Two-Dimensional Recursive Filter Stability with Application to Image Coding

Mark Andrews BE(Hons)

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering

April 1989

Department of Electrical and Electronic Engineering
University of Auckland
New Zealand

Abstract

The work presented in this thesis examines several aspects of two-dimensional recursive digital filter stability in the context of image coding via linear prediction. Unstable prediction filters in image coding models have a catastrophic effect on image quality. Namely, they render the resynthesised image unintelligible. Experimental results show that a significant proportion of analysis frames taken from natural images produce unstable prediction filters. The physical cause of the high rate of instability is examined, and is shown to be due to two separate mechanisms. Firstly, analysis frames which have high inter-pixel correlation produce a disproportionate number of unstable prediction filters. Secondly, an analysis frame provides a truncated view of the image data and may inadvertantly 'deceive' the linear prediction analysis, creating the impression that the image is unbounded globally.

Four related techniques, (based on the Fourier and Hadamard orthogonal transforms), are presented for reducing the intra-frame correlation and therefore reducing the instability rate. It is shown that the probability of deriving a stable prediction filter is increased if the dominant components in either Fourier or Hadamard space are removed. The performance of each method is evaluated and a cost analysis is presented, based on algorithm complexity and transmission bit-rate overhead. It is concluded that removing the dominant sequency components, (when ordered on a magnitude basis), performs most satisfactorily.

Unstable prediction filters underlie the more general problem of determining the stability margin of a two-dimensional infinite-extent impulse response (IIR) filter. While the mathematical formalisms describing the familiar concepts of poles and zeros extend readily to multiple dimensions, the simple and reassuring geometric interpretations do not. As a graphical aid and practical tool, the rootmap is a much-ignored device for visualising the stability condition of two-dimensional recursive filters. An algorithm for its calculation is presented, together with refinements which increase the speed of computation, as well as improve the robustness of the algorithm.

In order to rapidly detect unstable filters, (a basic requirement of real-time image coding systems), a suite of necessary (but insufficient) tests are developed. The new tests successfully detect a large fraction of unstable filters, and are shown to perform well when compared with another necessary (but insufficient) test, as well as against 'sure-fire' methods. The simplicity of the tests is such that they can be implemented in real-time, and the structure of the tests suggests a stabilization strategy which depends on which test (or tests) fail.

To mum and dad

Acknowledgements

During the lengthy course of this study, it has been my pleasure to meet many interesting and intelligent people, with some of whom I have developed strong friendships. This acknowledgement records my gratitude to them for their assistance over the last four years. In addition, there are a few people I would like to thank who, despite having no direct contribution to my recent academic exploits, provided inspiration and support, either outside the University environment or in my fledgling years as an apprentice electrician. Without running to several pages it would be impossible to mention everyone, so I would like to single out the following individuals (and one organization) for thanks:

Professor Jack Woodward for allowing this course of research to proceed and providing part-time work in the Department after my UGC Scholarship expired.

Associate Professor Thong Nguyen, my supervisor, for always being available for helpful discussions, suggesting new avenues of research, and letting me have a fair amount of free-reign.

Alissa Reid, girlfriend and psychoanalyst extraordinaire, for many pep talks, always providing a sympathetic ear, and proof-reading this text.

Merv Young and the late **Jim McIver** of the Devonport Dockyard for organising generous study leave during my apprenticeship despite volumes of regulations and miles of red tape.

Simon Walton, **Graeme Fraser**, **Tony Glucina** and **Scott O'Brian**, for those many late-night brain-storming sessions at the Kiwi Tavern.

The **New Zealand University Grants Committee** for bestowing a postgraduate research scholarship during the first three years of this study.

Gary Prentice, for writing the software that generated the grey-scale images appearing in this thesis.

Finally, I would like to thank **My Parents** who had the dubious pleasure of my company throughout this course of research. This poor couple had to alternately endure my enthusiasm when things went well, and my rantings when they didn't. Thank you both very much.

Table of Contents

Preface

The work that is reported in this thesis took place between July 1985 and September 1988, in the School of Engineering at the University of Auckland. An additional requirement of the engineering PhD program is the undertaking of 3 graduate papers in subjects broadly relevant to the thesis topic. These papers were completed between March and June, 1985. It was originally intended that the author would concentrate on the application of autoregressive models to selected areas of remote sensing, specifically texture modelling and image segmentation. Texture modelling is important because it allows one to distinguish between various types of ground cover such as vegetation, water, desert, urban/non-urban areas etc. Segmentation enables such regions to be mapped and catalogued, and any changes in regional boundaries may be tracked with time.

However, early experiments by the author and his supervisor, Professor Thong Nguyen, showed that a significant number of analysis frames (small subregions in the image) produced unstable prediction filters. This problem is even more catastrophic for synthesis (as opposed to identification) applications such as image coding, since unstable prediction filters simply cannot be tolerated. It was decided to concentrate on this interesting problem and its ramifications in the area of image coding. This precipitated an investigation into the cause of the instability, and the development of techniques which would alleviate the problem. A more general examination of filter stability followed, which resulted in new necessary tests for stability and a fast stabilization scheme. Finally, some preliminary work was completed on a new technique for designing two-dimensional recursive filters that are guaranteed to be stable.

Thesis Organization

The thesis is organized into five chapters. The first two chapters are mainly introductory in nature and the last three contain both background material and original work. Chapter 1 introduces the thesis as a whole and sets the scene by describing, in general terms, linear mathematical models and their place in engineering applications. The mathematical description of multidimensional signals and systems would be unwieldy if it was not for concise symbols. Accordingly, Chapter 2 introduces a compact notation and also spends some time describing the important regions of the multidimensional z-plane. Such regions are central to the Z transform (also described in Chapter 2), and later work on system stability (Chapter 4). Chapter 3 is concerned with image coding via linear prediction, and in particular, the problems that arise when analysis frames produce unstable prediction filters. Several possible solutions are investigated. The more general problem of determining the stability of a two-dimensional recursive filter is addressed in Chapter 4. The rootmap, a simple graphical tool useful for stability investigations, is introduced and a method for its calculation is described. A suite of new stability tests are proposed and their performance is evaluated relative to other methods. Finally, Chapter 5 presents several areas worthy of further research, followed by the major conclusions.

Publications

Several technical and conference publications have been presented during the course of this research. They are listed below in order of preparation.

1 *Edge detection via direction filtering*
DT Nguyen and M Andrews: **Proc. 1st Int. Conf. Future Advances in Comp.**, Christchurch, New Zealand, Feb. 16-21, 1986.

2 *Stability problems associated with two-dimensional linear predictive coding of images*
M Andrews: **Proc. GT Murray Memorial Award (IPENZ)**, 10p., 1987.

3 *Improving the stability rate of two-dimensional linear predictive coding schemes*
M Andrews and DT Nguyen: **Proc. ISSPA 87** (Int. Symp. on Sig. Proc. and its Applications), Brisbane, Australia, August 24-28, 1987, pp.677-682.

4 *A new technique in hi-low coding of images*
DT Nguyen, DJ Knowles and M Andrews: **Proc. ISSPA 87** (Int. Symp. on Sig. Proc. and its Applications), Brisbane, Australia, August 24-28, 1987, pp.662-666.

5 *Necessary conditions for stability and a simple stabilization scheme for two-dimensional digital filters*
M Andrews and DT Nguyen: **Electronics Letters**, Vol.24, No. 14, 7 July 1988, pp.843-844.

6 *Robust rootmap calculation*
M Andrews: **Electronics Letters**, Vol. 24, No. 15, 21 July 1988, pp.968-969.

7 *Techniques for improving the stability rate of linear predictive image coding schemes*
M Andrews and DT Nguyen: **IEE Proc. Part E: Computers and Digital Techniques**, Vol. 135, No. 6, 1988, pp.298-306.

Glossary

$\mathbf{R}^N$	N-dimensional field of real numbers
$\mathbf{C}^N$	N-dimensional field of complex numbers
$\mathbf{I}^N$	N-dimensional ring of integers
n_i, k_i, etc	integer indices in the i^{th} dimension
$\mathbf{n}$, $\mathbf{k}$, etc	N-dimensional integer indices
z_i	Z transform variable in the i^{th} dimension
$\mathbf{z}$	N-dimensional Z transform variable
$x(\mathbf{n})$, $y(\mathbf{n})$, etc	complex sequences defined over $\mathbf{I}^N$
$X(\mathbf{z})$, $Y(\mathbf{z})$, etc	Z transform of $x(\mathbf{n})$, $y(\mathbf{n})$, etc defined over $\mathbf{C}^N$
R_x, R_y, etc	region of support of $x(\mathbf{n})$, $y(\mathbf{n})$, etc
R_X, R_Y, etc	region of convergence of $X(\mathbf{z})$, $Y(\mathbf{z})$, etc
$\mathbf{U}^N$	open unit polydisc
$\overline{\mathbf{U}}^N$	closed unit polydisc
$\partial\mathbf{U}^N$	boundary of $\mathbf{U}^N$
$\mathbf{T}^N$	distinguished boundary of $\mathbf{U}^N$
ω_i	normalised frequency variable in the i^{th} dimension
ω	normalised N-dimensional frequency variable
ζ_1, ζ_2	rootmaps in the z_1 and z_2 planes respectively
$\eta(S)$	cardinality of the set S

BIBO	bounded-input bounded-output
CCD	charge-coupled device
DFT	discrete Fourier transform
FFT	fast Fourier transform
LHS	left hand side
RHS	right hand side
ROC	region of convergence
WHT(FHT)	Walsh-Hadamard (fast Hadamard) transform

Chapter 1

Introduction

1.1 Introduction

It often happens that identical mathematical techniques are developed in disparate fields of research for solving seemingly unrelated problems. Usually, each solution has its own language or symbolism, and this sometimes prevents immediate recognition of any similarities between them. Solving the linear prediction problem, and designing recursive digital filters, is a case in point, and are really opposite sides of the same mathematical coin. The philosophy driving each application is different, despite the fact that each may be implemented in a similar way, and the design methodologies are identical. Linear prediction finds most application in mathematical modelling. That is, it is postulated that a signal may be regarded as the output of some predictive (recursive) process, based on past samples of the signal. Recursive filter theory is concerned with the design and construction of devices for modifying the spectral content of a signal. By shifting our point of view accordingly, we may interpret linear prediction as a simple recursive filtering operation, or view (a certain class of) recursive filters as linear predictors.

The material presented in this thesis examines various aspects of both approaches. Specifically, stability problems in 2-D linear predictive schemes, and 2-D recursive filters in general, are examined. It transpires that many of the familiar and well-established one-dimensional results simply do not generalise to multiple dimensions in an obvious way. The purpose of this short chapter is to highlight several one-dimensional problems which have been successfully attacked using the linear prediction/recursive filtering approach, and then show that their (seemingly) natural extension to several dimensions is often fraught with difficulties. Section 1.2 discusses, in general terms, linear prediction and certain of its applications. Section 1.3 briefly examines the design problem for one-dimensional recursive filters. Finally, Section 1.4 discusses some of the unusual side effects that occur when traditional methods are extended to two dimensions.

1.2 One-Dimensional Linear Prediction Applications

As mentioned in Section 1.1, a linearly predictive model assumes that a signal sample may be estimated from a linear weighted sum of past samples of the waveform, and possibly some excitation or error term. One of the reasons why linear prediction has proved so popular in the literature is that, not only is the model linearly dependent on its parameters, but the determination of the optimum coefficients is a linear problem. The importance of this fact should not be overlooked. Being a linear problem ensures that the solution can be found in one step; that of solving a set of simultaneous linear equations (notwithstanding possible ill-conditioning). In contrast, the vast majority of non-linear models require that their optimum solution be found in an iterative fashion. Searching for a global minimum in, say, ten-dimensional parameter-space, involves much more computation than solving for ten unknowns in a system of linear equations. Moreover, without reasonably accurate starting values, the non-linear minimization procedure may easily stall in a local, rather than a global, minimum.

Linear prediction has had its greatest successes in time-series modelling. In the area of speech processing, for example, a knowledge of the prediction coefficients allows us to perform either analysis or synthesis. Thus, speaker identification may be achieved (in principle) by matching the derived filter coefficients with stored templates of speech fragments. Similarly, if the prediction filter is driven with a suitable excitation, and the filter coefficients are updated periodically, we have the capability of synthetic speech. The reason why predictive synthetic speech has proved possible is that the linear prediction process correctly estimates the formants (peaks) in the speech magnitude spectrum, although not the spectral phases. This does not hinder intelligibility as the human ear is relatively insensitive to phase errors. Phase insensitivity in hearing means that high-compression speech coding may be implemented by transmitting or storing the model coefficients rather than the speech samples themselves. Reasonable results are forthcoming, but in order to obtain high-quality performance, the model must be enhanced beyond its most basic topology.

Formant-matching in speech processing is a result of the so-called autocorrelation matching property of linear prediction [1]. This same feature makes linear prediction a prime candidate for spectral estimation. Specifically, the autocorrelation sequence derived from the model (when excited by an impulse), matches the autocorrelation samples derived from the data (up to the filter length). Spectral matching follows by virtue of the Weiner-Khinchine theorem [2,p.38].

1.3 Aspects of One-Dimensional Recursive Filtering

Recursive filters form a class of linear filter whose outputs depend, generally, on past and current input samples, and past output samples. (As a special case, if the output depends only on past output samples, and the current input sample, then the filter is termed all-pole. An all-pole filter has a transfer function identical to that of the linear predictor discussed in Section 1.2, provided the predictor uses only past output samples and the current input sample). The problem of designing a pole-zero filter to match a specified frequency response is not dissimilar to the spectral estimation problem; except, in addition, we often place constraints on the spectral phase. The extra flexibility that results from being able to position zeros as well as poles is obtained at the expense of solution complexity. The simultaneous determination of the numerator and denominator coefficients of the filter transfer function usually manifests itself as a non-linear optimization problem, along with its attendant difficulties, mentioned in the previous section.

Apart from the optimality of the design, an important implementation consideration is that the filter must be stable. Testing for stability in one-dimensional digital filters is a well-understood topic, and usually involves elementary arithmetic operations on the filter coefficients. If a filter proves unstable, it may be stabilized in one of several ways. For example, during the design stage the unstable transfer function may be algebraically cascaded with an all-pass section whose transmission zeros are co-incident with the unstable filter poles; or the filter coefficients may be modified, as in planar least-square inverse stabilization. If the filter under consideration is actually an all-pole prediction filter, we have the added option of simply 'flipping' the unstable pole back inside the unit circle to its conjugate reciprocal position. This will preserve the filter magnitude response and rescue the design from instability.

1.4 Extension to Several Dimensions

It is not unreasonable to suspect that one-dimensional theories and concepts may generalise to several dimensions. Consequently, linear prediction techniques and recursive filter design methodologies have been adapted from one dimension, with a view to obtaining similar results. Interestingly, the transition from one to several dimensions has not been a smooth one, and various problems (with no one-dimensional analogue), have plagued a number of applications. Some examples follow.

Given the large quantity of information in natural pictures, it is obviously desirable to compress an image prior to it being transmitted or stored. Unlike the high compression ratios obtained for speech, the results for image coding have been less than spectacular. The reason is that, except in the most abstract sense, there is no underlying physical process describing the image brightness in simple mathematical terms. To illustrate, an image may consist of a collection of objects arranged in some arbitrary manner. The brightness at any point in the image depends on the scene illumination, the reflectance of the various objects, and their relative positions. Any inter-pixel correlation, (which reduces the prediction error), is almost purely coincidental, rather than an inherent property of the image generation process. Consequently, any rapid variations in image contrast, typically found along the boundary points of an object, are modelled poorly by linear prediction, giving rise to a resultant increase in the variance of the error signal. Image correlation also plays an important role in predictor stability, and is examined in more detail in Chapter 3.

It should also be noted that the fidelity requirements are stricter for images than they are for one-dimensional signals such as speech. The human ear is relatively insensitive to phase errors in synthetic speech, relying instead on a good estimate of the spectral magnitude. On the other hand, phase errors in synthetic or reconstructed images cannot be tolerated since the human eye is most sensitive to the exact signal form, that is: accurate magnitude and phase reconstruction. The net result is to complicate the image coding problem and this at least partially contributes to the linear prediction compression ratio being lower for images than for speech.

In the area of multidimensional recursive filter design, producing an optimal and stable design is a difficult problem. The cause of the problem is easy to identify and is due to the fact that there is no fundamental thereom of algebra for polynomials with dimension greater than one [3]. Unlike one-dimensional poles and zeros, multidimensional zeros and singularities of rational functions form a continuum. The continuous and complex nature of these singularities make filter stability testing decidedly non-trivial (see Chapter 4), and has been called the "fundamental curse" on multidimensional filter work [4]. (Although it should be stated that there are areas, such as phase retrieval and blind deconvolution, where irreducible polynomials are a positive advantage. For example, see [5],[6]).

The inability to factorize the denominator polynomial of a recursive filter also means that an unstable design cannot be stabilized by cascading with an all-pass network. Such is the case because any two functions which are identical over a continuum, no matter how small, must be identical functions. Hence the stable portions of the denominator zero-set are cancelled along with the unstable portions; the net result being a completely obliterated frequency response. Designers cannot even rely on the multidimensional version of planar least-square inverse stabilization. The two-dimensional implementation was recently shown to fail in certain circumstances [7]. It seems that the best solution is to design a stable filter in the first place. To date, this has been accomplished using the spectral factorization properties of the two-dimensional complex cepstrum [8], (pronounced *kep*-strum), although an alternative procedure is proposed in Chapter 5.

