AN ABSTRACT OF THE THESIS OF

Peter Martin for the degree of Master of Science in Geography

presented on August 26, 2004.

Title: Spatial Interpolation in Other Dimensions

Abstract approved:

Dawn J. Wright

The purpose of this work is to expand the theoretical foundations of interpolation of spatial data, by showing how ideas and methods from information theory and signal processing are applicable to the the work of geographers.

Attention is drawn to the distinction between what we study and how we represent it as a sum of components; hence mathematical transforms are introduced as re-arrangements of information that result in alternative representations of the signals of interest. A spatial model is developed for understanding transforms as the means to obtain different views of function space, and the question of interpolation is recast in geometric terms, as a matter of placing an approximation within the bounds of likelihood according to context, using data to eliminate possibilities and estimating coefficients in an appropriate base.

With an emphasis on terrain elevation- and bathymetry signals, applications of the theory are illustrated in the second part, with particular attention to $1/f$ spectral characteristics and scale-wise self-similarity as precepts for algorithms to generate "expected detail". Methods from other fields as well as original methods are tested for scientific visualization and cartographic application to geographic data. These include fractal image super-resolution by pyramid decomposition, wavelet-based interpolation schemes, principal component analysis, and Fourier-base schemes, both iterative and non-iterative. Computation methods are developed explicitly, with discussion of computation time.

Finally, the quest to simulate "detailed" data is justified by challenging the traditional measure of interpolation accuracy in the standard base, proposing instead an alternative measure in a space whose transform reflects the relative importance of the components to communication of information.

Spatial Interpolation in Other Dimensions

by
Peter Martin

A THESIS

Submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented August 26, 2004
Commencement June 2005

Master of Science thesis of Peter Martin
presented on August 26, 2004.

APPROVED:

Major Professor, Representing Geography

Chair of the Department of Geography

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University Libraries. My signature below authorizes release of my thesis to any reader upon request.

Peter Martin, Author

ACKNOWLEDGMENTS

I am grateful to my major professor Dawn Wright, minor professor Chris Goldfinger, and to committee members Jon Kimerling and Court Smith for their patience and optimism through this delayed effort. I am especially grateful to Dawn Wright for her generous personal investment in my academic success, and for the freedom and encouragement that she has always given me. Likewise I thank Dr. K. for his good humor, encouragement, and shared interest in both the art and science of geography.

Thanks also to Bill Graham for providing GEODAS and GTOPO30 data, and for suggesting the works of Wolfram and of Ebert, *et. al.* Thanks to Julia Jones, whose class in Spatial Analysis (along with Jon Kimerling's class in Digital Image Processing) was a seedbed for many ideas that I've developed here.

I gratefully acknowledge the contributions of people everywhere who share information freely, including Tobias Werther (student of H. G. Feichtinger), who kindly provided me the IRSATOL toolbox files, and whose thesis served as both a model and reference for me; Eero Simoncelli, who has posted his PyrTools toolbox including helpful instruction; INRIA, who have made public FracLab; F. Auger, P. Flandrin, P. Gonçalvès, and O. Lemoine, authors of the Time-Frequency Toolbox and its excellent tutorial, also made public by INRIA; D. Donoho, M. Duncan, X. Huo, and O. Levi-Tsabari, responsible for the WaveLab toolbox released by Stanford University; C. Merkwirth, U. Parlitz, I. Wedekind, and W. Lauterborn, who assembled the Open TSToolbox released by the Third Polytechnic Institute of Göttingen.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

All information comes in context. I perceive the larger context in which this communication is written to be a time when people seem to be afraid of seeing things differently. Therefore the broader undeclared purpose of the work is to counter aversion to the unfamiliar and pride in provincial perspective, by encouraging circumspect consideration of diverse points of view.

1 Introduction

Our experience of the world is one of sampling and modeling. In ordinary perception, we fill in the gaps and "complete" the patterns, based on familiarity or expectation, in order to behold a coherent whole (see, *e.g.*, [47]). Striking well-known examples are the optical illusions where a certain background suggests the existence of a missing object, as shown in figure 1.1:

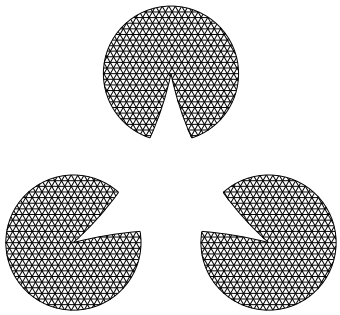


Figure 1.1: Background suggests triangle figure.

(a figure is imagined, given the ground). An even more striking converse example occurs when an abrupt sound heard while sleeping evokes an apparently antecedent dream context (the ground is imagined, given the figure). Since causality —*i.e.* the expected temporal order of things— seems to have been violated in the latter example, this example also suggests that information was being handled in a different "space" than the usual temporal domain, which is a theme that will be taken up in this study by the use of mathematical transforms.

The philosophy of this study is that interpolation deals fundamentally with information, the context in which it appears, and how it is perceived. Therefore my intent is to show how ideas from information theory, signal processing, and the study of natural vision systems can be applied theoretically to interpolation of spatial data

in general, with some simple practical examples of applications to interpolation of topography and bathymetry in particular.

2 Theory

The first part of this work is a discussion of a theoretical basis for approaching questions of interpolation of spatial data. The second part will illustrate applications of the theory in practical computation algorithms.

2.1 Foundations

For purposes of cartography or scientific visualization, one might wish to present a complete picture, supplying data that are most likely, or least misleading or distracting, where they are not known explicitly. For example, consider the synthetic seafloor profile of figure 2.1, comprising 256 points generated by my `MATLAB` function `FINV` (see appendix for all `MATLAB` code) to simulate a so-called $1/f$ spectral characteristic (vertical scale exaggerated for clarity), and the adjacent figure depicting a sampling of every sixteenth point.

Given the sampling of sixteen points, it may be a visual aid to "connect the dots"; a linear interpolation is shown in the lower left of figure 2.1, and a "smooth" interpolation is shown in the lower right of the same figure. Clearly the interpolations suggest different surface characteristics, and it is difficult to say what one would assume without the visual aid of interpolation.

Briggs [7], who described an "exact" (preserving the sample data points) minimum-curvature method of interpolating scattered data, stated that an interpolation method must not introduce information that is not in the data. But in a sense this is impossible; one just chooses the most artful, and we hope scientific, way to fabricate. Every interpolation method is a sort of hypothesis about the surface, which may or may not be true [41].

Interpolation often is not properly performed, for reasons of lack of expertise, lack of knowledge of data quality, or lack of options in software [8]. But I think the

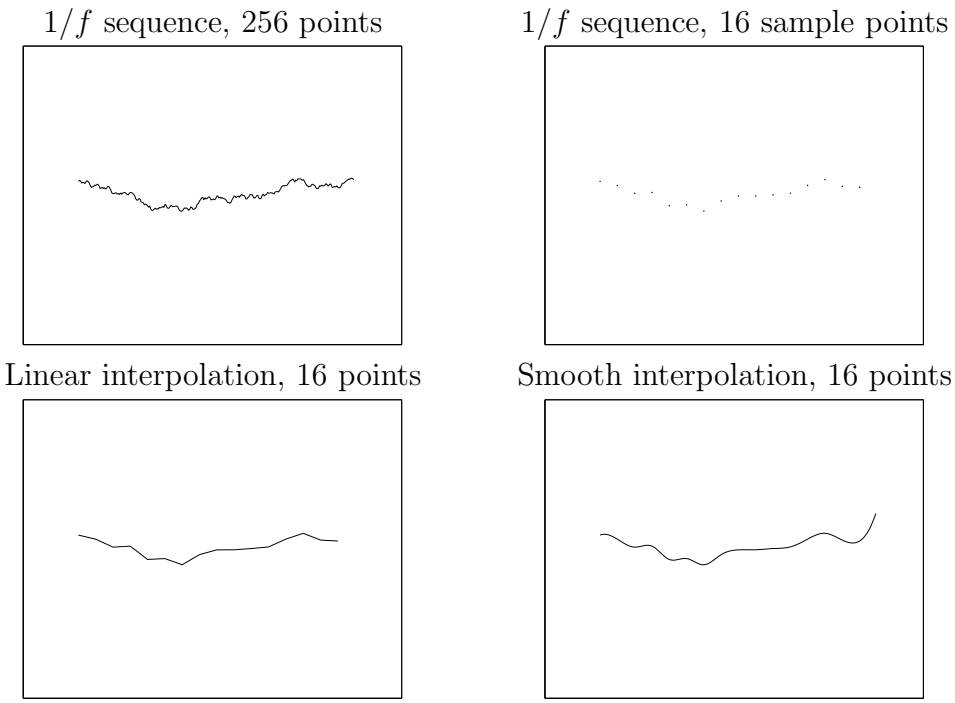


Figure 2.1: Synthetic $1/f$ sequence of 256 points, 16 sample points, and two interpolations.

problem of naive interpolation is even deeper: a great deal more attention needs to be given to the *context* of the signal to be interpolated. (In fact I offer the conjectural view that information consists solely of a nesting of contexts. This will be discussed further in Section 2.6.)

In a study of "the elusive nature of expertise in spatial interpolation", Mulugeta [50] observed that experts and non-experts recognized the same deficiencies in computer-generated interpolative maps. But experts were able to produce no more accurate maps of the data than non-experts, manually. The author suggests that more site-specific expertise might be necessary for such a difference to emerge, which is consistent with the idea that the consideration of context is of the essence in the art of interpolation.

The fact that experts and non-experts recognized the same deficiencies in the computer-generated interpolation is indicative of a second problem of naive interpolation, namely its artificial appearance. Naive or statistically conservative interpolation actually can be misleading, when presented for "scientific visualization". This calls for a reconsideration of the criteria of accuracy in interpolation. This is not merely an issue of taste or of cartographic convention. Those who use maps and other graphical representations probably are more interested in spatial relations than in point values. They might be interested in the extent of surface types, the spatial associations of features, or other "whole picture" characteristics. The function of a map sometimes is to present information for exploration, rather than to communicate what the cartographer already knows [43]. On the other hand, the cartographer may wish to communicate something that he knows about the surface, without point-wise accuracy.[1]

Since the deficiencies in the computer-generated maps were readily recognized by experts and non-experts alike, they must have been of a regional or global nature, and they must have had to do with fundamentals of intelligent perception —for example, the texture of variation may have appeared wrong, or certain scale-wise correlations may have been lacking. Yet the average error of test points might have proven to be less than for another more "realistic" interpolation which preserved certain recognizable regional or global characteristics.

The technique this study employs to quantify regional or global characteristics and to apply them in the process of interpolation is that of mathematical transforms. The familiar integral transforms, including Fourier, cosine, wavelet, and various adaptive principal-component transforms, turn local variables into global or regional variables and *vice versa*. Views in these transform spaces may actually correspond more closely

[1]Discussing hand-drawn maps of ocean-basin bathymetry, Smith & Sandwell [67] point out that bathymetrists who had seen classified data could "convey some of the essence of those data without revealing secret details".

to natural modes of perception: research in physiology suggests that the mammalian vision system employs integral transforms in the processing of raw image data received by the retina [22], [28]. In fact, the wavelet-like transforms of our own vision systems apparently emphasize smaller scales compared to larger scales (Hubbard [33], interpreting Field [22]), so these would not be what are called "orthogonal transforms", which preserve Euclidean distances (related to accuracy of an approximation). Rather, relative distances in the transform space might not be the same as in the original space. The consequence is that there may be a legitimate new basis for assessing accuracy in interpolation.

2.2 Review of Interpolation

Interpolation problems may be posed in the broad categories of *exact* (to include the known data points) *vs. curve-fitting*, (to minimize the rms residual of a simple function approximating the data points), and for the cases of *regularly-spaced vs. scattered* data. Fitting arbitrary data locations to points on a regular grid (*registration* may be a preliminary task of interpolation before further "filling in". Interpolation of data that is already regularly spaced on a grid is sometimes called "super-resolution", as in an attempt to "blow up" an image without rendering it blurry or grainy.

It is fundamentally obvious that images, and all sets of sampled data, have limits of resolution, beyond which one cannot hope to find unexpected details by any data processing magic. However, it may be that the data set belongs to a class of data that have *expected* detail beyond the limit of sampled resolution, detail which may be related to the data structure at resolvable scale. This is evidently the case, for example, with phytoplankton distribution in the ocean [42], and so bears on the question of sub-pixel variability of remotely-sensed ocean color data (which is a proxy for phytoplankton concentration). This sort of cross-scale correlation of data structure is the essence of fractal sets, and "fractal-based" methods of image interpolation

have been devised and patented by Pentland, Simoncelli, and others. One method of finding and applying the cross-scale correlation depends on the *pyramidal decomposition* of an image into a scale-wise cascade of approximations and differences between scales (which is related to wavelet decomposition) [57]. The scale-wise correlation is analyzed by blocks, after Jacquin [34], who applied the methods to image coding for data compression. The efficacy of fractal methods in image processing need not be confined to image processing; it is a reflection of the fractal geometry of natural objects [56]. In this study, variations on the super-resolution method of Pentland, *et al.* will be applied to the interpolation of digital elevation model (DEM) grids.

Kriging is a favored method of interpolating scattered data. It is based on estimation of how values in a field vary as a function of their spatial separation [37] [36] [13]. It typically places high in comparison tests of the statistical accuracy of interpolation techniques (*e.g.* [61], [16]). However, it is well to consider its origin as "A statistical approach to some basic mine valuation problems..." [40]; it was not conceived as a cartographic technique, but as a method of minimizing error in estimation of unknown values, given a set of data.

Whereas Kriging depends on estimation of the *variogram*, which is the representation of the variance of the data values as a function of *lag*, or spatial separation, Pesquet-Popescu and others ([58], [59]) have explored interpolation based on the *structure function*, which expresses the variance of the *increments* of the data, as a function of lag.[2] This is presented as a method appropriate to the interpolation of fractal surfaces [59], since it is adaptable to non-stationary fields with stationary increments.[3] So it might be considered to be based more on "texture" rather than on first-order variation or periodicity in the data. But this basis is not necessarily reflected in the resulting interpolation, which is still calculated as a weighted sum of

[2]These functions, of variance of data value or of variance of data increment, may be functions of a *vector* lag —*i.e.* dependent on direction as well as separation, if the data field is not isotropic.

[3]A field of data is called *stationary* with respect to a statistical parameter if the parameter can be considered in some sense to be constant over the field, not itself a function of location.

neighboring values.[58]

Interpolation techniques traditionally are supposed to be *smooth*, whether they are the *exact* methods that include all the control points, or *curve-fitting* methods that minimize deviation from control points using low-order polynomials. Presumably the two motivations for smooth interpolation are to avoid creation of apparently anomalous points (like sharp corners or false maxima and minima), and to reflect the inherently smooth nature of the phenomenon.

It is justifiable to use smooth interpolation techniques for maps of gravity and magnetics, whose fields are inherently smooth. Thus the minimum-curvature method of Briggs [7] amounts to forcing a sheet, that resists bending, through all the control points. The *equivalent source* technique of interpolation described by Dampney [14] and generalized by Cordell [12] models a field of gravity values by means of hypothetical point masses whose combined effect would produce the measured values at the sample locations; then calculation of the resultant field at intermediate points allows generation of a complete grid of interpolated values.

But other geophysical "fields", like seafloor topography, exhibit breaks and other irregularities that do not fit the minimum-curvature model. When you use a minimum-curvature model, you get the secondary inflection effect of a long truck on a short turn: an abrupt turn in the data is preceded, in the interpolation, by a deviation in the opposite direction, to minimize the curvature of the turn. This is illustrated in figure 2.2, which shows a (vertically exaggerated) bathymetry profile of ship soundings across a shelf break southwest of Ensenada, Mexico.

The false rise before the continental shelf break, which exemplifies the weakness of the minimum-curvature method upon which Smith & Wessel [68] sought to improve, is apparent in the lower left figure, both at the top and at the bottom of the break. "Splines in tension" were proposed by Smith & Wessel as an improvement over the flexible-sheet model, introducing a variable amount of tension in the sheet that was

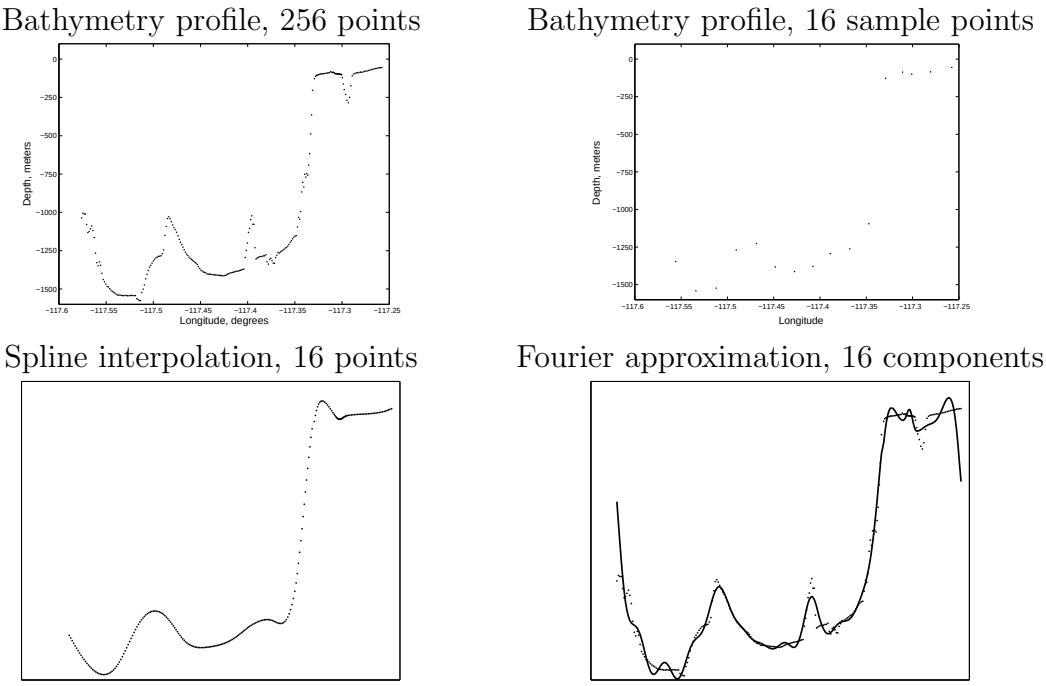


Figure 2.2: Bathymetry profile over shelf break,undesirable interpolation effects.

forced through the control points, so the sheet would be more flexible to changes of direction. (In the limit, too much tension simply results in linear interpolation.)

The improper adherence of minimum-curvature interpolation to sharp turns in data is related to the so called "Gibbs phenomenon" that is an artifact of representing sharp things with families of smooth functions [23], as in band-limited Fourier synthesis. This effect is shown for comparison in the lower right of figure 2.2, where the data have been approximated by a Fourier synthesis of the first 16 (of 128) frequency components.[4] In fact, the spectral characteristic of seafloor topography has been found to be approximately $1/f$ over seven orders of magnitude [6], and in general a fractal model of terrain seems to be more fitting than a smooth model [9] [45]

[4]The reason that the Fourier approximation of the data departs so much at the beginning and end of the sequence (as if it were trying to make ends meet) is that in the Fourier view signals either are infinite in extent or they are cyclic.

[80] [32] [44].

There may be a gray area between interpolation and simulation, and in the case of terrain modeling, as suggested by Clarke [11], smooth interpolation techniques at the "structure" scale might well be blended with fractal-based simulation techniques at the "texture" scale.[5]

2.3 Sampling, Bandwidth, and Reconstruction

The Fourier reconstruction shown in the lower right of Figure 2.2 suggests the possibility of interpolation carried out in other than the spatial domain (See, *e.g.*, [29]). Ordinarily, however, one does not sample components in the frequency domain of a signal; one samples point values in the spatial (or temporal) domain.[6] Therefore when interpolation is carried out in a transform domain, it is usually a procedure of finding transform coefficients interatively by least-squares fitting to data in the original domain.

The *Sampling Theorem*, often attributed to Shannon [65], [64] because of his clear exposition of its meaning in the context of questions of communication efficiency, relates sampling to *bandwidth*, *i.e.* the bounds of a signal's frequency components. Specifically, if a signal is "smooth", its spectrum of frequency components has an upper bound, and it can be synthesized perfectly if there are at least two samples per smallest wavelength.[7] The Sampling Theorem has been generalized by Papoulis [54], Unser [70], [71] and others (see [69] and [35] for overviews, [46] for basic text); moreover, Feichtinger and others have developed fast iterative algorithms for reconstruction of band-limited signals from irregularly-distributed samples [49], [21], [20],

[5]Note that "structure" here refers to overall shape, and does not correspond to the use of the word in the term "structure function" mentioned earlier.

[6]Often, as in remote sensing, the samples are not strictly point samples, but rather averages integrated over a pixel or other discrete quantum of spatial or temporal sampling.

[7]This sampling rate is called the *Nyquist* rate. For example, given that the upper bound of human hearing is supposed to be about 20,000 Hz, the sampling rate for digital audio is 44,100 samples per second.

[74]. In the algorithms to follow, we will explore the possibility of preliminary interpolation of the *structure* of terrain by adapting the methods of Feichtinger to the expected magnitude spectrum (*e.g.* $1/f$) of a certain class of signals, to be followed optionally by simulation of the *texture* based on other considerations.

Ford and Etter proposed a method of reconstruction of nonuniformly-sampled data based on the wavelet transform [24]. Because of the dual representation of the wavelet transform (location and scale), they claim that their method has the advantage of being able to interpolate on a coarse scale, where data are sparse, and more finely wherever samples are more dense. Their method also (like that of Feichtinger, *et al.*) is an iterative method that seeks a limited set of wavelet coefficients that yield the "best fit" to the observed data. But they apparently consider only the data set at hand, rather than trying to characterize the wavelet-coefficient characteristics of a class of data.

2.3.1 Generalization of "bandwidth"

This study implicitly will investigate whether there might be a practical generalization of the concept of band limitation as it appears in the Sampling Theorem. Band limitation can be thought of as the bounding of certain transform coefficients — namely those of the Fourier transform. A signal is said to be band-limited if its Fourier coefficients are zero outside a certain interval of frequencies. If the coefficients of *any* transform are found to be bounded in some way for a class of signals, this may have implications both for sampling and for reconstruction (interpolation) of unknown signals belonging to that class.

This is so because bounding the coefficients eliminates some possibilities. Suppose someone challenged you to guess a four-digit number, and you would get points for being close. To begin with, it would help immensely to know whether the number had four digits in base 10, or in, say, base 2, because in the first case there would be

10,000 possible numbers, while in the second case there would be only 16. If you were then allowed to sample the digit of your choice, at equal cost, before guessing, you would sample the leftmost digit (eliminating 9000 possible numbers) because that would guarantee that your guess could be within 500, whereas sampling the second-from-left digit, although also eliminating 9000 possible numbers, could still allow you to miss by 4550. Interpolation is essentially guessing coefficients, and sampling is eliminating possibilities.

The manner in which coefficients might be bounded does not need to be absolute, in the sense that certain specific coefficients need be zero. (For coefficients that have any aspect of locality, as do wavelet coefficients, that would be as absurd as the proposition that perhaps the southeast quadrant of all DEM's tends to be low-lying and flat...) Rather the coefficients may simply exhibit *correlation* —for example, Shapiro developed embedded zero-tree wavelet (EZW) coding for wavelet-based compression (the basis of the Joint Photographic Experts Group JPEG 2000 standard) based on the observation that, where coarse-scale wavelet coefficients are near zero, the finer-scale wavelet coefficients tend also to be near zero [72]. That corresponds to the much more plausible proposition that areas of low relief tend also to be relatively smooth, which in fact is an observation made by Musgrave, discussing landscape simulation in [18]. This distinction between *certain* coefficients tending to be near zero , *vs. most* coefficients tending to be near zero, is discussed further in Section 2.7.

2.4 Simulation *vis à vis* scientific interpolation

Simulation of terrain and other natural surfaces for cinema using computer algorithms is a technique of at least two decades' development (see, *eg.*, [18]). The methods that have evolved may seem *ad hoc* rather than scientific, but they are practical from an artistic and engineering point of view, and they provide insight into how classes of data might be characterized quantitatively. Notably, the methods (described in [18]) are

procedural rather than explicitly descriptive, which relates them more to algorithmic information theory (discussed by Chaitin, [10]) and to the fractal-generating Iterated Function Systems of Barnsley [4], [5], than to traditional information theory and analytic methods.

Fractal-based interpolation, or generation of fuzzy surfaces of interpolation, can also be viewed as a way of depicting *uncertainty itself*, rather than as a way of depicting a specific realization of a probability distribution [76]. In the present study however, the preferred objective of interpolation (for cartographic or scientific visualization purposes) will be to present a definite "educated guess", perhaps to be accompanied by separate metadata such as data quality maps.

Assessments have been made of the comparative accuracy of traditional methods of smooth interpolation versus the more procedural methods of simulation —for example, bilinear interpolation vs. Brownian (fractal) interpolation [60]. Typically, smooth interpolation methods prove superior by such statistical criteria as rms error and bias, but we may have lost separation of the executive and the judicial here, so to speak: the conservative methods of smooth interpolation are shown to be *consistent* with the statistical criteria of evaluation, but they are not *proven* by those criteria to communicate more effectively the information in the data or the meaning of it. As the cited study also points out, the Brownian technique better preserved texture and hydrographic patterns, as assessed by eye.

Since the purpose of this study is to encourage new views of the old problem of spatial data interpolation, it is fitting to observe that, concomitant to the development of computers, there has arisen "a new kind of science" [78], wherein understanding of the great fabric of order and disorder is approached via *algorithmic* rather than *analytic* methods —by simulation and modeling, studying how simple environments and rules of interaction can produce unexpectedly rich patterns. The work of Kauffman [38] accounting for the evolution of complex chemical and biological systems, and of

Bak [3], describing the emergent characteristics of natural systems operating near the critical point between order and disorder, are reminiscent of Barnsley's IFS fractal-generation as well as of the procedural methods of cinema mentioned above. Thus there is a growing scientific foundation for techniques of interpolation that previously might have seemed at best artistic, or at worst *ad hoc* or artificial, when regarded as lacking the traditional analytic foundation and not supported by the criteria of the traditional foundation. Now alternative techniques of interpolation may be developed and be justifiable not only because they produce results that look right, but because there is a better understanding of *why* they look right.

2.5 Linearity

The adjective "linear" is widely used but potentially ambiguous. Two quite distinct (albeit related) senses in which it might be used in connection with discussion of interpolation are the graphical sense and the system sense.

The graphical sense of linearity is exemplified by the idea of connecting points with straight lines, as in the lower left of Figure 2.1, or finding a straight line to "fit" a distribution of points, as in linear regression. In this sense, linear interpolation would refer to assigning intermediate values between data points, by inverse-distance weighting. This technique may be computationally simple, but scarcely produces results that are realistic (see, *e.g.*, [41], [79], [16], [17]). This is not surprising, since the processes or systems that give rise to the data would not be expected to output values of the form of line segments or plane surfaces. But this does not mean that the underlying processes or systems must be nonlinear.

The system sense of linearity is that a linear combination of inputs to a system will produce an output that is the same linear combination of the separate outputs that would result from the separate inputs. Formally (from [52]),

$$H\{\alpha\mathbf{x}(t) + \beta\mathbf{y}(t)\} = \alpha H\{\mathbf{x}(t)\} + \beta H\{\mathbf{y}(t)\},$$

where H is the system operator, α and β are constants, $\mathbf{x}$ and $\mathbf{y}$ are input functions (represented as vectors), and t is the independent variable of the generalized space in which signals are viewed (*e.g.* time).

System linearity is often expressed verbally in a commutative form, like"the approximation of the sum is equal to the sum of the approximations" (also called *additivity*), or "the transform of a times the signal is equal to a times the transform of the signal (also called *homogeneity*).

If a signal of interest is viewed as an effect of multiple causes, then in its system analysis the additivity and homogeneity of linearity imply a sort of independence and proportionality in the effects of separate causes, while a consequence of nonlinearity of systems is that changes in outputs may not be proportional to changes in inputs, and effects are not separable.

The focus of this study is not to understand spatial signals deterministically, but only to pursue knowledge of them in the predictive (or interpolative) sense.

2.6 Signals and Information

We will speak of spatial data sets as "signals", and also as "functions". The sort of signal representations that we will take to begin with will include distributions of values in one dimensional- or two dimensional space, such as bathymetry profiles or digital elevation models. Since a function associates a numerical (or more generally a vector) output, say z, to each input x (for a function of one variable), or a numerical output z to each pair of inputs x and y, the geographic signals we deal with can also be called functions.

A signal, in communication, is the carrier of information. The word usually evokes the idea of a recognizable sound, or of an orderly waveform or 2-D pictorial representation, existing in some sort of background medium which is not the signal. Often, whatever else that is in the medium, which is not the signal, is called "noise", with

the same logic that whatever that is not cultivated or desired is called a "weed".

2.6.1 Information as disorder

Noise is sometimes given a more precise definition, in terms of its disorderly, random character. By this definition, noise is supposed to be uncorrelated: you can't predict its value at one point, based on its value at another point.[8]This leads to an apparent paradox: the information content of a signal is identified not with its order, but with its disorder, because the only part that imparts any information is that which comes as a surprise [65]. Any other aspect of the signal "goes without saying", according to the context of the signal and its rules of formation. So in communication, and in the study of nature, our attention seems to be on the "noisy" quality, or disorder, existing within signals whose presence is discerned by their order, in contrast to a background that lacks that order. And the role of interpolation can only be to make explicit that which is implicit, to accommodate the interpreter by furnishing an orderly background, so to speak, against which to perceive the disorder of the data, and perhaps also better to perceive the implications of its inherent order.

2.6.2 The figure-ground context

So our view is that there is a complementary figure-ground relationship between signal and medium, and likewise between the information content of a signal and the typical order of that signal. In both cases, the figure is seen in the context of its ground. Perhaps because of the complementarity of the relationship, it seems to be fruitful to maintain a view across the boundaries of context, in order to clarify the content of each. I will give three non-academic examples of this idea, which has as much to do with perception and understanding as it has to do with any objective characteristics of information, signals, or media.

[8]See [62] for a classic treatise on noise.

The first example, which would seem most directly related to matters of geography, remote sensing, and so forth, comes from observations made while stationed at Wolf Mountain fire lookout in central Oregon. In order to be able accurately to report the location of a sighted smoke, one wished to become very familiar with the true detail of the terrain, in relation to the way it appeared from the vantage point of the tower. It turned out to be easier to judge distance and topography on a partly cloudy day than on a clear day. The cloud shadows seemed to lend relief and form to canyons and ridges, as if providing hints here and there for the interpolation machinery of the visual perception system to act upon.

What seemed remarkable about this was that the cloud shadows did not have an order that was known beforehand; they could have been called "noise", added to the scene. Yet this "noise", or this additional signal, evidently illuminated the terrain signal. Further reflection on the phenomenon revealed that this communication synergy operated both ways across the boundary of context: knowledge of the terrain allowed interpretation of the cloud-shadow patterns, while knowledge of the cloud-shadow patterns allowed interpretation of the terrain. And while this crossing of context boundary back and forth may sound like circular reasoning, one realizes that it may not be, since it is carried out more or less globally, across the spatial domain. (Also, the temporal aspect may come into play, as the cloud shadows move over the terrain and more is surmised about their pattern and hence about that of the terrain.) So one discovers a sort of information "bootstrap" effect, by virtue of viewing across boundaries of context.

The second example concerns the reception, by ear, of Morse code telegraphy over radio. As the reader probably knows, in telegraphy one sends words letter by letter, according to an alphabet of dots and dashes, together with certain rules of timing. It turns out in practice that, in manual telegraphy, one can often recognize another person on the air by his "fist". There is recognizable order in his transmission, which is

mainly a departure from the rules of timing, which express the prescribed, or typical, order. (There may be other patterns, such as signal quality, keying envelope, choice of words, that help to identify the sender.)

The unique order of the signal should be distinguished from the unique disorder of the message it carries, even though information was gained, in the larger context, by the very recognition of the signal and perhaps of something about the identity its source. And as in the case of the cloud shadows on the terrain, perception across boundaries of context seems to enhance intelligibility. In reading radiotelegraphy, one is constantly engaged in a mental game of anticipation *vs.* surprise. Orderly characteristics of the signal allow assessment, and subsequent anticipation, of background conditions, such as cyclic fading, which then feeds back to anticipation regarding the signal. If the signal fades entirely or becomes unintelligible, one interpolates, or extrapolates ("fills in") mentally, anticipating the return of the thread, and quickly adjusts to whatever surprise information it carries. It seems that the context space in which one is operating is always in flux, now collapsing, now expanding. So again in radiotelegraphy communication, one encounters the figure-ground information interplay which suggests that we might pay more attention to variable context, in addressing questions of information, communication, and interpolation.

The third example is a story about tracking. It illustrates the role of the imagination in constructing a hierarchy of tentative contexts, and is not so far removed from the activity of those who work with spatial data representation and interpretation.

Seeing marks in the snow, it is first obvious that they are animal tracks. They were noticed because they were a deviation from the expected order of the ground, while something about their own apparent order, in the context of all considered deviations, identified them as animal tracks. Thinking in that context, one imagines likely animals, and comparison is made, of mental models *vis à vis* the observed prints and stride, to further collapse the context: these are the tracks of coyote.

Now an oddity is noticed in the stride, something like left-right-right-left, left-right-right-left, in groups of four. This is information about the gait, possibly about the individual, and it makes the trail easier to follow, because one begins interpolating and extrapolating, filling in the gaps of a tentative picture. But once this information is accepted, seeing its evidence is no longer information, but rather the background against which to detect further information, always as deviation from the expected, in a perception/imagination process of context collapse and model construction.

It is from such experiences that I conjecture that information may in the final analysis consist not of so many unexpected values in a single context (as suggested by Shannon, [65]), nor of the simplest algorithm that could produce the observed signal (for such codes require the context of decoders [30]), but rather information might consist purely of a nesting of contexts.[9] If so, interpolation can be seen as an effort to fill in the innermost context, and it can be regarded as a sort of consequence or projection, proceeding from the model that comprises all the nested outer contexts.

2.7 Compression and Interpolation

The connection between data compression and data interpolation is that both exploit any order that is present in the signal or class of signals considered. Both leverage expected content. Compression does so by avoiding explicit inclusion of that which "goes without saying"; interpolation does so by filling in any expected part that was missing. "Correlation" and "redundancy" are names for the order that compression schemes locate, group, and code. Compression can be viewed as removal of redundancy, or *decorrelation*, and to some extent interpolation conversely can be viewed as insertion of redundancy, or "recorrelation".

Data compression is accomplished in three basic steps, to locate, group, and code information, as mentioned above. The steps are (see, *e.g.*, [27]): (1) application

[9]This notion is partly inspired by the set theory of Whitehead & Russell [75], in which the numbers are built from nothing: 0 is {}, the empty set; 1 is {{}}, 2 is {{},{{}}}, and so forth.

of a *transform*, to reorganize information that is spread out, in the original signal space, into relatively few components, that are uncorrelated; (2) *quantization* of the individual components, which means grouping values that are practically the same, so as not to have to code as many separate values; and (3) *entropy coding*, which is to assign smaller numbers, *i.e.* shorter names, to those values which occur more frequently, so as to reduce the total length of the coded string. To applications of data interpolation, it is the first step, working with transforms, that is most pertinent.

For a one-dimensional discrete signal, in the standard spatial representation, the component *coefficients*, in that representation, are the values at each of the positions. For a two-dimensional grid, the coefficients are the individual pixels' values. In this representation, the "energy" of the signal is spread out: many components have values around the average. In the well-chosen transform representation, however, the energy is likely to be concentrated, so that there are relatively few components with very large positive or negative values, while most of the rest of the components are near zero.

For example, the commonly seen JPEG image compression standard (of 1992) uses the discrete cosine transform (DCT), which is closely related to the discrete Fourier transform (DFT).[10] Under both of these transforms, most natural images, as well as many common one-dimensional signals such as musical sounds, are represented as a few very large components (those corresponding to lower frequency, or wavenumber), and many remaining components that are very small. So the first step in compression using these transforms could be to quantize and code the few important low-frequency components and to consider the bulk of the higher frequency components to have value zero. This kind of band limiting in the frequency domain is also called *low-pass* filtering. Similarly for purposes of interpolation, one might safely assume that higher frequency components (where not known) are likely near zero.

[10]The fast Fourier transform (FFT) is an algorithmic improvement, for efficient computation of the discrete Fourier transform.

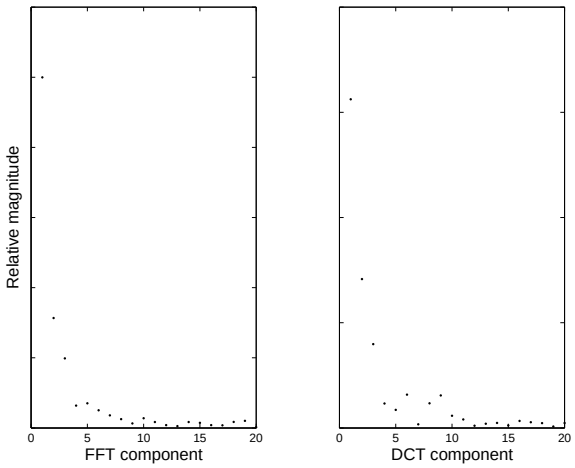


Figure 2.3: Largest components of FFT *vs.* DCT, profile of Figure 2.2.

2.7.1 Compact *vs.* sparse-distributed coefficients

Transforms may result in energy compaction, so that there are only a few large components, but they may not always be the same components. For example, see Figure 2.3, which shows the magnitudes of the first 20 components of the bathymetry profile of Figure 2.2, with Fourier transform representation on the left, and discrete cosine transform representation on the right. Notice that the five largest components of the FFT are the components 1–5, while the five largest components of the DCT are *not* components 1–5; they are components 1, 2, 3, 6, and 9.

