for $i = 0,\ldots,n-1$, since both $f$ and $x$ match $p_1,\ldots,p_n$ at the knot values $r_0,\ldots,r_{n-1}$. Thus $\int_{r_0}^{r_{n-1}} x'''\cdot g'\,dt = 0$, and hence $\int_{r_0}^{r_{n-1}} x''\cdot g''\,dt = 0$.

Note the computation above makes no use of the form of the space curve $g$ until the last step; we have shown that for *any* twice differentiable space curve $g$, we have

$$\int_{r_0}^{r_{n-1}} x''\cdot g''\,dt = \sum_{i=0}^{n-1} 6e_i\cdot g(r_i).$$

**Exercise 9.23:** What is the value $J(x)$, where $x$ is the natural global cubic spline that interpolates the points $p_1,\ldots,p_n$?

**Solution 9.23:** $J(x) = \sum_{i=0}^{n-1} 6e_i\cdot p_{i+1} = \sum_{i=0}^{n-1} 6p_{i+1}e_i^{\mathrm{T}}$.

## 9.4 Weighted $\nu$-Splines

We have shown above that if we ask for the space curve $x$ that interpolates the points $p_1, \ldots, p_n$ for which $J(x)$ is minimal, then $x$ is the natural global cubic spline for the points $p_1, \ldots, p_n$.

G. Nielson[Nie86] and T. Foley[Fol87] have suggested that we consider the cubic spline $x$ that interpolates the points $p_1, \ldots, p_n$ with the knot values $r_0, \ldots, r_{n-1}$ *and* minimizes the functional

$$N(x) := \sum_{i=1}^{n-1} \int_{r_{i-1}}^{r_i} \omega_i |x''(t)|^2 \, dt + \sum_{i=1}^{n} \nu_i |x'(r_{i-1})|^2.$$

This functional is inspired by the approximate energy functional $J$ associated with natural global cubic splines. In order for $N$ to have a minimum, the scalar values $\omega_1, \ldots, \omega_{n-1}$ and $\nu_1, \ldots, \nu_n$ must be non-negative.

Note the individual segment curve second derivative integrals appearing in $N$ are *weighted* by the weights $\omega_1, \ldots, \omega_{n-1}$, and also note that the shared first derivative tangent vector magnitudes at the knot values $r_0, \ldots, r_{n-1}$ are weighted by the weights $\nu_1, \ldots, \nu_n$. Nielson originally took $\omega_1 = \cdots = \omega_{n-1} = 1$ and called the resulting spline a *$\nu$-spline*, and Foley calls the generalization where the weights $\omega_1, \ldots, \omega_{n-1}$ do not have to be identical a *weighted $\nu$-spline*. The effect of increasing the weight-value $\omega_i$ is to shorten the arc length of $x$ between the points $p_i$ and $p_{i+1}$, making $x$ increasingly *taut* there. The effect of increasing $\nu_i$ is to cause a reduction in the magnitude of the tangent vector $x'(r_{i-1}) = m_i$ at the point $p_i$, so that $x$ hugs its tangent line at $p_i$ for a shorter span, thus permitting rapid turning in the case when the data points $p_{i-1}$, $p_i$, and $p_{i+1}$ are not colinear. Thus Foley calls $\omega_i$ the *interval tension* control parameter for the curve $x$ between $p_i$ and $p_{i+1}$, and calls $\nu_i$ the *point tension* control parameter for the curve $x$ at the point $p_i$.

**Exercise 9.24:** Suppose $\nu_1 = \cdots = \nu_n = 0$. Show that the associated weighted $\nu$-spline is a $C^1$ function, but need not be a $C^2$ function.

We wish to determine the weighted $\nu$-spline $x$ with given weight values $\omega_1, \ldots, \omega_{n-1}$, and $\nu_1, \ldots, \nu_n$ that interpolates the points $p_1, \ldots, p_n$ with the knot values $r_0, \ldots, r_{n-1}$ and the associated parameter limit values $t_1, \ldots, t_{n-1}$, and which satisfies particular end conditions, such as $x''(r_0) = 0$ and $x''(r_{n-1}) = 0$, or $x'(r_0) = a$ and $x'(r_{n-1}) = b$. (Recall that $r_0 = 0$ and $r_j = \sum_{1 \le i \le j} t_i$.) Foley recommends using the modified natural end conditions $\omega_1 \lim_{\varepsilon\downarrow 0} x''(r_0+\varepsilon) = \nu_1 x'(r_0)$ and $\omega_{n-1} \lim_{\varepsilon\downarrow 0} x''(r_0-\varepsilon) = -\nu_n x'(r_{n-1})$.

In order to compute $x$, we consider the segment curves $x_1, \dots, x_{n-1}$ of $x$: $x_i(t) = f_0(t/t_i)p_i + f_1(t/t_i)p_{i+1} + f_2(t/t_i)t_i m_i + f_3(t/t_i)t_i m_{i+1}$, for $i = 1, \dots, n-1$, and $f_0(s) = 1 - 3s^2 + 2s^3$, $f_1(s) = 3s^2 - 2s^3$, $f_2(s) = s - 2s^2 + s^3$, and $f_3(s) = -s^2 + s^3$. Then

$$N(x) := \sum_{i=1}^{n-1} \int_{r_{i-1}}^{r_i} \omega_i |x_i''(t - r_{i-1})|^2\,dt + \sum_{i=1}^{n-1} \nu_i |x_i'(0)|^2 + \nu_n |x_{n-1}'(t_{n-1})|^2,$$

and we want to determine the tangent vectors $m_1, \dots, m_n$ that minimize $N$.

Since $N$ is to be minimal, subject to our chosen end condition constraints, we may augment $N$ with Lagrange multiplier terms for these constraints and then seek to compute the vectors $m_1, \dots, m_n$ and Lagrange multiplier values that minimize $N$ and correspond to a critical point of this augmented functional $\tilde{N}$.

To be specific, let us impose the clamped end conditions $x'(r_0) = m_1 = a$ and $x'(r_{n-1}) = m_n = b$, where $a$ and $b$ are given vectors. Then our augmented functional is

$$\tilde{N}(x) := \sum_{i=1}^{n-1} \int_{r_{i-1}}^{r_i} \omega_i |x_i''(t - r_{i-1})|^2\,dt + \sum_{i=1}^{n-1} \nu_i |x_i'(0)|^2 + \nu_n |x_{n-1}'(t_{n-1})|^2$$
$$+\lambda_1 \cdot (x_1'(0) - a) + \lambda_2 \cdot (x_{n-1}'(t_{n-1}) - b)$$
$$= \sum_{i=1}^{n-1} \int_{r_{i-1}}^{r_i} \omega_i |x_i''(t - r_{i-1})|^2\,dt + \sum_{i=1}^{n} \nu_i |m_i|^2 + \lambda_1 \cdot (m_1 - a) + \lambda_2 \cdot (m_n - b),$$

where $\lambda_1$ and $\lambda_2$ are vectors in $\mathcal{R}^3$ whose components are the Lagrange multipliers corresponding to the six scalar constraints specified by $m_1 = a$ and $m_n = b$.

Now to compute the $3n+6$ values that comprise $m_1, \dots, m_n$ and $\lambda_1$ and $\lambda_2$, we may form the $3n+6$ equations $\partial\tilde{N}/\partial m_{ij} = 0$ for $i = 1, \dots, n$ and $j = 1, 2, 3$, and $\partial\tilde{N}/\partial\lambda_{1j} = 0$ and $\partial\tilde{N}/\partial\lambda_{2j} = 0$ for $j = 1, 2, 3$.

Solving these equations provides us with the vectors $m_1, \dots, m_n$ which minimize $N$. These tangent vectors determine the weighted $\nu$-spline $x$ with the given weight values $\omega_1, \dots, \omega_{n-1}$, and $\nu_1, \dots, \nu_n$ which interpolates the points $p_1, \dots, p_n$ with the knot values $r_0, \dots, r_{n-1}$ and the associated parameter limit values $t_1, \dots, t_{n-1}$, and which satisfies the clamped end conditions $x'(r_0) = a$ and $x'(r_{n-1}) = b$.

Let us write $\partial\tilde{N}/\partial m_i = (\partial\tilde{N}/\partial m_{i1}, \partial\tilde{N}/\partial m_{i2}, \partial\tilde{N}/\partial m_{i3})$. Then, by substituting $f_0(t/t_i)p_i + f_1(t/t_i)p_{i+1} + f_2(t/t_i)t_i m_i + f_3(t/t_i)t_i m_{i+1}$ for $x_i(t)$ in $\tilde{N}$, and forming the vector equations $\partial\tilde{N}/\partial m_i = 0$ for $i = 1, \dots, n$, together with the similar vector constraint equations $\partial\tilde{N}/\partial\lambda_1 = 0$ and $\partial\tilde{N}/\partial\lambda_2 = 0$, we may obtain the $n+2$ vector equations that define $m_1, \dots, m_n$. For $k = 2, \dots, n-1$, we have

$$\begin{aligned} 0 = \frac{\partial\tilde{N}}{\partial m_k} &= \sum_{i=k-1}^{k} \int_{r_{i-1}}^{r_i} 2\omega_i \frac{\partial x_i''(t - r_{i-1})}{\partial m_k} \cdot x_i''(t - r_{i-1})\,dt + 2\nu_k m_k \\ &= 2\omega_{k-1} \int_{r_{k-2}}^{r_{k-1}} \frac{\partial x_{k-1}''(t - r_{k-2})}{\partial m_k} \cdot x_{k-1}''(t - r_{k-2})\,dt \\ &\quad + 2\omega_k \int_{r_{k-1}}^{r_k} \frac{\partial x_k''(t - r_{k-1})}{\partial m_k} \cdot x_k''(t - r_{k-1})\,dt + 2\nu_k m_k. \end{aligned}$$

But $\partial x_{k-1}''(t - r_{k-2})/\partial m_k$ is a $3 \times 3$ diagonal matrix of the form $g_{k-1}I$, where

$$\begin{aligned} g_{k-1} &= \frac{1}{t_{k-1}} f_3''\left(\frac{t - r_{k-2}}{t_{k-1}}\right) = 6\frac{t - r_{k-2}}{t_{k-1}^2} - \frac{2}{t_{k-1}} \\ &= \frac{6t}{t_{k-1}^2} - \frac{6r_{k-2}}{t_{k-1}^2} - \frac{2}{t_{k-1}}, \end{aligned}$$

and $\partial x_k''(t - r_{k-1})/\partial m_k$ is a $3 \times 3$ diagonal matrix of the form $g_k I$, where

$$g_k = \frac{1}{t_k} f_2''\left(\frac{t - r_{k-1}}{t_k}\right) = 6\frac{t - r_{k-1}}{t_k^2} - \frac{4}{t_k} = \frac{6t}{t_k^2} - \frac{6r_{k-1}}{t_k^2} - \frac{4}{t_k}.$$

Using integration by parts, we have the formulas

$$\begin{aligned} \int_\alpha^\beta x_i''(t - \gamma)\,dt &= x_i'(\beta - \gamma) - x_i'(\alpha - \gamma) \\ \text{and} \quad \int_\alpha^\beta t x_i''(t - \gamma)\,dt &= \beta x_i'(\beta - \gamma) - \alpha x_i'(\alpha - \gamma) \\ &\quad - x_i(\beta - \gamma) + x_i(\alpha - \gamma). \end{aligned}$$

Thus, for $k = 2, \ldots, n-1$,

$$\begin{aligned}
0 &= 2\omega_{k-1}\Big(\frac{6}{t_{k-1}^2}[r_{k-1}x'_{k-1}(t_{k-1}) - r_{k-2}x'_{k-1}(0) \\
&\qquad\qquad - x_{k-1}(t_{k-1}) + x_{k-1}(0)] \\
&\quad - (6\frac{r_{k-2}}{t_{k-1}^2} + \frac{2}{t_{k-1}})[x'_{k-1}(t_{k-1}) - x'_{k-1}(0)]\Big) \\
&\quad + 2\omega_k\Big(\frac{6}{t_k^2}[r_k x'_k(t_k) - r_{k-1}x'_k(0) - x_k(t_k) + x_k(0)] \\
&\quad - (6\frac{r_{k-1}}{t_k^2} + \frac{4}{t_k})[x'_k(t_k) - x'_k(0)]\Big) + 2\nu_k m_k \\
&= 2\frac{\omega_{k-1}}{t_{k-1}}\Big(\frac{6}{t_{k-1}}[r_{k-1}m_k - r_{k-2}m_{k-1} - p_k + p_{k-1}] \\
&\quad - (6\frac{r_{k-2}}{t_{k-1}} + 2)[m_k - m_{k-1}]\Big) \\
&\quad + 2\frac{\omega_k}{t_k}\Big(\frac{6}{t_k}[r_k m_{k+1} - r_{k-1}m_k - p_{k+1} + p_k] \\
&\quad - (6\frac{r_{k-1}}{t_k} + 4)[m_{k+1} - m_k]\Big) + 2\nu_k m_k.
\end{aligned}$$

Now, define $\alpha_j := \omega_j/t_j$. Thus, since $r_0 = 0$ and $r_{j-1} = r_j - t_j$, we have, for $k = 2, \ldots, n-1$,

$$\begin{aligned}
&2\alpha_{k-1}m_{k-1} + (\nu_{k-1} + 4\alpha_{k-1} + 4\alpha_k)m_k + 2\alpha_k m_{k+1} \\
&\qquad = 6\alpha_k(p_{k+1} - p_k)/t_k + 6\alpha_{k-1}(p_k - p_{k-1})/t_{k-1}.
\end{aligned}$$

By similar manipulations, we have

$$\begin{aligned}
0 &= \frac{\partial \tilde{N}}{\partial m_1} = \int_{r_0}^{r_1} 2\omega_1 \frac{\partial x_1''(t)}{\partial m_1} \cdot x_1''(t)\,dt + 2\nu_1 m_1 + \lambda_1 \\
&= +2\frac{\omega_1}{t_1}\Big(\frac{6}{t_1}[r_1 m_2 - p_2 + p_1] - 4[m_2 - m_1]\Big) + 2\nu_1 m_1 + \lambda_1.
\end{aligned}$$

So, $(\nu_1 + 4\alpha_1)m_1 + 2\alpha_1 m_2 + \lambda_1/2 = 6\alpha_1(p_2 - p_1)/t_1$.

Further,

$$\begin{aligned}0 &= \frac{\partial \tilde{N}}{\partial m_n} = \int_{r_{n-2}}^{r_{n-1}} 2\omega_{n-1}\frac{\partial x''_{n-1}(t)}{\partial m_n}\cdot x''_{n-1}(t)\,dt + 2\nu_n m_n + \lambda_2\\ &= 2\frac{\omega_{n-1}}{t_{n-1}}\Big(\frac{6}{t_{n-1}}[r_{n-1}m_n - r_{n-2}m_{n-1} - p_n + p_{n-1}]\\ &\qquad - (6\frac{r_{n-2}}{t_{n-1}} + 2)[m_n - m_{n-1}]\Big) + 2\nu_n m_n + \lambda_2.\end{aligned}$$

So, $2\alpha_{n-1}m_{n-1} + (\nu_n + 4\alpha_{n-1})m_n + \lambda_2/2 = 6\alpha_{n-1}(p_n - p_{n-1})/t_{n-1}$.

Also,

$$0 = \frac{\partial \tilde{N}}{\partial \lambda_1} = m_1 - a \text{ and } 0 = \frac{\partial \tilde{N}}{\partial \lambda_2} = m_n - b$$

so we recover the constraints $m_1 = a$ and $m_n = b$, and the equations corresponding to $\partial\tilde{N}/\partial m_1 = 0$ and $\partial\tilde{N}/\partial m_n = 0$ reduce to $(\nu_1 + 4\alpha_1)a + 2\alpha_1 m_2 + \lambda_1/2 = 6\alpha_1(p_2 - p_1)/t_1$ and $2\alpha_{n-1}m_{n-1} + (\nu_n + 4\alpha_{n-1})b + \lambda_2/2 = 6\alpha_{n-1}(p_n - p_{n-1})/t_{n-1}$. Together with the $n-2$ vector equations

$$\begin{aligned}&2\alpha_{k-1}m_{k-1} + (\nu_{k-1} + 4\alpha_{k-1} + 4\alpha_k)m_k + 2\alpha_k m_{k+1}\\ &\quad = 6\alpha_k(p_{k+1} - p_k)/t_k + 6\alpha_{k-1}(p_k - p_{k-1})/t_{k-1},\end{aligned}$$

for $k = 2, \dots, n-1$, we have just enough independent equations to determine the vectors $m_2, \dots, m_{n-1}$, and $\lambda_1$ and $\lambda_2$. Only the latter $n-2$ vector equations given above are needed to obtain $m_2, \dots, m_{n-1}$; they form a diagonally dominant tridiagonal system of $n-2$ linearly-independent linear vector equations. Thus, given $m_1 = a$ and $m_n = b$, we can easily solve this system to obtain $m_2, \dots, m_{n-1}$ as well. (Note that we may allow $t_k = 0$, if we specify that $y/0 = 0$ for $y \in \mathcal{R}$.)

**Exercise 9.25:** Show that the equations $m_1 = a$, $m_n = b$, and

$$\begin{aligned}&2\alpha_{k-1}m_{k-1} + (\nu_{k-1} + 4\alpha_{k-1} + 4\alpha_k)m_k + 2\alpha_k m_{k+1}\\ &\quad = 6\alpha_k(p_{k+1} - p_k)/t_k + 6\alpha_{k-1}(p_k - p_{k-1})/t_{k-1},\end{aligned}$$

or $k = 2, \dots, n-1$, reduce to those that define a clamped global $C^2$ cubic spline when $\omega_1 = \cdots = \omega_{n-1} = 1$ and $\nu_1 = \cdots = \nu_n = 0$.

**Exercise 9.26:** The weights $\omega_1, \dots, \omega_{n-1}$ only appear in the form $\omega_j/t_j$ in the equations that define a clamped weighted $\nu$-spline. Does this mean that $\omega_1, \ldots, \omega_{n-1}$ are superfluous because suitably choosing $t_1, \ldots, t_{n-1}$ suffices to obtain the same values for $\alpha_1, \ldots, \alpha_{n-1}$ which we can get by choices of $\omega_1, \ldots, \omega_{n-1}$? *Hint*: look at how $t_k$ appears.

**Exercise 9.27:** What are the equations that define the weighted $\nu$-spline that minimizes $N$ with *no* special end conditions?

**Exercise 9.28:** What are the defining equations for the tangent vectors $m_1, \ldots, m_n$ of a weighted $\nu$-spline when we impose the natural end conditions $x''(r_0) = 0$ and $x''(r_{n-1}) = 0$?

**Exercise 9.29:** Foley [Fol87] shows that the weighted $\nu$-spline with modified natural end conditions that minimizes $N$ is, in fact, the minimizer of $N$ among *all* twice differentiable functions that interpolate $p_1, \ldots, p_n$ with the associated parameter-values $r_0, \ldots, r_{n-1}$ for which $N$ exists. Follow the classic proof that the minimizer of $J$ is a natural global cubic spline and write an analogous proof of Foley's result.

# 10
# Smoothing Splines

Let $f_1, f_2, \ldots, f_n$ be real values. Suppose we consider the data points $(r_0, f_1), (r_1, f_2), \ldots, (r_{n-1}, f_n)$, with $r_0 \leq r_1 \leq \ldots \leq r_{n-1}$, to be points of $\mathcal{R}^2$ sampled with error from the graph of some unknown function $f : \mathcal{R} \to \mathcal{R}$ such that $f_j = f(r_{j-1}) + \epsilon_j$ for $j = 1, \ldots, n$. If the form of the function $f$ were known, except for some unknown parameter values, we could then determine those values by the least squares method, where the desired parameter values are those parameter values that minimize $E = \sum_{j=1}^{n} \lambda_j (f_j - f(r_{j-1}))^2$ where $\lambda_1, \ldots, \lambda_n$ are weight values that are inversely proportional to the variances of the errors in the points. When the function $f$ is not known, we may seek a smooth function $x$ that "fits" the data points without regard to any particular functional form. A cubic spline can fit the data exactly, but we may obtain a less oscillatory approximating function if we take $x$ to be the function that minimizes the functional $E(f) + \rho J(f)$ over all suitable functions $f$, where $\rho$ is a positive constant and

$$E(f) = \sum_{j=1}^{n} \lambda_j (f_j - f(r_{j-1}))^2.$$

Here $J(f) = \int_{r_0}^{r_{n-1}} |f''(t)|^2 dt$ acts as a "penalty" term which favors functions with small second derivatives. The effect of $J$ can be increased or diminished by suitably choosing the multiplier $\rho$.

Now suppose $x$ is a function for which $E(x) + \rho J(x)$ is minimal. Let

$p_j := x(r_{j-1}) \in \mathcal{R}$ for $j = 1, \dots, n$. Whatever function we have for $x$, only its values at $r_0, r_1, \dots, r_{n-1}$ affect the value $E(x)$. And whatever the values $p_1, \dots, p_n$ are, the corresponding function $x$, for which $E(x) + \rho J(x)$ is minimal, is a natural global cubic spline that interpolates the points $(r_0, p_1), \dots, (r_{n-1}, p_n)$, since this choice minimizes $J$ while having no effect on $E$.

Thus we need only determine the scalar values $p_1, \dots, p_n$ that define a corresponding natural global cubic spline $x$ with knot values $r_0, \dots, r_{n-1}$, such that $E(x)+\rho J(x)$ is minimal over all choices of the values $p_1, \dots, p_n$. The spline $x$ is then our desired smooth approximating function for the points $(r_0, f_1), \dots, (r_{n-1}, f_n)$ with the weights $\lambda_1, \dots, \lambda_n$. The natural global cubic spline $x$ is called the *optimal smoothing spline* for the points $(r_0, f_1), \dots, (r_{n-1}, f_n)$ with the weights $\lambda_1, \dots, \lambda_n$ and the smoothing parameter $\rho$ [DeB78].

**Exercise 10.1:** What is the optimal smoothing spline for two points?

## 10.1 Computing an Optimal Smoothing Spline

Lancaster and Šalkauskas [Lv86] have given an elegant algorithm for computing the optimal smoothing spline function $x : \mathcal{R} \rightarrow \mathcal{R}$ for the $n \geq 3$ points in the $xy$-plane $(r_0, f_1), \dots, (r_{n-1}, f_n)$ with $r_0 < r_1 < \dots < r_{n-1}$. (We may replace every group of points having identical $x$-coordinates by their average in order to obtain a reduced set of data points for which $r_i < r_{i+1}$.) We can generalize this algorithm to produce an optimal smoothing spline for points in $\mathcal{R}^3$ with knot values $r_0, \dots, r_{n-1}$, but this is not necessary; once we know how to compute a simple functional optimal smoothing spline, we can compute the optimal smoothing spline space curve for points in 3-space by computing the 3 component real-valued optimal smoothing spline functions individually.

Let $\Lambda$ be the $n \times n$ diagonal matrix $diag(\lambda_1, \dots, \lambda_n)$. Then $E(x) = (p-f)^T \Lambda (p-f)$ where $p = [p_1, \dots, p_n]^T$ and $f = [f_1, \dots, f_n]^T$. We are seeking the values $p_1, \dots, p_n$, where $p_j := x(r_{j-1})$ for $j = 1, \dots, n$.

Also, let $L(t) = [l_1(t), \dots, l_n(t)]^T$, where $l_1(t), \dots, l_n(t)$ are the tent functions defined above, such that $x''(t) = z_1 l_1(t) + \dots + z_n l_n(t)$ where here $z_1, \dots, z_n$ are real values, not 3-tuples. Let $z = [z_1, \dots, z_n]^T$. Then $x''(t) = z^T L(t)$, and

$$J(x) = \int_{r_0}^{r_{n-1}} |x''(t)|^2 dt = \int_{r_0}^{r_{n-1}} z^T L(t) L(t)^T z \, dt.$$

Now recall the $n-2$ equations $Gz = Hp$ given above, where here $z$ and $p$ are column $n$-vectors; and $G$ and $H$ are both $(n-2) \times n$ matrices defined in terms of the values $t_1, \ldots, t_{n-1}$ where $t_i = r_i - r_{i-1}$. For the case of a natural global cubic spline, $z_1 = z_n = 0$, so we may drop the first and last columns of $G$ to obtain the $(n-2) \times (n-2)$ symmetric matrix $\overline{G} := G$ col $2 : (n-1)$ and write $\overline{G}v = Hp$ where $v := [z_2, \ldots, z_{n-1}]^T$. Then $v = \overline{G}^{-1}Hp$, and since $z_1 = z_n = 0$,

$$J(x) = \int_{r_0}^{r_{n-1}} (\overline{G}^{-1}Hp)^T \overline{L}(t)\overline{L}(t)^T (\overline{G}^{-1}Hp)\,dt$$

where $\overline{L}(t) = [l_2(t), \ldots, l_{n-1}(t)]^{\mathrm{T}}$.

Note $\overline{G}^{-1\mathrm{T}} = \overline{G}^{-1}$. Thus,

$$J(x) = p^T H^T \overline{G}^{-1} [\int_{r_0}^{r_{n-1}} \overline{L}(t)\overline{L}(t)^T dt](\overline{G}^{-1}Hp).$$

Now $\int_{r_0}^{r_{n-1}} \overline{L}(t)\overline{L}(t)^T dt$ is the $(n-2) \times (n-2)$ matrix whose $(i, j)$th element is $\int_{r_0}^{r_{n-1}} l_{i+1}(t) l_{j+1}(t)\,dt$, and this matrix is just the symmetric matrix $\overline{G}$! Thus, $J(x) = p^T H^T \overline{G}^{-1}\overline{G}\,\overline{G}^{-1} Hp = p^T H^T \overline{G}^{-1} Hp$.

Now to compute $p$ such that $E(x) + \rho J(x)$ is minimal, we differentiate $E(x) + \rho J(x)$ with respect to $p_1, \ldots, p_n$ and equate these derivatives to zero to obtain

$$\frac{1}{2}\frac{d}{dp}[(p-f)^T \Lambda (p-f) + \rho p^T H^T \overline{G}^{-1} Hp] = \Lambda(p-f) + \rho H^T \overline{G}^{-1} Hp = 0.$$

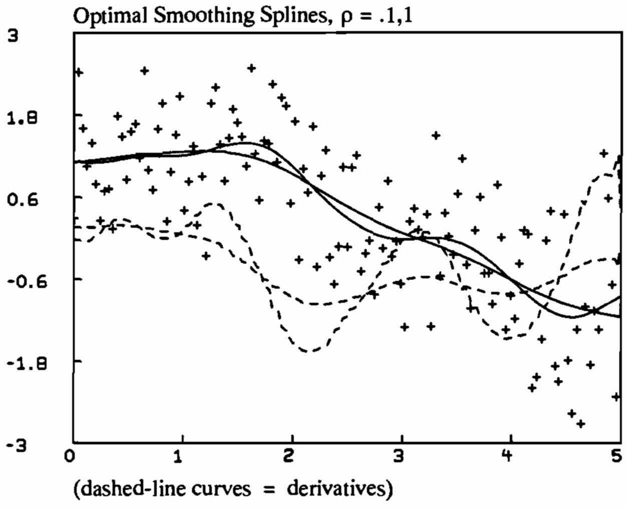
Since $\overline{G}v = Hp$, we have $H\Lambda^{-1}(\Lambda p - \Lambda f + \rho H^T \overline{G}^{-1}\overline{G}v) = 0$, so $\overline{G}v - Hf + \rho H \Lambda^{-1} H^T v = 0$. Thus, $(\overline{G} + \rho H\Lambda^{-1}H^T)v = Hf$, so $v = (\overline{G} + \rho H \Lambda^{-1} H^T)^{-1} Hf$. Thus, $\Lambda(p - f) + \rho H^T \overline{G}^{-1}\overline{G}v = \Lambda(p - f) + \rho H^T v = 0$, so $p = f - \rho\Lambda^{-1}H^T v$.

Now $z = [0, v_1, \ldots, v_{n-2}, 0]^T$, and given $p$ and $z$, we can compute the optimal smoothing spline $x$ using

$$x(t) = \frac{1}{6t_j}[z_j(r_j - t)^3 + z_{j+1}(t - r_{j-1})^3 + (6p_j - z_j t_j^2)(r_j - t) + (6p_{j+1} - z_{j+1} t_j^2)(t - r_{j-1})] \quad \text{for} \quad r_{j-1} \le t \le r_j.$$

This process involves solving the single $(n-2) \times (n-2)$ linear system $(\overline{G} + \rho H \Lambda^{-1} H^T)v = Hf$ which is given in terms of known matrices, i.e., $\overline{G}^{-1}$ is not directly involved.

Note that $p_1, \ldots, p_n$ are linear functions of the values $f_1, \ldots, f_n$, since $p = Mf$ where the $n \times n$ matrix $M$ is defined as $(\Lambda + \rho H^T \overline{G}^{-1} H)^{-1}\Lambda$.



**Exercise 10.2:** Show that when $\rho$ approaches 0, then $p$ approaches $f$, and when $\rho$ approaches $\infty$, $p_j$ approaches $s(r_{j-1})$ where the graph of the function $s$ is the best fitting line for the points $(r_0, f_1), \dots, (r_{n-1}, f_n)$ in the least squares sense, with the weights $\lambda_1, \dots, \lambda_n$.

**Exercise 10.3:** Let $s(t)$ be the best-fitting line for the points $(r_0, f_1), \dots, (r_{n-1}, f_n)$ in the least squares sense, with the weights $\lambda_1, \dots, \lambda_n$, and let $c(t)$ be the natural global cubic spline for these points. Criticize the smoothing function $\alpha b(t) + (1-\alpha)c(t)$.

**Exercise 10.4:** Show that if $f$ is an $n \times 3$ matrix whose rows are points in $\mathcal{R}^3$ with $f$ row $i$ corresponding to the parameter value $r_{i-1}$, then the 3-tuples $z_1, \dots, z_n$ and $p_1, \dots, p_n$ that determine the parametric optimal smoothing cubic spline space curve with the parameter knot values $r_0 < r_1 < \dots < r_{n-1}$ are given by $z_1 = z_n = 0$, $v = [z_2, \dots, z_{n-1}]^T = (\overline{G} + \rho H \Lambda^{-1} H^T)^{-1} H f$, and $p = f - \rho \Lambda^{-1} H^T v$. *Hint*: define

$$\begin{aligned} E(x) &= trace[(p_x - f)^T \Lambda (p_x - f)] \text{ and} \\ J(x) &= trace[\int_{r_0}^{r_{n-1}} {z_x}^T L(t) L(t)^T z_x \, dt] \end{aligned}$$

where $p_x = [x(r_0), \dots, x(r_{n-1})]^T$ and $z_x = [x''(r_0), \dots, x''(r_{n-1})]^T$. (Recall that $trace(M)$ is the sum of the elements of $diag(M)$.)

## 10.2 Computing the Smoothing Parameter

As mentioned above, the positive constant $\rho$ determines the amount of variation exhibited in the optimal smoothing spline $x$. In order to choose a value for $\rho$, we may use the method of *cross validation* [Wah90]. Let

$$E_{\langle i\rangle}(f) = \sum_{\substack{1\le j\le n\\ j\ne i}} \lambda_j (f_j - f(r_{j-1}))^2,$$

and let $x_{\langle i\rangle}(t)$ denote the natural global cubic spline with the knot values $r_0, \ldots, r_{i-2}, r_i, \ldots, r_{n-1}$ that minimizes $E_{\langle i\rangle}(f) + \rho J(f)$ when $f = x_{\langle i\rangle}$. Note that the limits of the integral defining the functional $J$ are always the values $r_0$ and $r_{n-1}$. The function $x_{\langle i\rangle}(t)$ depends upon $\rho$; it is the optimal smoothing spline with the weights $\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_n$ and the smoothing parameter $\rho$ for the data points $(r_0, f_1), \ldots, (r_{n-1}, f_n)$, *excluding* the point $(r_{i-1}, f_i)$. We will write $x_{\langle i\rangle}(t; \rho)$ to indicate this dependence on $\rho$. Now define the cross validation objective function $\mu(\rho) = \sum_{i=1}^{n} \lambda_i (f_i - x_{\langle i\rangle}(r_{i-1}; \rho))^2$. Then we may choose $\rho$ to be that value which minimizes $\mu(\rho)$.

**Exercise 10.5:** Show that the optimal smoothing spline for the points $(r_0, f_1), \ldots, (r_{n-1}, f_n)$ with the weights $\lambda_1, \ldots, \lambda_n$ and the smoothing parameter $\rho$ is $x_{\langle i\rangle}$ when $\lambda_i = 0$.

Wahba [Wah90] has derived a more amenable formula for $\mu(\rho)$ involving only the smoothing spline function $x$ instead of the functions $x_{\langle 1\rangle}, \ldots, x_{\langle n\rangle}$. Let $h(t)$ be the choice for the function $g(t)$ that minimizes $\lambda_i(z - g(r_{i-1}))^2 + E_{\langle i\rangle}(g) + \rho J(g)$. Note that $h$ depends on $i$, $z$, and $\rho$, as well as its argument $t$. Let us write $h(t)$ as $h(t; i, z, \rho)$ to indicate this dependence.

Let $\hat{f}_i := x_{\langle i\rangle}(r_{i-1}; \rho)$; this is the cross validation estimate of the "left-out" value $f_i$. Now note that $h(t; i, \hat{f}_i, \rho) = x_{\langle i\rangle}(t)$. This is because $\lambda_i(\hat{f}_i - x_{\langle i\rangle}(r_{i-1}))^2 + E_{\langle i\rangle}(x_{\langle i\rangle}) + \rho J(x_{\langle i\rangle}) = E_{\langle i\rangle}(x_{\langle i\rangle}) + \rho J(x_{\langle i\rangle}) < E_{\langle i\rangle}(g) + \rho J(g)$ for any $C^2$ function $g \ne x_{\langle i\rangle}$, so $x_{\langle i\rangle}$ minimizes $\lambda_i(\hat{f}_i - g(r_{i-1}))^2 + E_{\langle i\rangle}(g) + \rho J(g)$.

Now consider $\alpha_i(\rho) := (\hat{f}_i - x(r_{i-1}; \rho))/(\hat{f}_i - f_i)$, where $x(t; \rho)$ is the optimal smoothing spline for the points $(r_0, f_1), \ldots, (r_{n-1}, f_n)$ with the weights $\lambda_1, \ldots, \lambda_n$ and the smoothing parameter $\rho$. Note that $h(r_{i-1}; i, f_i, \rho) = x(r_{i-1}; \rho)$ (!) Then

$$\alpha_i(\rho) = (h(r_{i-1}; i, \hat{f}_i, \rho) - h(r_{i-1}; i, f_i, \rho))/(\hat{f}_i - f_i).$$

Thus $\alpha_i(\rho)$ is a forward difference estimate (if $f_i < \hat{f}_i$) or backward difference estimate (if $f_i > \hat{f}_i$) of the slope of $h(r_{i-1}; i, z, \rho)$, taken as a function of $z$, at the point $z = f_i$.

Recall that $p_i = x(r_{i-1}; \rho) = M_{i1} f_1 + \ldots + M_{in} f_n$, expressed in terms of the $n \times n$ matrix $M = (\Lambda + \rho H^T \overline{G}^{-1} H)^{-1} \Lambda$. Now if we consider $x(r_{i-1}; \rho)$ as a function of $f_i$ and denote this function as $x(r_{i-1}; \rho)[f_i]$, then we see that $x(r_{i-1}; \rho)[f_i]$ is a linear function of $f_i$ with $\partial x(r_{i-1}; \rho)[f_i]/\partial f_i = M_{ii}$. But $h(r_{i-1}; i, f_i, \rho) = x(r_{i-1}; \rho)[f_i]$, so $\partial h(r_{i-1}; i, f_i, \rho)/\partial f_i = M_{ii}$. And since $h(r_{i-1}; i, f_i, \rho)$ is a linear function of $f_i$, the difference estimate $\alpha_i(\rho)$ is *exact*, and $\alpha_i(\rho) = M_{ii}(\rho)$, where we now explicitly write $M_{ii}$ as a function of $\rho$ to show its dependence on $\rho$.