Finally, spectral estimation based on two-dimensional all-pole modelling behaves quite differently to its one-dimensional counterpart. In particular, autocorrelation matching is an inherent property in the one-dimensional case *only*. Thus, if we inverse Fourier transform the estimated power spectrum, the resulting autocorrelation estimates will not necessarily match the true autocorrelation samples of the signal. Even more remarkable is the fact that all-pole filters with positive-definite autocorrelation matrices may in fact be unstable [9,p.328]

In summary, multidimensional systems exhibit a richness of behaviour not found in traditional one-dimensional systems. Their counter-intuitive properties, coupled with the inherent difficulty of visualising constructs in several (i.e. more than three) dimensions, make the study of these interesting, but difficult, problems all the more challenging.

1.5 References

1 **SM Kay** and **SL Marple Jr**: 'Spectrum analysis - a modern perspective', *Proc. IEEE*, Vol. 69, No. 11, 1981, pp.1380-1419.

2 **KS Shanmugam**: 'Digital and analog communication systems', Wiley, New York, 1979.

3 **MH Hayes** and **JH McClellan**: 'Reducible polynomials in more then one variable', *Proc. IEEE*, Vol. 70, No. 2, 1982, pp.197-198.

4 **TS Huang, WF Schreiber** and **OJ Tretiak**: 'Image processing', *Proc. IEEE*, Vol. 59, No. 11, 1971, pp.1586-1609.

5 **RG Lane, WR Fright** and **RHT Bates**: 'Direct phase retrieval', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. 36, No. 4, 1987, pp.520-526.

6 **D Izraelevitz** and **JS Lim**: 'A new direct algorithm for image reconstruction from Fourier transform magnitude', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. 36, No. 4, 1987, pp.511-519.

7 **Y Genin** and **Y Kamp**: 'Counter-example in the least-square inverse stabilization of 2D recursive filters', *Electronics Letters*, Vol. 11, No. 15, 24 July 1975, pp.330.

8 **MP Ekstrom** and **JW Woods**: 'Two-dimensional spectral factorization with applications in recursive digital filtering', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-24, No. 2, 1976, pp.115-128.

9 **DE Dudgeon** and **RM Mersereau**: 'Multidimensional digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

Chapter 2

Mathematical Preliminaries

2.1 Introduction

As a precursor to the work which follows later in the thesis, this chapter briefly summarizes the mathematical concepts and tools necessary for a study of linear, shift-invariant, multidimensional digital systems. Unlike the theory of one-dimensional systems, which is familiar to all Engineers due to widespread and thorough exposition at the undergraduate level, multidimensional systems often require a level of abstraction slightly above the norm. For example, multidimensional sequences become dependent on several independent variables, often both spatial and temporal in a real situation. The Z Transform of such a sequence is similarly defined over a multidimensional complex space, and while it is straightforward to describe the situation mathematically, it becomes more difficult to concisely visualise the 'mechanics' of the mathematics. Our ability to do this easily and automatically for one-dimensional systems is often overlooked.

With a view to easing the transition to several dimensions, Section 2.2 introduces a compact and obvious notation for describing multidimensional sequences, along with the real and complex spaces in which they exist. Section 2.3 describes the most important subregions (strictly subsets) of the multidimensional complex space C^N. This section anticipates the results presented in Chapter 4, where the forementioned subsets play a central role in describing the concept of stability in multidimensional digital filters. A brief description of sampled signals follows in Section 2.4. Primarily intended to introduce the idea of *region of support* for a discrete signal, it also provides a bridge between (generally) continuous real-world signals and the necessarily discrete sequences found in digital hardware. The multidimensional Z transform, fundamental to any serious study of digital systems, is presented in Section 2.5, followed by an important special case, the Fourier transform, in Section 2.6. Finally, Section 2.7 briefly outlines the multidimensional Walsh-Hadamard transform (WHT). While the WHT is almost never used as an analysis tool, its greatest claim to fame is in the area of transform coding of signals, where its orthogonality and speed of computation are exploited with considerable success. These very desirable features will be used in Chapter 3.

2.2 Notation

In order to convey ideas and concepts as simply and concisely as possible, it is important to have an efficient and obvious form of notation. This thesis is primarily concerned with multidimensional signals; that is, single-valued functions with several 'dimensions' of independent variables. The signals may take on either real or complex values and each independent variable may be either an integer, a real, or a complex, number.

A complex number z is denoted by

$$z = x + jy, \qquad x,y \in \mathbf{R}, \quad z \in \mathbf{C} \tag{2.1}$$

where **R** is the set of real numbers and **C** is the set of complex numbers, formed by the cartesian product of **R** with itself:

$$\mathbf{C} = \mathbf{R} \times \mathbf{R} = \mathbf{R}^2 \tag{2.2}$$

An N-dimensional complex number **z** (note the bold-face type) is denoted by the N-tuple

$$\mathbf{z} = (z_1, z_2, \ldots, z_N) \quad z_i \in \mathbf{C} \quad \text{and} \quad \mathbf{z} \in \mathbf{C}^N \tag{2.3}$$

where $\mathbf{C}^N$ is the N-fold cartesian product of **C** with itself.

A function of N variables $x_1, x_2, \ldots, x_N$ may be written as

$$y = F(x_1, x_2, \ldots, x_N)$$

but is more succinctly written as

$$y = F(\mathbf{x}) \tag{2.4}$$

where, in general, y and **x** will be 1-D and N-D complex variables, respectively. It often happens that y is a function of a multidimensional 'index' **n**:

$$y = f(\mathbf{n}) \qquad y \in \mathbf{C} \quad \text{and} \quad \mathbf{n} \in \mathbf{I}^N \tag{2.5}$$

where **I** is the set of integers.

In addition to these, and following the useful notation in [1,p.1-2], we have

$$\mathbf{z}^{\mathbf{m}} = z_1^{m_1} z_2^{m_2} \ldots z_N^{m_N} \qquad z_i \in \mathbf{C} \quad \text{and} \quad m_j \in \mathbf{I} \tag{2.6}$$

$$\begin{aligned} \mathbf{m} + \mathbf{n} &= (m_1, m_2, \ldots, m_N) + (n_1, n_2, \ldots, n_N) \\ &= (m_1+n_1, m_2+n_2, \ldots m_N+n_N) \end{aligned} \tag{2.7}$$

$$\alpha\mathbf{z} = (\alpha z_1, \alpha z_2, \ldots, \alpha z_N) \qquad \alpha \in \mathbf{C} \quad \text{and} \quad \mathbf{z} \in \mathbf{C}^N \tag{2.8}$$

2.3 Subsets of C^N

Certain subsets of the N-dimensional complex space C^N occur frequently and are therefore given special labels because of their importance. Most of these regions are concerned with the generalization to N dimensions of the familiar unit disc or unit circle.

One Dimensional Case

For the 1-D case $\mathbf{z} = z \in \mathbf{C}$, and we have the following:

The open unit disc is defined as

$$\mathbf{U} = \{ z: \ |z| < 1 \} \tag{2.9}$$

The closed unit disc, denoted $\overline{\mathbf{U}}$, consists of $\mathbf{U}$ and its boundary points:

$$\overline{\mathbf{U}} = \{ z: \ |z| \leq 1 \} \tag{2.10}$$

The boundary ∂S of a region S is given by $\partial S = \overline{S} - S$, hence the boundary of $\mathbf{U}$ is

$$\partial \mathbf{U} = \overline{\mathbf{U}} - \mathbf{U} = \{ z: \ |z| = 1 \} \tag{2.11}$$

N-Dimensional Case

If $\mathbf{z} \in \mathbf{C}^N$, the previous results generalize as follows (refer to Figure 2.1 for a diagrammatic representation when N=2). The open unit *polydisc* is defined as

$$\mathbf{U}^N = \{ \mathbf{z}: \ |z_i| < 1, \quad i=1,...,N \} \tag{2.12}$$

Similarly, the closure of $\mathbf{U}^N$ gives the closed unit polydisc

$$\overline{\mathbf{U}}^N = \{ \mathbf{z}: \ |z_i| \leq 1, \quad i=1,...,N \} \tag{2.13}$$

The boundary, $\partial \mathbf{U}^N$, of $\mathbf{U}^N$ is given by

$$\partial \mathbf{U}^N = \overline{\mathbf{U}}^N - \mathbf{U}^N \tag{2.14}$$

and a very important region, called the *distinguished boundary* of the unit polydisc, denoted $\mathbf{T}^N$, is given by

$$\mathbf{T}^N = \{ \mathbf{z}: \ |z_i| = 1, \quad i=1,...,N \} \tag{2.15}$$

It may seem strange at first, but $\mathbf{T}^N \neq \partial \mathbf{U}^N$, except for N=1. In fact, for $N \geq 2$, $\mathbf{T}^N$ forms a proper subset of $\partial \mathbf{U}^N$:

$$\partial \mathbf{U}^N \supset \mathbf{T}^N, \qquad N \geq 2 \tag{2.16}$$

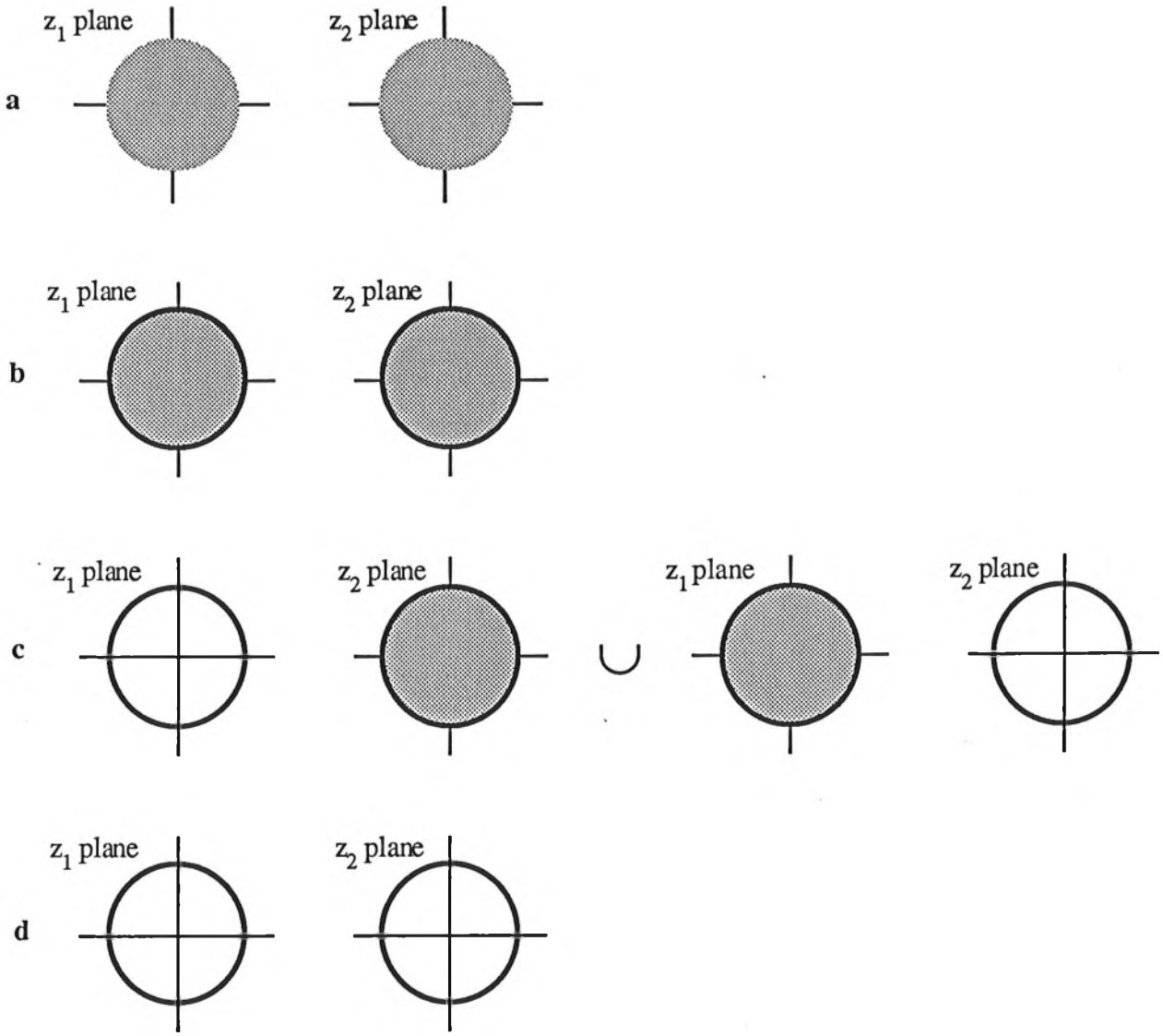


Figure 2.1: *Illustrating polydisc regions of C^N for the case N=2.*

a U^2 **b** $\overline{U}^2$ **c** ∂U^2 **d** T^2

and if the case N=1 is included, T^N forms a simple subset of ∂U^N:

$$\partial U^N \supseteq T^N \tag{2.17}$$

Thus, a point **z** belongs to ∂U^N if *any one or more* of its component variables z_i belong to T_i, where

$$T_i = \{ z_i: \ |z_i| = 1 \}$$

and the rest of the component variables z_j ($j \neq i$) remain in $\overline{U}^{N-1}$. These ideas are illustrated in Figure 2.2 for the case N=3. With the 3-tuples defined as in the caption to Figure 2.2, the following

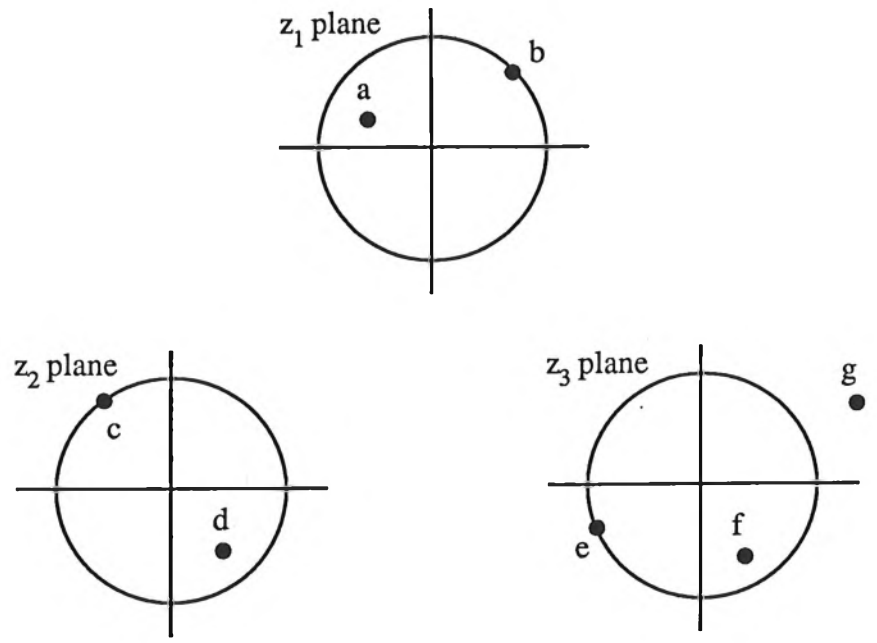


Figure 2.2: *Illustrating the important subsets of C^N for the case N=3.*
The 3-tuples in question are $X_1=(a,d,f)$; $X_2=(a,c,f)$; $X_3=(b,c,e)$; $X_4=(a,c,g)$.

statements are true:

$$X_1 \in \overline{U}^3, \quad X_1 \in U^3$$
$$X_2 \in \overline{U}^3, \quad X_2 \in \partial U^3$$
$$X_3 \in \overline{U}^3, \quad X_3 \in \partial U^3, \quad X_3 \in T^3$$
$$X_4 \in C^3$$

2.4 Sampled Signals

As far as our macroscopic observations of physical processes are concerned, nearly all appear continuous in both time (and space) and amplitude. In order to manipulate such 'real-world' signals in a digital computer with finite resources, it is first necessary to truncate or limit the extent of the signal in some way. To this end, it is usual to both sample and quantise the signal prior to any operations being performed on it, as shown diagrammatically in Figure 2.3.