In a discrete wavelet transform (DWT) representation (which is the transform basis of the JPEG 2000 image compression standard —see, *e.g.* [66], [73]), even though 90% of the signal energy may be in 10% of the coefficients, it may be quite a different set of coefficients for the next signal to be analyzed. This is the distinction between *compact coding* and *sparse distributed coding* [22], [33]. In the first case, all signals of a class are represented by a certain few coefficients, while in the second case all signals of a class are represented by a small number of coefficients.

Field [22], in considering wavelet-type transforms used by mammalian vision systems, distinguishes between the purpose of information *compression* and the purpose of information *recognition*. He argues that the sparse distributed coding of the wavelet transform is more efficient for purposes of recognition, because a different combination of a small number of nerves can encode the essential information of each different scene. Then memory can be based upon *combinations*, rather than on *levels*. The memory can use a few words from a large vocabulary, instead of many words from a small vocabulary (Hubbard, after Mallat [33]).

The distinction between compact coding and sparse distributed coding can also be understood in connection with *linear* versus *nonlinear* approximations, as discussed by Vetterli [73]: however we transform data and decorrelate coefficients, if approximation is done adaptively, choosing (*a-posteriori*) the coefficients by order of size, instead of choosing always the same coefficients (*a priori*), then the scheme is *nonlinear* because the approximation of a sum of two signals may not be the sum of the approximations of the two signals. (A different set of coefficients may have been chosen for the two signals, or in a different order.)

The distinction between transforms that yield compact coding and those that yield sparse distributed coding has this importance in the matter of interpolation: transforms that yield compact codes (like FFT or DCT *vs.* DWT) will be more useful whenever the signal is looked at in context, as a member of a class of signals whose similarities may be apparent by correlation of their transform coefficients; transforms that yield sparse distributed codes (like DWT) will be more useful whenever the signal is looked at by itself, or in parallel with other members of its class, with attention to autocorrelation rather than to cross-correlation —re Shapiro's Embedded Zerotree Wavelet scheme mentioned in Section 2.3.1. We will present examples of both approaches, the first using a discrete cosine transform characterization of a class of topographic signals, and the second using a wavelet-based reconstruction of irregularly

sampled topographic data, after methods described by Ford and Etter [24].

2.7.2 Information coding and context

We wish to think of signal compression, context, and representation, and its connection to problems of spatial data interpolation, in terms as broad as possible. In this spirit, consider MIDI music representation. MIDI (Musical Instrument Device Interface) is a digital data format for representing music compositions in various specified "voices", such as piano, etc. Computers with the requisite sound card and software are able to render the MIDI files as audio. A MIDI file of piano music is typically about 7kB/min (and can be compressed), as contrasted to MP3 (Moving Pictures Experts Group) format, which is typically about 1MB/min as a compressed audio format.

MIDI could be considered a compressed format, not for audio in general, but for a certain limited class of music. This is true notwithstanding the fact that certain machinery, including a sort of codebook, is required to render it as music. After all, most circuits of communication involve rules and machinery of context. For example, there is a mode of radio voice communication called "single sideband" (SSB), which is about four times as efficient, or compressed, as ordinary AM radio because neither the "carrier" (which carries no information!) nor the sideband which is mirror image to the single sideband that is transmitted, bear any information not present in one single sideband. (*cf.* Footnote 14.) But to be rendered as audio, a local "carrier" must be generated at the receiver, to supply the *context* , so to speak, or the *ground* against which to perceive the *figure* which is the audio signal.

An objection could be raised to the consideration of MIDI as a realization of music compression on the grounds that (differing from MP3 or SSB), it is restricted to certain voices, not capable of reproducing audio waveforms in general. But I think this is just the question of definition of *context* that is at the heart of the problem

of interpolation. To paraphrase Hofstadter's question on the location of meaning [30], does information reside in the MIDI file, or in the MIDI software and hardware, or at their boundary? In fact, neither MP3 nor SSB can represent arbitrary audio information; they are band limited. It is not intuitive to imagine *how limited* a class of signals may be, and even more difficult to imagine *how* a class of signals may be *limited*. But these are the essential questions to ask, when interpolating spatial data such as topography: just how big is this space of possibilities, and where is it located in the larger space of what possibly can be represented?

2.8 Transforms

Transforms will play a central role in the theory and experimental algorithms of this study of approaches to interpolation of spatial data. In essence, a transform is a change of perspective, a rearrangement of information, that might facilitate data recognition, compression, or, as we will show, interpolation. A brief survey of some common transforms is included in the appendix.

In general, a transform does not change a signal or its information content; it changes how a signal is represented. A transform may change one representation of data into another, equivalent, representation that reveals order (*i.e.* predictability) not apparent in the original representation. As discussed in section 2.6, order, predictability, can be considered complementary to disorder, information. This is relevant to interpolation because interpolation is ideally the addition of conformal non-information.

In order to construct a scientific basis for such addition, we wish to gain insight into the order and pattern that is not unique to a single data set, but which characterizes a whole class of data sets. This insight will allow us to "fill in the gaps" with whatever is appropriate to the context, without imparting misleading information. Thus the contextual order, which is characteristic of the class to which the data set belongs, is

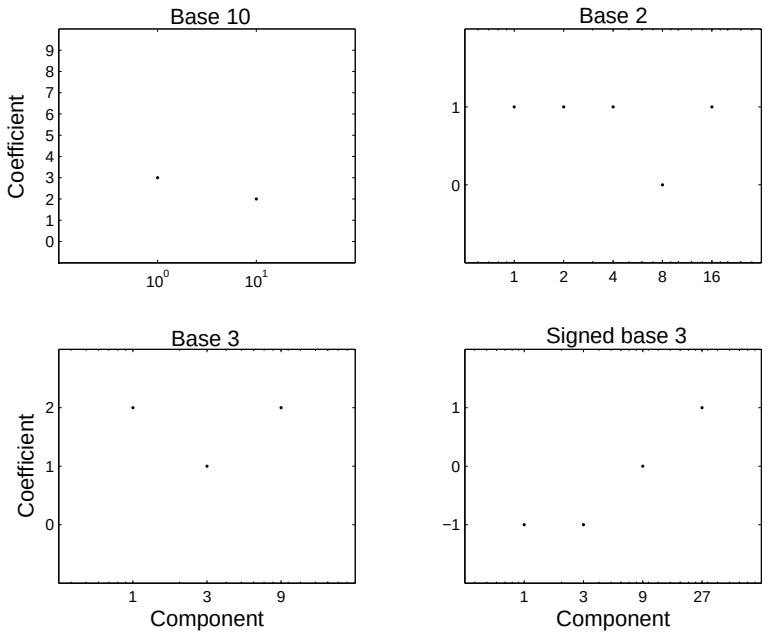


Figure 2.4: Four representations of the number *23*.

the order to which our additions (interpolation) should conform.

2.8.1 Representations of components

We start from the premise that the essence of a signal is its information, not its representation. Distinguishing between object and representation can be subtle, but is more clear when there is more than one representation. For example, the number 23 is represented, in base ten numerals, as *2*3, but in base two, it is represented as *1*0111; in base three it is represented as *2*12, and in a signed base-3 system it may be represented as $+0--$. Graphical versions of the four representations in different bases are shown in Figure 2.4.

It is apparent from these examples that the representations are compositions of parts, or *components*, in specified amounts, called *coefficients*. For base ten, the basis parts (components) are ones, tens, hundreds, etc. (*i.e.* $10^0, 10^1, 10^2 \ldots$ —the integer

powers of ten), with allowed amounts (coefficients) zero to nine of each; for base two, the base components are ones, twos, fours, eights, etc. ($2^0, 2^1, 2^2, \ldots$ —the integer powers of two), with the only allowed coefficients for each being zero or one. For base three, the base components are ones, threes, nines, etc., with allowed coefficients zero, one or two of each; for the signed base-three system,[11] in which both negative and positive numbers can be represented in the same way, the base components are the same integer powers of three, but the allowed coefficients for each are negative one, zero, or positive one (represented here as $-$, *0*, and $+$, respectively).

A transform is a change of the base components of which something is represented to be composed, as when one transforms from base-10 representation of the number 23, as a composition of 2 tens and 3 ones, to the signed base-3 representation of the number 23, as a composition of 1 twenty-seven, 0 nines, -1 three, and -1 one. Note that we have not transformed a number; we have transformed its representation. In this case a representation made up of two components was transformed into a representation made up of four components, as a consequence of the change in the extent of the range of possible coefficients for the components, in going from one representation to the other.[12] Alternatively, we could, for example, regard the base-10 representation of the number 23 to be *0*0023, and the signed base-3 representation to be $0 + 0 - -$, so that both representations have five components. In any case, each *representation* of a number can be thought of as a function defined over a domain that depends on the base of the representation. This is the basis of a graphical representation of a quantitative entity.

[11] Original invention, to author's knowledge.

[12] This complicating detail, which is not entirely peculiar to the number-base example, arises also in considering the complex-valued Fourier transform of a real-valued function, but usually we will be interested in transforms that yield representations with the same number of components and same range of coefficients as the original representation.

2.8.2 Functions and operators

Mathematically, a transform is considered to be a functional operator. This means that a transform operates on a whole input function to yield an output function, in analogy to the way that a function operates on an input number to yield an output number. Sometimes it is said that a function maps a *domain* (the set of input values for the independent variable, which might be called x) to a *range* (the set of values that the dependent variable, which might be called y, can assume). In the foregoing graphical representations of the number 23, the representations were viewed as functions, mapping domains (components comprising the integer powers of the base) onto ranges (coefficients that would be the digits of the numerals in the written representations of the number).

Likewise, a transform maps a domain of input functions to a range of output functions. So the idea of transforms can be thought of as the next step of generalization: transform is to function, as function is to variable. The analogy is helpful, because some of the spatial, geometric, and graphical models that are used to think about numbers and functions can be extended in the imagination to spatial, geometric, and graphical models used to think about functions and transforms. Where we are headed is to use spatial models to think about mathematical abstractions of spatial distributions. For example, we will think of the set of likely digital elevation models (DEMs) of a certain size as a region in many-dimensional function space —as many dimensions as the individual DEMs have grid cells. Transforms will afford views of this region from alternative perspectives in the abstract space, so that we may better choose interpolations that are located within the region of likely DEMs. This will be discussed further as we speak of function spaces and projections from one such space to another.

2.8.3 Discrete functions

We consider primarily transforms of *finite, discrete* functions, because the preeminent tool of scientific data representation today is the digital electronic computer, or more specifically, the pixel raster of the computer display. By *finite* we mean that both the domain of definition and the range of possible values of the function are limited in extent. By *discrete*, we mean that both domain and range are quantized. That means that their values vary by a minimum step size.

For example, a digital elevation model, such as a *GTOPO30* tile (global topography, at 30 arc-second per grid cell), might be defined for a domain that is a rectangular grid 4800 x 6000 pixels, with the range of possible elevation values being integers between -2^{15} and $+2^{15}$. In this case, the domain of definition is limited to 4800 in the x direction and 6000 in the y direction, or 28,800,000 grid cells in all, and the range of possible z values is limited to $-32768 < z < +32768$. The minimum step size of the domain is one pixel (which happens to correspond to 30 arc-seconds on the Earth's surface); the minimum step size of the range is one integer (which happens to correspond to 1 meter elevation on the Earth's surface). We would like our final representation of data to be a finite discrete representation, even though the underlying conceptual model might be infinite and continuous rather than finite and discrete.

2.8.4 Functions as vectors

Discrete functions, since they are characterized by a particular range value for each discrete domain value, can be thought of (mathematically) as ordered collections of numbers —that is, as vectors or as matrices. A vector is a one-dimensional arrangement of numbers, and a matrix is a two- or higher-dimensional arrangement of numbers.

Vectors and matrices can represent finite discrete functions of one or more independent variable in this way: For a function with a given domain of definition for one

independent variable (*e.g.*, x for position along a transect, or t for time), the corresponding range values of the dependent variable can be taken, in a presumed order, as a vector that represents the function. Likewise, a function of two independent variables (such as a digital elevation model with two geographic coordinates x and y yielding an elevation z) can be represented as a two-dimensional matrix of z values, as long as the order of the values is understood.[13]

Let us then consider a one-dimensional discrete function (such as a bathymetric profile) to be a vector. It has as many components as the function has points in the domain of definition. Then, for example, the base-10 representation of the number 23 (without the leading zeros) might be written $[3; 2]$, considered as a vector that has two components —a vector in an abstract two-dimensional space— and the signed base-3 representation of the number 23 might be written $[-1; -1; 0; +1]$, considered as a vector in an abstract four-dimensional space.

Here it bears emphasizing that we have passed from a spatial distribution (the graphical representation of a numeral as a function) through mathematical abstraction, to a spatial model of the mathematical abstraction. It is important not to confuse the space of the distribution with the space of the mathematical abstraction. The four dimensions of the space of the vector $[-1; -1; 0; +1]$ have nothing to do with (pertain to a different space than) the one dimension of the vector itself, and they are also distinct from the two dimensions of the graphical representation of the vector's components. Stated again, in terms of the 256-point bathymetry profile of Figure 2.2, our abstraction proceeds: from a 256-point bathymetry profile (usually viewed as a curve in two dimensions), we go to a one-dimensional vector (256 depth values in order), which we then proceed to think of as a single point in 256-dimension space. As Shannon said, we replace a complex entity in a simple environment with a simple

[13]Notice that, in the graphical representations of 23 shown in Figure 2.4, the vector representations of 23, *e.g.* $[3; 2]_{10}$ or $[-1; -1; 0; +1]_{\pm 3}$ might be taken as the reverse of the written representations, reading left-to-right. In fact, the presumed order for our Arabic numerals is right-to-left, as was the Arabic written language.

entity in a complex environment [63]. The reason for doing so is that powerful spatial intuition might help us in thinking about how the members of a class of functions, or data sets, (including proposed interpolations) are related —how we get from one to another, how close they are, and so forth.

2.8.5 Transforms as matrices

This study is concerned mainly with *linear* transforms, in the system sense defined in section 2.5 above. Linear transforms of finite discrete functions of one variable (which can be represented as vectors) can be identified with matrices. Then, for example, the discrete cosine transform operation $\mathbf{C}$ on a vector $\mathbf{x}$ to yield a vector $\mathbf{y}$ can be written as the simple matrix-vector multiplication

$$\mathbf{Cx} = \mathbf{y} \ .$$

If the vector $\mathbf{x}$ has only six components, for example, the DCT matrix $\mathbf{C}$ can be given explicitly (to four digits) as:

$$\begin{pmatrix} 0.4082 & 0.4082 & 0.4082 & 0.4082 & 0.4082 & 0.4082 \\ 0.5577 & 0.4082 & 0.1494 & -0.1494 & -0.4082 & -0.5577 \\ 0.5000 & -0.0000 & -0.5000 & -0.5000 & -0.0000 & 0.5000 \\ 0.4082 & -0.4082 & -0.4082 & 0.4082 & 0.4082 & -0.4082 \\ 0.2887 & -0.5774 & 0.2887 & 0.2887 & -0.5774 & 0.2887 \\ 0.1494 & -0.4082 & 0.5577 & -0.5577 & 0.4082 & -0.1494 \end{pmatrix} .$$

Row 1 shows how much of each $\mathbf{x}$ component $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \ldots \mathbf{x}_6$ goes into the first $\mathbf{y}$ component $\mathbf{y}_1$; row 2 shows how much of $\mathbf{x}_1 \ldots \mathbf{x}_6$ go into $\mathbf{y}_2$, etc. The reason for showing this is to "demystify" transforms and show that they are really numerical instructions for reassembling information, by summation of distributed parts.

2.8.6 Bases and their interpretation

Some transformed representations of data have bases that readily admit of physical interpretation. The Fourier transform, for example, yields components that are

conveniently interpreted as *frequencies*. Yet other transforms yield representations that do not so easily admit of physical interpretation, and it is convenient to have a conceptual framework that subsumes all transforms as being essentially changes of perspective as to composition.

Implicitly, a signal like a profile of bathymetry is represented as, and thought of as, a composition of depth values at certain places. We may not consciously be aware of it, but the implicit base of this representation is a bunch of discrete unit "impulses", one for each location in the domain (see, *e.g.* [52]). Then the vector representation of the profile is a recipe that tells how much of each unit component to take, added together, to make the representation of the profile in that base.

Figure 2.5 shows the first five components of the implicit "impulse" base for representing a 256-point profile, together with the first five elements of the bathymetry profile of Figure 2.2 as the profile is built up as a sum of products of coefficients times base components. This analysis of representation in the standard base will seem to be less of a tautological construction when less familiar bases are used in the same construction.

A Fourier transform yields a different representation of a signal, as a composition of periodic waveforms of various size and relative phase. For example, Figure 2.6 shows the same bathymetry profile of Figure 2.2, together with a typical Fourier magnitude representation.

Actually, the Fourier magnitude representation is not a complete representation of the Fourier-transformed information of the bathymetry profile signal.[14] A magnitude-only Fourier-transformed representation lacks *phase* information (discussed further in Section 3.1). Infinitely many signals can have the same Fourier magnitude spectrum, while their complete Fourier representations are unique, and hence *invertible* —the

[14]This can be surmised from its symmetry: there are half as many (128) non-redundant points of information as in the original profile and, unlike the base-10 *vs.* base-3 case discussed previously, there is not a corresponding difference in the allowed range of coefficient values.

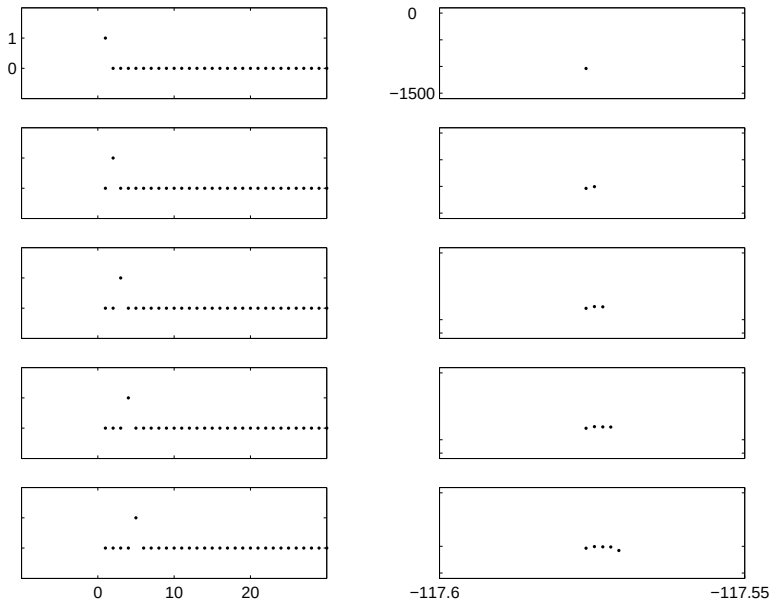
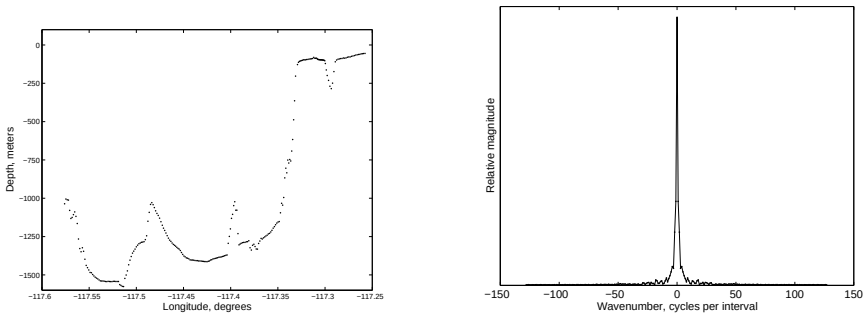


Figure 2.5: First five components of standard base, first five steps constructing representation of profile of Figure 2.2.



Bathymetry profile Fourier magnitude plot

Figure 2.6: 256-point bathymetry profile and 256-point Fourier magnitude plot.

transform to change from the Fourier representation back to the standard representation is as well-defined as the transform to change from the standard representation to the Fourier representation. The discrete Fourier transform (as implemented in `MATLAB`) is given by:

$$\mathbf{X}(k) = \sum_{j=1}^{N} \mathbf{x}(j) e^{\frac{2\pi i(j-1)(k-1)}{N}} \quad ,$$

where $\mathbf{X}(k)$ is the k^{th} coefficient of the transformed representation $\mathbf{X}$ of $\mathbf{x}$;
N is the length of $\mathbf{x}$;
e is the base of the natural logarithms, and
i is the imaginary unit, $\sqrt{-1}$.
The inverse discrete fourier transform is given by

$$\mathbf{x}(j) = \frac{1}{N} \sum_{k=1}^{N} \mathbf{X}(k) e^{\frac{2\pi i(j-1)(k-1)}{N}} \quad .$$

The symmetry of these expressions may be more apparent by reference to Figure 2.7, which shows, in the left column, standard representations of the first five components of the Fourier base for 256-point signals; in the right column are Fourier representations of the first five components of the standard base, *i.e.* the impulses which were shown on the left in Figure 2.5. So the view of "our" world from the other side of the Fourier looking glass is just the mirror image of our view of the Fourier representation!

The reason that the representations in Figure 2.7 are shown as 3-D helices is that complete Fourier analysis is conveniently done with complex numbers, which have two components, so that for every x (in this case, 1–256) there is both a y and a z, which can be considered as the *real* part and the *imaginary* part, or as the *cosine* component and the *sine* component, or as the *magnitude* and the *phase*.

To illustrate how the bathymetry profile of Figure 2.6 can be constructed as a sum of components, as in Figure 2.5, but this time using the first five Fourier components in turn, see Figure 2.8. Each profile descending in the right column incorporates one more of the Fourier component-coefficient products shown in the left column. Notice

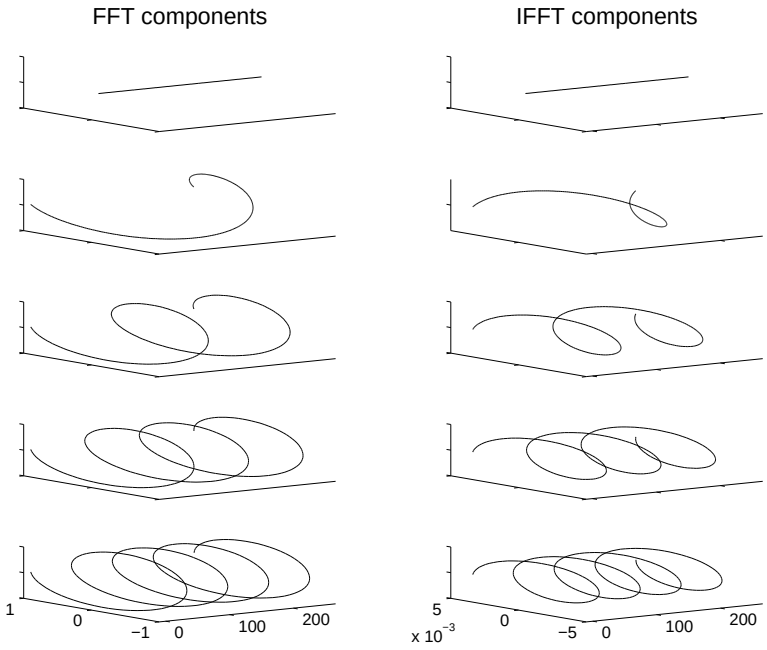


Figure 2.7: Fourier components in standard base; standard components in Fourier base.

that this figure is not a strict counterpart to Figure 2.5; the left column here shows the Fourier base components multiplied by their respective coefficients, so that the variations in amplitude and phase of the parts can be seen. It is apparent that, in contrast to the local construction illustrated in Figure 2.5, this process is a gradual refinement of global characteristics of the signal.

2.8.7 Orthogonality of bases and the generalization of "direction"

Most commonly used or "natural" representations of signals employ bases whose components are *orthogonal* in the same sense that the Cartesian coordinates x, y, and z are orthogonal, or mutually perpendicular: regarding the space itself, a shift in one coordinate direction entails no necessary shift in any other coordinate direction. An orthogonal base is desirable because of its economy: the number of components

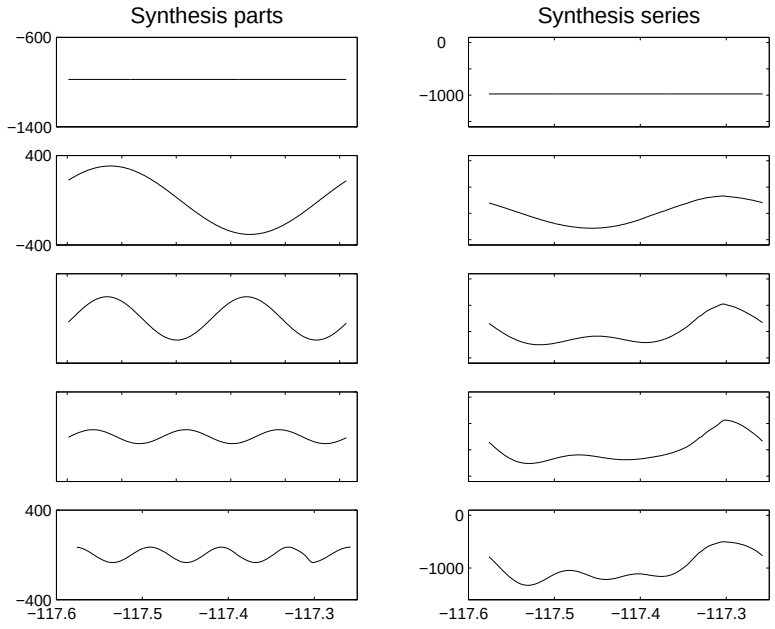


Figure 2.8: Fourier components and synthesis of profile as series.

required to specify a point is equal to the number of dimensions in the space.

The components of most transform bases for representing signals are families of *orthogonal functions*. Roughly speaking this means that, as points in many-dimensional function space, they lie in mutually perpendicular directions with respect to the origin. Formally stated, two functions, or equivalently two vectors, are said to be orthogonal if their *cross product* is zero:

$$< \mathbf{x}, \mathbf{y} >= 0, \text{where} < \mathbf{x}, \mathbf{y} >\doteq \sum_{i}^{n} \mathbf{x}_i \mathbf{y}_i \ ,$$

for

$$\mathbf{x} = [x_1, x_2, x_3, \ldots x_n]$$

and

$$\mathbf{y} = [y_1, y_2, y_3 \ldots y_n]$$

.

Figure 2.9 shows the first two members of the FFT family and the first two members of the DCT family for a space of 256 dimensions. The FFT components are shown on the left as separate Real (blue) and Imaginary (green) parts in the same plots. The cross product for any pair of these functions (and for any of 32,634 pairs not shown) is zero, and likewise for the DCT component functions shown on the right.[15]

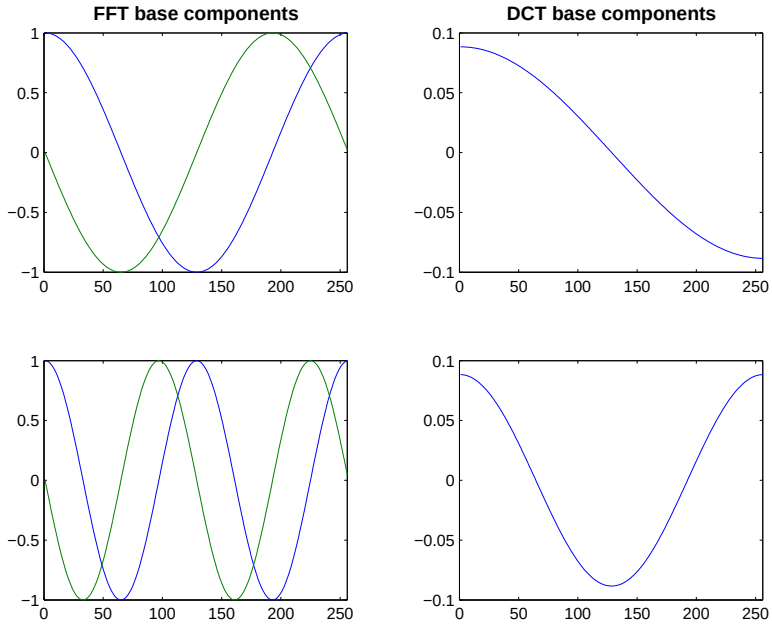


Figure 2.9: First members of orthogonal family of base functions for FFT and DCT.

It may seem odd that a family of such similar-looking functions would all be as distinct as "up" and "sideways", especially since they all appear to be horizontally-squeezed copies of each other. But they are distinct in the sense that no one of them could be built as the sum of *vertically* squeezed or stretched copies of any number of the others (because that is what it means to multiply a coefficient by a component).

[15]It is interesting to note that the orthogonality of the FFT functions, which appear as harmonics of one cycle, is shift-invariant, while the orthogonality of the DCT functions, which appear as harmonics of a half cycle, is not shift-invariant.

So what is it for one function to be similar to another, to have the same shape? It is possible to generalize further from the idea of orthogonality applied to functions in an abstract space, and define the *angle* between two functions to be the angle whose cosine is the cross-product of the functions (see *e.g.* [2]). Stated formally in terms of vectors,

$$\alpha = \cos^{-1}(< \mathbf{x}, \mathbf{y} >)$$

If the *angle* thus defined between two functions is small, the functions will appear to have the same *shape*.

2.8.8 Mixing Space and Time

Where there are spatial patterns there are usually accompanying temporal patterns [48]. The "signals" of nature that we would interpolate exhibit order that can be considered to be the result of underlying processes: "the flow of energy through a system tends to organize that system" [26]. Most of the patterns that geographers study are what Ilya Prigogine named *dissipative structures*, which exist in nonequilibrium systems sustained by the flow of energy and matter [38].

Since this flow of energy and matter, or any process underlying an observed geographic pattern, inherently extends in *time* as well as space, it is natural to view all geographic signals spatio-temporally. Likewise, sampling typically extends in time as well as in space. The foregoing remarks apply especially to remotely-sensed data gathered by satellites, whose combined passes represent a spatio-temporal section of a "signal" extending in space and time. In oceanic and atmospheric studies, the temporal scale for propagation of patterns is comparable to the temporal scale of sampling. Even if a purely spatial representation is desired (*i.e.* a "snapshot"), interpolation may have to be carried out in both spatial and temporal dimensions.

Partial and Relative Nature of Components

As has been suggested earlier, *sampling* of the world is necessarily partial, rather than complete, and the *model* of the world that we behold at any time might be considered to be a non-unique point of view in many-dimensional function space. Any signal of interest in this model is therefore a sum of components, such as field values at certain times and at certain locations, that can be considered to be neither absolute nor inmiscible, but rather to comprise one partial representation relative to other possible representations. A transform might result in an alternative representation in terms of distributed sums of both spatial and temporal components, so that the new components mix spatial and temporal aspects.

In "filling in the gaps" across a spatial dimension or across a temporal dimension, there is no reason to maintain separation of these dimensions, any more than one would strictly separate east-west interpolation from north-south interpolation in a DEM; rather one would want to use all dimensions of sampling for an optimal interpolation that combines the information available in all dimensions of sampled data.

Complex Empirical Orthogonal Function Analysis

Complex Empirical Orthogonal Function (CEOF) analysis is an example of this approach, and has been applied to interpolation of sea surface height data gathered by satellite altimetry, for the purpose of studying propagation of mesoscale eddies [81]. CEOF analysis is essentially a two-dimensional combination of Principal Component analysis and analysis by *complex* component functions (as in the case of Fourier analysis). The base transform seeks the *eigenmodes*, that are the mutually independent spatio-temporal data distributions that can be regarded as alternative constituent components of the original data field, presented in order of the amount of data variance that they account for. Each of these eigenmodes is in turn the product of

a spatial function component —complex, meaning that it has both magnitude and phase— and a temporal function component —also complex. Interpolation can be carried out by separately interpolating the phase part and the amplitude part, of both the spatial and the temporal wave components, of each eigenmode. The summed result is a smoother interpolated representation of the propagation of wave-like features through both space and time, than would be obtained by simpler interpolation methods that assumed (in effect) that the propagating feature's amplitude decreased at those locations or at those times between samples where it was not "caught" at its peak amplitude.

2.8.9 Periodic Functions as Projections

Part of the reason that the CEOF analysis works well for interpolation of sea surface eddy propagation is that it better represents the essential continuity of a signal that is most simply represented to be *supported* at once in three dimensions –*viz.* latitude, longitude, and time, rather than separately in space and in time.[16] The sampling of periodic signals can be misleading, if it is not understood that a sample value zero does not necessarily indicate absence of a wave; the sampled point may just be a node, between positive and negative peaks. In general, a point sample is not a measure of the instantaneous magnitude of a periodic function (*i.e.* a measure of the amplitude of its oscillation), if the phase of the periodic function (where it is on its cycle) is not known.

When dealing with a signal known to be periodic, it is sometimes advantageous to view the signal as a projection from a higher dimension.

As shown in Figure 2.10, a regular oscillation up and down can be regarded as a sideways view of something that is simply going round in a circle.[17] In fact, there

[16] *Support* here is used to mean the dimensions of the independent variables.

[17] This is the essence of Euler's formula relating the exponential and trigonometric functions, $e^{it} = cos(t) + isin(t)$.

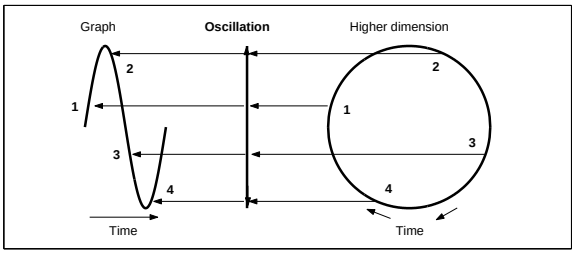


Figure 2.10: An oscillation viewed as a projection of a rotation in higher dimension.

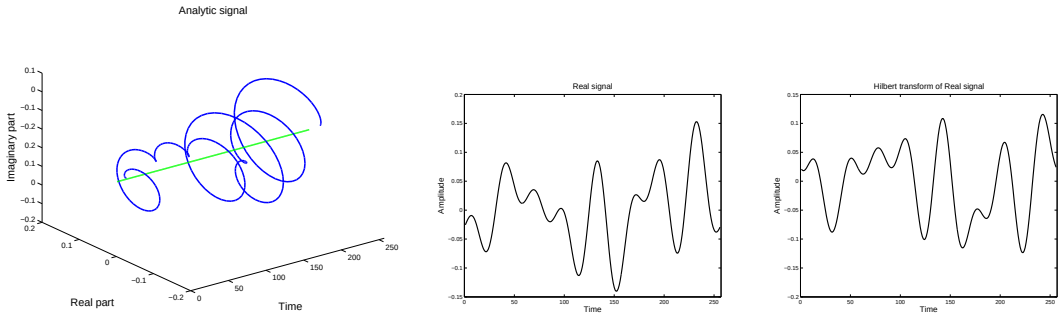


Figure 2.11: Analytic signal and its Real and Imaginary parts.

is a mathematical method to project a real-valued one-dimensional periodic signal back to a complex-valued two-dimensional signal, called the *analytic signal*.[18] The components of the IFFT transform, shown in Figure 2.7, can be considered to be the analytic signals derived from pure cosine waves for frequencies $0, 1, 2, \ldots$, up to the Nyquist frequency for that number of samples.

Figure 2.11 shows the analytic signal derived from a randomly-generated 256-point signal that is band-limited to 8 cycles over the 256-point interval. The zero axis of the helical plot is shown in green for reference. The view from "above" (*i.e.* the Real *vs.* Time plane) is the same as the Real signal graph, shown in the center of

[18]The analytic signal is obtained from the real-valued signal by adding the *Hilbert transform*, which is the imaginary-valued signal, all of whose frequency components lag 90 degrees phase with respect to the real-valued signal.

Figure 2.11. The view from the "side" (*i.e.* the Imaginary *vs.* Time plane) is the same as the Hilbert transform graph of the signal, shown on the right of Figure 2.11. With the higher-dimensional view, the instantaneous magnitude of the signal is seen to be simply the radius of the circulation. The analytic signal is also a "simpler" representation of the signal in question insofar as its Fourier transform has half as many non-zero components –all negative frequency components are zero.[19]

Appendix C is an introduction to nonlinear time series analysis, which I think can also be understood as a sort of transform that mixes time and "space" (state space), and which likewise can result in a "simpler" representation of an observed signal as a projection of a trajectory in higher dimension. Again (after Shannon [65]), one hopes to substitute a simple entity in a complex environment for a complex entity in a simple environment.

[19]For real-valued signals, the *real*, or cosine, frequency components have *even* symmetry —negative components same as positive components— and the *imaginary* or sine, frequency components have *odd* symmetry —negative components opposite of positive components.

3 Applications

The applications presented in this part are not intended to be solutions to particular interpolation problems, but rather illustrations of how the theory of the first part might be applied in practical algorithms. I hope that they can be improved upon.

3.1 A Magnitude *vs.* Phase Example

As mentioned in Section 2.8.6, the Fourier transform yields either one set of *complex* coefficients that have two components, or else two sets of *real* coefficients that have one component, depending on how you want to look at it. This comes about because Fourier analysis treats all signals as periodic functions, or oscillations, potentially complex-valued.

Having chosen to think of Fourier coefficients as belonging to two sets, one can think of them as the *sine* family and the *cosine* family, or as the *magnitude* family and the *phase* family, just as one can choose between Cartesian coordinates or polar coordinates. To represent what was a unified family of coefficients, *viz.* the complex Fourier coefficients, as the two distinct families of coefficients *magnitude* and *phase* is another sort of transformation that has the potential to reorganize information. In this section we will explore the proposition that *magnitude* spectrum tends to be associated with global characteristics, while *phase* spectrum tends to be associated with local characteristics.

Often in geographic Fourier spectral analysis, phase is ignored. Some image-processing software applications with FFT capabilities do not depict phase nor allow direct manipulation of it. The relation between fractal dimension and Fourier magnitude spectrum has been investigated for sea floor topography [25], but phase has been treated as random in terrain analysis. On the other hand, in many cases of signal analysis, it has been shown that there is more information in phase than in

magnitude [53].