Now $\alpha_i(\rho) = M_{ii}(\rho) = (\hat{f}_i - x(r_{i-1}; \rho))/(\hat{f}_i - f_i)$, so $f_i - \hat{f}_i = (f_i - x(r_{i-1}; \rho))/(1 - M_{ii}(\rho))$. Thus the cross validation objective function $\mu(\rho) = \sum_{i=1}^{n} \lambda_i (f_i - x(r_{i-1}; \rho))^2/(1 - M_{ii}(\rho))^2$. We may now compute the smoothing parameter $\rho$ by minimizing this expression for $\mu(\rho)$.

**Exercise 10.6:** Explain how to compute $diag(M)$.

Wahba recommends [Wah90] that we instead compute $\rho$ by minimizing the *generalized cross validation objective function*: $U(\rho) := \sum_{i=1}^{n} \lambda_i (f_i - x(r_{i-1}; \rho))^2/(1 - d(\rho))^2$, where $d(\rho) := trace(M\Lambda) = \lambda_1 M_{11} + \ldots + \lambda_n M_{nn}$ with $\lambda_1 + \ldots + \lambda_n = 1$.

**Exercise 10.7:** Show that

$$U(\rho) = | \Lambda^{1/2}(I - M) f |^2 / (1 - trace(\Lambda M))^2.$$

An alternate somewhat less costly approach to determining $\rho$ is to first choose *adjusted* points $(r_0, \tilde{f}_1), \ldots, (r_{n-1}, \tilde{f}_n)$ which are points we "prefer" the smoothing spline to pass through or near. For example, we may determine these adjusted points by means of another distinct smoothing process such as a weighted moving average or moving median process. Let $\tilde{f} := [\tilde{f}_1, \ldots, \tilde{f}_n]^T$. Now we may choose $\rho$ such that $|\tilde{f} - f + \Lambda^{-1} H^T (\rho^{-1}\overline{G} + H\Lambda^{-1}H^T)^{-1} H f|$ is minimal.

Note that $(\rho^{-1}\overline{G} + H\Lambda^{-1}H^T)^{-1} H f$ can be computed by solving the $(n-2) \times (n-2)$ linear system $(\rho^{-1}\overline{G} + H\Lambda^{-1}H^T) y = H f$ for the vector $y$; this can be efficiently done by means of Gaussian elimination. Iterative conjugate gradient methods may be used to save space, but this is not necessary if we exploit the banded nature of the matrices involved.

**Exercise 10.8:** Show that $\rho^{-1}\overline{G} + H\Lambda^{-1}H^T$ is a symmetric matrix of bandwidth 5.

**Exercise 10.9:** Let $\eta(\rho) = |\tilde{f} - f + \Lambda^{-1}H^T(\rho^{-1}\overline{G} + H\Lambda^{-1}H^T)^{-1}Hf|^2$. Show that $\eta(\rho)$ is minimal exactly when $\eta(\rho)^{1/2}$ is minimal. Show that if $\eta'(\rho_1) = 0$, then $\eta(\rho_1)$ is minimal. Show that $\eta'(\rho) = -2[\tilde{f} - f + \Lambda^{-1}H^T\ (\rho^{-1}\overline{G} + H\Lambda^{-1}H^T)^{-1}Hf]^T\ [\Lambda^{-1}H^T(\rho^{-1}\overline{G} + H\Lambda^{-1}H^T)^{-1}\ (\rho^{-2}\overline{G})(\rho^{-1}\overline{G} + H\Lambda^{-1}H^T)^{-1}Hf] = -2[\tilde{f} - f + \Lambda^{-1}H^T\rho\ (\overline{G} + \rho H\Lambda^{-1}H^T)^{-1}Hf]^T\ [\Lambda^{-1}H^T(\overline{G} + \rho H\Lambda^{-1}H^T)^{-1}\overline{G}(\overline{G} + \rho H\Lambda^{-1}H^T)^{-1}\ Hf]$. *Hint*: differentiate $A^{-1}A = I$ to obtain a formula for $(A^{-1})'$. Explain how to compute $\eta'(\rho)$ by sucessively solving two linear systems in place of computing matrix inverses.

## 10.3 Best Fit Smoothing Cubic Splines

Often it is useful to compute a real-valued non-optimal smoothing spline function for the data points $(s_1, f_1), \ldots, (s_k, f_k)$ as a range dimension 1 cubic spline $x$ with the *fixed* knot values $r_0 < r_1 < \ldots < r_{n-1}$, where $r_0 \le s_1 \le s_2 \le \ldots \le s_k \le r_{n-1}$, for which $\sum_{i=1}^k (f_i - x(s_i))^2$ is minimal. Generally $n << k$.

We shall describe how to compute such an associated range dimension 1 join order 1 non-optimal smoothing natural cubic spline function $x$ for the data points $(s_1, f_1), \ldots, (s_k, f_k)$. For the fixed knot values $r_0, \ldots, r_{n-1}$, the spline function $x$ is determined by the real values $p_1, \ldots, p_n$ and $m_1, \ldots, m_n$, so we want to determine real values $p_1, \ldots, p_n$ and $m_1, \ldots, m_n$ corresponding to the knot values $r_0, \ldots, r_{n-1}$, such that $\sum_{i=1}^k (f_i - x(s_i))^2$ is minimal.

Let $t_i = r_i - r_{i-1}$. The cubic spline polynomial segment function $x_i$ defined on $[r_{i-1}, r_i]$ so that $x_i(r_{i-1}) = p_i$, $x_i(r_i) = p_{i+1}$, $x_i'(r_{i-1}) = m_i$, and $x_i'(r_i) = m_{i+1}$ can be written in terms of the Hermite blending functions as $x_i(t) = f_0((t - r_{i-1})/t_i)p_i + f_1((t - r_{i-1})/t_i)p_{i+1} + f_2((t - r_{i-1})/t_i)t_i m_i + f_3((t - r_{i-1})/t_i)t_i m_{i+1}$ where $f_0(t) = 1 - 3t^2 + 2t^3$, $f_1(t) = 3t^2 - 2t^3$, $f_2(t) = t - 2t^2 + t^3$, and $f_3(t) = -t^2 + t^3$.

For $0 \le i < n - 2$, let $k_i$ be the number of data points $(s_j, f_j)$ such that $r_i \le s_j < r_{i+1}$, let $k_{n-2}$ be the number of data points $(s_j, f_j)$ such that $r_{n-2} \le s_j \le r_{n-1}$ and let $h_i = k_0 + k_1 + \ldots + k_{i-1}$ with $h_0 = 0$ and $h_{n-1} = k$. Thus $r_0 \le s_1 \le \cdots \le s_{h_1} < r_1 \le s_{h_1+1} \le \cdots \le s_{h_2} < r_2 \le \ldots \le s_{h_{n-2}+1} \le \cdots \le s_k \le r_{n-1}$.

For $q = 1, \ldots, n - 1$, let $C_q$ denote the $k_q \times 4$ matrix whose

$i$th row is $[C_{qi1}, C_{qi2}, C_{qi3}, C_{qi4}]$ where $C_{qi1} = f_0((s_{h_{q-1}+i} - r_{q-1})/t_q)$, $C_{qi2} = t_q f_2((s_{h_{q-1}+i} - r_{q-1})/t_q)$, $C_{qi3} = f_1((s_{h_{q-1}+i} - r_{q-1})/t_q)$, and $C_{qi4} = t_q f_3((s_{h_{q-1}+i} - r_{q-1})/t_q)$. Thus $(C_q \text{ row } i)[p_q, m_q, p_{q+1}, m_{q+1}]^T = x(s_{h_{q-1}+i})$.

Now define the $k \times 2n$ matrix $X$ such that $X_{ij} = 0$, except that the $X$ row $((h_{q-1}+1) : h_q)$ col $((2q-1) : (2q+2)) = C_q$ for $q = 1, \ldots, n-1$. Then $\sum_{i=1}^{k}(f_i - x(s_i))^2 = |Xv - f|^2$ where $v = [p_1\, m_1\, p_2\quad m_2 \ldots p_n\, m_n]^T$. Then $v = X^+ f$ minimizes $|Xv - f|^2$ where $X^+$ denotes the Moore-Penrose pseudo-inverse of the matrix $X$. Thus we have determined the real values $p_1, \ldots, p_n$ and $m_1, \ldots, m_n$ as required.

**Exercise 10.10:** Suppose $s_1 < s_2 < \ldots < s_k$. Explain how $n$ and $r_0, \ldots, r_{n-1}$ can be chosen so that the spline $x$ exactly interpolates the points $(s_1, f_1), \ldots, (s_k, f_k)$.

**Exercise 10.11:** Explain how to determine $x$ as the global range dimension 1 *join order 2* natural cubic spline function with the fixed knot values $r_0 < r_1 < \ldots < r_{n-1}$ which minimizes $\sum_{i=1}^{k}(f_i - x(s_i))^2$.

## 10.4 Monotone Smoothing Splines

Suppose we have a sequence of points $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_1 < x_2 < \ldots < x_n$ that are known to be noisy samples from the graph of a "hidden" *monotonic* function $m$, with $y_i = m(x_i) + \epsilon_i$, and suppose the errors $\epsilon_1, \ldots, \epsilon_n$ cause the sequence $y_1, y_2, \ldots, y_n$ to fail to be monotonic. We may adjust the values $y_1, y_2, \ldots, y_n$ by adding various positive or negative numbers $d_1, d_2, \ldots, d_n$ to define $z_i = y_i + d_i$ for $1 \le i \le n$, where we will choose $d_1, d_2, \ldots, d_n$ such that $y_1 + d_1 \le y_2 + d_2 \le \ldots \le y_n + d_n$, i.e., so that the points $(x_1, z_1), \ldots, (x_n, z_n)$ are monotonically increasing. (The monotonically decreasing case proceeds in a parallel manner). We may say that we have *monotonized* the data points $(x_1, y_1), \ldots, (x_n, y_n)$. This is a form of smoothing of the data points [Har91].

Generally, we want to choose $d_1, d_2, \ldots, d_n$ to minimize $d_1^2 + d_2^2 + \ldots + d_n^2$, subject, of course, to the $n - 1$ constraints $y_1 + d_1 \le y_2 + d_2$, $y_2 + d_2 \le y_3 + d_3, \ldots, y_{n-1} + d_{n-1} \le y_n + d_n$. The solution to this linearly-constrained minimization problem is the point $d = (d_1, d_2, \ldots, d_n)$ in the constraint region $\{v \in \mathcal{R}^n \mid y_1 + v_1 \le y_2 + v_2 \le \cdots \le y_n + v_n\}$ that is closest to $(0, 0, \ldots, 0)$.

**Exercise 10.12:** The $i$th constraint $y_i + d_i \le y_{i+1} + d_{i+1}$ corresponds to $c_i \cdot d \le b_i$, where $b_i = y_{i+1} - y_i$ and $c_i = (0, \dots, 0, 1, -1, 0, \dots, 0)$, i.e., $c_{ij} = 0$ except $c_{ii} = 1$ and $c_{i,i+1} = -1$. Let the $n \times (n-1)$ matrix $C$ be defined by $C\ col\ i = c_i^T$. Then our constraints can be written in matrix form as $dC \le b$. Now we may delete those columns of $C$ and the corresponding elements of the vector $b$ where the matching constraints are satisfied; this yields the reduced set of *active* constraints $d\tilde{C} \le \tilde{b}$. Is $d = \tilde{b}\tilde{C}^+$ the desired solution of our linearly-constrained minimization problem? Recall that $d = \tilde{b}\tilde{C}^+$ is the vector of least length that minimizes $| d\tilde{C} - \tilde{b} |$.

Algorithmically $d_1, d_2, \dots, d_n$ can be computed by repetitively replacing every pair of adjacent values $y_i$ and $y_{i+1}$ for which $y_i > y_{i+1}$ with their average $(y_i + y_{i+1})/2$ until no such pairs are present. Although this process may never terminate, the values $y_1, \dots, y_n$ will approach limiting values. The following program jumps directly to those limiting values and constructs the monotonizing correction vector $d$ for the vector $y = (y_1, \dots, y_n)$.

```
1. set $z \leftarrow y$, set $j \leftarrow 1$.
2. set $i \leftarrow j$, set $j \leftarrow j + 1$, if $j > n$ go to step 8.
3. if $z_i \le z_j$ go to step 2.
4. set $a \leftarrow (z_i + \dots + z_j)/(j - i + 1)$.
5. if $i - 1 \ge 1$ and $z_{i-1} > a$ {set $i \leftarrow i - 1$ and go to step 4}.
6. if $j + 1 \le n$ and $a > z_{j+1}$ {set $j \leftarrow j + 1$ and go to step 4}.
7. set $z_i \leftarrow a$, set $z_{i+1} \leftarrow a, \dots,$ set $z_j \leftarrow a$ and go to step 2.
8. set $d_i \leftarrow z_i - y_i$ for $1 \le i \le n$ and stop.
```

**Exercise 10.13:** Revise the algorithm above to avoid repetitive calculation in the computation of the average in step 4.

**Exercise 10.14:** Show that at step 8, the mean of the values $y_1, \dots, y_n$ and the mean of the values $z_1, \dots, z_n$ are identical. What can you say about the mean of the original error values $\epsilon_1, \dots, \epsilon_n$?

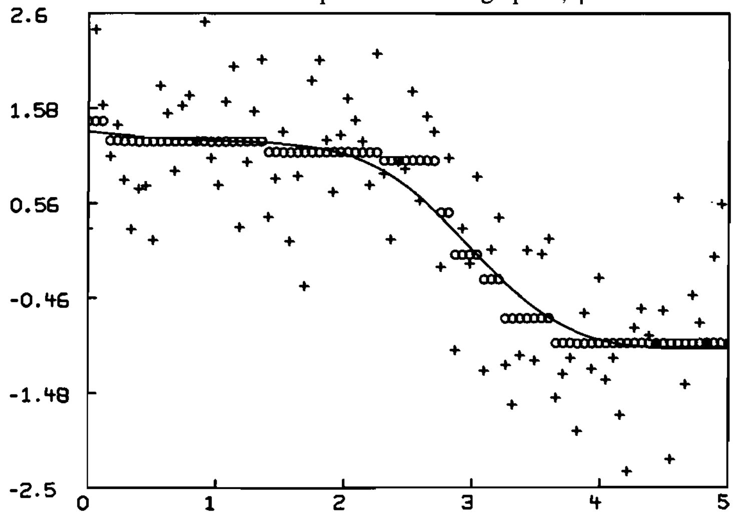
**Exercise 10.15:** Explain how to solve the linearly constrained minimization problem of minimizing $| d_1 | + \ldots + | d_n |$ subject to the constraints $y_1 + d_1 \le y_2 + d_2, y_2 + d_2 \le y_3 + d_3, \ldots, y_{n-1} + d_{n-1} \le y_n + d_n$.

We may now construct a monotone smoothing spline for the points $(x_1, y_1), \ldots, (x_n, y_n)$ by first monotonizing these data points, then computing the points $p_1, \ldots, p_n$ for the ordinary optimal smoothing spline and monotonizing them, if necessary. Then we can compute the natural global cubic spline function slopes for these points and adjust these slopes with Hyman's monotonizing slope adjustment scheme discussed earlier, if necessary. Now the resulting points and slopes define our desired monotonic smoothing spline function.

**Exercise 10.16:** Is the optimal smoothing spline function for a sequence of monotonic data points monotonic?

**Exercise 10.17:** Explain how to use monotonic smoothing splines to estimate the density function for a random variable that is assumed to have a unimodal density, given a sequence of independent samples of this random variable.

Monotonic Optimal Smoothing Spline, ρ = .5



(circles = monotonized data points)

# 11 Geometrically Continuous Cubic Splines

Recall that we can construct a double tangent cubic spline which is tangent-vector geometrically continuous at its join points by choosing pairs of entry and exit tangent vectors at each point $p_i$, where each pair may have differing magnitudes, but the same direction. It is common to call a tangent vector geometrically continuous curve a $G^1$ curve, in the same way that a tangent vector algebraically continuous curve is commonly called a $C^1$ curve. A $C^0$ curve is merely a continuous curve, and in general, a $C^k$ curve has $k$ or more successive continuous derivative vectors. A $G^0$ curve is just a $C^0$ continuous curve. A $G^1$ curve has a continuous unit tangent vector curve, and a $G^2$ curve also has a continuous curvature function. A regularly parameterized $C^k$ curve, whose tangent vector does not vanish, is necessarily also a $G^k$ curve. We often wish to focus on a particular point $x(t)$ of a space curve $x$ and consider whether $x$ is $G^k$ continuous *at that point*.

A general definition of $G^k$ continuity is that a curve $c$ is $G^k$ *continuous* at the point $c(t)$ if there exists a regularly parameterized curve whose trace is equal to the trace of $c$ and which is $C^k$ continuous at $c(t)$. In particular, we can always look at the arc length parameterization $\hat{c}$ of $c$; if $\hat{c}$ is a $C^k$ curve at $c(t)$, then $c$ is a $G^k$ curve at $c(t)$. Thus, a $G^k$ continuous curve would be be a $C^k$ continuous curve, if not for an "unfortunate" parameterization.

Recall that a global cubic spline is not generally a regularly parameterized curve and it may exhibit isolated cusps; that is, a global cubic spline

$x$ may have a discontinuous unit tangent vector at points where $x' = 0$. However, a join order 2 global cubic spline *is* a $C^\infty$ continuous curve between join points and is a $C^2$ continuous curve everywhere. We will seek to modify this property at the interior join points $p_2, \ldots, p_{n-1}$ and only ask for $G^k$ continuity at the interior join points (generally with $k = 2$). Thus we want a global cubic spline which is $C^\infty$ continuous between join points and which is $G^k$ continuous at the interior join points; we shall call such a spline a $G^k$ *joined* spline.

**Exercise 11.1:** Show that a cubic spline curve is $C^\infty$ continuous everywhere except at its join points.

Clearly then, a $G^1$-joined cubic spline curve $x$ has the property that there exist $n-2$ positive constants $\beta_1, \ldots, \beta_{n-2}$, such that $\beta_i x_i'(t_i) = x_{i+1}'(0)$ for $1 \leq i < n-1$.

A $G^2$ curve $x$ is a $G^1$ curve with the additional property that the curvature of $x$, $|x' \times x''|/|x'|^3$, is a continuous function. For a $G^2$ joined cubic spline curve $x$, this need only be checked at the interior join points $p_2, \ldots, p_{n-1}$. In order for the curvature of $x$ to be continuous at $p_{i+1}$, $|x_i'(t_i) \times x_i''(t_i)|/|x_i'(t_i)|^3 = |x_{i+1}'(0) \times x_{i+1}''(0)|/|x_{i+1}'(0)|^3$ must hold. Suppose the scalar $\beta_i > 0$ is the value such that $\beta_i x_i'(t_i) = x_{i+1}'(0)$ holds as a consequence of the $G^1$ continuity of $x$ at $p_{i+1}$. Then $x$ is $G^2$ continuous at $p_{i+1}$ if $\beta_i^2 |x_i'(t_i) \times x_i''(t_i)| = |x_i'(t_i) \times x_{i+1}''(0)|$; this will be true if $\beta_i^2 x_i''(t_i)$, plus any multiple of $x_i'(t_i)$, equals $x_{i+1}''(0)$. A multiple of $x_i'(t_i)$ may be included because the cross product of two vectors that are multiples of each other is 0. Let $\gamma_1, \ldots, \gamma_{n-2}$ be any desired real multiplier values. Then $\beta_i^2 x_i''(t_i) + \gamma_i x_i'(t_i) = x_{i+1}''(0)$ together with $\beta_i x_i'(t_i) = x_{i+1}'(0)$ for $1 \leq i < n-1$, are sufficient conditions for $x$ to be $G^2$ continuous at the interior join points $p_2, \ldots, p_{n-1}$. A slightly deeper analysis reveals that these are necessary conditions as well.

Given a non-retrograde space curve $x$, not necessarily regularly parameterized, we seek conditions on $x$, such that $x$ is guaranteed to be a $G^k$ curve. Let $\hat{x}(s)$ denote the arc length parameterization corresponding to the space curve $x$. By definition, $x$ is a $G^k$ curve if and only if $\hat{x}$ is a $C^k$ curve.

Let us focus on the behavior of $x$ at a point $x(t_1) = \hat{x}(s_1)$. Suppose $x$ is a $G^k$ curve at $x(t_1)$ and that, in the neighborhood of the point $x(t_1)$, $x$ can be defined piecewise as the joining of two space curves $a$ and $b$ at the point $x(t_1)$, so that $x(t) = a(t)$ for $t \leq t_1$ and $x(t) = b(t - t_1)$ for $t \geq t_1$. Now suppose further that there exist two real valued functions $u$ and $v$ such that the arc length parameterization of $x$ satisfies $\hat{x}(s) = a(u(s))$ for $s \leq s_1$ and $\hat{x}(s) = b(v(s))$ for $s \geq s_1$. When these assumptions hold, we can deduce conditions that determine that $x$ is a $G^k$ curve at $x(t_1)$. Basically, the smoothness with which the arc length parameterized forms of $a$ and $b$ join determines the algebraic continuity of $\hat{x}$ at $x(t_1)$, and the smoothness with which the curves $a$ and $b$, with their parameterizations inherited from $x$, join determines the coefficients in the associated geometric continuity relations.

**Exercise 11.2:** Show that the above assumptions do hold when $x$ is a regularly parameterized curve.

Since $\hat{x}$ is a $C^k$ curve at $x(t_1)$, the left derivative, defined as $\lim_{\epsilon\downarrow 0}(\hat{x}(s_1) - \hat{x}(s_1 - \epsilon))/\epsilon$, equals the ordinary derivative $[a(u(s))]' \mid_{s=s_1}$ and the right derivative, defined as $\lim_{\epsilon\downarrow 0}(\hat{x}(s_1 + \epsilon) - \hat{x}(s_1))/\epsilon$, equals the ordinary derivative $[b(v(s))]' \mid_{s=s_1}$; and this same correspondence holds up to the $k$th derivative.

Now, by the chain rule,

$$[a(u(s))]' \mid_{s=s_1} = u'(s_1)a'(u(s_1)),$$

and

$$[b(v(s))]' \mid_{s=s_1} = v'(s_1)b'(v(s_1)),$$

so

$$\frac{u'(s_1)}{v'(s_1)}a'(u(s_1)) = b'(v(s_1)).$$

Similarly,

$$[a(u(s))]'' \mid_{s=s_1} = u''(s_1)a'(u(s_1)) + u'(s_1)^2 a''(u(s_1)),$$

and

$$[b(v(s))]'' \mid_{s=s_1} = v''(s_1)b'(v(s_1)) + v'(s_1)^2 b''(v(s_1)),$$

so

$$\frac{u'(s_1)^2}{v'(s_1)^2}a''(u(s_1)) + [u''(s_1) - v''(s_1)\frac{u'(s_1)}{v'(s_1)}]a'(u(s_1)) = b''(v(s_1)).$$

Now the space curves $a$ and $b$ can be adjusted so that $u'(s_1)$, $v'(s_1)$, $u''(s_1)$ and $v''(s_1)$ can be any desired non-zero values. Thus the relation $\frac{u'(s_1)}{v'(s_1)}a'(u(s_1)) = b'(v(s_1))$ can be written $\beta a'(u(s_1)) = b'(v(s_1))$, where $\beta$ is an arbitrary positive constant. Similarly, the relation $\frac{u'(s_1)^2}{v'(s_1)^2}a''(u(s_1)) +$

$[u''(s_1) - v''(s_1)\frac{u'(s_1)}{v'(s_1)}]a'(u(s_1)) = b''(v(s_1))$ can be written $\beta^2 a''(u(s_1))) + \gamma a'(u(s_1)) = b''(v(s_1))$, where $\beta$ is the positive constant in the prior relation and where $\gamma$ is an arbitrary constant.

Finally, $a^{(j)}(u(s_1)) = x_-^{(j)}(t_1)$ and $b^{(j)}(v(s_1)) = x_+^{(j)}(t_1)$ for $j = 1, 2, \ldots, k$, where $x_-^{(j)}(t_1)$ denotes the $j$th left derivative vector of $x$ and $x_+^{(j)}(t_1)$ denotes the $j$th right derivative vector of $x$. Thus $\beta x'_-(t_1) = x'_+(t_1)$ whenever $x$ is $G^1$ continuous at $x(t_1)$, and in addition, $\beta^2 x''_-(t_1) + \gamma x'_-(t_1) = x''_+(t_1)$ whenever $x$ is also $G^2$ continuous at $x(t_1)$. By repeated differentiation and use of the chain rule, we can obtain a set of necessary conditions that must hold for $x$ to be $G^k$ continuous at $x(t_1)$. The two such conditions given here for $G^2$ continuity at a point are exactly the sufficient conditions for $G^2$ continuity of a spline curve at a join point obtained earlier.

**Exercise 11.3:** Given the points $p_1, \ldots, p_n$, the entry tangent vectors $l_1, \ldots, l_n$, the exit tangent vectors $m_i, \ldots, m_n$, and the parameter limit values $t_i, \ldots, t_{n-1}$, suppose that the associated double tangent cubic spline $x$ is a $G^1$ joined space curve. Propose a method to adjust the magnitudes of the entry and exit tangent vectors so that $x$ becomes a $C^1$ joined curve. Can this be done by adjusting only the parameter limit values $t_1, \ldots, t_{n-1}$?

**Exercise 11.4:** Given the points $p_1, \ldots, p_n$, the tangent vectors $m_1, \ldots, m_n$, and the parameter limit values $t_1, \ldots, t_{n-1}$ that determine a cubic spline space curve $x$, construct a procedure to compute $\hat{x}(s)$, where $\hat{x}(s)$ is the arc length parameterized version of $x$.

**Solution 11.4:** We can compute $x(t)$ for $0 \le t \le t_1 + t_2 + \ldots + t_{n-1}$, and thus we can compute the arc length $l(t)$ along the curve $x$ from $x(0)$ to $x(t)$ for $0 \le t \le t_1 + t_2 + \ldots + t_{n-1}$ as $l(t) = \int_0^t |x'(\tau)| d\tau$. Now to compute $\hat{x}(s)$, we first determine the integer $i$ such that $l(t_1 + t_2 + \ldots + t_{i-1}) < s \le l(t_1 + t_2 + \ldots + t_i)$. Then $\hat{x}(s)$ is a point on the cubic segment curve $x_i$; this point may be determined by solving the equation $\int_{t_1+t_2+\ldots+t_{i-1}}^t |x'(\tau)| d\tau = s$ for the value $t$. Then $\hat{x}(s) = x(t)$.

## 11.1 Beta Splines

Barsky [BB83] [BBB87] has proposed several distinct families of $G^2$ joined global splines called *beta splines*. We shall present a method for computing one form of beta spline represented as a double tangent $G^2$ joined global cubic spline that interpolates a sequence of points $p_1, \ldots, p_n$. Such curves include the $C^2$ global cubic splines as a special case.

A $G^2$ joined global cubic spline must satisfy the $G^2$ continuity conditions at the interior knot values $r_1, r_2, \ldots, r_{n-2}$ corresponding to the join points $p_2, \ldots, p_{n-1}$, where $r_i = t_1 + \ldots + t_i$. Thus for $1 \le i < n-1$, we have $\beta_i x_i'(t_i) = x_{i+1}'(0)$ and $\beta_i^2 x_i''(t_i) + \gamma_i x_i'(t_i) = x_{i+1}''(0)$ for some values $\beta_1 > 0, \beta_2 > 0, \ldots, \beta_{n-2} > 0$ and $\gamma_1, \ldots, \gamma_{n-2}$. When $\beta_i = 1$ and $\gamma_i = 0$ for $1 \le i < n-1$, we have an ordinary $C^2$ global cubic spline. As with $C^2$ global cubic splines, two additional vector end conditions must be postulated to completely determine a particular $G^2$ joined global cubic spline that interpolates the points $p_1, p_2, \ldots, p_n$.

Let $m_1, m_2, \ldots, m_n$ be the exit tangent vectors at points $p_1, p_2, \ldots, p_n$ and let $l_1, l_2, \ldots, l_n$ be the corresponding entry tangent vectors at the points $p_1, p_2, \ldots, p_n$. Thus, $x_i'(0) = m_i$ and $x_i'(t_i) = l_{i+1}$. We want to determine $2n-2$ vectors that define a double tangent cubic spline, such that the $2n-4$ $G^2$ continuity condition equations that ensure $G^2$ joining at $p_1, \ldots, p_{n-1}$ hold, together with two additional end condition equations.

Recall that the $i$th segment curve of a double tangent cubic spline is $x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$, so that $x_i'(t) = b_i + 2c_i t + 3d_i t^2$ and $x_i''(t) = 2c_i + 6d_i t$ for $1 \le i < n$ where $a_i = p_i$, $b_i = m_i$, $c_i = 3(p_{i+1} - p_i)/t_i^2 - (2m_i + l_{i+1})/t_i$, and $d_i = 2(p_i - p_{i+1})/t_i^3 + (m_i + l_{i+1})/t_i^2$. The $G^2$ continuity condition $\beta_i x_i'(t_i) = x_{i+1}'(0)$ is equivalent to the relation $\beta_{i-1} l_i = m_i$, and the $G^2$ continuity condition $\beta_i^2 x_i''(t_i) + \gamma_i x_i'(t_i) = x_{i+1}''(0)$ is equivalent to the relation $\beta_i^2(2c_i + 6d_i t_i) + \gamma_i(b_i + 2c_i t_i + 3d_i t_i^2) - 2c_{i+1} = 0$.

Let $h_i := 2(\beta_i^2 + \gamma_i t_i)$ and let $k_i := 6\beta_i^2 + 3\gamma_i t_i$. Then we may eliminate $a_i, b_i, c_i, d_i$, and $l_i$ in the equations above to obtain

$$\left[\frac{2k_i - 3h_i}{t_i^2}\right] p_i + \left[\frac{6}{t_{i+1}^2} + \frac{3h_i - 2k_i}{t_i^2}\right] p_{i+1} - \left[\frac{6}{t_{i+1}^2}\right] p_{i+2}$$
$$= \left[\frac{2h_i - k_i - \gamma_i}{t_i}\right] m_i - \left[\frac{4}{t_{i+1}} + \frac{k_i - h_i}{t_i \beta_i}\right] m_{i+1} - \left[\frac{2}{t_{i+1}\beta_{i+1}}\right] m_{i+2}$$

for $1 \le i < n-1$.

These $n-2$ vector equations constitute an $(n-2) \times n$ tridiagonal system of linear vector equations whose solution requires that values for the constants $\beta_1, \ldots, \beta_{n-1}$ and $\gamma_1, \ldots, \gamma_{n-2}$ be known. Thus, when we choose the constant $\beta_{n-1}$ in addition to $\beta_1, \ldots, \beta_{n-2}$ and $\gamma_1, \ldots, \gamma_{n-2}$ and add two additional independent equations which specify $m_1$ and $m_n$, we may then compute $m_1, \ldots, m_n$ and then compute $l_2, \ldots, l_n$ via the relation $l_i = m_i/\beta_{i-1}$. The double tangent cubic spline defined by the computed exit and entry tangent vectors $m_1, \ldots, m_{n-1}$ and $l_2, \ldots, l_n$ is a $G^2$-joined global cubic spline that interpolates the points $p_1, \ldots, p_n$; this spline is the principal beta spline defined by Barsky. The $2n-3$ shape control parameters $\beta_1, \ldots, \beta_{n-1}$ and $\gamma_1, \ldots, \gamma_{n-2}$ may be chosen to obtain any of a variety of shape effects not available with a $C^2$ global cubic spline.

**Exercise 11.5:** Experiment with choices of $\beta_1, \ldots, \beta_{n-1}$ and $\gamma_1, \ldots, \gamma_{n-2}$. What happens if $\beta_i = 0$ or if $\beta_i < 0$ for some $i$?

**Exercise 11.6:** Does having the freedom to choose the parameter limit values $t_1, \ldots, t_{n-1}$ allow any new curves to be had beyond those obtained by fixing $t_1 = t_2 = \ldots = t_{n-1} = 1$ and varying just $\beta_1, \ldots, \beta_{n-1}$ and $\gamma_1, \ldots, \gamma_{n-2}$?

**Exercise 11.7:** Suppose we specify a $G^2$ joined double tangent cubic spline $x$ by requiring that, for $i = 1, \ldots, n-1$, the $i$th segment curve $x_i$ with the parameter limit value $t_i$ satisfies $x_i(0) = p_i$, $x_i(t_i) = p_{i+1}$, $x_i'(0) = \alpha_i u_i$, and $x_i'(t_i) = \delta_{i+1} u_{i+1}$, where $u_i$ and $u_{i+1}$ are unit vectors, and $\alpha_i$ and $\delta_{i+1}$ are positive constants; moreover we specify that the curvature of $x$ be continuous with specified values $K_i$ at the join points, so that $|x_i''(0) \times x_i'(0)|/|x_i'(0)|^3 = K_i$ and $|x_i''(t_i) \times x_i'(t_i)|/|x_i'(t_i)|^3 = K_{i+1}$. Show that the values $\alpha_1, \ldots, \alpha_{n-1}$ and $\delta_2, \ldots, \delta_n$ satisfy the $2n-2$ equations:

$$|[6(p_{i+1} - p_i)/t_i - 2\delta_{i+1} u_{i+1}] \times u_i| = \alpha_i^2 K_i t_i, \text{ and}$$
$$|[6(p_i - p_{i+1})/t_i + 2\alpha_i u_i] \times u_{i+1}| = \delta_{i+1}^2 K_{i+1} t_i,$$

for $i = 1, \ldots, n-1$. Thus when these equations can be solved for $\alpha_1, \ldots, \alpha_{n-1}$ and $\delta_2, \ldots, \delta_n$ in terms of the specified points $p_1, \ldots, p_n$, the specified parameter limit values $t_1, \ldots, t_{n-1}$, the *unit* tangent vectors $u_1, \ldots, u_n$, and the positive curvature values $K_1, \ldots, K_n$, we can explicitly obtain the $G^2$ joined double tangent cubic spline curve with specified unit tangents and specified curvature values at the interpolation points $p_1, \ldots, p_n$.

# 12 Quadratic Space Curve Based Cubic Splines

**Exercise 12.1:** Show that a quadratic space curve $q(t) = a + bt + ct^2$ is a planar curve, and thus prove that a cubic is the lowest degree polynomial space curve which has non-zero torsion.

**Exercise 12.2:** Show that a quadratic space curve $q(t) = a + bt + ct^2$ is a parabola.

**Solution 12.2:** A parabola is the only conic section that is the graph of a quadratic function. We have $q(t) = [t^2\ t\ 1]A$ where $A$ row $1 = c$, $A$ row $2 = b$, and $A$ row $3 = a$. Any real $m \times n$ matrix $A$ can be written as $A = LQ$, where $Q$ is an $n \times n$ orthogonal matrix and $L$ is an $m \times n$ lower triangular matrix. Thus, we may write $q(t) = [t^2\ t\ 1]LQ$. Now since $Q$ and $Q^{-1}$ are orthogonal matrices, $q(t)Q^{-1}$ has the same curvature and torsion as $q(t)$. But $q(t)Q^{-1} = [t^2\ t\ 1]L = (L_{11}t^2 + L_{21}t + L_{31}, L_{22}t + L_{32}, L_{33})$. Now changing the parameterization with $s = L_{22}t + L_{32}$, we obtain the identical curve $g(s) := q((s - L_{32})/L_{22})Q^{-1} = (L_{11}[(s - L_{32})/L_{22}]^2 + L_{21}[(s - L_{32})/L_{22}] + L_{31}, s, L_{33})$. This is clearly a planar quadratic curve defined by a quadratic function, and thus a parabola.

**Exercise 12.3:** Find the parametric quadratic space curve which passes through the point $a$, with the tangent vector $v$, and which passes through the point $b$, based on a parameter $t \in [0, h]$.

**Solution 12.3:** $q_h(t; a, v, b) = q_h(t) = a + vt + ((b - a - hv)/h^2)t^2$ with $q_h(0) = a$, $q_h'(0) = v$, and $q_h(h) = b$. This family of parabolas is related by $q_h(\alpha h; a, v, b) = q_r(\alpha r; a, (h/r)v, b)$, so that different choices of $h$ are equivalent to fixing $h$ and adjusting the magnitude of the vector, $v$. Differentiating $q_h$ shows that $q_h$ is not a uniform velocity parameterized curve for any $h$, so that no choice of $h$ gives the arc length parameterized parabola. The coefficient vector $(b - a - hv)/h^2$ is parallel to the principal axis of the parabola $q_h$. The point of greatest curvature on $q_h$ is $q_h(-(c, v)/(2(c, c)))$ and that maximal curvature value is $|2c|$, where $c = (b - a - hv)/h^2$.