2.4.1 Definitions

Nearly all real-world signals can be modelled as

$$x(t), \qquad t \in R^N \quad \text{and} \quad x \in R \tag{2.18}$$

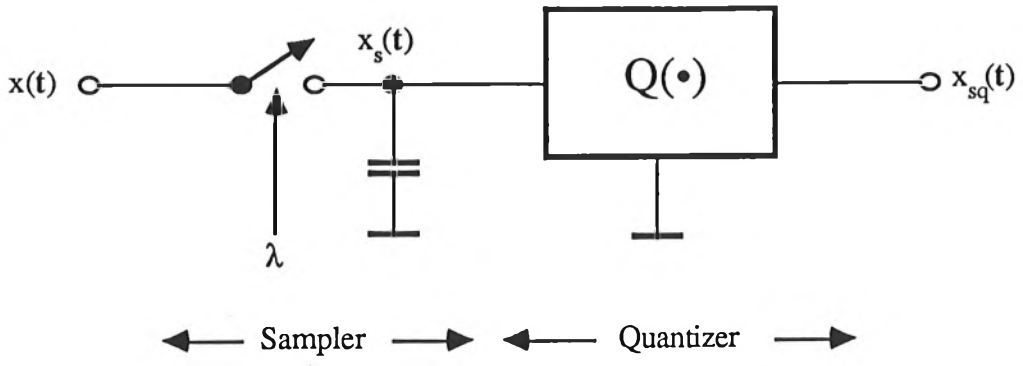


Figure 2.3: *Classical diagrammatic interpretation of the mathematical process of sampling and quantization.*

where the N-D real space **t** is mapped to a single real value, x. The sampled signal $x_s(\mathbf{t})$ is derived by discretizing the **t** variables according to

$$t_i = n_i \lambda_i, \quad t_i,\lambda_i \in \mathbf{R} \text{ and } n_i \in \mathbf{I} \tag{2.19}$$

where λ_i is called the *sampling interval* and n_i is called the *sample index* or *sample number* for the i^{th} dimension. The discrete **t** variables may now be conveniently expressed as

$$\mathbf{t} = \Lambda\mathbf{n}, \quad \mathbf{t} \in \mathbf{R}^N, \ \Lambda \in \mathbf{R}^{2N} \text{ and } \mathbf{n} \in \mathbf{I}^N \tag{2.20}$$

where **n** is an integer 'vector' and Λ is called the *periodicity matrix*:

$$\Lambda = \text{diag}\ (\lambda_1,\lambda_2,\ldots,\lambda_N) \tag{2.21}$$

The sampling scheme expressed in eqn. (2.19) is called rectangular sampling, for obvious reasons, and gives rise to the diagonal nature of the Λ matrix. A more general sampling strategy is possible (such as hexagonal sampling) if the off-diagonal elements of Λ assume non-zero values. Only rectangular sampling will be considered in this thesis†, although extending the results to general sampling schemas would appear straightforward [2,p.39].

In view of eqn. (2.20), the sampled signal $x_s(\mathbf{t})$ is given by

$$x_s(\mathbf{t}) = x(\Lambda\mathbf{n})$$

and it is usual to drop the explicit dependence of x_s on Λ since values of x_s can only be referenced

---

† If Λ is always a diagonal matrix, then it is often more convenient to write **t** as the Hadamard product [3,p.54] $\mathbf{t} = \Lambda_d \bullet \mathbf{n}$, where Λ_d is an N-vector formed from the diagonal elements of Λ.

via **n**, giving

$$x_s(t) = x(\mathbf{n}), \qquad \mathbf{n} \in \mathbf{I}^N \tag{2.22}$$

In order to complete the conversion of x(**t**) to a computer-usable form, it is necessary to quantize x, the range of the domain, **t**. In a similar manner to eqn. (2.19), the quantization of x may be achieved as

$$x = \bar{x}.\Delta, \qquad \bar{x} \in \mathbf{I} \quad \text{and} \quad \Delta \in \mathbf{R} \tag{2.23}$$

where Δ is the *amplitude quantization interval.* Thus, the sampled and quantized signal, $x_{sq}(\mathbf{t})$, is given as

$$x_{sq}(\mathbf{t}) = (\bar{x}.\Delta)(\mathbf{n}) = \bar{x}(\mathbf{n}) \tag{2.24}$$

where the last expression has its dependence on Δ implied. Now, while the formulation of the discrete signal $x_{sq}(\mathbf{t})$ in eqn. (2.24) is correct, it introduces complications far exceeding any advantages which may ensue from an accurate description of the true physical signal. The reason is that quantizing the amplitude as per eqn. (2.23) forces the use of finite or fixed-point arithmetic. All mathematical operations are then plagued with the non-linearities inherent in fixed-point operations. For this reason, it is usual to ignore the effect of quantization and instead assume that the computer has available a sampled signal x(**n**), with an amplitude which can take on a continuum of values. This assumption is perfectly reasonable if the quantization interval Δ is very small, and mathematical operations performed in the computer use floating-point arithmetic.

2.4.2 Practical Sequences and Regions of Support

It often happens that the sequence x(**n**) is non-zero in only a subset of $\mathbf{I}^N$. Such situations arise when measuring aperiodic or transient phenomena, or when the recording instrumentation forces a restricted view of the signal, such as a CCD camera with only a finite number of photoreceptive picture elements. Even if this were not the case it would still be necessary to restrict the number of degrees of freedom of the signal, simply to contain the sampled values within finite real-world computers. Thus we must modify eqn. (2.22) to the following

$$x_s(t) = x(\mathbf{n}), \qquad \mathbf{n} \in R_x \quad \text{with} \quad \mathbf{I}^N \supset R_x \tag{2.25}$$

where R_x is called the *region of support* of the signal x(**n**). (Note that R_x should not be confused with the field of real numbers **R**). If x(**n**) is transitory in nature, then R_x is of the form

$$R_x = \{\, \mathbf{n}:\ x(\mathbf{n}) \neq 0 \,\} \tag{2.26}$$

otherwise R_x may be arbitrarily specified (for practical reasons) or implicitly defined by physical limitations imposed be the sampling/recording instrumentation.

2.5 The Multidimensional Z Transform

Transform methods were originally introduced into mathematical work in order to simplify certain types of problems. For example, the Laplace transform is often used to reduce a system of linear differential equations to a system of linear algebraic equations. Whatever the specific application, the impetus is always to transform the problem definition into another 'domain', where the recast problem is hopefully trivial or at least tractable. The classical method used in connection with discrete sequences is the Z transform, which is really a discrete Laplace transform [3,p.321], and may be viewed as an analytic continuation of the Fourier transform of a discrete sequence.

2.5.1 Definition of the Z Transform

For a bounded sequence

$$\{x(\mathbf{n})\}, \qquad \mathbf{n} \in R_x \quad \text{and} \quad x \in \mathbf{C}$$

with support in R_x, the Z transform simply maps the discrete sequence $\{x(\mathbf{n})\}$ onto the complex hyperplane $\mathbf{C}^N$ via the relation

$$X(z_1,z_2,\ldots,z_N) = \sum_{n_1} \sum_{n_2} \cdots \sum_{n_N} x(n_1,n_2,\ldots,n_N)\, z_1^{n_1} z_2^{n_2} \cdots z_N^{n_N}$$

or more compactly, using the notation introduced in Section 2.2:

$$X(\mathbf{z}) = \sum_{\mathbf{n}} x(\mathbf{n})\, \mathbf{z}^{\mathbf{n}} \tag{2.27}$$

Note that eqn. (2.27) defines the Z transform in positive powers of **z**, whereas the more familiar 1-D Z transform is defined in negative powers of z, for example [4,p.28],[5,p.78],[6,p.13]. It certainly makes no difference to the technical results which definition is used, and if either is adopted, it is usually on the grounds of convenience. For this reason, eqn. (2.27) is the definition of choice because in later work (Chapter 4), where the regions of analyticity of $X(\mathbf{z})$ must be identified, this form will lead most readily to the results of interest.

2.5.2 Region of Convergence

In general, the summation in eqn. (2.27) will be finite for only certain values (maybe only one value) of **z**. The values of **z** for which $X(\mathbf{z})$ is finite is called the *region of convergence (ROC)* R_X, of $X(\mathbf{z})$, and for arbitrary values of $x(\mathbf{n})$, the region is very difficult to identify. Formally, we may define

$$R_X = \{\, \mathbf{z}: \ |X(\mathbf{z})| < \infty \,\} \tag{2.28}$$

but without very detailed knowledge of $x(\mathbf{n})$ the most we can say is that $\mathbf{C}^N \supset R_X$ – obviously of little use. For the 1-D and 2-D cases, the region R_X may be conveniently viewed as a line segment and plane, respectively.

Given the 1-D Z transform

$$X(z) = \sum_{n} x(n)\, z^{n} \qquad \text{where} \quad n \in R_x,$$

the ROC of X(z) will in general be the line segment shown in Figure 2.4. Note that only the modulus of z is important in defining R_X, since if z_o is in the ROC, then so is $z_o \exp(j\theta)$, for real θ.

$$R_X = \{\, z:\ a < |z| < b \,\}$$
$$= \{\, z:\ |X(z)| < \infty \,\}$$

Figure 2.4: *Minimal requirements for displaying the ROC of a 1-D Z transform.*

Given the 2-D Z transform

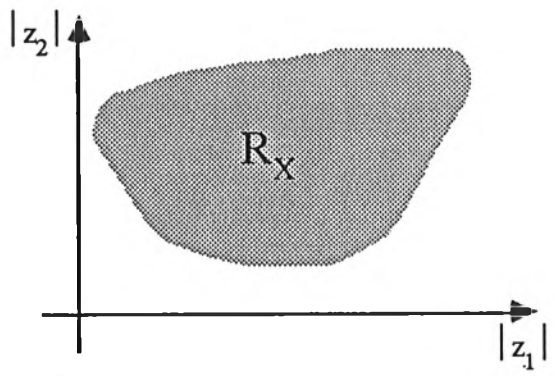
$$X(z_1,z_2) = \sum_{n_1} \sum_{n_2} x(n_1,n_2)\, z_1^{n_1}\, z_2^{n_2} \qquad \text{where} \quad (n_1,n_2) \in R_x,$$

the ROC of $X(z_1,z_2)$ will in general be a simply connected region such as the one shown in Figure 2.5. Obviously we may extend the concept to higher dimensions, for example a volume of convergence for 3-D Z transforms, but the results rapidly become difficult to visualise.

2.5.3 Inversion of the Z Transform

Given the Z transform of a signal or digital system, it is often desirable to invert the transform to recover the sequence of interest. In principle, this is most easily accomplished as follows. Rewrite the Z transform

$$X(\mathbf{z}) = \sum_{\mathbf{n}} x(\mathbf{n})\, \mathbf{z}^{\mathbf{n}}, \qquad \mathbf{n} \in R_x$$



$$R_X = \{ (z_1,z_2): |X(z_1,z_2)| < \infty \}$$

Figure 2.5: *Minimal requirements for displaying the ROC of a 2-D Z transform. R_X is not bordered by a solid line indicating that the ROC does not include its boundary points.*

as

$$X(\mathbf{z}) = x(\mathbf{k})\, \mathbf{z}^{\mathbf{k}} + \sum_{\mathbf{n}} x(\mathbf{n})\, \mathbf{z}^{\mathbf{n}}, \qquad \mathbf{n} \in R_x - \{\mathbf{k}\}$$

where a particular point, **k**, has been isolated from the region of support of the sequence. Now multiply both sides by $\mathbf{z}^{-\mathbf{k}-\mathbf{1}}$ and integrate around N closed contours $C_1,C_2,...,C_N$ which circle the origin $\mathbf{z} = \mathbf{0}$ in the positive sense (counterclockwise):

$$\int_{C_1}(N)\int_{C_N} X(\mathbf{z})\, \mathbf{z}^{-\mathbf{k}-\mathbf{1}}\, d\mathbf{z} = \int_{C_1}(N)\int_{C_N} x(\mathbf{k})\, \mathbf{z}^{-\mathbf{1}}\, d\mathbf{z}$$

$$+ \int_{C_1}(N)\int_{C_N} \sum_{\mathbf{n}} x(\mathbf{n})\, \mathbf{z}^{\mathbf{n}-\mathbf{k}-\mathbf{1}}\, d\mathbf{z}, \qquad \mathbf{n} \in R_x - \{\mathbf{k}\} \tag{2.29}$$

where

$$\int_{C_1}(N)\int_{C_N}(\bullet)\, d\mathbf{z}$$

symbolizes the N-dimensional contour integral and $d\mathbf{z} = dz_1 dz_2 ... dz_N$. The first N-D integral on the RHS of eqn. (2.29) is simply

$$\int_{C_1}(N)\int_{C_N} x(\mathbf{k})\, \mathbf{z}^{-\mathbf{1}}\, d\mathbf{z} = x(\mathbf{k}) \prod_{i=1}^{N} \int_{C_i} z_i^{-1}\, dz_i$$

$$= x(\mathbf{k})\, (j2\pi)^N \tag{2.30}$$

by virtue of Cauchy's integral [7,p.81]. The second N-D integral on the RHS of eqn. (2.29) may be written as

$$\int_{C_1}(N)\int_{C_N}\sum_{\mathbf{n}} x(\mathbf{n})\,\mathbf{z}^{\mathbf{n}-\mathbf{k}-\mathbf{1}}\,d\mathbf{z} = \sum_{\mathbf{n}} x(\mathbf{n}) \int_{C_1}(N)\int_{C_N} \mathbf{z}^{\mathbf{n}-\mathbf{k}-\mathbf{1}}\,d\mathbf{z}$$

$$= \sum_{\mathbf{n}} x(\mathbf{n}) \prod_{i=1}^{N} \int_{C_i} z_i^{n_i-k_i-1}\,dz_i \qquad (2.31)$$

and since $n_i \neq k_i$, i=1,...,N (see eqn. (2.29)), the integrand on the RHS of eqn. (2.31) is either analytic within C_i (if $n_i > k_i$) or has a pole of multiplicity ≥ 2 (if $n_i < k_i$). In either case, the result is zero as a consequence of the residue theorem [8,p.172]. Substituting this result, and eqn. (2.30), back into eqn. (2.29) gives the required inversion formula:

$$x(\mathbf{k}) = (j2\pi)^{-N} \int_{C_1}(N)\int_{C_N} X(\mathbf{z})\,\mathbf{z}^{-\mathbf{k}-\mathbf{1}}\,d\mathbf{z} \qquad (2.32)$$

where the contours $C_1, C_2, \ldots, C_N$ all lie wholly within the region of convergence of $X(\mathbf{z})$ and encircle the origin. (The contours must lie within the region of convergence otherwise the last part of eqn. (2.31) would be invalid since the summations would not converge on each C_i).

2.6 The Multidimensional Fourier Transform

The main advantage in using the Z transform for the analysis of digital systems, as alluded to in the introduction to Section 2.5, is that it converts difference equations into algebraic equations, thereby simplifying the 'spadework' of the analysis. Such a method of attack is possible because the system response is deterministic and we have *a priori* knowledge regarding its characteristics. However, when we come to take the Z transform of a real-world signal, as is necessary, for example, to compute the output signal, we are forced to explicitly calculate the summation in eqn. (2.27) for each value of **z**; since for arbitrary $x(\mathbf{n})$, $X(\mathbf{z})$ will have no closed form. An efficient, and by now familiar, compromise is the Fourier transform.

2.6.1 Definition of the Fourier Transform

If for each z variable we have

$$z_i = \exp(\sigma_i + j\omega_i), \qquad i=1,\ldots,N \quad \text{and} \quad \omega_i \in (-\pi,\pi]$$

and the region of convergence of $X(\mathbf{z})$ includes the distinguished boundary of the unit polydisc (i.e. $\sigma_i = 0, \forall\, i$), then the Fourier transform of the sequence $x(\mathbf{n})$ is defined as

$$X(\omega_1,\omega_2,\ldots,\omega_N) = \sum_{n_1}\sum_{n_2}\cdots\sum_{n_N} x(n_1,n_2,\ldots,n_N)\, e^{j\omega_1} e^{j\omega_2} \cdots e^{j\omega_N}, \quad \mathbf{n} \in R_x$$

which may be written compactly as

$$X(\omega) = \sum_{\mathbf{n}} x(\mathbf{n})\, e^{j\omega}, \qquad \mathbf{n} \in R_x \tag{2.33}$$

where **e** is the N-tuple (e,e,...,e). Note that if X(**z**) exists (converges) on $\mathbf{T}^N$, then eqn. (2.33) exists and X(ω) converges in the entire ω–hyperplane, thus precluding a discussion on the region of convergence. X(ω) either exists or it does not exist.

2.6.2 Inversion of the Fourier Transform

If $X(\omega) < \infty$, then the Z transform X(**z**) necessarily converges on the distinguished boundary of the unit polydisc, and the original sequence x(**n**) may be recovered from the contour inversion integral (eqn. (2.32)):

$$x(\mathbf{n}) = (j2\pi)^{-N} \int(N)\int X(\mathbf{z})\, \mathbf{z}^{-\mathbf{n}-1}\, d\mathbf{z}, \qquad \mathbf{z} \in R_X$$

Since $R_X \supset \mathbf{T}^N$, we may integrate over the 'hyper-contour'

$$\zeta = \bigcap_{i=1}^{N} \zeta_i$$

where

$$\zeta_i = \{\, z_i:\ z_i = \exp(j\omega_i),\ \ \omega_i \in (-\pi,\pi] \,\}$$

and noting that on ζ: $d\mathbf{z} = j^N \mathbf{z}\, d\omega$ we have finally

$$x(\mathbf{n}) = (2\pi)^{-N} \int_{-\pi}^{\pi}(N)\int_{-\pi}^{\pi} X(\omega)\, e^{-j\mathbf{n}\cdot\omega}\, d\omega \tag{2.34}$$

where **n**•ω is the Hadamard product [3,p.54] of the N-tuples **n** and ω.

2.6.3 Approximate Evaluation of the Fourier Transform

The most profitable advantage in using the Fourier transform over the Z transform is that there is a fast, efficient and compact method for computing a very close approximation (exact in special cases) to the former: the so-called fast Fourier transform (FFT). The detailed mechanics of the

FFT have been thoroughly and extensively documented elsewhere, for example [2,p.74],[3,p.80],[4,p.356],[9,p.390],[10,p.78] and especially [11,Chap.4], so will not be presented here. In most work, the use of the FFT in approximating the 'real' Fourier transform yields such reasonable results, we will assume that we have available exact samples of $X(\omega)$.