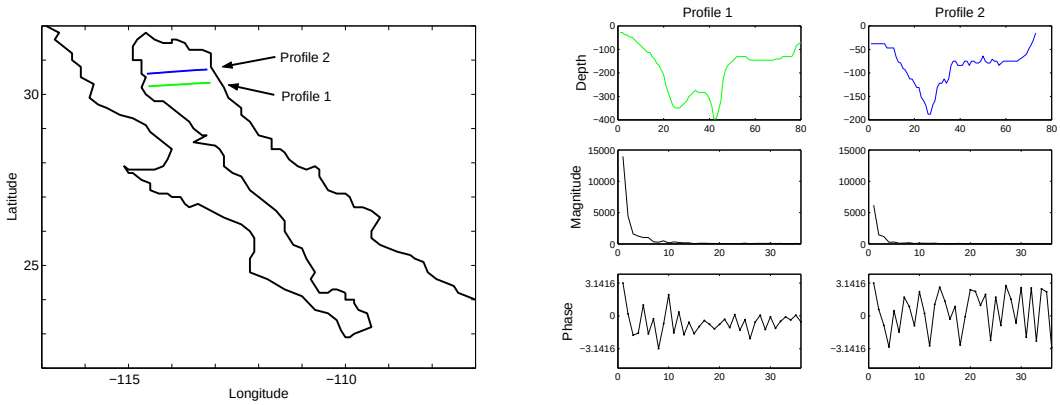


Figure 3.1: Two transects in Sea of Cortez; depth profiles with Fourier magnitude and phase representations.

Figure 3.1 shows the location of two transects in the Sea of Cortez together with the associated depth profiles. Below the depth profiles are shown the first halves of the (symmetrical) Fourier magnitude and phase representations, comprising 36 data points including the zero frequency, or "DC-bias" components. It is apparent that, while the magnitude plots show similar "$1/f$" shapes, the phase plots are quite distinct.[1]

To investigate which terrain characteristics might be represented by magnitude, *vs.* which terrain characteristics might be represented by phase, we synthesize a profile, taking the magnitude components of Profile 1 and the phase components of Profile 2. The result is shown in Figure 3.2.

Evidently much of the recognizable shape of Profile 2 was captured by using its phase spectrum against the magnitude spectrum of Profile 1. Notice, however, the different depth scales: the greater bias (*i.e.* average depth) and amplitude of Profile 1, also apparent by the size, but not shape, of its magnitude spectrum as shown in

[1] One should imagine the phase plot wrapped into a cylinder lengthwise; $-\pi$ and $+\pi$ are equivalent, so the "amplitude" of the plot must be interpreted carefully.

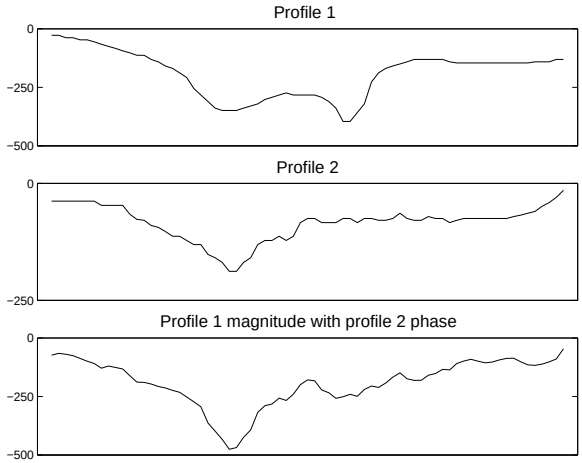


Figure 3.2: The two profiles of Figure 3.1, and a profile synthesized from the magnitude of the first and the phase of the second.

Figure 3.1, are reflected in the hybrid synthetic profile.

This experiment suggests that Fourier magnitude spectra can be considered to carry "global" information, while Fourier phase spectra carry more local information. However, a final observation is that the hybrid profile seems to exhibit texture more like that of Profile 2 (its phase kin) than that of Profile 1 (its magnitude kin).

3.2 Image pyramid super-resolution

In a patent for "fractal-based image compression and interpolation" [57], Pentland *et. al.* describe an algorithm for super-resolution of images based on presumed self-similarity between image sub-bands. The term "super-resolution" here refers to a procedure for multiplying the number of data points in a regular grid of data. In this section we test the effectiveness and computational practicality of an application of the algorithm to interpolation of a digital elevation model. As a reference data set we take a 512 x 512 (henceforth referred to as 512^2) grid from the GTOPO30 data set for an area including part of the Rio Fuerte drainage in the Sierra Madre

of northwestern Mexico, northeast of Los Mochis. A regular subsample of this grid, 128^2, is taken as the starting grid for interpolation. The subsampled data is therefore at 1/16 resolution in terms of area, or 1/4 in each direction.

3.2.1 The method of Pentland, Simoncelli, and Stephenson

The method is based on the multiresolution pyramidal decomposition of images by filtering and subsampling, in a cascade process closely related to wavelet analysis.[2] The general idea is to use a low-pass filter to blur the image, and subsample this at 1/2 to obtain the next smaller *Gaussian* image with half as many pixels in each direction. The inverse of this step is to "up-blur" the Gaussian image, doubling the number of pixels in each direction by inserting pixels of value zero between existing pixels, and then filtering again to smooth the result. The difference between the up-blurred Gaussian image and the original image is the first *Laplacian* image (which has as many pixels as the original image), called $\mathbf{L_1}$.

At this point, the original image can be discarded, because all information is contained in the Laplacian image and the Gaussian image at half resolution in each direction. The decomposition can now proceed to the next step, treating the Gaussian image as the first image. A second Gaussian image and a second Laplacian image, called $\mathbf{L_2}$, are generated, the Gaussian image used at this stage can be discarded, and so forth. A multi-resolution *Laplacian pyramid* of images is thus constructed (so called because each is half the size of the previous in both directions), and the original image can still be recovered by reversing the decomposition cascade.

The super-resolution algorithm is based on finding the cross-scale correlation that might exist between Laplacian images. At any stage, generating the next higher resolution image is a matter of "up-blurring" a given Gaussian image and adding the next higher Laplacian image from the inverted pyramid. So we could interpolate an

[2]The "PyrTools" MATLAB toolbox of Eero Simoncelli contains a nice tutorial on pyramidal decomposition. See software list appendix.

image if we could determine what the next higher Laplacian image is like. Going "up" the Laplacian pyramid (from lower resolution to higher resolution), one pixel of $\mathbf{L_1}$ goes to four pixels of $\mathbf{L_0}$. How can the four new pixels be guessed?

If there is scale-wise self-similarity in the image, we might guess that the details of similar regions expand in similar ways. So the algorithm is to make the unknown expansion from $\mathbf{L_1}$ to $\mathbf{L_0}$ similar to the known expansion from $\mathbf{L_2}$ to $\mathbf{L_1}$ by searching for the 9-pixel block in $\mathbf{L_2}$ that is most similar to the 9-pixel neighborhood of each pixel in $\mathbf{L_1}$, and expanding that pixel to the corresponding four pixels of $\mathbf{L_0}$ exactly as the center pixel of the best-matching 9-pixel neighborhood of $\mathbf{L_2}$ was expanded to the corresponding four pixels of $\mathbf{L_1}$.

The interpolation can be iterated, searching again in $\mathbf{L_2}$ for the best 9-pixel match to the neighborhood of each pixel of $\mathbf{L_0}$ (previously generated by interpolation), again generating the corresponding four pixels of $\mathbf{L_{-1}}$ the same as the four pixels of $\mathbf{L_1}$ corresponding to the center pixel of the 9-pixel block of $\mathbf{L_2}$ that was the best match to the 9-pixel neighborhood of that pixel of $\mathbf{L_0}$. Each iteration of interpolation refers back to the known Laplacian image pair $\mathbf{L_1}$ and $\mathbf{L_2}$.

3.2.2 Particulars of this implementation

In this experiment we carry out two stages of interpolation, from 128^2 to 256^2 to 512^2. The simulated low-resolution grid is generated by subsampling and filtering the reference DEM. Starting with a DEM grid 128^2, the first Laplacian image, called $\mathbf{L_1}$, is 128^2. The 9-pixel neighborhood search must be carried out in the second Laplacian image, called $\mathbf{L_2}$, which is 64^2. The number of contiguous 9-pixel blocks is $62^2 = 3844$: that is the size of the set to search for best match.

The criterion used for best match is minimization of the squared norm of the difference, *i.e.*

$$\sum_{i=1}^{9} (L1_i - L2_i)^2 \ ,$$

where $L1$ is the 9-pixel neighborhood in $\mathbf{L_1}$ and $L2$ is the 9-pixel block in $\mathbf{L_2}$. (This is the square of the Euclidean distance between the pixel groups, considered as points in 9-dimension function space.)

Filter and scaling considerations

In this invertible pyramid decomposition scheme, the filter used is a binomial 5 filter[3] It is scaled to have norm 2, explicitly:

$$\begin{pmatrix} 0.0078 & 0.0313 & 0.0469 & 0.0313 & 0.0078 \\ 0.0313 & 0.1250 & 0.1875 & 0.1250 & 0.0313 \\ 0.0469 & 0.1875 & 0.2813 & 0.1875 & 0.0469 \\ 0.0313 & 0.1250 & 0.1875 & 0.1250 & 0.0313 \\ 0.0078 & 0.0313 & 0.0469 & 0.0313 & 0.0078 \end{pmatrix}$$

This has the shape of a Gaussian "hill" in two directions and acts as a low-pass filter when convolved with the data to yield a "Gaussian" image. The reason it is scaled to have norm 2 is that, in the decomposition-reconstruction procedure, it will be applied twice, resulting in multiplication by four, which just compensates for the upsampling performed in the reconstruction step, where the addition of three zeros for every one matrix value effectively diminishes the average amplitude of the matrix by a factor of 1/4.

This means that the second Gaussian image will have amplitudes twice that of the first Gaussian image, and so on up the image pyramid toward coarser resolution. The same is true of the Laplacian images. So in comparing Laplacian images in a search for similar neighborhoods, it is for this reason appropriate to divide the Laplacian images by 2 for each step back up the pyramid toward coarser resolution.

But there is more. The amplitude of detail at any level of resolution is expected to vary more or less with the horizontal scale of that detail, so that it is appropriate also to divide the Laplacian images by a *second* factor, *viz.* the ratio of detail amplitude

[3]So called because it is derived from the coefficients of the expansion of a binomial, *e.g.* $(a+b)$, raised to an integer power: $(a+b)^4 = 1a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + 1b^4$, hence the 1-D filter vector [1;4;6;4;1].

at one resolution to the detail amplitude at the next finer resolution. The statistics of the Laplacian pyramid derived from the given data provide a means of estimating this ratio.

If the spectrum of terrain is supposed to exhibit a $1/f$ characteristic, one might guess that the amplitude of the Laplacian might be expected to halve, for each step down the pyramid toward finer resolution. However, in this case the apparent ratio for each step toward finer resolution was about 1/1.3 rather than 1/2, as shown in Figure 3.3.

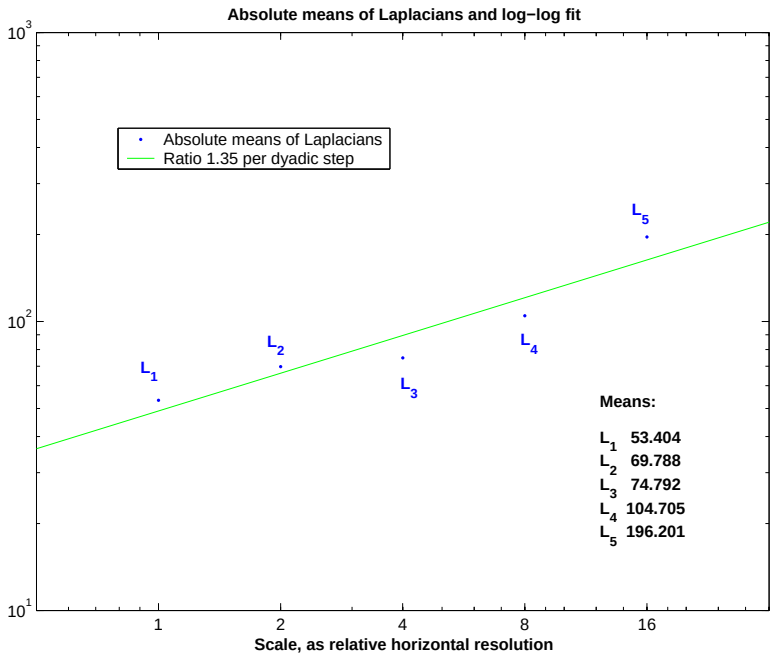


Figure 3.3: Absolute means of Laplacians and derivation of scaling ratio.

This suggests a $1/f^{0.43}$ spectral characteristic instead of a $1/f$ characteristic, although this relation is inferred by differences between subbands which really overlap to some extent.

Because the number of data in the Laplacians decreases inversely by powers of two, and because the scaling ratio may not be constant across scales, I chose to

use the ratio between absolute means of $\mathbf{L_1}$ and $\mathbf{L_2}$, which is 1.307, as scaling adjustment factor, rather than using the ratio obtained by the log-log least-squares fit illustrated in Figure 3.3. The effect of the filter-compensating adjustment and the scale-compensating adjustment is seen in the histograms shown in Figure 3.4. The greater similarity in the distributions is expected to result in better matches found in the search process.

Adjustment of the Laplacians is not mentioned in the algorithm described in Pentland's patent [57], but the consequence of these two considerations is that the optimal comparison between Laplacians $\mathbf{L}_1$ and $\mathbf{L}_2$, in order to extrapolate $\mathbf{L}_0$ for the first stage super-resolution, should use $\mathbf{L}_2/2.614$ for the comparison.

Results

The starting data, simulated low-resolution at 120 arc-seconds/pixel, derived from the GTOPO30 DEM of the Rio Fuerte drainage mentioned above, is displayed on the left in Figure 3.5 in gray-scale shaded relief; on the right is the reference data for the southwest quarter of the grid, at 30 arc-seconds per pixel. Figure 3.6 shows a standard image interpolation at 30 arc-seconds per pixel of the same southwest quarter using Lanczos filter (which is well-known for image magnification), on the right is the result of the super-resolution interpolation, also at 30 arc-seconds per pixel.

The result is not entirely disappointing; it is an improvement over the ordinary "smooth" interpolation represented by the image magnification.[4] A notable weakness of the method is its failure to extrapolate drainages nicely to finer scale. Perhaps an analogous vector-based super-resolution algorithm could be used in conjunction with the present raster-based super-resolution algorithm, to achieve better results.

[4] I cannot explain the block-shaped anomaly in the image magnification.

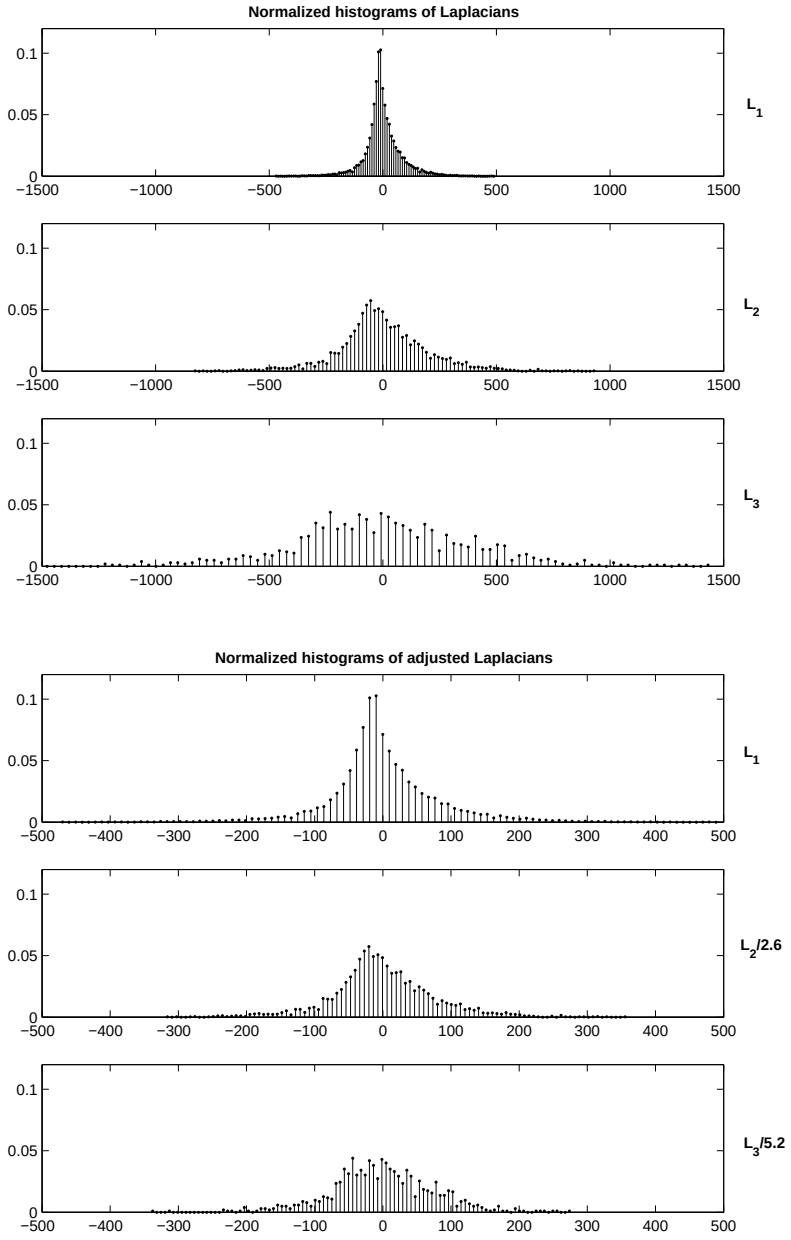


Figure 3.4: Histograms of Laplacian images. Before (above) and after (below) adjustment for filter and scaling effects.

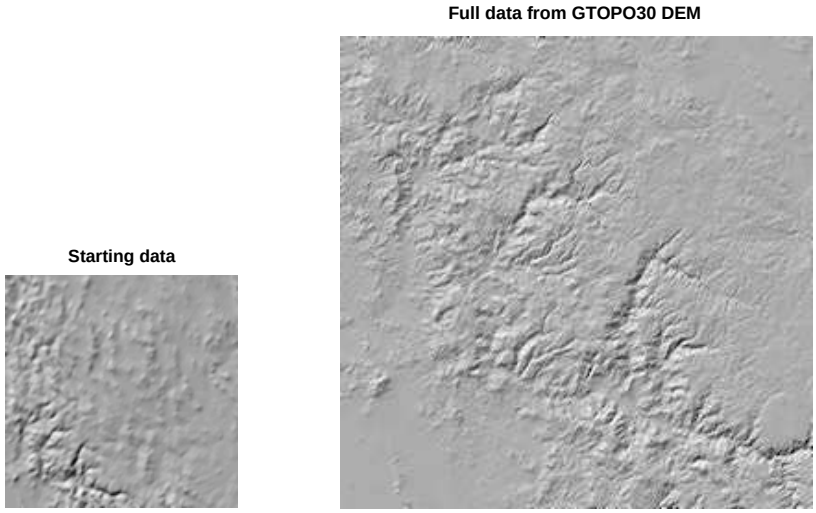


Figure 3.5: Simulated low-resolution DEM data, and southwest quarter of reference data. Low-resolution data (left) 128 x 128 pixels, at 120 arc-seconds/pixel; reference data at 30 arc-seconds/pixel.

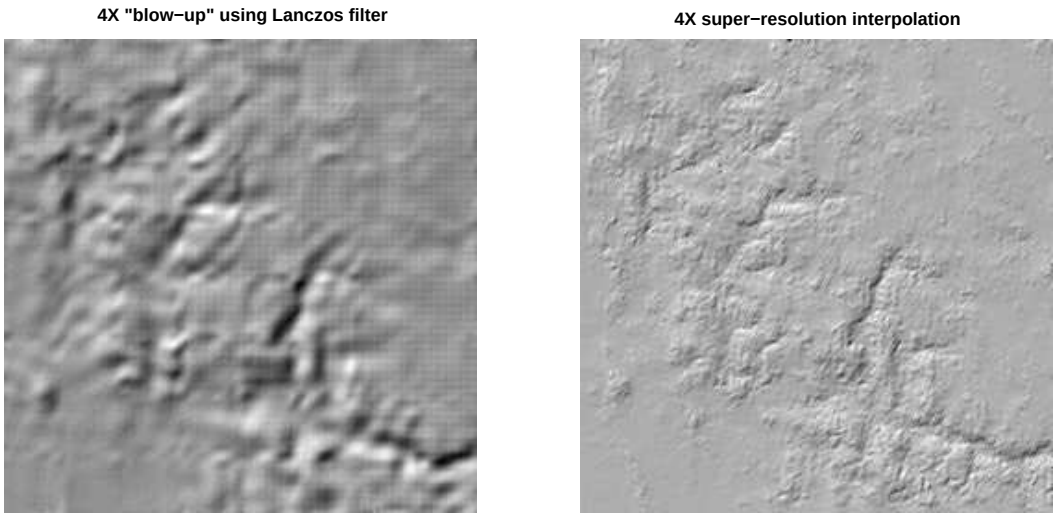


Figure 3.6: Image magnification (left) and super-resolution (right), for the southwest quarter of the starting data.

3.2.3 The importance of sampling mode in scale analysis

Earlier it was mentioned that samples are sometimes considered to be point values (such as bathymetry soundings), while sometimes they are considered to be integrals of areas (such as the pixel values of satellite-borne radiometers).[5] Experiments with scale-based analysis described in this section and the next section drew attention to the importance of distinguishing between two cases of uniformly-sampled "low resolution" data, to which one might attempt to apply the "super-resolution" methods of interpolation. The two cases are uniformly distributed point samples, with no data for intervening points, on the one hand, and a grid representing complete coverage by integral samples of low pixel density, on the other. The "fractal-based" methods which presume scale-wise self-similarity are more appropriately applied to the case of integral samples. This can be explained by pointing out that revelation of the similarity of form in data at two different scales depends on viewing the data at those two different scales, more or less to the exclusion of other scales. If the low-resolution data are of the point-value sort, then interpolation methods based on self-similarity might produce better results if the data are first filtered to simulate integral samples.

Aliasing

Aliasing refers to artificial patterns that arise from discrete sampling, especially as the result of a simple relation between the periodicity of the sampling and the periodicity of the signal being sampled. Common examples include the stair-step appearance of diagonal lines in subsampled images, and wagon wheels appearing to turn backwards in old western films. Aliasing is usually understood in these terms: if there is something going on at a frequency higher than the sampling frequency, the sample values may describe a pattern that is really the interference pattern of the sampling frequency and the higher frequency in the data being sampled.

[5] "Point" measurements are always blurred spatially or temporally, when examined finely enough.

The multiresolution analysis considered in this section and in the following section, in connection with image pyramids and wavelet analysis, is sometimes said to decompose a signal into *subbands*, which refer to frequency bands. If the hierarchy of Laplacian images described above is viewed in the customary 2-D Fourier transform magnitude representation, with low frequency at the center and high frequency at the periphery, the Laplacian components do indeed appear as squarish annular bands at decreasing radius from the center. Hence the use of *filters* (which term in signal processing usually evokes frequency discrimination) to separate the subbands of multiresolution analysis.

If the data being sampled belong to a fractal set, they will necessarily have frequency components higher than the sampling frequency, and for this reason the characterization of scale-wise self-similarity can be confounded by the very fact that finer-scale form obscures coarser scale form.

Filtering point samples

Applying super-resolution methods to a regularly-spaced subset of the reference DEM, as an example of point samples, was problematic. It is not clear what scaling ratio to use with the Laplacian images (see Figure 3.7). Using scaling ratio 1, the ranges of the resulting super-resolution images were 20 to 25 percent larger than those of both the starting data and the reference data. An *ad hoc* correction factor of 1/3 was applied to the extrapolated Laplacian images $\mathbf{L_0}$ and $\mathbf{L_{00}}$ in order to maintain elevation ranges consistent with the starting data and the reference data.

To test whether simple filtering of the point samples would improve the outcome of the super-resolution algorithm, another interpolation was performed on the point-sample data after filtering with the same "binomial 5" filter used in the other parts of the algorithm. This was supposed to prevent aliasing and ensure that coarse-scale form was not confounded by finer-scale false texture arising from separated sample

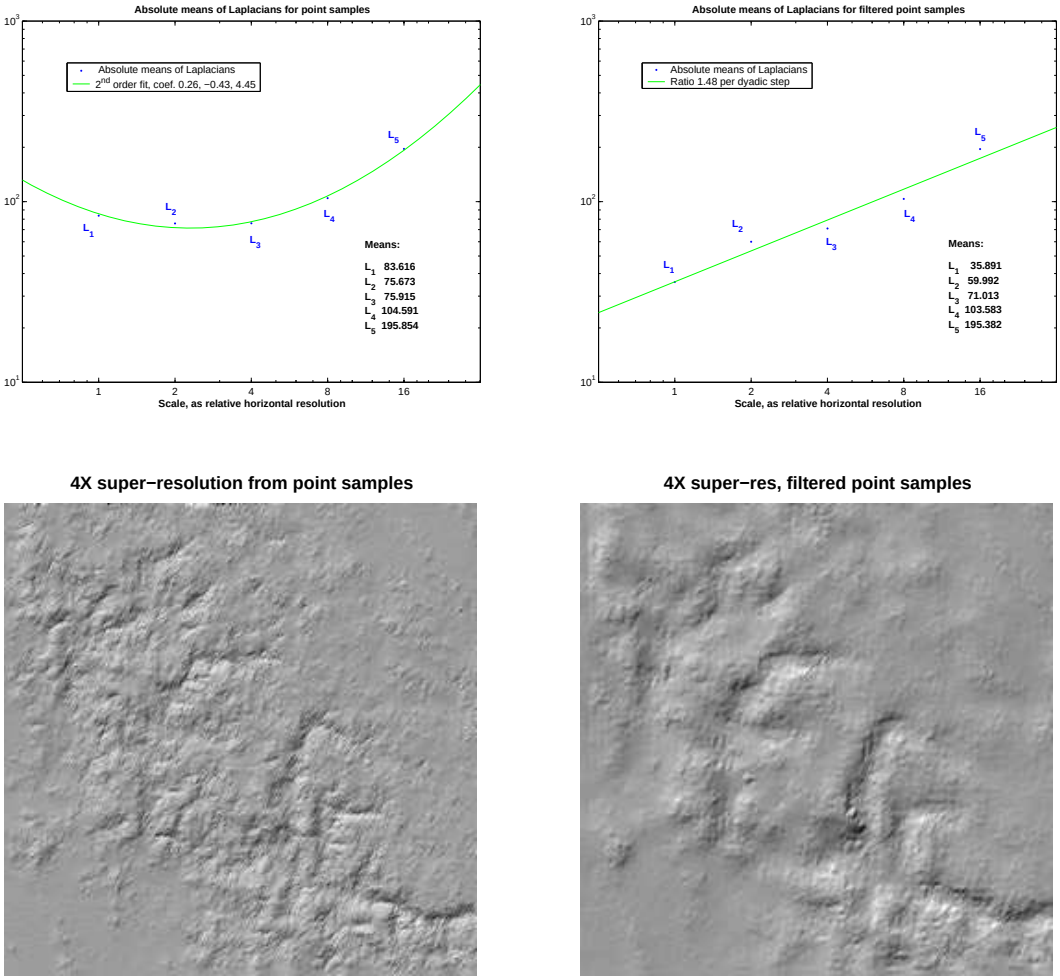


Figure 3.7: Comparison of super-resolution from point samples and super-resolution from filtered point samples.

points. The Laplacian pyramid so generated seemed to be more predictable (see Figure 3.7), but the resulting interpolation was very poor, as seen in the lower half of Figure 3.7. In fact, the interpolation obtained from unfiltered point samples, with *ad hoc* correction applied to the extrapolated Laplacian images, appears superior!

3.2.4 Possible improvements in search algorithm

Since we seek a match to *shape*, or *form*, rather than to absolute magnitude, a mean-removal scheme, applied to the 9-pixel blocks, might be considered, in hopes of finding a better match. But the Laplacian images are already *difference* images, and so they tend to be zero-bias.[6] For this reason, mean removal is ignored.

However, the size of the set of 9-pixel blocks to be searched for best match can be increased from 3844 to 30,752, using the same $\mathbf{L_2}$ image of size 64^2, by taking the eightfold path: There are eight ways of looking at both a 9-pixel block and a 4-pixel block —four rotations and their reflections. This theoretical augmentation of the similarity search library presumes a certain isotropy, rather than directional segregation, in any self similarity, which seemed to me a reasonable assumption. Yet the results for two-stage (16-fold, in terms of pixel count) super-resolution displayed in Figure 3.8 show that results appeared better using the one-way search!

3.2.5 Computational complexity

The method of this experiment was not to build a Laplacian interpolation look-up table, as suggested in the Pentland patent, but rather to conduct a "brute-force" search, writing or overwriting a 4-pixel block whenever a better match was found. The second stage of interpolation (from 256^2 to 512^2) was not "trained" by the first stage of interpolation. Therefore the method proved to be computationally intensive.

We regard the nine-pixel blocks and four-pixel blocks as points in function spaces; the algorithmic form that training might take would be to associate, with some

[6]It is interesting to note, however, asymmetry apparent in the histograms of Figure 3.4.

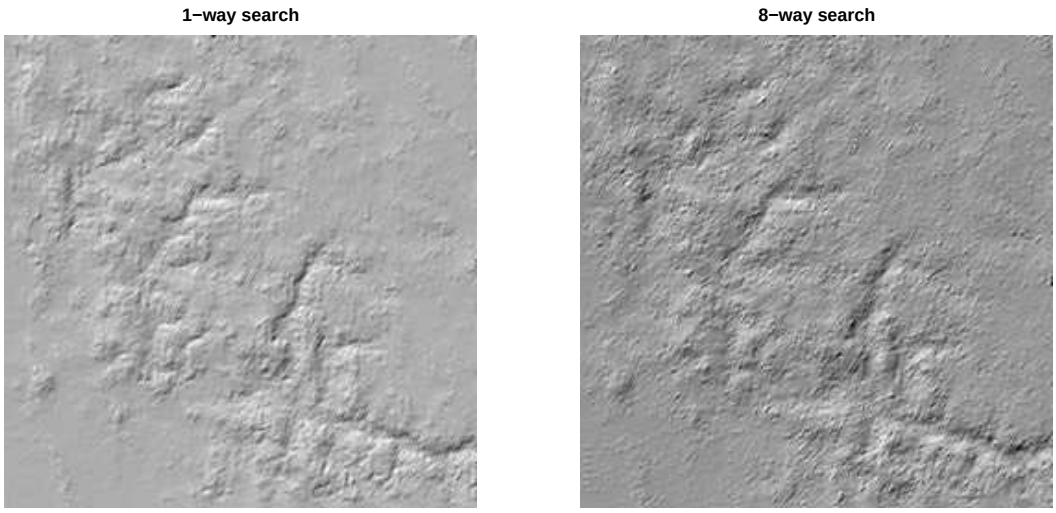


Figure 3.8: Southwest quarter of 16-fold super-resolution, using one-way search (left) and eight-way search (right).

function-space neighborhoods of nine-pixel blocks already encountered, the four-pixel blocks previously determined for those nine-pixel blocks. If a new nine-pixel block is found to lie in one of the previously-defined neighborhoods, assignment of its four-pixel block could be made on that basis, precluding further search. But I suspect that such a scheme would do little to relieve the computational complexity of the procedure, especially as applied to a single data set.

Computational complexity has been a problem with fractal image compression (*e.g.* [5]), and so it seems to be with fractal image interpolation. Even though the *algorithmic* information content of an image may be quite small, in that it has a small recipe, reconstruction may not be computationally trivial; in fact it may be impractical.

The first-stage computation time for the algorithms presented here increases approximately as the fourth power of the linear size of the grid to be interpolated. First-stage interpolation from a 256^2 image to a 512^2 image takes about 16 times as

long as first-stage interpolation from a 128^2 image to a 256^2 image. Second-stage interpolation, since it refers back to the same pair of known Laplacian images as first-stage interpolation, takes only four times as long. Taking the eightfold path predictably increases computation time eightfold.

The algorithms were first written as MATLAB m-files, but the predicted computation time for first-stage interpolation from 256^2 to 512^2 was 2 days (300MHz/256MB laptop PC). Equivalent programs were then written in C and compiled as MATLAB mex-files, which ran about 180 times faster, so that the same 256^2 to 512^2 interpolation was executed in less than 13 minutes. Some computation times are summarized below:

Interpreted Matlab m-file execution time

	16^2 *to* 32^2	32^2 *to* 64^2	64^2 *to* 128^2
One-way	$2.293s$	$25.656s$	$448.62s$
Eight-way	$5.849s$	$110.829s$	–
2nd It. One-way	–	$6.960s$	102.408

Compiled C (Matlab mex-file) execution time

	16^2 *to* 32^2	32^2 *to* 64^2	64^2 *to* 128^2
One-way	–	$0.151s$	$2.393s$
Eight-way	$0.040s$	$1.122s$	$20.730s$
2nd It. One-way	–	$0.020s$	$0.581s$

3.3 Wavelet-basis Interpolation

The wavelet transform and associated techniques of wavelet analysis are important to the question of interpolation because the essence of wavelet analysis is scale-wise hierarchical decomposition of signals into components of *approximation* and *detail*. Regarding interpolation, it is often the case that we have the approximation and we want the detail. Moreover, it is convenient to be able to move up or down in a

hierarchy of scale, depending on sample density or desired resolution of presentation. In this section we further explore how we might use the idea mentioned in Part 1, that if a signal has *expected detail* beyond the sampled resolution, which depends somehow on the known detail at coarser resolution, then a scale-wise analysis might allow us to *interpolate*, in effect, by *extrapolating* coefficients to the next finer scale.

Wavelet analysis has the further advantage that it operates *locally*; the meaning of the phrase "compactly supported wavelets" [15] is that the convolution filter used in the transform is of finite extent[7]. This means that the transformed components are not distributed sums of *all* the old components, but rather the spatial extent of influence is itself varied, as part of the analysis "cascade". One consequence of this is that, differing from the case of, say, Fourier transform coefficient representation, a representation of the wavelet transform coefficients actually resembles the representation in the standard base, as shown in Figure 3.9.

The original topography cross-section signal is shown on the top. Analysis by the "db2" wavelet transforms this signal, by a filtering (*i.e.* convolution) process, into the two sets of coefficients shown below it.[8] The approximation coefficients actually resemble the signal in its standard representation, but number only half the signal length (note the change in x-scale). The detail coefficients also number half as many, but they don't resemble the *structure*, or form, of the signal; rather to some extent they resemble the *texture*.

The transform can be inverted separately for the two sets of wavelet transform coefficients, yielding the reconstructions shown in the lower part of Figure 3.9. Here we are looking at things in the usual way, in the standard base. Adding the inverse discrete wavelet tranform (IDWT) of the approximation coefficients to the IDWT of the detail coefficients results in the reconstruction shown third from the bottom,

[7]For the "db2" wavelet used here, the filter is just four points.

[8]The filters used for the purpose are called "quadrature mirror filters", and they have the remarkable property that they separate the signal into entirely orthogonal and equal parts.

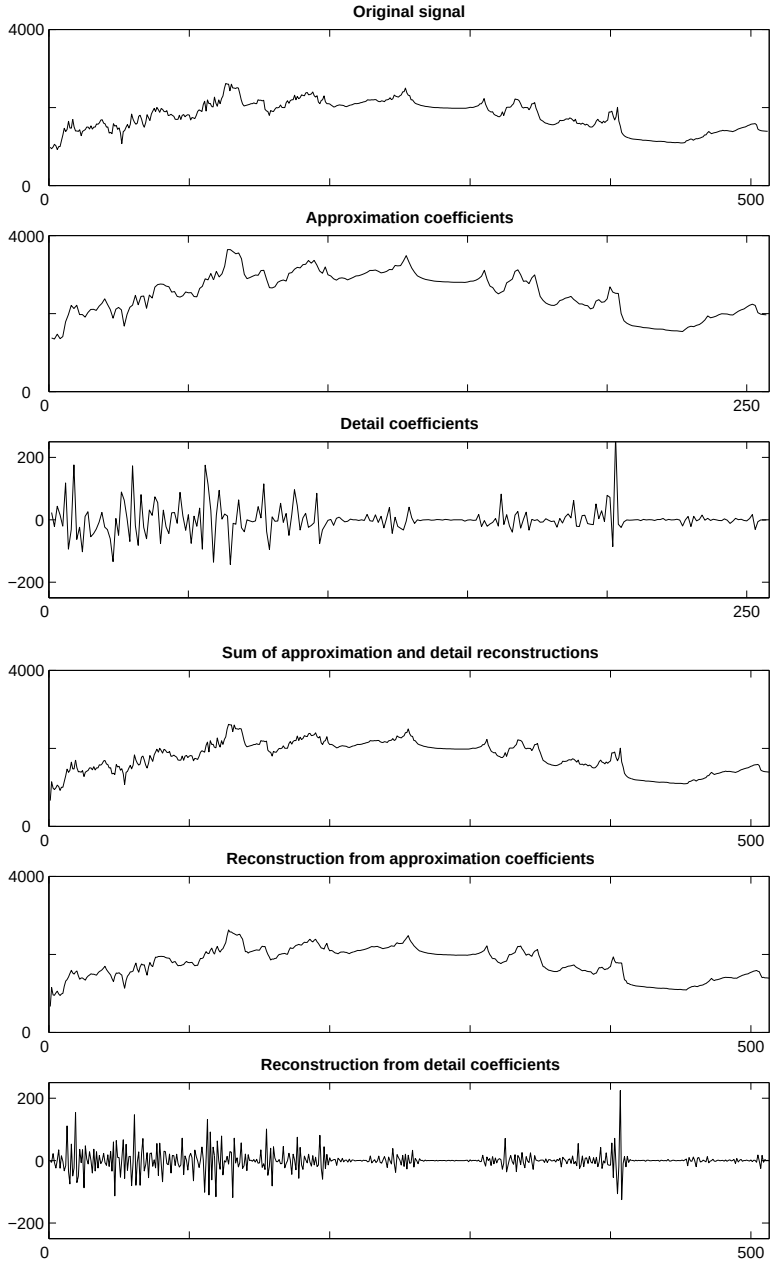


Figure 3.9: Signal, wavelet transform representations, and reconstructions.

which is the same as the original signal top).