**Exercise 12.4:** Find the parametric family of parabolas which passes through the three distinct points $p_1$, $p_2$, and $p_3$ in $\mathcal{R}^3$.

**Solution 12.4:** For each choice of parameter values $g$ and $h$ such that $0 < g < h$, we have the quadratic space curve, $q(t) = a + bt + ct^2$, for $-\infty < t < \infty$, where $a = p_1$, $b = -gp_3/(h(h - g)) + hp_2/(g(h - g)) - (h+g)p_1/(hg)$, and $c = p_3/(h(h-g)) - p_2/(g(h-g)) + p_1/(hg)$. The corresponding space curve $q$ satisfies $q(0) = p_1$, $q(g) = p_2$, and $q(h) = p_3$. $a + bt + ct^2$ is not an arc length parameterized constant velocity representation for any admissible choice of $g$ and $h$. We can show that $q'(g) = 3gp_3/(h(h-g)) + (h-2g)p_2/(g(h-g)) + (g-h)p_1$, and, when $h = 2g$, we have $q'(g) = (p_3 - p_1)/(2g)$.

**Exercise 12.5:** How many numbers determine a quadratic space curve?

**Solution 12.5:** Seven numbers and one bit are sufficient to determine a quadratic space curve. Three numbers and a bit suffice to specify the plane of the curve, and one number is the angle with respect to the projection of the positive $x$-axis (or positive $y$-axis if the plane is normal to the $x$-axis) which determines the direction of the principal axis of the parabola. Finally, choose this principal axis direction as the direction of the $y$-axis in the plane of the parabola and take the $x$-axis to be perpendicular, with both axes passing through the projection of the origin in the specified plane. Then three further numbers corresponding to the values $y(-1)$, $y(0)$, and $y(1)$ for the planar parabola: $y(x) = ax^2 + bx + c$ (which has a point for every value of $x$) determine the coefficients $a$, $b$, and $c$, and hence, in total, seven numbers plus one bit determine a parabola in 3-space.

**Exercise 12.6:** Compute the curvature function of the quadratic space curve $q(t) = a + bt + ct^2$.

We may estimate the tangent vectors for a cubic spline by using the tangents of quadratic curves. Let $p_1, \dots, p_n$ be the points interpolated by the cubic segment curves $x_1, \dots, x_{n-1}$ with parameter limit values $t_1, \dots, t_{n-1}$. Consider the quadratic space curve $q_{i-1}(t)$ which satisfies $q_{i-1}(0) = p_{i-1}$, $q_{i-1}(t_{i-1}) = p_i$, and $q_{i-1}(t_{i+1} + t_i) = p_{i+1}$, for $2 \le i \le n-1$. We may then estimate the tangent vectors $m_2, \dots, m_{n-1}$ by $m_i = q'_{i-1}(t_{i-1})$. We may use the special choices $m_1 = q'_1(0)$ and $m_n = q'_{n-2}(t_{n-2} + t_{n-1})$. This tangent estimation method produces so-called Bessel tangents, and the corresponding cubic spline is called an Overhauser spline [Ove68].

Quadratic space curves may be "mixed" together to achieve an interpolating curve which passes through two points, $p_i$ and $p_{i+1}$, with the given tangent vectors, $m_i$ and $m_{i+1}$, respectively. One such mixture is: $x_i(t) = [p_i + m_i t + (p_{i+1} - p_i - m_i)t^2](1-t) + [p_{i+1} - m_{i+1}(1-t) + (p_i - p_{i+1} + m_{i+1})(1-t)^2]t$, for $0 \le t \le 1$. The mixture $x_i(t)$ coincides with the Hermite cubic polynomial which passes through $p_i$ with the tangent vector, $m_i$, and through $p_{i+1}$ with the tangent vector, $m_{i+1}$, and whose parameter range is $0 \le t \le 1$ between $p_i$ and $p_{i+1}$.

This same convex combination mixing device may also be employed with any other interpolating curves. Mixing the two circular arcs which fit $p_{i-1}, p_i, p_{i+1}$ and $p_i, p_{i+1}, p_{i+2}$, respectively is sometimes a useful way of computing an interpolatory curve in 3-space.

Another mixing of quadratics is to compute the interpolated point between $p_i$ and $p_{i+1}$ as a mixture of the quadratic space curve $q_{i-1}(t)$ which satisfies $q_{i-1}(0) = p_{i-1}$, $q_{i-1}(g_{i-1}) = p_i$, and $q_{i-1}(h_{i-1}) = p_{i+1}$, and the quadratic space curve $q_i(t)$ which satisfies $q_i(0) = p_i$, $q_i(g_i) = p_{i+1}$, and $q_i(h_i) = p_{i+2}$. The parameter limit values $g_1, \dots, g_{n-2}$ and $h_1, \dots, h_{n-2}$ are chosen to satisfy $0 < g_i < h_i$. This mixture is $x_i(t) = (1-t)q_{i-1}(g_{i-1} + (h_{i-1} - g_{i-1})t) + tq_i(g_i t)$ for $0 \le t \le 1$. Since this is a cubic form, it is identical with the Hermite cubic polynomial which passes through $p_i$ with the tangent vector $[h_{i-1} - g_{i-1}]q'_{i-1}(g_{i-1})$ and through $p_{i+1}$ with the tangent vector $g_i q'_i(g_i)$, whose parameter range is $0 \le t \le 1$ between $p_i$ and $p_{i+1}$. In effect, we have estimated the tangent vectors for the cubic spline segment $x_i$ by $m_i = [h_{i-1} - g_{i-1}]q'_{i-1}(g_{i-1})$ and $m_{i+1} = g_i q'_i(g_i)$. We may use the special choices $m_1 = q'_1(0)$ and $m_n = q'_{n-2}(h_{n-2})$. Let us call this cubic spline the *mixed quadratic cubic spline.*

**Exercise 12.7:** Show that the above mixed quadratic cubic spline that interpolates the points $p_1, \dots, p_n$ is a $G^1$ joined double tangent spline, and explain its relationship to an Overhauser spline.

**Exercise 12.8:** Suggest a method for choosing the parameter limit values $g_1, g_2, \dots, g_{n-2}$ and $h_1, \dots, h_{n-2}$ to be used in defining the quadratics $q_1, \dots, q_{n-2}$ to be used to compute the entry and exit tangents for the mixed quadratic cubic spline.

**Exercise 12.9:** Study the mixture of the quadratic space curves $q_{i-1}$ and $q_i$ defined as $f_0(t)q_{i-1}(g_{i-1} + (h_{i-1} - g_{i-1})t) + f_1(t)q_i(g_i t)$ for $0 \le t \le 1$ where the non-linear mixing coefficients are given as $f_0(t) = 1 - 3t^2 + 2t^3$ and $f_1(t) = 3t^2 - 2t^3$.

**Exercise 12.10:** Study the double tangent quadratic spline that interpolates the points $p_1, \dots, p_n$, where the quadratic segment curve $x_i$ is defined as the mixture of $q_{i-1}$ and $q_i$ as $x_i(t) = [q_{i-1}(g_{i-1} + (h_{i-1} - g_{i-1}t) + q_i(g_i t)]/2$. Can you modify this spline to obtain a related cubic $G^1$-joined spline?

**Exercise 12.11:** Devise weight values $\alpha$, $\beta$, and $\gamma$ for the cubic spline tangent estimation procedure: $m_i = \alpha q'_{i-2}(h_{i-2}) + \beta q'_{i-1}(g_{i-1}) + \gamma q'_i(0)$.

**Exercise 12.12:** Given the points $p_1, \dots, p_n$ with the associated knot values $r_0 < r_1 < \dots < r_{n-1}$, define $j_i(t)$ to be the cubic polynomial space curve that satisfies $j_i(r_{i-1}) = p_i$, $j_i(r_i) = p_{i+1}$, $j_i(r_{i+1}) = p_{i+2}$, and $j_i(r_{i+2}) = p_{i+3}$ for $1 \le i \le n-3$. Note $(j_i)_k$ is the Lagrange interpolating polynomial for the points $(r_{i-1}, p_{i,k})$, $(r_i, p_{i+1,k})$, $(r_{i+1}, p_{i+2,k})$, $(r_{i+2}, p_{i+3,k})$.

Define and study an interpolant piecewise-composed of segment space curves, where the $i$th segment curve $x_i(t)$ that connects the points $p_i$ and $p_{i+1}$ as $t$ ranges from $r_{i-1}$ to $r_i$ is constructed as a mixture of the polynomials $j_{i-2}$, $j_{i-1}$, and $j_i$. Resolve the definition of this interpolant at the boundaries of the sequence of data points $p_1, \dots, p_n$. *Hint*: first form a mixture of $j_{i-2}$ and $j_i$, and then mix this with $j_{i-1}$; the result will be a piecewise quintic.

# 13 Cubic Spline Vector Space Basis Functions

Finding a basis $\langle \phi_1, \ldots, \phi_d \rangle$ for a $d$-dimensional vector space of cubic splines allows us to express any element $x$ of the vector space as $x = \alpha_1\phi_1 + \ldots + \alpha_d\phi_d$ where $\alpha_1, \ldots, \alpha_d \in \mathcal{R}$. For the case of a vector space of cubic spline functions, some basis sets can be developed by focusing on a representation of the cubic polynomial spline segments as component-wise linear combinations of fixed functions.

A Hermite cubic polynomial, $x_i(t)$, with $0 \le t \le 1$, satisfies $x_i(0) = p_i$, $x_i'(0) = m_i$, $x_i(1) = p_{i+1}$, and $x_i'(1) = m_{i+1}$. The curve segment $x_i$ is based on four vectors: $p_i$, $m_i$, $p_{i+1}$, and $m_{i+1}$; and $x_i$ can be written as a linear combination of these four vectors: $x_i(t) = f_0(t)p_i + f_1(t)p_{i+1} + f_2(t)m_i + f_3(t)m_{i+1}$, where $0 \le t \le 1$, with $f_0(t) = 1 - 3t^2 + 2t^3$, $f_1(t) = 3t^2 - 2t^3$, $f_2(t) = t - 2t^2 + t^3$, and $f_3(t) = -t^2 + t^3$. These coefficient functions $f_0$, $f_1$, $f_2$, and $f_3$ are called the *blending* functions for the Hermite cubic polynomial interpolating between the points $p_i$ and $p_{i+1}$, with tangent vectors $m_i$ and $m_{i+1}$ respectively, with the parameter range [0,1]. If we use shifted forms of such blending functions to write each segment curve $x_i$ as a linear combination of (shifted) blending functions, we may obtain an overall representation of the corresponding spline $x$ as a linear combination of shifted blending functions. This will lead to an explicit representation of a basis for the space of Hermite cubic splines.

**Exercise 13.1:** Show that $f_0(t) + f_1(t) = 1$, $f_0(1-t) = f_1(t)$, and $f_3(t) = -f_2(1-t)$.

**Exercise 13.2:** Show that $f_0(t)p_i + f_1(t)p_{i+1} + f_2(t)m_i + f_3(t)m_{i+1} = f_0(1-t)p_{i+1} + f_1(1-t)p_i - f_2(1-t)m_{i+1} - f_3(1-t)m_i$.

**Solution 13.2:** The Hermite cubic spline segment connecting $p_i$ to $p_{i+1}$ with the tangent vector $m_i$ at $p_i$ and the tangent vector $m_{i+1}$ at $p_{i+1}$ and having the parameter $t$ in $[0, 1]$ is the same curve as the Hermite cubic spline segment going in the other direction, from $p_{i+1}$ to $p_i$ with the tangent vector $-m_{i+1}$ at $p_{i+1}$ and the tangent vector $-m_i$ at $p_i$.

**Exercise 13.3:** Consider the space $h^*$ of join order 1, range dimension 1 $h$-splines with respect to $r_0 < r_1 < \cdots < r_{n-1}$, where $r_i = i$ and where $h$ is the set of cubic polynomials which map $\mathcal{R}$ to $\mathcal{R}$. Show that $h^*$ is a set of single-valued functions from $\mathcal{R}$ to $\mathcal{R}$. Also show that, in this case, the blending functions: $f_0$, $f_1$, $f_2$, and $f_3$, are a basis for $h$ interpreted as a vector space but are *not* a basis for $h^*$ unless $n = 2$. Finally, show that $h^*$ is a vector space of dimension $2n$.

## 13.1 Bases for $C^1$ and $C^2$ Space Curve Cubic Splines

Consider the vector space $H^*_{(1)}$ of join order 1, range dimension 3 cubic polynomial splines with respect to the knot values $r_0, r_1, \ldots, r_{n-1}$ where $r_0 = 0$ and $r_j = t_1 + \cdots + t_j$ for arbitrary positive values of the segment parameter limits $t_1, \ldots, t_{n-1}$. The space $H^*_{(1)}$ is a vector space of dimension $6n$. A basis for this vector space called the Hermite blending function basis can be determined as follows.

Given $t_1, \ldots, t_{n-1}$, take $t_0 := 1$, $t_n := 1$ and define the *generalized Hermite blending functions*

$$\begin{aligned}
f_{i0}(t) &= 1 - 3(t/t_i)^2 + 2(t/t_i)^3, \\
f_{i1}(t) &= 3(t/t_i)^2 - 2(t/t_i)^3, \\
f_{i2}(t) &= t_i(t/t_i - 2(t/t_i)^2 + (t/t_i)^3), \quad \text{and} \\
f_{i3}(t) &= t_i(-(t/t_i)^2 + (t/t_i)^3),
\end{aligned}$$

for $0 \le i \le n$.

Now take $r_{-1} := -1$, $r_n := r_{n-1} + 1$, and define

$$F_{ij}(t) = \begin{cases} 0 & \text{for } t < r_{i-1} \\ f_{ij}(t - r_{i-1}) & \text{for } t \in [r_{i-1}, r_i) \\ 0 & \text{for } t \geq r_i \end{cases}$$

for $0 \leq j \leq 3$ and $0 \leq i \leq n$. Then an element $x$ of $H^*_{(1)}$ determined by the points $p_1, \ldots, p_n$ with the tangent vectors $m_1, \ldots, m_n$ can be written as

$$x(t) = \sum_{1 \leq i < n} \left[F_{i0}(t)p_i + F_{i1}(t)p_{i+1} + F_{i2}(t)m_i + F_{i3}(t)m_{i+1}\right].$$

Thus,

$$x(t) = \sum_{1 \leq i \leq n} \left[(F_{i-1,1}(t) + F_{i0}(t))p_i + (F_{i-1,3}(t) + F_{i2}(t))m_i\right].$$

Let $G_{i1}(t) := F_{i-1,1}(t) + F_{i0}(t)$ and $G_{i2}(t) := F_{i-1,3}(t) + F_{i2}(t)$ for $1 \leq i \leq n$. Note that $G_{i1}$ corresponds to 0, joined with $F_{i-1,1}$, joined with $F_{i0}$, joined with 0, to form a join order 1 piecewise cubic polynomial spline which is zero outside the interval $(r_{i-2}, r_i)$, and $G_{i2}$ corresponds to 0, joined with $F_{i-1,3}$, joined with $F_{i2}$, joined with 0 to form another join order 1 piecewise cubic polynomial spline which is zero outside the interval $(r_{i-2}, r_i)$.

Now,

$$\begin{aligned} x(t) \quad = \quad & \sum_{1 \leq i \leq n} [p_{i1}(G_{i1}(t), 0, 0) + p_{i2}(0, G_{i1}(t), 0) + p_{i3}(0, 0, G_{i1}(t)) \\ & \quad + m_{i1}(G_{i2}(t), 0, 0) + m_{i2}(0, G_{i2}(t), 0) + m_{i3}(0, 0, G_{i2}(t))], \end{aligned}$$

and so the $6n$ join order 1 range dimension 3 cubic polynomial spline space curves $(G_{i1}, 0, 0)$, $(0, G_{i1}, 0)$, $(0, 0, G_{i1})$, $(G_{i2}, 0, 0)$, $(0, G_{i2}, 0)$, and $(0, 0, G_{i2})$ for $1 \leq i \leq n$ form a basis for $H^*_{(1)}$.

This basis has been obtained by constructing shifted forms of the generalized Hermite blending functions, extending them to 3-tuples, and determining that these $6n$ 3-tuples are multiplied by the $6n$ "free-parameters" $p_{11}, p_{12}, p_{13}, \ldots, p_{n1}, p_{n2}, p_{n3}, m_{11}, m_{12}, m_{13}, \ldots, m_{n1}, m_{n2}, m_{n3}$ in the expression for a general cubic spline in $H^*_{(1)}$.

**Exercise 13.4:** Show that $G_{i1}$ and $G_{i2}$ are join order 1 piecewise cubic polynomials on $(-\infty, \infty)$. Note that there are three join points to be considered. Draw graphs of $G_{i1}$ and $G_{i2}$.

Note that $G_{i1}$ and $G_{i2}$ are 0 outside the interval $[r_{i-2}, r_i)$; this interval is called the *support* of $G_{i1}$ and $G_{i2}$. Since this support interval is as small as possible, we say that the functions $G_{i1}$ and $G_{i2}$ have *small support.*

Due to the small support property of the functions $G_{i1}$ and $G_{i2}$, the basis summation expression for computing a spline in $H^*_{(1)}$ involves many 0 terms which need not be explicitly computed. Avoiding computing these 0 terms is equivalent to computing a Hermite cubic spline by computing the individual cubic polynomial segment curves $x_1, \ldots, x_{n-1}$.

When the $3n$ components of the tangent vectors $m_1, \ldots, m_n$ are determined as linear functions of the $3n$ components of the interpolated points $p_1, \ldots, p_n$, the resulting vector space of cubic splines is a $3n$-dimensional subspace of $H^*_{(1)}$. In simple cases, where a non-singular $n \times n$ matrix $M$ specifies the relationship in the form of $n$ vector equations as

$$\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix} = M \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix},$$

then a basis for the corresponding subspace can be determined as the $3n$ vector components of

$$\begin{aligned}
&[(G_{11}, 0, 0), \ldots, (G_{n1}, 0, 0)] + [(G_{12}, 0, 0), \ldots, (G_{n2}, 0, 0)]M, \quad \text{and}\\
&[(0, G_{11}, 0), \ldots, (0, G_{n1}, 0)] + [(0, G_{12}, 0), \ldots, (0, G_{n2}, 0)]M, \quad \text{and}\\
&[(0, 0, G_{11}), \ldots, (0, 0, G_{n1})] + [(0, 0, G_{12}), \ldots, (0, 0, G_{n2})]M.
\end{aligned}$$

This basis thus consists of the $3n$ space curves:
$(G_{11} + [G_{12}, \ldots, G_{n2}][M \text{ col } 1],\ 0,\ 0), \ldots,$
$(G_{n1} + [G_{12}, \ldots, G_{n2}][M \text{ col } n],\ 0,\ 0)$, and
$(0,\ G_{11} + [G_{12}, \ldots, G_{n2}][M \text{ col } 1],\ 0), \ldots,$
$(0,\ G_{n1} + [G_{12}, \ldots, G_{n2}][M \text{ col } n],\ 0)$, and
$(0,\ 0,\ G_{11} + [G_{12}, \ldots, G_{n2}][M \text{ col } 1]), \ldots,$
$(0,\ 0,\ G_{n1} + [G_{12}, \ldots, G_{n2}][M \text{ col } n])$.

In particular, we can use this schema to produce a basis for any of the $3n$-dimensional subspaces consisting of the join order 2 range dimension 3 global natural splines, global quadratic end condition splines, global cyclic splines, global anti-cyclic splines, or global third derivative constrained splines.

For example, the $n \times n$ matrix $M$ that can be used to construct a basis for the global natural splines for $n$ knot values is $\tilde{D}^{-1}\tilde{C}$, where

$$\tilde{D} = \begin{bmatrix} 2 & 1 & & & \\ t_2 & 2(t_2+t_1) & t_1 & & \\ & t_3 & \ddots & & \\ & & t_{n-1} & 2(t_{n-1}+t_{n-2}) & t_{n-2} \\ & & & 1 & 2 \end{bmatrix}, \text{ and}$$

$$\tilde{C} = 3 \cdot \begin{bmatrix} \frac{-1}{t_1} & \frac{1}{t_1} & & \\ \frac{-t_2}{t_1} & \frac{t_2}{t_1} - \frac{t_1}{t_2} & \frac{t_1}{t_2} & \\ & & \ddots & \\ & \frac{-t_{n-1}}{t_{n-2}} & \frac{t_{n-1}}{t_{n-2}} - \frac{t_{n-2}}{t_{n-1}} & \frac{t_{n-2}}{t_{n-1}} \\ & & \frac{-1}{t_{n-1}} & \frac{1}{t_{n-1}} \end{bmatrix}.$$

**Exercise 13.5:** Show that the basis for the subspace of global natural splines constructed above consists of join order 2 functions (i.e., $C^2$ functions.)

**Exercise 13.6:** Use the above defined shifted generalized blending functions $F_{i0}(t)$, $F_{i1}(t)$, $F_{i2}(t)$, and $F_{i3}(t)$, each supported in the interval $[r_{i-1}, r_i)$ for $0 \leq i \leq n$, with $r_{-1} := -1$ and $r_n := r_{n-1} + 1$, to express the double tangent local cubic spline $x$ with the knot values $r_0, \ldots, r_{n-1}$ that interpolates the points $p_1, \ldots, p_n$ with the exit tangent vectors $m_1, \ldots, m_n$ and the entry tangent vectors $l_1, \ldots, l_n$ as

$$x(t) = \sum_{1 \leq i \leq n} (F_{i-1,1}(t) + F_{i0}(t))p_i + \sum_{1 \leq i \leq n} F_{i2}(t)m_i + \sum_{1 \leq i \leq n} F_{i-1,3}(t)l_i.$$

Then write the $9n - 6$ 3-tuples which specify the space curves that form a basis for the $(9n-6)$-dimensional space $S^*$ of double tangent join order 0 local cubic splines with range dimension 3.

Define the $n \times n$ matrices $N$ and $Q$ from the $n-2$ equations that relate the array of interpolation points $p$ to the array of exit tangent vectors $m$ for a $G^2$ joined global cubic spline, together with the two additional natural spline end condition equations $2m_1 + m_2 = 3(p_2 - p_1)/t_1$ and $m_{n-1} + 2m_n = 3(p_n - p_{n-1})/t_{n-1}$, such that $Np = Qm$. The elements of $N$ and $Q$ involve the parameter limit values $t_1, \ldots, t_{n-1}$, and the beta spline shape parameters $\beta_1, \ldots, \beta_{n-1}$ and $\gamma_1, \ldots, \gamma_{n-2}$. Now define the $n \times n$ diagonal matrix $L = diag(\beta_0, \ldots, \beta_{n-1})$; then the array of entry tangent vectors $l$ is defined by $l = L^{-1}m$.

Now write an element $x$ of $S^*$ in terms of the $n \times 3$ matrices $p$, $m$ and $l$ as

$$\begin{aligned} x(t) \quad & = [F_{01}(t) + F_{10}(t), \ldots , F_{n-1.1}(t) + F_{n0}(t)]p \\ & +[F_{12}(t), \ldots , F_{n-1.2}(t), 0]m + [0, F_{13}(t), \ldots , F_{n-1.3}(t)]l, \end{aligned}$$

and substitute $Q^{-1}Np$ for $m$ and $L^{-1}Q^{-1}Np$ for $l$ to obtain a basis for the $3n$-dimensional subspace of $G^2$ joined double tangent global cubic splines with the parameter limit values $t_1, \ldots , t_{n-1}$ corresponding to the knot values $r_0, \ldots , r_{n-1}$, and the beta spline shape parameters $\beta_1, \ldots , \beta_{n-1}$ and $\gamma_1, \ldots , \gamma_{n-2}$.

## 13.2 Cardinal Bases for Cubic Spline Vector Spaces

**Exercise 13.7:** Present a basis for the $3n$-dimensional subspace of join order 2, range dimension 3 clamped global cubic splines with $m_1 = u$ and $m_n = v$.

**Solution 13.7:** Let $C[p_1, \ldots , p_n](t)$ be the global clamped cubic spline with $m_1 = u$ and $m_n = v$ that interpolates $p_1, p_2, \ldots , p_n$. Then one basis for the global clamped cubic splines consists of the $3n$ spline space curves:

$$\begin{aligned} &C[(1,0,0),(0,0,0),\ldots,(0,0,0)],\ldots,C[(0,0,0),\ldots,(1,0,0)], \\ &C[(0,1,0),(0,0,0),\ldots,(0,0,0)],\ldots,C[(0,0,0),\ldots,(0,1,0)], \\ &C[(0,0,1),(0,0,0),\ldots,(0,0,0)],\ldots,C[(0,0,0),\ldots,(0,0,1)]. \end{aligned}$$

This basis is called the cardinal basis for the global clamped cubic splines.

Consider a vector space $V$ of splines of range dimension 1 with respect to the knot values $r_0 < r_1 < \ldots < r_{n-1}$, such that the points $(r_0, y_0), \ldots , (r_{n-1}, y_{n-1})$ are interpolated by a unique function $f_y(t)$ in $V$ for any choice of scalar values $y_0, \ldots , y_{n-1}$ for the components of the vector $y \in \mathcal{R}^n$. Let $e_i$ denote the $n$-vector whose $j$th component is $\delta_{ij}$. Then the function $f_{e_i}$ denotes that function in $V$ that interpolates the points $(r_0, 0),\ldots,(r_{i-2}, 0)$, $(r_{i-1}, 1)$,$(r_i, 0)$,$\ldots$,$(r_{n-1}, 0)$. The functions $f_{e_1}, f_{e_2}, \ldots , f_{e_n}$ are called the *cardinal functions* of $V$ and they form a basis for $V$ called the *cardinal basis* of $V$. Like all basis functions, $f_{e_i} \in V$, so $f_{e_i}$ must possess the same join order properties that all the functions in

$V$ possess, i.e., $f_{e_i}$ must be at least as smooth as the least smooth function in $V$. In order to explicitly state the cardinal functions $f_{e_1}, f_{e_2}, \ldots, f_{e_n}$, we must be able to compute the functions that solve the associated interpolation problems whose solutions then make up the cardinal basis.

**Exercise 13.8:** Take $V$ to be the space of piecewise linear functions with joints at the knot values $r_0 < r_1 < \ldots < r_{n-1}$. Show that the cardinal functions constituting a basis for these piecewise-linear range dimension 1 splines are defined for $i = 1, \ldots, n$ by the tent functions:

$$f_{e_i}(t) = l_i(t) = \begin{cases} 0 & \text{for } t < r_{i-2} \\ (t - r_{i-2})/(r_{i-1} - r_{i-2}) & \text{for } r_{i-2} \le t < r_{i-1} \\ (r_i - t)/(r_i - r_{i-1}) & \text{for } r_{i-1} \le t < r_i \\ 0 & \text{for } r_i \le t \end{cases}$$

with $r_{-1} := -1$ and $r_n := r_{n-1} + 1$.

Given the space of range dimension 1 splines, $V$, a corresponding space, $V^d$, of range dimension $d$ splines, can be formed by taking each of the $d$ component functions of an element of $V^d$ from $V$. The cardinal functions $f_{e_1}(t), \ldots, f_{e_n}(t)$ can then be used to express any function $z$ in $V^d$ as $z(t) = \alpha_1 f_{e_1}(t) + \ldots + \alpha_n f_{e_n}(t)$ where $\alpha_1, \ldots, \alpha_n$ are $d$-tuples. This is the cardinal function representation of the spline function $z \in V^d$. This representation leads directly to a basis for $V^d$. For example, when $d = 3$, the set of $3n$ functions mapping $\mathcal{R}$ to $\mathcal{R}^3$ given by $(f_{e_1}, 0, 0)$, $(0, f_{e_1}, 0)$, $(0, 0, f_{e_1})$, $\ldots$, $(f_{e_n}, 0, 0)$, $(0, f_{e_n}, 0)$, $(0, 0, f_{e_n})$ forms the cardinal basis for $V^3$, and a similar construction applies for any range dimension $d$.

Not every range dimension 1 interpolation scheme having an associated vector space $V$ of interpolation functions for the knot values $r_0, \ldots, r_{n-1}$ admits a set of cardinal functions; $V$ must have dimension $n$. If the interpolating function $f_y$ is not uniquely determined by the $n$-tuple $y$ for all $y \in \mathcal{R}^n$, then there are many choices for cardinal functions and a basis is not determined by the chosen functions. In such cases, however, there is always a dimension $n$ subspace of $V$ which has a cardinal function basis.

We can generalize by considering a vector space $V$ of splines of range dimension 1 with respect to the knot values $r_0 \le \ldots \le r_{n-1}$, such that there is a unique function $f_y(t) \in V$ so that, for the fixed sequence of integers $m_0 \ge 0, m_1 \ge 0, \ldots, m_{n-1} \ge 0$ and any choice of values $y_0, y_1, \ldots, y_{n-1}$ for the components of the vector $y \in \mathcal{R}^n$, $f_y^{(m_0)}(r_0) = y_0$, $f_y^{(m_1)}(r_1) = y_1, \ldots$, and $f_y^{(m_{n-1})}(r_{n-1}) = y_{n-1}$, where $f_y^{(m)}$ denotes the $m$th derivative of $f_y$ and $f_y^{(0)} := f_y$. Then the functions $f_{e_1}, \ldots, f_{e_n}$ are the cardinal functions of $V$ with respect to the derivative orders $\langle m_0, \ldots, m_{n-1}\rangle$, and $f_{e_1}, \ldots, f_{e_n}$ form a basis for $V$; in particular $f_y(t) = y_0 f_{e_1}(t) + \ldots + y_{n-1} f_{e_n}(t)$.

**Exercise 13.9:** Show that $f_{e_i}^{(m_j)}(r_j) = \delta_{i-1,j}$ for $1 \le i \le n$ and $0 \le j \le n-1$.

We have already obtained above the cardinal functions $G_{11}, G_{12}, \ldots, G_{n1}, G_{n2}$ for the vector space of join order 1 range dimension 1 cubic splines with respect to the knot values $r_0, r_0, r_1, r_1, \ldots, r_{n-1}, r_{n-1}$ and the derivative order sequence $\langle 0, 1, 0, 1, \ldots, 0, 1\rangle$, and used them to obtain the associated cardinal basis for the space $H^*_{(1)}$.

Given a set of data points, computing an interpolating spline in a vector space $V$ of range dimension 1 splines which admits a set of cardinal functions can be characterized as a *projection* operation onto $V$. Let $V$ be the space of range dimension 1 splines with respect to the knot values $r_0 \le \ldots \le r_{n-1}$, such that there is a unique function $f_y(t) \in V$ so that, for the fixed sequence of integers $m_0 \ge 0, m_1 \ge 0, \ldots, m_{n-1} \ge 0$ and any choice of values $y_0, y_1, \ldots, y_{n-1}$ for the components of the vector $y \in \mathcal{R}^n$, $f_y^{(m_0)}(r_0) = y_0$, $f_y^{(m_1)}(r_1) = y_1$, $\ldots$, and $f_y^{(m_{n-1})}(r_{n-1}) = y_{n-1}$. (Recall that the 0th derviative function $f_y^{(0)}$ is just $f_y$ itself.) Let $m := \max(m_0, \ldots, m_{n-1})$, and let $C_m$ be the class of $m$-times differentiable functions on $\mathcal{R}$. Note that $V \subset C_m$. Then we can define the projection operator $P_m$ such that every function $g \in C_m$ which satisfies $g^{(m_0)}(r_0) = y_0$, $g^{(m_1)}(r_1) = y_1$, $\ldots$, and $g^{(m_{n-1})}(r_{n-1}) = y_{n-1}$ is mapped to $f_y \in V$ by $P_m$, so that $gP_m = f_y$. That is, $gP_m$ satisfies $(gP_m)(t) = \sum_{i=0}^{n-1} f_{e_{i+1}}(t) g^{(m_i)}(r_i)$. Note that the restriction of $P_m$ to $V$, $P_m|V$, is the identity operator on $V$, and thus $P_m^2 = P_m$.

**Exercise 13.10:** Show that $P_m$ is a linear transformation on $C_m$.

**Exercise 13.11:** Suppose we have a spline function $f$ in an $n$-dimensional space of splines $V$ with the knot values $r_0, r_1, \ldots, r_{n-1}$ given in terms of a particular basis $\langle B_1, B_2, \ldots, B_n\rangle$ for the space $V$, so that $f(t) = c_1 B_1(t) + \cdots + c_n B_n(t)$; and suppose that $\langle G_1, G_2, \ldots, G_n\rangle$ is another basis for the space $V$. Explain how to represent $f$ with respect to the basis $\langle G_1, G_2, \ldots, G_n\rangle$, i.e., explain how to compute the coefficients $d_1, d_2, \ldots, d_n$ so that $f(t) = d_1 G_1(t) + \cdots + d_n G_n(t)$. *Hint*: consider the *generalized Vandermonde matrix*

$$\begin{bmatrix} G_1(r_0) & G_2(r_0) & \dots & G_n(r_0) \\ \vdots & . & \dots & \vdots \\ G_1(r_n) & G_2(r_n) & \dots & G_n(r_n) \end{bmatrix}.$$

## 13.3 The B-Spline Basis for Global Cubic Splines

The vector space $H^*_{(2)}$ of join order 2 range dimension 3 cubic polynomial splines without definite end conditions with respect to the knot values $r_0$, $\dots, r_{n-1}$ where $r_0 = 0$, and $r_j = t_1 + \cdots + t_j$ is a $(3n+6)$-dimensional subspace of $H^*_{(1)}$ where $t_1, \dots, t_{n-1}$ are arbitrary positive values. A famous basis for this subspace is the set of so-called B-spline basis functions [DeB78][Far90][BBB87].

The bases for the various spaces of global cubic splines presented above involve component functions that do not have small support intervals; thus the computation of a global cubic spline via its basis representation involves the summation of many terms. The functions that constitute the so-called B-spline basis do have small support; however the B-spline basis usually need not be used, since an effective method for computing a global cubic spline that interpolates the points $p_1, \dots, p_n$ consists of solving the appropriate linear system for the tangent vectors $m_1, \dots, m_n$ and then computing the individual segment curves $x_1, \dots, x_n$. (If the requirement for interpolation of the data points is suitably relaxed, a linear system need not be solved, and the direct use of the B-spline basis becomes attractive; the B-spline basis is particularly valuable in this context when we wish to recompute a global cubic spline curve after changing only one control point; then only the terms involving that point need be removed and recomputed).

The global cubic spline basis based on the particular join order 2 piecewise polynomials known as B-splines (B stands for "basis") is developed by starting with a simple basis for piecewise constant (degree 0) functions. We may then employ linear combination "mixing" similar to the mixing device used above in constructing cubic spline segments from quadratic spline segments. By mixing the degree 0 B-spline basis functions, we obtain the degree 1 B-spline basis functions, then we take linear combinations of these degree 1 functions to obtain degree 2 B-spline functions, and finally, the degree 3 B-spline basis functions are constructed as mixtures of the quadratic B-spline functions. The initial degree 0 B-spline functions have limited support, and this property is preserved for the higher degree B-spline functions.

Let us fix the sequence of knot values $r_0 < r_1 < \cdots < r_{n-1}$ where $r_0 = 0$ and $r_j = t_1 + t_2 + \cdots + t_j$. To construct the degree $k$ B-spline functions, we must add $k$ new knot values at each end of the interval $[r_0, r_{n-1}]$; these knot values $r_{-k}, r_{-(k-1)}, \ldots, r_{-1}$ and $r_n, r_{n+1}, \ldots, r_{n+k-1}$ can be arbitrarily chosen, subject to the constraint that $r_i < r_j$ whenever $i < j$. To be concrete, we may define $r_{-j} := r_0 - j$ and $r_{n-1+j} := r_{n-1} + j$ for $j \geq 1$.

Now define the degree 0 B-spline functions with respect to the knot values $\ldots, r_{-1}, r_0, \ldots, r_{n-1}, r_n, \ldots$ as

$$B_{i0}(t) = \begin{cases} 0 & \text{for } t < r_i \\ 1 & \text{for } t \in [r_i, r_{i+1}) \\ 0 & \text{for } t \geq r_{i+1}. \end{cases}$$

Then the following recursive linear combination mixing of the degree $k-1$ B-spline functions defines the degree $k$ B-spline functions:

$$B_{ik}(t) = \frac{t - r_i}{r_{i+k} - r_i} \cdot B_{i,k-1}(t) + \frac{r_{i+k+1} - t}{r_{i+k+1} - r_{i+1}} \cdot B_{i+1,k-1}(t),$$

for $k > 0$ and $i = \ldots, -1, 0, 1, \ldots, n-1, n, \ldots$, where we impose the convention that $v/0 = 0$.