2.7 The Multidimensional Walsh-Hadamard Transform

The Walsh-Hadamard transform differs from the Z transform (and its simplified version, the Fourier transform), in that it can not be used as a tool for the solution of difference equations; mainly because the Walsh basis functions are not eigen functions of finite difference equations with constant coefficients (unless the samples are taken at instants of dyadic time [15]). The sole reason for introducing the Walsh-Hadamard transform at this stage is because it will be used in Chapter 3, where its primary advantages – that of orthogonality and speed of computation – will be exploited effectively. This is precisely the reason why the Hadamard transform has been sucessfully applied to signal and image-coding problems in the past, for example [10,p.282],[12].

2.7.1 Definition of the Walsh-Hadamard Transform

Not surprisingly, the Walsh-Hadamard transform has Walsh functions as its set of basis functions [11,p.303]. Unlike the Z or Fourier transform kernels, the Walsh functions are discontinuous in nature, but do in fact form an orthogonal set over a specified interval [13] – usually [0,1). The first 8 Walsh basis functions are shown in Figure 2.6 for the 1-D case. *Sequency* [11,p.304] refers to the number of sign changes in the interval [0,1) and is analogous to the familiar notion of frequency for sinusoidal signals. Note that in Figure 2.6 the sequency of the basis functions increases uniformly, giving the so-called Walsh or sequency ordering. This is not the only possible ordering scheme; for example, the functions may be arranged in Paley (dyadic) (see Figure 2.7), Hadamard (natural) or cal-sal, order [11,p.306]. However, for a lot of engineering work, it makes sense to adopt a sequency ordered transform as this choice agrees with our notions about frequency increasing as a point moves away from the transform-domain origin.

The Walsh-Hadamard transform of a sequence $\mathbf{x(n)}$ is given by

$$\mathbf{X(k)} = \sum_{\mathbf{n}} \mathbf{x(n)\ wal(k,n)}\,, \qquad \mathbf{n} \in R_x \ \text{ and } \ \mathbf{k} \in R_w \tag{2.35}$$

where R_w is the smallest hyper-cube containing R_x. The index $\mathbf{k}$ locates a particular basis function in R_w and each basis function has been normalised so that its domain of definition is [0,K), where K is length of one side of the hyper-cube, R_w. The notation $\mathbf{wal(k,n)}$ should be interpreted as the

outer product

$$\mathbf{wal(k,n)} = \prod_{i=1}^{N} \mathrm{wal}(k_i,n_i) \tag{2.36}$$

where, as usual, N is the number of independent dimensions of $\mathbf{x(n)}$. In order to illustrate the form taken by eqn. (2.36) for the case N > 1, Figure 2.8 shows the first 4 by 4 basis functions of the 2-D Walsh-Hadamard kernel. Note that the general rapidity of sign changes increases as a point moves, in any direction, away from the zero-sequency point at top left.

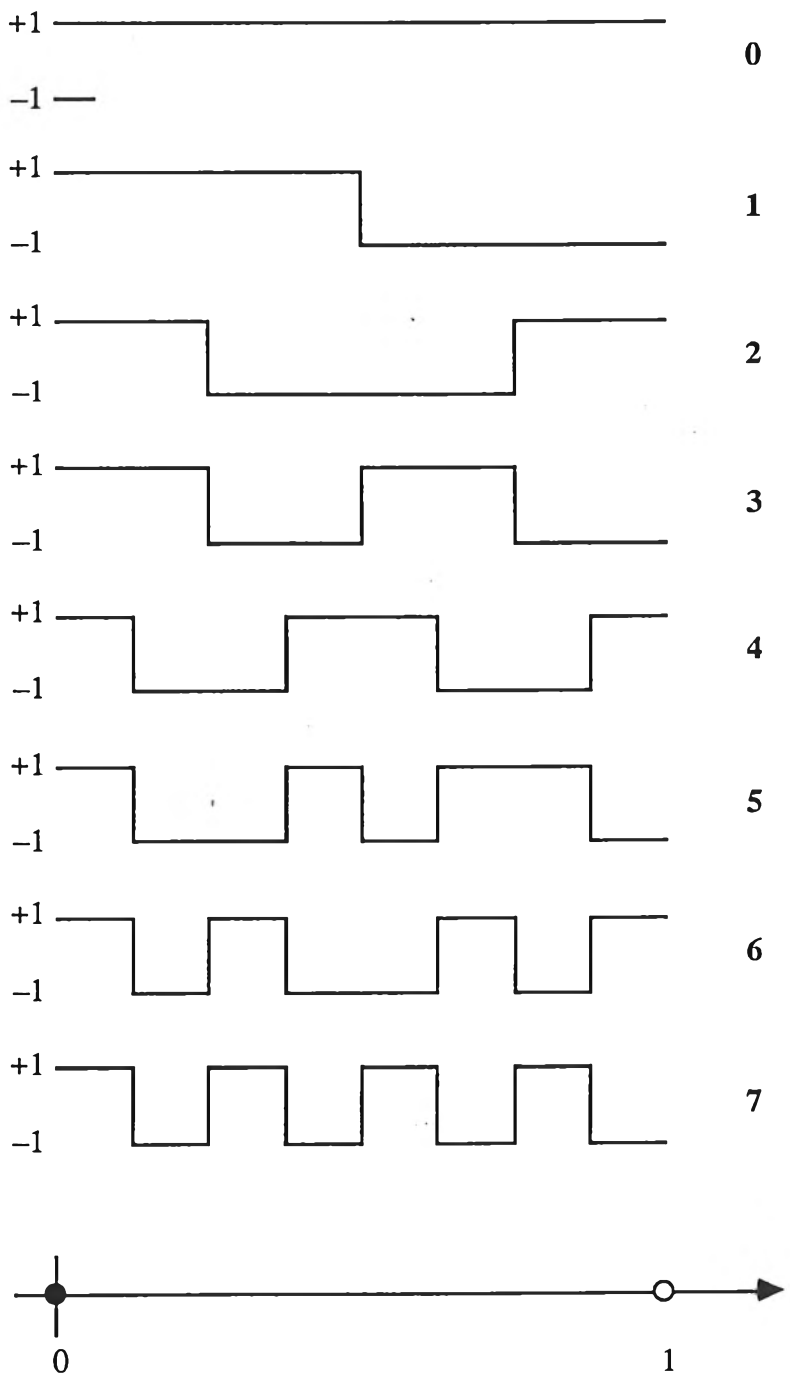


Figure 2.6: *First 8 1-D Walsh basis functions with Walsh or sequency ordering.*

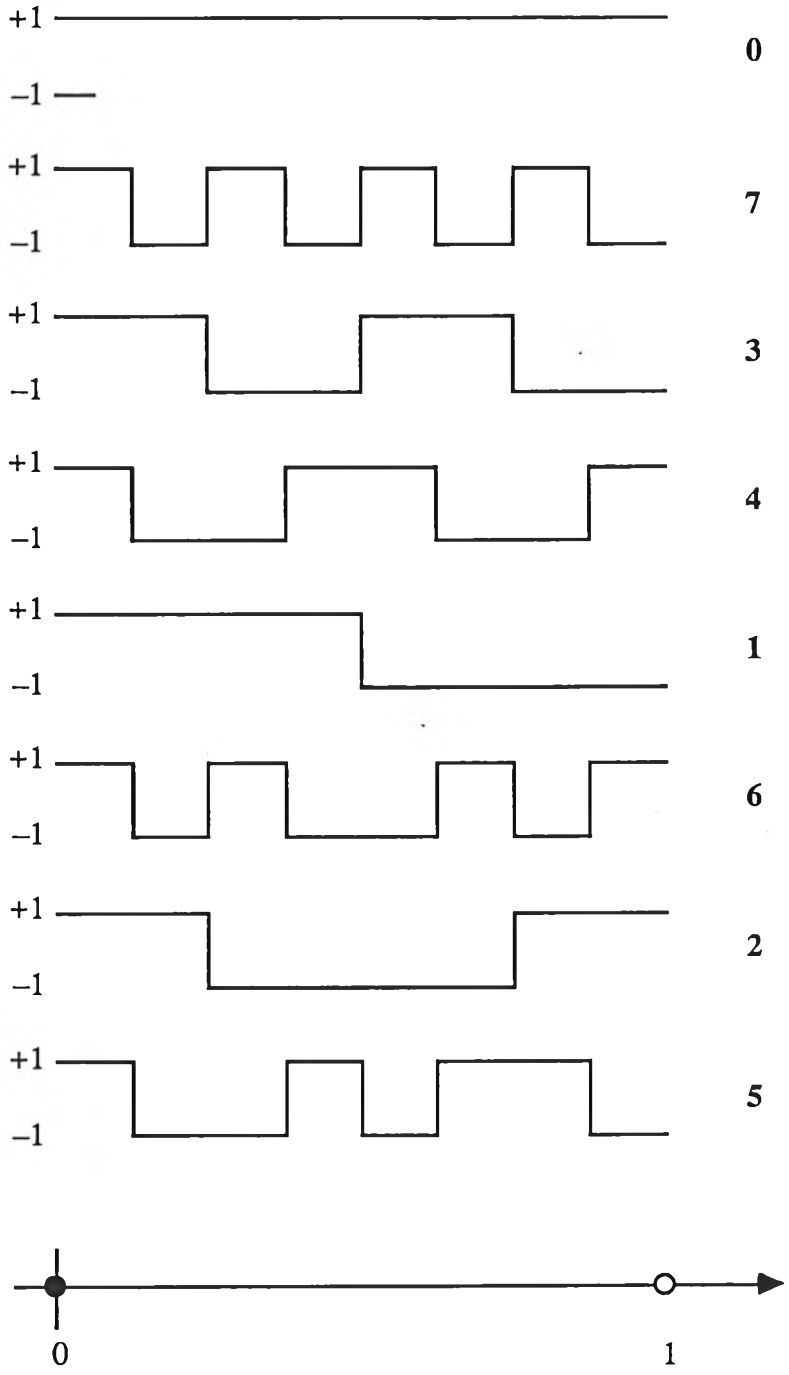


Figure 2.7: *First 8 1-D Walsh basis functions with Paley or dyadic ordering.*

2.7.2 Inversion of the Walsh-Hadamard Transform

The Walsh-Hadamard transform has an orthonormal kernel matrix and is therefore its own inverse. Consequently, given the transform $X(\mathbf{k})$, a second application of eqn. (2.35) recovers $x(\mathbf{n})$ to within a constant scale factor.

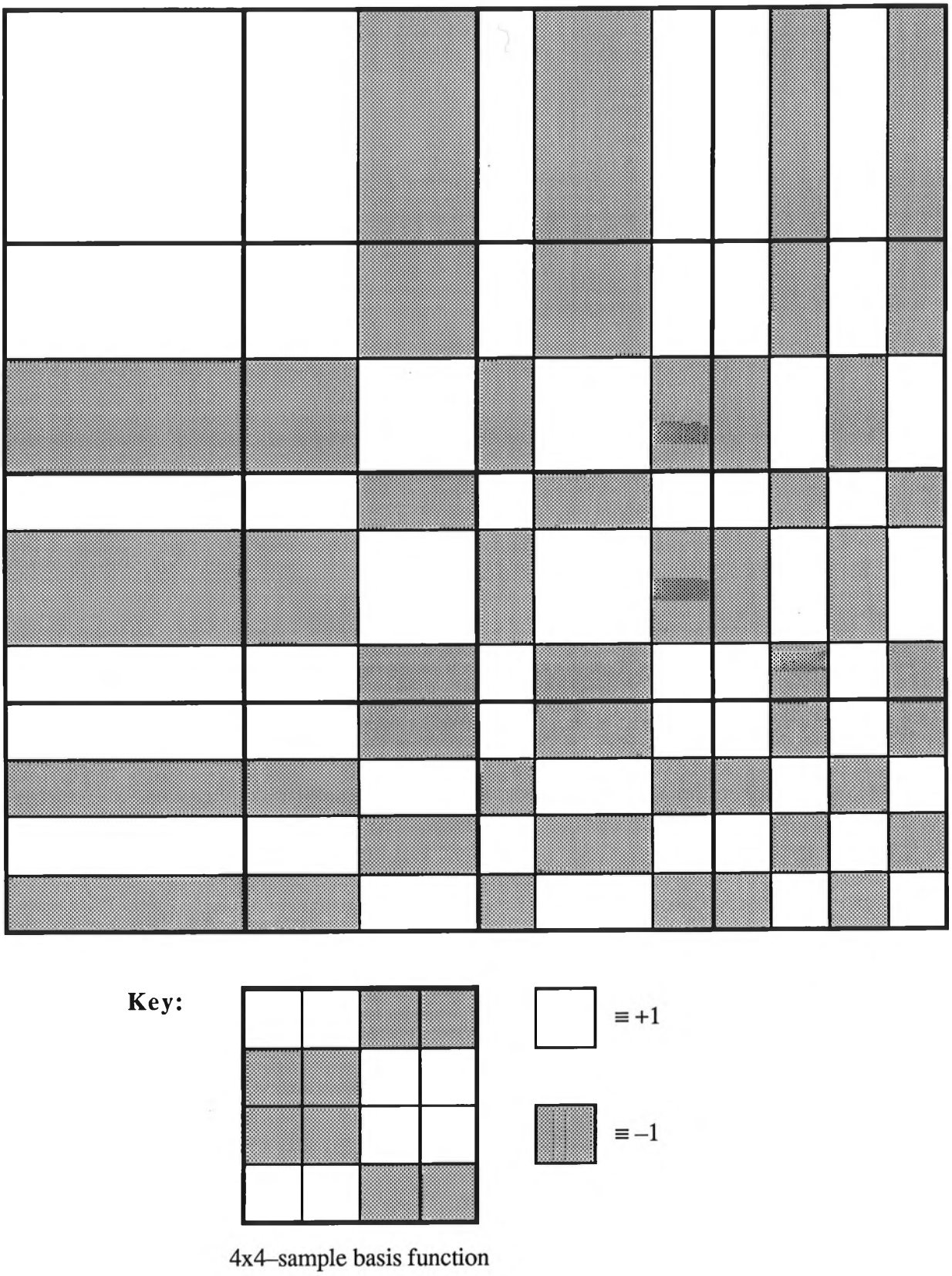


Figure 2.8: *First 4 by 4 2-D Walsh basis functions with sequency ordering. Zero sequency is in the top left hand corner.*

2.7.3 Computation of the Walsh-Hadamard Transform

There are several 'divide and conquer' strategies, very similar to the FFT, available for computing the Walsh-Hadamard transform of a discrete signal, for example [11,p.310],[14]. Consequently the mechanics of efficiently evaluating eqn. (2.35) will not be pursued, and instead, the reader is directed to the two previous references for detailed information.

2.8 References

1 **R Narasimhan**: 'Several complex variables', The University of Chicago Press, 1971.

2 **DE Dudgeon** and **RM Mersereau**: 'Multidimensional digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

3 **P Henrici**: 'Applied and computational complex analysis. Vol. 3', Wiley, New York,1986.

4 **LR Rabiner** and **B Gold**: 'Theory and application of digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

5 **BC Kuo**: 'Digital control systems', Holt-Saunders, 1980.

6 **EI Jury**: 'Sampled-data control systems', Wiley, New York, 1958.

7 **EC Titchmarsh**: 'The theory of functions', Oxford University Press, 2nd Ed., 1939.

8 **RV Churchill, JW Brown** and **RF Verhey**: 'Complex variables and applications', 3rd Ed., McGraw-Hill, 1974.

9 **WH Press, BP Flannery, SA Teukolsky** and **WT Vetterling**: 'Numerical recipes – the art of scientific computing', Cambridge University Press, New York, 1986.

10 **RC Gonzalez** and **P Wintz**: 'Digital image processing', Addison-Wesley, Massachussetts, 1977.

11 **DF Elliott** and **KR Rao**: 'Fast transforms – algorithms, analyses, applications', Academic Press Inc., 1980.

12 **WK Pratt, J Kane** and **HC Andrews**: 'Hadamard transform image coding', *Proc. IEEE*, Vol. 57, No. 1, 1969, pp.58-68.

13 **JL Shanks**: 'Computation of the fast Walsh-Fourier transform', *IEEE Trans. Computers*, May 1969, pp.457-459.

14 **JW Manz**: 'A sequency-ordered fast Walsh transform', *IEEE Trans. Audio and Acoustics*, Vol. AU-20, No. 3, 1972, pp.204-205.

15 **JE Gibbs**: 'Discrete complex Walsh Functions', *Proc. of Symp. and Workshop on Applications of Walsh Functions*, Naval Res. Lab., Washington DC, March-April, 1970.

Image Coding via Linear Prediction: Practice and Problems

3.1 Introduction

Recursive filtering is an implicit operation in any linear prediction application. Unlike the most general form of recursive filter, with both non-constant numerator and denominator polynomials, linear prediction filters possess only singularities. A desirable consequence of a constant numerator polynomial is that the determination of the denominator coefficients is a linear procedure, whereas the design of a general recursive filter is a non-linear problem. This makes linear prediction a prime candidate for coding applications where it is desirable to not only model a signal accurately, but to solve the modelling problem rapidly.