3.3.1 The method of Ford and Etter

Ford and Etter [24] proposed a "multi-resolution basis fitting reconstruction" (MBFR) method for reconstruction of nonuniformly sampled data. Their method is not to apply iterative methods to the first approximation error in order to find the best projection of sampled data onto a subspace (as described in Section 3.4); rather they propose to take the first approximation error at one resolution as input for an approximation at the next hierarchical level of resolution.

The wavelet transform is taken as a system of simultaneous equations at each level of resolution —one set for the wavelet "approximation" coefficients, and one set for the wavelet "detail" coefficients at that level of resolution. It is presumed that at some sufficiently coarse level of resolution, the system should be solvable since the number of unknowns (*i.e.* the number of wavelet coefficients) approximately halves at each hierarchical step, while the number of equations in the system is the number of signal points known.[9] If a "least squares" solution can be found for the system of equations at some level of resolution for the approximation coeffients, an approximation may be obtained to the sample data by means of inverting the wavelet transform for those approximation coefficients. The error at the sample points can then be considered to pertain to a signal component representable by the "detail" wavelet coefficients at that level of resolution, and a solution can be sought for the system of equations associated with those coefficients.

Ford & Etter would proceed to finer levels of resolution locally, wherever the sampling density allowed an "overdetermined" system of equations. Since the number of equations in the system decreases with the length of the signal section considered,

[9]The number of wavelet coefficients at a given resolution level J –higher J means coarser resolution– is approximately 2^{-J} times the number of signal points, both for the "approximation" and the "detail" coefficients.

the degree of resolution possible locally would seem to depend only on sampling density.

Test method

To prove the method, I began with a section of the DEM used for Section 3.2, specifically row 256, containing 512 data points. Two random irregular sample sets were created, one set containing 259 sample points, beginning with original data point 2 and ending with 511, with the largest gap being 6 skipped points of the original data; the second set containing 109 sample points beginning with original data point 3 and ending with 510, with the largest gap being 8 skipped points of the original data.

The level 1 wavelet transform (DWT) for the complete data set of 512 points, using the "db2" wavelet, can be represented by a pair of matrices $\mathbf{W}_{a1}$ and $\mathbf{W}_{d1}$ of dimensions 257 (rows) x 512 (columns) which, when multiplied by the column vector $\mathbf{s}$ of length 512 (the complete signal), yield the column vectors **a1** and **d1** of lengths 257, which comprise the wavelet coefficients:

$$\mathbf{W}_{a1}\mathbf{s} = \mathbf{a1} \ ;$$

$$\mathbf{W}_{d1}\mathbf{s} = \mathbf{d1} \ .$$

Likewise the level 1 inverse discrete wavelet transform (IDWT) can be represented by a pair of matrices $\mathbf{I}_{a1}$ and $\mathbf{I}_{d1}$ of dimensions 512 (rows) by 257 (columns) which, when multiplied respectively by the column vectors **a1** and **d1**, yield the approximation $\mathbf{s}_{a1}$ and detail $\mathbf{s}_{d1}$ parts of the original signal $\mathbf{s}$, which can be added to recover the original signal:

$$\mathbf{I}_{a1}\mathbf{a1} = \mathbf{s}_{a1}$$

$$\mathbf{I}_{d1}\mathbf{d1} = \mathbf{s}_{d1}$$

$$\mathbf{s}_{a1} + \mathbf{s}_{d1} = \mathbf{s} \ .$$

The system of equations to solve for approximation coefficients at level 1, for the 259 sample points, can be represented by the matrix equation

$$\mathbf{I}_{a1,p}\mathbf{a1} = \mathbf{p} \ ,$$

where $\mathbf{I}_{a1,p}$ is a 259 (row) x 257 (column) matrix whose 259 rows are those rows of $\mathbf{I}_{a1}$ that correspond to the 259 sample points, **a1** is the set of 257 unknown approximation coefficients, and **p** is the set of 259 samples. The least-squares solution obtained for this system, called $\hat{\mathbf{a}}$, will in general not be exact; there will be an error at each of the sample points, which we'll call $\mathbf{e}_{a1,p}$:

$$\mathbf{e}_{a1,p} = \mathbf{p} - \hat{\mathbf{a}}_{\mathbf{p}} \ .$$

The system to solve for the detail coefficients, then, is:

$$\mathbf{I}_{d1,p}\mathbf{d1} = \mathbf{e}_{a1,p} \ .$$

Similar matrices $\mathbf{I}_{a2}$, $\mathbf{I}_{d2}$, $\mathbf{I}_{a3}$, and $\mathbf{I}_{d3}$ were constructed for levels 2 and 3.

Discussion and conclusions concerning this method

Even though the system of equations at level 1 appears to be "overdetermined" since there are 259 equations against 257 unknowns, it turns out that the *rank* of the matrix of coefficients $\mathbf{I}_{a1,p}$ is only 227. The meaning of rank is the number of dimensions in function space that can be represented by the equations. The original matrix $\mathbf{I}_{a1}$ is built of 512 row vectors which are the new *approximation* wavelet base representations of the 512 orthogonal "directions" of the original standard representation of function space.[10] But these representations are actually *projections*, complementary to the projections of the original base directions onto **d1** *detail* space.[11] Consequently

[10] They are analogous to the helical Fourier representations of the unit impulse functions that comprise the standard base, as depicted previously.

[11] *Projection* means a change in the dimensionality; here, specifically, a reduction.

the approximation base can only represent 257 dimensions as a full matrix, before removing rows where there are no samples!

Therefore, the solution for the full sample set had to be calculated at level 2 (corresponding to half resolution, not counting interpolation error). Results are shown in Figure 3.10.

The full data set is shown on the top; the next plot shows the sample points (green) and the reconstruction obtained at level 2 by solving for approximation coefficients only. The third plot shows the reconstruction (blue) obtained by adding the detail solution of the error equation, again with the sample points (green) and original data (black). At the bottom is a magnification of the same plot, in the area either side of profile point 100, which appeared problematic. Rather large excursions in the interpolation are apparent, about point 80 (negative) and point 110 (positive). On the other hand, the "real" data shows a similar departure at about 126, and the interpolation does not in general appear out of character. It is apparent that the interpolation does not pass through all the sample points, but the objective might not be statistical accuracy.

An apparent problem with this method for interpolation is the "end effect" on the interpolation. I did not confine the interpolation to the segment between first and last sample points, but even when this is done, I found instability near the ends. Since short data segments are more "end" than "middle", this is a likely shortcoming of the method.

One characteristic of wavelet-based interpolation subtly apparent here, but more apparent in the next example is the imposition of the wavelet shape to the interpolation. Figure 3.11 shows the results of the method applied to the smaller sample set, of 109 points.

In the approximation-only reconstruction, shown in the second plot, the asymmetrical wavelet shape seems to make the peaks look like the teeth of a pruning saw,

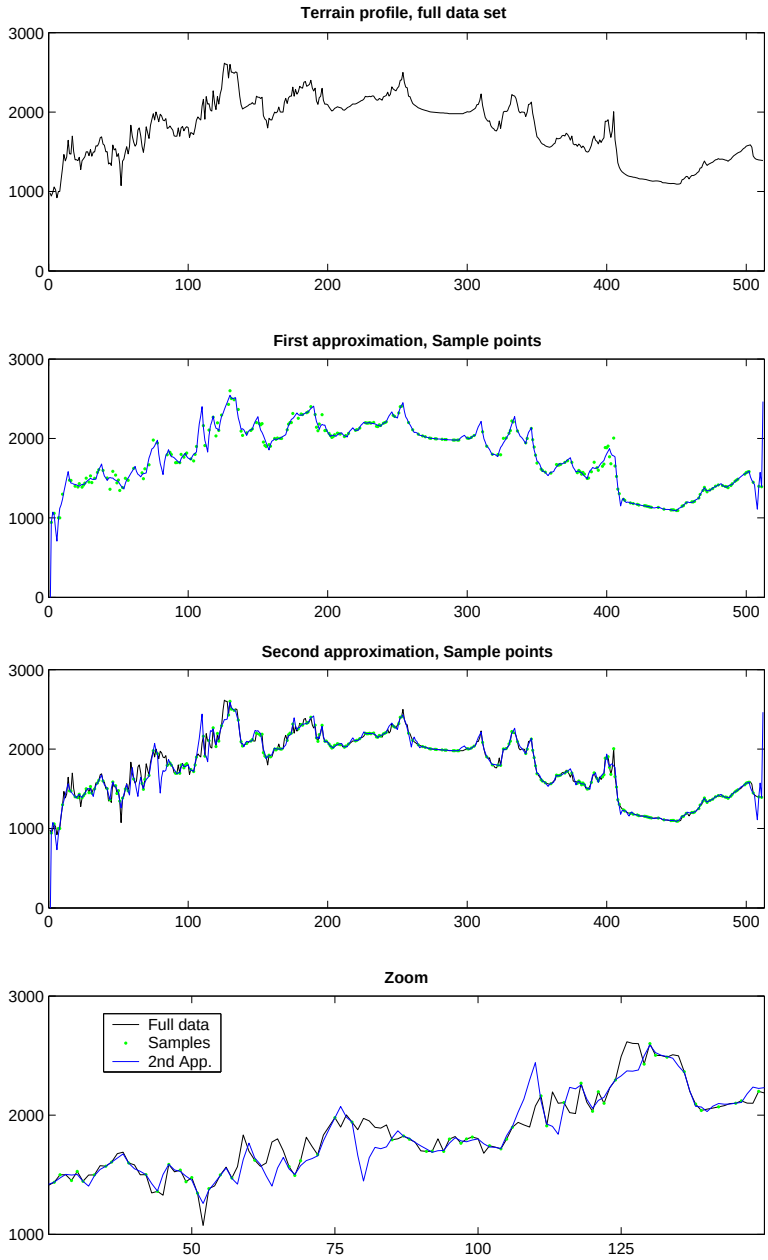


Figure 3.10: Level-2 Daubechies-2 wavelet-base reconstruction of 512-point terrain profile from 259 sample points.

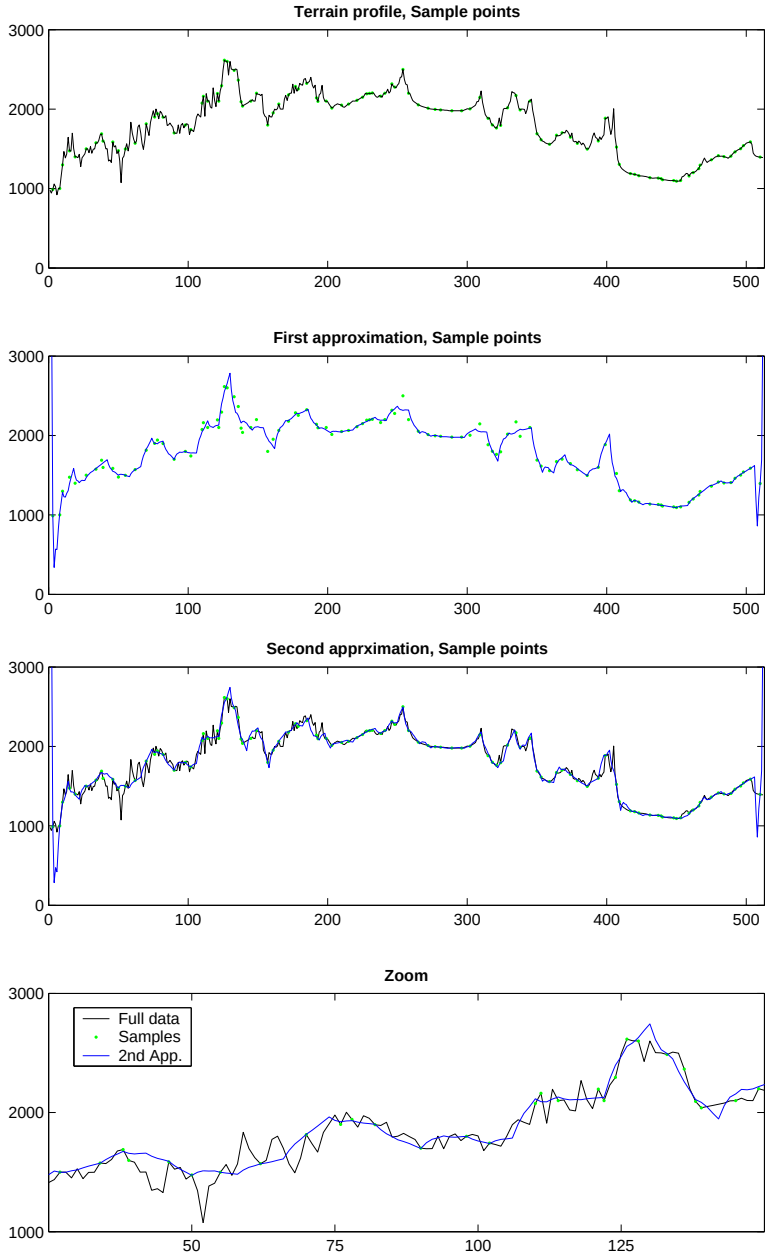


Figure 3.11: Level-3 reconstruction from 109 sample points, and lower-level reconstruction of segment.

or as waves breaking eastward.

It was necessary in this case to go to level 3 to obtain a solution to the equations. This no longer conveys the finer texture of the data, even as seen in the third and fourt plots of Figure 3.11 , where the reconstruction of solution for detail coefficients is added to the interpolation. The norm error for the sample point set was reduced about 53 percent by adding the level 3 detail solution to the level 3 approximation solution; the norm error against the whole data set actually increased slightly by this step. The same region from data point 25 to data point 150 is shown magnified in the bottom plot.

To test the potential for "zooming in" on segments of higher data density, I took the section from (original) data point 110 to data point 124, containing 6 sample points, for solution at level 2, and at level 1. The results are shown in Figure 3.12. The reconstructed solution at level 2 is shown on the top, and the reconstructed solution at level 1 is shown at the bottom. This method does not produce satisfactory interpolation, using the "db2" wavelet base. Even though an essentially "exact" solution was possible at level 1, the reconstructed points between sample points are beyond reasonable bounds (note the change in vertical scale of the lower plot). Evidently wavelet-based solution makes little sense when the size of the wavelet is comparable to the size of the data set, and "end effect" predominates. After all, convolution with the 4-pole "db2"wavelet filter about any given point uses a neighborhood of seven points —three points either side of the given point. The end of data can be handled in various ways; the method used here was end reflection.

Further investigation was done using other wavelet bases, including biorthogonal (recommended by Ford & Etter), but solutions were even less satisfactory than for the db2 wavelet.

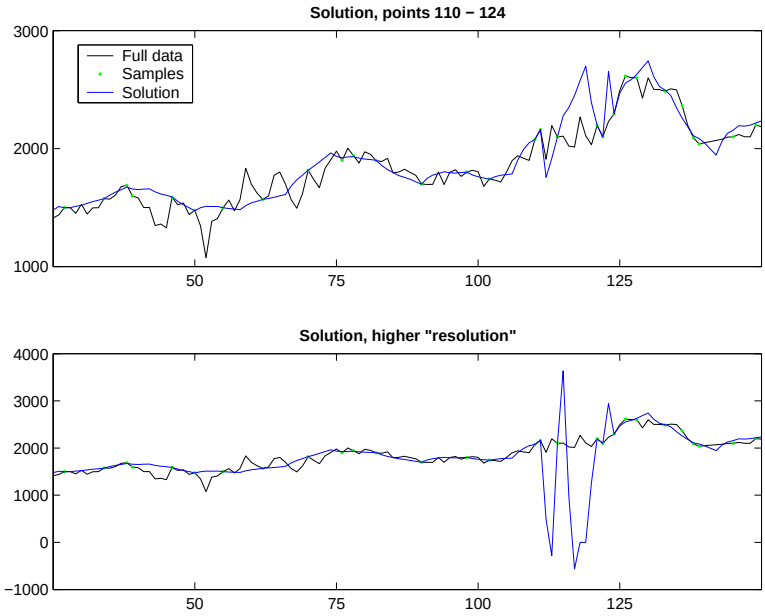


Figure 3.12: Results of segment interpolation, level 2 (top) and level 1 (bottom).

3.3.2 Wavelet basis super-resolution

The multi-resolution cascade of Wavelet analysis is closely related to the pyramidal decomposition of Section 3.2. Both decompositions are based on filtering and subsampling, retaining the higher-frequency components, and iterating on the lower frequency components, in hierarchical steps toward the final coarse-scale representation. The wavelet details correspond to the Laplacian images, while the wavelet approximations correspond to the Gaussian images. Two approaches to super-resolution are explored here: one is a search method operating on wavelet detail coefficients, in analogy to the method of Section 3.2, and the other is a cruder, but perhaps just as effective, method based on *not* subsampling the decomposition convolution.

Both methods can readily be extended to two dimensions, for direct comparison with the method of Section 3.2. As before, computation time may become an issue for the search method in two dimensions.

Search method

For both experiments I start with a regularly subsampled set from the same 512-point profile discussed previously, taking data points $2, 6, 10, \ldots 510$: 128 points in all. The object will be to double the resolution twice, for comparison with the original data.

Again the "db2" wavelet base is used, for the sake of consistency, though other bases may be found to be more suitable. A wavelet decomposition is carried out to level 2, obtaining wavelet detail coefficients **d1** and **d2**. For each element (65 in number) of **d1**, a neighborhood of five elements is compared to the five-element neighborhoods of each of the 34 elements of **d2**. End reflection is used to allow comparison of neighborhoods for the end two elements. After all have been checked for the closest match, two elements of the synthetic **d0** are written the same as the two elements of **d1** that are "above" the element of **d2** that was the center of the best-neighborhood match. The computation is implemented in my MATLAB m-file EXTENDWAVDET. Some results are shown in Figure 3.13.

In the top two plots are shown the input detail coefficients $\mathbf{D}_2$ and $\mathbf{D}_1$ from wavelet decomposition levels 2 and 1. Below are shown the extrapolated detail coefficients $\mathbf{D}_0$ and $\mathbf{D}_{-1}$ found by the search functions EXTENDWAVDET and EXWAVDET3 (the latter used for generating $\mathbf{D}_{-1}$ from $\mathbf{D}_0$ based on the same relation between $\mathbf{D}_1$ and $\mathbf{D}_2$ used to generate $\mathbf{D}_0$). In the lower four plots are shown the interpolations (blue), against the sample points (green) and the original data (black). The interpolation in the fourth plot from bottom, which is a doubling (called $\mathbf{R}_0$), is derived from the first step additional detail represented by $\mathbf{D}_0$; the bottom three plots show the fourfold interpolation derived from the additional detail represented by both $\mathbf{D}_0$ and $\mathbf{D}_{-1}$. The second plot from bottom is a magnification (in x-direction) of the section between data points 25 and 150 (which section was also shown magnified in Figure 3.11), and the last plot is a magnification of the section between data points 250 and 375.

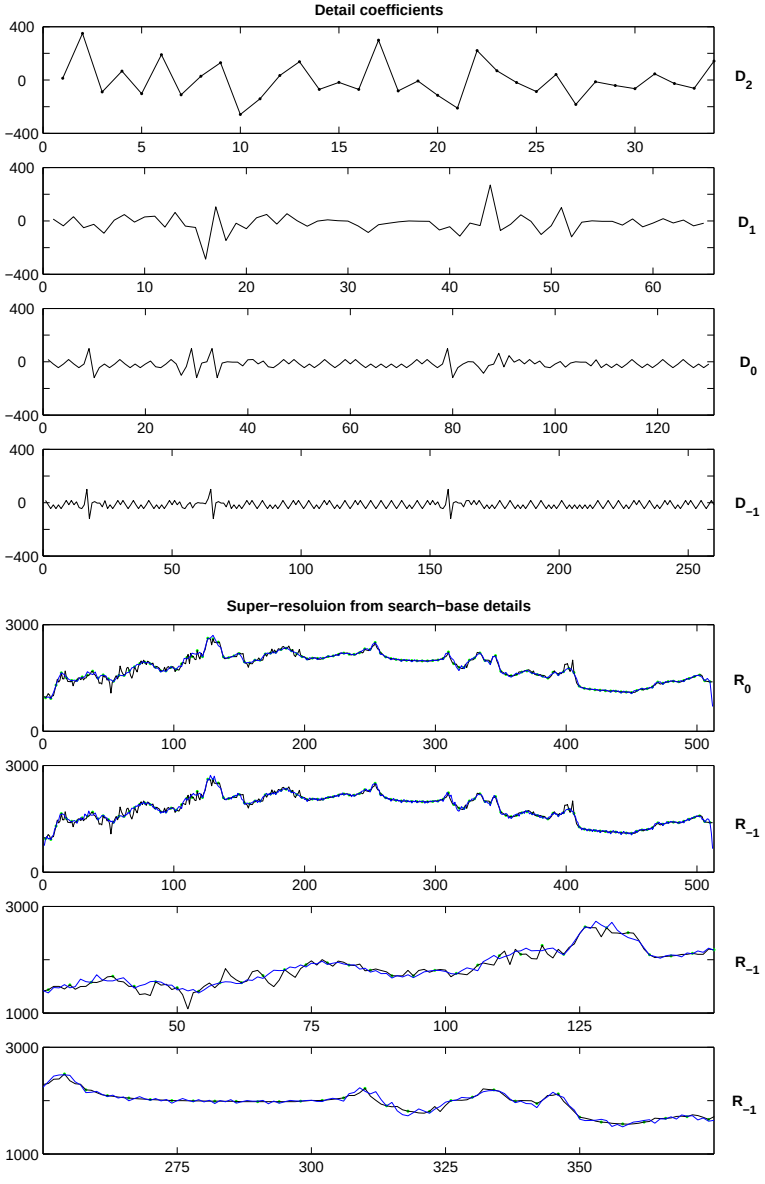


Figure 3.13: Super-resolution by 2 and by 4, based on presumed self-similarity of wavelet detail coefficients.

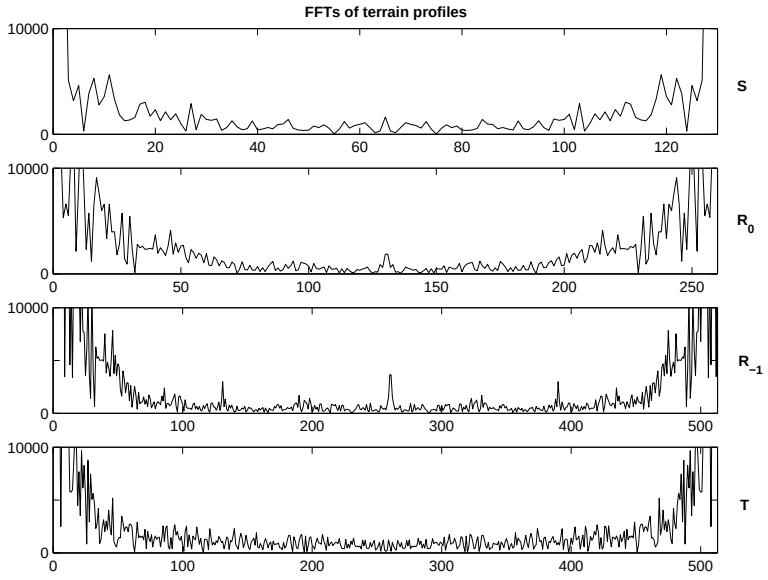


Figure 3.14: FFTs of 128-point data $\mathbf{S}$, 256-point interpolation $\mathbf{R}_0$, 512-point interpolation $\mathbf{R}_{-1}$,and 512-point reference data $\mathbf{T}$.

It is apparent that the method may have applied a texture that is too uniform: broad valleys (as between data points 275 and 300, bottom right plot) appear too rough, while mountainsides (as between data points 50 and 75, third plot on right) appear too smooth. Indeed, reference to the synthetic detail coefficients shown in the top half of Figure 3.13 suggests that three particular areas of discontinuity have been emphasized, at the same time that other detail has tended to become homogeneous or even periodic. It may be possible to obtain more "realistic" results by filtering or dithering the detail coefficients after they are generated by the algorithm. A weighted comparison algorithm which favors diminished coefficients might produce more stable results.

Fourier transform representation of the data can afford insight into the problem as well. Figure 3.14 is a non-standard way to display Fourier magnitude spectrum, but it draws attention to patterns in the distribution of high frequency components.

The magnitude spectrum of the starting sample of 128 data points is shown in the top plot, the magnitude spectra of the interpolated doubling and quadrupling are shown in the second and third plots, and the magnitude spectrum of the "target" original data is shown in the bottom plot.

It is apparent that the method has produced an artificial peak around the Nyquist frequency (middle of the plots), and other "harmonics" are apparent in the $\mathbf{R}_{-1}$ interpolation, which scarcely exist in the target data. Also noteworthy is the fact that the spectrum of the target data falls off more rapidly from the low end, even though the amplitude toward the high end, before approaching the Nyquist frequency, is similar to that of the interpolation. This suggests that an improvement in the method might be possible by filtering the result to obtain a magnitude spectral shape in better conformity to that taken from know data in context.

Double convolution upsample method

The wavelet transform, regarded as the way to get from standard base representation of a signal to representation of that signal in terms of approximation and detail coefficients, is performed algorithmically by *convolution* (filtering) followed by *subsampling* (removing every other point). The inverse is performed by *upsampling* (adding a zero between every pair of points) followed by *reverse convolution*. The essence of this proposed method of super-resolution is simply to omit the subsampling.

To effect this interpolation, approximation coefficients and detail coefficients are obtained from the previously-used 128-point subsample of a 512-point terrain profile, by the usual convolutions with the decomposition filters, but without subsequent subsampling. These sets of coefficients are then upsampled and convolved with the respective reconstruction filters and added together, which is the same as to say that they are taken as the inputs for an IDWT transform back to the standard base, now with double "resolution". We iterate this procedure twice to obtain fourfold

super-resolution, as shown in Figure 3.15.

The top plot shows the reference data (black) and sample points (green) with a spline interpolation (blue) through the sample points, for data points 25 to 150. The second plot shows the results of the first iteration, interpolating from 128 points to 256. The plots on the bottom show the result of the second iteration, to 512 points. The third plot is the entire profile, and the bottom plot shows data points 25 to 150.

Here the synthetic texture appears to conform better to the local context than it did for the previous (search) method. However, there is more departure from the sample points: the norm of the error at the sample points for this method turns out to be almost 31 percent greater than for the search method.

The filters of the orthogonal wavelet bases, including the "db2" wavelet used here, are not symmetrical. But the filters used before subsampling are simply the reverse (left-to-right) of the filters used after upsampling. For this reason, asymmetrical artifacts of texture may be suppressed. However, the convolution-upsampling-convolution operations are not commutative; the effect is not the same as upsampling-convolution-convolution. But it would be possible to alternate the order in which the asymmetrical filters are applied.

Even though this method seemed to distribute details more appropriately according to spatial context on the profile, than the search method, defects of this method are revealed by reference to the Fourier magnitude spectrum, shown in Figure 3.16.

The artificial peak about the Nyquist frequency, seen in the magnitude spectra of the results of the search method, is absent, but an artificial enhancement of a subband has been produced, between 150 and 200.[12] Again, post-filtering may be an option, but ruins the simplicity of the method.

[12]The index numbers of the FFT can be thought of as standing for the number of cycles over the whole data interval.

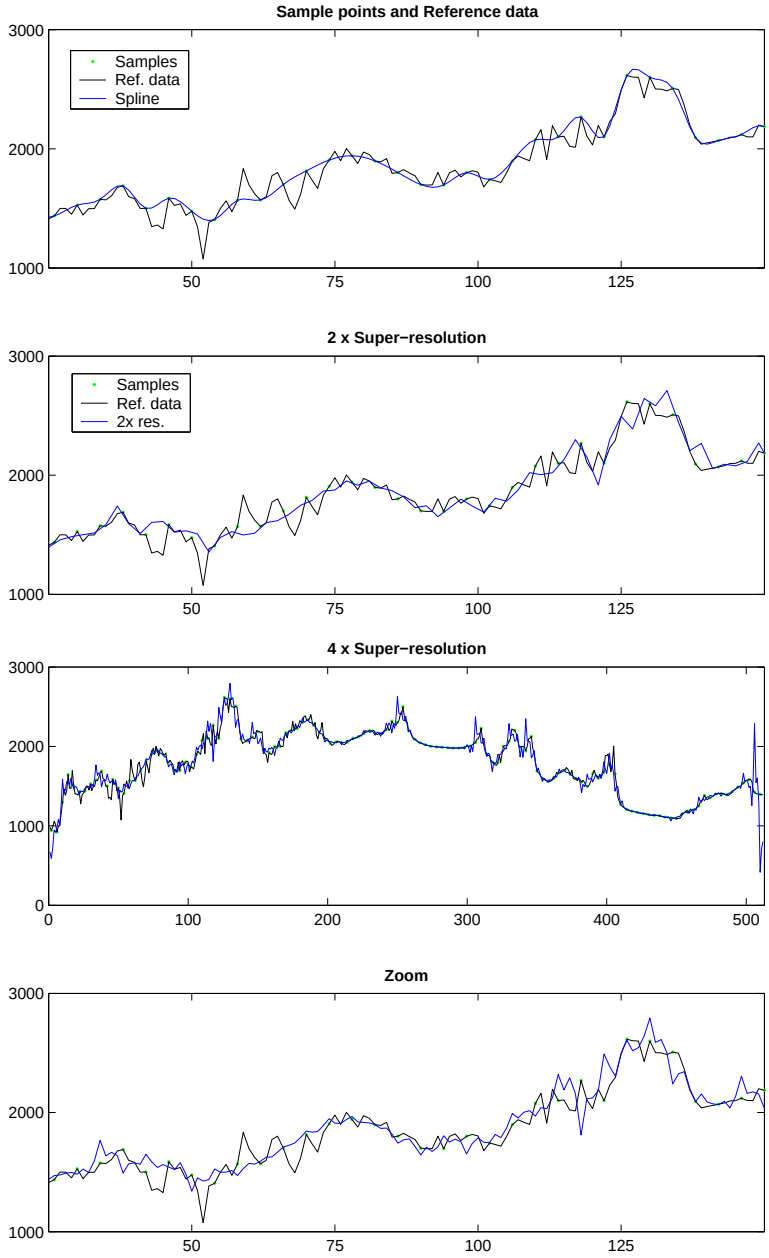


Figure 3.15: Super-resolution by 2 and by 4, based on convolution and upsampling using "db2" filters.

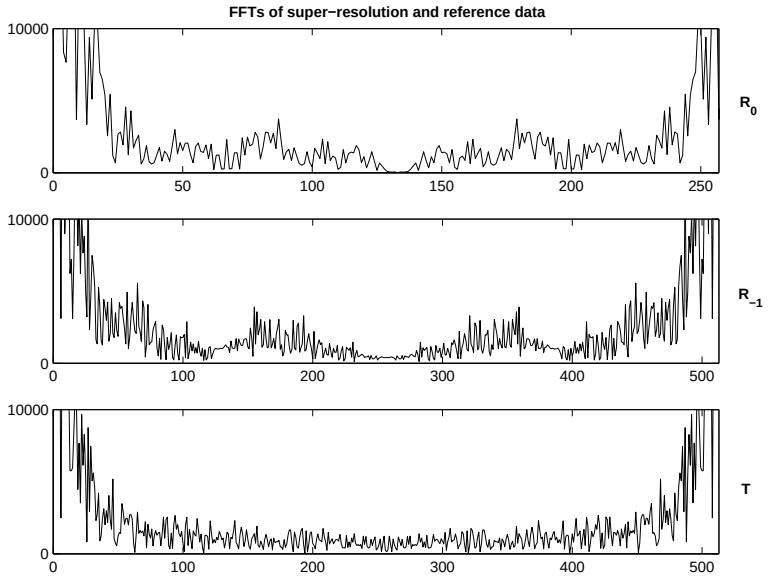


Figure 3.16: FFTs of convolution super-resolution, 256-point interpolation $\mathbf{R}_0$, 512-point interpolation $\mathbf{R}_{-1}$, and 512-point reference data $\mathbf{T}$.

3.4 Iterative Fourier basis interpolation for irregularly-spaced samples

Computationally efficient methods for reconstruction of band-limited signals from irregular samples have been devised, for both the 1-dimensional case and the 2-dimensional case. [49], [20], [21] Speaking in terms of function space, these algorithms effectively project an initial unrefined interpolation of the signal, which has out-of-band frequency content, onto the band-limited subspace in which the signal is supposed to reside. This results in an *error* signal (irregularly sampled) at the sample points, to which the same interpolation and projection can be applied and added to the first approximation, iterating until the error at the sample points is very small.

Figure 3.17 illustrates the method's application to the reconstruction of a randomly-generated bandlimited signal. Unlike the terrain signals that we are using for examples, this signal does *not* have 1/f spectral characteristic; higher-frequency components may have amplitude just as great as lower-frequency components, up to the band

limit, which is 32 cycles in the 512-point interval. The "adaptive weights, conjugate gradient" method of Feichtinger *et. al.* [20] is used to reconstruct (interpolate) this signal from irregular samples, using a "boxcar" frequency filter (so called because its appearance is as a rectangle in the Fourier representation) of width 65.[13] According to the Sampling Theorem [65], only two samples per cycle are required for perfect reconstruction of a bandlimited signal, and these need not be regularly-spaced samples. [54]

On the top is shown a reconstruction from 109 samples. The second plot actually shows the interpolation (green) over the original signal (black), but the interpolation is so near perfect that the original signal is not seen. The bottom two plots illustrate a reconstruction from 60 samples. This is less than the minimum two samples per cycle (64), and the imperfection of the reconstruction can be seen in the bottom plot.

The reconstruction obtained by this method is fast and remarkably good, if the bandwidth of the signal is known or is well-estimated from context. However, this is a critical issue, because if a too-wide bandwidth is assumed, the interpolation can deviate unnecessarily, even as it passes through the sample points (as in the case of a high-order polynomial interpolation); if a too-narrow bandwidth is assumed, the interpolation becomes an inexact smooth approximation which can never pass through all the sample points, regardless of the number of iterations performed in the algorithm. This is illustrated in Figure 3.18, showing the results of assuming a bandwidth half the actual (top) and twice the actual (second plot), with the Fourier magnitude spectra of the original signal and the two incorrect reconstructions shown below.

The "projection" to band-limited subspace mentioned above is another way to speak of *filtering*, in the frequency domain after a Fourier transform, or of *convolution*, in the original representation. Computational efficiency is achieved by working

[13]The width is 65, rather than 32, because it is a symmetrical filter including zero frequency and negative frequencies.

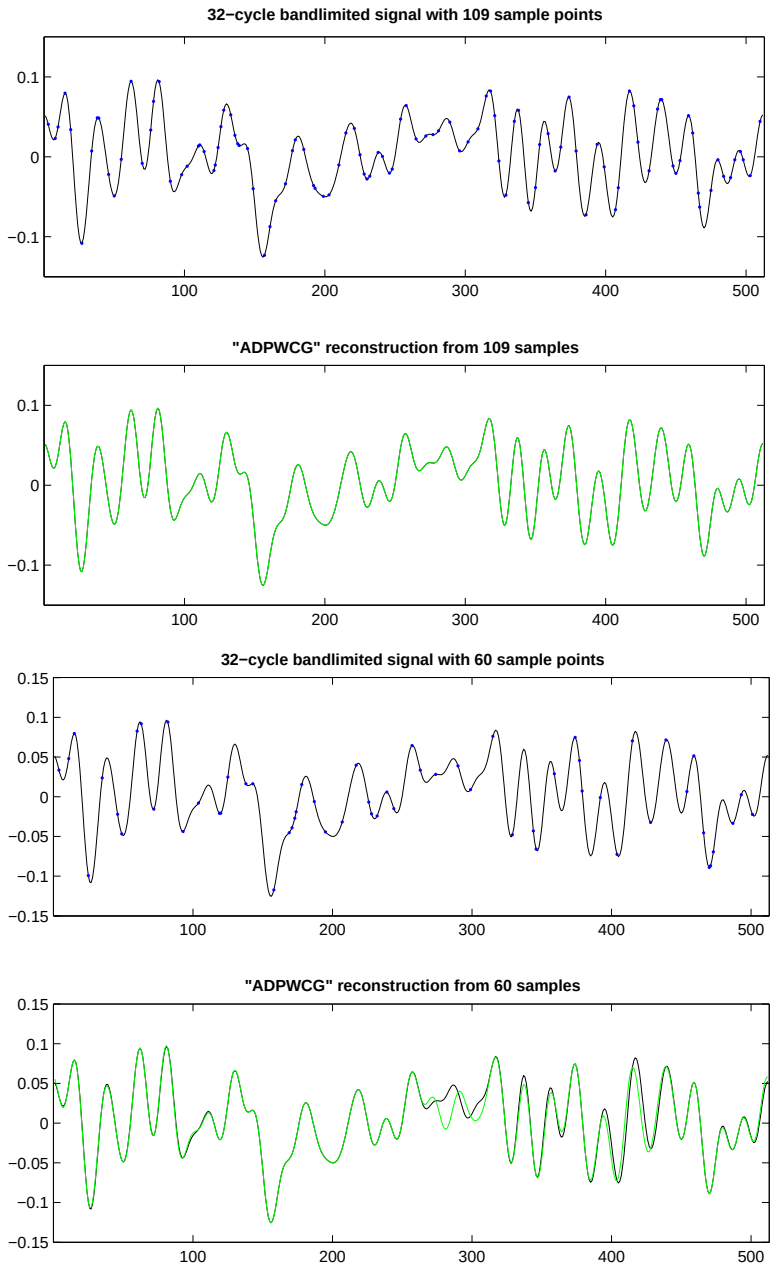


Figure 3.17: Reconstruction (green) of bandlimited signal (black) from irregular samples (blue) using iterative method.