In [BBB87], Bartels, Beatty and Barsky give a thorough step-by-step development of the B-spline functions. You can see from the above recursive definition that the function $B_{ik}$ is a spline composed of $k+1$ degree $k$ polynomial segments, each defined on an interval between two knot values, and supported on $k+1$ adjacent intervals between knot values; outside these $k+1$ intervals, $B_{ik}$ is 0.

**Exercise 13.12:** Explicitly write the degree 1 B-splines $B_{i1}$ for $i = \ldots, -3, \ldots, n-2, \ldots$; graph several of them, and show they have join order 0 (i.e., are continuous) at the knot values $r_0, \ldots, r_{n-1}$. Explain why $B_{ik}$ has join order $k-1$ at its knot values.

B-splines for uniformly spaced knot values have beautiful properties, and are richly interconnected with other mathematical topics, such as Whittaker's interpolation formula touched on previously. DeBoor [DeB87] and Chui [Chu88] discuss these topics in their various works.

The above recursive definition is useful for computing the value $B_{ik}(t)$. Note that computing $B_{i3}(t)$ involves computing values of $B_{i2}$ and $B_{i+1,2}$ which, in turn, involves computing values of $B_{i1}$, $B_{i+1,1}$, and $B_{i+2,1}$. It is possible to explicitly state the $k+1$ degree $k$ polynomial segment functions that comprise $B_{ik}$ [Chu88], but for $k = 3$, it is more convenient and efficient to use the recursive computation.

> **Exercise 13.13:** Explicitly compute the degree 3 B-splines $B_{i3}$ for $i = \ldots, -3, \ldots, n-2, \ldots$ (Each involves at most 5 cubic polynomial segments). Draw graphs of $B_{-33}$, $B_{-23}$, $B_{-13}$, $B_{03}$, and $B_{13}$, where we have $n = 3$ original knot values.

Since the degree $k$ B-spline function $B_{ik}$ is zero outside the support interval $[r_i, r_{i+k+1})$ and is positive in its interior, exactly $B_{ik}, B_{i-1,k}, \ldots, B_{i-k,k}$ are positive in $(r_i, r_{i+1})$. Also, $\sum_{i-k\leq j\leq i} B_{jk}(t) = 1$ for $t \in [r_i, r_{i+1}]$.

> **Exercise 13.14:** Show that $\sum_{-k\leq i<n+k} B_{ik}(t) = 1$ for $t \in [r_0, r_{n-1}]$.

The B-spline function $B_{ik}$ is $k-1$ times continuously differentiable, so, in particular, $B_{i0}$ is discontinuous, $B_{i1}$ is a join order 0 piecewise linear continuous function, and $B_{i3} \in H^*_{(2)}$ for $i = -3, -2, \ldots, n-1, n-2$.

The piecewise-polynomial B-spline functions $B_{ik}$ for $i = \ldots, -1, 0, 1, \ldots$, are linearly independent. In particular, the $n+2$ functions $B_{-3,3}, \ldots, B_{n-2,3}$ form a basis for the $(n+2)$-dimensional vector space of join order 2 range dimension 1 global cubic splines with the knot value sequence $r_0, r_1, \ldots, r_{n-1}$. Thus the $3n+6$ space curves $(B_{i3}, 0, 0)$, $(0, B_{i3}, 0)$, and $(0, 0, B_{i3})$ for $i = -3, \ldots, n-2$ form a basis called the B-spline basis for the subspace of global cubic spline space curves $H^*_{(2)}$. Some authors call any global cubic spline written with respect to the $B$-spline basis a *$B$-spline*.

A global cubic spline $x$ that interpolates the 3-space points $p_1, \ldots, p_n$ with the associated knot values $r_0, \ldots, r_{n-1}$ can thus be written in terms of 3-tuple coefficients $a_{-3}, \ldots, a_{n-2}$ as

$$x(t) = \sum_{-3\leq i\leq n-2} a_i B_{i3}(t) \text{ where } B_{i3} \text{ is 0 outside } [r_i, r_{i+4}).$$

In this formulation, the $B$-spline functions, $B_{i3}$, for $i = -3, \ldots, n-2$, act as blending functions that blend the vectors $a_{-3}, \ldots, a_{n-2}$.

The vector coefficient values $a_{-3}, \ldots, a_{n-2}$ can be determined from the $n$ vector equations

$$x(r_j) = \sum_{-3\leq i\leq n-2} a_i B_{i3}(r_j) = p_j,$$

for $j = 1, \ldots, n$, plus two more vector equations corresponding to the desired end conditions. The solvability of these equations demonstrates that the B-spline basis functions are indeed a basis. For example, for a natural global cubic spline with $x''(r_0) = 0$ and $x''(r_{n-1}) = 0$, these two additional equations are

$$\sum_{-3\le i\le n-2} a_i B''_{i3}(r_0) = 0 \quad \text{and} \quad \sum_{-3\le i\le n-2} a_i B''_{i3}(r_{n-1}) = 0.$$

Computing a global cubic spline $x(t)$ via its B-spline basis representation is often recommended because, like the de Casteljau algorithm for the Bézier form, the computation is numerically stable when we tabulate the B-spline basis functions using their recursive linear combination definitions.

Given the sequence of points $p_1, \ldots, p_n$, we may adjoin two additional points $p_0$ and $p_{n+1}$ at each end which serve, in essence, to determine the tangent vectors at the adjacent points $p_1$ and $p_n$. Then we may form the space curve

$$b(t) = \sum_{-3\le i\le n-2} p_{i+3} B_{i3}(t).$$

This space curve $b(t)$ is called the B-spline smoothing curve for the *control points* $p_0, p_1, \ldots, p_n, p_{n+1}$ with the knot values $r_{-3}, r_{-2}, \ldots, r_{n+2}$; the curve $b(t)$ passes close to the points $p_1, \ldots, p_n$, but, in general, does not interpolate them. Global cubic splines are often introduced as B-spline smoothing curves, and interpolation is disregarded in favor of approximation.

**Exercise 13.15:** Show that if two adjacent knot values, $r_i$ and $r_{i+1}$, among $r_1, \ldots, r_{n-2}$ are equal, then the cubic spline represented with the B-spline basis that interpolates the points $p_1, \ldots, p_n$ has only one continous derivative at $r_i$ when $p_i \ne p_{i+1}$. What happens if $r_i = r_{i+1} = r_{i+2}$? What happens if $r_i = r_{i+1} = r_{i+2} = r_{i+3}$?

**Exercise 13.16:** Show that $\{(1,0,0), (0,1,0), (0,0,1), (t,0,0), (0,t,0), (0,0,t), (t^2,0,0), (0,t^2,0), (0,0,t^2), (t^3,0,0), (0,t^3,0), (0,0,t^3)\} \cup \{((t-r_j)^3_+, 0, 0) \mid j = 1, \ldots, n-2\} \cup \{(0, (t-r_j)^3_+, 0) \mid j = 1, \ldots, n-2\} \cup \{(0,0,(t-r_j)^3_+) \mid j = 1, \ldots, n-2\}$ is a basis for $H^*_{(2)}$.

**Exercise 13.17:** Show that $1, t, t^2, t^3, |t - r_1|^3, \ldots, |t - r_{n-2}|^3$ is a basis for the vector space of join order 2, range dimension 1 global cubic splines with the knot value sequence $r_0 < r_1 < \cdots < r_{n-1}$.

Note we can use a basis representation of a natural global cubic spline to solve the problem of determining the 2D-smoothing spline for the points $(r_0, f_1), \ldots, (r_{n-1}, f_n)$ with the weights $\lambda_1, \ldots, \lambda_n$ and the smoothing parameter $\rho$ which minimizes the previously introduced functional $E(x) + \rho J(x)$. Let $\langle \phi_1, \ldots, \phi_n \rangle$ be a basis for the vector space of join order 2, range dimension 1 natural global cubic splines with the knot values $r_0, r_1, \ldots, r_{n-1}$. Write $x(t) = \alpha_1\phi_1(t) + \ldots + \alpha_n\phi_n(t)$. Then the minimizer $x(t)$ of $E(\cdot) + \rho J(\cdot)$ can be determined by solving for the coefficient values $\alpha_1, \ldots, \alpha_n$ that minimize $E(\alpha_1\phi_1 + \ldots \alpha_n\phi_n) + \rho J(\alpha_1\phi_1 + \ldots + \alpha_n\phi_n)$. When it is convenient, we can use a basis of $n + 2$ functions for the global cubic splines with undetermined end conditions by solving an under-determined minimization problem. Indeed this same device can be used with *any* finite-dimensional space of functions to obtain the minimizer of $E(\cdot) + \rho J(\cdot)$ in that space.

# 14
# Rational Cubic Splines

A cubic polynomial curve in the $xy$-plane, $(x_1(t), x_2(t))$, whose cubic term has a coefficient of 0 reduces to a parabola in this special case; but a cubic polynomial cannot represent other conic section curves such as a circular arc, an elliptic arc, or a segment of an hyperbola. It is an interesting fact, however, that an elliptic or hyperbolic arc in $\mathcal{R}^3$ can be parametrically represented by three component functions $c_1(t)$, $c_2(t)$, and $c_3(t)$, where each component function is a ratio of two quadratic polynomials.

Let $\Lambda$ be the class of space curves which are ratios of cubic polynomials in each component; this includes ratios of quadratic polynomials as a special case. Then the corresponding class of $\Lambda$-splines are called rational cubic splines. When the knot sequence being used may be non-uniformly spaced and when the cubic polynomials involved are expressed in the $B$-spline basis, then these rational cubic splines are often called NURB-splines (for Non-Uniform Rational B-splines).

Although rational cubic splines have additional degrees of freedom, they seem to have little practical added value over ordinary cubic splines, except for their ability to exactly produce conic section curves. In some cases this latter property is more easily achieved by using the desired conic section curve directly.

The reason that a space curve whose component functions are ratios of quadratic polynomials can be a conic section curve is because a parabola is specified parametrically by quadratic component functions (e.g., $(1, t, t^2)$), and the perspective projection of a parabola in $\mathcal{R}^3$ onto a suitable plane in $\mathcal{R}^3$ can yield any desired conic section curve $c$ in that projection plane. This is clear by taking the center of projection to be the vertex of a cone of which the parabola is a section; then the cone itself determines the mapping of the parabola section to another section of that cone in any chosen sectioning plane. A perspective projection maps a space curve with cubic or quadratic component fuctions to a space curve with a ratio of such polynomials as component functions. Thus rational cubic splines can match any desired conic section curve.

# 15
# Two Spline Programs

Two packages of C functions are given below. The first package is contained in a file entitled `gspline.c` and provides a routine called `cinterpolate` that can be used to compute a variety of different interpolating cubic spline curves (or their derivatives or integrals) which differ in the choice used of the tangent estimation method. The second package is contained in a file entitled `ssp.c` and provides a routine called `smoothspline` that can be used to compute an optimal cubic smoothing spline curve (or the associated derivative or integral) using a user specified or program estimated smoothing factor.

The two source code files given here can be downloaded from `www.birkhauser.com`.

## 15.1 Interpolating Cubic Splines Program

```
/* FILE gspline.c       REVISION DATE: April 7, 1997 */
/*---------------------------------------------------
This file contains the routine:
  (export)  cinterpolate(double *hp, double *vp,
                         double f, int16 k, double *mp,
                         int16 tansw, int16 idsw)
            which may be used to compute an
```

```
          interpolation curve or its integral or
          derivative in 3-space or in the plane,
          given a set of data points and, possibly,
          associated tangent vectors,
  and the ancillary routines:
  (private)  allocharray(int32 nr,int32 nc)
  (private)  dist(int16 a, double *hp)
  (private)  curve(double *hp, int16 j, double p,
                   double *dest, double f, int16 idsw)
-----------------------------------------------------*/
/* macros used to improve
readability */ #define int16 short int
#define int32 long int
#define EQ ==
#define NE  !=
#define AND &&
#define OR  ||
#define NOT !
#define XOR ^
#define private static
#define forward
#define import extern
#define export
#define MAX(x,y) ((x) > (y) ? (x) : (y) )
#define MIN(x,y) ((x) < (y) ? (x) : (y) )
#define ABS(x) (((x)<0)?-(x):(x))
/**********SYSTEM GLOBALS *****************************/
#include <Clib.h>
/*Clib.h defines the C-library fcts, e.g. sqrt, exp */
/*---------------------------------------------------*/
/********* FILE GLOBALS ******************************
 **(usable by all the functions in this file) *******/

/* macros for accessing the bounds of header-augmented
   arrays (harrays)*/
#define harrayrows(m)  ( *((int32 *)(m)) )
#define harraycols(m)  ( *((int32 *)(m)+1) )

/* macros to access linear arrays as 2-D matrices */
#define DNC(i,j) (((i)-1)*(nc)+(j))
#define DQ(i,j)  (((i)-1)*(q)+(j))

/* macros for computation */
#define Z(x) ((x)?x:1.0)
#define S(i,j) ((hp[DNC((i)+1,(j))] \
               -hp[DNC((i),(j))])/Z(dist((i),hp)))

/* forward declarations. */
forward private void
         curve(double *hp, int16 j, double p,
               double *dest, double f, int16 derivsw);

forward private double dist(int16 a, double *hp);

private double  *m;
   /* matrix of tangent vectors allocated in
      cinterpolate() if no matrix of tangent vectors
      is supplied */

private double  *pa;
   /* estimated arc length or user-given parameter
      values for the curve */

private int16
   nc,      /* number of columns in hp */
   q,       /* number of dimensions of data points */
   n,       /* used in curve() to index dest[] */
   prevj;   /* used in curve() to see if the cubic
               coefficients need to be recalculated */

/*====================================================*/
private double *allocharray(int32 nr,int32 nc)
/*----------------------------------------------------
allocharray() gets space for a matrix of doubles of
size nr by nc, and fills-in the 0th double in the
array with the row and column sizes and returns a
pointer to this space.
-----------------------------------------------------*/
{double *d;
 d = (double *)calloc(nr*nc+1,sizeof(double));
 /* install the row and column sizes in the zero
    element of the matrix */
 *((int32 *)d)=nr; *(((int32 *)d)+1)=nc;
 return(d);
}

/*====================================================*/
export double *cinterpolate(double *hp, double *vp,
                        double f, int16 k, double *mp,
                        int16 tansw, int16 idsw)
```

```
/*-----------------------------------------------------
  double  *hp;
hp is a matrix pointer.  The matrix hp[1:nh,1:nc]
contains data to be interpolated in sequential order.
The matrix hp is an harray (i.e.  the first 32 bits of
*hp contains the number of rows nh, and the second 32
bits contains the number of columns nc); The element
h[i,j] is accessed by writing h[DNC(i,j)], where
DNC(i,j) = (i-1)*nc + j. There are five different
forms the data can take:

(1) nc = 2 and k = 1: the data hp[] is a set of
    points in 2-space
(2) nc = 3 and k = 1: the data hp[] is a set of
    points in 3-space
(3) nc = 3 and k = 0: the data hp[] is a set of
    points in 2-space with a 3rd column containing
    parameter values for each point.
(4) nc = 4 and k = 0: the data hp[] is a set of
    points in 3-space with a 4th column containing
    parameter values for each point.
(5) nc = 2 and k = 0; the data hp[] is a set of
    points in 2-space from the graph of a
    2D-function with column 1 used as the
    associated parameter values.

For all cases it is assumed that the data is sorted by
parameter value, but only cases (3), (4) and (5)
contain the parameter values.  Cases (1) and (2) use
implicit estimated arc-length parameters.

  double  *vp;
vp is a matrix pointer to an harray vp[1:nv,1:1].  If
k=0, vp contains the parameter values for which a point is
desired, or if k=1, vp contains arc length values relative
to the length of the curve for which a point is desired.
That is, let the curve have total length 1, with the
relative arclength values being in between.

  double f;
f is a flatness parameter used to scale all tangent
vector magnitudes. Generally f should be 1 when case
(5) above holds.

  int16  k;
```

k determines what vp and hp represent. k = 0 means a range of explicit parameter values are given in hp[], and vp[] contains such parameter values in the same units. k = 1 means relative polygonal(straight line) arc length from 0 to 1 over the span of the data in hp[] is the parameterization, and vp[] contains relative polygonal arc length values.

```
  double  *mp;
```
mp is a matrix pointer to an harray mp[1:nh,1:q]. mp[] contains tangent vectors for each data point. This input is optional; if mp = NULL, the procedure will estimate the tangent vectors, according to tansw.

```
  int16 tansw;
```
If mp = NULL, tansw determines the method used to estimate the tangent vectors at each point. Otherwise, tansw is ignored.

-------------------------------------

If tansw=0, the tangent vectors (m[i,1),m[i,2),m[i,3)) at each point i, with 1<i<nh, are estimated using the following formula, when point(i-1) != point(i) != point(i+1).

```
m[i,j] = .5*[(dist(i)*s(i-1,j) + dist(i-1)*s(i,j))]
```

for 1<i<nh, where dist(i) is the distance between point i and the adjacent point i+1 of hp[] and s(i,j) is the jth component of the unit vector from point i to point i+1. The tangent vectors at the end points, m[1,*) and m[nh,*), are set to be the vectors dist(1)*s(1,*) and dist(nh-1)*s(nh-1,*), respectively. This is a form of the 'distance-weighted adjacent tangents method' of tangent-estimation.

-------------------------------------

If tansw=1, the tangent vectors (m[i,1),m[i,2),m[i,3)) at each point i, with 1<i<nh, are estimated using the following formulas.

```
m[1,j] = (hp[2,j]-hp[1,j]),
m[nh,j] = (hp[nh,j]-hp[nh-1,j]), and
m[i,j] = (hp[i+1,j]-hp[i-1,j])/2  for i = 2,...,nh-1.
```

This is a form of the 'chordal method' for tangent-estimation.

```
------------------------------------
If tansw=2, the tangent vectors (m[i,1),m[i,2),m[i,3))
at each point i, with 1<i<nh, are estimated using the
following formula.  Let g(i)=pa(i)-pa(i-1), let
h(i)=pa(i+1)-pa(i-1), and let d(i)=pa(i+1)-pa(i) for
1<i<nh where pa(i) is the given or computed parameter
value associated with the point i.  (Usually pa(1)=0.)

m[1,j] = -g(2)hp[3,j]/(h(2)d(2))+h(2)hp[2,j]/(g(2)d(2))
          -(h(2)+g(2))hp[1,j]/(h(2)g(2)),

m[nh,j] =-g(nh-1)hp[nh,j]/(h(nh-1)d(nh-1))
          +h(nh-1)hp[nh-1,j]/(g(nh-1)d(nh-1))
          -(h(nh-1)+g(-nh-1))hp[nh-2,j]/
             (h(nh-1)g(nh-1))
          +h(nh-1)hp[nh,j]/(h(nh-1)d(nh-1))
          -h(nh-1)hp[nh-1,j]/(g(nh-1)d(nh-1))
          +h(nh-1)hp[nh-2,j]/(h(nh-1)g(nh-1)), and

m[i,j] =  -g(i)hp[i+1,j]/(h(i)d(i))
           +(h(i)-2g(i))hp[i,j]/(g(i)d(i))
           -d(i)hp[i-1,j],       for i = 2,...,nh-1.

This is the 'Bessel-tangents method' for tangent
estimation.
-------------------------------------
If tansw=3, then for i=2,...,nh-1, if hp[i,j] is
greater than both hp[i-1,j] and hp[i+1,j] or hp[i,j]
is less than both hp[i-1,j] and hp[i+1,j], then m[i,j]
is set to 0. Otherwise, hp[i-1,j], hp[i,j], and
hp[i+1,j] are a monotonic triplet, and then the
tangent vector component m[i,j] is computed as:

m[i,j] = [(hp[i,j]-hp[i-1,j])/(pa(i)-pa(i-1))]*
         [(hp[i+1,j]-hp[i,j])/(pa(i+1)-pa(i))]*
         [(pa(i+1)-pa(i-1))/(hp[i+1,j]-hp[i-1,j])]

where pa(i) is the given or computed parameter value
associated with the point i.

The tangent vectors at the endpoints, m[1,*) and
m[nh,*), are set to the following:

m[1,j] = [(hp[2,j]-hp[1,j])/(pa(2)-pa(1))]^2 *
                ((hp[3,j]-hp[1,j])/(pa(3)-pa(1))
m[nh,j] = [(hp[nh,j]-hp[nh-1,j])/(pa(nh)-pa(nh-1))]^2
           *(hp[nh,j]-hp[n-2,j])/(pa(nh)-pa(n-2))

This is the 'Davis-Dowden method' for tangent
estimation.  It is particularly suitable for monotonic
data.
-----------------------------------
If tansw = 4, then the C^2 global natural cubic spline
tangent vectors are computed and used.
-----------------------------------
If tansw = 5, then the C^2 global clamped cubic spline
tangent vectors are computed and used, where:

m[1,j] = [(hp[2,j]-hp[1,j])/dist(1)

m[nh,j] = [(hp[nh,j]-hp[nh-1,j])/dist(nh-1).

Here dist(i) denotes the euclidean distance between
the points hp row i and hp row (i+1).
-----------------------------------

  int16 idsw;
idsw<0 when we want to compute the tangent vector
(i.e.  the derivative) of the spline curve. idsw>0
when we want to compute the integral of the spline
curve. idsw = 0 when we want to compute the spline
curve itself.
-------------------------------------------------------

This procedure takes as input an harray matrix hp[]
that represents a set of points in 2-space or 3-space,
sometimes accompanied by a set of associated parameter
values and also accompanied by a set of associated
tangent vectors mp[] or by a switch tansw that
specifies a tangent estimation method.  It also takes
an array vp[] of explicitly given parameter values (if
k=0) or of relative arc length values (if k=1).
Relative arc length is 0 at the first point of the
curve and 1 at the last point of the curve.  Below is
a table of the types of inputs accepted by this
routine.

(1) Points on a space curve with parameters. The
matrix hp[] has 4 columns; x , y  and z coordinates in
```

the first three columns, and parameter values for each point in the fourth column. In this case, k must be 0 and the array vp[] must contain parameter values for which coordinates on the curve are desired. The output is a three-column matrix dest[] that contains nv points corresponding to the parameter values in vp[].

(2) Points on an xy-planar curve with parameters. The matrix hp[] has 3 columns; x and y coordinates in the first two columns, and parameter values for each point in the third column. As in case 1, k must be 0 and the array vp[] must contain parameter values. The output is a 2-column matrix dest[] that contains nv points corresponding to the parameter values in vp[].

(3) Points on a space curve only. The matrix hp[] has 3 columns containing x , y and z coordinates. The points must be in the parameterized order for the routine to work. In this case, k must be 1 and the array vp[] must contain relative arc length values (the arc length value is 0 at the first point of the curve and 1 at the last point). The output is a 3-column matrix dest[] that contains nv points corresponding to the arc-length values in vp[]

(4) Points on an xy-plane curve. The matrix hp[] has 2 columns containing x and y coordinates. As in case (3), the points must be in correct order for the routine to work. k must also be 1 and the array vp[] must contain relative arc length values. The output is a 2-column matrix dest[] that contains nv points corresponding to the arc length values in vp[].

(5) Points on a two-dimensional function y=f(x). This is similar to case (2), except that the parameter values are identical to the x values of the input points. As in case (2), k must be 0 and the array vp[] must contain x values for which y values are desired. The output is a 2-column matrix dest[] that contains nv points having the x values from vp[] and y values corresponding to these x values.

The input also includes a flatness parameter, f, that determines how much the interpolation curve follows

```
the straight lines between the points.  If f = 0, the
interpolation curve is merely a set of straight line
segments connecting the points. As f increases the
curve becomes "looser."

If no points are input, all output coordinates are 0.
If the input is one point with no tangent vector, the
curve that the interpolated points will be on is the
line defined by the vector consisting of the single
input point.  If the input is one point with a tangent
vector or 2 points without tangent vectors, that is
enough to define a line to interpolate points on.  All
other cases, including the case of two points and no
tangent vectors, are handled by the general algorithm.

The result is a pointer to an allocated harray
dest[1:nv,1:q].  dest[] is a matrix in which
coordinates of points matching parameter values or
arc length values given in vp[1:nv] are returned.
These will be spline function values if idsw=0,
derivative values if idsw<0, and integral values if
idsw>0.  The global value q=nc if we have a function,
and q=nc-1+k if we have a curve.  Thus q is the number
of physical dimensions of data points.
-----------------------------------------------------*/
{double temp,ti,ti1,ti2,
        vpval,
        *dest,    /* the result harray pointer */
        d1,d2,    /* used to calculate m */
        a;        /* the estimated arc length of the
                     entire curve */

 double *ra,     /* temporary working arrays for    */
        *ta,     /* back-substitution, parameter-   */
        *da;     /* limits and diagonal values      */

 int16  nh,       /* number of rows in hp */
        nv,       /* number of values in vp */
        funcsw,   /* 1 if we have 2-column functional
                     data, else 0 */
        v,        /* indexing offset */
        i,j;      /* loop control variables */

 nh=harrayrows(hp); /* nh = number of rows in hp    */
 nc=harraycols(hp); /* nc = number of columns in hp */
 nv=harrayrows(vp); /* nv = number of rows in vp    */

 /*funcsw=1 means: hp col 1=the explicit parm. vals.
   in place of hp col 3 */
 funcsw=(k EQ 0 AND nc EQ 2);
 q=(funcsw)?nc:(nc-1+k); /* q = physical dimension
                            of the data points */

 /* allocate the result harray, dest. */
 dest=allocharray(nv,q);
 /* If no points are given as input, set all
    coordinates to 0 */
 if (nh EQ 0) {clearharray(dest); return(dest);}

 /* If the input is one point without a tangent
    vector, use the point itself as the tangent vector
    and interpolate on the line thus determined.  If
    the input is one point with a tangent vector,
    interpolate on the defined line */
 if (nh EQ 1)
    {if (funcsw) a=hp[1]; else a=(k EQ 0)?hp[nc]:0.0;
     for (i=1; i<=nv; i++) for (j=1; j<=q; j++)
        {if (mp EQ NULL) temp=hp[j]; else temp=mp[j];
         if (idsw <0) d1=temp;
         else {d1=hp[j]+(vp[i]-a)*f*temp;
               if(idsw>0) d1=(vp[i]-a)*(d1+hp[j])/2.;}
         dest[DQ(i,j)]=d1;
        }
     return(dest);
    }

/* Allocate and load the parameter vector pa[1:nh] */
 pa=allocharray(nh,1);
/* If k=1, use estimated arc length as the
   parameter-value */
 if (k EQ 1)
    {/* calculate polygonal 'arc length' values */
     pa[1]=0.0;
     for (a=0.0,i=2; i<=nh; i++)
         pa[i]=(a+=dist(i-1,hp));
     for (a=Z(a),i=2; i<=nh; i++) pa[i]/=a;
    }
 else
/* The explicit parm. vals. are in hp col j in
   increasing order. */
```

```
    {j=(funcsw)?1:(q+1);
     for (i=1; i<=nh; i++) pa[i]=hp[DNC(i,j)];}

/* If mp is not given, compute the estimated tangent
   vectors; otherwise take m[] as the given input
   matrix mp[]. */
 if (mp != NULL) {m=mp; goto fixslopes;}

 m=allocharray(nh,q); /* allocate m[1:nh,1:q] */

 switch (tansw) /* compute tangent-vector estimates */
 {case 4:/* tansw=4: natural C^2 spline tangents */
  /* Compute m[1:nh,1:q] = global C^2 tangent vectors
     for a natural global cubic spline, using the nh
     data points in hp[1:nh,1:q] and the associated
     parameter values in pa[1:nh] to determine the
     component slopes m[1:nh,1:q]. Here 'natural'
     means we specify the second-derivative vectors
     at the endpoints, hp[1] and hp[n], to be 0.
     (hp[j,k] is accessed as hp[DNC(j,k)].)  */

  ra = allocharray(nh,1); /* allocate space for the
                             working space array ra.*/
  da = allocharray(nh,1); /* allocate space for the
                             working space array da.*/
  ta = allocharray(nh,1); /* allocate space for the
                             working space array ta.*/

  da[1]=2; ti=1; ta[nh]=1;

  for (i=1; i<nh; i++)
    {ra[i]=ti/da[i];
     ti=pa[i+1]-pa[i]; ta[i]=ti;
     if ((i+1) EQ nh) ti1=1;
     else ti1=pa[i+2]-pa[i+1];
     da[i+1]=2*(ti1+ti)-ti1*ra[i];
    }

  for (j=1; j<=q;j++)
    {m[DQ(1,j)]=3*(hp[DNC(2,j)]-hp[DNC(1,j)])/ta[1];

     for (i=1; i<nh; i++)
       {/* elimination-loop:
           convert to upper triangular form */
        m[DQ(i,j)]/=da[i];
        if ((i+1) EQ nh) {v=1; ti1=1;}
        else {v=2; ti1=ta[i+1];}
        temp=ti1/ta[i];
        m[DQ(i,j)]=3*(hp[DNC(i+v,j)]/temp
                    +(temp-1/temp)*hp[DNC(i+1,j)]
                    -temp*hp[DNC(i,j)])
                    -ti1*m[DNC(i,j)];
      }

    m[DQ(nh,j)]/=2-ra[nh-1];

    for (i=nh-1; i>=1; i--) /* back-substitution-loop*/
    m[DQ(i,j)] -= m[DQ(i+1,j)]*ra[i];
   }

 free(ra); free(da); free(ta);
 break;


 case 5:/* tansw=5: clamped C^2 spline tangents */
 /* Compute m[1:nh,1:q] = global C^2 tangent vectors
    for a clamped global cubic spline, using the nh
    data points in hp[1:nh,1:q] and the associated
    parameter values in pa[1:nh] to determine the
    component slopes m[1:nh,1:q]. Here 'clamped'
    means we specify the tangent vectors at the
    endpoints, hp[1] and hp[n], explicitly as the
    unit vectors in the directions (hp row 2)-
    (hp row 1) and (hp row nh)-(hp row (nh-1)).
    (mp is ignored.)  Thus there are n-2 unknown
     vectors to find. (hp[j,k] is accessed as
    hp[DNC(j,k)].)   */

 ra = allocharray(nh,1); /* allocate space for the
                            working space array ra.*/
 for (j=1; j<=q;j++)
     {m[DQ(1,j)]=S(1,j); m[DQ(nh,j)]=S(nh-1,j);}

 ra[1]=1.0;
 ti1=pa[2]; ti2=0;
 for (i=2; i<nh; i++)
   {/* elimination-loop:
       convert to upper triangular form */
    ti=pa[i+1]-pa[i]; temp=ti/ti1;
    for (j=1; j<=q; j++)
```

```
       m[DQ(i,j)]=3*(hp[DNC(i+1,j)]/temp
                    +(temp-1/temp)*hp[DNC(i,j)]
                    -temp*hp[DNC(i-1,j)])
                    -(ti/ra[i-1])*m[DNC(i-1,j)];

    ra[i]=2*(ti+ti1)-(ti/ra[i-1])*ti2;
    ti2=ti1; ti1=ti;
   }

 for (i=nh-1; i>1; i--)  /* back-substitution-loop */
    {ti1=pa[i]-pa[i-1];
     for (j=1; j<=q; j++)
       m[DQ(i,j)]=(m[DQ(i,j)]-m[DQ(i+1,j)]*ti1)/ra[i];
    }

 free(ra);
 break;

 /*tansw=0: compute distance-weighted ave. tangents */
 case 0:
 for (i=2; i<nh; i++) /*compute tan. vectors 2:nh-1 */
     {/* note: if point[i-1] != point[i] = point[i+1]
         then tangent[i] is in the direction S[i-1].
         If point[i-1] = point[i] != point[i+1], then
         tangent[i] is in the direction S[i].
         If point[i-1] = point[i] = point[i+1] then
         tangent[i]=S[i](= 0). */
      d1=dist(i-1,hp); d2=dist(i,hp);
      for (j=1; j<=q; j++)
          m[DQ(i,j)] =.5*(d2*S(i-1,j)+d1*S(i,j));
     }

 /* Go assign the first and last tangent vectors
    m[1] and m[nh]. */
 goto setends;

 case 1: /* tansw=1: compute chordal tangents */
 for (i=2; i<nh; i++)
 /* compute m[1:nh,1:q]=chordal tangent vectors */
 for (j=1; j<=q; j++)
     m[DQ(i,j)]=.5*(hp[DNC(i+1,j)]-hp[DNC(i-1,j)]);

setends:
 /* Assign the first and last tangent vectors
    m[1] and m[nh] */
```

```
for (j=1; j<=q; j++)
    {/* compute m[1,j] */
     m[j]=(hp[DNC(2,j)]-hp[j]);
     /* compute m[nh,j] */
     m[DQ(nh,j)]=(hp[DNC(nh,j)]-hp[DNC(nh-1,j)]);
    }
break;

case 2: /* tansw=2: compute Bessel tangents */
for (i=2; i<nh; i++)
    {/* compute m[1:nh,1:q]=Bessel tangent vectors */
     ti=pa[i]-pa[i-1];
     ti1=pa[i+1]-pa[i];
     ti2=pa[i+1]-pa[i-1];
     for (j=1; j<=q; j++)
        m[DQ(i,j)]=(ti2-2*ti)*hp[DNC(i,j)]/
                    (ti*ti1)-ti1*hp[DNC(i-1,j)]
                    -ti*hp[DNC(i+1,j)]/(ti1*ti2);
    }

/* Assign the first and last tangent vectors
   m[1] and m[nh]. */
ti=pa[2]-pa[1]; ti1=pa[3]-pa[2]; ti2=pa[3]-pa[1];
for (j=1; j<=q; j++)
    m[j] = -ti*hp[DNC(3,j)]/(ti2*ti1)
           +ti2*hp[DNC(2,j)]/(ti*ti1)
           -(ti2+ti)*hp[DNC(1,j)]/(ti2*ti);

ti=pa[nh-1]-pa[nh-2];
ti1=pa[nh]-pa[nh-1];
ti2=pa[nh]-pa[nh-2];
for (j=1; j<=q; j++)
    m[nh,j] = -ti*hp[DNC(nh,j)]/(ti2*ti1)
              +ti2*hp[DNC(nh-1,j)]/(ti*ti1)
              -(ti2+ti)*hp[DNC(nh-2,j)]/(ti2*ti)
              +ti2*(hp[DNC(nh,j)]/(ti1*ti2)
              -hp[DNC(nh-1,j)]/(ti*ti1)
              +hp[DNC(nh-2,j)]/(ti*ti2));
break;

case 3: /* tansw=3: compute Davis-Dowden tangents */
/* set tangents at relative maxima and relative
   minima to 0 */
for (i=2; i<nh; i++) for (j=1; j<=q; j++)
    {d1=hp[DNC(i,j)]-hp[DNC(i-1,j)];
```

```
      d2=hp[DNC(i+1,j)]-hp[DNC(i,j)];
      if (SIGN(d1) != SIGN(d2)) m[DQ(i,j)]=0.0;
      else m[DQ(i,j)]=(d1/(pa[i]-pa[i-1]))*
                      (d2/(pa[i+1]-pa[i]))*
                      ((pa[i+1]-pa[i-1])/(d1+d2));
     }

  /* Compute the tangent vectors at the endpoints
     m[1,*] and m[nh,*]. */
  for (j=1; j<=q; j++)
     {temp=(hp[DNC(2,j)]-hp[DNC(1,j)])/(pa[2]-pa[1]);
      m[DQ(1,j)]=temp*temp*
                 ((hp[DNC(3,j)]-hp[DNC(1,j)])/
                 (pa[3]-pa[1]));
      temp=(hp[DNC(nh,j)]-hp[DNC(nh-1,j)])/
           (pa[nh]-pa[nh-1]);
      m[nh,j]=temp*temp*
              (hp[DNC(nh,j)]-hp[DNC(nh-2,j)])/
              (pa[nh]-pa[nh-2]);
     }
 } /* end switch(tansw) */


fixslopes:
 /* If we are interpolating a function, scale each
    tangent vector m[i] to be of the form
    (1,slopeval), except if any m[i,1] = 0, we will
    produce (1,0) as a result.*/
 if (funcsw)
    for (i=1; i<=nh; i++)
       {if (m[DQ(i,1)] != 0.0) m[DQ(i,2)]/=m[DQ(i,1)];
        else m[DQ(i,2)]=0.0;
        /* Note we divide out the flatness factor so
           that this will be restored to 1 when we
           multiply by f in curve(). */
        m[DQ(i,1)]=1.0/f;
       }

 /* Now compute the output array dest[1:nv,1:q]. */

 n=1; /* start by setting n to row 1 of dest[] */

/* Setting prevj=-1 will cause us to calculate cubic
   coefficients in curve() the first time through */
 prevj=-1;
```

```
/* vp[] contains a list of parameter values (k=0), or
   a list of relative arc length values(k=1). Take
   vp[] and use curve() to find the coordinates
   associated with each vp[]-value and place them in
   the output array dest[].  That is, for each
   parameter in vp[], find which two points it is
   between, or if it is before the 1st or beyond the
   last point, and then call curve(). */
 for (j=i=1; i<=nv; i++)
    {/* loop for each parameter-value in vp[] */
     vpval=vp[i];
     while ((j>1) AND (vpval<pa[j])) j--;
     while ((j<nh-1) AND (vpval>=pa[j+1])) j++;
     /* Now pa[j] <= vpval < pa[j+1], or
        j=1 and vpval < pa[1], or j=nh-1 and
        vpval>=pa[nh-1]. */
     curve(hp,j,vpval-pa[j],dest,f,idsw);
    }

/* If we have functional data, place the parameter
   values vp[1:nv] in dest col 1 */
 if (funcsw EQ 1)
    for (j=1; j<=nv; j++) dest[2*j-1]=vp[j];

 free(pa); if (mp EQ NULL) free(m);
 return(dest);
}

/*=====================================================*/
private double dist(int16 a, double *hp)
/*------------------------------------------------------
 dist(a,hp) returns the distance between points
 hp row a and hp row (a+1).
-----------------------------------------------------*/
{int16 i;
 double t,temp = 0.0;
 for (i=1; i<=q; i++)
     {t=hp[DNC(a,i)]-hp[DNC(a+1,i)]; temp+=t*t;}
 return(sqrt(temp));
}

/*=====================================================*/
private void curve(double *hp, int16 j, double p,
                   double *dest, double f, int16 idsw)
```

```
/*----------------------------------------------------
   double *hp; matrix of points on curve.
   int16 j;    index of the data point defining the
               curve segment to be used for p.
   double *dest;    matrix to place coordinates
                    of parameter values and corres-
                    ponding interpolated values in.
   double p;   parameter value at which to
               interpolate coordinates for.
   double f;   flatness parameter that modifies
               all tangent vector magnitudes.
   int16 idsw; idsw<0 when we want to compute the
               tangent vector (i.e. the derivative)
               of the spline curve. idsw>0 when we
               want to compute the integral of the
               spline curve. idsw = 0 when we want
               to compute the spline curve itself.
------------------------------------------------------
This procedure takes a scalar parameter value p
and returns in dest[] the coordinates of the
corresponding point on the spline curve (if
idsw=0) or on the derivative (if idsw<0) or on the
integral (if idsw>0).  We use cubic polynomial
space splines. Given points hp[1,*],...,hp[nh,*]
and corresponding tangent vectors
m[1,*],...,m[nh,*], the curve between the two
points i and i+1 is interpolated with the Hermite
cubic polynomial space curve with vector coefficients
ai, bi, ci, and di, parametrically defined by

xi(p) = ai + bi*p + ci*p*p + di*p*p*p
where     ai = hp[i,*)
          bi = m[i,*)
          ci = 3 * (hp[i+1,*]-hp[i,*])/
               (t*t) - (2*m[i,*]+m[i-1,*])/t
  di = 2 * (hp[i,*]-hp[i+1,*])/(t*t*t)
               + (m[i+1)+m[i,*])/(t*t),
    and t is (parameter value of point (i+1))
              - (parameter value of point t).

 The flatness parameter, f, is also used herein.
 Each tangent vector m[i,*] is multiplied by f.
-----------------------------------------------------*/
{int16  i,k;
 double t,sv,r;
```

```
static double ival[4];  /* previous integral values */
static double a[4],b[4],c[4],d[4];  /* coefficients
                                   of cubic polynomial */

/* Note for 1<j<nh-1, the modified parameter value
   p+pa[j] lies in [pa[j],pa[j+1]], and if j=1,
   p+pa[j]<pa[2], and if j=nh-1, p+pa[j]>=pa[nh-1].

   When we are to compute an integral value, we need
   to accumulate the integral values for every spline
   segment 1,2,...,j-1 correponding to the intervals
   [pa[1],pa[2]],...,[pa[j-1],pa[j]]. If prevj != -1,
   the values ival[1:3] holds the component integral
   values over the interval pa[1] to pa[prevj].
*/
k = j; /* p+pa[j] ''belongs'' to the cubic for
          [pa[j],pa[j+1]]. if idsw<=0,
          we keep k=j.  otherwise, we will reset k
          appropriately so that we add the integral
          from pa[k] to pa[j] to ival[1:3]. */
if (idsw > 0)
   /* If this is the first entry, or if we are given
      a "back" parameter value, then we will recompute
      the prior integral starting from pa[1].*/
   {if (j < prevj) prevj=-1;
    if (prevj EQ -1)
       {ival[1]=ival[2]=ival[3]=0.; k=1;}
    else k=prevj;
   }

if (prevj EQ j) goto calc; /*don't compute
                         coefficients unless needed */
prevj = j;
/* compute currently-needed spline coefficients,
   and if k<j, compute any additional integral parts
   needed first.*/
for (; k<=j; k++)
    {/* set t to the parameter or relative
        arc length difference value
        covering the curve part to be used.
        The parameter value p to be
        interpolated at is in [0,t]. */
     t = pa[k+1]-pa[k]; r=t;
     if (t EQ 0.0) t = 1.0; /* avoid zero divide */
     for (i=1; i<=q; i++) /*calculate coefficients */
```

```
          {a[i] = hp[DNC(k,i)];
           b[i] = f*m[DQ(k,i)];
           c[i] = (3.*(hp[DNC(k+1,i)]-a[i])/t
                  -(2.*b[i]+f*m[DQ(k+1,i)]))/t;
           d[i] = (2.*(a[i]-hp[DNC(k+1,i)])/t
                  + b[i]+f*m[DQ(k+1,i)])/(t*t);

           if (k < j) /* If k<j, then idsw > 0! */
              ival[i] += (a[i]+(b[i]/2
                         +(c[i]/3+(d[i]/4)*r)*r)*r)*r;
          }
     }

 calc:/* calculate coordinates of spline pt. at p */
 for (i=1; i<=q; i++)
    {if (idsw > 0)
        sv = ival[i] +(a[i] +(b[i]/2
             +(c[i]/3 +(d[i]/4)*p)*p)*p)*p;
     else if (idsw<0) sv = (b[i]+(2*c[i] +3*d[i]*p)*p);
     else sv = (a[i]+(b[i]+(c[i]+d[i]*p)*p)*p);
     dest[DQ(n,i)] = sv;
    }

 /* For functions, the x-parametric part is a straight
    line, with all its estimated slope entries (in
    dest col 1) equal to 1/f; often we want to replace
    these 1/f values with the given x values found in
    vp when we are computing the derivative of a
    spline model for functional data. [This
    replacement is done in cinterpolate()!]

    When f EQ 1 then dest[n,1] = hp[j,1]+p.  Note f
    should be 1 for the x-curve when a function is
    being interpolated, and placing v[1:nv] into
    dest col 1 and scaling mp col 2 by 1/f prior
    to entering this routine enforces this. */

 n++;    /* advance to next output row of dest[] */
}
/*=====================================================*/
/* end of file gspline.c */
```