A simple theory of linear prediction is presented in Section 3.2. It is shown that determining the optimal set of prediction coefficients, whether based on deterministic or stochastic grounds, involves solving a set of simultaneous linear equations. Linear prediction applied to image coding is discussed in Section 3.3, as well as a short overview of other coding techniques: notably waveform and transform coding. This background material sets the scene for the rest of the chapter, which is concerned with certain stability problems encountered in 2-D LPC implementations. Section 3.5 discusses the causes of instability in the LP model and Section 3.6 investigates four related methods for reducing the instability rate. The effectiveness of each method is examined, followed by a cost analysis, based on algorithm complexity and bit-rate overhead. Finally, the appendix to this chapter discusses an improvement to a relatively new hybrid coding scheme based on directional filters.

3.2 Linear Prediction Theory

At the heart of linear prediction is the assumption that the value of a particular sample, $x(\mathbf{n})$, may be estimated from a linear weighted sum of samples in the neighbourhood of the point $\mathbf{n}$, (the idea of a neighbourhood will be be made more precise shortly) and possibly some input (or excitation), $u(\mathbf{n})$. Figure 3.1 illustrates such a model. Using Figure 3.1 it is possible to deduce that the

difference equation relating x(**n**) and u(**n**) must be of the form

$$x(\mathbf{n}) = \sum_{\mathbf{k}} a(\mathbf{k})\, x(\mathbf{n}-\mathbf{k}) + u(\mathbf{n}), \qquad \mathbf{k} \in R_a \tag{3.1}$$

where R_a specifies the region of support of the prediction coefficients, a(**k**), and is usually called the *prediction mask*. The specific shape of R_a is quite important in certain situations, dramatically affecting the model performance, and is briefly discussed in Section 3.2.3.

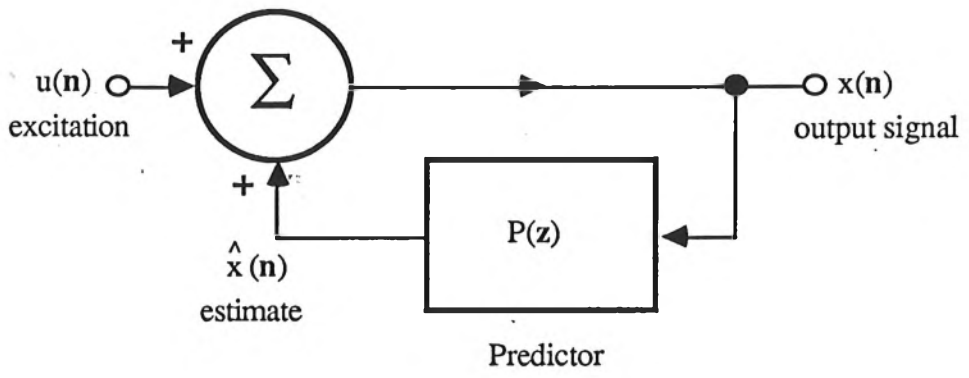


Figure 3.1: *Autoregressive model for generating the N-dimensional signal x(**n**).*

Simplistically, the linear prediction problem is solved once an optimal (in some sense) set of prediction coefficients have been found. In the usual engineering way, we define, and then minimise, an error criterion which gives us the 'best' prediction coefficients. However, the particular path toward an optimal solution depends on our philosophical interpretation of eqn. (3.1). That is, x(**n**) may be viewed as a completely deterministic signal, or instead, as a single realization of a stochastic process.

3.2.1 Deterministic Scenario

If x(**n**) is viewed as completely deterministic, then eqn. (3.1) may be interpreted as a prediction, based on neighbouring samples:

$$\hat{x}(\mathbf{n}) = \sum_{\mathbf{k}} a(\mathbf{k})\, x(\mathbf{n}-\mathbf{k}) \tag{3.2}$$

and a correction term, u(**n**), which must be added to $\hat{x}(\mathbf{n})$ in order to achieve the correct value x(**n**). If we now form the error sequence

$$e(\mathbf{n}) = x(\mathbf{n}) - \hat{x}(\mathbf{n}) \tag{3.3}$$

it is usual to adjust the unknown parameter set {a(**k**)} until the mean squared error has been

minimised:

$$E = \frac{1}{\eta(R_e)} \sum_{\mathbf{n}} e^2(\mathbf{n}), \qquad \mathbf{n} \in R_e \tag{3.4}$$

R_e specifies the region over which E is minimised. Using simple calculus, it is easy to show that

$$\frac{\partial E}{\partial a(\mathbf{p})} = \frac{2}{\eta(R_e)} \sum_{\mathbf{k}} a(\mathbf{k}) \sum_{\mathbf{n}} x(\mathbf{n}-\mathbf{k})\, x(\mathbf{n}-\mathbf{p}) - \frac{2}{\eta(R_e)} \sum_{\mathbf{n}} x(\mathbf{n})\, x(\mathbf{n}-\mathbf{p}),$$

$$\mathbf{n} \in R_e \text{ and } \mathbf{k},\mathbf{p} \in R_a \tag{3.5}$$

and defining

$$\phi(\mathbf{k};\mathbf{p}) = \sum_{\mathbf{n}} x(\mathbf{n}-\mathbf{k})\, x(\mathbf{n}-\mathbf{p}), \qquad \mathbf{n} \in R_e \tag{3.6}$$

then $\partial E/\partial a(\mathbf{p})$ is zero for

$$\sum_{\mathbf{k}} a(\mathbf{k})\, \phi(\mathbf{k};\mathbf{p}) = \phi(\mathbf{0};\mathbf{p}), \qquad \mathbf{k},\mathbf{p} \in R_a \tag{3.7}$$

Equation (3.7) is linear in $a(\mathbf{k})$ and may be solved using any one of the standard techniques; however, since $\phi(\mathbf{k};\mathbf{p})$ is symmetric, $a(\mathbf{k})$ may be efficiently obtained via Cholesky decomposition [1,p.24]. For example, consider the 3 dimensional case with the simplest prediction mask geometry shown in Figure 3.2. In this case eqn. (3.7) becomes

$$\sum_{k_1} \sum_{k_2} \sum_{k_3} a(k_1,k_2,k_3)\, \phi(k_1,k_2,k_3;p_1,p_2,p_3) = \phi(0,0,0;p_1,p_2,p_3)$$

$$(k_1,k_2,k_3),(p_1,p_2,p_3) \in R_a \tag{3.8}$$

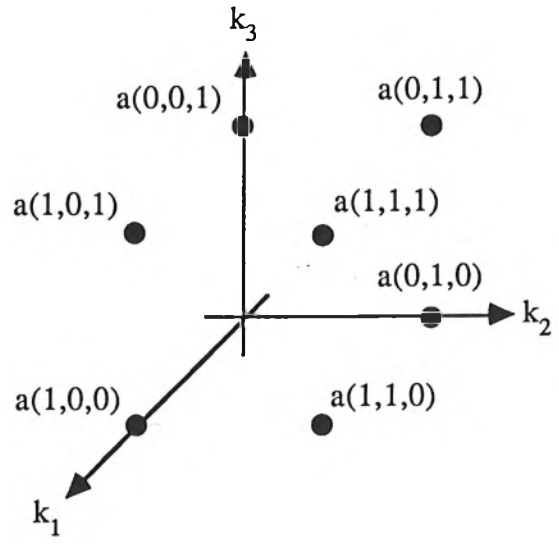


Figure 3.2: *3-D prediction mask with support in the generalised first quadrant.*

3.2.2 Stochastic Scenario

If the input signal u(**n**) and output signal x(**n**) are treated as random variables, then the determination of the parameters a(**k**) usually proceeds as follows. Multiply eqn. (3.1) by x(**n**–**m**) and take expected values

$$E\{x(\mathbf{n})\, x(\mathbf{n}-\mathbf{m})\} = \sum_{\mathbf{k}} a(\mathbf{k})\, E\{x(\mathbf{n}-\mathbf{k})\, x(\mathbf{n}-\mathbf{m})\} + E\{u(\mathbf{n})\, x(\mathbf{n}-\mathbf{m})\}$$

and if we further assume that x(**n**) is stationary and that the input is uncorrelated with the output, then the expected values depend only on the difference between samples:

$$r(\mathbf{m}) = \sum_{\mathbf{k}} a(\mathbf{k})\, r(\mathbf{m}-\mathbf{k}) \tag{3.9}$$

where r(**m**) is the autocorrelation function of x(**n**) for a lag of **m** samples. Equation (3.9) is linear in the parameters a(**k**), and since r(**m**) is symmetric about **m** = **0**, the a(**k**) values could be found using Cholesky's method. However, our assumption of stationarity imposes a Toeplitz structure on the matrix version of eqn. (3.9). Consequently, the unknown parameters may be found using the computationally efficient Levinson-Durbin algorithm [2,p.47]. For example, for the simple 2-D predictor shown in Figure 3.3, eqn. (3.9) has the form

$$r(m_1,m_2) = \sum_{k_1} \sum_{k_2} a(k_1,k_2)\, r(m_1-k_1, m_2-k_2) \tag{3.10}$$

which in matrix form becomes

$$\begin{bmatrix} r(0,0) & r(1,-1) & r(0,-1) \\ r(-1,1) & r(0,0) & r(-1,0) \\ r(0,1) & r(1,0) & r(0,0) \end{bmatrix} \begin{bmatrix} a(1,0) \\ a(0,1) \\ a(1,1) \end{bmatrix} = \begin{bmatrix} r(1,0) \\ r(0,1) \\ r(1,1) \end{bmatrix}$$

In the case of images, x(**n**) is often only quasi-stationary and we usually have no *a priori* knowledge regarding the true autocorrelation function. In this case, we are forced to estimate the autocovariance samples from the signal (eqn. (3.6)) and therefore cannot take advantage of the Levinson-Durbin algorithm. This point is elaborated in Section 3.3.3: *LP Analysis.*

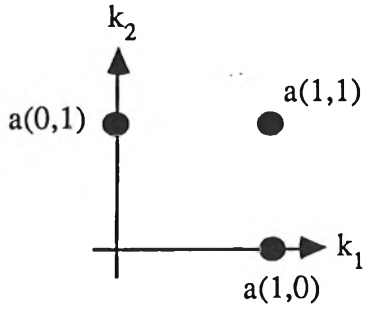


Figure 3.3: *2-D prediction mask with support in the first quadrant.*

3.2.3 Prediction Mask Geometries

Figure 3.2 illustrates the quarter-plane (QP) prediction mask, so called because non-zero mask values only exist in the generalized first quadrant. The 2-D QP mask is very popular in the literature but is by no means the only possible geometry; the QP and other major mask shapes are depicted in Figure 3.4. The QP mask has received most attention to date, probably because it is most closely related to the 1-D autoregressive model, although 2-D IIR filter designers appear to be favouring the non-symmetric half-plane mask. (See, for example: Chapter 5, references [5],[7], [8],[9],[13]). For certain applications, such as 2-D spectral estimation, mask geometry plays an important role in determining the shape of the power spectral density function [3,pp.325-339], [4],[5,p.456]. However, for image coding work, results relating mask shape to image quality appear to be non-existant. Also, to the authors knowledge, no quantitative results have been reported regarding mask geometries for dimensions greater than two.

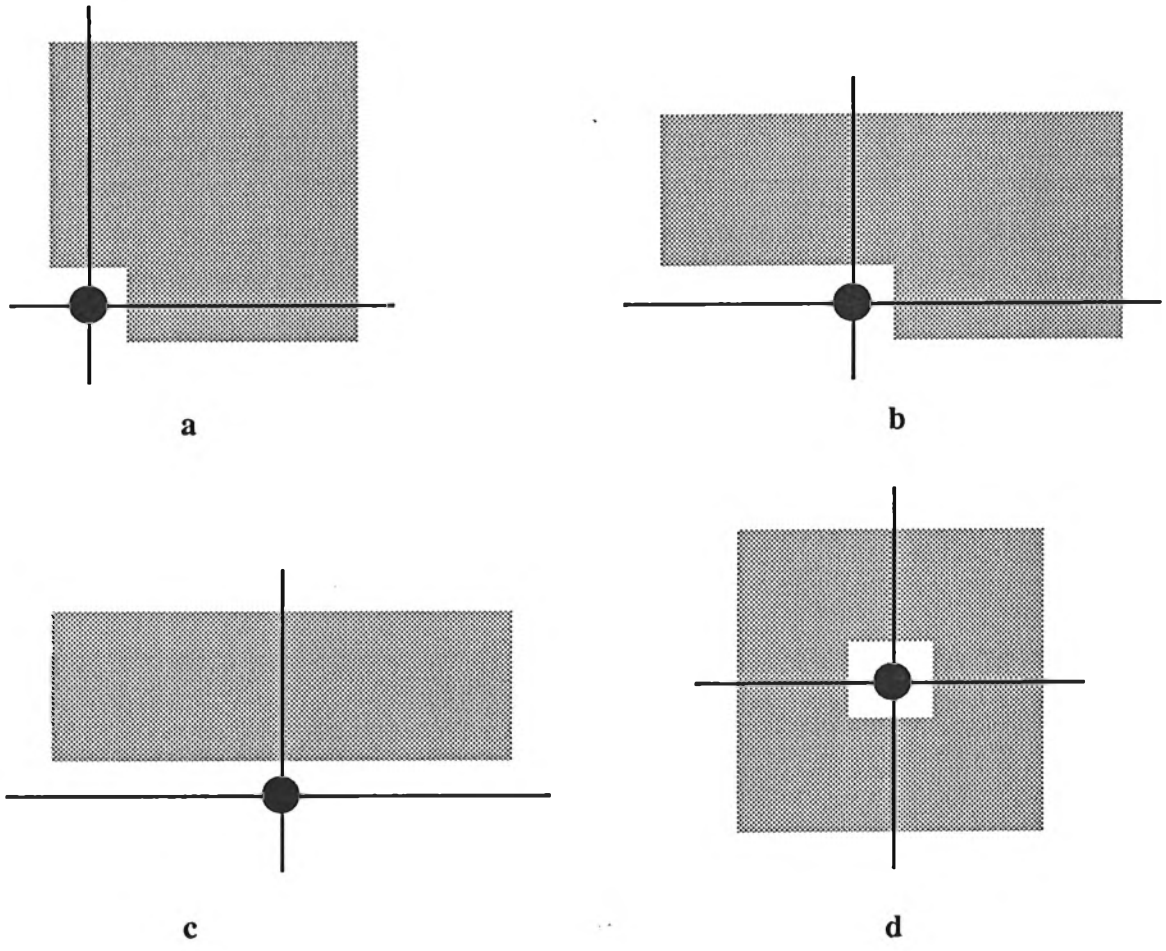


Figure 3.4: *Prediction mask geometries*

a *Quarter-plane (QP).*

b *Non-symmetric half-plane (NSHP).*

c *Symmetric half-plane (SHP).*

d *Full-plane (FP). [Non-causal].*

3.3 Application to Image Coding

The objective of this section is to explain how linear prediction is integrated into an image coding system. Given the considerable amount of information present in an image, it is not surprising that some form of compression is desirable. For example, a typical image processing application using grey-scale data may consist of 512 picture lines, each line containing 512 pixels. Using the almost universal quantization value of 8 bits/pixel gives 2,097,152 bits of information. Furthermore, if each of the above images is transmitted at the normal TV rate of 30 frames per second, a digital bandwidth of 62.9 Mbits/second is required. As another example, restoration of static electronmicroscope images often involves digitizing each picture (via a scanning microdensitometer) to a resolution of 7000 by 8000 pixels, with each pixel quantized to 10 bits – this represents a staggering 560 Mbits of data!

The first example requires some form of image compression simply to reduce the rather expensive bandwidth requirements (both technologically and economically) of a digital TV transmission system. The second example cries out for data compaction for essentially the same reason: that of reducing the cost of storage media for these very highly sampled microscope pictures. Even using state-of-the-art technology (circa 1988), only 8 electronmicroscope images would fit on a 600 MByte magneto-optical disk[6].

The two simple examples above highlight the pressing need for some form of data compression for images, whether for storage or transmission, given todays current resource restrictions. However, it could be cogently argued that even if storage/transmission resources far exceeded current demand, applications would soon emerge which demanded the maximum available bandwidth or storage densities. Hence, it appears that there will always be a need for information reduction techniques. In order to 'set the scene', so that linear prediction may be viewed in its correct place relative to other signal coding methods, we first describe a typical digital communication system.

3.3.1 Model of a Digital Transmission or Storage System

The basic components of a digital communication system are shown in Figure 3.5. Despite the fact that the diagram is obviously that of a transmission system, it is equally applicable to information storage applications if the 'channel' block (far RHS of Figure 3.5) is replaced with an appropriate storage medium – such as a disk.

Source

Our present interest dictates that the source must be an image. The image may be anything from a TV frame to a still photograph, although for convenience we assume that the picture has already been digitized, typically to 512x512 pixels of spatial resolution with 8 bits/pixel intensity resolution.