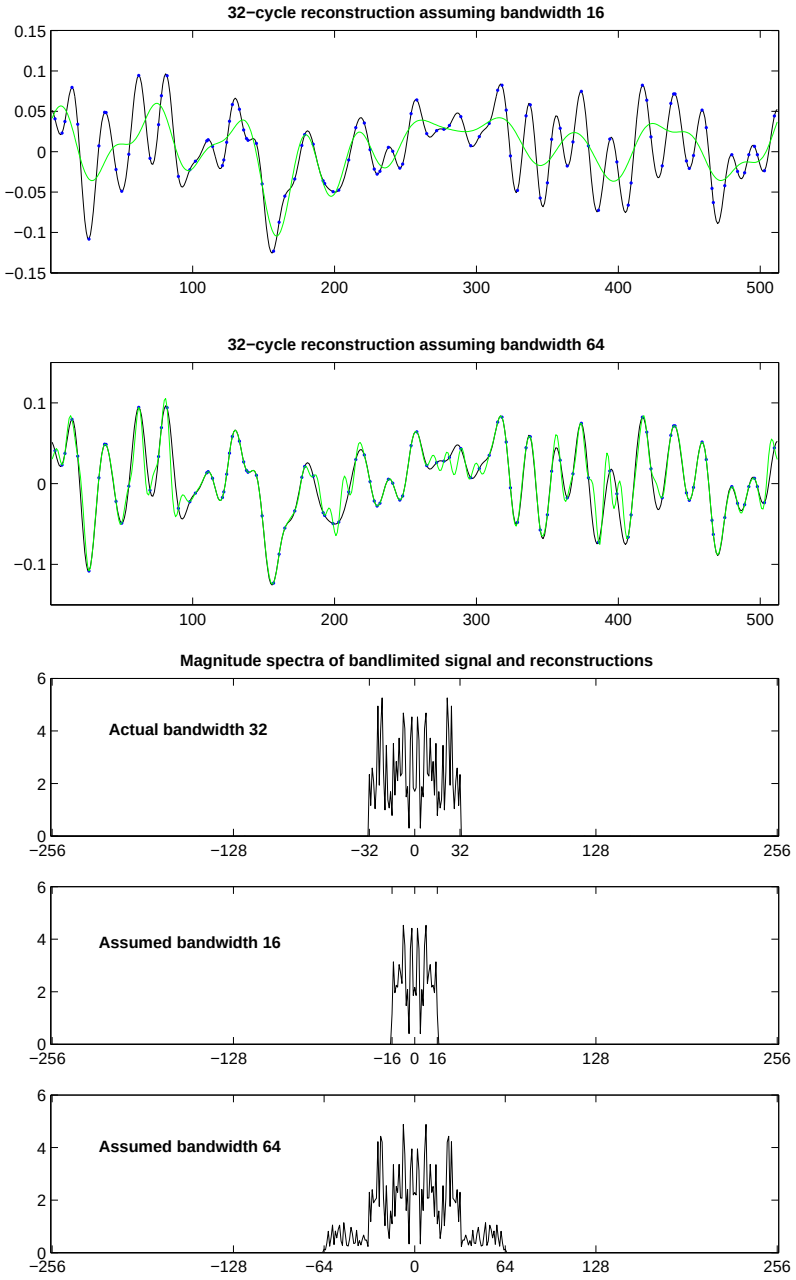


Figure 3.18: Reconstruction (green) of bandlimited signal (black) from irregular samples (blue) assuming too small (top) and too large (bottom) bandwidth. Fourier magnitude spectra shown below.

in the frequency domain. The irregular samples by themselves do not provide a complete set of coefficients for a Fourier transform, hence the iterative reconstruction algorithm starts with an initial unrefined interpolation, which can be nothing more than zeros inserted for the missing sample values (called the *Marvasti* method) [49], a step function that assumes the nearest sample value (called the *Voronoi* method), or the *adaptive weights* method [20], [21] which is to insert zeros between samples while weighting sample values according to their proximity to other sample points.

Improvements have been made in the computation algorithms, to accelerate convergence[14] [20] and to allow better reconstruction from limited local samples by appropriate choice of an optimum filter. [74]

3.4.1 Adaption to $1/f$ signals

In this section we explore a possible adaption of the reconstruction method for irregularly-sampled signals that are band limited, described above, to irregularly-sampled topography signals that are more likely to exhibit $1/f$ spectral characteristic than to be band limited. First we proceed in two stages, in an attempt to reconstruct first "structure", and then "texture" (after Clarke [11]). Finally we look at a one-step compromise that employs a filter that "gets the job done".

The reconstruction algorithms that we are using can be used with a filter chosen for the purpose; it does not have to be a boxcar filter. In fact, if reconstruction is attempted using such a filter, the resulting interpolation is either very inexact (assuming small bandwidth), or oscillates wildly (assuming large bandwidth). Instead, the filter can be made to the $1/f$ shape. Figure 3.19, top, shows that data in context (*viz.* a terrain section 6 km to the north) does indeed have spectral magnitude characteristic close to $1/f$ —the least squares log-log fit indicates $1/f^{0.925}$. But it does not do to use a $1/f$ filter the length of the signal (no band limit), even though the

[14] Re the *conjugate gradient* method and the *conjugate gradient Toeplitz* matrix methods for discrete signals, included in the "IRSATOL" MATLAB toolbox.

signal might be expected to have no particular limit of frequency components. Because the inverse Fourier transform of our $1/f$ filter (which is the convolution "atom") looks like a cuspid peak, the reconstruction using a full-length $1/f$ filter inherits this characteristic appearance of cobwebs hanging on the sample data points, as shown in Figure 3.19, bottom.

Instead, we use a $1/f$ filter that goes to zero beyond a certain width —it is a boxcar filter multiplied by a $1/f$ filter, which is illustrated in Figure 3.20.

Two-step approach, adding texture

Experimentally it was found in this application that if a simple truncated $1/f$ filter of the sort described above was used, the best width for this filter was about 3/4 the number of sample points (with sampling density roughly 1/4).

The result of the reconstruction at this stage is an interpolation which is almost "exact" in the sense of passing through the sample points. It shows some extra wiggles where they don't belong, as polynomial interpolation would, but they are attenuated by the $1/f$ characteristic of the filter that was used. The problem now is that the interpolation looks too sinusoidal.

The second stage of interpolation, or simulation of texture, is accomplished by adding high-frequency components in context: those components of the FFT of the first-stage interpolation which are zero, as a result of the band limit of the truncated $1/f$ filter used in the first stage, are replaced with the high-frequency components of a signal which is expected to be similar in terms of texture. The result of this texturing stage is less "exact" in the sense of departure from sample points, but if the context spectrum used for adding texture has amplitude that follows the $1/f$ relation, then the error is likely to remain very small. Figure 3.21 shows, above, Fourier magnitude spectra of the signal reconstruction based on the width-80 truncated $1/f$ filter (top), and of a terrain profile six km to the north (second plot); below is the textured

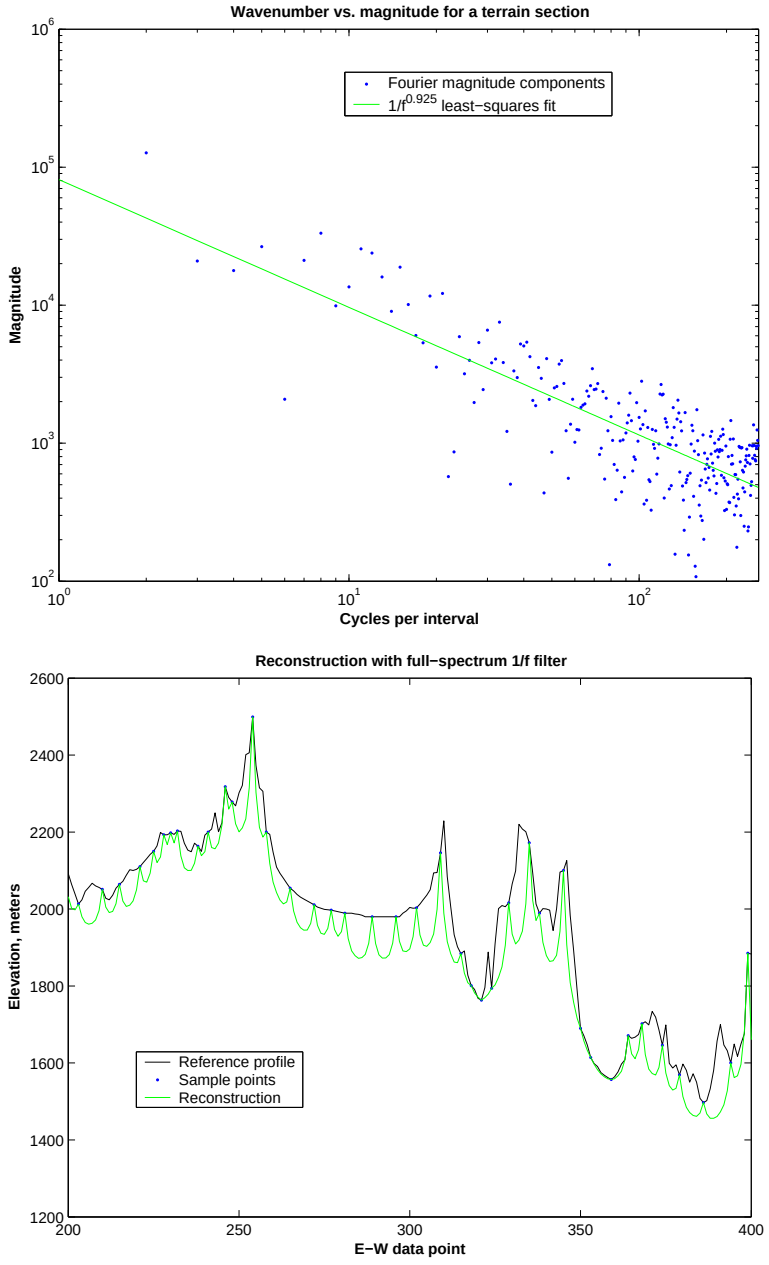


Figure 3.19: Bottom: reconstruction (green) of terrain profile (black) from irregular samples (blue) using full-spectrum $1/f$ filter. Fourier magnitude spectrum of nearby section shown above.

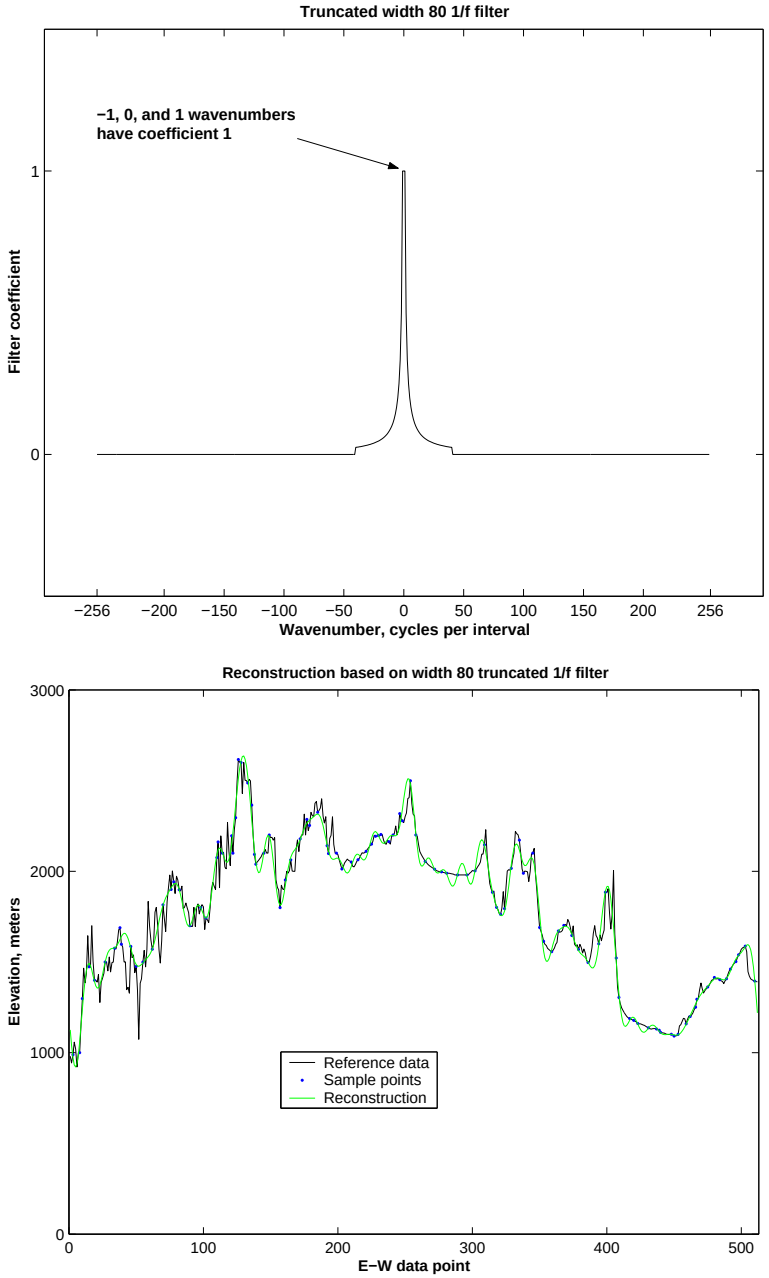


Figure 3.20: 1/f filter of width 80 (above), and reconstruction based on this filter (below).

reconstruction obtained by addition of the surrogate high-frequency components of the spectrum on the lower left to the spectrum on the upper left.

One-step approach, using wider filter with faster decay

Figure 3.21 suggests that the real data magnitude spectrum components may fall off more quickly at the lower frequencies than those of the reconstruction. By experimentation it was found that a wider $1/f$ filter could be used in the reconstruction without generating such objectionable cobweb effects if the higher frequencies were suppressed by applying a larger negative exponent to f. Figure 3.22 shows reconstructions based on $1/f^{1.5}$ filters of width 256 (top) and of full signal width (bottom).

The narrower filter still seems preferable, and the result may be considered to be a good compromise interpolation, as it is closer to "exact" for the sample points than the interpolation textured by spectral addition, yet it appears less sinusoidal than a more strictly bandlimited interpolation.

3.4.2 Final remarks on iterative Fourier-based interpolation of irregularly-sampled data

Although I have not shown examples here, all the above methods are adaptable to interpolation of data in two or higher dimensions (within the limits of computational complexity). For signals that are bandlimited, and for which the bandwidth is known, the results are impressive, especially for images (see, *e.g.* [21]), and the computation algorithms are quite efficient, compared to other advanced interpolation methods such as Kriging. I have shown that it is possible to adapt the method to signals of special (known) spectral characteristics, which provides the opportunity to carry out efficient interpolation with attention to this aspect of signal context. The main concern would seem to be to adapt the filter to the spectral context of the signal to be interpolated.

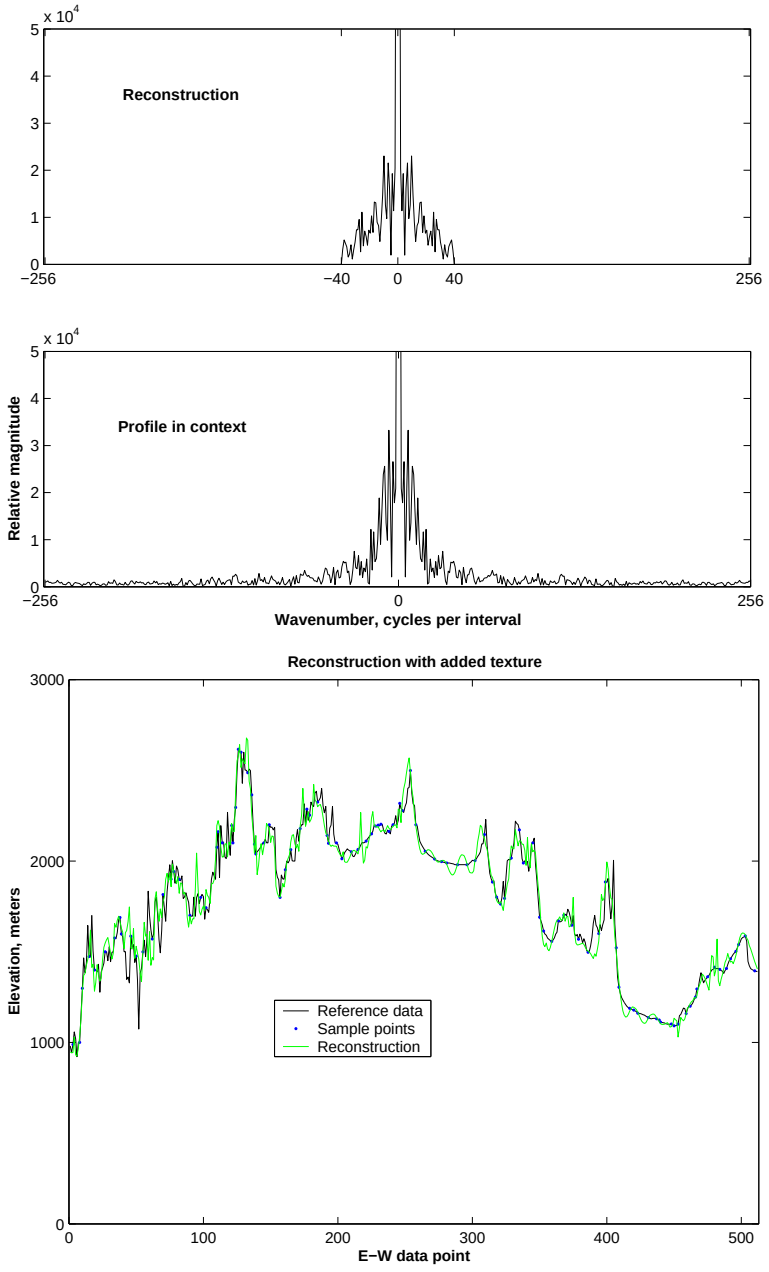


Figure 3.21: Above: Fourier magnitude spectra of reconstruction from width-80 $1/f$ filter and of nearby profile. Below: textured reconstruction with high-frequency components from nearby profile added.

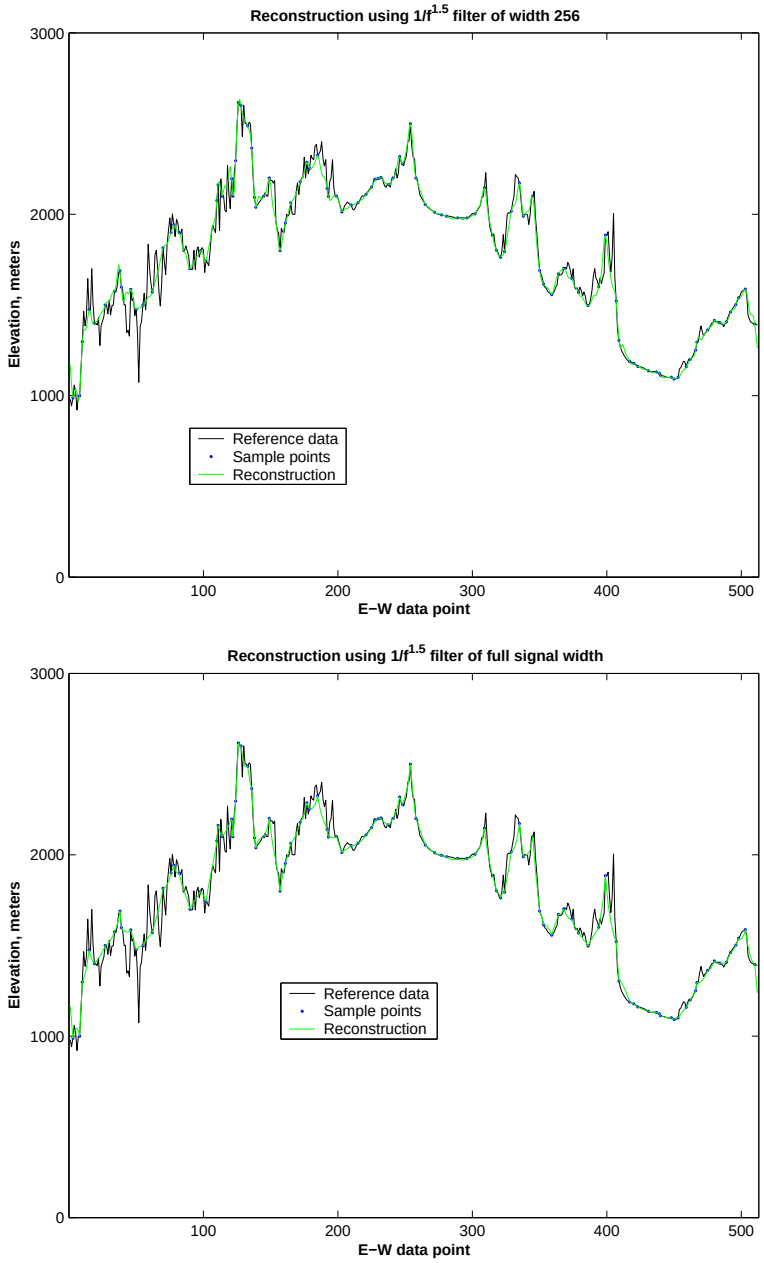


Figure 3.22: Reconstruction using wider filters with faster decay.

3.5 Principal component basis for interpolation

The transforms that we have considered so far have been well-defined for any signal, regardless of context. Selection of components may have been adaptive, as described in Part 1 in the discussion of nonlinear compression schemes which select discrete cosine transform or discrete wavelet transform coefficients in order of size. But the transform itself has not depended on the signal or class of signals.

In contrast, principal component transforms, also called *Karhunen-Loève* transforms or *Hotelling* transforms, after various originators (see, *e.g.* [31]) are *adaptive* transforms that are custom made for a signal source, or class of signals. So they are very useful, in fact in some ways optimal, for looking at any signals known to belong to that class.

From a statistical point of view, the object of principal component analysis is to transform a number of measured variables into a new set of abstract component "variables" which are linear combinations of the original variables, the first component containing the most variance, with the remaining mutually independent components accounting for decreasing amounts of variance. For our purposes, the "variables" are the coefficients (values) of the signal at basis points.

From a geometric point of view, a class of signals occupies a region in a function space of as many dimensions as the signals have points of definition. The class of signals is like a cloud of points in function space. There will be some axis through this cloud which is its *major axis*, the direction in which it has the greatest extent (variance). This direction, then, is the Principal Component function direction, the direction of the first component *eigenvector*, or *eigenfunction* of the distribution.[15] Next, there will be an axis, orthogonal to the first, in which the cloud has next greatest extent. *Given a number N of completely sampled signals that are members*

[15]The *eigenvalue* associated with this eigenvector is related to the extent of the cloud in that dimension.

of the class, this imaginary process of finding next largest orthogonal axes can be continued until one has found $N-1$ orthogonal axes, or as many orthogonal axes as the space has dimensions, whichever is lesser. At this point a new *orthogonal base* has been found for the function space or an $N-1$-dimension projected subspace of the function space. Equivalently one might say that a point of view has been found which is especially suited for viewing this distribution of signals (and for viewing interpolation candidates for inclusion within the bounds of the distribution).

3.5.1 Application to a class of bathymetry profiles

To prove how principal component analysis can be applied to interpolation of geographic data, we start with a set of eight bathymetry profiles from the Sea of Cortez, each comprising 45 soundings. These profiles and their locations are shown in figure 3.23. Profile 4 is the reference profile, which will be reconstructed from 15 irregularly-spaced sample points.

Profiles $\mathbf{p1}, \mathbf{p2}, \mathbf{p3}, \mathbf{p5}, \mathbf{p6}, \mathbf{p7}$ and $\mathbf{p8}$ are the presumed representatives of a class of profiles to which principal component analysis is applied. The result is a set of what I call *eigenprofiles* (afer the "eigenfaces" of Everson & Sirovich [19], who applied the method to images of faces). They are so called because they are computed as the eigenfunctions of the covariance matrix of the given representative profiles. This is a *Karhunen-Loève transform* (KLT), and the eigenprofiles make up the new basis for representing profiles that are taken to be members of the class.

The method of computation (using MATLAB) was to construct a 7 x 45 matrix $\mathbf{P}$ whose 7 rows are the 7 profiles used to find an eigenprofile basis. The covariance matrix $\mathbf{C}$ is computed

$$\mathbf{C} = cov(\mathbf{P}) \ ;$$

and after this, a matrix $\mathbf{V}$ of eigenvectors and a diagonal matrix $\mathbf{D}$ of eigenvalues is

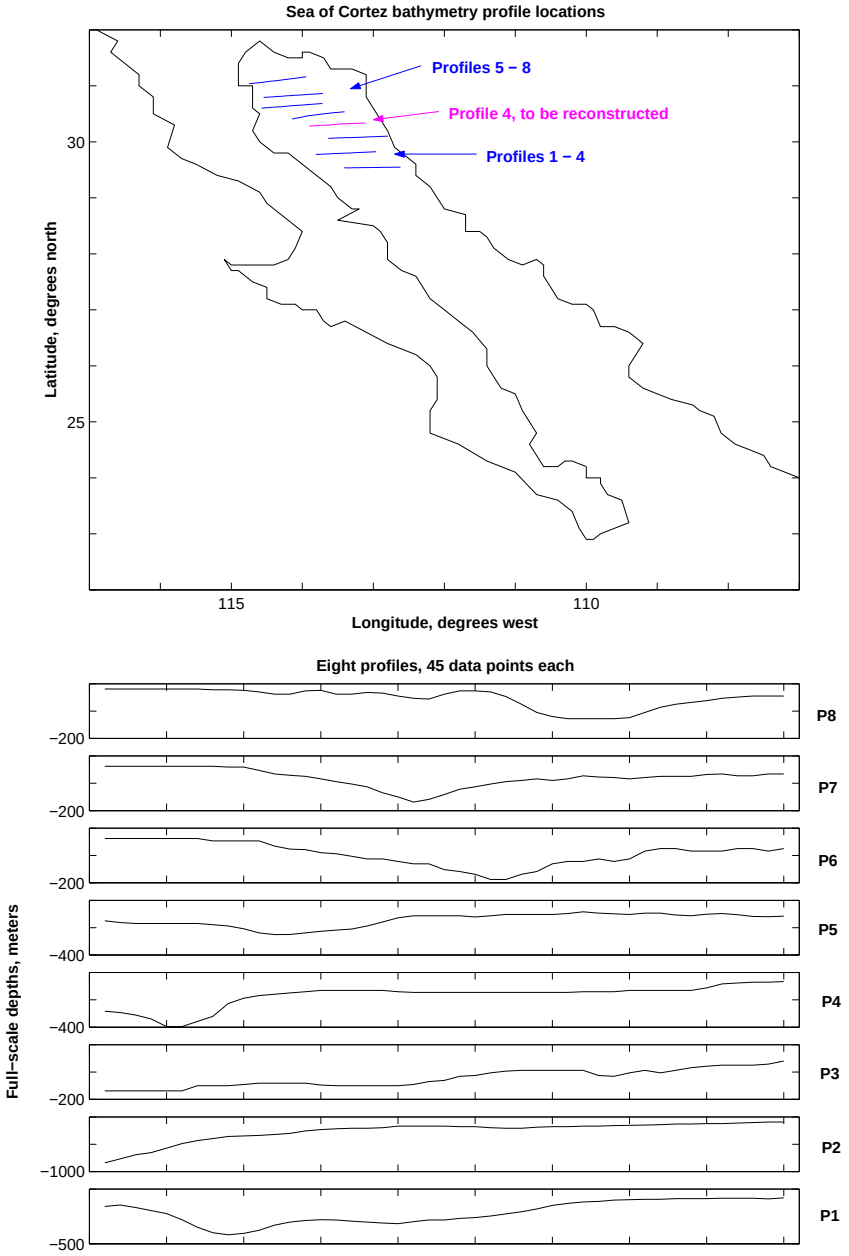


Figure 3.23: Sea of Cortez bathymetry profiles used for "eigenprofile" interpolation.

computed

$$[\mathbf{V}, \mathbf{D}] = eig(\mathbf{C}) \ .$$

The last six columns of $\mathbf{V}$ are the basis eigenfunctions in reverse order of importance, which importance is indicated by the last six eigenvalues on the diagonal of $\mathbf{D}$, which in this case are

$$[400, 3920, 4970, 13140, 93410, 464700] \ .$$

The six basis eigenprofiles are shown in Figure 3.24, along with a reconstruction sequence for **p1** (which was one of the profiles from which the basis was derived).[16]

Evidently the identity of this profile is well determined from no more than three components. This is the strength of principal component analysis, *provided that the signal to be reconstructed is a member of the class described by the representative signals used to generate the eigenfunction basis.* [19]

3.5.2 Computing eigenprofile coefficients from irregular samples of signal

The method of finding a least-squares solution for the eigenprofile coefficients of the sampled profile is analogous to the method detailed in Section 3.3 for wavelet coefficients. The inverse KLT transform matrix $\mathbf{I}$ is constructed by assembling the eigenprofiles as columns of a 45 (row) x 6 (column) matrix. This is the matrix which may be multiplied by a column vector $\mathbf{e_p}$, of eigenprofile coefficients for some profile $\mathbf{p}$, to yield the reconstructed profile $\hat{\mathbf{p}}$ in the standard basis:

$$\mathbf{I}\mathbf{e_p} = \hat{\mathbf{p}} \ .$$

To solve for a partially sampled profile, remove the rows of $\mathbf{I}$ corresponding to the missing data points, leaving a 15 x 6 matrix $\mathbf{I_s}$. There are 15 sample points in the sample vector which we'll call $\mathbf{p4_s}$, and we seek 6 eigenprofile coefficients, a vector

[16] This reconstruction sequence is analogous to the Fourier-based synthesis sequence for a bathymetry profile that was presented in Part 1.

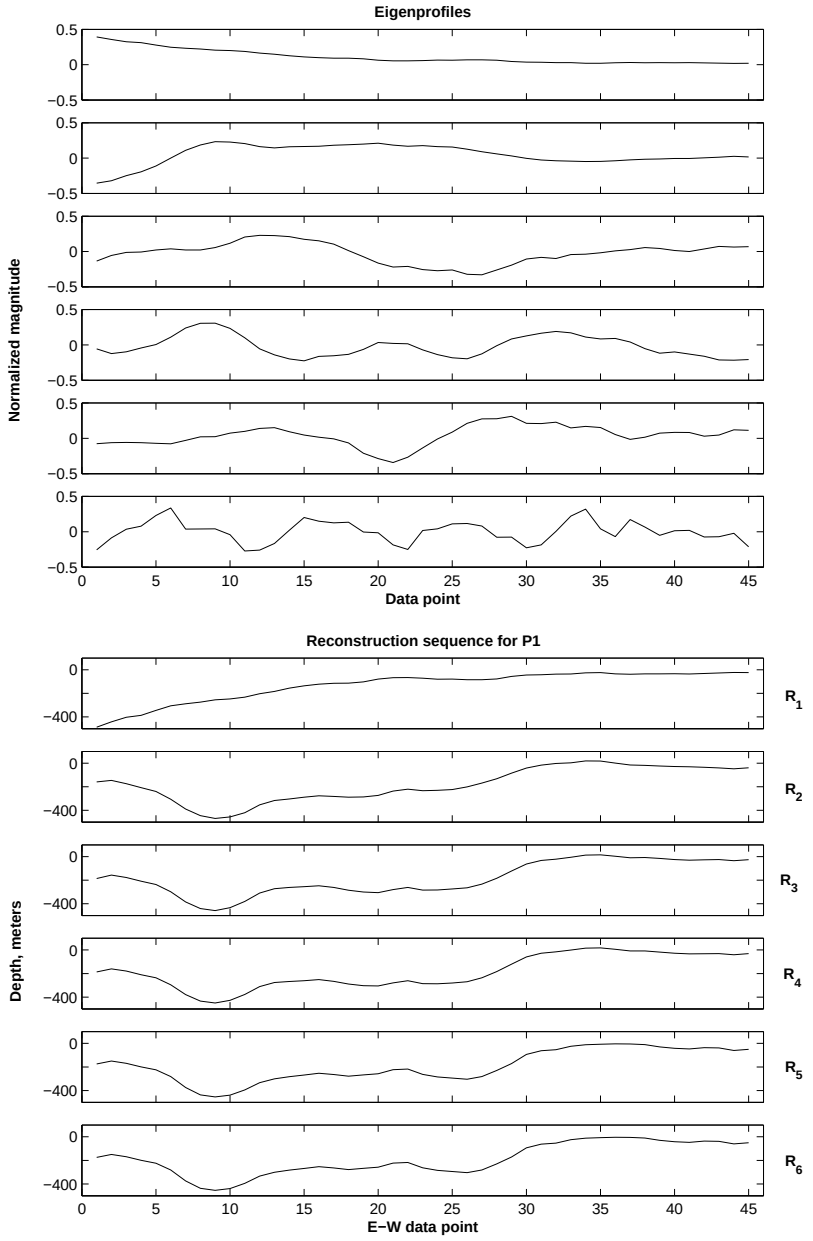


Figure 3.24: Basis of six eigenprofiles derived from seven representative profiles (above) and reconstruction of one member by sequential component addition (below.

called $\mathbf{e_{p4}}$. The system of equations to solve for the eigenprofile coefficients can then be represented by the matrix equation

$$\mathbf{I_s e_{P4}} = \mathbf{p4_s} \ .$$

The least-squares solution to this matrix equation is computed (in MATLAB notation) by

$$\mathbf{\hat{e}_{p4}} = \mathbf{I_s} \backslash \mathbf{p4_s} \ .$$

The least-squares solution for the eigenprofile coefficients of $\mathbf{p4}$, $\mathbf{\hat{e}_{p4}}$, is then multiplied by $\mathbf{I}$ to arrive at $\mathbf{\hat{p}4}$, the desired interpolation of $\mathbf{p4}$ from samples via the eigenprofile basis:

$$\mathbf{\hat{p}4} = \mathbf{I\hat{e}_{p4}} \ .$$

The result is shown in Figure 3.25. Clearly the outcome is not impressive from a practical point of view, probably because the representative profiles were too few to very well "surround" the unknown profile in function space., But the method has been proven to be workable, and should be considered when data sets are known to be closely related (*i.e.* to occupy a small region in function space), and when a large number of complete signals from context are available for analysis. It is interesting to observe that a reconstruction of the unknown profile $\mathbf{p4}$ using only *three* estimated eigenprofile coefficients, shown at the bottom of Figure 3.25, appears as good, perhaps even better, than the reconstruction using six, shown at the top.

3.6 Generalization of least-squares basis fitting

In Sections 3.3 and 3.5, transform basis reconstructions of signals were generated by finding least-squares solutions to systems of equations. The computations were carried out by formulating matrix equations using parts of the inverse transform matrices, for the wavelet transform and for a principal component transform. This method can be generalized to the case of any transform whose inverse (for a given

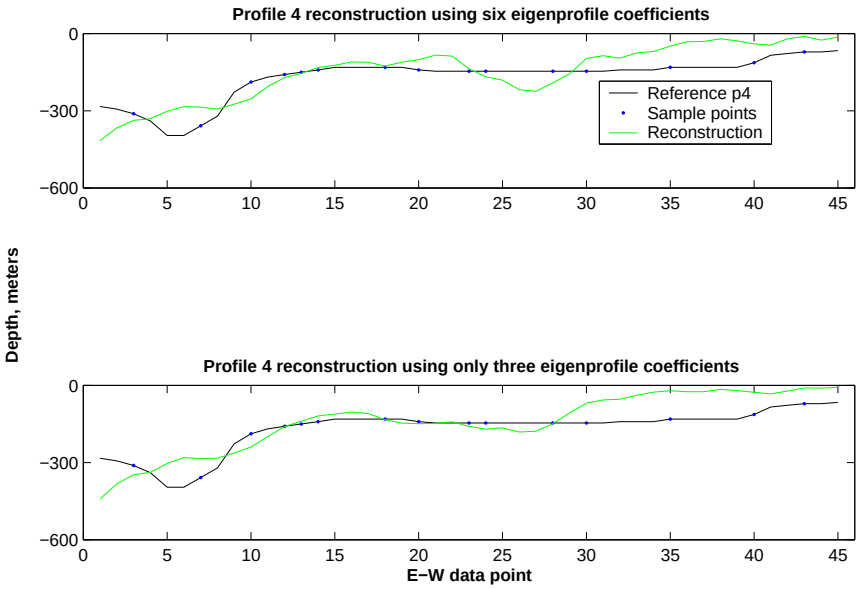


Figure 3.25: Reconstruction of 45-point bathymetry profile from 15 sample points, using basis of six eigenprofiles (top) and three eigenprofiles (bottom).

number of dimensions) can be written as a matrix $\mathbf{T}^{-1}$. In this matrix, the columns are the representations (*i.e.* the coefficients), in the standard base, of the new base components. In this case, if $\acute{\mathbf{s}}$ is the transform-basis representation of the signal, and $\mathbf{s}$ is the standard-base representation of the signal, the inverse transform is written as the matrix multiplication

$$\mathbf{T}^{-1}\acute{\mathbf{s}} = \mathbf{s} \ .$$

Then if a set of samples of $\mathbf{s}$ is taken at points $\mathbf{p}$, so that we can write

$$\mathbf{s}_\mathbf{p} = \mathbf{s}(\mathbf{p}) \ ,$$

the general system of equations to solve can be written

$$\mathbf{T}_\mathbf{p}^{-1}\acute{\mathbf{s}} = \mathbf{s}_\mathbf{p} \ ,$$

where $\mathbf{T}_\mathbf{p}^{-1}$ is $\mathbf{T}^{-1}$ having only the rows corresponding to $\mathbf{p}$. In MATLAB notation,

$$\mathbf{T}_\mathbf{p}^{-1} = \mathbf{T}^{-1}(\mathbf{p}, :) \ . \tag{3.1}$$

After an estimate of transform coefficients $\hat{\mathbf{s}}$ is found by

$$\hat{\mathbf{s}} = \mathbf{T}_{\mathbf{p}}^{-1} \backslash \mathbf{s}_{\mathbf{p}} \ , \tag{3.2}$$

the signal reconstruction $\hat{\mathbf{s}}$ is computed as

$$\hat{\mathbf{s}} = \mathbf{T}^{-1} \hat{\mathbf{s}} \ . \tag{3.3}$$

A "band limited" matrix computation can be formulated if solution is sought for a limited set of significant components.[17] This is done by removing the columns of the inverse transform matrix $\mathbf{T}^{-1}$ corresponding to components that are considered to be insignificant. For certain transforms, in particular principal component transforms, Fourier transforms, and cosine transforms, we have seen that components have an order of importance, with respect to the relative sizes of transform coefficients used to represent given signals —the order of importance is from low frequency to high frequency in the case of the discrete cosine transform, for example. Empirically one discovers that, given incomplete sampling, in general it is practical to solve for about 2/3 as many coefficients as the number of sample points.