## 15.2 Optimal Smoothing Spline Program

```
/* FILE ssp.c            REVISION DATE: May 7, 1997 */
/*------------------------------------------------------
This file contains the routine:
 (export) smoothspline(double *hp,double *L,
                      double r,double *vp,int16 idsw)
          which may be used to compute an optimal
          2D-smoothing spline or its integral or
          derivative, with the smoothing parameter
          specified or estimated,
 and the ancillary routines:
 (private)  allocharray(int32 nr, int32 nc)
 (private)  ox(int16 i,int16 j,int16 h, int16 w)
 (private)  bandsolve(double *A, double *b)
 (private)  dnu(double r)
-----------------------------------------------------*/
/* macros used to improve readability */
#define int16 short int
#define int32 long int
#define EQ ==
#define NE  !=
#define AND &&
#define OR  ||
#define NOT !
#define XOR ^
#define private static
#define forward
#define import extern
#define export
#define MAX(x,y) ((x) > (y) ? (x) : (y) )
#define MIN(x,y) ((x) < (y) ? (x) : (y) )
#define ABS(x) (((x)<0)?-(x):(x))
/************SYSTEM GLOBALS ***********************/
#include <Clib.h> /*Clib.h defines the C-library
                    functions, e.g. sqrt, exp */

import double *cinterpolate(double *hp, double *vp,
                            double f, int16 k,
                            double *mp, int16 tansw,
                            int16 idsw);

/* See the MLAB manual for details on mmean */
import double *mmean(double *M, int32 x, int32 y,
                     int32 z, double *f);
```

```
/* See the MLAB manual for details on mmedian */
import double *mmedian(double *M, int32 x, int32 y,
                      int32 z, double *f);
/*-----------------------------------------------*/
/*************FILE GLOBALS *************************
 ** (usable by all the functions in this file) ******/

/* macros for harrays */
#define harrayrows(m)  ( *((int32 *)(m)) )
#define harraycols(m)  ( *((int32 *)(m)+1) )

/* macros to access 1-dimensional matrices as
   multi-dimensional matrices */
#define DNC(i,j) (((i)-1)*(nc)+(j))
#define DQ(i,j)  (((i)-1)*(q)+(j))

/* band-matrix access macros */
#define GbarM(i,j) Gbar[ox(i,j,1,n2)]
#define HM(i,j) H[ox(i,j,2,n)]
#define lhtM(i,j) lht[ox(i,j,2,n)]
#define qfM(i,j) qf[ox(i,j,2,n2)]
#define AM(i,j) A[ox(i,j,2,n2)]
#define ZM(i,j) Z[ox(i,j,2,n)]

/* Gbar, qf, fd, lht, hf are all file globals computed
   in smoothspline() and used in dnu(). */
private double *Gbar;
private double *qf;
private double *fd;
private double *lht;
private double *hf;
private double *A;

forward private int16   ox(int16 i,int16 j,int16 h,
                           int16 w)
forward private double *bandsolve(double *A, double *b)
forward private double  dnu(double r)

/*===============================================*/
private double *allocharray(int32 nr,int32 nc)
/*-----------------------------------------------
allocharray() gets space for a matrix of doubles
of size nr by nc, fills in 0th double in the array
with the row and column sizes and returns a
```

```
pointer to this space.
----------------------------------------------------*/
{double *d;
 d = (double *)calloc(nr*nc+1,sizeof(double));
 /* install the row and column sizes in the 0th
    element of the matrix */
 *((int32 *)d)=nr; *(((int32 *)d)+1)=nc;
 return(d);
}

/*==================================================*/
export double *smoothspline(double *hp, double *L,
                            double r, double *vp,
                            int16 idsw)
/*--------------------------------------------------
double *hp,
hp[1:n,1:2] is a 2-column matrix.  Each row of hp
represents a data point in the xy plane, and the
two columns hold the x  and y coordinates of the
data points for which we want to compute the
interpolation points, p, for the optimal smoothing
spline curve.  The rows of the matrix hp[] are assumed to
be sorted in increasing order on column 1.

double *L,
L[1:n,1:1] is a 1-column harray of weight values
for the data points in hp.  These weights are
taken as the reciprocal variances for the data
points in hp[].  If L=NULL, then L[] is taken to
be an n-vector of 1's.

double r,
r is the smoothness factor. If -1<=r<0, then r is
to be computed by minimizing the difference
between a moving mean curve for the data and the
r-parameterized smoothing spline. If r<-1, then a
moving quantile is used instead of a moving mean.
A non-negative r value is used as-is. r=0 implies
a perfect-fitting natural global cubic spline.

double *vp,
vp[1:m,1:1] is a 1-column matrix containing the
x values whereas we want values of our smoothing
spline computed and returned.  If vp=NULL then vp
is taken as hp col 1.
```

```
int16 idsw,
idsw is the integral/derivative control switch.
If idsw<0, then we want points of the derivative
of the smoothing spline to be returned.  If
idsw>0, then we want points of the integral of the
smoothing spline to be returned.  If idsw=0, we
want points of the smoothing spline itself to be
returned.

-------------------------------------------------------
This procedure computes the optimal smoothing
spline for the 2D functional data points given in
the rows of hp[1:n,1:2] with the associated
weights L[1:n], and the smoothing parameter r.
This is done by computing the points p[1:n,1:2]
for which the natural global cubic spline c that
interpolates the points in p[1:n,1:2] is the
minimizer of sum(i,1,n,L[i]*(hp[i,2]-c(hp[i,1]))^2) +
r*integral(x,hp[1,1],hp[n,1],|c''(x)|^2).

Given p[1:n,1:2], the optimal smoothing spline c
is determined.  We return the points on the graph
of c(x) (if idsw=0), or the derivative c'(x) (if
idsw<0), or the integral integral (s,hp[1,1],x,c(s))
(if idsw>0) for x=vp[1],vp[2],...,vp[m], as the rows
of a matrix dest[1:m,1:2].

Let x[1:n] = hp col 1, let f[1:n] = hp col 2, and
define the (n-1)-vector t[1:(n-1)] so that t[i] =
x[i+1]-x[i] for i = 1,...,(n-1).

To compute the actual interpolation points p[1:n,1:2]:

1) compute v[1:(n-2)] in (Gbar+r*H*L^(-1)*H')v = Hf,
   where f = hp col 2.
2) compute p[1:n] as f-rL^(-1)H'v.
3) return the matrix x&'p, where x = hp col 1.

Let a1 = t row 1:(n-1), let a2 = t row 1:(n-2),
let b1 = t row 2:(n-1), let b2 = t row 2:(n-2).

Gbar[1:(n-2),1:(n-2)] is the symmetric tridiagonal
matrix whose lower diagonal is (b2/6), whose main
diagonal is (a2+b1)/3, and whose upper diagonal is
(b2/6).
```

```
H[1:(n-2),1:n] is a matrix with a main diagonal
and two upper diagonals.  The main diagonal is 1/a2,
the first upper diagonal is -((1/a2)+(1/b1)), and
the second upper diagonal is 1/b1.

L[1:n,1:n] is a diagonal matrix whose diagonal
element L[i,i] is the weight (reciprocal variance)
for the i-th data point hp row i.
----------------------------------------------------*/
{int16 i,j,k,n;
 int16 vpsw=0, Lsw=0;
 int16 kl,kh,jl,jh,n2,winsize;
 double s,c,d;
 double *H;
 double *v;
 double *t;
 double *b;
 double *p;
 double *f;
 double *dest;

 /* Get the number of data points, n, in hp[1:n,1:2],
    remember hp is assumed to be sorted on col 1. */
 n = harrayrows(hp); n2 = n-2;

 /* Get the vector of interpolation x values, vp,
    whereat we want points. */
 if (vp EQ NULL) /* create vp = hp col 1 */
    {vpsw=1; vp=allocharray(n,1);
     for (j=1; j<=n; j++) vp[j] = hp[DX(j,1,2)];
    }
 m = harrayrows(vp);

 /* If n<=2, we have a simple straight-line
    interpolating spline which will be handled
    by cinterpolate(). */
 if (n < 3) goto c2spline;

 /* Get the weight vector L[1:n] */
 if (L EQ NULL) /* create L = 1^^n */
    {Lsw=1; L = allocharray(n,1);
     for (j=1; j<=n; j++) L[j] = 1.0;
    }
```

```
/* Now begin to compute the harray of desired actual
   data points p[1:n,1:2] for the global natural
   cubic spline which is our final desired optimal
   smoothing spline. */

/* Construct the local harray vector t[1:(n-1)]. We
   assume that no t[i]-value will be zero! */
t = allocharray(n-1,1);
for (i=1; i<=n-1; i++)
    t[i] = hp[DX(i+1,1,2)]-hp[DX(i,1,2)];

/* Construct the file global harray
   Gbar[1:(n-2),1:(n-2)].
   Let a1 = t row 1:(n-1), let a2 = t row 1:(n-2),
   let b1 = t row 2:(n-1), let b2 = t row 2:(n-2).
   Gbar[1:(n-2),1:(n-2)] is the symmetric tridiagonal
   matrix whose lower diagonal is (b2/6), whose main
   diagonal is (a2+b1)/3, and whose upper diagonal
   is (b2/6).*/
Gbar = allocharray(3,n2); /*size [1:n2,1:n2],
                          symmetric with 3 bands */
for (i=1; i<=n2; i++)
   {GbarM(i,i) = (t[i]+t[i+1])/3;
    if (i != n2)
       GbarM(i+1,i) = GbarM(i,i+1) = t[i+1]/6;
   }

/* Construct the local harray H[1:(n-2),1:n].
   H is a matrix with a main diagonal and two upper
   diagonals.  The main diagonal is 1/a2, the first
   upper diagonal is -((1/a2)+(1/b1)), and the second
   upper diagonal is 1/b1.*/
H = allocharray(3,n); /*size [1:n2,1:n], with 1
                    main-diagonal and 2 upper-bands*/
for (i=1; i<=n2; i++)
   {HM(i,i) = 1/t[i];
    HM(i,i+1) = -((1/t[i])+(1/t[i+1]));
    HM(i,i+2) = 1/t[i+1];
   }

/* Construct the file global harray (n-2)x1 vector
   hf = H*f, where f = hp col 2 */
hf = allocharray(n2,1);
for (i=1; i<=n2; i++)
   {kh = MIN(n,i+2);
```

```
     for (s=0.,k=i; k<=kh; k++)
         s += HM(i,k)*hp[DX(k,2,2)];
     hf[i] = s;
    }

/* We need the file global harray A[1:n2,1:n2] in
   dnu(), and also later, so we allocate it now. */
A = allocharray(5,n2); /*diagonals -2,-1,0,1,2
                          of A[1:n-2,1:n-2] */

/* If the smoothing factor r is given, use it.*/
if (r >= 0.) goto computespline;

/*--------------------------------------------------*/
/* If r < 0, compute r by minimizing the difference
   between the moving mean smoother( when -1<=r<0) or
   the moving median smoother (when r<-1) and the
   optimal spline smoother.

   To do this, we must:
   1) compute fbar = the moving mean of the vector
      of values f or moving median of f.
   2) compute the matrices H*L^(-1)*H', Gbar, fbar-f,
      L^(-1)*H', and H*f for reuse in the subroutine
      that computes -.5 times the derivative of our
      objective function denoted by dnu(r), where
      dnu(r) = (fbar-f+r*L^(-1)*H'*
               (Gbar+r*H*L^(-1)*H')^(-1)*H*f)'*
               [L^(-1)*H'*(Gbar+r*H*L^(-1)*H')^(-1)*
               Gbar*(Gbar+r*H*L^(-1)*H')^(-1)*H*f]

   To compute dnu(r):
   1) solve for x in (Gbar+r*H*L^(-1)*H')*x = H*f.
   2) compute y = Gbar*x.
   3) solve for z in (Gbar+r*H*L^(-1)*H')*z = y.
   4) compute dnu(r) = (fbar-f+r*L^(-1)*H'*x)'*L^(-1)*
                       H'*z.

   We wish to solve for r in dnu(r) = 0.
   The function dnu() given above depends upon the
   file global arrays Gbar, qf, fd, lht, hf, which
   are computed below.
*/
```

```
/* Compute the file global harray n x (n-2) matrix
   lht = L^(-1)*H'.  lht is stored in a 5-row,
   n-column diagonal table. */
lht = allocharray(5,n);
for (i=1; i<=n; i++)
  {kl = MAX(1,i-2); kh = MIN(i,n2);
   for (k=kl; k<=kh; k++)
       lhtM(i,k) = HM(k,i)/L[i];
  }

/* Compute the file global harray (n-2) x (n-2)
   matrix qf = H*L^(-1)*H'.  Note H*L^(-1)*H' is a
   bandwidth 5 symmetric (n-2)x(n-2) matrix.  qf is
   stored in a 5-row n2-column diagonal table. */
qf = allocharray(5,n2); /* qf is stored as 5
                           diagonals (initially zero)*/
for (i=1; i<=n2; i++)
  {jl = MAX(1,i-2); jh = MIN(n2,i+2);
   for (j=jl; j<=jh; j++)
     {kl = MAX(i,j);
      kh =kl+2-ABS(i-j);
      kh = MIN(kh,n);
      for (s=0.,k=kl; k<=kh; k++)
          s += (HM(i,k)/L[k])*HM(j,k);
      qfM(i,j) = s;
     }
   }

/* Compute the file global harray (n x 1) vectors f
   and fd = fbar-f, where fbar is computed via a
   moving mean computation.*/
f = allocharray(n,1); /* construct f = hp col 2 */
for (i=1; i<=n; i++) f[i] = hp[DX(i,2,2)];

winsize = n/10;
if (winsize < 5) winsize = 5;
if (winsize > n) winsize = n-1;
winsize= winsize | 1; /* OR to make winsize odd */

p = allocharray(winsize,1); /* set-up triangular
                               weight vector */
j = (winsize+1)/2;
for (i=1; i<j; i++) p[i] = p[winsize+1-i] = i;
p[j] = j-1;
```

```
 /* If -1<=r<0, then compute the moving-mean curve
    via the externally-supplied subroutine mmean(),
    otherwise r<-1, and we want to compute the
    moving-median curve via the externally-supplied
    subroutine mmedian(). In either case, we use the
    window size winsize and the weights
    p[1:winsize,1:1]. */
 if (r<-2.) fd = mmedian(f,winsize,0,-1,p);
 else fd = mmean(f,winsize,0,-1,p);
 free(p);

 for (i=1; i<=n; i++) fd[i] -= f[i];

 /* Compute r as the solution to dnu(r) = 0 via a
    binary search bracketing. We assume (safely) that
    dnu(0)<0 and we search for a value d such that
    dnu(d)>=0. Our starting interval is thus [0,d].*/
 for (d=10.; dnu(d)<0.; d+=10.);
 for (c=0.,d=10.; (d-c)>.0001;)
   {r=(c+d)/2.; s=dnu(r); if (s<=0.) c=r; else d=r;}

 /* Free the no-longer-needed global harrays. */
 free(qf); free(lht); free(f); free(fd);

/*--------------------------------------------------*/
computespline:

 /* construct the file global harray (n-2)x(n-2)
    matrix A = Gbar+r*H*L^(-1)*H'. Note H*L^(-1)*H'
    is a bandwidth 5 symmetric (n-2) x (n-2) matrix.*/
 A[1] = 0.; /* A is reinitialized to zero */
 for (i=1; i<=n2; i++)
   {jl = MAX(1,i-2); jh = MIN(n2,i+2);
    for (j=jl; j<=jh; j++)
      {kl = MAX(i,j); kh =kl+2-ABS(i-j); kh= MIN(kh,n);
       for (s=0.,k=kl; k<=kh; k++)
           s += (HM(i,k)/L[k])*HM(j,k);
       AM(i,j) = GbarM(i,j)+r*s;
      }
   }

 /* compute the local harray vector v[1:(n-2)] in
    (Gbar+r*H*L^(-1)*H')v = Hf, where f = hp col 2.
    Do this by calling bandsolve(A,hf) to return
    the vector v such that A*v = hf.*/
```

```
v = bandsolve(A,hf);

 /* compute the local harray p[1:n,1:2], where
    p col 1 = hp col 1, and p col 2 = f-r*L^(-1)H'v.*/
 p = allocharray(n,2);
 for (i=1; i<=n; i++)
    {p[DX(i,1,2)] = hp[DX(i,1,2)];
     kl = MAX(1,i-2); kh=MIN(i,n2);
     for (s=0.,k=kl; k<=kh; k++)
         s += HM(k,i)*v[k];
     p[DX(i,2,2)] = hp[DX(i,2,2)]-(r/L[i])*s;
    }

 /* free all unneeded allocated temporary arrays */
 free(Gbar); free(H);
 free(hf); free(A);
 free(t); free(v);

 c2spline:
 /* Compute the desired points (specified by vp[]) of
    the global natural C^2 spline (or its derivative
    or integral) for  the nx2 matrix p */

 /* Compute dest[1:m,1:2] = the desired output. */
 dest = cinterpolate(p,vp,1.,0,NULL,4,1,idsw);

 /* If idsw !=0, copy vp into dest col 1. (it's
    already done for idsw=0) */
 if (idsw) for (j=1; j<=m; j++)
                 dest[DX(j,1,2)] = vp[j];

 /* Free p if needed, and free L and/or vp if we
    created either. */
 if (n > 2) free(p);
 if (Lsw) free(L);
 if (vpsw) free(vp);

 return(dest);
}

/*=================================================*/
private int16 ox(int16 i, int16 j, int16 h, int16 w)
/*-------------------------------------------------
Given a banded matrix M[1:nr,1:nc] with 2h+1
bands, we can save space by storing these bands in
```

```
an array D whose rows correspond to the diagonals
of M which may contain non-zero elements.

ox(i,j,h,w) returns the 1-origin index of the
element in the Diagonal Storage array D that
corresponds to the banded matrix M.  The values i
and j are assumed to be in the correct ranges:
1<=i<=nr and 1<=j<=nc.  This computation does not
require accessing the actual storage area, so the
address D is not provided as an input.

The matrix M is assumed to have diagonals
h,h-1,...,1,0,-1,-2,...,-h, where the main
diagonal is diagonal 0, and the numbers of lower
diagonals are negative, and the numbers of upper
diagonals are positive.  Thus M has bandwidth
2h+1.  (Actually, the list of stored diagonals may
be truncated if they will not be accessed! i.e.
if diagonals -h and -(h-1) are not accessed, they
need not be stored.  But the diagonals that are
stored must be contiguous and must include the
main-diagonal 0 and diagonal 1.

The diagonals h,h-1,... of M[1:nr,1:nc] are stored
in the successive rows of an associated diagonal
storage array D[0:2h,1:w].  For k=0,1,...,h, the
diagonals k and -k are stored with k zeros
prefixed.  The argument w is the "width" of D
(i.e. the number of columns), which is that value
needed to completely hold the zero-padded stored
diagonals; w depends upon the values nr and nc.

For example, in the functions below, we access the
matrix element Gbar[i,j] of the matrix
Gbar[1:n-2,1:n-2] via ox(i,j,1,n-2), and we access
the matrix element H[i,j] of the matrix
H[1:n-2,1:n] via ox(i,j,2,n). Macros are given
above for all the matrices which involve the use
of ox().
-----------------------------------------------------*/
{if (ABS(j-i)>h) return(1); /* 1 indexes a 0-value!*/
 return(w*(h+i-j)+MAX(i,j)); }
/*===================================================*/
private double *bandsolve(double *A, double *b)
/*---------------------------------------------------
```

```
Solve for and return x[1:n] in A*x=b, where A is a
bandwidth 5 symmetric n x n matrix.
 ----------------------------------------------------*/
{int16 j,i,i1,i2,n,n1,j1,jh;
 double mv,Zii;
 double *x;
 double *Z;

 Z = copyharray(A);
 x = copyharray(b);
 n = harrayrows(b); n1=n-1;

 /* For i=1,...,n:
        divide [A&'b] row i by A[i,i].
        if i<n, subtract A[i+1,i]*([A&'b] row i)
                from [A&'b] row i+1.
        if i<n-1, subtract A[i+2,i]*([A&'b] row i)
                  from [A&'b] row i+2.
    Now A is upper-triangular with ones on its main
    diagonal and 2 upper diagonals.  Back-solve to
    compute x such that Ax=b.
 */
 for (i=1; i<=n; i++)
    {i1=i+1; i2=i+2; /*j1 = MAX(1,i-2);*/
     jh=MIN(n,i2);
     Zii = ZM(i,i); x[i] /= Zii;
     for (j=i1; j<=jh; j++) ZM(i,j) /= Zii;

     if (i < n)
       {mv = ZM(i1,i); x[i1] -= mv*x[i];
        ZM(i1,i1) -= mv*ZM(i,i1);
        if (i<(n1)) ZM(i1,i2) -= mv*ZM(i,i2);
       }

      if (i<(n1))
        {mv = ZM(i2,i); x[i2] -= mv*x[i];
         ZM(i2,i1) -= mv*ZM(i,i1);
         ZM(i2,i2) -= mv*ZM(i,i2);
        }
    }

 for (i=n1; i>0; i--)
    {jh = MIN(n,i+2);
     for (j=i+1; j<=jh; j++) x[i] -= ZM(i,j)*x[j];
    }
 free(Z);
 return(x);
}

/*====================================================*/
private double dnu(double r)
/*-----------------------------------------------------
dnu(r) is -.5 times the derivative of our objective
function.
  dnu(r) = (fbar-f+r*L^(-1)*H'*
           (Gbar+r*H*L^(-1)*H')^(-1)*H*f)'*
           [L^(-1)*H'*(Gbar+r*H*L^(-1)*H')^(-1)*Gbar*
           (Gbar+r*H*L^(-1)*H')^(-1)*H*f].

We are given Gbar, and qf = H*L^(-1)*H', fd =
fbar-f, lht = L^(-1)*H', and hf = H*f.  The
harrays Gbar[1:n-2,1:n-2], qf[1:n-2,1:n-2],
fd[1:n], lht[1:n,1:n-2], and hf[1:n-2] are all
file globals. We are also given the file global
array A[1:n-2,1:n-2] to use to hold Gbar+r*qf
(computed for each call during root-finding.) The
arrays A, qf,Gbar, and lht are all stored by
diagonals and accessed via the gnomin-based index
function computed in ox().

To compute dnu(r):
 1) solve for x in (Gbar+r*qf)*x = hf.
 2) compute y = Gbar*x.
 3) solve for z in (Gbar+r*qf)*z = y.
 4) compute dnu(r) = (fd+r*lht*x)'*lht*z.
-----------------------------------------------------*/
{int16 n2,kl,kh;
 int16 i,j,k,n;
 double *x;
 double *y;
 double *z;
 double nuv,s,t;

 n = harrayrows(fd); n2 = n-2;

 /* Compute A=Gbar+r*qf */
 A[1] = 0.; /* To be sure there is zero there. */
 for (i=1; i<=n2; i++)
   {kh = MIN(n2,i+2);
```

```
    for (j=i; j<=kh; j++)
      {t = GbarM(i,j)+r*qfM(i,j);
       AM(i,j) = t;
       if (i != j) AM(j,i) = t;
      }
   }

 x = bandsolve(A,hf); /* x is (n-2) x 1 */

 /* Compute y=Gbar*x */
 y = allocharray(n2,1);
 for (i=1; i<=n2; i++)
    {kl = MAX(1,i-1); kh = MIN(n2,i+1);
     for (s=0.,k=kl; k<=kh; k++) s += GbarM(i,k)*x[k];
     y[i] = s;
    }

 z = bandsolve(A,y); /* z is (n-2) x 1 */

 /* compute dnu(r) = (fd+r*lht*x)'*lht*z. */
 for (nuv=0.,i=1; i<=n; i++)
    {kl = MAX(1,i-2); kh = MIN(n2,i);
     for (s=0., k=kl; k<=kh; k++) s += lhtM(i,k)*x[k];
     t = fd[i]+r*s;
     for (s=0., k=kl; k<=kh; k++) s += lhtM(i,k)*z[k];
     nuv += t*s;
    }

 /* clean up allocated arrays */
 free(z); free(y); free(x);

 return(nuv);
}

/*=====================================================*/
/* end of file ssp.c */
```

# 16
# Tensor Product Surface Splines

In this section we will construct various parametric functions piecewise-composed of bicubic functions that map from $\mathcal{R}^2$ to $\mathcal{R}^3$. These functions are bicubic surface splines analogous to the cubic space curve splines that we studied above. Again we want to construct spline surfaces that contain given points in 3-space, and, generally, that have specified directional derivatives or directional tangent vectors at these given interpolation points.

## 16.1 Bicubic Tensor Product Surface Patch Splines

Suppose we are given four distinct points $b_{00}$, $b_{10}$, $b_{01}$, and $b_{11}$ on a parametrically defined surface $\tilde{S} := \{S(u, v) \mid u, v \in \mathcal{R}\} \subset \mathcal{R}^3$, where $S$ is a function mapping $\mathcal{R}^2$ into $\mathcal{R}^3$ such that $b_{00} = S(0, 0)$, $b_{10} = S(1, 0)$, $b_{01} = S(0, 1)$, and $b_{11} = S(1, 1)$. The surface $\tilde{S}$ need not be single-valued, i.e., a line parallel to the $z$-axis may intersect $\tilde{S}$ several times, and $\tilde{S}$ may even be self-intersecting. The four surface points $b_{00}$, $b_{10}$, $b_{11}$, and $b_{01}$ are the *corner* points of a so-called *quad-patch* $Q$ on the surface $\tilde{S}$ whose boundary is determined by connecting $b_{00}$ to $b_{01}$, $b_{01}$ to $b_{11}$, $b_{11}$ to $b_{10}$, and $b_{10}$ to $b_{00}$ by certain suitable curve segments on the surface $\tilde{S}$. The four surface points $b_{00}$, $b_{10}$, $b_{11}$, and $b_{01}$ are thus listed in cyclic order according to their occurrence on the complete boundary curve $\partial Q$ of the patch $Q$. Explicitly, the quad-patch $Q = \{S(u, v) \mid 0 \leq u \leq 1, 0 \leq v \leq 1\}$. Usually we have only partial geometric information about the patch $Q$, and the defining two-argument vector valued function $S$ is not known.

**Exercise 16.1:** What are the boundary curves of the quad-patch $Q$?

Let $h(t,a,b,c,d) = f_0(t)a + f_1(t)b + f_2(t)c + f_3(t)d$, where $f_0(t) = 1-3t^2+2t^3$, $f_1(t) = 3t^2-2t^3$, $f_2(t) = t-2t^2+t^3$, and $t_3(t) = -t^2+t^3$; $h(t,a,b,c,d)$ is the Hermite cubic interpolating polyomial space curve segment with the parameter limit value 1, based on the *control vectors* $a$, $b$, $c$, and $d$, where $c$ is the tangent vector at the point $a = h(0,a,b,c,d)$, and $d$ is the tangent vector at the point $b = h(1,a,b,c,d)$. If we specify tangent vectors at the points $b_{00}$ and $b_{10}$ for the boundary curve segment of the patch $Q$ from $b_{00}$ to $b_{10}$, to be denoted by $b_{20}$ at the point $b_{00}$ and by $b_{30}$ at the point $b_{10}$, we can take the Hermite cubic space curve segment $\{h(u, b_{00}, b_{10}, b_{20}, b_{30}) \mid 0 \le u \le 1\}$ as an estimated boundary curve segment of $Q$ between $b_{00}$ and $b_{10}$. Similarly, we may specify starting and ending tangent vectors for the $Q$-boundary curve from the point $b_{01}$ to the point $b_{11}$, where the tangent vector at the point $b_{01}$ is denoted by $b_{21}$ and the tangent vector at the point $b_{11}$ is denoted by $b_{31}$, and we may then take $\{h(u, b_{01}, b_{11}, b_{21}, b_{31}) \mid 0 \le u \le 1\}$ to be an estimated $Q$-boundary curve segment between $b_{01}$ and $b_{11}$.

The so-called tensor product bicubic patch $\hat{Q}$ that serves as an estimate for the patch $Q$ is now formed by sweeping the curve segment $\{h(u, b_{00}, b_{10}, b_{20}, b_{30}) \mid 0 \le u \le 1\}$ through space while deforming it so that it matches the curve segment $\{h(u, b_{01}, b_{11}, b_{21}, b_{31}) \mid 0 \le u \le 1\}$ at the end of the sweep [Far90]. The deformation is achieved by making the points and tangent vectors, which define the cubic segment estimated boundary curve between $b_{00}$ and $b_{01}$ that is being swept, smoothly change from $b_{00}$, $b_{10}$, $b_{20}$, and $b_{30}$, into $b_{01}$, $b_{11}$, $b_{21}$, and $b_{31}$ during the sweep. This can be done by defining the cubic interpolation functions $b_i(v) = h(v, b_{i0}, b_{i1}, b_{i2}, b_{i3})$ such that $b_i(0) = b_{i0}$ and $b_i(1) = b_{i1}$ for $i = 0, 1, 2, 3$. In order to do this we must introduce the 8 tangent vectors $b_{i2}$ and $b_{i3}$ which allow us to define the $b_i(v)$ curves. Note that $b_0(v)$ is the estimated boundary curve between $b_{00}$ and $b_{01}$ and $b_1(v)$ is the estimated boundary curve between $b_{10}$ and $b_{11}$.