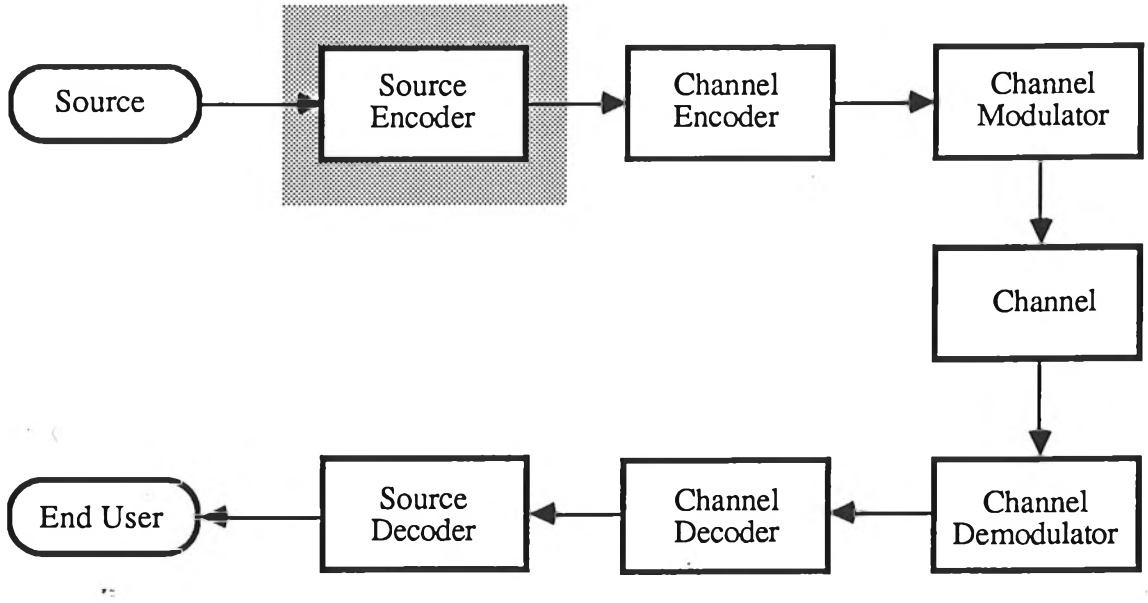


Figure 3.5: *Block diagram of a general digital communication system. (After Proakis, [18,p.59])*

Source Encoder

The source encoder is really the focus of this entire chapter. The purpose of source encoding is to reduce the number of bits which must be transmitted or stored, and yet still represent the source image to within a specified tolerance, given some fidelity criteria. As far as image coding is concerned, the most prevalent forms of source encoding are waveform coding (such as PCM,ADM), transform coding (notably Fourier, Cosine and Hadamard) and parametric modelling (LPC in particular). Contemporary waveform and transform coding techniques will be examined in Section 3.3.2, and problems peculiar to 2-D linear prediction consume the remainder of the chapter.

Channel Encoder

Unlike the source encoder, which attempts to minimise information content by reducing redundancy in the image signal, channel encoding usually results in an increase in data in its efforts to reduce the susceptability of the system to channel errors. In its simplist form this may be a parity bit, progressing towards block convolutional or cyclic codes in more advanced systems. In any case, the aim is to add a small amount of redundant information, so that in the event of channel errors corrupting the coded signal, the channel decoder has enough information to reconstruct (or at least detect) corrupted bits in the data stream.

Channel Modulator

The channel modulator is really a buffering unit which translates the characteristics of the data stream being emitted by the channel encoder, to a form compatible with the characteristics of the channel. The channel modulator does not alter the information content of the signal. For example, if the channel is actually a floppy diskette, then the serial bit stream from the channel encoder is used to frequency modulate the orientation of magnetic domains in tracks on the disk. Similarly, if the channel is a microwave link, then the baseband channel encoder output must be frequency translated to an appropriate band via, say, amplitude modulation.

Channel

By now it is obvious that the channel is any medium through which the modulator output signal passes. Examples from communication systems include free space (e.g. microwave, RF, fibreoptic transmission), twisted-pair cabling (e.g. telephone networks), coaxial cabling (e.g. packet networks such as Ethernet) while examples of storage technologies include paper (e.g. punched paper-tape, still used in military establishments), magnetic materials (e.g. floppy and hard disks, core memories – still used in military aircraft), solid-state materials (e.g. dynamic or static RAMs).

Channel Demodulator, Channel Decoder and Source Decoder

The last three blocks before the end-user in Figure 3.5 are required to unravel the various layers of encoding prior to storage or transmission. The channel demodulator restores the channel signal to its basenband form – a stream of binary digits. As noted previously, the channel decoder is responsible for restoring or detecting any lost information, due to channel errors, using the redundant information appended to the bit-stream by the channel encoder. Finally, the source decoder, with a knowledge of the encoding scheme used at the transmitter, reconstructs the original source message, which in this case happens to be an image.

3.3.2 Source Coding using Waveform and Transform Coding

Before beginning a discussion on linear prediction and its application to signal coding schemes, it is useful to first review other image coding strategies, specifically waveform and transform coding, as well as a recent innovation: a hybrid method involving waveform modelling and transform techniques.

3.3.2.1 Waveform coding [19]

Waveform coding involves exactly what the name implies: individual samples of the signal (in this case a discrete image) are coded prior to storage or transmission (See Figure 3.6). Familiar

examples include Pulse Code Modulation (PCM), Logarithmic PCM (LogPCM), Differential PCM (DPCM) and Delta Modulation (DM).

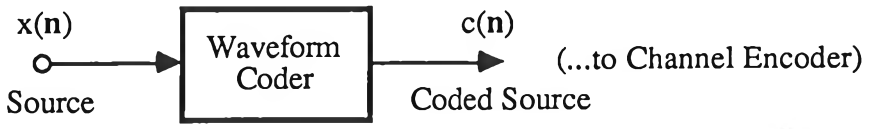


Figure 3.6: *Source encoder using waveform coding.*

For PCM, the input and output of the coder are related via

$$c(n) = Q[x(n)] \tag{3.11}$$

where Q is an operator which quantizes the amplitude of x(**n**) to one of several discrete levels (typically 256). Thus, with PCM, no data compaction takes place other than that which coincidently occurs during the quantization process.

In an attempt to reduce the dynamic range occupied by the signal, LogPCM compresses the signal amplitude logarithmically. This is usually accomplished using a non-linear quantizer, where the quantization intervals are related logarithmically, giving

$$c(n) = Q[\log\{x(n)\}] \tag{3.12}$$

The simplest technique which exploits the favourable statistical characteristics of the signal is DPCM. Many discrete signals possess autocorrelation functions which remain relatively constant for lags of several samples. DPCM exploits this feature by coding the difference between successive samples rather than the signal samples themselves. It is fairly straightforward to prove that the variance of the error signal (and hence the dynamic range of the transmitted signal) is less than the variance of the original signal. The simplified input–output relationship is

$$c(n) = Q[x(n)-x(n-1)] \tag{3.13}$$

Delta Modulation is a simplification of DPCM in that the error signal is quantized to one of two levels, thus only occupying 1 bit of data. Obviously such coarse quantization introduces marked distortion in the reconstructed signal if the rate of change of source signal is too great. Consequently, in order to minimise distortion, DM implementations usually feature very high sampling rates.

3.3.2.2 Transform coding

Transform coding may be effected with virtually any orthogonal transform although, historically, researchers have concentrated on the Fourier, Walsh, Cosine, and very recently the Affine [20], transforms. When discrete transforms are used, the transform of the signal always possesses an identical number of degrees of freedom. Data compression is possible because most of the image energy occupies a fairly small region in the transform domain. By truncating the transform outside this region, and transmitting only those components which are non–zero, a bit–rate reduction is achieved. To illustrate this idea, consider the Fourier transform of a grey–scale image, typically characterised by a very narrowband spectrum centred on the origin. If those Fourier components are retained which contribute to, say, 99% of the image energy (usually a very small region of pixels) then a considerable compression ratio is obtained.

Unfortunately, most transform coding techniques are plagued by distortions in image reconstruction as a result of the truncation process. For example, images reconstructed from truncated Fourier components appear blurred because, although the 'edge energy' occupies the lower few percent of the power spectrum, edges provide important clues for human visual perception. A slightly different problem arises when using Walsh basis functions to code images. Since the basis functions remain constant between discontinuities, the truncated transform results in a reconstructed image which appears decidedly 'blocky' in nature. A slightly more formal description of transform coding follows.

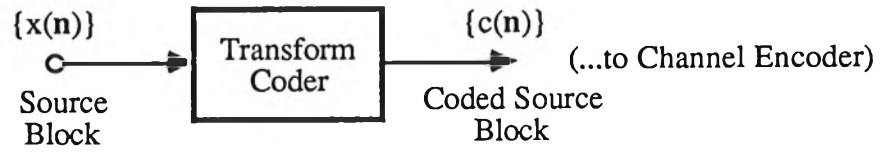


Figure 3.7: *Source encoder using transform coding.*

It should be obvious by now that transform coding cannot proceed on a sample by sample basis, hence Figure 3.7 is meant to convey the idea of successive blocks of source messages being transform coded. If the region of support, R_x, of the signal $x(\mathbf{n})$ is small, then the entire image may be coded in one fell swoop. However, if R_x is extensive (or infinite if a dimension of $\mathbf{n}$ is temporal) then $x(\mathbf{n})$ must be segmented into conveniently sized blocks. We partition R_x into blocks B_k:

$$R_x = \bigcup_{k=0}^{\infty} B_k \qquad \text{with} \qquad B_i \cap B_j = \emptyset \quad \forall \; i,j:\; i \neq j \tag{3.14}$$

where it is seen that the partition must be exhaustive and adjacent blocks must not overlap. A particular source block is denoted

$$S_x^k = \{x(n):\ n \in B_k\} \tag{3.15}$$

and S^k_x is transformed via

$$S_y^k = TS_x^k = \{y(n):\ n \in B_k\} \tag{3.16}$$

where T is an unspecified orthogonal discrete transform. As it stands, S^k_y has the same region of support as S^k_x and thus represents zero compression. In order to compact the data, the coded source block is derived as

$$S_c^k = \Omega S_y^k \tag{3.17}$$

where the operator Ω is simply a rule for assigning elements of S^k_x to S^k_y such that

$$\eta(S_c^k) < \eta(S_y^k) \tag{3.18}$$

The transformation in eqn. (3.16) is information preserving, whereas the requirement in eqn. (3.18) implies a loss of information. Hence, in contrast to waveform coding, which in principle may be an information lossless procedure, transform coding *must* involve some form of distortion, and can therefore only be used in situations where such distortion can be tolerated.

It may not be obvious why one should transform the source block at all, given that signal fidelity is immediately compromised via eqn. (3.18). The reason is that the transformation is actually performing part of the task of the channel encoder. This can be illustrated most easily by considering the 1-D discrete Fourier transform of the signal x(n):

$$X(k) = \sum_n x(n) \exp(-j2\pi kn/N), \qquad n \in B \tag{3.19}$$

where B=[0,N). The signal x(n) is projected onto the N basis functions exp(–j2πkn/N) and the amplitude of one particular basis function depends on all N samples of x(n). Thus x(n) is 'smeared' throughout the transform domain. In order to satisfy eqn. (3.18) we truncate the set B to obtain the region of support of the source code block

$$D = [0,M), \quad M < N \tag{3.20}$$

and recover an *estimate* of x(n) using the inverse transform

$$x_{est}(n) = \sum_k X_{trunc}(k) \exp(j2\pi kn/N), \qquad k \in B \tag{3.21}$$

where

$$X_{trunc}(k) = \begin{cases} X(k) & \text{if } k \in D \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

Obviously, distortion is present because $B \supset D$. However, channel errors in the transmitted samples $X_{trunc}(k)$ contribute a smaller error to $x_{est}(n)$ because the information contained in x(n) is distributed throughout {X(k)}.

3.3.2.3 Image coding via directional filtering

Transform and waveform coding suffer from deficiencies which effectively place an upper limit on the compression ratios which may be obtained. The reason is that neither technique responds well to the presence of sharp discontinuities (i.e. edges) in the signal intensity. Waveform coding performs poorly because the bit-rate must increase to compensate for the extra 'information' (for example, the variance of the error signal in DPCM must increase), while transform coding fairs no better because it simply ignores the high frequency content of the signal – exactly where the edge information is contained. Image coding via directional filtering attempts to alleviate these problems by using a dual coding strategy: the high and low frequency components of the signal are coded separately. Although the idea of a hybrid coding scheme has been suggested previously, the technique developed by Ikonomopoulos and Kunt [7][8] shows considerable promise and will be examined briefly in this section.

Motivated by the knowledge that edges, vitally important to visual perception, are not well catered for by classical coding methods, our aim is to detect and code the edges independently of other information in the image. The method adopted in [7] is to detect edges of a particular orientation using a directionally tuned filter such as the one shown in Figure 3.8.

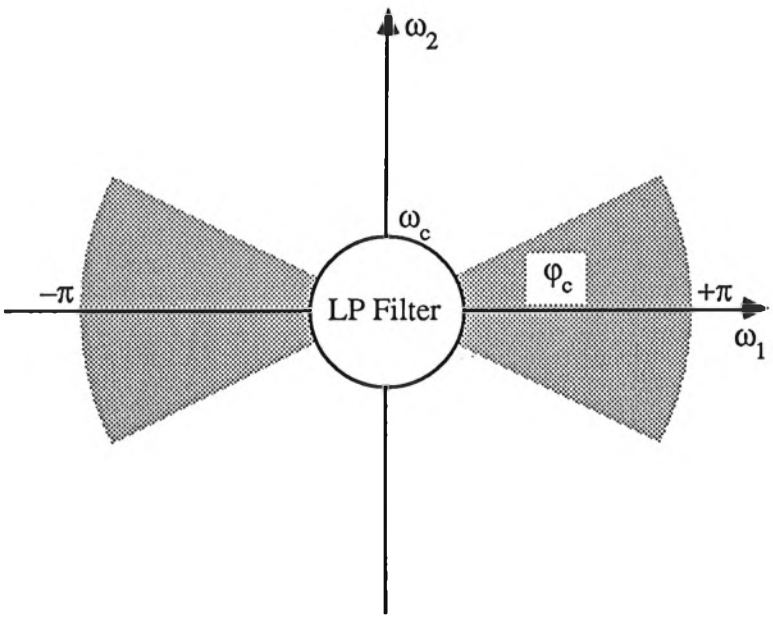


Figure 3.8: *Ideal frequency response of the directionally-tuned filter.*

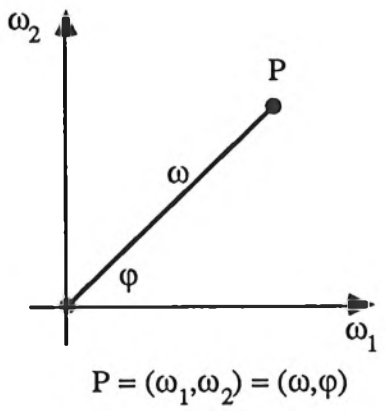


Figure 3.9: *Specifying a 2-D frequency in polar coordinates.*

The frequency response is most easily expressed in polar coordinates (refer to Figure 3.9) as

$$H_0(\omega_1,\omega_2) \equiv H_0(\omega,\varphi) = \begin{cases} 1 & (\omega,\varphi) \in R_H \\ 0 & \text{otherwise} \end{cases} \tag{3.23}$$

where R_H is the region of support of the filter:

$$R_H = \{ (\omega,\varphi): \ \omega_c \le \omega \le \pi, \ |\varphi| \le \varphi_c \} \tag{3.24}$$

The *principal direction* of the filter is 0 radians and the *orthogonal direction* is $\pi/2$ radians. φ_c is the angular bandwidth of the filter. It is well known that for an edge of a particular orientation in the spatial domain, the energy in the edge is distributed in the frequency domain in a direction orthogonal to the original edge. Thus the ideal filter frequency response shown in Figure 3.8 will detect edges orientated in the range $\pi/2 \pm \varphi_c$ radians. In order to detect edges of any orientation, the frequency plane is tessalated with rotated versions of the prototypical filter shown in Figure 3.8 and described by eqn. (3.23). For exactly N filters to cover the frequency plane (see Figure 3.10), we require

$$N = \frac{\pi}{2\varphi_c}$$

in which case, each filter is spaced $2\varphi_c = \pi/N$ radians apart. The response of the k^{th} filter is

$$H_k(\omega,\varphi) = H_0(\omega,\varphi - k\pi/N)$$

and the total response of the N filters is simply

$$H_T(\omega,\varphi) = \sum_{k=0}^{N-1} H_k(\omega,\varphi) = \sum_{k=0}^{N-1} H_0(\omega,\varphi - k\pi/N) \tag{3.25}$$

A circular region exists in the neighbourhood of the origin, not covered by any of the N filters in