Such a solution is actually an estimated *projection*, onto a bandlimited subspace of the full-dimension space spanned by the full set of transform basis components. In any case, the matrix computation takes the following form: from the full inverse transform matrix $\mathbf{T}^{-1}$ one might select the first n components to generate an inverse projection ("bandlimited") transform matrix $\mathbf{T}_b^{-1}$; in MATLAB notation

$$\mathbf{T}_b^{-1} = \mathbf{T}^{-1}(:, 1:n) \ ;$$

then one proceeds as above, substituting $\mathbf{T}_b^{-1}$ for $\mathbf{T}^{-1}$ in equations 3.1 to 3.3.

3.6.1 New routines for DCT and FFT basis interpolation

My MATLAB routines DCTINTERP and FFTINTERP, for computation of one-dimensional interpolations using the discrete cosine transform and the fast Fourier transform, are

[17] The term "band limited" is used in the generalized sense, introduced in Part 1, to refer to limitation of the set of components whose coefficients are non-zero.

included in the appendix. The routines accommodate irregularly-spaced samples, and allow for selection of the number of coefficients to estimate. Note that the routine FFTINTERP is different from the MATLAB function INTERPFT, which is not for irregularly-spaced samples.[18]

Even as an interpreted function (*i.e.* not a compiled C program), FFTINTERP actually ran about 50 percent faster in tests than INTERPFT. More importantly, FFTINTERP also appears to be a viable alternative to the iterative reconstruction methods discussed in section 3.4, with the notable advantage that *the true signal bandwidth need not be known*, and it can be applied to signals with $1/f$ spectral characteristic without any special preparation.

Figure 3.26 illustrates the similar results of FFTINTERP and Fourier-based iterative reconstruction. The upper half shows interpolation from 109 sample points of a 512-point signal bandlimited to 64 (zero, positive, and negative frequencies). The reference signal was first plotted in black; in both cases the reconstruction is near perfect. The lower half shows interpolation of the same 512-point terrain profile used in previous sections. The results using FFTINTERP were obtained without applying any *a priori* knowledge of the $1/f$ signal spectrum, but a more "exact" reconstruction was had by solving for 92 coefficients (from the same 109 sample points used in previous sections), than by solving for the default 72 coefficients (two thirds the number of sample points).

Computation times were compared: for the bandwidth-64 signal, FFTINTERP took 0.321 sec, solving for 72 coefficients; the iterative method using ADPWCG took 1.152 sec, using a width-64 boxcar filter. For the $1/f$ terrain profile, FFTINTERP took 0.38 sec, solving for 92 coefficients; ADPWCG took 0.86 sec, using a truncated 1/f filter of length 92. From these results I conclude that the FFTINTERP method may be a

[18] FFTINTERP handles regularly-spaced data as a special case and returns essentially the same result as INTERPFT in that case, if the coefficient-number parameter is set close to the original number of data points.

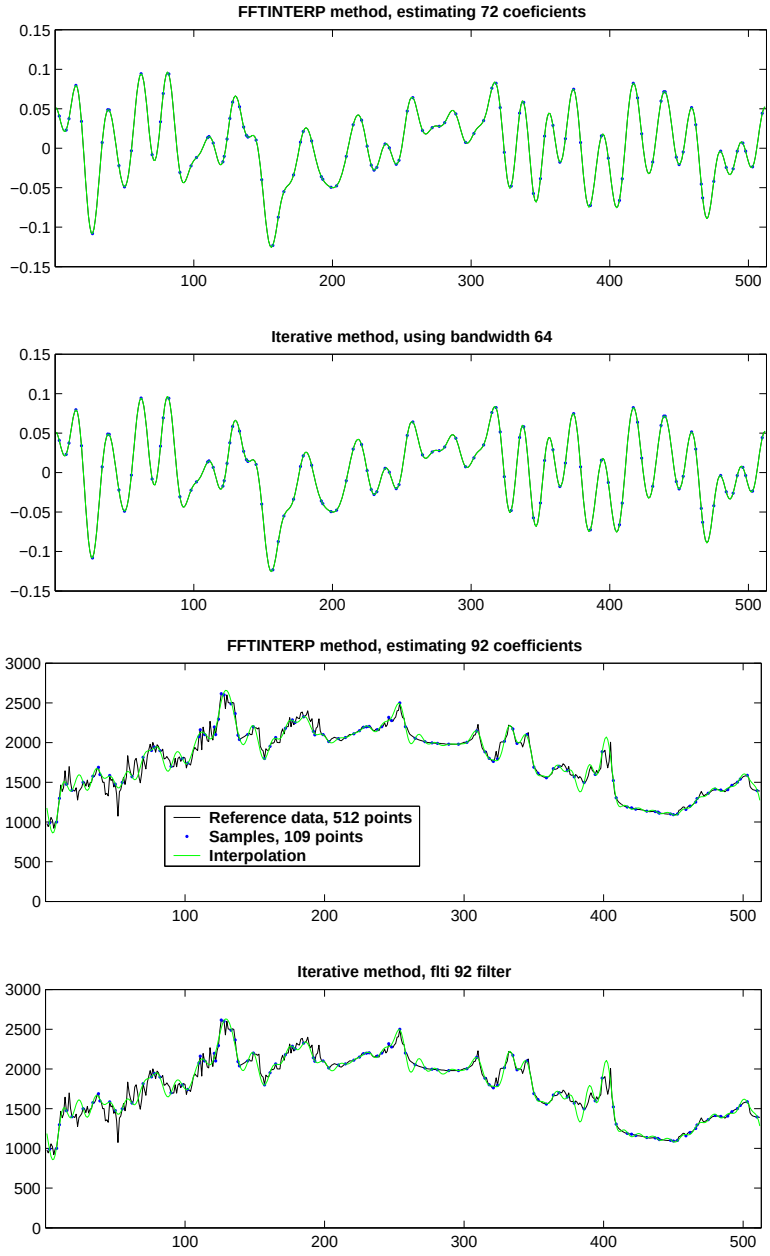


Figure 3.26: Comparison of FFTINTERP and ADPWCG, interpolating 64/512 bandlimited signal and $1/f$ terrain profile.

better starting point than the iterative method for the interpolation ideas discussed in section 3.4.

3.6.2 Example of discrete cosine transform basis interpolation

As further illustration of the general method of interpolation in any basis, we do a DCT-basis reconstruction of the same bathymetry profile **p4** of Section 3.5, from 15 irregularly-spaced sample points, using my MATLAB routine DCTINTERP. As was illustrated in Part 1, most of the "energy" of a DCT representation of a bathymetry profile is in the first few components (as was also eminently the case for the KLT representation of Section 3.5, and for FFT representations as well). For good results, it was therefore found best to limit the number of coefficient equations to solve to approximately two thirds the number of sample points. Figure 3.27 shows the DCT coefficients (top and third plots) and the standard-base coefficients (second and fourth plots). Notice that the plot of DCT coefficients for the reference data and the 10 estimated coefficients suggests that one might do well to estimate an additional 5 DCT components. However, the results of such an attempt are shown in the bottom plot.

3.7 A New Basis for Assessing Interpolation Accuracy

Research in naturally evolved visual information processing systems suggests that perception of *texture*, and *edges*, is as important for recognition as perception of *shape* [47], [28]. According to Field, [22] the mammalian vision system likely employs a transform that emphasizes higher-frequency components compared to lower-frequency components.

This is a motivation for seeking a transform that might have such an effect —in order to represent the components of a scene in better reflection of their relative importance in perception. Such a transform might in turn be of interest to us in questions of

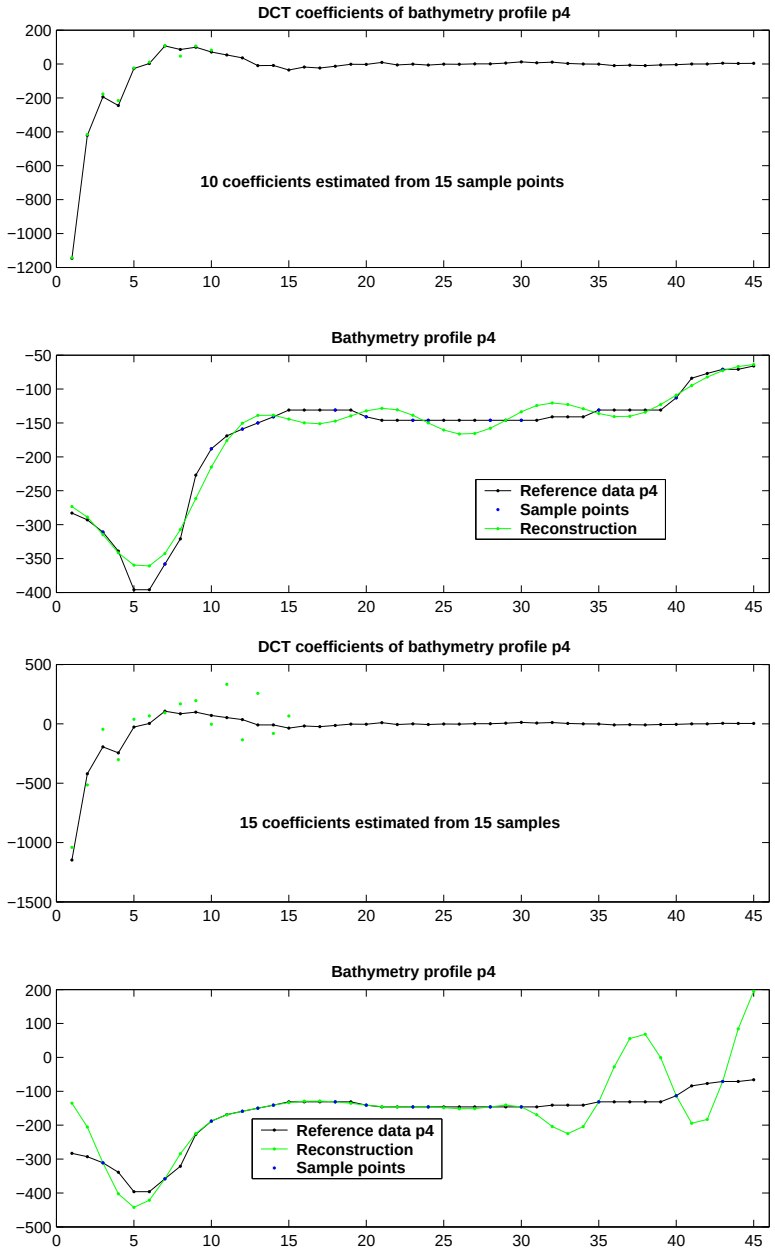


Figure 3.27: Discrete cosine transform basis interpolation of 45-point profile from 10 sample points.

interpolation and representation of field data, on the grounds that naturally-evolved vision systems are likely well-suited for sorting the patterns of order and disorder that characterize spatial distributions in general, and on the grounds that the ultimate objective for our purposes is effective visual communication of whatever *meaning* there might be in spatial information.[19]

3.7.1 Proposed frequency-weighted discrete cosine transform

As a simple exploration of the idea of a transform in which higher-frequency components figure more importantly than lower-frequency components, I propose the frequency-weighted discrete cosine transform (implemented in my MATLAB function DCTF), which is just the discrete cosine transform followed by multiplication of the components in proportion to the frequencies that they represent. The first component, representing frequency 0 (or "DC bias"), has unity weight; the second component, representing one half cycle over the period of the signal, is multiplied by two, the third component by three, and so forth.

This is a linear transform, and it is invertible. It results in an orthogonal basis for representation of all functions that could be represented in the original basis, but it is specifically *not* an orthonormal base. This means that, since the amplitudes of the component waves increase with frequency, the *norms*, or lengths, of the base components (thought of as vectors) vary, when they are regarded in the original base.

Figure 3.28 shows how the frequency-weighted discrete cosine transform (DCTF) could be used to compare the accuracy of two approximations of a reference signal. The reference signal (top) is synthetic, 256 points generated by FINV. Its DCTF representation is shown in the third plot from bottom. The first approximation (red) is a "smooth" approximation obtained by taking the inverse discrete cosine transform of just the first 64 DCT coefficients. Its DCTF representation is also shown in the

[19] This is human engineering, so we take clues from nature, our source.

second plot from bottom. The second approximation (blue) was obtained by taking the FFT *phase* information *only* from the reference signal, and adding generic $1/f$ magnitude "information" generated by my FLTI routine.[20]

The accuracy of an approximation to a signal can be measured by the distance between the two signals regarded as points in function space. This is called the *norm* of the difference between the signals, component by component. The norms of the differences for this example can be summarized as follows: in the standard basis, Reference - Bandlimited = 0.0976; Reference - Synthetic = 0.5019 —that is, 5.14 times as far away, as might be expected from observation that the *structure* of the synthetic signal departs noticeably from that of the reference signal. But in the DCTF basis, the norms are: Reference - Bandlimited = 12.1147; Reference - Synthetic = 6.7191 —the approximation with more appropriate texture is at 0.55 times the distance from the reference signal compared to the smooth approximation.

Just viewing the DCTF representations allows a quicker assessment of which signal is odd man out, and the relatively small norm differences actually point to a weakness of the proposed DCTF representation as a reflection of human perception. The DCTF representation overemphasizes the differences between particular high-frequency components, whereas natural perception, while sensitive to those components, may treat them integrally. In any case, the potential to quantify accuracy by a different measure has been demonstrated.

All of the linear transforms that we have considered heretofore could be understood as changes of perspective in many-dimensional space —or, equivalently, as an arbitrary rotation or translation of the "coordinate system" in which each *direction* is a certain function *shape*, and the distance from the origin is the *size*. The relative distances between functions was unchanged: if a certain proposed interpolation appeared

[20]There is no particular reason in this case for generating appropriate texture this way, except to illustrate once again the remarkable importance of phase in signals. Interpolation on this basis might have application in remote sensing where signals' amplitude can be more affected than phase, in transmission through the atmosphere. [53]

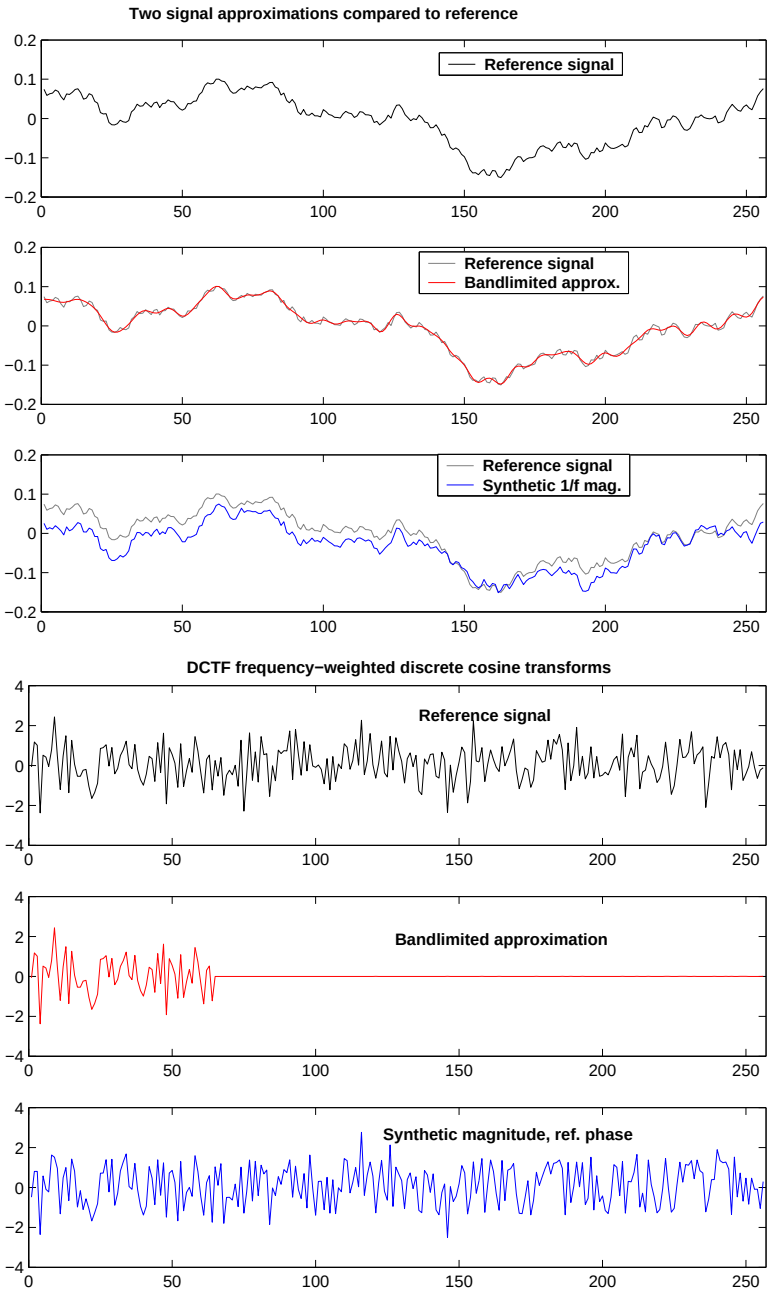


Figure 3.28: Synthetic $1/f$ signal and two approximations, with DCTF representations to illustrate alternative accuracy criteria.

to be closer to a target function than another proposed interpolation, when viewed in the standard base, that relationship did not change with the new perspective afforded by a transform.

Now we are entertaining the idea of a transform that involves some *stretching* of the function space (or of the coordinate system supporting it). Even though the transformed base vectors (the component functions of the new base representation) are not all of unit length when viewed in the original base, you might say that they *have* to be regarded as being of equal length from the new perspective, because there is nothing to measure them against in that view!

3.7.2 Implication for assessing interpolation accuracy

Euclidean distance is not preserved by the proposed frequency-weighted transform. The *norms* compared above are more precisely referred to as "2-norms" because they are computed as the 1/2 power of the sum of components raised to the power 2. They are the same as Euclidean distance. The Euclidean distance between two real-valued functions (vectors) $\mathbf{x}$ and $\mathbf{y}$ of length n is

$$[\sum_{n}[\mathbf{x}_n - \mathbf{y}_n]^2]^{0.5} \ ,$$

essentially the same (without normalization) as the "root mean square" (rms) measure of accuracy of an approximation to a given function. So by considering this transform which does not preserve Euclidean distance, we introduce a new measure for judging the accuracy of an interpolation. It is not a rubber yardstick fabricated *ad hoc* for the occasion, but is founded in understanding of how our own vision system works. It should be considered in the spirit of relativity of perspective.

4 Summary and Conclusions

In this work we explored unconventional approaches to the problem of interpolation of spatial data, in an attempt to strengthen inter-disciplinary connections between geographic techniques and the fields of information theory, signal processing, and (to a lesser extent) the study of natural perception systems. Following geographers' preference for thinking spatially, we've employed geometric models of mathematical concepts in order to understand problems of spatial analysis. Sets of geographic data have been considered as signals —points in a function space— and various transforms have been considered as ways to obtain different views of that signal space.

The question of what geographic information is and where it resides has been explored, with reference to traditional probabilistic information theory (information as the unexpected), to algorithmic information theory (information as a program running in a context), and to the conjecture that information is ultimately nothing more than a nesting of contexts. Interpolation has been interpreted to be an attempt to add non-information, or an attempt to place a construction in context.

Several concepts and methods of signal processing have been shown to be appropriate to the analysis of spatial data, when such data are taken to be signals. Two examples of the importance of spectral *phase*, as well as spectral magnitude, to both texture and shape, were shown. A fractal-based method for interpolation, described in a patent for super-resolution of images, was implemented in computation routines and applied with some success to interpolation of a digital elevation model. The method was found to provide a means of analyzing data *detail* as a function of *scale*. An eightfold search enhancement was invented, but tested with negative results. Implementation of the similarity search routine in the forms of both interpreted MATLAB "m-files" and compiled C "mex files" provided the opportunity to address the question of computational complexity.

The discovery was made that scale analysis depends critically on the sampling

mode: whether it comprises point samples or integral area averages. Since scale analysis is akin to frequency band analysis, it is subject to aliasing in the case of point samples. But one experiment to simulate integral area samples from point samples by gaussian filtering, before carrying out fractal-based super-resolution, produced poor results.

The relation between image pyramid decomposition and wavelet analysis has been noted, and analogous search algorithms for super-resolution based on wavelet decompositions were developed and found to produce results of questionable practical value for the case of a terrain profile. Ford & Etter's "multi-resolution basis fitting reconstruction" method of interpolation based on two-stage solution for wavelet coefficients was tested, using the "db2" wavelet base, with satisfactory results. However, solving for higher detail where data were more dense proved impractical if the number of data points was not much greater than the length of the wavelet filter. Biorthogonal wavelet bases were not found to be workable.

A new method of wavelet-based super-resolution was proposed, based on not subsampling wavelet coefficients after convolution, but feeding them into an inverse discrete wavelet transform. Results and consideration of the simplicity of the method suggested that it might be worth investigating further, although analysis of the Fourier magnitude spectra of the results revealed weaknesses, as with the other wavelet-based methods.

Fourier-based iterative methods for the reconstruction of bandlimited signals from irregular samples were applied to terrain profile data, with the effect of various filters investigated. The addition of surrogate texture was also investigated, but satisfactory results were had most simply by use of a $1/f^{1.5}$ filter of half the signal length. Although not specifically investigated herein, the Fourier-based iterative methods should be very well suited to reconstruction of satellite imagery from which scan lines are missing or corrupt.

The application of principal component analysis to topography data was demonstrated for a set of eight 45-point bathymetry profiles, deriving the "eigenprofiles" for the class of data. This was introduced as a possible way to take into account *context* of data, after its great success for certain applications of image reconstruction. But the class of profiles was small, and the results were not convincing for this method.

General methods of matrix-based computation have been presented; in particular a new routine FFTINTERP shows considerable promise as an alternative to iterative Fourier-based methods. The two apparent advantages of this computationally faster method are that bandwidth of the unknown signal need not be known, and that the method can be applied to signals of $1/f$ spectral characteristic without any special provisions beyond choice of the number of coefficients to estimate.

In the final section the DCTF function (for frequency-weighted discrete cosine transform) was introduced as an example of a linear invertible transform leading to a non-orthonormal base of representation that might more closely reflect what is important to human perception. An example was constructed showing two approximations to a reference signal, the smoother one being closer by the standard measure, and the one with more similar texture being closer by the alternative measure. This transform was not offered as the optimum alternative for this purpose, but to demonstrate that not only are representations of information relative to bases, but also the criteria for judging the accuracy of interpolation need not be considered absolute.

Bibliography

[1] H. Abarbanel, T. Frison, and L. Tsimring. Obtaining order in a world of chaos. *IEEE Signal Processing Magazine*, 15(3):49–65, 1998.

[2] A. Aleksandrov, A. Kolmogorov, and M. Lavrent´ev, editors. *Mathematics: Its Content, Methods, and Meaning*. M.I.T. Press, Cambridge, MA, 1956. Translated by S. Gould and T. Bartha, 1963.

[3] P. Bak. *How Nature Works*. Springer-Verlag, New York, 1996.

[4] M. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. *Proc. Roy. Soc. London, A, Math. Phy. Sci.*, 399:243–275, 1985.

[5] M. Barnsley and A. Sloan. A better way to compress images. *Byte*, pages 215–223, January 1988.

[6] T. Bell. Statistical features of sea floor topography. *Deep Sea Research*, 22:883–892, 1975.

[7] I. Briggs. Machine contouring using minimum curvature. *Geophysics*, 39:39–48, 1974.

[8] F. Bucher and A. Vckovski. Improving the selection of appropriate spatial interpolation models. In A. Frank and W. Kuhn, editors, *Spatial Information Theory*. Springer-Verlag, Berlin, 1995.

[9] P. Burrough. Fractal dimensions of landscapes and other environmental data. *Nature*, 294:240–242, 1981.

[10] G. Chaitin. Godel's theorem and information. *International Journal of Theoretical Physics*, 22:941–954, 1982.

[11] K. Clarke. Scale-based simulation of topographic relief. *The American Cartographer.*, 15:173–181, 1988.

[12] L. Cordell. A scattered equivalent-source method for interpolation and gridding of potential field data in three dimensions. *Geophysics*, 57(4):629–636, 1992.

[13] N. Cressie. *Statistics for Spatial Data*. Wiley, New York, 1993.

[14] C. Dampney. The equivalent source technique. *Geophysics*, 34(1):39–53, 1969.

[15] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications in Pure and Applied Mathematics*, 41:909–996, 1988.

[16] F. DeClercq. Interpolation methods for scattered sample data: Accuracy, spatial patterns, processing time. *Cartography and Geographic Information Systems*, 23:128–144, 1996.

[17] P. Doucette and K. Beard. Exploring the capability of some GIS surface interpolators for dem gap fill. *Photogrammetric Engineering and Remote Sensing*, 66:881–888, 2000.

[18] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann, San Francisco, 3rd edition, 2003.

[19] R. Everson and L. Sirovich. Karhunen-Loève procedure for gappy data. *J. Opt. Soc. Am. A*, 12(8):1657–1664, 1995.

[20] H. Feichtinger and K. Gröchenig. Theory and pracice of irregular sampling. In J. Benedetto and M. Frazier, editors, *Wavelets: Mathematics and Applications*. CRC Press, Boca Raton, FL, 1994.

[21] H. Feichtinger and T. Strohmer. Fast iterative reconstruction of band-limited images from irregular sampling values. In D. Chetverikov and W. Kropatsch, editors, *Computer Analysis of Images and Patterns*, Berlin, 1993. Springer-Verlag. 5th international conference, CAIP '93, Budapest, Hungary.

[22] D. J. Field. Scale invariance and self-similar wavelet transforms: An analysis of natural scenes and mammalian visual systems. In M. Farge, J. C. R. Hunt, and J. C. Vassilicos, editors, *Wavelets, Fractals, and Fourier Transforms*. Oxford University Press, New York, 1993.

[23] I. Florinsky. Errors of signal processing in digital terain modelling. *Int. J. Geographic Inf. Sci.*, 16:475–501, 2002.

[24] C. Ford and D. Etter. Wavelet-based reconstruction of nonuniformly sampled data. *IEEE Transactions on Circuits and Systems II*, 45:1165–1168, 1998.

[25] C. Fox. Empirically-derived relationships between fractal dimension and power law form frequency spectra. *Pure andApplied Geophysics*, 131:211–239, 1989.

[26] R. B. Fuller. *Synergetics 2: Explorations in the Geometry of Thinking*. MacMillan, New York, 1979.

[27] V. Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, 2001.

[28] J. Havlicek, A Bovik, and D. Chen. AM-FM image modeling and Gabor analysis. In C. Chen and Y-Q Zhang, editors, *Visual Information Representation Communication and Image Processing*. Marcel Dekker, New York, 1999.

[29] K. Hindricks and A. Duijndam. Reconstruction of 3-D seismic signals irregularly sampled along two spatial coordinates. *Geophysics*, 65:253–263, 2000.

[30] D. Hofstadter. The location of meaning. In *Gödel, Escher, Bach: an Eternal Golden Braid*, chapter 6, pages 158–180. Basic Books, New York, 1979.

[31] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, and 498–520, 1933.

[32] J. Huang and D. Turcotte. Fractal image analysis: application to the topography of oregon and synthetic images. *J. Opt. Soc. Am.*, A7:1124–1130, 1990.

[33] B. Hubbard. *The World According to Wavelets*. A. K. Peters, Ltd., Wellesley, 2nd edition, 1998.

[34] A. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1:18–30, 1992.

[35] A. Jerri. The Shannon sampling theorem —its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, 1977.

[36] A. Journel. *Fundamentals of Geostatistics in Five Lessons*. American Geophysical Union, Washington, 1989.

[37] A. Journel and C. Huijbregts. *Mining Geostatistics*. Academic Press, New York, 1978.

[38] S. Kauffman. *At Home in the Universe, The Search for the Laws of Self-organization and Complexity*. Oxford University Press, New York, 1995.

[39] A. J. Kimerling. Predicting data loss and duplication when resampling from equal-angle grids. *Cartography and Geographic Information Science*, 29(2):111–126, 2002.

[40] D. G. Krige. A statistical approach to some basic mine valuation problems on the Witwaterstrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52:119–139, 1951.

[41] N. Lam. Spatial interpolation methods: A review. *The American Cartographer*, 10(2):129–149, 1983.

[42] S. Lovejoy, D. Schertzer, Y. Tessier, and H. Gaonach́. Multifractals and resolution-independent remote sensing algorithms: The example of ocean color. *International Journal of Remote Sensing*, 22(7):1191–1234, 2001.

[43] A. MacEachren. *How Maps Work*. The Guilford Press, New York, 1995.

[44] A. Malinverno. Fractals and ocean floor topography: a review and model. In C. Barton and P. LaPointe, editors, *Fractals in the Earth Sciences*. Plenum Press, New York, 1995.

[45] J-C Mareschal. Fractal reconstruction of sea-floor topography. *Pure & Applied Geophysics*, 131:197–210, 1989.

[46] R. Marks. *Introduction to Shannon Sampling and Inerpolation Theory*. Springer-Verlag, New York, 1991.

[47] D. Marr. *Vision*. W. H. Freeman, New York, 1982.

[48] P. Martin. Energy flow: spatial and temporal patterns. *Solstice*, 15(1):http://www–personal.umich.edu/˜sarhaus/image/solstice, 2004.

[49] F. Marvasti. Recovery of signals from nonuniform samples using iterative methods. *IEEE Transactions on Signal Processing*, 39:872–877, 1991.

[50] G. Mulugeta. The elusive nature of expertise in spatial interpolation. *Cartography and Geographic Information Systems*, 25(1):33–41, 1998.

[51] NASA. Huang-Hilbert transform white paper. http://techtransfer.gsfc.gov/HHT/HHT.htm. Accessed May, 2004.

[52] H. Neff. *Continuous and Discrete Linear Systems*. Harper & Row, New York, 1984.

[53] A. Oppenheim and J. Lim. The importance of phase in signals. *Proceedings of the IEEE*, 69(5):529–541, 1981.

[54] A. Papoulis. *Signal Analysis*. McGraw-Hill, New York, 1977.

[55] G. Pasternack. Does the river run wild? Assessing chaos in hydrologic systems. *Advances in Water Resources*, 23:253–260, 1999.

[56] A. Pentland. Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:661–674, 1984.

[57] A. Pentland, E. Simoncelli, and T. Stephenson. Fractal-based image compression and interpolation. United States Patent No. 5,148,497, 1992.

[58] B. Pesquet-Popescu and P. Larzabal. Interpolation of nonstationary fields with stationary increments. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2197–2200, 1998.

[59] B. Pesquet-Popescu and J. Véhel. Stochastic fractal models for image processing. *IEEE Signal Processing Magazine*, 19(5):48–62, 2002.

[60] L. Polidori and J. Chorowicz. Comparison of bilinear and Brownian interpolation for Digital Elevation Models. *ISPRS Journal of Photogrammetry and Remote Sensing.*, 48(2):18–23, 1993.

[61] M. Rauth. Application of 2D methods for scattered data approximation to data sets. In *Proc. Conf. SampTA-95*, http://www.mat.univie.ac.at/~nuhag/papers/PS/1995/rau0695.ps.gz, 1995. NUHAG. Accessed October 2002.

[62] S. Rice. Mathematical analysis of random noise. *Bell System Technical Journal*, 23(3):282–332, 1944. Continued 24(1):46–157, 1945.

[63] C. Shannon. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 37:10–21, 1949.

[64] C. Shannon. Information theory. In N. Sloane and A. Wyner, editors, *Claude Elwood Shannon, Collected Papers*. IEEE Press, Piscataway, NJ, 1993. Originally published in *Encyclopedia Brittanica*, 4th ed., 1968.

[65] C. Shannon and W. Weaver. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1949.

[66] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.

[67] W. Smith and D. Sandwell. Conventional bathymetry, bathymetry from space, and geodetic altimetry. *Oceanography*, 27(1):8–23, 2004.

[68] W. Smith and P. Wessel. Gridding with continuous curvature splines in tension. *Geophysics*, 55:293–305, 1990.

[69] M. Unser. Sampling —50 years after Shannon. *Proceedings of the IEEE*, 88:569–587, 2000.

[70] M. Unser and J. Zerubia. Generalized sampling: Stability and performance analysis. *IEEE Tansactions on Signal Processing*, 45(12):2941–2950, 1997.

[71] M. Unser and J. Zerubia. A generalized sampling theory without band-limiting constraints. *IEEE Transactions on Circuits & Systems II*, 45(8):959–969, 1998.

[72] B. Usevitch. A tutorial on modern lossy wavelet image compression: Foundations of JPEG 2000. *IEEE Signal Processing Magazine*, 18(5):22–35, 2001.

[73] M. Vetterli. Wavelets, approximation, and compression. *IEEE Signal Processing Magazine*, 18(5):59–73, 2001.

[74] T. Werther. Reconstruction from irregular samples with improved locality. Master's thesis, University of Vienna, 1999. http://www.unet.univie.ac.at /~a9301647/publications/mthesis.pdf accessed September, 2003.

[75] A. Whitehead and B. Russell. *Principia Mathematica*. The University Press, Cambridge, U.K., 1927.

[76] C. Wittenbrink. IFS fractal interpolation for 2D and 3D visualization. ftp.cse.ucsc.edu /pub/reinas/papers/ifs_ fractal_ int.pdf. Baskin Center for Computer Engineering and Information Science, University of California, Santa Cruz, accessed October 2002.

[77] Ludwig Wittgenstein. *Tractatus Logico-Philosophicus*. Harcourt, Brace & Company, New York, 1922.

[78] S. Wolfram. *A New Kind of Science*. Wolfram Media, Champaign, Illinois, 2002.

[79] J. Wood and P. Fisher. Assessing interpolation accuracy in elevation models. *Computer Graphics and Applications*, 13(3):48–56, 1993.

[80] N. Yokoya, K. Yamamoto, and N. Funakubo. Fractal-based analysis and interpolation of 3-D natural surface shapes and their application to terrain modeling. *Computer Vision, Graphics, and Image Processing*, 46(3):284–302, 1989.

[81] Peng Yu, S. Morey, and J. Zavala-Hidalgo. New mapping methods to observe propagating features. *Sea Technology*, pages 20–24, May 2004.

APPENDICES

A Transform Sampler

This appendix is intended to provide a brief overview of some well-known transforms, in very general and informal terms. As I tried to show in the first part of this thesis, transforms can be regarded as conversion operators that allow a "signal" to be represented as *a composition of different basis components.* These components can themselves be regarded as signals of an elementary sort. The standard basis is usually a set of unit values within a domain of space (1, 2, or 3 dimensions) or time. *These are the standard basis components.* Any one standard basis component is a signal of an elementary sort, an impulse whose value is 1 at one point in the domain, and 0 everywhere else in the domain. One way to get a handle on transforms is to see what the new basis components are, as represented in the standard basis. Thus every transform can be identified with a set of elementary signals, or component parts, that can be added in various amounts to build any signal representable in the basis associated with that transform, and so every transform affords a new view of what the world of signals is made up of.

Why would one want to transform the basis for representing things anyway? Aside from the philosophical reason that traveling in order to get different viewpoints is better than remaining provincial, the practical reason is that, if you are looking for certain aspects of a signal, you will more likely see those aspects clearly if the signal is represented in terms of those very aspects as elementary building blocks.

In what follows, we see the alternative building blocks of signals for various transforms, always with respect to the familiar standard basis.

A.1 Fourier transform

I prefer to think of the components of the Fourier domain as a family of complex-valued helices of increasing pitch, like tighter and tighter springs. They are the graphs of elementary exponential functions, with an imaginary i to make them go around

faster and faster instead of blowing up. Considering only real-valued numbers, they are sinusoidal waves of increasing frequency. Nature seems to be full of sine waves and exponentials, maybe because they are similar to their own derivatives. Hence the Fourier transform is fundamental to the analysis of any signal that has periodicity or is part of a system with feedback, so that its own value might be related to the rate of change of its value.

A.2 Cosine transform

The cosine transform (or Discrete Cosine Transform, as we have referred to it, since we have been concerned with discrete signals) is similar to the Fourier transform except that it operates in the realm of real numbers only, not complex numbers. The family of building blocks are also sinusoidal waves of increasing frequency, with smaller steps (*viz.* half-cycle) in order to represent the full space of phase possibilities.

The cosine transform is useful when you are interested in periodicity but not the complex exponential aspect. For certain fields of data which can be modeled as quasi-random processes with high correlation between neighbors (as is the case for natural grayscale images, terrain models, etc.), the discrete cosine transform has asymptotic equivalences to a principal component transform [27], which is one reason for its use as the basis for JPEG image compression.

A.3 Principal component transform

Also known as Karhunen-Loève transform (KLT) or Hotelling transform, this is not a specific defined transform, but is a sort of optimum transform that can be determined for a *source*, or class of signals. This is the transform that has the most to do with *context*. It has little meaning to speak of the KLT of one signal in isolation; it has one principal component, and that is itself! The components of the KLT are the eigenvectors of the covariance matrix for the source. These can be visualized in abstract function space as the major, semi-major, etc. orthogonal axes of the

distribution of points that belong to the class of functions.