Recall that $h(t, b_{i0}, b_{i1}, b_{i2}, b_{i3}) = f_0(t)b_{i0} + f_1(t)b_{i1} + f_2(t)b_{i2} + f_3(t)b_{i3}$ where $f_0(t) = 1-3t^2+2t^3$, $f_1(t) = 3t^2-2t^3$, $f_2(t) = t-2t^2+t^3$, and $f_3(t) = -t^2 + t^3$. Then the control vectors $b_i(v)$ of the spline segment being swept are $b_i(v) = \sum_{0 \le j \le 3} f_j(v)b_{ij}$ for $i = 0, 1, 2, 3$. Thus, $b_i(v) = h(v, b_{i0}, b_{i1}, b_{i2}, b_{i3})$ is a space curve cubic spline segment which connects the point $b_{i0}$ with $b_{i1}$, having the tangent vector $b_{i2}$ at the point $b_{i0}$ and the tangent vector $b_{i3}$ at the point $b_{i1}$. Remember that $b_2(v)$ and $b_3(v)$ define tangent vectors at the points $b_0(v)$ and $b_1(v)$.

The *parametric tensor product bicubic patch interpolation function* for the patch $Q$ is thus defined as

$$x_Q(u, v) = h(u, b_0(v), b_1(v), b_2(v), b_3(v)) = \sum_{0 \le i \le 3} f_i(u) \sum_{0 \le j \le 3} f_j(v) b_{ij},$$

and $\hat{Q} = \{x_Q(u, v) \mid 0 \le u \le 1, 0 \le v \le 1\}$ is an estimate of the patch $Q$. The patch $\hat{Q}$ coincides with the patch $Q$ at the points $b_{00}$, $b_{10}$, $b_{01}$, and $b_{11}$, and, as shown below, has the same $u$- and $v$ directional derivatives and mixed partial derivatives there as are assigned to the patch $Q$. The term "tensor product" is a description of the algebraic form of this double summation expression. Note $x_Q$ maps $\mathcal{R}^2$ into $\mathcal{R}^3$, so that $x_Q(u, v) \in \mathcal{R}^3$.

The surface defined by $x_Q(u, v)$ corresponds to sweeping the associated cubic spline segment $\{ h(u, b_0(v), b_1(v), b_2(v), b_3(v)) \mid 0 \le u \le 1 \}$ through space by varying $v$ between 0 and 1. Thus, $v$ is the sweep parameter and $u$ is the deformation parameter. As the following exercise shows, the roles of $u$ and $v$ can be reversed.

**Exercise 16.2:** Suppose we sweep the spline segment $\{h(v, b_{00}, b_{01}, b_{02}, b_{03}) \mid 0 \le v \le 1\}$ while deforming it to end up with $\{h(v, b_{10}, b_{11}, b_{12}, b_{13}) \mid 0 \le v \le 1\}$ based on the 8 additional control vectors $b_{20}$, $b_{21}$, $b_{30}$, $b_{31}$, $b_{22}$, $b_{23}$, $b_{32}$, $b_{33}$. Show that we get the same bicubic patch interpolation function that is defined above.

**Solution 16.2:** This construction yields:
$$x_Q(u, v) = \textstyle\sum_{0 \le j \le 3} f_j(v) \sum_{0 \le i \le 3} f_i(u) b_{ij}.$$

**Exercise 16.3:** What is the "diagonal" space curve $x_Q(t, t)$ for $0 \le t \le 1$? Is this curve the intersection of the patch $\{x_Q(u, v) \mid 0 \le u \le 1,\ 0 \le v \le 1\}$ and a plane?

**Exercise 16.4:** Show that $x_Q(u, v) = \left[1\ u\ u^2\ u^3\right] C^{\mathrm{T}} B C \left[1\ v\ v^2\ v^3\right]^{\mathrm{T}}$, where $B$ is the $4 \times 4$ matrix of 3-vector components with $B_{ij} = b_{ij}$ and

$$C = \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

If we precompute the matrix of vectors $A = C^T BC$, then we can evaluate $x_Q$ for any arguments $u$ and $v$ via Horner's rule for evaluating a polynomial with the following procedure. The value of $x_Q(u, v)$ ends up in $s$.

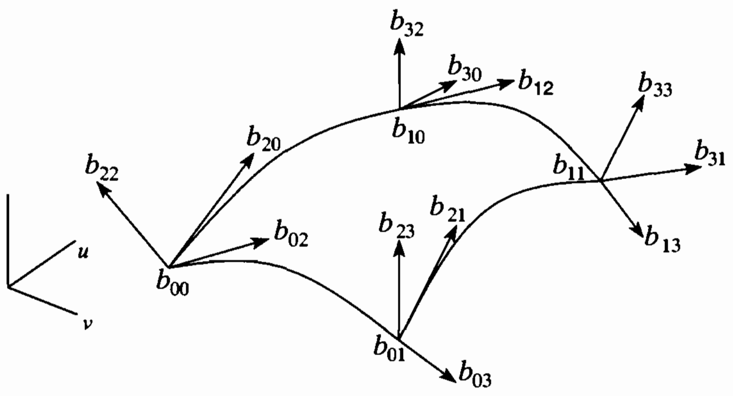
```
$s \leftarrow 0$;
for ($i \leftarrow 3 : 0 : -1$)
  { $r \leftarrow A_{i3}$;
    for ($j \leftarrow 2 : 0 : -1$) $r \leftarrow r \cdot u + A_{ij}$;
    $s \leftarrow s \cdot v + r$ }
```

Note that the blending functions $f_0$, $f_1$, $f_2$, and $f_3$ satisfy $f_0(0) = 1$, $f_1(0) = 0$, $f_2(0) = 0$, $f_3(0) = 0$, and $f_0(1) = 0$, $f_1(1) = 1$, $f_2(1) = 0$, $f_3(1) = 0$, i.e., $f_i(0) = \delta_{i0}$ and $f_i(1) = \delta_{i1}$. Thus, the bicubic patch interpolation function $x_Q(u, v)$ satisfies $x_Q(0, 0) = b_{00}$, $x_Q(0, 1) = b_{01}$, $x_Q(1, 0) = b_{10}$, and $x_Q(1, 1) = b_{11}$.

Let us compute the partial derivative functions $(x_Q)_u$, $(x_Q)_v$, and $(x_Q)_{uv}$ of the bicubic patch interpolation function $x_Q(u, v)$. (We use the common *subscript* notation for partial derivatives, so that $\partial F/\partial z = F_z$.) Note that $f_0'(0) = 0$, $f_1'(0) = 0$, $f_2'(0) = 1$, $f_3'(0) = 0$, and $f_0'(1) = 0$, $f_1'(1) = 0$, $f_2'(1) = 0$, $f_3'(1) = 1$. Thus, the bicubic patch interpolation function $x_Q(u, v)$ satisfies $(x_Q)_u(0, 0) = b_{20}$, $(x_Q)_u(0, 1) = b_{21}$, $(x_Q)_u(1, 0) = b_{30}$, and $(x_Q)_u(1, 1) = b_{31}$. Similarly, $(x_Q)_v(0, 0) = b_{02}$, $(x_Q)_v(0, 1) = b_{03}$, $(x_Q)_v(1, 0) = b_{12}$, and $(x_Q)_v(1, 1) = b_{13}$. Also, $(x_Q)_{uv}(0, 0) = b_{22}$, $(x_Q)_{uv}(0, 1) = b_{23}$, $(x_Q)_{uv}(1, 0) = b_{32}$, and $(x_Q)_{uv}(1, 1) = b_{33}$.

When the (unknown) parametric surface function $S(u, v)$ associated with the patch $Q$ is such that $b_{00} = S(0, 0)$, $b_{10} = S(1, 0)$, $b_{01} = S(0, 1)$, $b_{11} = S(1, 1)$, $b_{20} = S_u(0, 0)$, $b_{30} = S_u(1, 0)$, $b_{21} = S_u(0, 1)$, $b_{31} = S_u(1, 1)$, $b_{02} = S_v(0, 0)$, $b_{03} = S_v(1, 0)$, $b_{12} = S_v(0, 1)$, $b_{13} = S_v(1, 1)$, $b_{22} = S_{uv}(0, 0)$, $b_{32} = S_{uv}(1, 0)$, $b_{23} = S_{uv}(0, 1)$, and $b_{33} = S_{uv}(1, 1)$, the parametric tensor product bicubic patch interpolation function $x_Q$ for the patch $\hat{Q}$ matches the unknown function $S$, its $u$- and $v$ directional derivatives and its mixed partial $uv$-derivatives at the parameter value pairs $(0, 0)$, $(1, 0)$, $(0, 1)$, $(1, 1)$[Fer64].

The mixed partial vectors $b_{22}$, $b_{23}$, $b_{32}$, and $b_{33}$ used at the corners of the quad-patch $\hat{Q}$ are called the *corner twist vectors* of the bicubic patch function $x_Q$ and the patch $\hat{Q}$. The bicubic patch function $x_Q$ is thus determined by 48 numbers which form the 4 corner points, the 8 corresponding $u$ directional and $v$ directional tangent vectors, and the 4 corner twist vectors of the quad-patch $\hat{Q}$. These same 16 vectors have been identified above as



those vectors defining the sweeping and deforming of a patch boundary cubic spline curve segment to the opposite patch boundary cubic spline curve segment. The figure above is useful, but somewhat deceptive; the tangent and twist vectors are shown translated from the origin; thus, for example, the point shown as $b_{22}$ should be labeled $b_{22} + b_{00}$.

**Exercise 16.5:** Compute the unit normal vector $n(u, v)$ for the tensor product bicubic patch interpolation function $x_Q$ corresponding to the point $x_Q(u, v)$.

**Exercise 16.6:** How must we choose the 12 control tangent and twist vectors to insure that the components of $x_Q(u, v)$ satisfy $x_Q(u, v)_1 = u$ and $x_Q(u, v)_2 = v$?

**Exercise 16.7:** Describe the locally extreme points of the bivariate cubic spline patch function $x_Q(u, v)$. How many distinct isolated locally extreme points can $x_Q$ have? What are the possible loci of the non-isolated locally extreme points in the various cases where such points arise?

## 16.2 A Generalized Tensor Product Patch Spline

In order to obtain a geometrically pleasing estimate for the unknown patch $Q$, it is usually necessary to control the lengths of the spline curves that form the boundary of the patch $\hat{Q}$. Since the length of a cubic spline segment depends on the parameter limit value used and on the magnitudes of the starting and ending tangent vectors, we have several options for controlling the individual boundary curve lengths; we can individually scale the various tangent vectors $b_{02}, b_{02}, b_{03}, b_{21}, b_{31}, b_{12}, b_{13}, b_{20}$, and $b_{30}$ used in defining $\hat{Q}$, or we can introduce variable parameter limit values.

Note that in the preceding development, we have restricted the five cubic spline segment functions $b_0(v), b_1(v), b_2(v), b_3(v)$, and $h(u, b_0(v), b_1(v), b_2(v), b_3(v))$ that define our tensor product bicubic patch interpolation function to all have their parameters $u$ and $v$ range between 0 and 1 as the spline curves range from their starting interpolation point to their ending interpolation point. We can remove this restriction by introducing individual parameter ranges for $u$ and $v$ as follows. Replace the basic Hermite blending functions with the following 2-argument *generalized Hermite blending functions*:

$$\begin{aligned}
f_0(t,d) &= 1-3(t/d)^2+2(t/d)^3,\\
f_1(t,d) &= 3(t/d)^2-2(t/d)^3,\\
f_2(t,d) &= d(t/d-2(t/d)^2+(t/d)^3), \quad \text{and}\\
f_3(t,d) &= d(-(t/d)^2+(t/d)^3).
\end{aligned}$$

In evaluating $f_i(0,0)$, we may choose $0/0$ to be either 0 or 1, as long as the same choice is used consistently.

Now we may introduce the parameter limit values $t_u$ and $t_v$ and express the *generalized parametric tensor product bicubic patch interpolation function* as

$$x_Q(u,v) = \sum_{0\le i\le 3} f_i(u,t_u) \sum_{0\le j\le 3} f_j(v,t_v) b_{ij}.$$

Then $\hat{Q} = \{x_Q(u,v) \mid 0 \le u \le t_u, 0 \le v \le t_v\}$ is an estimate of the patch $Q$. (Beware: the subscripts $u$ and $v$ in the symbols $t_u$ and $t_v$ do *not* denote differentiation as might be expected at first glance; $t_u$ and $t_v$ are merely mnemonic symbols for particular constants.) The patch $\hat{Q}$ coincides with the patch $Q$ at the points $b_{00}, b_{10}, b_{01}$, and $b_{11}$, and has the same $u$- and $v$ directional derivatives there. The boundary splines $x_Q(u,0)$ and $x_Q(u,t_v)$ have the common parameter limit value $t_u$, and the boundary splines $x_Q(0,v)$ and $x_Q(t_u,v)$ have the common parameter limit value $t_v$.

Note that even with individual parameter limit values $t_u$ and $t_v$, $x_Q(u,v)$ is not necessarily a close estimate of the associated unknown parametric surface function $S$, since the parameterization of $S$ is not known, and hence the parameter limits for the patch $Q$ defined by $S$ need not coincide with the parameter limits $t_u$ and $t_v$. Thus we are estimating a patch, not a function.

## 16.3 Regular Grid Multi-Patch Surface Interpolation

Suppose we have $nm$ surface points belonging to an unknown surface in $\mathcal{R}^3$, together with associated $u$- and $v$ directional tangent vectors and corner twist vectors corresponding to the *regular grid* of $nm$ pairs of parameter values $(u_i, v_j)$ with $0 \le i \le n-1$ and $0 \le j \le m-1$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$. Explicitly, this grid is the set $\{u_0, \ldots, u_{n-1}\} \times \{v_0, \ldots, v_{m-1}\}$. We wish to construct a surface which interpolates the given points and whose corresponding tangent vectors and twist vectors match the given tangent and twist vectors. This surface can be computed by computing the $(n-1)(m-1)$ individual quad-patch estimates for the $(n-1)(m-1)$ unknown surface quad-patches. Moreover, given the underlying grid, we can also specify a uniform parametric function that defines our overall estimated surface and which may serve as an estimate of the unknown two-argument vector valued function defining the surface being estimated.

The $nm$ parameter value pairs $(u_i, v_j)$ form a regular grid of points in the $uv$-plane which *underlies* the given $4nm$ data vectors. Such a grid of parameter value pairs constitutes the set of 2-dimensional *knot values* associated with the $nm$ data points in direct analogy to the simpler situation of a sequence of 1-dimensional knot values. The parameter value pairs $(u_i, v_j)$, $(u_i, v_{j+1})$, $(u_{i+1}, v_{j+1})$, and $(u_{i+1}, v_j)$ in cyclic order are the corners of the $i, j$-th *grid rectangle* for $0 \le i < n-1$ and $0 \le j < m-1$. The $i, j$-th grid rectangle has the associated $u$-parameter limit value $u_{i+1} - u_i$ and the associated $v$-parameter limit value $v_{j+1} - v_j$. Note the $u$-parameter limit value is the same for each grid rectangle that shares the same $u$-parameter range, and the $v$-parameter limit value is the same for each grid rectangle that shares the same $v$-parameter range.

The 16 data vectors and two parameter limit values associated with the $i, j$-th grid rectangle thus defines the corresponding generalized parametric tensor product bicubic patch interpolation function $x_{Q_{ij}}$ for the associated 4-cornered quad-patch $Q_{ij}$ of the unknown surface function being interpolated. These $(n-1)(m-1)$ generalized parametric tensor product bicubic patch interpolation functions join with $C^1$-continuity across their common boundaries and define the *generalized parametric tensor product* *bicubic spline interpolation function* $x$ for the union of the $(n-1)(m-1)$ individual quad-patches. For $(u, v) \in [u_0, u_{n-1}] \times [v_0, v_{m-1}]$, the vector $x(u, v)$ is computed by determining the integers $i$ and $j$ such that $(u, v) \in [u_i, u_{i+1}] \times [v_j, v_{j+1}]$ and then computing $x(u, v) = x_{Q_{ij}}(u - u_i, v - v_j)$ where $Q_{ij}$ is the quad-patch corresponding to the $i, j$-th grid rectangle.

Note that each of the real valued two-argument component functions $x_1, x_2$, and $x_3$ is a parametric bicubic spline function for $nm$ points on a single-valued surface, namely: $\{ (u_i, v_j, z_{ij}) \mid 0 \leq i \leq n-1, 0 \leq j \leq m-1 \}$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$ are the given values that define the underlying grid, and where $z_{ij} \in \mathcal{R}$, and the $z_{ij}$ values are variously the first, second, or third component values of the $nm$ given data points.

## 16.4 Estimating Tangent and Twist Vectors

In order to construct a continuously differentiable bicubic spline interpolation function $x$ for a network of 4-cornered patches, each defined by four surface points, we want to specify the remaining 12 control vectors for each patch, such that the cross boundary tangent vectors for adjacent patches are identical to what we have done above, and so that the corner twist vectors shared between patches are also identical. Let $p_{ij}$ denote the 3-tuple data point associated with the parameter value pair $(u_i, v_j)$. In general, we can use any one of the tangent estimation methods available for univariate cubic splines to estimate the required $2nm$ cross boundary tangent vectors at the various patch corner points by estimating $n$ tangent vectors for the sequence of points $p_{0j}, \ldots, p_{n-1,j}$ for each value $j = 0, 1, \ldots, m-1$ and estimating $m$ tangent vectors for the sequence of points $p_{i0}, \ldots, p_{i,m-1}$ for each value $i = 0, 1, \ldots, n-1$.

Corner twist vectors may be estimated in several possible ways. One way is to treat all the $u$ direction tangent vectors for a fixed $u$-value $u_i$ as a set of points on a space curve and take estimated tangent vectors for this set of points as the corresponding estimated corner twist vectors. We may similarly use the $v$ direction tangent vectors for a fixed $v$-value $v_j$ to obtain associated estimated corner twist vectors; finally, we may use the average of the corresponding vectors in these two groups to get an improved estimated corner twist vector at each parameter value pair $(u_i, v_j)$.

**Exercise 16.8:** A bicubic join order 1 spline, defining an interpolating surface for $(n-1)(m-1)$ quad-patches corresponding to an underlying regular grid defined by the parameter value pairs $\{(u_i, v_j) \mid 0 \le i \le n-1,\ 0 \le j \le m-1\}$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$, can be considered to be an element of a vector space of such two-argument bicubic spline functions of dimension $12nm$, since there are four 3-vectors to be specified at each parameter value pair. Find a basis for this vector space.

**Exercise 16.9:** Given $nm$ points on a single-valued surface $\tilde{S}$ over a regular grid as $\{(u_i, v_j, z_{ij}) \mid 0 \le i \le n-1, 0 \le j \le m-1\}$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$ are given values that also define the underlying grid, and where $z_{ij} \in \mathcal{R}$, we may estimate the $u$ and $v$ directional tangent vectors at the $nm$ surface points using global second derivative continuous univariate cubic splines with specific choices of end conditions. Suppose the corner twist vectors are arbitrarily specified non-zero vectors. Does the resulting bicubic interpolation surface function $x(u, v)$ for the entire net of $(n-1)(m-1)$ quad-patches have continuous second derivatives $x_{uu}$, $x_{vv}$, $x_{uv}$, $x_{uuv}$, $x_{uvv}$, and $x_{uuvv}$ everywhere? *Hint*: recall that $x(u, v)$ is defined here in terms of join order 1 Hermite blending functions.

We may compute estimated corner twist vectors for a single patch Q with the corner points $b_{00}, b_{10}, b_{11}$, and $b_{01}$ by using the mixed partials of the *bilinear* interpolation surface for the points $b_{00}, b_{10}, b_{11}$, and $b_{01}$ defined by the parametric function: $S_L(u, v) = b_{00}(1-u)(1-v) + b_{10}u(1-v) + b_{01}(1-u)v + b_{11}uv$. At the junction of four patches we may use the average of the four separately determined bilinear surface twist vectors at that junction point.

Another approach to estimating the $u$- and $v$ directional tangent vectors and the twist vector associated with each of the points in a given collection of $nm$ points $p_{ij}$ over an underlying regular grid $\{(u_i, v_j) \mid 0 \le i \le n-1, 0 \le j \le m-1\}$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$, is to use the biquadratic vector valued functions $B_{ij}(u, v)$ defined so that $B_{ij}$ fits the set of 9 points centered at the point associated with the parameter value pair $(u_i, v_j)$, for $1 \le i \le n-2$ and $1 \le j \le m-2$. The set of 9 points "centered" at the point $p_{ij}$ with the associated parameter value pair $(u_i, v_j)$ consists of the $3 \times 3$ array of points $p_{i-1,j-1}, p_{i,j-1}, p_{i+1,j-1}, p_{i-1,j}, p_{i,j}, p_{i+1,j}, p_{i-1,j+1}, p_{i,j+1}, p_{i+1,j+1}$ with the associated parameter value pairs $(u_{i-1}, v_{j-1})$, $(u_i, v_{j-1})$, $(u_{i+1}, v_{j-1})$, $(u_{i-1}, v_j)$, $(u_i, v_j)$, $(u_{i+1}, v_j)$, $(u_{i-1}, v_{j+1})$, $(u_i, v_{j+1})$, $(u_{i+1}, v_{j+1})$.

For $1 \le i \le n-2$ and $1 \le j \le m-2$, the $u$ directional tangent vector associated with the surface point $p_{ij}$ is estimated by $(B_{ij})_u(u_i, v_j)$, the $v$ directional tangent vector associated with the surface point $p_{ij}$ is estimated by $(B_{ij})_v(u_i, v_j)$, and the twist vector associated with the surface point $p_{ij}$ is estimated by $(B_{ij})_{uv}(u_i, v_j)$.

The tangent and twist vectors at surface points associated with parameter value pairs at the boundary of the grid can be estimated from the biquadratic functions centered at the nearby interior parameter value pairs of the grid. Thus for $1 \leq j \leq m-2$, the $u$ directional tangent vector associated with the surface point $p_{0j}$ is estimated by $(B_{1j})_u(u_0, v_j)$, the $v$ directional tangent vector associated with the surface point $p_{0j}$ is estimated by $(B_{1j})_v(u_0, v_j)$, and the twist vector associated with the surface point $p_{0j}$ is estimated by $(B_{1j})_{uv}(u_0, v_j)$.

For $1 \leq j \leq m-2$, the $u$ directional tangent vector associated with the surface point $p_{n-1,j}$ is estimated by $(B_{n-2,j})_u(u_{n-1}, v_j)$, the $v$ directional tangent vector associated with the surface point $p_{n-1,j}$ is estimated by $(B_{n-2,j})_v(u_{n-1}, v_j)$, and the twist vector associated with the surface point $p_{n-1,j}$ is estimated by $(B_{n-2,j})_{uv}(u_{n-1}, v_j)$.

For $1 \leq i \leq n-2$, the $u$ directional tangent vector associated with the surface point $p_{i0}$ is estimated by $(B_{i1})_u(u_i, v_1)$, the $v$ directional tangent vector associated with the surface point $p_{i0}$ is estimated by $(B_{i1})_v(u_i, v_1)$, and the twist vector associated with the surface point $p_{i0}$ is estimated by $(B_{i1})_{uv}(u_i, v_1)$. For $1 \leq i \leq n-2$, the $u$ directional tangent vector associated with the surface point $p_{i,m-1}$ is estimated by $(B_{i,m-2})_u(u_i, v_{m-1})$, the $v$ directional tangent vector associated with the surface point $p_{i,m-1}$ is estimated by $(B_{i,m-2})_v(u_i, v_{m-1})$, and the twist vector associated with the surface point $p_{i,m-1}$ is estimated by $(B_{i,m-2})_{uv}(u_i, v_{m-1})$.

For the surface point $p_{00}$, the $u$ directional tangent vector is estimated by $(B_{11})_u(u_0, v_0)$, the $v$ directional tangent vector by $(B_{11})_v(u_0, v_0)$, and the twist vector by $(B_{11})_{uv}(u_0, v_0)$. For the surface point $p_{n-1,0}$, the $u$ directional tangent vector is estimated by $(B_{n-2,1})_u(u_{n-1}, v_0)$, the $v$ directional tangent vector is estimated by $(B_{n-2,1})_v(u_{n-1}, v_0)$, and the twist vector is estimated by $(B_{n-2,1})_{uv}(u_{n-1}, v_0)$. For the surface point $p_{0,m-1}$, the $u$ directional tangent vector is estimated by $(B_{1,m-2})_u(u_0, v_{m-1})$, the $v$ directional tangent vector is estimated by $(B_{1,m-2})_v(u_0, v_{m-1})$, and the twist vector is estimated by $(B_{1,m-2})_{uv}(u_0, v_{m-1})$. For the surface point $p_{n-1,m-1}$, the $u$directional tangent vector is estimated by $(B_{n-2,m-2})_u(u_{n-1}, v_{m-1})$, the $v$ directional tangent vector is estimated by $(B_{n-2,m-2})_v(u_{n-1}, v_{m-1})$, and the twist vector is estimated by $(B_{n-2,m-2})_{uv}(u_{n-1}, v_{m-1})$.

For $1 \leq i \leq n-2$ and $1 \leq j \leq m-2$, the vector coefficients in the biquadratic function

$$\begin{aligned} B_{ij}(u, v) &= c_{ij9}u^2v^2 + c_{ij8}u^2v + c_{ij7}uv^2 + c_{ij6}u^2 + c_{ij5}v^2 \\ &\quad + c_{ij4}uv + c_{ij3}u + c_{ij2}v + c_{ij1} \end{aligned}$$

centered at the $3 \times 3$ array of points $p_{i-1,j-1}$, $p_{i,j-1}$, $p_{i+1,j-1}$, $p_{i-1,j}$, $p_{i,j}$, $p_{i+1,j}$, $p_{i-1,j+1}$, $p_{i,j+1}$, $p_{i+1,j+1}$ with the associated parameter value pairs $(u_{i-1}, v_{j-1})$, $(u_i, v_{j-1})$, $(u_{i+1}, v_{j-1})$, $(u_{i-1}, v_j)$, $(u_i, v_j)$, $(u_{i+1}, v_j)$, $(u_{i-1}, v_{j+1})$, $(u_i, v_{j+1})$, $(u_{i+1}, v_{j+1})$ are computed as the solution to the system of 9 linear vector equations: $c_{ij9}u_r{}^2v_s{}^2 + c_{ij8}u_r{}^2v_s + c_{ij7}u_rv_s{}^2 + c_{ij6}u_r{}^2 + c_{ij5}v_s{}^2 + c_{ij4}u_rv_s + c_{ij3}u_r + c_{ij2}v_s + c_{ij1} = p_{rs}$ for $r = i-1, i, 1+1$ and $s = j - 1, j, j + 1$.

**Exercise 16.10:** Suppose we are given $nm$ points on a single-valued surface $\tilde{S}$ over a regular grid as $\{ (u_i, v_j, z_{ij}) \mid 0 \le i \le n - 1, 0 \le j \le m - 1 \}$ where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$ are given values that also define the underlying grid, and where $z_{ij} \in \mathcal{R}$ with $z_{i+1,j} > z_{ij}$ for $0 \le i < n-1$, and $0 \le j \le m-1$ and $z_{i,j+1} > z_{ij}$ for $0 \le i \le n - 1$ and $0 \le j < m - 1$. Thus the surface points are monotonically increasing in both the $u$ direction and the $v$ direction. Devise a method for estimating $u$ direction and $v$ direction tangent vectors so that the resulting two-argument tensor product cubic spline is itself monotonically increasing in both the $u$ direction and the $v$ direction. Try to extend to the case where $z_{i+1,j} > z_{ij}$ and $z_{i,j+1} < z_{ij}$, so as to obtain an interpolation surface which is simultaneously monotonically increasing in the $u$ direction and monotonically decreasing in the $v$ direction.

**Exercise 16.11:** How can we compute a two-argument smoothing spline function for given data points $\{ (u_i, v_j, z_{ij}) \mid 0 \le i \le n - 1, 0 \le j \le m - 1 \}$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$ are given values, and where $z_{ij} \in \mathcal{R}$ for $0 \le i < n$ and $0 \le j < m$, analogous to the one-argument smoothing spline introduced above?

## 16.5 Tensor Product Cardinal Basis Representation

Let $A$ be a vector space of univariate splines of range dimension 1 with respect to the knot values $u_0 \le u_1 \le \ldots \le u_{n-1}$, such that there is a unique function $f_x(u) \in A$ so that, for the fixed sequence of integers $k_0 \ge 0, k_1 \ge 0, \ldots, k_{n-1} \ge 0$, and any choice of values $x_0, x_1, \ldots, x_{n-1}$

for the components of the vector $x \in \mathcal{R}^n$, the $k_j$-th derivative $f_x^{(k_j)}$ satisfies $f_x^{(k_j)}(u_j) = x_j$ for $j = 0, 1, \ldots, n-1$. Thus the cardinal functions $f_{e_1}, \ldots, f_{e_n}$ form the associated cardinal basis for $A$. Here the subscripting vector $e_i$ is the $n$-tuple such that $(e_i)_j = \delta_{ij}$ for $1 \le j \le n$, where $\delta_{ij}$ is 1 when $i = j$ and is 0 otherwise.

Let $B$ be a vector space of univariate splines of range dimension 1 with respect to the knot values $v_0 \le v_1 \le \ldots \le v_{m-1}$, such that there is a unique function $g_y(v) \in B$ so that, for the fixed sequence of integers $h_0 \ge 0, h_1 \ge 0, \ldots, h_{m-1} \ge 0$, and any choice of values $y_0, y_1, \ldots, y_{m-1}$ for the components of the vector $y \in \mathcal{R}^m$, the $h_j$-th derivative $g_y^{(h_j)}$ satisfies $g_y^{(h_j)}(v_j) = y_j$ for $j = 0, 1, \ldots, m-1$. Thus the cardinal functions $g_{e_1}, \ldots, g_{e_m}$ form the associated cardinal basis for $B$. Here the subscripting vector $e_i$ is the $m$-tuple such that $(e_i)_j = \delta_{ij}$ for $1 \le j \le m$.

Now we may form the $n \times m$ grid of two-argument knot value pairs $\{(u_i, v_j) | 0 \le i \le n-1, 0 \le j \le m-1\}$ and the $n \times m$ derivative order matrix $\omega$ with $\omega_{ij} := k_i + h_j$ for $0 \le i \le n-1, 0 \le j \le m-1$.

Consider the $nm$ two-argument functions $f_{e_i}(u)g_{e_j}(v)$ for $1 \le i \le n, 1 \le j \le m$. These functions have the property that $f_{e_i}(u_p)g_{e_j}(v_q) = \delta_{i-1,p}\delta_{j-1,q}$. The vector space of two-argument range dimension 1 functions spanned by this two-argument cardinal basis is the *tensor product space* $A \times B$; the space $A \times B$ is composed of those two-argument splines of the form

$$\alpha(u, v) = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} \beta_{ij} f_{e_{i+1}}(u) g_{e_{j+1}}(v),$$

which satisfy

$$\alpha_{u^{(k_i)}v^{(h_j)}}(u_i, v_j) = \frac{\partial^{\omega_{ij}}\alpha}{(\partial u)^{k_i}(\partial v)^{h_j}}(u_i, v_j) = \beta_{ij}$$

for $0 \le i \le n-1$ and $0 \le j \le m-1$. The real valued function $\alpha(u, v)$ is uniquely determined by the values $\beta_{ij}$. The surface $\{\alpha(u, v) \mid u_0 \le u \le u_{n-1}, v_0 \le v \le v_{m-1}\}$ is the surface corresponding to the two-argument spline $\alpha$ defined by the $\beta_{ij}$ values.

Each component function of the bicubic spline for a single quad-patch $Q$ originally presented above belongs to the tensor product space $h \times h$ where $h$ is the space of cubic polynomial segments with respect to the knot values 0, 0, 1, 1, and the derivative orders $\langle 0, 1, 0, 1 \rangle$.

One benefit of having a basis representation for a two-argument spline is that we may, in principle, obtain an interpolation spline for *any* given data points for which associated knot value pairs are given. This can be done by forming an appropriate system of linear equations and solving them in the least squares sense.

## 16.6 Bicubic Splines with Variable Parameter Limits

Fletcher and McAllister [FM87] have proposed an extended form of the generalized parametric tensor product bicubic spline interpolation function for a rectangular patch $Q$ with the underlying grid rectangle $[0, 1] \times [0, 1]$ which permits each edge of the estimating patch $\hat{Q}$ to have a distinct associated parameter limit value. This is done by making the parameter limit value in the $u$ direction, $t_u$, be a function of $v$ and making the parameter limit value in the $v$ direction, $t_v$, be a function of $u$. For example, let the parameter limit values be denoted by $u_0$, $v_1$, $u_1$, and $v_0$; define $t_u(v) = u_0(1 - v) + u_1 v$ and define $t_v(u) = v_0(1 - u) + v_1 u$. Then the *extended generalized parametric tensor product bicubic spline interpolation function* for the rectangular patch $Q$ is

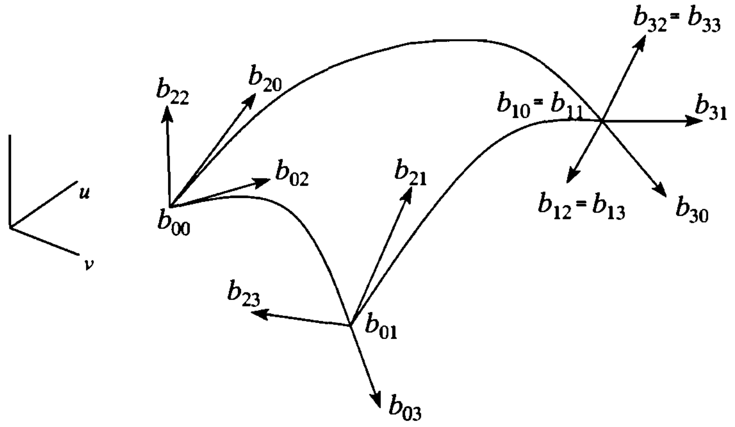
$$x_Q(u, v) = \sum_{0 \le i \le 3} f_i(t_u(v)u, t_u(v)) \sum_{0 \le j \le 3} f_j(t_v(u)v, t_v(u))b_{ij},$$

and $\hat{Q} = \{x_Q(u, v) \mid 0 \le u \le 1, 0 \le v \le 1\}$ is an estimate of the patch $Q$. The patch $\hat{Q}$ coincides with the patch $Q$ at the points $b_{00}$, $b_{10}$, $b_{01}$, and $b_{11}$, and has the same $u$- and $v$ directional derivatives there.

When we are given such a set of four surface points determining the four corners of the quad-patch to be estimated, we may choose the parameter limit values $u_0, u_1, v_0, v_1$ corresponding to the four sides of the quad-patch as the straight line distances between the pairs of corner points that define the edges of the quad-patch.

## 16.7 Triangular Patches

Note that for an extended generalized parametric tensor product bicubic spline, the control vectors and parameter limit values can be chosen so that one or more of the patch boundary curves degenerates to a single point. For example, we may choose $b_{10} = b_{11}$ with the parameter limit value $v_1$ for this edge equal to 0 so as to form a triangular patch as shown below. Generally, in this case, the tangent vectors $b_{12}$ and $b_{13}$ for the degenerate curve at the common point should be identical, as should the twist vectors $b_{32}$ and $b_{33}$. The tangent vectors $b_{30}$ and $b_{31}$, on the other hand, need not



be colinear. However the vectors $b_{12}$, $b_{13}$, $b_{30}$ and $b_{31}$ should all lie in the same (tangent) plane.

**Exercise 16.12:** What does our estimating bicubic patch look like if the parameter limit value for the degenerate edge is *not* 0?

One way to choose the three usually distinct tangent vectors at the surface point used as a "degenerate" corner in a patch is to estimate the tangent plane at the given surface point, perhaps by a nearest neighbor method; and then choose all the needed tangent vectors at the surface point to lie in the tangent plane determined for that surface point.