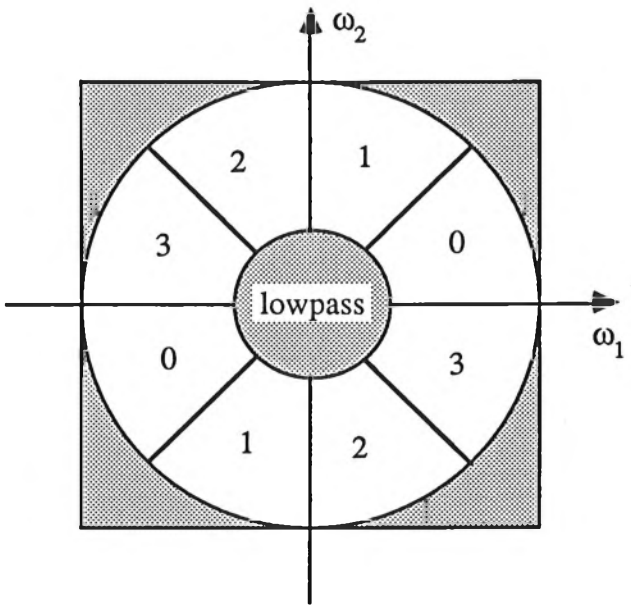


Figure 3.10: *Covering the frequency plane with directional filters and an additional lowpass filter. In this case, N = 4.*

eqn. (3.25). This region may be extracted from the image using a circularly symmetric lowpass filter with cutoff frequency ω_c (refer to eqn. (3.24)) and because it contains only low frequency Fourier components (i.e. no edges) it may be efficiently coded using transform coding [7]. The lowpass filter, together with the N directional filters, cover the entire frequency plane and the original image may be recovered by summing the output of all N+1 filters†. The key to the success of this method is twofold. Firstly, high and low frequency components are coded separately, thus avoiding the difficult situation of having to design a filter which responds optimally to both high and low frequency signal components. Secondly, the edge information (that is, the output of each

---

† Note that a small amount of distortion does take place because the region

$$L = \{(\omega_1,\omega_2): |\omega_1|\leq\pi, |\omega_2|\leq\pi\} - \{(\omega,\varphi): \omega\leq\pi\}$$

is excluded from the total response. Exclusion of the region L is equivalent to convolving the image with a filter with a frequency response $H(\omega,\varphi)=1$ for $\omega \leq \pi$, and distortion is minimised if the impulse response h(m,n) is narrow. Using eqn. (2.34) it is readily shown that $h(m,n) = h(r) = J_1(\pi r)/(2r)$ where $r^2=m^2+n^2$. Treating m and n as continuous variables, the energy contained within the circle C:r=ρ is given as

$$E(\rho) = \iint_C h^2(r)dmdn = \int_0^{\rho}\int_0^{2\pi} h^2(r)rdrd\varphi$$

and the fractional energy contained within C is $E(\rho)/E(\infty)$, which is easily evaluated as

$$1-J_0^2(\pi\rho)-J_1^2(\pi\rho).$$

For a fractional energy ratio of 90%, ρ is as small as 2, indicating quite a narrow impulse response.

directional filter) is not waveform coded, but modelled. If φ_c is small, then the filters are highly selective directionally, and each filter $H_k(\omega,\varphi)$ responds maximally for edges in the direction $k\pi/N+\pi/2$. In effect, each H_k produces, as output, straight line segments indicating edges in the original image. By coding the strength (slope), position and length of each edge, compression ratios in the order of 40 to 60:1 are possible [7].

Although the situation so far sounds ideal, the technique does have some implementation problems. For example, extremely narrow transition bands (refer to Figure 3.8) cause the filter outputs to ring badly – a manifestation of Gibbs' phenomenon – so the impulse response must be windowed. This has two desirable effects: firstly it reduces the 'echo' of each edge and thus makes the search for true zero crossings (which locate edges) more reliable. Secondly, it inhibits edge-ends from becoming extended. Edge extension occurs when the apparent length of an edge increases because of the narrow *lowpass* bandwidth of the directional filter. (The lowpass bandwidth is orthogonal to the principal direction). Put simply, insufficient Fourier components in the lowpass direction are passed by the filter to sharply define the end of an edge. Unfortunately the two benefits just described are obtained at the expense of directional selectivity. More importantly, the selection of a constant passband gain for each $H_k(\omega,\varphi)$ is a naive and bruteforce approach to detecting edges. A more subtle approach, involving shaping the passband into an optimal edge detector, was proposed by Nguyen and Andrews [9]. Discussion of this improvement would stray too far from the main purpose of this section, so details of the technique may be found in Appendix 3A.

3.3.3 Source Coding using Linear Prediction

The philosophy of linear predictive coding is a departure from that behind waveform and transform coding, which assume minimal knowledge about the signal, except perhaps bounds on the variance or frequency distribution of energy. As with all source coding techniques, the aim is to reduce the number of degrees of freedom of the signal. By imposing a model on the image generation process, we hope to store or transmit parameters that characterise the model, rather than the image itself. If the model parameters occupy less space than the signal, then we have achieved our goal of compressing the amount of information; the only question remaining is to what degree has the method been successful.

An image coding scheme based on an autoregressive model of image generation is shown in Figure 3.11. The operation of the major blocks is probably self-evident, however a short explanation follows.

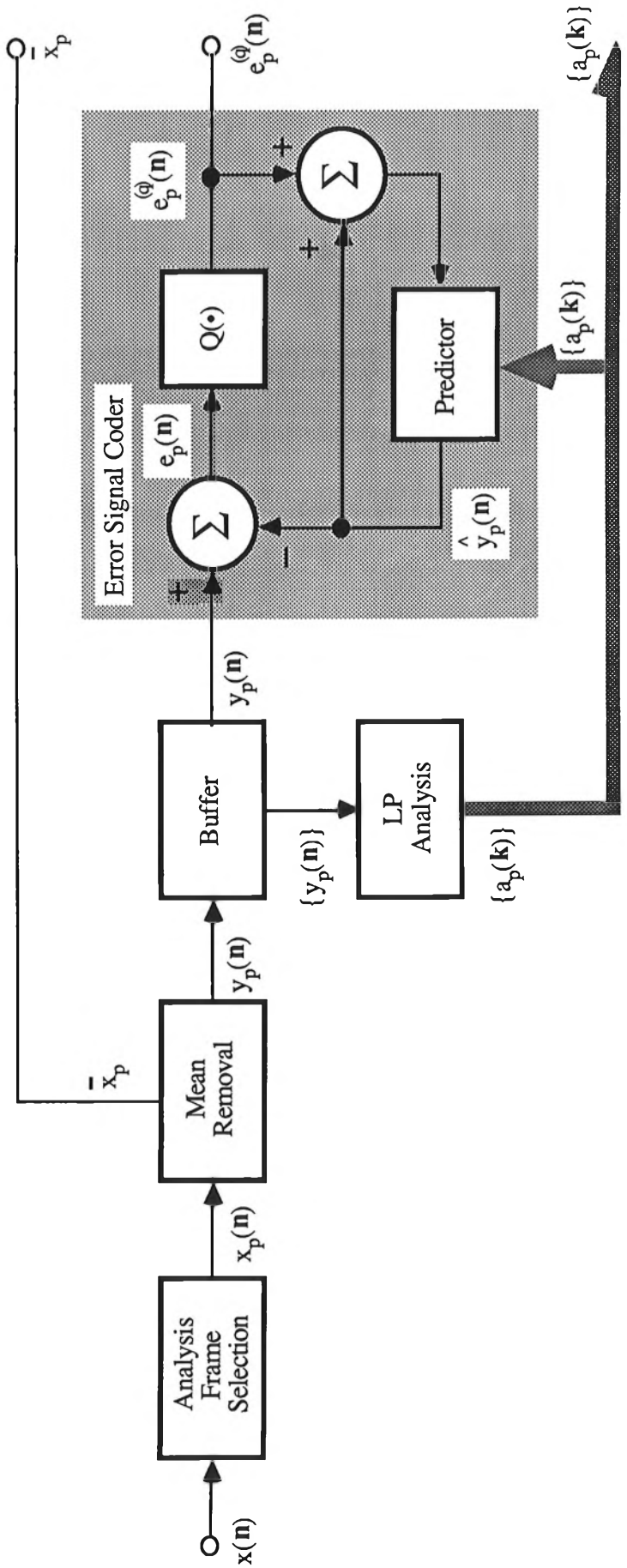


Figure 3.11: *Simplified block diagram of a linear predictive source encoder*

Analysis Frame Selection

The statistics of an entire image are almost always non–stationary in nature [10], so it is usual to segment the image into a number of smaller sub-images or analysis frames. Formally, the image $x(\mathbf{n})$, with region of support R_x, is segmented into a finite number of analysis frames, F_p:

$$R_x = \bigcup_p F_p \qquad \text{with} \qquad F_i \cap F_j = \emptyset \quad \text{for} \quad i \neq j \tag{3.26}$$

where adjacent frames do not overlap. Each frame, F_p, is of the form

$$F_p = \{(n_1,n_2): n_j^{(p)} \leq n_j \leq n_j^{(p)}+L-1;\ j = 1,2\} \tag{3.27}$$

where for convenience we have chosen square analysis frames with L pixels on a side. The co-ordinate $(n_1^{(p)},n_2^{(p)})$ locates the lower left hand corner of a particular analysis frame F_p in R_x. The sub-image, $x_p(\mathbf{n})$, passed to the rest of the encoder (see Figure 3.11) is given by

$$x_p(\mathbf{n}) = \begin{cases} x(\mathbf{n}) & \text{if } \mathbf{n} \in F_p \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

Mean Removal

This block removes the local mean of the anlysis frame $x_p(\mathbf{n})$. This step is desirable because it is found experimentally that the mean value (when non-zero) is responsible for a proportion of analysis frames yielding unstable prediction filters (see for example, Maragos *et al* [11]). Even ignoring the possibility of unstable prediction filters, a linear prediction model which explicitly caters for signals with non-zero mean is at best marginally stable. For example, consider the N-D recursive system $g(\mathbf{n})$ with mean μ:

$$g(\mathbf{n}) = \mu + \sum_{\mathbf{k}} a(\mathbf{k})\, g(\mathbf{n}-\mathbf{k}) + u(\mathbf{n}) \tag{3.29}$$

Taking the N-D Z transform of eqn. (3.29), and assuming that $u(\mathbf{n}) = \delta(\mathbf{n})$, we have

$$G(\mathbf{z}) = \mu \prod_{i=1}^{N} (1-z_i)^{-1} + \sum_{\mathbf{k}} a(\mathbf{k})\, G(\mathbf{z})\, \mathbf{z}^{\mathbf{k}} + 1 \tag{3.30}$$

and solving for $G(\mathbf{z})$:

$$G(\mathbf{z}) = \frac{\mu + \prod_{i=1}^{N}(1-z_i)}{\prod_{i=1}^{N}(1-z_i)\left[1-\sum_{\mathbf{k}} a(\mathbf{k})\mathbf{z}^{\mathbf{k}}\right]} \tag{3.31}$$

Obviously $G(\mathbf{z})$ has a singularity on $\mathbf{T}^N$ at the point $\mathbf{z}=\mathbf{1}$ which is only cancelled by the numerator zero for $\mu=0$. Thus $G(\mathbf{z})$ is marginally stable irrespective of a stable parameter set $\{a(\mathbf{k})\}$. In view

of this result, coding proceeds using a new zero-mean analysis frame, $y_p(\mathbf{n})$, given by

$$y_p(\mathbf{n}) = x_p(\mathbf{n}) - \bar{x}_p \tag{3.32}$$

where

$$\bar{x}_p = \frac{1}{\eta(F_p)} \sum_{\mathbf{n}} x_p(\mathbf{n}), \quad \mathbf{n} \in F_p \tag{3.33}$$

Buffer

The buffer provides pixels on a sample by sample basis to the Error Signal Coder and also provides the entire frame $\{y_p(\mathbf{n})\}$ to the LP Analysis block. The buffer does not modify the signal in any way.

LP Analysis

Given a particular analysis frame $\{y_p(\mathbf{n})\}$, the sole purpose of the LP Analysis block is to generate an optimum set of model coefficients $\{a_p(\mathbf{k})\}$ which characterises the p^{th} sub-image with minimum variance. If $\{y_p(\mathbf{n})\}$ is stationary and the autocorrelation samples are known *a priori* then $\{a_p(\mathbf{k})\}$ is given by the solution to eqn. (3.9). Unfortunately, it is often the case that $\{y_p(\mathbf{n})\}$ is at best quasi-stationary, and we are usually entirely ignorant of the autocorrelation function. In this case, we must use eqn. (3.7) in which $\varphi(\mathbf{k};\mathbf{p})$ (which is really an estimate of the non-stationary autocovariance) depends solely on the samples $y_p(\mathbf{n})$. Thus, the governing equations are (repeated from eqns. (3.7) and (3.6) for convenience):

$$\sum_{\mathbf{k}} a(\mathbf{k})\, \varphi(\mathbf{k};\mathbf{q}) = \varphi(\mathbf{0};\mathbf{q}), \qquad \mathbf{k},\mathbf{q} \in R_a \tag{3.34}$$

where

$$\varphi(\mathbf{k};\mathbf{q}) = \sum_{\mathbf{k}} y_p(\mathbf{n}-\mathbf{k})\, y_p(\mathbf{n}-\mathbf{q}), \qquad \mathbf{n} \in R_e \tag{3.35}$$

What remains to be established is the exact form of R_e (refer to eqns. (3.4) and (3.35)) given the description of each analysis frame, F_p (refer to eqn. (3.27)). If we force $R_e = F_p$, then the first error sample (from eqns. (3.2),(3.3) and (3.27)) is

$$e(\mathbf{n}^{(p)}) = y_p(\mathbf{n}^{(p)}) - \sum_{\mathbf{k}} a(\mathbf{k})\, y_p(\mathbf{n}^{(p)}-\mathbf{k}) \tag{3.36}$$

Unfortunately, the summation on the right-hand side of eqn. (3.36) requires samples outside of F_p

for $\mathbf{k} > \mathbf{0}^{\dagger}$, so it is usual to restrict R_e to a proper subset of F_p so that the samples required in eqn. (3.36) are contained completely within F_p. Thus, R_e must be of the form

$$R_e = \{(n_1,n_2):\ n_j^{(p)}+Q-1 \le n_j \le n_j^{(p)}+L-1,\ \ j = 1,2\} \tag{3.37}$$

if the support of $a(\mathbf{k})$ is

$$R_a = \{(k_1,k_2):\ 0 \le k_1,k_2 \le Q-1\} \tag{3.38}$$

so that $(\mathbf{n}^{(p)}-\mathbf{k}) \in F_p$ even for the most extreme $\mathbf{k}$ values in R_a. Once the optimal set $\{a_p(\mathbf{k})\}$ has been determined, $\{a(\mathbf{k})\}$ is sent to both the channel encoder and the error signal coder.

When the prediction process starts, attention must be given to the boundary conditions of the frame of interest. For analysis frames not on the boundary of the image, it is a simple matter to recall samples from previous fames as they are needed. If the analysis frame lies on the boundary of the image, samples required from outside the frame are non-existant and it is necessary to assume appropriate values. The strategy most often adopted in practice is to assume zero boundary values. The simplicity of this approach is offset slightly by an increase in the error signal at the start of the frame. An alternative approach, which maintains the error signal at its intra-frame level, is to use samples just inside the frame boundary as the required external boundary samples. The rather obvious drawback with this scheme is that the boundary samples must be transmitted before the prediction process can begin.

Error Signal Coder

It is highly unlikely that the predicted pixel intensity $\hat{y}_p(\mathbf{n})$ will be exactly equal to the true pixel intensity $y_p(\mathbf{n})$. In order to faithfully reconstruct the image at the receiver, it is necessary to code the error signal

$$e_p^{(q)}(\mathbf{n}) = y_p(\mathbf{n}) - \hat{y}_p(\mathbf{n}) \tag{3.39}$$

Referring to Figure 3.11, it is seen that coding $e_p(\mathbf{n})$ is simply a quantization (i.e. truncation) process, yielding the quantized error signal $e_p^{(q)}(\mathbf{n})$:

$$e_p^{(q)}(\mathbf{n}) = Q[e_p(\mathbf{n})] \tag{3.40}$$

Notice that since the receiver is given $e_p^{(q)}(\mathbf{n})$ rather than $e_p(\mathbf{n})$, the predictor in the source encoder must be driven by the quantized error signal. This ensures that the coded error signal represents the difference between the true sample value and the predicted value based on $e_p^{(q)}(\mathbf{n})$, rather than the true sample value and the predicted value based on $e_p(\mathbf{n})$ – which is not transmitted.

---

† In this context, $\mathbf{k} > \mathbf{0}$ means $(k_1 > 0) \cup (k_2 > 0)$.

3.3.4 General Comments

The 2-D coding scheme outlined in the previous section differs from the simplist 1-D linear prediction schemes for speech in two important ways. Firstly, except in relatively uniform areas, image regions can have extremely narrow autocorrelation functions which naturally lead to an increased variance in the error signal, $e_p(\mathbf{n})$. Hence a major portion of the channel bit-rate is taken up with the encoded error signal $e_p^{(q)}(\mathbf{n})$; the parameter set $\{a_p(\mathbf{k})\}$ accounts for only a small fraction of the channel bandwidth. Secondly, even if an image was found which really was the product of some underlying 2-D autoregressive process, it would still be necessary to transmit the error signal. Such is the case because of the different way in which humans perceive visual and audio signals. If the receiver model was excited with some other error signal, an image would be produced that was in fact another realization of the same stochastic process, but it would *look* completely different. This should be contrasted with the 1-D case where the same scenario would be tolerable because the ear responds to formants in the signal spectrum rather than the time-domain waveform of the signal. (Recall that the power spectral density is the same for all realizations of a stationary 1-D stochastic process [12,p.265]).