The utility of the KLT is that it affords the most distinction between class members based on knowledge of the fewest components. Conversely, if the first few principal components of an unknown signal can be determined from minimal samples, reconstruction (*i.e.* interpolation) can be performed efficiently on the basis of class membership (*i.e.* context).

A.4 Hilbert transform

The Hilbert transform is the means to represent a real-valued signal as a complex-valued ("analytic") signal. It is effectively the addition, to the real-valued signal, a pure imaginary-valued signal that is 90 degrees out of phase for all frequency components. This is the same as setting all negative frequency components of the original real-valued signal to zero.

The reason for doing so is that this higher-dimension view of the signal simplifies analysis by making certain quantities appear to be more constant —allowing, for example, simple access to instantaneous frequency and instantaneous amplitude.

A.5 Huang-Hilbert transform

This transform operates adaptively on a signal, ultimately representing it in terms of *Intrinsic Mode Function* [51] components of constant amplitude and various numbers of cycles, to which the Hilbert transform is applied in order to obtain analytic-signal representations of the Intrinsic Mode Functions.

The transform proceeds by an algorithm called *Empirical Mode Analysis*, which is an iterative process of subtracting the signal *envelope* and *mean* until arriving at a zero-mean signal of constant amplitude, which is an *Intrinsic Mode Function*. This is then subtracted from the original signal, and the whole process is repeated to arrive at the next Intrinsic Mode Function. The last Intrinsic Mode Function has only one

zero crossing, and the collected set of Intrinsic Mode Functions forms the new basis for representing the signal.

A.6 Radon transform

Applied to two-dimensional data, the Radon transform effectively uses *lines* as the building blocks of images. Since a line in the x-y plane can be specified by a direction and a distance from the origin, the component lines can be represented as points at a certain distance and in a certain direction from the origin, with z value (such as intensity) corresponding to the coefficient for that component. As an example of the idea suggested above, that if you are looking for certain aspects of a signal, you will more likely see those aspects clearly if the signal is represented in terms of those very aspects as elementary building blocks, this transform to a basis of lines facilitates detection and analysis of linear features in the data.

B Information Impedance Matching

In electrical and electronics engineering, there is a principle called "impedance matching". The basic idea is that, for most effective transfer of energy from one part of a circuit to another part, the parts must have certain characteristics matched in a complementary way.[1] For example, if you want to produce as much light as possible with a certain battery and an incandescent bulb you should use a bulb not of extremely high resistance nor of extremely low resistance, but of resistance equal to the battery's (internal) resistance —the filament impedance is matched to the battery impedance. Likewise in the communication of information, say a grid of data such as an image, the best representation of the information occurs when the data are displayed at the resolution indicated by the source —that is, one pixel to one pixel. Thus in loose analogy to the principle of electrical impedance matching I propose the principle of information impedance matching, to describe the optimal matching of an information source and an information receiver.

This analogy seems especially appropriate in light of the fact that the fields of information theory and signal processing implicitly recognize the equivalence of energy and information when speaking of "energy compaction" in transform coding, etc. Thus the 2-norm of a signal, as a measure of its distance from the origin of function space, is called its energy; "energy compaction" refers to use of a transform to arrive at a coordinate system in which the location of the signal in function space can be given by a few cardinal directions of the coordinate system. Then the *projection* of the signal onto those axes has high energy as well.[2]

Energy is always defined relative to a frame of reference, *i.e.* a coordinate system, and the same is true of information. In this thesis, we tried to pay attention to signal

[1] The impedances should be complex conjugates at the operating frequency, meaning that resistances should be equal and the capacitive reactance of one should equal the inductive reactance of the other.

[2] The sum of mutually-exclusive projection energies should equal the full-dimension energy.

context, suggesting transforms that would result in coordinate systems wherein the total energy of the class of signals would be minimized. Then low-energy signals (those with small 2-norms) would be those that conform to the class —the sort of signal that is sought for interpolation. All this sometimes seems too abstract for practical application, but I think that even subconscious awareness of such connections can help prevent errors in geographic information science.

Cartographers in particular are concerned with what might be called the effective transfer of spatial information, which I think depends on attention to information impedance matching in data *collection*, data *conversion*, and data *representation*. The risks of inattendance to information impedance matching are information loss, pseudo-information generation, and loss of efficiency.

In data collection, a well-known rule is not to record more apparent significant digits of a numerical measurement than are justified by the precision of the instrument or by other considerations of maximum possible precision. Ignorance of this rule results in an information impedance mismatch insofar as much of the flow of numbers conveyed is overburden, not representing information.

In data conversion, any sort of re-sampling or re-projection of data likely constitutes an information impedance mismatch. Re-projection generally entails a special application of interpolation of a regular grid, where original data are discarded and new data are created. It is inevitable that information will be lost and pseudo-information created in this process; it is just a question of the severity of the mismatch for the information that is of most interest: some spatial patterns will be affected in different ways than others by the mismatch.[3] For example, even if a grid of data is simply "reprojected" to a grid of coarser resolution, it may in some cases be preferable simply to subsample; in other cases it may be preferable to use a convolution filter,

[3]This is reminiscent of the frequency selectivity of reactive matches in electronics.

as discussed in the sections on super-resolution and detection of scale-wise correlation for fractal analysis and synthesis. Or again, in the familiar case of image size reduction, it will be found that subsampling might preserve "sharpness" of certain features, while resampling with a convolution filter may better display continuity of areas and of edges not aligned with the grid axes.

A form of pseudo-information that might arise in resampling is *aliasing*, discussed in the body of this thesis. Kimerling [39] reported on the Moiré-like patterns apparent in data quality maps of resampled equal-angle grids. Information impedance mismatching can produce similar patterns (or similarly-caused patterns) in the presentation of *the data itself*, which should be of considerable concern to those who prepare and analyze spatial information. Figure B.1 is a pair of images of the 256^2 discrete cosine transform matrix. The image on the right was sized down at one point along the line and sized back up at another point, which is expected to produce subsampling discontinuities. Displayed properly, the images would reveal hyperbolic bands bending toward the upper left corner, and a fainter hyperbolic cross at 2/3 across and 2/3 down from upper left.[4] Any other variations seen are aliasing artifacts that result from information impedance mismatching. If you are viewing this document on a computer screen, try changing the magnification, and see if the patterns change.

These DCT matrix representations have undergone several conversions, including 32-bit floating point number to 8-bit integer, resizing, raster to vector, vector to raster, all while being accepted as pictures of the matrix, rather than as pictures of conversions of representations of... etc.

Because of the profusion of electronically-manipulated spatial data and the demands to reformat data sets for compatibility, it is incumbent upon those who work in geographic information science to be consciously aware of the distinction between

[4]My understanding of this cross is that it is a "legitimate" aliasing effect of discrete signal analysis, related to the predominance of the 5^{th} as a music interval (frequency ratio 2:3).

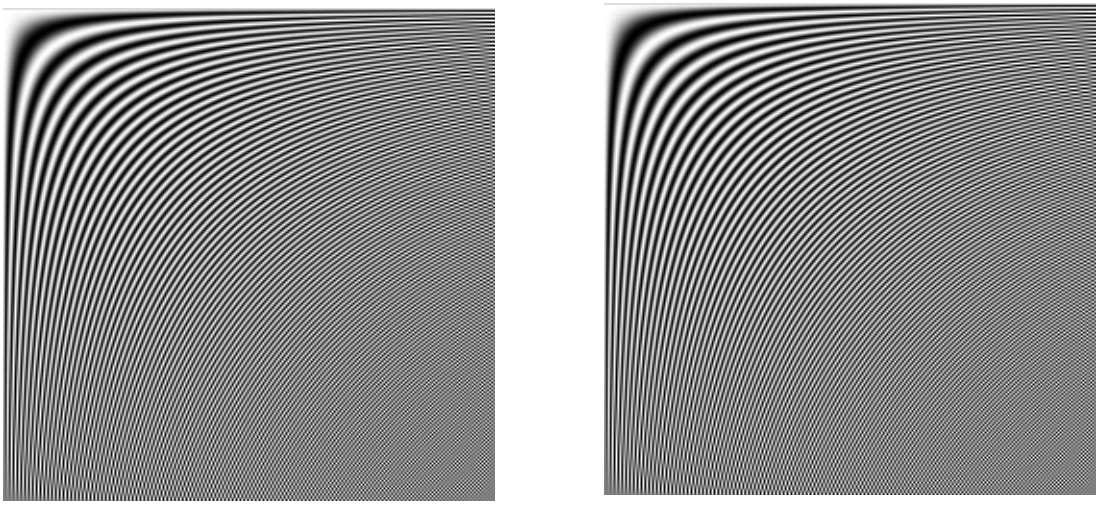


Figure B.1: 256^2 DCT matrix. Reduced and enlarged image on right.

the signal and its representation, and to minimize conversion and representation mismatches.

Loss of efficiency in information transmission matters because when efficiency drops, so does communication —consider for example a cluttered map, or a web page that is slow to load. Any representation of information is a sort of symbol. I think the key to avoiding information impedance mismatch is to have a sense of what are the essential information components of a set of data, and to employ representations that encode information in similar terms.[5] For example, graphs comprising lines and plots are not efficiently represented by the JPEG image format, whose components are smooth waves; such information would be represented more efficiently in vector format, such as EPS (Encapsulated PostScript), or if they must be in raster format for compatibility, then GIF (Graphics Interchange Format), PNG (Portable Network Graphics) or compressed TIFF (Tagged Image File Format) might be a better choice —the ratio of display quality to file size would be much higher.

[5]Wittgenstein [77] claimed that symbols must have something in common with that which they represent.

Information impedance matching applies to non-spatial data as well. A common example of an information impedance mismatch that results in loss of efficiency is the conversion of text to image.[6] In the parlance of this thesis, this is a projection from one signal space to a much higher dimension signal space. Conversely, when numerical data are encoded as ASCII (American Standard Code for Information Interchange) characters, a double-conversion has taken place, with resulting loss of efficiency. A case in point is the passage of information electronically over the Internet, as in the case of a map server or other geographic data server. The price paid for a poor choice of data format is slow transfer of data. More importantly, data are not necessarily information, and it is only information that really needs to be served.

Information impedance matching can be summarized as facilitating information flow by making appropriate joints and transmission lines between information source and information interpreter. Modular thinking can't be discarded, but geographic information science practitioners must take the responsibility to understand the so-called "transparent" processes (otherwise known as "black boxes") that affect their geographic information and its effective communication.

[6] *E.g.* the 1.2 MB PDF email attachment including high-resolution scanned graphics of a hand-lettered advert for next Friday's party.

C Nonlinear Series Analysis

In this thesis we have encountered *algorithmic* information theory, which quantifies the information content of a signal in terms of the smallest program that could generate the signal as output. I see a connection to currently popular nonlinear time series analysis, which is most aptly applied to systems which have few degrees of freedom (*i.e.* few controlling factors), but whose output is an apparently "chaotic" signal. So these signals would have low algorithmic information content.

The essence of nonlinear time series analysis is to transform a system's observed signal into a trajectory in *phase space*. The dimensions of phase space are the degrees of freedom of the system. The system trajectory in phase space may be a sort of orbit of fractional dimension that can be viewed in a higher-dimension of phase space, allowing prediction and even control that would seem impossible otherwise. If there is better access to one output signal of a system than to another, the first may be taken as a proxy for the second, allowing prediction and interpolation with results superior to those obtained by statistical methods of correlation. [1]

Nonlinear time series analysis is not the panacea for analysis of signals that do not yield to other methods. Pasternack [55] has shown that it is probably not applicable to hydrologic systems —*i.e.* times series of stream discharge[7]— nor is it *inapplicable* to systems whose observed output is *not* "chaotic". But it is possible that for some data it would be applicable to "space series", and might prove to be another tool for interpolation and extrapolation of spatial data.

The trick with which I am familiar, used for "embedding" a signal in phase space, is to take time-delayed observations of the signal as projections of phase-space co-ordinates. In general the shortest time delay is chosen for which the signal values are most decorrelated, and then the dimension of the embedding space is determined by finding the dimension in which points which appear to be neighbors really are

[7] I found it inapplicable to a 40-year record of Eagle Creek stream flow.

neighbors in the series.

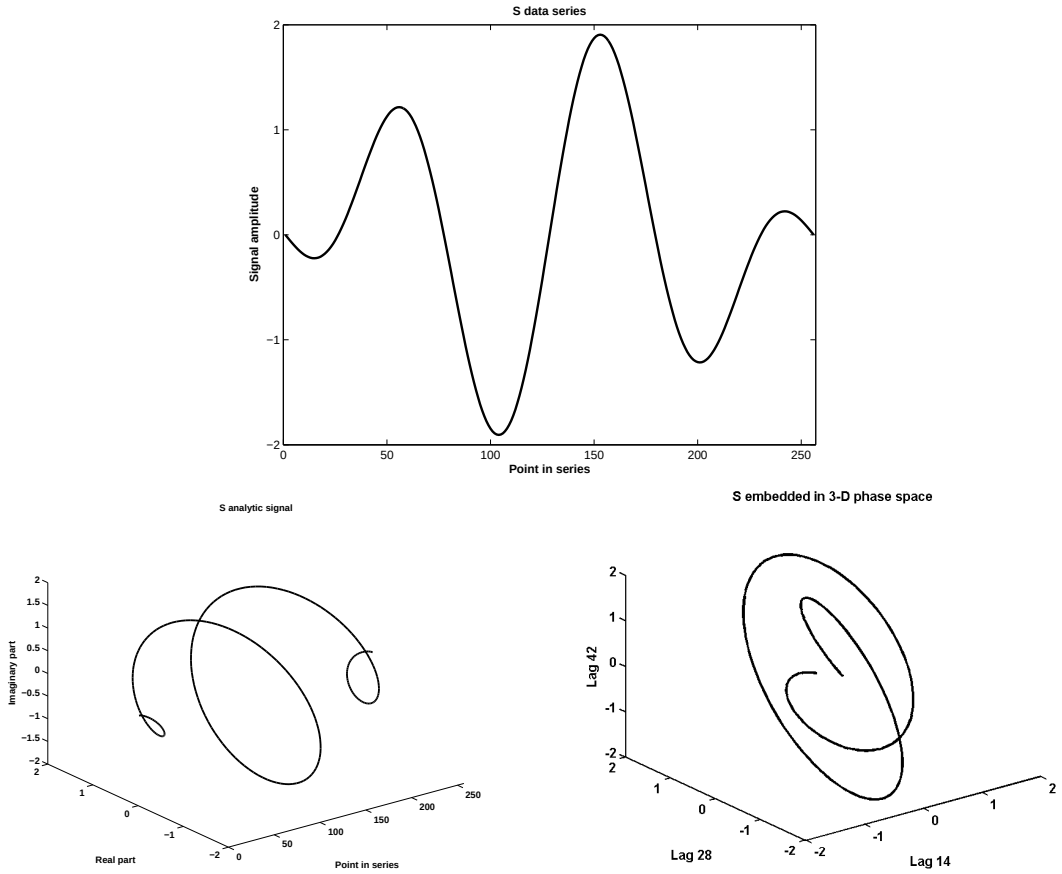


Figure C.1:Signal S (top, sum of two sines), its analytic signal(left), and its inferred phase-space trajectory(right).

Figure C.1 shows a "time series" signal that is is *not* chaotic, along with the analytic signal representation (embedded in three dimensions) and a representation of its phase space trajectory embedded in three dimensions. Here I chose a lag of 14, rather than the minimum lag for decorrelation (lag 2 was indicated as the first minimum of the "auto-mutual information" function, often used for determining appropriate lag), because the plot of the orbit was more open with that choice. This figure illustrates that conversion to phase space representation is a case of signal projection to greater

dimension, analogous to other transforms.

The object of it all is to resolve the apparently irregular and unpredictable series into a trajectory of an orbit (called an 'attractor" in chaos theory) in phase space. Thus I see the process in analogy to use of the analytic signal to view a signal, oscillating in one dimension, as an analytic signal, orbiting in two dimensions. The conceptual basis that we've developed for understanding transforms and projections in function space are helpful in understanding nonlinear time series analysis.

D Software List

This is a list of software resources that I used in my research.

Matlab

http://www.mathworks.com

Matlab was used for all the computations and to produce all of the figures in this work. The various "toolboxes" are add-ons to Matlab, although they may work with SciLab, a free alternative to Matlab offered by the same good people at INRIA who offer FracLab and the Time-Frequency toolbox, http://scilabsoft.inria.fr/

PyrTools

http://www.cns.nyu.edu/~eero/software.html

Eero Simoncelli's toolbox was my practical introduction to pyramid image decomposition. There are many useful routines, and many are compiled to run fast. The included "demo" (tutorial) is free education.

Open TStool

http://www.physik3.gwdg.de/tstool/

There is variation is the quality of Matlab toolboxes, especially those that provide graphical user interface. This is one of the best. It uses the full power of Matlab without generating excessive baggage, and what is more, it is a very good hands-on introduction to nonlinear time series analysis.

Time-Frequency toolbox

http://crttsn.univ-nantes.fr/~auger/tftb.html

One of the best parts of this package is the documentation, including a tutorial that is textbook quality. The free education and resources available by Internet,

especially from France, is amazing.

WaveLab

http://www-stat.stanford.edu/~wavelab

This is a free alternative to the Matlab wavelet toolbox. In some ways it is more extensive, and there are compiled routines and source code.

MikTex

http://www.miktex.org

This is the free version of LaTeX typesetting software that I used to compile the thesis document. It is good for creating documents with a lot of math symbols and vector-type figures. I used WinEdt as an editor for composing LaTeX code.

WinEdt

http://www.winedt.com

This is a nice inexpensive Windows editor that interfaces richly with LaTeX. In fact it includes sample LaTeX and BibTeX (for bibliographies) documents and templates, including a Dalhousie University doctoral thesis.

Additional Matlab toolboxes

http://stommell.tamu.edu/~baum/toolboxes.html

This is a page of links to many other Matlab toolboxes that are available free.

E Program source code

This is a collection of some of the programs that I wrote or modified for this thesis work. Probably the most useful or noteworthy are FFTINTERP, FINV, FLTI, and the pyramid super-resolution routines EXTENDLPYR and EXLPYR3. The analogous routines for wavelet-based super-resolution, EXTENDWAVDET, etc., may be of interest as well, along with DWTUP for texturing by not subsampling wavelet coefficients.

E.1 C code for mex files

exLpyr.c

```
/*==========================================================
 * exLpyr.c
 *
 * This is a MEX-file for MATLAB.
 *
 * This is a mex file for generating the next-higher-resolution
 * Laplacian image, called L_0, given L_1 and L_2, based on a
 * search for pixel-block similarity. It is thus part of a scheme
 * for "fractal-based super-resolution", or interpolation based on
 * the assumption of scale-wise self-similarity.
 *
 * It is based on an algorithm described in U.S. Patent 1,148,497,
 * 1992, by Pentland, Simoncelli, and Stephenson, and was created
 * to be used in conjunction with other pyramidal decomposition
 * routines contained in the "PyrTools" Matlab toolbox of Eero
 * Simoncelli.
 *
 * The function takes two real double arrays and does an
 * element-by-element norm comparison of nine-pixel neighborhoods
 * for each interior element of L_1 and L_2, and when the best
 * match is found, the new 4-pixel block of L_0 corresponding to
 * the one pixel of L_1 is generated in accordance withthe 4-pixel
 * block of L_1 that corresponds to the one pixel of L_2 with
 * 9-pixel neighborhood most similar to 9-pixel neighborhood of L_1.
 *
 * Because this program has no provision to accommodate edges, the
 * resulting L_0 has a 2-pixel border of zeros.
```

```
 *
 *
 * Pete Martin
 * pmartin@ieee.org
 *
 *=========================================================*/
 /* $Revision: 1.1 $ */

#include "mex.h" #include <math.h>

void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray
*prhs[]) {

 const int  *dims_first, *dims_second;
 int c,  ncolsIm1, ncolsIm2, j, k, m, n, r, w;
 double inf, normDiff, leastDiff, *Im1_ptr, *Im2_ptr, *out_ptr;
 int nrowsIm1, nrowsIm2;

 /* Check for proper number of input and output arguments */
 if (nrhs != 2) {
 mexErrMsgTxt("Two input arguments required.");
 }
 if (nlhs > 1) {
 mexErrMsgTxt("Too many output arguments.");
 }

 /* Check data type of first input argument */
 if (!mxIsDouble(prhs[0]) || !mxIsDouble(prhs[1])
 || mxIsComplex(prhs[0]) ||mxIsComplex(prhs[1])) {
 mexErrMsgTxt("Input arguments must be real of type double.");
 }

 /* Check that dimensions are the same for input arguments. */
 if ( mxGetNumberOfDimensions(prhs[0]) != 2 ...
     || mxGetNumberOfDimensions(prhs[1])!= 2){
 mexErrMsgTxt("Inputs must 2-dimensional matrices.\n");
 }

 dims_first = mxGetDimensions(prhs[0]);
 dims_second = mxGetDimensions(prhs[1]);
```

```
/* Check relative sizes of inputs. */
for (c=0; c<mxGetNumberOfDimensions(prhs[0]); c++){
if (dims_first[c]!= 2*dims_second[c]){
mexPrintf...
 ("First argument was expected to be twice the size of...
  second argument...\n");
}
}

/* Set "inf", to be used as first value for for normDiff. */
inf = mxGetInf();

/* Get the row and column counts for the input matrices. */
nrowsIm1 = dims_first[0];
ncolsIm1 = dims_first[1];
nrowsIm2 = dims_second[0];
ncolsIm2 = dims_second[1];


/* Get the input values */
Im1_ptr = (double *)mxGetPr(prhs[0]);
Im2_ptr = (double *)mxGetPr(prhs[1]);

/* Create output and get its input values */
plhs[0]=mxCreateDoubleMatrix(2 * dims_first[0],2 * ...
dims_first[1],mxREAL);
out_ptr = mxGetPr(plhs[0]);

/* For every interior pixel of Im1, search for the interior
* pixel of Im2 with the best 9-pixel neighborhood match,
* as measured by the squared Eucliean difference, called
* "normDiff". After all neighborhoods have been checked,
* write the four Im0 pixels same as the Im1 4-pixel group
* matched that of the Im1 pixel being tested at that point.

* "m" is the Im2 row index; "n" is the Im2 column index;
* "w" is the output matrix pointer index (to write);     */

for(j=2; j<nrowsIm1; j++){
 for(k=2; k<ncolsIm1; k++){
  leastDiff = inf;
```

```
for(m=2; m<nrowsIm2; m++){
 for(n=2; n<ncolsIm2; n++){
  normDiff =
    (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[nrowsIm2 ...
     * (n-2) + (m-1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-2) + (j-1) -1] - Im2_ptr[nrowsIm2 * (n-2) + (m-1) -1])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[nrowsIm2 ...
     * (n-2) + (m) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-2) + (j) -1] - Im2_ptr[nrowsIm2 * (n-2) + (m) -1])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nrowsIm2 ...
     * (n-2) + (m+1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-2) + (j+1) -1] - Im2_ptr[nrowsIm2 * (n-2) + (m+1) -1])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[nrowsIm2 ...
     * (n-1) + (m-1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-1) + (j-1) -1] - Im2_ptr[nrowsIm2 * (n-1) + (m-1) -1])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[nrowsIm2 ...
     * (n-1) + (m) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-1) + (j) -1] - Im2_ptr[nrowsIm2 * (n-1) + (m) -1])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[nrowsIm2 ...
     * (n-1) + (m+1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-1) + (j+1) -1] - Im2_ptr[nrowsIm2 * (n-1) + (m+1) -1])
 + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nrowsIm2 ...
     * (n) + (m-1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k) + (j-1) -1] - Im2_ptr[nrowsIm2 * (n) + (m-1) -1])
 + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[nrowsIm2 ...
     * (n) + (m) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k) + (j) -1] - Im2_ptr[nrowsIm2 * (n) + (m) -1])
 + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nrowsIm2 ...
     * (n) + (m+1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k) + (j+1) -1] - Im2_ptr[nrowsIm2 * (n) + (m+1) -1]);
       if(normDiff < leastDiff){
       leastDiff = normDiff;
       w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
       r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

       out_ptr[w] = Im1_ptr[r];
       out_ptr[w+1] = Im1_ptr[r+1];
       out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r + nrowsIm1];
       out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r + nrowsIm1 + 1];
      }
   }
```

```
    }
  }
 }
}
```

exLpyr3.c

```
/*==================================================================
 * exLpyr3.c
 *
 * This is a MEX-file for MATLAB.
 *
 * This is a mex file for generating the next-higher-resolution
 * Laplacian image, called L_0, given L_1, L_2, and L_3, based on a
 * search for pixel-block similarity. It is thus part of a scheme
 * for "fractal-based super-resolution", or interpolation based on
 * the assumption of scale-wise self-similarity. It is a second-
 * stage extension of the one-stage routine "extendlpyr", in effect
 * using the same "lookup table" generated from L2 and L3.
 *
 * It is based on an algorithm described in U.S. Patent 1,148,497,
 * 1992, by Pentland, Simoncelli, and Stephenson, and was created
 * to be used in conjunction with other pyramidal decomposition
 * routines contained in the "PyrTools" Matlab toolbox of Eero
 * Simoncelli.
 *
 * The function takes two real double arrays and does an
 * element-by-element norm comparison of nine-pixel neighborhoods
 * for each interior element of L_1 and L_3, and when the best
 * match is found, the new 4-pixel block of L_0 corresponding to
 * the one pixel of L_1 is generated in accordance with the 4-pixel
 * block of L_2 that corresponds to the one pixel of L_3 with
 * 9-pixel neighborhood most similar to 9-pixel neighborhood of L_1.
 *
 * Because this program has no provision to accommodate edges, the
 * resulting L_0 has a 2-pixel border of zeros.
 *
 *
```

```
 * Pete Martin
 * pmartin@ieee.org
 *
 *================================================================*/
 /* $Revision: 1.1  May 2004$ */

#include "mex.h" #include <math.h>

void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray
*prhs[]) {

    const int  *dims_first, *dims_second, *dims_third;
    int c,  ncolsIm1, ncolsIm2, ncolsIm3, j, k, m, n, r, w;
    double inf, normDiff, leastDiff, *Im1_ptr, *Im2_ptr, *Im3_ptr,
    *out_ptr;
    int nrowsIm1, nrowsIm2, nrowsIm3;

    /* Check for proper number of input and output arguments */
    if (nrhs != 3) {
    mexErrMsgTxt("Three input arguments required.");
    }
    if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments.");
    }

    /* Check data type of first input argument */
    if (!mxIsDouble(prhs[0]) || !mxIsDouble(prhs[1])
    || mxIsComplex(prhs[0]) ||mxIsComplex(prhs[1])
    || mxIsComplex(prhs[2]) || !mxIsDouble(prhs[2])) {
    mexErrMsgTxt("Input arguments must be real of type double.");
    }

    /* Check that dimensions are the same for input arguments. */
    if ( mxGetNumberOfDimensions(prhs[0]) != 2 || ...
     mxGetNumberOfDimensions(prhs[1])!= 2 || ...
     mxGetNumberOfDimensions(prhs[2])!= 2){
    mexErrMsgTxt("Inputs must 2-dimensional matrices.\n");
    }

    dims_first = mxGetDimensions(prhs[0]);
    dims_second = mxGetDimensions(prhs[1]);
```

```
dims_third = mxGetDimensions(prhs[2]);

/* Check relative sizes of inputs. */
for (c=0; c<mxGetNumberOfDimensions(prhs[0]); c++){
if (dims_second[c]!= 2*dims_third[c]){
mexPrintf ...
("Second argument was expected to be twice the size of...
 third argument...\n");
}
}

/* Set "inf", to be used as first value for for normDiff. */
inf = mxGetInf();

/* Get the row and column counts for the input matrices. */
nrowsIm1 = dims_first[0];
ncolsIm1 = dims_first[1];
nrowsIm2 = dims_second[0];
ncolsIm2 = dims_second[1];
nrowsIm3 = dims_third[0];
ncolsIm3 = dims_third[1];


/* Get the input values */
Im1_ptr = (double *)mxGetPr(prhs[0]);
Im2_ptr = (double *)mxGetPr(prhs[1]);
Im3_ptr = (double *)mxGetPr(prhs[2]);

/* Create first output and get its input values */
plhs[0]=mxCreateDoubleMatrix(2 * dims_first[0],2 * ...
dims_first[1],mxREAL);
out_ptr = mxGetPr(plhs[0]);


/* For every interior pixel of Im1, search for the interior
* pixel of Im3 with the best 9-pixel neighborhood match, as
* measured by the squared Eucliean difference, called
* "normDiff". Whenever a better match is found, write the
* four output pixels same as the Im2 4-pixel group at the
* same relative location as the single Im3 pixel that matched
* that of the Im1 pixel being tested at that point.
```

```
    * "m" is the Im3 row index; "n" is the Im3 column index;
    * "w" is the output matrix pointer index (to write);     */

for(j=2; j<nrowsIm1; j++){
 for(k=2; k<ncolsIm1; k++){
  leastDiff = inf;
   for(m=2; m<nrowsIm3; m++){
    for(n=2; n<ncolsIm3; n++){
     normDiff =
    (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im3_ptr[nrowsIm3 ...
     * (n-2) + (m-1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-2) + (j-1) -1] - Im3_ptr[nrowsIm3 * (n-2) + (m-1) -1])
  + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im3_ptr[nrowsIm3 ...
     * (n-2) + (m) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-2) + (j) -1] - Im3_ptr[nrowsIm3 * (n-2) + (m) -1])
  + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im3_ptr[nrowsIm3 ...
     * (n-2) + (m+1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-2) + (j+1) -1] - Im3_ptr[nrowsIm3 * (n-2) + (m+1) -1])
  + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im3_ptr[nrowsIm3 ...
     * (n-1) + (m-1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-1) + (j-1) -1] - Im3_ptr[nrowsIm3 * (n-1) + (m-1) -1])
  + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im3_ptr[nrowsIm3 ...
     * (n-1) + (m) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-1) + (j) -1] - Im3_ptr[nrowsIm3 * (n-1) + (m) -1])
  + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im3_ptr[nrowsIm3 ...
     * (n-1) + (m+1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k-1) + (j+1) -1] - Im3_ptr[nrowsIm3 * (n-1) + (m+1) -1])
  + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im3_ptr[nrowsIm3 ...
     * (n) + (m-1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k) + (j-1) -1] - Im3_ptr[nrowsIm3 * (n) + (m-1) -1])
  + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im3_ptr[nrowsIm3 ...
     * (n) + (m) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k) + (j) -1] - Im3_ptr[nrowsIm3 * (n) + (m) -1])
  + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im3_ptr[nrowsIm3 ...
     * (n) + (m+1) -1]) * (Im1_ptr[nrowsIm1 ...
     * (k) + (j+1) -1] - Im3_ptr[nrowsIm3 * (n) + (m+1) -1]);
     if(normDiff < leastDiff){
        leastDiff = normDiff;
        w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
        r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;
```

```
        out_ptr[w] = Im2_ptr[r];
        out_ptr[w+1] = Im2_ptr[r+1];
        out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + nrowsIm2];
        out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + nrowsIm2 + 1];
     }
    }
   }
  }
 }
}
```

exLpyr8way.c

```
/*==================================================================
 * exLpyr8way.c
 *
 * This is a MEX-file for MATLAB.
 *
 * This is a mex file for generating the next-higher-resolution
 * Laplacian image, called L_0, given L_1 and L_2, based on a
 * search for pixel-block similarity. It is thus part of a scheme
 * for "fractal-based super-resolution", or interpolation based on
 * the assumption of scale-wise self-similarity.
 *
 * It is based on an algorithm described in U.S. Patent 1,148,497,
 * 1992, by Pentland, Simoncelli, and Stephenson, and was created
 * to be used in conjunction with other pyramidal decomposition
 * routines contained in the "PyrTools" Matlab toolbox of Eero
 * Simoncelli.
 *
 * The function takes two real double arrays and does an
 * element-by-element norm comparison of nine-pixel neighborhoods
 * for each interior element of L_1 and L_2, and when the best
 * match is found, the new 4-pixel block of L_0 corresponding to
 * the one pixel of L_1 is generated in accordance withthe 4-pixel
 * block of L_1 that corresponds to the one pixel of L_2 with
 * 9-pixel neighborhood most similar to 9-pixel neighborhood of L_1.
 *
```

```
 * Differing from the function "exLpyr", this function checks
 * all 8 rotations and reflections of the 9-pixel groups, to find
 * the best match.
 *
 * Because this program has no provision to accommodate edges, the
 * resulting L_0 has a 2-pixel border of zeros.
 *
 * This program produces poor results and is slow.
 *
 * Pete Martin
 * pmartin@ieee.org
 *
 *=============================================================*/
 /* $Revision: 1.1 $ */

#include "mex.h" #include <math.h>

void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray
*prhs[]) {

    const int  *dims_first, *dims_second;
    int c,  ncolsIm1, ncolsIm2, j, k, m, n, r, w;
    double inf, normDiff, leastDiff, *Im1_ptr, *Im2_ptr, *out_ptr;
    int nrowsIm1, nrowsIm2, ne, no, nw, we, ce, ea, sw, so, se;

    /* Check for proper number of input and output arguments */
    if (nrhs != 2) {
    mexErrMsgTxt("Two input arguments required.");
    }
    if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments.");
    }

    /* Check data type of first input argument */
    if (!mxIsDouble(prhs[0]) || !mxIsDouble(prhs[1])
    || mxIsComplex(prhs[0]) ||mxIsComplex(prhs[1])) {
    mexErrMsgTxt("Input arguments must be real of type double.");
    }

    /* Check that dimensions are the same for input arguments. */
    if ( mxGetNumberOfDimensions(prhs[0]) != 2 || ...
```

```
mxGetNumberOfDimensions(prhs[1])!= 2){
mexErrMsgTxt("Inputs must 2-dimensional matrices.\n");
}

dims_first = mxGetDimensions(prhs[0]);
dims_second = mxGetDimensions(prhs[1]);

/* Check relative sizes of inputs. */
for (c=0; c<mxGetNumberOfDimensions(prhs[0]); c++){
if (dims_first[c]!= 2*dims_second[c]){
mexPrintf...
 ("First argument was expected to be twice the size of ...
 second argument...\n");
}
}

/* Set "inf", to be used as first value for for normDiff. */
inf = mxGetInf();

/* Get the row and column counts for the input matrices. */
nrowsIm1 = dims_first[0];
ncolsIm1 = dims_first[1];
nrowsIm2 = dims_second[0];
ncolsIm2 = dims_second[1];


/* Get the input values */
Im1_ptr = (double *)mxGetPr(prhs[0]);
Im2_ptr = (double *)mxGetPr(prhs[1]);

/* Create output and get its input values */
plhs[0]=mxCreateDoubleMatrix(2 * dims_first[0],2 * ...
dims_first[1],mxREAL);
out_ptr = mxGetPr(plhs[0]);

/* For every interior pixel of Im1, search for the interior
* pixel of Im2 with the best 9-pixel neighborhood match, as
* measured by the squared Eucliean difference, called
* "normDiff". After all neighborhoods have been checked,
* write the four Im0 pixels same as the Im1 4-pixel group
* matched that of the Im1 pixel being tested at that point.
```

```
* "m" is the Im2 row index; "n" is the Im2 column index;
* "w" is the output matrix pointer index (to write);    */

for(j=2; j<nrowsIm1; j++){
    for(k=2; k<ncolsIm1; k++){
    leastDiff = inf;
        for(m=2; m<nrowsIm2; m++){
            for(n=2; n<ncolsIm2; n++){
            nw = nrowsIm2 * (n-2) + (m-1) -1;
            we = nrowsIm2 * (n-2) + (m) -1;
            sw = nrowsIm2 * (n-2) + (m+1) -1;
            no = nrowsIm2 * (n-1) + (m-1) -1;
            ce = nrowsIm2 * (n-1) + (m) -1;
            so = nrowsIm2 * (n-1) + (m+1) -1;
            ne = nrowsIm2 * (n) + (m-1) -1;
            ea = nrowsIm2 * (n) + (m) -1;
            se = nrowsIm2 * (n) + (m+1) -1;

 /* Case 1: North on top */

 normDiff =
 (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[nw]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no]) ...
  * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
  * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so]) ...
  * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne]) ...
  * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea]) ...
  * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[se]) ...
  * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[se]);
    if(normDiff < leastDiff){
```

```
        leastDiff = normDiff;
        w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
        r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

        out_ptr[w] = Im1_ptr[r];
        out_ptr[w+1] = Im1_ptr[r+1];
        out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r + nrowsIm1];
        out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r + nrowsIm1 + 1];
         }

  /* Case 2: East on top */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[no]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[so]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]);
      if(normDiff < leastDiff){
         leastDiff = normDiff;
         w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
         r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

         out_ptr[w] = Im1_ptr[r+nrowsIm1];
         out_ptr[w+1] = Im1_ptr[r];
         out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r + nrowsIm1 + 1];
         out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r+1];
         }
```

```
   /* Case 3: South on top */

   normDiff =
   (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[ne]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[ne])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no])
 + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[sw]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[sw])
 + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we])
 + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]);
       if(normDiff < leastDiff){
         leastDiff = normDiff;
         w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
         r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

         out_ptr[w] = Im1_ptr[r+nrowsIm1+1];
         out_ptr[w+1] = Im1_ptr[r+nrowsIm1];
         out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r+1];
         out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r];
         }

   /* Case 4: West on top */

   normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se]) ...
```

```
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]);
      if(normDiff < leastDiff){
         leastDiff = normDiff;
         w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
         r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

         out_ptr[w] = Im1_ptr[r+1];
         out_ptr[w+1] = Im1_ptr[r+nrowsIm1+1];
         out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r];
         out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r+nrowsIm1];
         }

   /* Case 5: North on top, mirror image */

    normDiff =
    (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne]) ...
     * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne])
  + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea]) ...
     * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea])
  + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se]) ...
     * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se])
  + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no]) ...
     * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no])
  + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
     * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
  + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so]) ...
     * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so])
  + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw]) ...
     * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw])
```

```
 + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we])
 + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]);
      if(normDiff < leastDiff){
      leastDiff = normDiff;
      w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
      r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

      out_ptr[w] = Im1_ptr[r+nrowsIm1];
      out_ptr[w+1] = Im1_ptr[r+nrowsIm1+1];
      out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r];
      out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r+1];
      }

  /* Case 6: East on top, mirror image */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]);
     if(normDiff < leastDiff){
        leastDiff = normDiff;
        w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
        r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;
```

```
        out_ptr[w] = Im1_ptr[r+nrowsIm1+1];
        out_ptr[w+1] = Im1_ptr[r+1];
        out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r+nrowsIm1];
        out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r];
        }

  /* Case 7: South on top, mirror image */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]);
      if(normDiff < leastDiff){
         leastDiff = normDiff;
         w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
         r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

         out_ptr[w] = Im1_ptr[r+1];
         out_ptr[w+1] = Im1_ptr[r];
         out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r+nrowsIm1+1];
         out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r+nrowsIm1];
         }

   /* Case 8: West on top, mirror image */

   normDiff =
```

```
      (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw]) ...
       * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw])
    + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so]) ...
       * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so])
    + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se]) ...
       * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se])
    + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we]) ...
       * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we])
    + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
       * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
    + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea]) ...
       * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea])
    + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw]) ...
       * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw])
    + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no]) ...
       * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no])
    + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]) ...
       * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]);
         if(normDiff < leastDiff){
           leastDiff = normDiff;
           w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
           r = 2 * nrowsIm2 * (2 * n - 2) + 2 * m -1 -1;

           out_ptr[w] = Im1_ptr[r];
           out_ptr[w+1] = Im1_ptr[r+nrowsIm1];
           out_ptr[w + 2 * nrowsIm1] = Im1_ptr[r+1];
           out_ptr[w + 2 * nrowsIm1 + 1] = Im1_ptr[r+nrowsIm1+1];
    }
   }
  }
 }
}
```