This ability to construct either a quadrangularly shaped surface patch (i.e., a patch with four given distinct corner points) or a triangularly shaped surface patch (i.e., a patch with three given distinct corner points) means that extended generalized tensor product bicubic spline interpolation can be used in cases where we have a collection of interpolation points in $\mathcal{R}^3$ belonging to an unknown surface such that these points can be organized as the corner points of a network of unknown possibly degenerate quad-patches to be estimated. Each possibly degenerate quad-patch is determined by four corner points from our collection of interpolation points, where at most two of these points may be identical. Each such quad-patch is assigned an underlying set of four parameter limit values, together with associated corner tangent vectors and corner twist vectors. (Generally the parameter limit value for a degenerate 0-length edge is taken to be 0.) We can then compute points on each estimating quad-patch for the network of quad-patches that make up the surface; we cannot, however, in general construct an explicit parametric surface vector valued function for this network of estimating quad-patches without introducing a globally applicable consistent bivariate parameterization.

## 16.8 Parametric Grids

In order to construct an explicit parametric function for our interpolating surface, we must establish a suitable planar domain for the parameters, upon which we will impose an appropriate grid of parameter knot value pairs. This can be easily done when our quad-patches are arranged in a regular $a \times b$ grid of patches; then the associated parameter domain is the rectangular region $[0, a] \times [0, b]$. This case often arises when we have a rectangular grid for a domain wherein we have solved for the points on a surface defined by a system of partial differential equations corresponding to the underlying grid parameter value pairs.

When we cannot establish an underlying regular grid domain for parameter knot value pairs associated with the network of possibly degenerate quad-patches, whose corners are our given interpolation points in $\mathcal{R}^3$, such that the underlying domain is partitioned into disjoint subregions corresponding one-to-one with the possibly degenerate quad-patches that make up our network, we can still construct an interpolating surface, one patch at a time. But we cannot present an associated parametric function, since we have no effective way to define an overall parameterization. Even when such an underlying grid domain is established, we may need to introduce mappings that map the domain partitions associated with individual quad-patches to rectangles for computational purposes.

In the case where we have a set of $n$ points $P = \{(u_i, v_i, f_i) \mid 1 \leq i \leq n\} \subseteq \mathcal{R}^3$ taken to be points on a single-valued surface $\tilde{S}$ defined by a function $S$ such that $S(u_i, v_i) = f_i$, we can construct an explicit domain $D$ for parameter value pairs that enables us to prescribe an explicit overall parameterization for the extended generalized bicubic spline interpolation function $x$ which defines an interpolating surface for the points $P$. In particular, the parameter domain $D$ can be chosen as the region $convexhull(\{(u_i, v_i) \mid 1 \leq i \leq n\})$. The set of points $K = \{(u_i, v_i) \mid 1 \leq i \leq n\}$ are the knot points associated with the surface points in $P$. Now let us introduce a triangulation of the points $P$; this triangulation induces a matching triangulation of the points $K$ in $D$ such that the triangles form a partition of $D$. Let $w_T$ denote a mapping which maps the triangle $T$ in $\mathcal{R}^2$ onto the unit square in

$\mathcal{R}^2$, except possibly for a segment of its boundary. We may use this family of mappings to map the triangles in $D$ to the unit square on which the basic extended generalized bicubic spline function is directly defined.

> **Exercise 16.13:** Explicitly state the mapping $w_T$ postulated above. *Hint*: consider barycentric coordinates for $\mathcal{R}^2$ with respect to the triangle $T$. Use this mapping to explicitly state the extended generalized bicubic spline interpolating function $x$ defined on $D$ such that $x(u_i, v_i) = f_i$ for $1 \le i \le n$.

## 16.9 3D-Function Interpolation

Heretofore we have studied the general case of constructing an interpolating surface for an arbitrary set of points in $\mathcal{R}^3$. Just as in the $\mathcal{R}^2$ case, in order to specify an explicit parametric vector valued interpolation function whose graph is the desired interpolating surface, it is necessary to establish a domain for parameter values and an assignment of particular parameter points (knots) to the given surface points. In the $\mathcal{R}^3$ case, this imposes a definite organization of the given surface points as the corners of a definite network of possibly degenerate quad patches. For points in $\mathcal{R}^2$, it is relatively straightforward to devise effective ways to assign suitable parameter values to the data points. In $\mathcal{R}^3$, we want to assign suitable parameter pairs to the data points, but it is harder to devise effective general schemes for doing this assignment. Although this can be difficult to do, it is an unavoidable requirment if we want to construct an explicit complete parametric interpolation function.

Let $x : \mathcal{R}^2 \to \mathcal{R}^3$ be a parametric bicubic spline. Note that each of the real valued two-argument component functions $x_1, x_2$, and $x_3$ is a non-parametric bicubic spline function for a 3D single valued surface. But the necessary parameter points to associated with $x_1, x_2$ and $x_3$ are missing.

When, however, we do have points taken from the graph of a non-parametric function $f$ mapping $\mathcal{R}^2$ to $\mathcal{R}$, the parameter knot value pairs we need are given *a priori* as the $x$ and $y$ coordinates of the data points. However, the regular grid structure we desire may be lacking. We can easily construct a suitable regular grid of parameter value pairs by using the cross-product of the set of distinct $x$ values and the set of distinct $y$ values, but then we may be faced with estimating values of the unknown function $f$ at various generated grid points as needed to obtain a complete set of surface points associated with our regular grid of parameter value pairs.

Let us suppose we are given the set of $p$ points $P := \{(x_k, y_k, g_k) \mid 0 \le k \le p-1\}$. Let $u_0 < u_1 < \cdots < u_{n-1}$ be the distinct values among $x_0, \ldots, x_{p-1}$ and let $v_0 < v_1 < \cdots < v_{m-1}$ be the distinct values among $y_0, \ldots, y_{p-1}$. Then $G = \{(u_i, v_j) \mid 0 \le i \le n-1, 0 \le j \le m-1\}$ is a regular grid of parameter value pairs such that every parameter value pair in $\{(x_k, y_k) \mid 0 \le k \le p-1\}$ is found in $G$.

We want to construct $nm$ points on a single-valued surface, namely: $\{(u_i, v_j, z_{ij}) \mid 0 \le i \le n-1, 0 \le j \le m-1\}$, where $u_0 < u_1 < \cdots < u_{n-1}$ and $v_0 < v_1 < \cdots < v_{m-1}$ are the given values that define the underlying grid $G$, and where $z_{ij} \in \mathcal{R}$. Of these $nm$ 3-tuple points, $p$ of the $nm$ $z_{ij}$ values are known to be the originally-given $g_k$ values. The remaining $nm - p$ $z_{ij}$ values are not known and must somehow be estimated. Once we have obtained these estimates, we have a set of points on a surface together with a regular grid of parameter value pairs from which we can construct an interpolating spline function.

Our problem is now reduced to the particular interpolation problem of estimating values of an unknown real valued function $f$ defined on $\mathcal{R}^2$ corresponding to particular parameter pairs in the grid $G$, where we are given the particular points $P$ contained in the graph of the unknown function $f$. Given these estimated function values, we can then further estimate slopes and mixed partial values and construct a bicubic spline interpolation function for the data points given in $P$. There are a variety of ways to approach this problem.

One effective way to estimate a value of $f$ at a parameter point $q \in \mathcal{R}^2$ is to compute the desired estimate as a weighted average of the function values associated with the $h$ nearest neighbors of the point $q$ in the parameter pair set $\{(x_k, y_k) \mid 0 \le k \le p-1\}$.

**Exercise 16.14:** Propose a suitable weighting scheme for the $h$ function values associated with the nearest neighbor parameter pairs of the point $q$.

After estimating the unknown $z_{ij}$ values on the grid $G$, we can obtain smoother estimates by computing smoothing splines in the $x$ direction and in the $y$ direction, and then replacing each estimated value associated with a parameter value pair $(u, v) \in G$ by the average of the two values that predict the unknown function value at $(u, v)$ which are generated on the two smoothing splines that we expect to "intersect" at the point determined by $(u, v)$.

# 17 Boundary Curve Based Surface Splines

The device of sweeping a cubic polynomial space curve through space while simultaneously deforming it so as to match a second cubic polynomial space curve at the end of the sweep can be generalized so that we can sweep and deform *any* given initial boundary curve so that it matches any given final boundary curve at the end of the sweeping process. This sweeping can be characterized as forming intermediate curves by interpolating between each corresponding pair of points on the initial and final curves. A. R. Forrest gives a thorough review of this methodology in [For72]; there much of this work is credited to Steven Coons.

## 17.1 Boundary Curve Based Bilinear Interpolation

Suppose we are given the boundary space curves $x_0(u) = S(u, 0), x_1(u) = S(u, 1), y_0(v) = S(0, v)$, and $y_1(v) = S(v, 1)$ for the quad patch $Q$ of the surface $\tilde{S} = \{S(u, v) \mid u, v \in \mathcal{R}\}$ with the corner points $b_{00}, b_{10}, b_{11}$, and $b_{01}$ in cyclic order, where the parametric surface function $S : \mathcal{R}^2 \to \mathcal{R}^3$ is such that $x_0(0) = b_{00}, x_0(1) = b_{10}, x_1(0) = b_{01}, x_1(1) = b_{11}, y_0(0) = b_{00}$, $y_0(1) = b_{01}, y_1(0) = b_{10}$, and $y_1(1) = b_{11}$.

Consider the so-called *ruled* surface formed by connecting each pair of points $x_0(u)$ and $x_1(u)$ on the opposite boundary curves $x_0$ and $x_1$ of the patch $Q$ with a straight line; this surface is defined by the parametric function $S_x(u, v) = x_0(u)(1-v)+x_1(u)v$. Note $S_x$ maps $\mathcal{R}^2$ into $\mathcal{R}^3$. The ruled surface $\{S_x(u, v) \mid u, v \in \mathcal{R}\}$ is called the *linearly lofted surface* between the curves $x_0$ and $x_1$. Similarly, the linearly lofted ruled surface between the curves $y_0$ and $y_1$, formed by connecting each pair of points $y_0(v)$ and $y_1(v)$ on the opposite boundary curves $y_0$ and $y_1$ of $Q$ with a straight line, is defined by the parametric function $S_y(u, v) = y_0(v)(1-u) + y_1(v)u$.

When the boundary curves $x_0, x_1, y_0$, and $y_1$ are themselves straight lines, the linearly lofted surfaces defined by $S_x$ and $S_y$ both coincide with the so-called *bilinear interpolation surface* for the patch Q with the corner points $b_{00}, b_{10}, b_{11}$, and $b_{01}$. The bilinear interpolation surface is defined by the parametric function: $S_L(u, v) = b_{00}(1-u)(1-v) + b_{10}u(1-v) + b_{01}(1-u)v + b_{11}uv$.

**Exercise 17.1:** Draw the partition of the unit square defined by the vertical and horizontal lines through the point $(u, v)$ and identify the areas corresponding to the points $b_{00}, b_{10}, b_{01}$, and $b_{11}$ in the bilinear interpolation function $S_L(u, v)$.

Now, if we examine the sum $S_x + S_y$ of the linearly lofted surface functions $S_x$ and $S_y$, we see that $S_y$ contributes the straight line segments $segment[x_0(0), x_0(1)]$ and $segment[x_1(0), x_1(1)]$, together with the curves $\{y_0(v) \mid 0 \le v \le 1\}$ and $\{y_1(v) \mid 0 \le v \le 1\}$, to the boundary of the surface patch $\Sigma := \{S_x(u, v) + S_y(u, v) \mid 0 \le u \le 1, 0 \le v \le 1\} \subset \mathcal{R}^3$; $S_x$ contributes the straight line segments $segment[y_0(0), y_0(1)]$ and $segment[y_1(0), y_1(1)]$, together with the curves $\{x_0(u) \mid 0 \le u \le 1\}$ and $\{x_1(u) \mid 0 \le u \le 1\}$, to the boundary of the surface patch $\Sigma$. Thus, subtracting the bilinear interpolation surface function $S_L$ from $S_x+S_y$ removes the straight line segments of the boundary of the patch $\Sigma$ contributed by $S_x$ and $S_y$, and leaves the patch $\{S_x(u, v) + S_y(u, v) - S_L(u, v) \mid 0 \le u \le 1, 0 \le v \le 1\}$, which has the boundary $\{x_0(u) \mid 0 \le u \le 1\} \cup \{y_0(v) \mid 0 \le v \le 1\} \cup \{x_1(u) \mid 0 \le u \le 1\} \cup \{y_1(v) \mid 0 \le v \le 1\}$, which is the same as the boundary postulated for the patch $Q$. The parametric function $S_B(u, v) = S_x(u, v) + S_y(u, v) - S_L(u, v)$ is called the *bilinearly blended patch interpolation function* for the patch $Q$ with the boundary $\{x_0(u) \mid 0 \le u \le 1\} \cup \{y_0(v) \mid 0 \le v \le 1\} \cup \{x_1(u) \mid 0 \le u \le 1\} \cup \{y_1(v) \mid 0 \le v \le 1\}$.

Let $h(t, a, b, c, d) = f_0(t)a + f_1(t)b + f_2(t)c + f_3(t)d$, where $f_0(t) = 1-3t^2+2t^3$, $f_1(t) = 3t^2-2t^3$, $f_2(t) = t-2t^2+t^3$, and $t_3(t) = -t^2+t^3$; $h(t, a, b, c, d)$ is the Hermite cubic interpolating polyomial space curve segment based on the control vectors $a$, $b$, $c$, and $d$, where $c$ is the tangent vector at $a$, and $d$ is the tangent vector at $b$. The bilinearly-blended patch interpolation function for the boundary $\{x_0(u) \mid 0 \le u \le 1\} \cup \{y_0(v) \mid 0 \le v \le 1\} \cup \{x_1(u) \mid 0 \le u \le 1\} \cup \{y_1(v) \mid 0 \le v \le 1\}$, in the case where $x_0(u)$ is the cubic spline curve $h(u, b_{00}, b_{10}, b_{20}, b_{30})$, $x_1(u)$ is the cubic spline curve $h(u, b_{01}, b_{11}, b_{21}, b_{31})$, $y_0(v)$ is the cubic spline curve $h(v, b_{00}, b_{01}, b_{02}, b_{03})$, and $y_1(v)$ is the cubic spline curve $h(v, b_{10}, b_{11}, b_{12}, b_{13})$, is identical to the tensor product bicubic patch interpolation function for the control points $b_{00}$, $b_{10}$, $b_{01}$, $b_{11}$, $b_{20}$, $b_{02}$, $b_{21}$, $b_{12}$, $b_{30}$, $b_{03}$, $b_{22}$, $b_{31}$, $b_{13}$, $b_{32}$, $b_{23}$, $b_{33}$, with $b_{22} = b_{32} = b_{23} = b_{33} = 0$.

## 17.2 Boundary Curve Based Bicubic Interpolation

Suppose we are given *cross boundary tangent vectors* at each point on the boundary of the patch $Q$ of the surface $\tilde{S}$ in addition to the boundary curves $S(u, 0), S(0, v), S(u, 1), S(1, v)$. Thus, suppose we are given the partial derivative space curves $S_v(u, 0)$, $S_v(u, 1)$, $S_u(0, v)$, and $S_u(1, v)$ which specify the cross boundary tangent vectors along the boundary curves $S(u, 0)$,$S(u, 1)$, $S(0, v)$, and $S(1, v)$, respectively. We can then form the *cubically lofted* surfaces between the curves $S(u, 0)$ and $S(u, 1)$, and between the curves $S(0, v)$ and $S(1, v)$. If we sum the functions that define these two surfaces and correct the sum by subtracting the tensor product bicubic patch interpolation surface function $x_Q$ for the control points $b_{00} = S(0, 0)$, $b_{10} = S(1, 0)$, $b_{01} = S(0, 1)$, $b_{11} = S(1, 1)$, $b_{20} = S_u(1, 0)$, $b_{02} = S_v(1, 0)$, $b_{21} = S_u(0, 1)$, $b_{12} = S_v(1, 0)$, $b_{30} = S_u(1, 0)$, $b_{03} = S_v(0, 1)$, $b_{22} = S_{uv}(0, 0)$, $b_{31} = S_u(1, 1)$, $b_{13} = S_v(1, 1)$, $b_{32} = S_{uv}(1, 0)$, $b_{23} = S_{uv}(0, 1)$, and $b_{33} = S_{uv}(1, 1)$, we will obtain the *bicubically blended patch interpolation* function for the patch $Q$[Gor69].

We will define the cubically lofted surface between the boundary curves given by $S(u, 0)$ and $S(u, 1)$ via the cubic spline blending functions $f_0(t) = 1 - 3t^2 + 2t^3$, $f_1(t) = 3t^2 - 2t^3$, $f_2(t) = t - 2t^2 + t^3$, and $f_3(t) = -t^2 + t^3$, as $\{S_x(u, v) \mid u, v \in \mathcal{R}\}$ where $S_x(u, v) = f_0(v)S(u, 0) + f_1(v)S(u, 1) + f_2(v)S_v(u, 0) + f_3(v)S_v(u, 1)$. Similarly, the cubically lofted surface between the boundary curves given by $S(0, v)$ and $S(1, v)$ is defined as $\{S_y(u, v) \mid u, v \in \mathcal{R}\}$ where $S_y(u, v) = f_0(u)S(0, v) + f_1(u)S(1, v) + f_2(u)S_u(0, v) + f_3(u)S_u(1, v)$. Thus the bicubically blended patch interpolation function for the boundary $\{S(u, 0) \mid 0 \le u \le 1\} \cup \{S(1, v) \mid 0 \le v \le 1\} \cup \{S(u, 1) \mid 0 \le u \le 1\} \cup \{S(1, v) \mid 0 \le v \le 1\}$ with the cross boundary tangent vector functions $S_u(0, v)$, $S_u(1, v)$, $S_v(u, 0)$, and $S_v(u, 1)$ is $S_B(u, v) := S_x(u, v) + S_y(u, v) - x_Q(u, v)$. The surface

$\{S_B(u, v) | 0 \le u \le 1, 0 \le v \le 1\}$ is often called a *Coons patch* in computer graphics [Coo67].

**Exercise 17.2:** Show that the cubically lofted surface functions $S_x$ and $S_y$ are both identical to the tensor product bicubic patch interpolation function $x_Q$ for the control points $b_{00}, b_{10}, b_{01}, b_{11}, b_{20}, b_{02}, b_{21}, b_{12}, b_{30}, b_{03}, b_{22}, b_{31}, b_{13}, b_{32}, b_{23}, b_{33}$ in the case where the boundary curves and the cross boundary tangent vectors that define the cubically lofted functions $S_x$ and $S_y$ are determined by cubic spline segment functions as $S(u, 0) = h(u, b_{00}, b_{10}, b_{20}, b_{30})$, $S(u, 1) = h(u, b_{01}, b_{11}, b_{21}, b_{31})$, $S(0, v) = h(v, b_{00}, b_{01}, b_{02}, b_{03})$, $S(1, v) = h(v, b_{10}, b_{11}, b_{12}, b_{13})$, $S_v(u, 0) = h(u, b_{02}, b_{12}, b_{22}, b_{32})$, $S_v(u, 1) = h(u, b_{03}, b_{13}, b_{23}, b_{33})$, $S_u(0, v) = h(v, b_{20}, b_{21}, b_{22}, b_{23})$, and $S_u(1, v) = h(v, b_{30}, b_{31}, b_{32}, b_{33})$.

**Exercise 17.3:** Show that the bicubically blended patch interpolation function for the boundary $\{S(u, 0) \mid 0 \le u \le 1\} \cup \{S(1, v) \mid 0 \le v \le 1\} \cup \{S(u, 1) \mid 0 \le u \le 1\} \cup \{S(1, v) \mid 0 \le v \le 1\}$ formed of cubic polynomial space curves with $S(u, 0) = h(u, b_{00}, b_{10}, b_{20}, b_{30})$, $S(u, 1) = h(u, b_{01}, b_{11}, b_{21}, b_{31})$, $S(0, v) = h(v, b_{00}, b_{01}, b_{02}, b_{03})$, and $S(1, v) = h(v, b_{10}, b_{11}, b_{12}, b_{13})$, and with the cross boundary tangent vector functions $S_u(0, v)$, $S_u(1, v)$, $S_v(u, 0)$, and $S_v(u, 1)$ defined as the cubic polynomial space curves $S_v(u, 0) = h(u, b_{02}, b_{12}, b_{22}, b_{32})$, $S_v(u, 1) = h(u, b_{03}, b_{13}, b_{23}, b_{33})$, $S_u(0, v) = h(v, b_{20}, b_{21}, b_{22}, b_{23})$, and $S_u(1, v) = h(v, b_{30}, b_{31}, b_{32}, b_{33})$, is identical to the tensor product bicubic patch interpolation function $x_Q$ for the control points $b_{00}$, $b_{10}$, $b_{01}, b_{11}, b_{20}, b_{02}, b_{21}, b_{12}, b_{30}, b_{03}, b_{22}, b_{31}, b_{13}, b_{32}, b_{23}, b_{33}$.

**Exercise 17.4:** Reprise the above material and develop the *generalized* bicubically blended patch interpolation function for the patch $Q$ with the underlying grid rectangle $[0, t_u] \times [0, t_v]$ replacing $[0, 1] \times [0, 1]$.

**Exercise 17.5:** If the boundary curves that crisscross a regular network of patches associated with a regular grid of knot value pairs are $C^2$ functions, is each component of the resulting overall bicubically blended interpolation function a $C^2$ function of two arguments? *Hint*: look at the derivatives of the Hermite blending functions at $t = 1$.

## 17.3 General Boundary Curve Based Spline Interpolation

General blended two-argument splines can be defined in terms of cardinal basis functions, analogous to the way that such expressions were found for general tensor product two-argument splines. Let $f_{e_1}, \ldots, f_{e_n}$ be the cardinal functions for a vector space of splines of range dimension 1 with respect to knot values $r_0 \le r_1 \le \ldots \le r_{n-1}$ and derivative orders $\langle k_0, \ldots, k_{n-1}\rangle$. Similarly, let $g_{e_1}, \ldots, g_{e_m}$ be the cardinal functions for a vector space of splines of range dimension 1 with respect to knot values $s_0 \le s_1 \le \ldots \le s_{m-1}$ and derivative orders $\langle h_0, \ldots, h_{m-1}\rangle$.

Let $F_n$ denote the projection operation corresponding to the cardinal functions $f_{e_1}, \ldots, f_{e_n}$, that is, $(yF_n)(t) = f_{e_1}(t)y^{(k_0)}(r_0) + f_{e_2}(t)y^{(k_1)}(r_1) + \ldots + f_{e_n}(t)y^{(k_{n-1})}(r_{n-1})$. Note $F_n$ maps real valued univariate functions to range dimension 1 univariate splines with knot values $r_0 \le \ldots \le r_{n-1}$.

Let $G_m$ denote the projection operation corresponding to the cardinal functions $g_{e_1}, \ldots, g_{e_m}$, that is, $(yG_m)(t) = g_{e_1}(t)y^{(h_0)}(s_0) + g_{e_2}(t)y^{(h_1)}(s_1) + \ldots + g_{e_m}(t)y^{(h_{m-1})}(s_{m-1})$. Note $G_m$ maps real valued univariate functions to range dimension 1 univariate splines with knot values $s_0 \le \ldots \le s_{m-1}$.

Now given the space curves $S_{v^{(h_0)}}(u, s_0), \ldots, S_{v^{(h_{m-1})}}(u, s_{m-1})$ and also $S_{u^{(k_0)}}(r_0, v), \ldots, S_{u^{(k_{n-1})}}(r_{n-1}, v)$, which we imagine to be derived from an unknown surface function $S$, let $S_2(v) = S(u, v)$; we treat $u$ as an auxillary parameter in $S_2$. The *$F_n$-lofted surface* corresponding to the given space curves is defined by the vector valued function $S_x(u, v)$ where

$$S_x(u, v) := [S_2F_n](u, v) = f_{e_1}(u)S_{u^{(k_0)}}(r_0, v) + \ldots + f_{e_n}(u)S_{u^{(k_{n-1})}}(r_{n-1}, v).$$

We thus extend the projection operator $F_n$ to apply to the vector valued two-argument function $S$ by taking $S$ as a function of its *second* argument, so that $SF_n := S_2F_n$.

Let $S_1(u) = S(u, v)$; we treat $v$ as an auxillary parameter in $S_1$. The *$G_m$-lofted surface* corresponding to the given space curves is defined by the vector valued function $S_y(u, v)$ where

$$S_y(u, v) := [S_1G_m](u, v) = g_{e_1}(v)S_{v^{(h_0)}}(u, s_0) + \ldots + g_{e_m}(v)S_{v^{(h_{m-1})}}(u, s_{m-1}).$$

We thus extend the projection operator $G_m$ to apply to the vector valued two-argument function $S$ by taking $S$ as a function of its *first* argument, so that $SG_m := S_1G_m$.

Recall that the two-argument generalized parametric tensor product spline for the control points

$S_{u^{(k_i)}v^{(h_j)}}(r_i, s_j)$ with $0 \le i \le n-1$ and $0 \le j \le m-1$ is:

$$S_{xy}(u, v) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} f_{e_{i+1}}(u) g_{e_{j+1}}(v) S_{u^{(k_i)}v^{(h_j)}}(r_i, s_j).$$

Note that $S_{xy}(u, v) = [(SG_m)F_n](u, v) = [(SF_n)G_m](u, v)$, i.e., the projection operators $F_n$ and $G_m$ *commute* when applied to a two-argument function.

Now, the *$(F_n, G_m)$-blended interpolation function* for the boundary curves $S_{v^{(h_0)}}(u, s_0)$, $S_{v^{(h_1)}}(u, s_1)$, $\ldots$, $S_{v^{(h_{m-1})}}(u, s_{m-1})$ and $S_{u^{(k_0)}}(r_0, v)$, $S_{u^{(k_1)}}(r_1, v)$,$\ldots$, $S_{u^{(k_{n-1})}}(r_{n-1}, v)$ is

$$S_B(u, v) = S_x(u, v) + S_y(u, v) - S_{xy}(u, v) = [S(F_n + G_m - F_n G_m)](u, v).$$

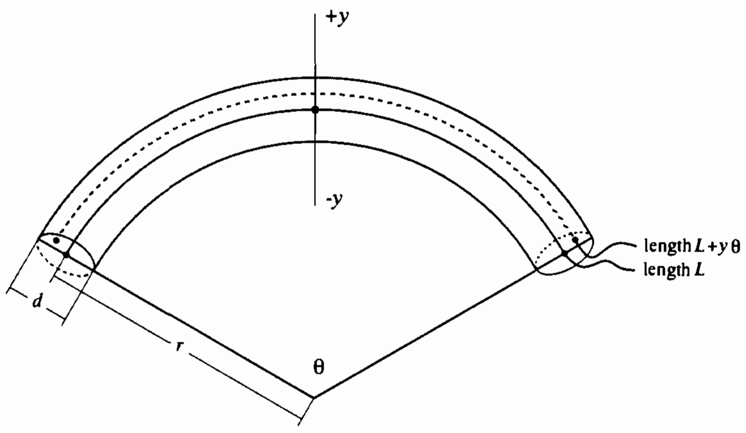
When we can construct join order 2 cardinal functions $f_{e_1}, \ldots, f_{e_n}$, and $g_{e_1}, \ldots, g_{e_m}$, note that the resulting $(F_n, G_m)$-blended interpolation function $S_B$ is also a join order 2 bivariate function. Since we have presented several bases for spaces of join order 2 splines, we can explicitly construct such a join order 2 bivariate interpolation function. For example, we can use the B-spline basis functions for $n$ knot values, and for $m$ knot values, with specific fixed end conditions imposed, and write the explicit computational recipes for $f_{e_1}, \ldots, f_{e_n}$ and $g_{e_1}, \ldots, g_{e_m}$ in this case. When we use B-spline bases, these recipes involve solving $n + m$ systems of linear equations.

# 18 Physical Splines

Let $p_1, \ldots, p_n \in \mathcal{R}^3$. A flexible springy wire of length $h$ pinned at the end points $p_1$ and $p_n$, and passing through the points $p_2, \ldots, p_{n-1}$ with or without specified directions is physically determined to have one of several stable shapes, each corresponding to a locally minimal energy value. We may seek these minimal energy shapes; such a curve will be called a *physical spline* curve. Although a cubic spline may be a good approximation to a physical spline, this is often not the case, and moreover the length constraint applied to a cubic spline segment is difficult to honor. It is important in various engineering applications to be able to compute the exact physical spline curve of a given length that satisfies given boundary conditions.

Our first step is to compute the energy in a thin circular cross section physical spline bent in the shape of an arc length parametrized space curve $x$ of length $h$. Malcolm [Mal77] reports that this idea goes back to Daniel Bernoulli circa 1742. We will model the spline by a sequence of $n$ segments bent in circular arcs and adjoined end-to-end to approximate the space curve $x$. Each segment is of length $L = h/n$. A segment is modeled by a bundle of elastic length $L$ fibers arranged to form a cylinder of length $L$. Each fiber behaves as a spring that is compressed or extended in length with a force proportional to the change in its length according to Hooke's law.

As each cylindrical segment of fibers is bent into a circular arc, the length



of the central fiber is unchanged in length, the fibers closer to the inside of the arc are compressed, and the fibers on the outside of the arc are extended. The potential energy stored in all these compressed and extended fibers is the energy of the segment, and the sum of the energies of the $n$ segments approximates the energy in the spline bent to follow the space curve $x$. When we consider the limiting case where $n \to \infty$, we obtain the desired energy of the physical spline; this will also be defined to be the energy of the space curve $x$.

The potential energy $e$ in a length $L$ spring fiber compressed or extended to the length $L+\Delta$ is the work done to change the length from the 0-energy length $L$ to the length $L + \Delta$. This work is the quantity $\int_L^{L+\Delta} F(x)\,dx$ where $F(x)$ is the force needed to compress or extend the spring to length $x$. By Hooke's law $F(x) \approx \varepsilon(x-L)/L$ when $x$ is not too different from $L$, where $\varepsilon$ is the modulus of elasticity of the spring material. Thus the energy $e$ is $\frac{\varepsilon}{2L}\Delta^2$.

Now let us consider the length $L$ cylindrical segment of fibers bent in a circular arc of a circle of radius $r$ as shown above. The origin is shown placed at the center of the cylindrical segment.

The central fiber of length $L$ is shown. A coordinate axis is established by which we may measure the distance $y$ above or below the central fiber. A representative non-central fiber of length $L + \theta y$ is also shown.

The values $r$, $L$, and $\theta$ are related by $r\theta = L$. The curvature of the central fiber is $1/r$. In general, the length of a fiber offset vertically by the distance $y$ is $(r + y)\theta$. The energy of the fiber offset vertically by the distance $y$ is thus $\varepsilon(y\theta)^2/(2L)$.

Let $A$ denote the circular disk of diameter $d$ centered at $(0, 0)$ in the $xy$-plane. The total energy stored in the entire cylindrical segment is now obtained as the integral

$$\int_A \varepsilon(y\theta)^2/(2L)\,dxdy = \frac{\varepsilon}{2L}\theta^2 \int_A y^2\,dxdy.$$

But $\int_A y^2\,dxdy$ is a constant which we call $I$; in fact $I$ is the so-called moment of inertia of the disk $A$ about the $x$-axis. Thus the energy in our bent segment is $\frac{\varepsilon I}{2L}\theta^2 = \frac{\varepsilon I}{2} \cdot \frac{L}{r^2}$. Since $1/r$ is the curvature of the central fiber, we may write the energy of the segment as $\frac{\varepsilon I}{2} \cdot Lk^2$, where $k = 1/r$ denotes the curvature of the segment.

Now define the arc length parameter value $s_i := (i - 1)L + L/2$. Let $K(s)$ denote the curvature $|x''(s)|$ of our given arc length parametrized space curve $x$ at $s$.

Place $n$ length $L$ cylindrical segments along the space curve $x$ with the $i$th segment centered at $x(s_i)$. The $i$th segment is taken to be bent in a circular arc whose curvature matches the curvature value $K(s_i)$. When $n$ is large, the circular end faces of the adjacent segments will approximately join.

The total potential energy $E$ in all $n$ segments is then

$$\sum_{1 \le i \le n} \frac{\varepsilon I}{2}(K(s_i))^2 L.$$

Now, if we let $n \to \infty$, then $L \to 0$ such that $nL$ remains equal to the curve length $h$, and the summation expression for $E$ is seen to be a Riemann sum which converges to the integral $\frac{\varepsilon I}{2}\int_0^h K(s)^2\,ds$. Thus the energy in a diameter $d$ circular cross section physical spline bent to follow the space curve $x$ is $\frac{\varepsilon I}{2}\int_0^h K(s)^2\,ds$. We see that a physical spline with an infinitesimal cross section diameter has an infinitesimal energy; however we shall depart from physical reality and *assign* the value $\int_0^h K(s)^2\,ds$ to be the mathematical energy of the length $h$ space curve $x$.

Note that the shapes of two physical splines composed of circular cross section wires of two different materials that have the same boundary conditions will be essentially identical, although the actual energy in each will be different due to their differing moduli of elasticity. Also note that the *torsion* of $x$ does not directly affect the potential energy stored in $x$ as would be the case for a physically realized bundle of fibers bent to follow the curve $x$. This is because we assigned no energy budget to account for joining the length $L$ circular arc shaped segments in a non-planar (twisted) fashion in the construction given above. This is appropriate for a small cross section diameter spline, since the energy due to torsion is negligible as the cross section diameter approaches zero. Of course, the torsion can adversely affect the curvature, so the torsion is implicitly constrained in a minimal energy curve.

**Exercise 18.1:** What is the potential energy in a diameter $d$ circular cross section physical spline of length $h$ whose central fiber follows the space curve $x$, taking the torsion of $x$ into account?

Note that a ruled surface composed of a rectangle of stiffly flexible material which admits a family of parallel rule lines, bent according to uniform constant boundary conditions along two opposing edges formed by rule lines assumes the same shape in any individual cross section curve. That shape is the shape of a (planar) physical spline curve determined by the distance between the two opposing edges and the slope boundary conditions at those edges.

**Exercise 18.2:** How can we define the energy of a finite area surface $S$?

**Solution 18.2:** One possiblity is to let $e_p(\theta)$ be the energy of a curve embedded in the surface $S$ within a small disk $D_p(r)$ of radius $r$ centered at the point $p$, where the curve passes through $p$ in the direction $\theta$. $f_p(r) := \frac{1}{2\pi}\int_0^{2\pi} e_p(\theta)d\theta$ is the total energy of the surface $S$ in the disk $D_p(r)$. Then $\int_{p\in S} \lim_{r\to 0} f_p(r)$ is the energy of the surface $S$.

Alternatively, we may define the energy as the combination of all the energies of all the parallel curves in some fixed direction that are embedded in the surface and whose union form the surface, together with the energies of all the parallel surface curves in the orthogonal direction. This solution admits a physically based model formed with a rectangular grid of springs and hence is to be preferred. Various models have been proposed for defining the energy of a surface; however, unlike a curve, the energy depends on the modes of bending and distortion we shall admit that are applied to construct our surface, and the problem is more complex than that of defining the energy in a curve.

Recall that the curvature of a real valued function $f : \mathcal{R}^1 \to \mathcal{R}^1$ is $|f''|/(1+(f')^2)^{\frac{3}{2}}$. When $f'$ is small, the curvature of $f$ is approximated by $|f''|$. Since a cubic polynomial that passes through two given points with given slopes minimizes $\int (f'')^2$, we see that when $f'$ is small, a cubic polynomial spline segment is a good approximation to a physical spline $x$ that satisfies the same boundary conditions and minimizes $\int K^2$ where $K$ is the curvature of $x$. When $f'$ is not small, however, this approximation can be very poor.

Now we may consider the problem of computing the shape of the physical spline of length $h$ which interpolates the given points $p_1, p_2, \ldots, p_n$ in this order. We may optionally impose tangent vector direction constraints at some or all of these points. The desired space curve $x$ can be defined as that curve which has minimal energy, subject to the required constraints. In the simplest case, $x$ is to be chosen so that the functional

$$\mathcal{E}(x) := \min_{\substack{t_1+\cdots+t_{n-1}=h \\ t_j \geq |p_{j+1}-p_j| \text{ for } 1\leq j<n}} \sum_{1\leq i\leq n-1} \int_{a_{i-1}}^{a_i} |x''(s)|^2 \, ds$$

is minimal, subject to the constraints: $t_0 = 0$, $a_i = t_0 + \cdots + t_i$ and $x(a_{i-1}) = p_i$ for $0 \leq i \leq n-1$, and $\int_0^v |x'(s)| \, ds = v$ for $0 \leq v \leq h$, or equivalently, $|x'(s)| = 1$ for $0 \leq s \leq h$. This latter set of constraints forces $x$ to be an arc length parameterized space curve of overall length $h$ for which the curvature $K(s) = |x''(s)|$. If a tangent vector constraint that $x'(a_i) = m_i/|m_i|$ is to be imposed, then it must be added to the primary set of constraints just given.