exLpyr8way3.c

```
/*============================================================
 * exLpyr8way3.c
 *
 * This is a MEX-file for MATLAB.
```

```
 *
 * This is a mex file for generating the next-higher-resolution
 * Laplacian image, called L_0, given L_1, L_2, and L_3, based
 * on a search for pixel-block similarity. It is thus part of
 * a scheme for "fractal-based super-resolution", or
 * interpolation based on the assumption of scale-wise self-
 * similarity. It is a second-stage extension of the one-stage
 * routine "extendlpyr", in effect using the same "lookup
 * table" generated from L2 and L3.
 *
 * It is based on an algorithm described in U.S. Patent
 * 1,148,497, 1992, by Pentland, Simoncelli, and Stephenson,
 * and was created to be used in conjunction with other
 * pyramidal decomposition routines contained in the "PyrTools"
 * Matlab toolbox of Eero Simoncelli.
 *
 * The function takes two real double arrays and does an
 * element-by-element norm comparison of nine-pixel
 * neighborhoods for each interior element of L_1 and L_2,
 * and when the best match is found, the new 4-pixel block of
 * L_0 corresponding to the one pixel of L_1 is generated in
 * accordance withthe 4-pixel block of L_1 that corresponds to
 * the one pixel of L_2 with 9-pixel neighborhood
 * most similar to 9-pixel neighborhood of L_1.
 *
 * Differing from the function "exLpyr", this function checks
 * all 8 rotations and reflections of the 9-pixel groups,
 * to find the best match.
 *
 * Because this program has no provision to accommodate edges,
 * the resulting L_0 has a 2-pixel border of zeros.
 *
 * This program produces poor results and is slow.
 *
 * Pete Martin
 * pmartin@ieee.org
 *
 *============================================================*/
 /* $Revision: 1.1 $ */

#include "mex.h" #include <math.h>
```

```
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray
*prhs[]) {

    const int  *dims_first, *dims_second, *dims_third;
    int c,  ncolsIm1, ncolsIm2, ncolsIm3, j, k, m, n, r, w;
    double inf, normDiff, leastDiff, *Im1_ptr, *Im2_ptr, ...
    *Im3_ptr, *out_ptr;
    int nrowsIm1, nrowsIm2, nrowsIm3, nw, no, ne, we, ce, ...
     ea, sw, so, se;

    /* Check for proper number of input and output arguments */
    if (nrhs != 3) {
    mexErrMsgTxt("Three input arguments required.");
    }
    if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments.");
    }

    /* Check data type of first input argument */
    if (!mxIsDouble(prhs[0]) || !mxIsDouble(prhs[1])
    || mxIsComplex(prhs[0]) ||mxIsComplex(prhs[1])
    || mxIsComplex(prhs[2]) || !mxIsDouble(prhs[2])) {
    mexErrMsgTxt("Input arguments must be real of type double.");
    }

    /* Check that dimensions are the same for input arguments. */
    if ( mxGetNumberOfDimensions(prhs[0]) != 2 || ...
        mxGetNumberOfDimensions(prhs[1])!= 2 || ...
        mxGetNumberOfDimensions(prhs[2])!= 2){
    mexErrMsgTxt("Inputs must 2-dimensional matrices.\n");
    }

    dims_first = mxGetDimensions(prhs[0]);
    dims_second = mxGetDimensions(prhs[1]);
    dims_third = mxGetDimensions(prhs[2]);

    /* Check relative sizes of inputs. */
    for (c=0; c<mxGetNumberOfDimensions(prhs[0]); c++){
    if (dims_second[c]!= 2*dims_third[c]){
    mexPrintf...
```

```
 ("Second argument was expected to be twice the size ...
  of third argument...\n");
}
}

/* Set "inf", to be used as first value for for normDiff.*/
inf = mxGetInf();

/* Get the row and column counts for the input matrices. */
nrowsIm1 = dims_first[0];
ncolsIm1 = dims_first[1];
nrowsIm2 = dims_second[0];
ncolsIm2 = dims_second[1];
nrowsIm3 = dims_third[0];
ncolsIm3 = dims_third[1];


/* Get the input values */
Im1_ptr = (double *)mxGetPr(prhs[0]);
Im2_ptr = (double *)mxGetPr(prhs[1]);
Im3_ptr = (double *)mxGetPr(prhs[2]);

/* Create first output and get its input values */
plhs[0]=mxCreateDoubleMatrix(2 * dims_first[0],2 * ...
dims_first[1],mxREAL);
out_ptr = mxGetPr(plhs[0]);

/* For every interior pixel of Im1, search for the interior
* pixel of Im3 with the best 9-pixel neighborhood match, as
* measured by the squared Eucliean difference, called
* "normDiff". Whenever a better match is found, write the
* four output pixels same as the Im2 4-pixel group at the
* same relative location as the single Im3 pixel whose
* neighborhood matched that of the Im1 pixel being tested
* at that point.

* "m" is the Im3 row index; "n" is the Im3 column index;
* "w" is the output matrix pointer index (to write);    */

for(j=2; j<nrowsIm1; j++){
    for(k=2; k<ncolsIm1; k++){
```

```
  leastDiff = inf;
      for(m=2; m<nrowsIm3; m++){
          for(n=2; n<ncolsIm3; n++){
          nw = nrowsIm2 * (n-2) + (m-1) -1;
          we = nrowsIm2 * (n-2) + (m) -1;
          sw = nrowsIm2 * (n-2) + (m+1) -1;
          no = nrowsIm2 * (n-1) + (m-1) -1;
          ce = nrowsIm2 * (n-1) + (m) -1;
          so = nrowsIm2 * (n-1) + (m+1) -1;
          ne = nrowsIm2 * (n) + (m-1) -1;
          ea = nrowsIm2 * (n) + (m) -1;
          se = nrowsIm2 * (n) + (m+1) -1;

  /* Case 1: North on top */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[se]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[se]);
 if(normDiff < leastDiff){
    leastDiff = normDiff;
    w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
    r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

    out_ptr[w] = Im2_ptr[r];
    out_ptr[w+1] = Im2_ptr[r+1];
```

```
      out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + nrowsIm2];
      out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + nrowsIm2 + 1];
      }

  /* Case 2: East on top */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]);
   if(normDiff < leastDiff){
      leastDiff = normDiff;
      w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
      r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

      out_ptr[w] = Im2_ptr[r + nrowsIm2];
      out_ptr[w+1] = Im2_ptr[r];
      out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + nrowsIm2 + 1];
      out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + 1];
      }

  /* Case 3: South on top */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se])
```

```
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]);
 if(normDiff < leastDiff){
    leastDiff = normDiff;
    w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
    r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

    out_ptr[w] = Im2_ptr[r + nrowsIm2 + 1];
    out_ptr[w+1] = Im2_ptr[r+nrowsIm2];
    out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + 1];
    out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r];
    }

  /* Case 4: West on top */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se]) ...
  * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we]) ...
  * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
  * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea]) ...
```

```
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]);
   if(normDiff < leastDiff){
      leastDiff = normDiff;
      w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
      r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

      out_ptr[w] = Im2_ptr[r+1];
      out_ptr[w+1] = Im2_ptr[r+nrowsIm2+1];
      out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r];
      out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + nrowsIm2];
                }

 /* Case 5: North on top, mirror image */

   normDiff =
   (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[ne])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[ea])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[no])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
 + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so]) ...
    * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[so])
 + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw])
 + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[we])
 + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]) ...
    * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[sw]);
     if(normDiff < leastDiff){
        leastDiff = normDiff;
```

```
      w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
      r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

      out_ptr[w] = Im2_ptr[r+nrowsIm2];
      out_ptr[w+1] = Im2_ptr[r+nrowsIm2+1];
      out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r];
      out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + 1];
      }

  /* Case 6: East on top, mirror image */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[ne])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[nw]);
   if(normDiff < leastDiff){
      leastDiff = normDiff;
      w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
      r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

      out_ptr[w] = Im2_ptr[r+nrowsIm2+1];
      out_ptr[w+1] = Im2_ptr[r+1];
      out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + nrowsIm2];
      out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r];
      }
```

```
  /* Case 7: South on top, mirror image */

  normDiff =
  (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[we])
+ (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw]) ...
   * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[nw])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[so])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
+ (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no]) ...
   * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[no])
+ (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[se])
+ (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[ea])
+ (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]) ...
   * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]);
     if(normDiff < leastDiff){
        leastDiff = normDiff;
        w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
        r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

        out_ptr[w] = Im2_ptr[r+1];
        out_ptr[w+1] = Im2_ptr[r];
        out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + nrowsIm2+1];
        out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + nrowsIm2];
        }

   /* Case 8: West on top, mirror image */

   normDiff =
   (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j-1) -1] - Im2_ptr[sw])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j) -1] - Im2_ptr[so])
 + (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se]) ...
    * (Im1_ptr[nrowsIm1 * (k-2) + (j+1) -1] - Im2_ptr[se])
```

```
    + (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we]) ...
       * (Im1_ptr[nrowsIm1 * (k-1) + (j-1) -1] - Im2_ptr[we])
    + (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce]) ...
       * (Im1_ptr[nrowsIm1 * (k-1) + (j) -1] - Im2_ptr[ce])
    + (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea]) ...
       * (Im1_ptr[nrowsIm1 * (k-1) + (j+1) -1] - Im2_ptr[ea])
    + (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw]) ...
       * (Im1_ptr[nrowsIm1 * (k) + (j-1) -1] - Im2_ptr[nw])
    + (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no]) ...
       * (Im1_ptr[nrowsIm1 * (k) + (j) -1] - Im2_ptr[no])
    + (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]) ...
      * (Im1_ptr[nrowsIm1 * (k) + (j+1) -1] - Im2_ptr[ne]);
       if(normDiff < leastDiff){
          leastDiff = normDiff;
          w = 2 * nrowsIm1 * (2 * k - 2) + 2 * j -1 -1;
          r = 2 * nrowsIm3 * (2 * n - 2) + 2 * m -1 -1;

          out_ptr[w] = Im2_ptr[r];
          out_ptr[w+1] = Im2_ptr[r+nrowsIm2];
          out_ptr[w + 2 * nrowsIm1] = Im2_ptr[r + 1];
          out_ptr[w + 2 * nrowsIm1 + 1] = Im2_ptr[r + nrowsIm2 + 1];
     }
    }
   }
  }
 }
}
```

E.2 m-files

dctf.m

```
function yy = dctf(xx);
%   yy = DCTF(xx) returns the frequency-weighted
%   discrete cosine transform yy of signal xx,
%   as a column vector. If xx is a matrix, DCTF(xx)
%   operates on the columns of xx (as DCT does).
%
```

```
%   The purpose is to explore an abstract signal space
%   wherein texture is weighted more in relation to
%   structure, as is supposed to be the case for
%   mammalian vision systems. Ref works of D. Marr,
%   and J. Havlicek.
%
%   See also IDCTF

yy = dct(xx); if (size(yy,1)==1);
    yy = yy.*[1:length(yy)];
elseif (size(yy,2)==1);
    yy = yy.*[1:length(yy)]';
else
    for ii = 1:size(xx,2);
    yy(:,ii) = yy(:,ii).*[1:size(xx,1)]';
    end
end

%   End of file.
```

dctfinterp.m

```
function [yy,cc] = dctfinterp(xp, xs, n, l);
%   DCTFINTERP   -   Computes interpolation of signal from
%                    irregularly-spaced samples, based on
%                    the "DCTF" transform (see DCTF.m).
%   USAGE
%       [yy,cc] = dctfinterp(xp, xs, n l)
%
%   INPUTS
%       xp  -   vector of sample points.
%       xs  -   vector of sample values.
%       n   -   [optional] number of DCT coefficients
%               to estimate. Default is floor(2*length(xp)/3).
%       l   -   [optional] length of signal to be interpolated.
%               Default is xp(length(xp)).
%   OUTPUTS
%       yy  -   Column vector interpolation.
%       cc  -   Column vector of DCT coefficients.
```

```
%
%   See also FFTINTERP
%
%   Author: Pete Martin 2004. martinp@science.oregonstate.edu
%
if nargin < 2
    error('not enough input arguments')
end
%   Check that both xp and xs are either row or column vectors
if
(ndims(xp)>2|ndims(xs)>2|(size(xp(1))>1&size(xp(2))>1)|...
(size(xs(1))>1&size(xs(2))>1))
    error('xp and xs must be vectors')
end
%   If not given, set number of coefficients to estimate equal
% to two thirds the number of sample points
if nargin < 3
    n = floor(2*length(xp)/3);
end
%   If not given, set interpolation length equal to sampling
%   length
if nargin < 4
    l = xp(length(xp));
end

xp = xp(:); xs = xs(:); D = idctf(eye(l)); Dn = D(:,1:n); Dxp =
Dn(xp,:); cc = Dxp\xs; yy = Dn*cc;

%   End of file.
```

dctinterp.m

```
function [yy,cc] = dctinterp(xp, xs, n, l);
%   DCTINTERP   -   Computes interpolation of signal from
%                   irregularly-spaced samples, based on
%                   discrete cosine transform.
%   USAGE
%       [yy,cc] = dctinterp(xp, xs, n l)
```

```
%
%   INPUTS
%       xp  -   vector of sample points.
%       xs  -   vector of sample values.
%       n   -   [optional] number of DCT coefficients
%               to estimate. Default is floor(2*length(xp)/3).
%       l   -   [optional] length of signal to be interpolated.
%               Default is xp(length(xp)).
%   OUTPUTS
%       yy  -   Column vector interpolation.
%       cc  -   Column vector of DCT coefficients.
%
%   See also FFTINTERP
%
%   Author: Pete Martin 2004. martinp@science.oregonstate.edu
%
if nargin < 2
    error('not enough input arguments')
end
%   Check that both xp and xs are either row or column vectors
if
(ndims(xp)>2|ndims(xs)>2|(size(xp(1))>1&size(xp(2))>1)|...
(size(xs(1))>1&size(xs(2))>1))
    error('xp and xs must be vectors')
end
%   If not given, set number of coefficients to estimate equal to
%   two thirds the number of sample points
if nargin < 3
    n = floor(2*length(xp)/3);
end
%   If not given, set interpolation length equal to
%   last sampling point
if nargin < 4
    l = xp(length(xp));
end

xp = xp(:); xs = xs(:); D = idct(eye(l)); Dn = D(:,1:n); Dxp =
Dn(xp,:); cc = Dxp\xs; yy = Dn*cc;

%   End of file.
```

dwtup.m

```
function [a,d] = dwtup(x,varargin)
%   Obtain set of wavelet coefficients without dyadic subsampling,
%   which can be used to sythesize a "super-resolution" version
%   of an original signal.
%
%   INPUTS
%   x         -  signal, a vector
%   varargin  -  either 'wname', e.g. 'db2', or decomposition
%                filters. An extension mode (as 'per' or 'sym')
%                may also be included. Open this m-file or dwt.m
%                to read further comments.

%   This is a simple modification of MATLAB Wavelet Toolbox
%   m-file for DWT. Coments from that m-file are included:
%   DWT Single-level discrete 1-D wavelet transform.
%   DWT performs a single-level 1-D wavelet decomposition
%   with respect to either a particular wavelet ('wname',
%   see WFILTERS for more information) or particular wavelet
%   filters (Lo_D and Hi_D) that you specify.
%
%   [CA,CD] = DWT(X,'wname') computes the approximation
%   coefficients vector CA and detail coefficients vector CD,
%   obtained by a wavelet decomposition of the vector X.
%   'wname' is a string containing the wavelet name.
%
%   [CA,CD] = DWT(X,Lo_D,Hi_D) computes the wavelet
%   decomposition as above given these filters as input:
%   Lo_D is the decomposition low-pass filter.
%   Hi_D is the decomposition high-pass filter.
%   Lo_D and Hi_D must be the same length.
%
%   Let LX = length(X) and LF = the length of filters; then
%   length(CA) = length(CD) = LA where LA = CEIL(LX/2),
%   if the DWT extension mode is set to periodization.
%   LA = FLOOR((LX+LF-1)/2) for the other extension modes.
%   For the different signal extension modes, see DWTMODE.
%
%   [CA,CD] = DWT(...,'mode',MODE) computes the wavelet
```

```
%   decomposition with the extension mode MODE you specify.
%   MODE is a string containing the extension mode.
%   Example:
%     [ca,cd] = dwt(x,'db1','mode','sym');
%
%   See also DWTMODE, IDWT, WAVEDEC, WAVEINFO.

%   M. Misiti, Y. Misiti, G. Oppenheim, J.M. Poggi 12-Mar-96.
%   Last Revision: 02-Aug-2000.
%   Copyright 1995-2002 The MathWorks, Inc.
%   $Revision: 1.15 $

% Check arguments.
if errargn(mfilename,nargin,[2:7],nargout,[0:2]), error('*'),
end

if ischar(varargin{1})
    [Lo_D,Hi_D] = wfilters(varargin{1},'d'); next = 2;
else
    Lo_D = varargin{1}; Hi_D = varargin{2};  next = 3;
end

% Default: Shift and Extension.
dwtATTR = dwtmode('get'); shift   = dwtATTR.shift1D; dwtEXTM =
dwtATTR.extMode;

% Check arguments for Extension and Shift.
for k = next:2:nargin-1
    switch varargin{k}
      case 'mode'  , dwtEXTM = varargin{k+1};
      case 'shift' , shift   = mod(varargin{k+1},2);
    end
end

% Compute sizes.
lf = length(Lo_D); lx = length(x);

% Extend, Decompose &  Extract coefficients.
flagPer = isequal(dwtEXTM,'per'); if ~flagPer
    lenEXT = lf-1; lenKEPT = lx+lf-1;
else
```

```
    lenEXT = lf/2; lenKEPT = 2*ceil(lx/2);
end y = wextend('1D',dwtEXTM,x,lenEXT); a =
convdown(y,Lo_D,lenKEPT,shift); d =
convdown(y,Hi_D,lenKEPT,shift);

%-----------------------------------------------------%
% Internal Function(s)
%-----------------------------------------------------%
function y = convdown(x,f,lenKEPT,shift)

y = wconv('1D',x,f); y = wkeep(y,lenKEPT);

%-----------------------------------------------------%
```

exLpyr8way.m

```
function Im0 = exLpyr8way(Im1,Im2)
% Im0 = exLpyr8way(Im1,Im2)
%
% Generate next larger Laplacian difference image,
% given two Laplacian images in pyramid. Algorithm
% seeks closest match to 9-pixel neighborhood to
% do scale-wise extrapolation. All 8 rotations and
% reflections are used.
%
% IM1 should be image at twice the resolution of IM2;
% output IM0 will be at twice the resolution of IM1.
%
% See also EXLPYR, EXLPYR3, EXLPYR8WAY3

% Pete Martin May 2004.
% The method is after Pentland, et. al.,
% U.S. Patent 5,148,497 dated Sep.15, 1992.
% Refer also to Image and Multi-Scale Pyramid Tools,
% Version 1.1 Mar-2001, created by Eero Simoncelli,
% eero.simoncelli@nyu.edu


if (~isreal(Im1))
```

```
  error('Args must be real-valued matrices');
end

[nrowsIm1,ncolsIm1] = size(Im1); [nrowsIm2,ncolsIm2] = size(Im2);
Im0 = zeros(2*[nrowsIm1,ncolsIm1]); DiffMtx = zeros(3); for
rowindexIm1 =2:nrowsIm1-1
 for colindexIm1 = 2:ncolsIm1-1
  sqEuc = inf;
  C = inf;
  Im1block=zeros(3);
  DiffMtx=zeros(3);
   for rowindexIm2 = 2:nrowsIm2-1
    for colindexIm2 =2:ncolsIm2-1
     FourPix = Im1(2*rowindexIm2-1:2*rowindexIm2,...
     2*colindexIm2-1:2*colindexIm2);
     Im1block = Im1(rowindexIm1-1:rowindexIm1+1,...
     colindexIm1-1:colindexIm1+1);
     Im2block0 = Im2(rowindexIm2-1:rowindexIm2+1,...
     colindexIm2-1:colindexIm2+1);
     Im2block1 = Im2block0';
     Im2block2 = Im2block0(:,3:-1:1);
     Im2block3 = Im2block0(3:-1:1,:);
     Im2block4 = Im2block2';
     Im2block5 = Im2block3';
     Im2block6 = Im2block3(:,3:-1:1);
     Im2block7 = Im2block6';
     Im2block0diff = norm(reshape(Im1block-Im2block0,9,1));
     Im2block1diff = norm(reshape(Im1block-Im2block1,9,1));
     Im2block2diff = norm(reshape(Im1block-Im2block2,9,1));
     Im2block3diff = norm(reshape(Im1block-Im2block3,9,1));
     Im2block4diff = norm(reshape(Im1block-Im2block4,9,1));
     Im2block5diff = norm(reshape(Im1block-Im2block5,9,1));
     Im2block6diff = norm(reshape(Im1block-Im2block6,9,1));
     Im2block7diff = norm(reshape(Im1block-Im2block7,9,1));
     if Im2block0diff < sqEuc
      sqEuc = Im2block0diff;
      Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
      2*colindexIm1)= FourPix;
     end
     if Im2block1diff < sqEuc
      sqEuc = Im2block1diff;
```

```
      Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
      2*colindexIm1)=FourPix';
     end
      if Im2block2diff < sqEuc
       sqEuc = Im2block2diff;
       Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
       2*colindexIm1)= FourPix(:,2:-1:1);
      end
      if Im2block3diff < sqEuc
       sqEuc = Im2block3diff;
       Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
       2*colindexIm1)= FourPix(2:-1:1,:);
      end
      if Im2block4diff < sqEuc
       sqEuc = Im2block4diff;
       Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
       2*colindexIm1) = FourPix(:,2:-1:1)';
      end
      if Im2block5diff < sqEuc
       sqEuc = Im2block5diff;
       Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
       2*colindexIm1)= FourPix(2:-1:1,:)';
      end
      if Im2block6diff < sqEuc
       sqEuc = Im2block6diff;
       Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
       2*colindexIm1) = FourPix(2:-1:1,2:-1:1);
      end
      if Im2block7diff < sqEuc
       sqEuc = Im2block7diff;
        Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1-1:...
        2*colindexIm1)= FourPix(2:-1:1,2:-1:1)';
      end
     end
    end
   end
 fprintf('Row %g done\n',rowindexIm1);
end
```

extendLpyr.m

```
function Im0 = extendLpyr(Im1,Im2)
% Im0 = extendLpyr(Im1,Im2)
%
% Generate next larger Laplacian difference image,
% given two Laplacian images in pyramid. Algorithm
% seeks closest match to 9-pixel neighborhood to
% do scale-wise extrapolation.
%
% IM1 should be image at twice the resolution of IM2;
% output IM0 will be at twice the resolution of IM1.
%
% See also EXTLPYRFIVE

% Pete Martin May 2004.
% The method is after Pentland, et. al.,
% U.S. Patent 5,148,497 dated Sep.15, 1992.
% Refer also to Image and Multi-Scale Pyramid Tools,
% Version 1.1 Mar-2001, created by Eero Simoncelli,
% eero.simoncelli@nyu.edu


if (~isreal(Im1))
  error('Args must be real-valued matrices');
end

[nrowsIm1,ncolsIm1] = size(Im1); [nrowsIm2,ncolsIm2] = size(Im2);
Im0 = zeros(2*[nrowsIm1,ncolsIm1]); DiffMtx = zeros(3); for
rowindexIm1 =2:nrowsIm1-1
 for colindexIm1 = 2:ncolsIm1-1
  sqEuc = inf;
   for rowindexIm2 = 2:nrowsIm2-1
    for colindexIm2 =2:ncolsIm2-1
     DiffMtx = Im1(rowindexIm1-1:rowindexIm1+1,...
     colindexIm1-1:colindexIm1+1)-Im2(rowindexIm2...
     -1:rowindexIm2+1,colindexIm2-1:colindexIm2+1);
     C = norm(reshape(DiffMtx,9,1));
      if C<sqEuc
      sqEuc = C;
```

```
      Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1...
      -1:2*colindexIm1) = Im1(2*rowindexIm2-1:2*rowindexIm2...
      ,2*colindexIm2-1:2*colindexIm2);
    end
   end
  end
 end
    fprintf('Row %g done\n',rowindexIm1);
end
%   End of file.
```

extendLpyr3.m

```
function Im00 = extendLpyr3(Im0,Im1,Im2)
% Im00 = extendLpyr(Im0,Im1,Im2)
%
% Generate next larger Laplacian difference image,
% given two Laplacian images in pyramid and a third
% Laplacian image, as from first-stage interpolation.
% Algorithm seeks closest match in Im2 to 9-pixel
% neighborhood in Im0, taking corresponding 4-pixel
% group in Im1 to write 4-pixel group of Im00.
%
% Adjustment factor should be applied to Laplacians
% to normalize ranges.
%
% IM0 should be image at twice the resolution of IM1;
% output IM00 will be at twice the resolution of IM0.
%
% See also EXTLPYRFIVE

% Pete Martin May 2004.
% The method is after Pentland, et. al.,
% U.S. Patent 5,148,497 dated Sep.15, 1992.
% Refer also to Image and Multi-Scale Pyramid Tools,
% Version 1.1 Mar-2001, created by Eero Simoncelli,
% eero.simoncelli@nyu.edu
```

```
if (~isreal(Im1))
  error('Args must be real-valued matrices');
end

[nrowsIm0,ncolsIm0] = size(Im0); [nrowsIm1,ncolsIm1] = size(Im1);
[nrowsIm2,ncolsIm2] = size(Im2); Im0 =
zeros(2*[nrowsIm1,ncolsIm1]); DiffMtx = zeros(3); for rowindexIm0
=2:nrowsIm0-1
 for colindexIm0 = 2:ncolsIm0-1
  sqEuc = inf;
   for rowindexIm2 = 2:nrowsIm2-1
    for colindexIm2 =2:ncolsIm2-1
     DiffMtx = Im0(rowindexIm0-1:rowindexIm0...
     +1,colindexIm0-1:colindexIm0+1)-Im2(rowindexIm2...
     -1:rowindexIm2+1,colindexIm2-1:colindexIm2+1);
     C = norm(reshape(DiffMtx,9,1));
      if C<sqEuc
       sqEuc = C;
       Im00(2*rowindexIm0-1:2*rowindexIm0,2*colindexIm0...
       -1:2*colindexIm0) = Im1(2*rowindexIm2-1:2*rowindexIm2...
       ,2*colindexIm2-1:2*colindexIm2);
      end
    end
  end
 end
    fprintf('Row %g done\n',rowindexIm0);
end
%   End of file.
```

exLpyrFive.m

```
function Im0 = extLpyrFive(Im1,Im2)
% Im0 = extLpyrFive(Im1,Im2)
%
% Generate next larger Laplacian difference image,
% given two Laplacian images in pyramid. Algorithm
% seeks closest match to 25-pixel neighborhood to
% do scale-wise extrapolation.
%
```

```
% IM1 should be image at twice the resolution of IM2;
% output IM0 will be at twice the resolution of IM1.
%
% See also EXTENDLPYR

% Pete Martin May 2004.
% The method is after Pentland, et. al.,
% U.S. Patent 5,148,497 dated Sep.15, 1992.
% Refer also to Image and Multi-Scale Pyramid Tools,
% Version 1.1 Mar-2001, created by Eero Simoncelli,
% eero.simoncelli@nyu.edu

[nrowsIm1,ncolsIm1] = size(Im1); [nrowsIm2,ncolsIm2] = size(Im2);
Im0 = zeros(2*[nrowsIm1,ncolsIm1]); DiffMtx = zeros(3); for
rowindexIm1 = 3:nrowsIm1-2
 for colindexIm1 = 3:ncolsIm1-2
  sqEuc = inf;
   for rowindexIm2 = 3:nrowsIm2-2
    for colindexIm2 = 3:ncolsIm2-2
     DiffMtx = Im1(rowindexIm1-2:rowindexIm1+2,colindexIm1...
     -2:colindexIm1+2)-Im2(rowindexIm2-2:rowindexIm2...
     +2,colindexIm2-2:colindexIm2+2);
     C = norm(reshape(DiffMtx,25,1));
      if C<sqEuc
      sqEuc = C;
      Im0(2*rowindexIm1-1:2*rowindexIm1,2*colindexIm1...
      -1:2*colindexIm1) = Im1(2*rowindexIm2-1:2*rowindexIm2...
      ,2*colindexIm2-1:2*colindexIm2);
      end
     end
   end
 end
    fprintf('Row %g done\n',rowindexIm1);
end
%   End of file.
```

extendWavDet.m

```
function DD = extendWavDet(D1,D2);
```

```
%   This is an interpolation routine, for generating
%   a next-finer resolution set of wavelet coefficients,
%   based on a search for similar neighbors, as in the
%   "EXLPYR" and related functions for use with pyramidal
%   image decomposition. This is for 1-D case only.
%
%   OUTPUT  -   DD, a vector of wavelet coefficients
lD1 = length(D1); lD2 = length(D2); D1EndRef = [D1(2) D1(1) D1
D1(lD1) D1(lD1-1)]; D2EndRef = [D2(2) D2(1) D2 D2(lD2)...

D2(lD2-1)]; DD = zeros(1,2*lD1); for j = 3:lD1+2
    leastDiff = inf;
    for k = 3:lD2+2
        diff  = norm(D1EndRef(j-2:j+2)-D2EndRef(k-2:k+2));
        if diff<leastDiff
            leastDiff = diff;
            DD(2*(j-2)-1:2*(j-2)) = D1(2*(k-2)-1:2*(k-2));
        end
    end
end
%   End of file.
```

extendWavDet3.m

```
function DD = extendWavDet3(D0,D1,D2);
%   This is an interpolation routine, for generating
%   a next-finer resolution set of wavelet coefficients,
%   based on a search for similar neighbors, as in the
%   "EXLPYR" and related functions for use with pyramidal
%   image decomposition. This is for 1-D case only.
%
%   INPUTS
%       D0  -   Detail coefficients, from first stage
%               interpolation based on D1 and D2.
%       D1  -   Detail coefficients at first stage starting
%               level of resolution.
%       D2  =   Detail coefficients at level 2 of first
%               stage decomposition.
%   OUTPUT  -   DD, a vector of wavelet coefficients
```

```
lD0 = length(D0); lD1 = length(D1); lD2 = length(D2); D0EndRef =
[D0(2) D0(1) D0 D0(lD0) D0(lD0-1)]; D1EndRef = [D1(2) D1(1) D1
D1(lD1) D1(lD1-1)]; D2EndRef = [D2(2) D2(1) D2 D2(lD2) D2(lD2-1)];
DD = zeros(1,2*lD0); for j = 3:lD0+2
    leastDiff = inf;
    for k = 3:lD2+2
        diff  = norm(D0EndRef(j-2:j+2)-D2EndRef(k-2:k+2));
        if diff<leastDiff
            leastDiff = diff;
            DD(2*(j-2)-1:2*(j-2)) = D1(2*(k-2)-1:2*(k-2));
        end
    end
end
%   End of file.
```

fftinterp.m

```
function [yy,cc] = fftinterp(xp, xs, n, l);
%   DCTINTERP   -   Computes interpolation of signal from
%                   irregularly-spaced samples, based on
%                   fast Fourier transform.
%   USAGE
%           [yy, cc] = fftinterp(xp, xs, n, l)
%   INPUTS
%         xp  -   vector of sample points.
%         xs  -   vector of sample values.
%         n   -   [optional] number of FFT coefficients
%                 to estimate. Default is floor(2*length(xp)/3).
%         l   -   [optional] length of signal to be interpolated.
%                 Default is xp(length(xp)).
%   OUTPUTS
%         yy  -   COMPLEX-valued column vector interpolation.
%         cc  -   Estimated FFT coefficients.
%
%   See also DCTINTERP
%
%   Author: Pete Martin 2004. martinp@science.oregonstate.edu
%
if nargin < 2
```

```
    error('not enough input arguments')
end
%   Check that both xp and xs are either row or column vectors
if
(ndims(xp)>2|ndims(xs)>2|(size(xp(1))>1&size(xp(2))>1)|...
(size(xs(1))>1&size(xs(2))>1))
    error('xp and xs must be vectors')
end
%   If not given, set number of coefficients to estimate
%   (zero, positive, and negative frequencies) equal to
%   two thirds the number of sample points
if nargin < 3
    n = floor(2*length(xp)/3);
end
%   If not given, set interpolation length equal to
%   last sampling point
if nargin < 4
    l = xp(length(xp));
end

xp = xp(:); xs = xs(:); D = fft(eye(l)); Dn = [D(:,1:round(n/2))
D(:,l-floor(n/2)+1:l)]; Dxp = Dn(xp,:); cc = Dxp\xs; yy = Dn*cc;

%   End of file.
```

finv.m

```
function xx=finv(n);

% FINV       - generates a random 1/f sequence of length n
%              with 2-norm 1.
%
% Input:     n    - length of signal
%
% Output:    xx   - column vector
%
% Usage:     xx=finv(n);
```

```
% Author: P. Martin, 04.2004, (martinp@science.oregonstate.edu)
%         adapted from the "blim" function by T. Werther,
%         part of the IRSATOL set of m-files developed by
%         NUHAG, Dept.Math., University of Vienna, AUSTRIA
%         http://nuhag.mat.univie.ac.at/
%
% The "blim.m" file has the following copyright:
% Copyright:(c) NUHAG, Dept.Math., University of Vienna, AUSTRIA
%                        http://nuhag.mat.univie.ac.at/
%         Permission is granted to modify and re-distribute this
%         code in any manner as long as this notice is preserved
%         and a copy of the modified m-file is made available to
%         NUHAG. All standard disclaimers apply.



yy=rand(n,1)-0.5; k1 = round(n/2);

%Define the positive half of a symmetric frequency filter
ki = 2; filt = ones(1,n); while ki<=k1+1 filt(ki) = 1/(ki-1);
ki=ki+1; end
%Define the negative half of a symmetric frequency filter
ki = n; while ki>=n-k1
    filt(ki)=filt(n-ki+2);
    ki=ki-1;
end
%Filter yy by multiplying its fft by "filt"; take only real part
yy=real(ifft(fft(yy).*filt')); xx=yy/norm(yy);

%   End of file.
```

flti.m

```
function   filt = flti(n,k);
%   A function to make a 1/f filter, optionally with band limit.
%
%   filt = flti(n,k)
%   INPUTS:
%   n   -   length of filter
```

```
%   k   -   width (negative + zero + positive) of band limit
%           (defaults to n)
%   OUTPUT:
%   filt -  the filter
%
%   Author: Pete Martin, 04.2004. (pmartin@ieee.org)

if nargin == 2;
%   Reverse arguments in case bandwidth is given first
if   n < k ; m = n;  n = k; k = m;  end;
%   Make the bandlimit filter
km = round(k/2); filtbl = [ones(1,km+1)  zeros(1,n-2*km-1)
ones(1,km)];

%   Make full-width filter if k not given:
elseif nargin == 1;
   filtbl=ones(1,n);
end;

% Make the 1/f filter:
%First, define the positive half:
km = round(n/2); ki = 2; filt = ones(1,n); while ki<=km+1 filt(ki)
= 1/(ki-1); ki=ki+1; end
%Define the negative half of a symmetric frequency filter:
ki = n; while ki>=n-km
    filt(ki)=filt(n-ki+2);
    ki=ki-1;
end
%Multiply the band-limiting filter by the 1/f filter:
filt = filt .* filtbl;

%   End of file.
```

idctf.m

```
function xx = idctf(yy);
%   xx = IDCTF(yy) returns the frequency-weighted
%   inverse discrete cosine transform xx of signal yy,
%   as a column vector. If yy is a matrix, IDCTF(yy)
```

```
%   operates on the columns of yy (as IDCT does).
%
%   The purpose is to explore an abstract signal space
%   wherein texture is weighted more in relation to
%   structure, as is supposed to be the case for
%   mammalian vision systems. Ref works of D. Marr,
%   and J. Havlicek.
%
%   See also DCTF

if (size(yy,1)==1);
    yy = yy./[1:length(yy)];
elseif  (size(yy,2)==1);
    yy = yy./[1:length(yy)]';
else
    for ii = 1:size(yy,2);
    yy(:,ii) = yy(:,ii).*[1:size(yy,1)]';
    end
end xx = idct(yy);

%   End of file.
```