We may drop the constraint that $x$ is an arc length parameterized curve by using the general formulation $K(s) = [|x'|^2|x''|^2 - (x', x'')^2]^{1/2}/|x'|^3$. However then the other constraints that define $a_0, \ldots, a_{n-1}$ must be correspondingly reformulated and the constraints that specify the arc length of $x$ between $p_{j+1}$ and $p_j$ to be $t_j$ must be added.

Each space curve $x$ corresponding to a local minimum of $\mathcal{E}$ is a stable shape for the length $h$ physical spline interpolating the points $p_1, \ldots, p_n$. Every initial space curve in a small-enough neighborhood of $x$ as defined by the functional $\mathcal{E}$ will relax to the shape $x$ with an accompanying loss of energy.

**Exercise 18.3:** Given an arc length parameterized space curve $x$ of length $h$, deduce the energy stored in $x$ as follows. Consider a sequence of $n$ points $p_1, \ldots, p_n$ equidistantly positioned along $x$, so that $|p_{i+1} - p_i|$ is a constant $L$ for $i = 1, \ldots, n-1$. The piecewise linear interpolating curve $\hat{x}$ that interpolates the points $p_1, \ldots, p_n$ approximates the space curve $x$. Imagine that a spring $S_i$ is attached connecting the point $(p_i + p_{i-1})/2$ with the point $(p_{i+1} + p_i)/2$ for $2 \leq i \leq n-1$.

The natural uncompressed length of each such spring is $L$. The curve $x$ may bend so that the angle $angle(p_{i-1}, p_i, p_{i+1})$ at $p_i$ is smaller than $\pi$, and then the spring $S_i$ is compressed. Define the energy of the piecewise linear curve $\hat{x}$ to be the sum of the potential energies stored in the springs $S_2, \ldots, S_{n-1}$. Take the limit as $n \to \infty$ to obtain the energy of the space curve $x$.

There are several approaches to computing the physical spline space curve $x$ that minimizes $\mathcal{E}(x)$. One approach, due to Mehlum [Meh69], [Meh74], is to define $x$ by its curvature and torsion functions via the Frenet equations, and then to deduce defining conditions on the curvature $K$ and the torsion $T$ which reduce the problem to that of estimating a collection of scalars that minimize the energy of a curve defined in terms of an instance of the set of Frenet differential equations, together with the equations for the curvature and torsion functions obtained from the Euler differential equations that define $x$ via the calculus of variations. Mehlum's equations are discussed in detail below.

Yet another approach is to approximate $x$ by a cubic spline curve $\hat{x}$ made up of $m - 1$ cubic polynomial segments[Mal77]. Write $\hat{x} = \alpha_1\phi_1 + \ldots + \alpha_d\phi_d$ where $\langle \phi_1, \ldots, \phi_d \rangle$ is a basis for the chosen space of splines containing $\hat{x}$. Then we may compute $\hat{x}$ by computing $\alpha_1, \ldots, \alpha_d$ as the solution to the equations: $\partial\mathcal{E}(\alpha_1\phi_1 + \ldots + \alpha_d\phi_d)/\partial\alpha_i = 0$ for $1 \leq i \leq d$ subject to the associated arc length constraint.

For both this latter method and Mehlum's approach, we can only proceed when we have the reduced problem where we have just two points ($n = 2$) to be connected by a physical spline curve $x$ of length $h$. Nevertheless, when the basic 2-point problem is solved, we can, in principle, compute the interpoint lengths $t_1, \ldots, t_{n-1}$ which partition $h$ and minimize $\mathcal{E}$ by solving the $n - 1$ derivative equations $\partial\mathcal{E}(x)/\partial t_i = 0$, where $x$ is piecewise computable; this requires that many subsidiary solutions of the basic two-point problem be generated, so the entire process is extremely computationally demanding.

## 18.1 Computing a Space Curve Physical Spline Segment

To derive Mehlum's equations for a length $h$ physical spline, let $x(s) = (x_1(s), x_2(s), x_3(s))$ be the minimal energy arc length parameterized space curve such that $x(0) = a$, $x(h) = b$, $x'(0) = c$, and $x'(h) = d$ where

$a, b, c, d \in \mathcal{R}^3$ and $|c| = |d| = 1$. Then $x$ must satisfy $|x'(s)| = 1$ for $0 \le s \le h$ and must minimize $\int_0^h |x''| ds$.

Let us associate the Lagrange multiplier $\lambda(s)$ with the constraint $|x'(s)| = 1$; thus the ensemble of $\lambda(s)$ values constitute a Lagrange multiplier *function* $\lambda : [0, h] \to \mathcal{R}$, which we take to be continuously differentiable. Then the physical spline $x$ minimizes the functional $\int_0^h F(g, g', g'')ds$ among all smooth length $h$ space curves $g$ which satisfy the specified boundary conditions, where the functional $F$ is defined for space curves $u, v, w$ by $F(u, v, w) := |w|^2 + \lambda(s)(|v|^2 - 1)$. That is $\int_0^h F(x, x', x'')ds = \int_0^h [|x''|^2 + \lambda(s)(|x'|^2 - 1)]ds$ is minimal. Note we have replaced the constraint $|x'| = 1$ with the equivalent constraint $|x'|^2 = 1$.

The Euler differential equations associated with the variational problem of determining the function $x : [0, h] \to \mathcal{R}^3$ for which $\int_0^h F(x, x', x'')ds$ is minimal are given as the three differential equations:

$$\partial_{u_i} F(x, x', x'') - \frac{d}{ds}\partial_{v_i} F(x, x', x'') + \frac{d^2}{ds^2}\partial_{w_i} F(x, x', x'') = 0,$$

for $i = 1, 2, 3$ [Buc56].

**Exercise 18.4:** Show that, for *any* smooth functional of the form $\int_0^h F(u, v, w)ds$, if $x : [0, h] \to \mathcal{R}^3$ is a space curve for which $\int_0^h F(x, x', x'')ds$ is minimal for $x$ ranging over a class of suitable smooth functions, then $x$ satisfies the three differential equations

$$\partial_{u_i} F(x, x', x'') - \frac{d}{ds}\partial_{v_i} F(x, x', x'') + \frac{d^2}{ds^2}\partial_{w_i} F(x, x', x'') = 0,$$

for $i = 1, 2, 3$.

**Solution 18.4:** Suppose the function $x : [0, h] \to \mathcal{R}^3$ is chosen so that $\int_0^h F(u, v, w)\, ds$ is minimal. Let $y : [0, h] \to \mathcal{R}^3$ be an arbitrary smooth space curve with $y(0) = 0, y(h) = 0, y'(0) = 0$ and $y'(h) = 0$. Form the function $z = x + \alpha y$ and consider the function

$$g(\alpha) := \int_0^h F(x + \alpha y, x' + \alpha y', x'' + \alpha y'')\, ds = \int_0^h F(z, z', z'')\, ds.$$

Since $g(0)$ is minimal by assumption, $g'(0) = 0$. Computing $g'(0)$ then yields the Euler equations. We have

$$g'(\alpha) = \int_0^h [\sum_{i=1}^{3} \partial_{u_i} F(z, z', z'')y + \partial_{v_i} F(z, z', z'')y'$$
$$+ \partial_{w_i} F(z, z', z'')y''] \, ds,$$

so,

$$g'(0) = \int_0^h [\sum_{i=1}^{3} \partial_{u_i} F(x, x', x'')y + \partial_{v_i} F(x, x', x'')y'$$
$$+ \partial_{w_i} F(x, x', x'')y''] \, ds.$$

Now recall the integration by parts rule: $\int_a^b pq' = p(b)q(b) - p(a)q(a) - \int_a^b p'q$. Applying this rule yields

$$\int_0^h \partial_{v_i} F(x, x', x'')y' \, ds = \partial_{v_i} F(x, x', x'')y \mid_{s=0}^{s=h}$$
$$- \int_0^h (\partial_{v_i} F(x, x', x''))'y \, ds.$$

But $y$ is chosen to satisfy $y(0) = y(h) = 0$, so

$$\int_0^h \partial_{v_i} F(x, x', x'')y' \, ds = - \int_0^h (\partial_{v_i} F(x, x', x''))'y \, ds.$$

Also, we may apply integration by parts to write

$$\int_0^h \partial_{w_i} F(x, x', x'')y'' ds = \partial_{w_i} F(x, x', x'')y' \mid_{s=0}^{s=h}$$
$$- \int_0^h (\partial_{w_i} F(x, x', x''))'y' \, ds.$$

The function $y$ is chosen so that $y'(0) = y'(h) = 0$. This results in $\partial_{w_i} F(x, x', x'')y' \mid_{s=0}^{s=h} = 0$. Then we may again apply integration by parts to obtain

$$- \int_0^h (\partial_{w_i} F(x, x', x''))'y' \, ds = -(\partial_{w_i} F(x, x', x''))'y \mid_{s=0}^{s=h}$$
$$+ \int_0^h (\partial_{w_i} F(x, x', x''))''y \, ds.$$

But $y(0) = y(h) = 0$, so altogether, we have

$$\int_0^h \partial_{w_i} F(x, x', x'')y''\,ds = \int_0^h (\partial_{w_i} F(x, x', x''))'' y\,ds.$$

Thus,

$$0 = g'(0) = \int_0^h y[\sum_{i=1}^{3} \partial_{u_i} F(x, x', x'') - (\partial_{v_i} F(x, x', x''))' \\ + (\partial_{w_i} F(x, x', x''))'']\,ds.$$

Since this is true for *every* admissible choice of $y$, *each* term of the integrand factor

$$\sum_{i=1}^{3}[\partial_{u_i} F(x, x', x'') - (\partial_{v_i} F(x, x', x''))' + (\partial_{w_i} F(x, x', x''))'']$$

must be identically zero! Thus we have the Euler equations

$$\partial_{u_i} F(x, x', x'') - (\partial_{v_i} F(x, x', x''))' + (\partial_{w_i} F(x, x', x''))'' = 0$$

for $i = 1, 2, 3$. Any space curve that corresponds to a minimum value of $\int_0^h F(x, x', x'')\,ds$ must satisfy these differential equations. (However, satisfying the Euler equations is only a necessary condition on $x$. There may be various solutions corresponding to distinct local minima and local maxima of $\int_0^h F(x, x', x'')ds$, or there may be other functions that satisfy the Euler equations which do not correspond to an extreme value of $\int_0^h F(x, x', x'')ds$, or it may be that the Euler equations have no solutions among the class of admissible functions).

**Exercise 18.5:** What are the Euler equations for the function $x$ where $F(u, v, w) = |w|^2$?

The Euler equations associated with minimizing $\int_0^h F(x, x', x'')ds$, where $F(x, x', x'') = |x''|^2 + \lambda(|x'|^2 - 1)$, reduce to the vectorized differential equation

$$\frac{d}{ds}[2x''' - 2\lambda x'] = 0.$$

A space curve $x$ must satisfy this differential equation in order to be a minimal energy curve, but this is not a sufficient condition. In particular, curves for which the energy is only locally minimal may arise as solutions.

**Exercise 18.6:** Show that there is no *maximal* energy length $h$ curve that connects the points $a$ and $b$.

**Exercise 18.7:** Consider all the smooth functions $f : \mathcal{R}^1 \to \mathcal{R}^1$ that connect two given points with given slopes. Show that the Euler equation for that function for which $\int_0^1 \mid f'' \mid$ is minimal is $f'''' = 0$.

Let $u$ denote the unit tangent vector $x'$. Then our vector Euler equation $[2x''' - 2\lambda x']' = 0$ can be integrated to obtain $u'' - \lambda u = e$ where $e \in \mathcal{R}^3$ is a vector of constants of integration and $\lambda$ is an unknown Lagrange multiplier function.

Since the vector cross product of a vector with itself is 0, we can eliminate the unknown function $\lambda$ by computing the cross product of the above equation with $u$ to obtain $(u'' - \lambda u - e) \times u = u'' \times u - e \times u = 0$. Note that $(u' \times u)' = u'' \times u$. Thus $u'' \times u = (u' \times u)' = e \times u$, and integrating $(u' \times u)' = e \times u$ yields $u' \times u = e \times x + f$ where $f \in \mathcal{R}^3$ is a vector of constants of integration.

**Exercise 18.8:** Show that $a \cdot (b \times c) = b \cdot (c \times a) = c \cdot (a \times b) = -a \cdot (c \times b) = -b \cdot (a \times c) = -c \cdot (b \times a)$.

Now computing the dot product of $u'$ with the equation $u'' \times u = e \times u$ yields $(u'' \times u) \cdot u' = (e \times u) \cdot u'$, and the vector triple product identities in the above exercise yield $u \cdot (u' \times u'') = e \cdot (u \times u')$. Recall that the curvature $K$ and the torsion $T$ of an arc length parameterized space curve $x$ satisfy $K^2 T = -x' \cdot (x'' \times x''')$. Thus our physical spline curve $x$ satisfies $K^2 T = -e \cdot (u \times u') = e \cdot (e \times x + f) = e \cdot (e \times x) + e \cdot f = e \cdot f$. Let $g := e \cdot f$. Then $K^2 T = g$.

Now computing the dot product of $u'$ with our earlier equation $u'' - \lambda u - e = 0$ yields the identity $u' \cdot u'' - \lambda u' \cdot u - e \cdot u' = 0$. But $u' \cdot u'' = \frac{1}{2}(u' \cdot u')'$, and, for an arc length parameterized curve, $u \cdot u' = 0$, so we have $\frac{1}{2}(u' \cdot u')' = e \cdot u'$. Also, for an arc length parameterized curve, $K = |u'|$, so $K^2 = u' \cdot u'$, and thus $e \cdot u' = \frac{1}{2}(K^2)'$. Now we may integrate this equation to obtain $K^2 = 2e \cdot u + q$, where $q$ is a constant of integration. Thus $e \cdot u = \frac{1}{2}(K^2 - q)$.

**Exercise 18.9:** Prove the triple product identity $[a \cdot (b \times c)]^2 = (a \cdot a)(b \cdot b)(c \cdot c) - (a \cdot a)(b \cdot c)(c \cdot b) - (a \cdot b)(b \cdot a)(c \cdot c) - (a \cdot c)(b \cdot b)(c \cdot a) + (a \cdot b)(b \cdot c)(c \cdot a) + (a \cdot c)(b \cdot a)(c \cdot b)$.

Recall the relation $u' \times u = e \times x + f$ obtained above. Taking the dot product of this equation with $e$ yields $e \cdot (u' \times u) = e \cdot (e \times x) + e \cdot f = g$. Thus, from the above exercise, the square of the identity $e \cdot (u' \times u) = g$ is

$$(e \cdot e)(u \cdot u)(u' \cdot u') - (e \cdot e)(u \cdot u')^2 - (u \cdot u)(e \cdot u')^2$$
$$- (u' \cdot u')(e \cdot u)^2 + 2(e \cdot u)(u \cdot u')(e \cdot u') = g^2.$$

Let $m^2 = e \cdot e$. Then, since $u \cdot u = 1$, $u \cdot u' = 0$, $u' \cdot u' = K^2$, $e \cdot u = \frac{1}{2}(K^2 - q)$, and $e \cdot u' = \frac{1}{2}(K^2)' = KK'$, we have

$$m^2K^2 - K^2K'^2 - K^2(\frac{1}{2}(K^2 - q))^2 = g^2.$$

This reduces to the following differential equation for $K^2$ as a function of $s$:

$$(K^2)' = z[K^2(4m^2 - (K^2 - q)^2) - 4g^2]^{\frac{1}{2}},$$

where $z(s)$ variously equals 1 or $-1$. We may assume that $K^2$ is a continuously differentiable function; thus the function $z$ changes value between 1 and $-1$ only at a finite number of separated values of $s$ in the interval $[0, h]$ and, at each such change point, $K^2(4m^2 - (K^2 - q)^2) - 4g^2 = 0$. Since $K^2T = g$, we know the torsion $T$ when we know $K^2$ and $g$.

**Exercise 18.10:** Show that, if $(K(s)^2)' = 0$, then either $K(s) = 0$ or $K'(s) = 0$, and if $K$ is ever 0, then the constant $g$ is 0. Then show that if $K$ is ever 0, then $T = 0$, so a twisted space curve physical spline where $T \neq 0$ also has $K > 0$ for $s \in [0, h]$.

**Exercise 18.11:** Show that there at most 3 values of $K$ for which $z$ can have a corresponding sign change. What can you say about the maximum number of times that $z$ can change sign, in both the case where $T \neq 0$ and in the case where $T = 0$? When is $K$ a periodic function?

**Exercise 18.12:** Show that, when $g = 0$, $K'' = -\frac{1}{2}K^3 + \frac{q}{2}K$.

The arc length parameterized physical spline $x$ is thus defined by the Frenet equations

$$\begin{aligned} u' &= Kn, \\ n' &= -Ku - Tb, \\ b' &= Tn, \text{ and} \\ x' &= u, \end{aligned}$$

augmented by Mehlum's equations

$$\begin{aligned}(K^2)' &= z[K^2(4m^2 - (K^2 - q)^2) - 4g^2]^{\frac{1}{2}}, \text{ and}\\ T &= g/K^2\end{aligned}$$

together with the boundary conditions $x(0) = a, x(h) = b, x'(0) = u(0) = c$, and $x'(h) = d$ where $a, b, c, d \in \mathcal{R}^3$ and $|c| = |d| = 1$. In order to compute the curve $x$ as the solution to an initial value problem, we must determine the initial normal and binormal vectors $n(0)$ and $b(0)$, and the scalars $K(0)^2$, $q$, $g$, and $m^2$. Note $n(0)$ and $b(0)$ must satisfy $|n(0)| = |b(0)| = 1, n(0) \cdot c = 0, b(0) \cdot c = 0$, and $n(0) \cdot b(0) = 0$. And $q$, $g$ and $m^2$ must satisfy $K(s)^2(4m^2 - (K(s)^2 - q)^2) - 4g^2 \geq 0$ for $0 \leq s \leq h$.

**Exercise 18.13:** Show that the energy $E(s)$ in the space curve $x$ of length $s$ that connects the points $x(0)$ and $x(s)$ satisfies the differential equation $E' = K^2$ with the initial condition $E(0) = 0$. Is it true that $E'(h) = 0$? *Hint*: $E(h)$ is minimal over all suitable choices of the space curve $x$. Also, $K \geq 0$.

The Frenet-Mehlum equations require 13 initial values and 3 parameter values subject to 6 orthogonality and unit length constraints. We are given 12 specified values as the components of the vectors $a$, $b$, $c$, and $d$, subject to 2 unit length constraints. Thus, in principle, we have a non-degenerate problem with a finite number of separated points in 16-space where the space curve $x$ is determined with locally minimal energy, and one of these points corresponds to the globally minimal energy curve. After establishing the initial conditions $x(0) = a$ and $u(0) = c$, we have ten values to be determined with five applicable constraints, and the remaining vectors $b$ and $d$ constitute six given boundary values which have the one applicable constraint $|d| = 1$.

Now a set of five independent parameters can be computed by curve fitting $x$ to the point $(h, b_1, b_2, b_3)$ and $u$ to the point $(h, d_1, d_2, d_3)$, i.e., by computing the parameter values $b_1$, $b_2$, $b_3$, $d_1$, $d_2$, and $d_3$ so that $(x_1(h) - b_1)^2 + (x_2(h) - b_2)^2 + (x_3(h) - b_3)^2 + (u_1(h) - d_1)^2 + (u_2(h) - d_2)^2 + (u_2(h) - d_2)^2 + (d_1^2 + d_2^2 + d_3^2 - 1)^2$ is minimal. This involves repetitively solving the Frenet-Mehlum differential equations, and that is complicated by the need to resolve the unknown function $z$. The function $z$ can be resolved by combinatorial trials: at $s = 0$, and at each potential switching point where $z$ could switch values between 1 and $-1$, we follow both possibilities, and we finally choose $z$ so that the energy of $x$ is minimized. This combinatorial resolution of $z$ in the context of an iterative minimization procedure makes computing the general two-point physical spline curve extremely difficult.

It may be somewhat easier to define $x$ via the Frenet equations augmented with the equations $K^2 = 2e \cdot u + q$ and $T = g/K^2$. This introduces four parameters and removes three parameters, so we now have a total of 17 values to be determined. However, even though this parameter estimation problem is potentially degenerate, we may fare better than when we tackle the non-degenerate combinatorial curve fitting problem based on Mehlum's equations directly. We can introduce another condition by forcing the energy $E(h)$ to be as small as possible; this can be done by fitting $E$ to the point $(h, 0)$.

Note that we may, in principle, allow the tangent directions $c$ and $d$ to vary so as to obtain the length $h$ overall minimal energy physical spline for all possible choices of $c$ and $d$ subject to $|c| = |d| = 1$.

**Exercise 18.14:** Does it make sense to allow the length $h$ to vary in order to obtain the *absolutely global minimum energy* physical spline connecting the points $a$ and $b$? *Hint*: Does this depend on the choice of the directions $c$ and $d$? Consider the case where the curve $x$ approaches a circle as $h \to \infty$. (Note that there are cases where there *is* a finite length which produces an locally minimal energy less than the energy for other slightly different lengths: consider $a = (0, 0, 0)$ and $b = (1, 0, 0)$ with $c = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)$ and $d = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0)$. Although the corresponding physical spline has minimal energy when $h \to \infty$, the graph of energy versus $h$ is *not* monotonically decreasing.)

**Exercise 18.15:** Suppose $x$ is the length $h$ mimimum energy arc length parameterized space curve that connects the points $a$ and $b$ with the associated unit tangent directions $c$, and $d$. Is the associated energy function $E(s)$ minimal at each value $s \in [0, h]$ with respect to all suitable choices of length $s$ space curves that connect the points $x(0)$ and $x(s)$ with the associated unit tangent vectors $x'(0) = c$ and $x'(s)$? What about the case where the tangent directions at $a$ and $b$ are freely assignable?

**Exercise 18.16:** Show that the shape of the length $h$ minimal energy physical spline space curve $x$ that connects $a$ and $b$ with the associated tangent directions $c$ and $d$ is invariant with respect to a uniform scaling transformation. That is, show that $\alpha x$ is the length $\alpha h$ minimal energy physical spline space curve that connects the point $\alpha a$ to the point $\alpha b$ with the associated tangent directions $c$ and $d$.

## 18.2 Computing a 2D Physical Spline Segment

Kallay [Kal86] and Jou [Jou95][JH90] have studied the more tractable case where our physical spline lies in the $xy$ plane. In this case, we consider the problem of computing the shape of the minimum energy physical spline $x(s)$ where $x : \mathcal{R} \to \mathcal{R}^2$ is of length $h$ and connects the two given points $a = (a_1, a_2)$ and $b = (b_1, b_2)$ with the tangent angle $c$ at $s = 0$ and the tangent angle $d$ at $s = h$. Note here $c$ and $d$ are scalars.

Now, to compute the real valued functions $x_1$ and $x_2$ that define the minimal energy length $h$ planar physical spline $x$, we may proceed as follows. Let $\theta(s)$ denote the direction angle of the tangent vector of the arc length parameterized planar curve $x$ at $s$. Then the signed curvature of $x$ is $\theta'$, and the curvature of $x$, $K$, is $|\theta'|$, so the corresponding energy is $E = \int_0^h (\theta'(s))^2 ds$.

Note $x_1'(s) = \cos(\theta(\mathrm{s}))$ and $x_2'(s) = \sin(\theta(\mathrm{s}))$. Thus

$$\begin{aligned}\int_0^h x_1'(s)ds &= x_1(h) - x_1(0) = \int_0^h \cos(\theta(\mathrm{s}))\mathrm{ds}, \text{ and}\\ \int_0^h x_2'(s)ds &= x_2(h) - x_2(0) = \int_0^h \sin(\theta(\mathrm{s}))\mathrm{ds}.\end{aligned}$$

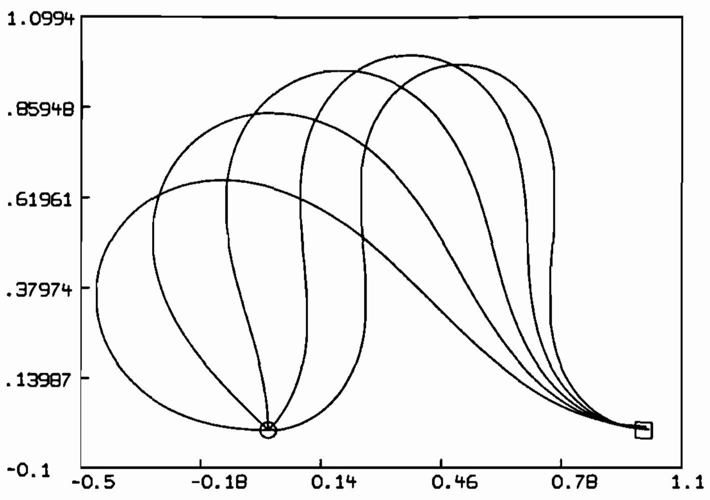
We wish to find the function $\theta$ that minimizes $E$ subject to the directional constraints: $\theta(0) = c$ and $\theta(h) = d$, together with the constraints: $\int_0^h \cos(\theta(\mathrm{s}))\mathrm{ds} = \mathrm{b}_1 - \mathrm{a}_1$ and $\int_0^h \sin(\theta(\mathrm{s}))\mathrm{ds} = \mathrm{b}_2 - \mathrm{a}_2$. Given the function $\theta$, we can compute the curve $x$ as the solution to the differential equations $x_1'(s) = \cos(\theta(\mathrm{s}))$ and $x_2'(s) = \sin(\theta(\mathrm{s}))$, with $x_1(0) = a_1$ and $x_2(0) = a_2$.

Kallay [Kal86] and Jou [Jou95] have computed the Euler equation that defines $\theta$ to be

$$\theta'' + m_1\sin(\theta) - \mathrm{m}_2\cos(\theta) = 0,$$

with the boundary conditions $\theta(0) = c$ and $\theta(h) = d$, where $m_1$ and $m_2$ are unknown Lagrange multipliers to be determined.

This differential equation arises in a specialized form in describing the shape of a bent beam in structural engineering. There we have a beam of length $h$ with one end fixed at the origin with slope 0, and with a load or force $f$ applied at the other end of the beam. Then the angle of the tangent along the beam, $\theta$, satisfies the differential equation $\theta'' = -\frac{f_2}{\epsilon I}\cos(\theta) - \frac{\mathrm{f}_1}{\epsilon \mathrm{I}}\sin(\theta)$ with $\theta(0) = \theta'(0) = 0$. The value $\epsilon$ is the modulus of elasticity of the beam material and $I$ is the moment of inertia of the cross section of the beam about its horizontal axis. Solutions of this differential equation have been obtained in terms of incomplete elliptic integrals [Jou95].



**Exercise 18.17:** Show that $\theta' = m_2 x_1 - m_1 x_2 + m_0$ where $m_0$, $m_1$, and $m_2$ are constants.

We are given the vectors $a$ and $b$, and the scalars $c$ and $d$, and we want to compute the values $\theta(0)$, $\theta'(0)$, $x_1(0)$, $x_2(0)$, $m_1$, and $m_2$ which yield the desired curve $x$. Of course, $x_1(0) = a_1$, $x_2(0) = a_2$ and $\theta(0) = c$. Then $\theta'(0)$, $m_1$ and $m_2$ can be determined by curve fitting $x_1$ to $(h, b_1)$ and $x_2$ to $(h, b_2)$ and $\theta$ to $(h, d)$; this is equivalent to solving the equations $x_1(h) = b_1$, $x_2(h) = b_2$, and $\theta(h) = d$ for the values of $\theta'(0)$, $m_1$ and $m_2$. Jou [Jou95] gives a survey of planar physical splines and groups them corresponding to the values of $m_1$ and $m_2$; this provides a strategy for obtaining initial guesses for $m_1$ and $m_2$. The physical spline curves shown in [Jou95] are often unexpected and always interesting. The graph above shows the five planar physical spline curves of length 2.5 that connect the point $(0, 0)$ to the point $(1, 0)$ with the entry angle 0 where the exit angle from $(0, 0)$ ranges through $\{\pi, \frac{3\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4}, 0\}$.

**Exercise 18.18:** What planar curve results when the angles $c$ and/or $d$ are chosen outside the range $(-\pi, \pi]$?

**Exercise 18.19:** Describe how to compute the minimal energy hermite cubic polynomial spline segment space curve $x$ of length $h$ with the parameter limit value $t_1$ that satisfies $x(0) = a$, $x(t_1) = b$, $x'(0)/|x'(0)| = c$ and $x'(t_1)/|x'(t_1)| = d$, where $a$, $b$, $c$ and $d$ are given 3-component vectors with $|c| = |d| = 1$.

**Solution 18.19:** Define $x(t) = y(t/t_1)$ where $y(r) = (1 - 3r^2 + 2r^3)a + (3r^2 - 2r^3)b + (r - 2r^2 + r^3)t_1\alpha c + (r^3 - r^2)t_1\beta d$, and define the curvature $K(t) = [|x'(t)|^2|x''(t)|^2 - (x'(t) \cdot x''(t))^2]^{\frac{1}{2}}/|x'(t)|^3$. Now compute $\alpha$ and $\beta$ to minimize the energy $\int_0^{t_1} K(t)^2\,dt$ subject to the constraint $(\int_0^{t_1} |x'(t)|\,dt - h)^2 = 0$.

# References

[Aki70] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM*, 17(4):589–602, October 1970.

[B́74] Pierre E. Bézier. Mathematical and practical possibilities of unisurf. In *Computer Aided Geometric Design (editors: Robert E. Barnhill and Richard F. Riesenfeld)*, pages 127–152. Academic Press, NY, 1974.

[BB83] Brian A. Barsky and John C. Beatty. Local control of bias and tension in beta-splines. *ACM TOGS*, 2(2):193–218, April 1983.

[BBB87] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, Los Altos, CA, 1987.

[Buc56] R. Creighton Buck. *Advanced Calculus*. McGraw-Hill, N.Y., 1956.

[Chu88] Charles K. Chui. *Multivariate Splines*. Soc. for Indust. and Applied Math., Philadelphia, 1988.

[Coo67] Stephen A. Coons. Surfaces for computer aided design of space forms. Technical Report Project Mac-TR-41, Mass. Inst. of Technology, 1967.

[CP92] Jin J. Chou and Leslie A. Piegl. Data reduction using cubic rational b-splines. *IEEE Computer Graphics and Appl.*, 12(3):60–68, May 1992.

[DD87] M. Davis and J. Dowden. Interpolation by a local taut cubic polynomial. *Computing*, 38(1):299–313, 1987.

[DeB78] Carl DeBoor. *A Practical Guide to Splines.* Springer-Verlag, NY, 1978.

[DeB87] Carl DeBoor. B-form basics. In *Geometric Modeling (editor: G. Farin)*, pages 131–148. Soc. for Indust. and Applied Math., Philadelphia, 1987.

[DM72] H. Dym and H. P. McKean. *Fourier Series and Integrals.* Academic Press, NY, 1972.

[DoC76] Manfredo P. DoCarmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[Far90] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide (2nd ed.).* Academic Press, NY, 1990.

[FB84] F. N. Fritsch and J. Butland. A method for constructing local monotone piecewise cubic interpolants. *SIAM J. Sci. Stat. Comp.*, 5(2):300–304, June 1984.

[FC80] F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal.*, 17(2):238–246, April 1980.

[Fer64] J. C. Ferguson. Multivariate curve interpolation. *J. ACM*, 11(2):221–228, April 1964.

[FM86] G. Yates Fletcher and David F. McAllister. Natural bias approach to shape preseving curves. *C.A.D.*, 18(1):48–52, Jan./Feb. 1986.

[FM87] G. Yates Fletcher and David F. McAllister. Methods for convexity-preserving interpolation. *Computer Graphics and Appl.*, 7(8):7–14, Aug. 1987.

[Fol87] Thomas A. Foley. Interpolation with interval and point tension controls using cubic weighted v-splines. *ACM Trans. on Mathematical Software*, 13(1):68–96, 1987.

[For72] A. Robin Forrest. On coons and other methods for the representation of curved surfaces. *Computer Graphics and Image Processing*, 1:341–359, 1972.

[Gor69] William J. Gordon. Spline-blended surface interpolation through curve networks. *J. Math. Mechanics*, 18(10):931–952, April 1969.

[Har91] Wolfgang Hardle. *Applied Nonparametric Regression*. Cambridge University Press, NY, 1991.

[Hym83] James M. Hyman. Accurate monotonicity preserving cubic interpolation. *SIAM J. Sci. Stat. Computing*, 4(4):645–654, 1983.

[JH90] Emery D. Jou and Weimin Han. Minimal energy splines: I. plane curves with angle constraints. *Mathematical Methods in the Applied Sciences*, 13:351–372, 1990.

[Jou95] Emery D. Jou. Minimal energy splines. Technical Report Ph.D. Thesis, University of Maryland, 1995.

[Kal86] Michael Kallay. Plane curves of minimal energy. *ACM Transactions on Mathematical Software*, 12(3):219–222, 1986.

[KB84] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics (SIGGRAPH 84)*, 18(3):33–41, July 1984.

[Lee89] Eugene T. Y. Lee. Choosing nodes in parametric curve interpolation. *Computer Aided Design*, 21(6), 1989.

[Lv86] Peter Lancaster and Kęstutis Šalkauskas. *Curve and Surface Fitting: An Introduction*. Academic Press, London, 1986.

[Mal77] Michael A. Malcolm. On the computation of nonlinear spline functions. *SIAM J. of Numerical Analysis*, 14:254–282, 1977.

[Meh69] Even Mehlum. Curve and surface fitting based on variational criteriae for smoothness. Technical report, Central Institute for Industrial Research, Oslo, Norway, Dec. 1969.

[Meh74] Even Mehlum. Nonlinear splines. In *Computer Aided Geometric Design (editors: Robert E. Barnhill and Richard F. Riesenfeld)*, pages 173–207. Academic Press, NY, 1974.

[Nie74] Gergory M. Nielson. Some piecewise polynomial alternatives to splines in tension. In *Computer Aided Geometric Design (editors: Robert E. Barnhill and Richard F. Riesenfeld)*, pages 209–235. Academic Press, NY, 1974.

[Nie86] Gergory M. Nielson. Rectangular $\nu$-splines. *IEEE Computer Graphics*, 6(2):35–40, Feb. 1986.

[Ove68] A. W. Overhauser. Analytic definition of curves and surfaces by parabolic blending. Technical Report SL 68-40, Ford Motor Co., 1968.

[RFS63] Robert B. Leighton Richard P. Feynman and Matthew Sands. *Lectures on Physics, Vol. 1–3*. Addison-Wesley, Reading MA, 1963.

[Sch46] Issac J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Appl. Math.*, 4(1):45–99, 112–141, 1946.

[Sch64a] Issac J. Schoenberg. On interpolation by spline functions and its minimum properties. *Int. Ser. of Numerical Analysis*, 5:109–129, 1964.

[Sch64b] Issac J. Schoenberg. Spline functions and the problem of graduation. *Proc. Nat. Acad. of Science*, 52:947–950, 1964.

[Sch67] Issac J. Schoenberg. On spline functions. In *Inequalities (editor: O. Shisha)*, pages 255–291. Academic Press, NY, 1967.

[Sch83] Larry L. Schumaker. On shape preserving quadratic spline interpolation. *SIAM J. Numer. Anal.*, 20(3):854–863, Aug. 1983.

[Spi75] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish Press, Boston, 1975.

[Wah90] Grace Wahba. *Spline Models for Observational Data*. SIAM CBMS-NSF Conf. Series in Applied Math., No. 59, 1990.

[War65] K. L. Wardle. *Differential Geometry*. Dover Pub., NY, 1965.

[Yam88] F. Yamaguchi. *Curves and Surfaces in Computer-Aided Geometric Design*. Springer-Verlag, NY, 1988.

# Index