3.4 Test Images

Since our interest lies in linear predictive coding of images, it is necessary to introduce a few test images in order to demonstrate the various ideas and techniques which follow. Given the essentially infinite number of possible images†, we require a small number of images (in this case 2) which are hopefully representative of the much larger class of all possible images. Figures 3.12 and 3.13 show the images chosen for this task. The picture of the woman's face (hereafter called 'the woman') was chosen for its soft tones, whereas the busy street scene ('the street') was chosen for its activity. Each picture consists of 128 by 128 pixels with each pixel intensity quantized to 256 grey levels, numbered 0 to 255 inclusive.

3.5 Instability in the Linear Prediction Model

High correlation in the signal $x(\mathbf{n})$ is desirable as it leads to a smaller variance in eqn. (3.4). However, too much correlation tends to destabilize the resulting filter. For example, consider the simple 1-dimensional 2nd-order all-pole filter

$$x(n) = a_1\, x(n-1) + a_2\, x(n-2) + u(n) \tag{3.41}$$

---

† There are in fact only a finite number of images since the total number of pixels, and the pixel intensities, are finite integers. However, given a 128x128 pixel image with intensity quantized to 8 bits, the number of different images is 2^{262144} or approximately $1.6\text{x}10^{78913}$.

Figure 3.12: *First test image – soft-toned face*

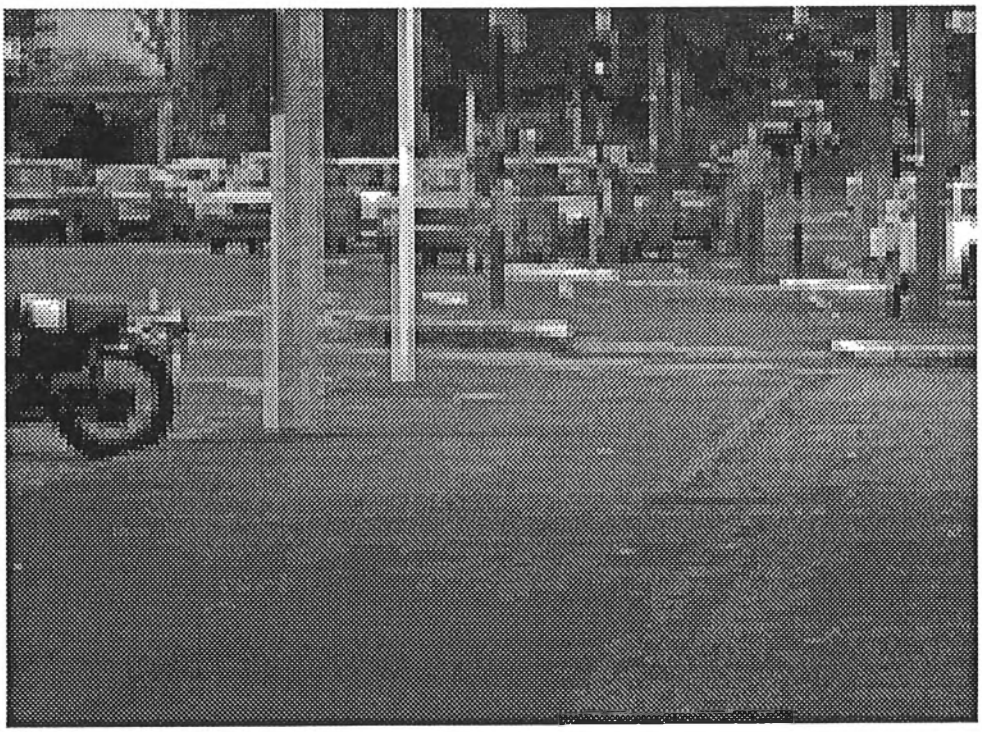


Figure 3.13: *Second test image – busy street scene*

with z-transform

$$X(z) = \frac{1}{1 - a_1 z - a_2 z^2} = \frac{1}{(\alpha - z)(\beta - z)} \tag{3.42}$$

It is not difficult to show that the optimum coefficients, (found by minimising the 1-D equivalent of eqn. (3.4)), are given by

$$a_1 = \frac{\rho_1(1-\rho_2)}{1-\rho_1^2} \tag{3.43a}$$

and

$$a_2 = \frac{\rho_2 - \rho_1^2}{1-\rho_1^2} \tag{3.43b}$$

where ρ_k is the normalised correlation coefficient for a lag of k samples. Equations (3.43a) and (3.43b) show how signal correlation bears directly on the model stability, via a_1 and a_2. Stability bounds on ρ_1 and ρ_2 are illustrated in Figure 3.14 and it is easy to show that X(z) represents a stable filter provided

$$|\rho_1| < 1, \quad |\rho_2| < 1 \text{ and } \rho_2 > 2\rho_1^2 - 1$$

It is apparent from Figure 3.14 that the stability margin of the AR model deteriorates for large magnitudes of ρ_1 and ρ_2.

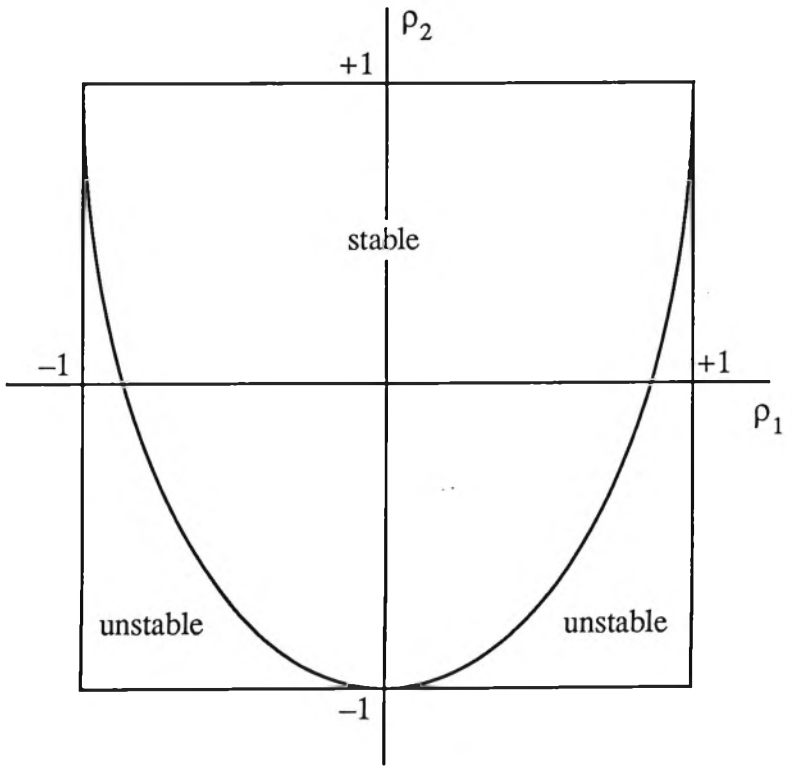


Figure 3.14: *Effect of signal correlation on stability for the 1-D 2nd-order AR model.*

For multidimensional signals the situation is similar but more complicated; this can be attributed to the fact that N-dimensional Z-transforms possess zero sets which form continuous (2N-2)-dimensional surfaces (extending out to infinity) embedded in 2N-dimensional space [13,p.52]. Not only does this make it difficult to imagine the zero set of a system function, but more importantly it makes testing of the system's stability a non-trivial task (see Chapter 4). The effect of an unstable filter on image reconstruction is catastrophic. If the zero-set of the prediction filter extends beyond the unit bidisc, the filter output will grow without bound and that particular analysis frame will display the characteristic 'washout' effect, where the frame is saturated at either the maximum or minimum grey level (usually 255 and 0, respectively). For example, returning to the 1-D AR model examined previously, for a given ρ_1 and ρ_2, we may find that the poles of eqn. (3.42) are complex:

$$\alpha = \bar{\beta} = r \exp(j\phi)$$

It is easy to show that if $u(n) = \delta(n)$, then $x(n)$ is given by

$$x(n) = \frac{1}{r^{n+2}} \frac{\sin (n+1)\phi}{\sin \phi} \tag{3.44}$$

Obviously, if $r < 1$, the modulus of $x(n)$ grows exponentially; the sign of $x(n)$ being determined by the trigonometric terms. An analogous effect occurs in 2-D systems and for small values of r, $x(n_1,n_2)$ saturates for relatively small values of (n_1,n_2), thereby 'washing out' the entire frame.

Another source of instability is that which arises when the image data in an analysis frame 'looks' unstable. For example, consider the contrived signal

$$s(\mathbf{n}) = \begin{cases} 2^{n_1+n_2} & (n_1,n_2) \in F_p \\ 0 & (n_1,n_2) \notin F_p \end{cases} \tag{3.45}$$

Obviously $s(\mathbf{n})$ is bounded for finite $\eta(F_p)$. However, as far as the linear prediction is concerned, it 'thinks' it has a small window on a truely unstable signal. Choosing an analysis frame slightly larger than F_p usually rectifies the problem.

A Note on Error Dynamic Range

It may appear from Figure 3.11 that an unstable predictor should have no effect on the quality of the image, since the error signal is constantly correcting any discrepency between the true and estimated signals. This is *only* true if the error signal has unlimited dynamic range. In this case, the unstable predictor output is delicately cancelled by an equally large error signal (of opposite sign), producing a resynthesised signal with an appropriately small dynamic range. However, in

practice, the error signal has very limited dynamic range (usually only one byte) and therefore cannot correct for an unbounded output signal from the predictor.

Since the primary aim of this chapter is to present a method for improving the stability rate of any 2-D LPC coding scheme, we continue with the assumption that an unstable filter (and hence a washed out frame) is totally unacceptable and must be avoided; while stable filters will be used in conjunction with contemporary coding schemes. (For example: [11],[14]).

3.6 Decorrelation as an Aid to Increasing Stability Rates [15]

As we are interested in using the AR model for coding, we have the added option of modifying the signal, (prior to analysis), in such a way that the probability of deriving a stable filter is greatly increased. Providing the receiver is given details of the modification, the original signal can be exactly reconstructed. The penalty for such improved 'stability rates' is that we incur an overhead because we must also transmit details of the modification.

As illustrated in Section 3.5, signal correlation interacts strongly with filter stability; in particular, that too much correlation tends to destabilize the filter. An obvious method for improving the stability margin, (and hence the stability rate), is to reduce the signal correlation. This idea will be pursued in the next 2 sections.

The experimental results which follow were obtained from the two representative images shown in Figures 3.12 and 3.13. Each picture consists of 128x128 pixels with each pixel intensity quantized to 256 grey levels. In order to identify candidate frames for stabilization, 5000 addresses were randomly generated, (identifying $(n_1^{(p)},n_2^{(p)})$ in eqn. (3.27)), for each image. Each frame was analysed, (with N=32,Q=3), and the parameter set $\{a(\mathbf{k})\}$ was tested for stability. The addresses of those frames which were deemed unstable are shown, highlighted, in Figures 3.15 and 3.16. The relative frequency of the frames identified as unstable is shown in Table 3.1.

Table 3.1

Rate of instability for the images shown in Figures 3.12 and 3.13

Image	# Unstable	% Unstable
Face	484	9.63
Street	69	1.38

Figure 3.16: *Regions of the woman producing unstable filters*

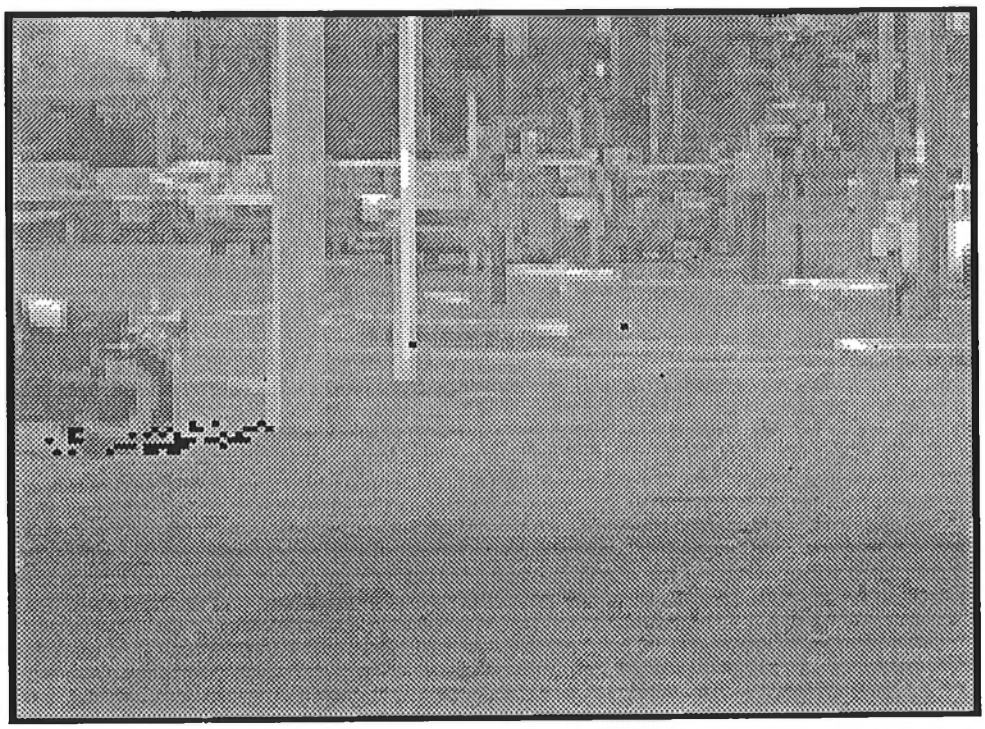


Figure 3.15: *Regions of the street producing unstable filters*

3.6.1 Frequency Removal

A method which naturally suggests itself for reducing the correlation length of the signal, is to remove those components which dominate the 2-D frequency spectrum of the analysis frame. For a frame containing well-defined edges, (such as the street), the spectrum will have appreciable high-frequency energy. If the frame has fairly gradual spatial variations, (such as the face), the spectrum will consist mainly of low-frequency energy.

In both cases, the spectrum has a dominant low-frequency lobe due to the inherent bias in the image. Bearing this in mind, a naive approach is to successively remove "shells" of low-frequency terms from the spectrum (see Figure 3.17), until a stable filter is produced. A more sophisticated scheme is to order the components in Fourier space on a magnitude basis, and then remove the largest components, (consisting of complex-conjugate pairs), until a stable filter results. An important issue, then, is how many components must be removed in order to stabilize the filter. Removing too many components is to be avoided since this represents a bit-rate overhead which must be transmitted. Using all of the unstable frames identified in Figures 3.15 and 3.16 as a database, Figure 3.18 shows cumulative histograms detailing the number of frequency components which must be removed in order to achieve a specified stabilization rate for the Ordered and Shell methods.

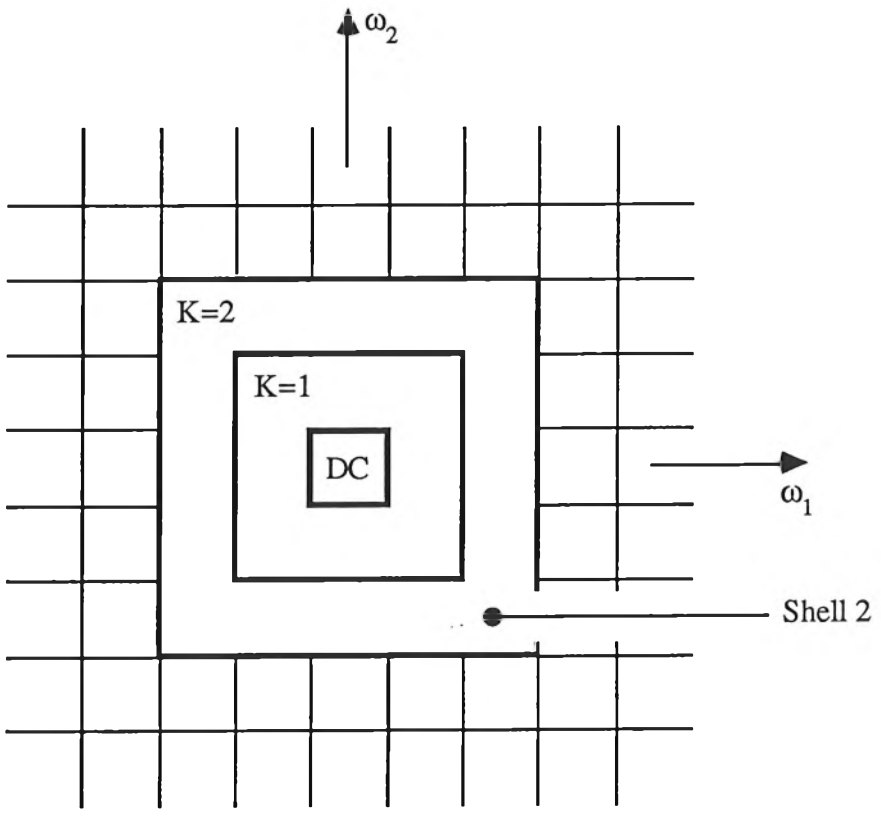


Figure 3.17: *Removing successive shells from Fourier space for the Shell method.*

Note that the number of components, P, (conjugate pairs), removed for the Ordered method increments by 1 each time. However, the number of components removed for the Shell method forms the sequence 4,12,24,...,2K(K+1) for positive integers K. This situation is illustrated in Figure 3.17 where shell K encloses P=2K(K+1) complex-conjugate pairs, (excluding DC). Moving from shell K to shell (K+1) admits an extra 4(K+1) conjugate pairs.

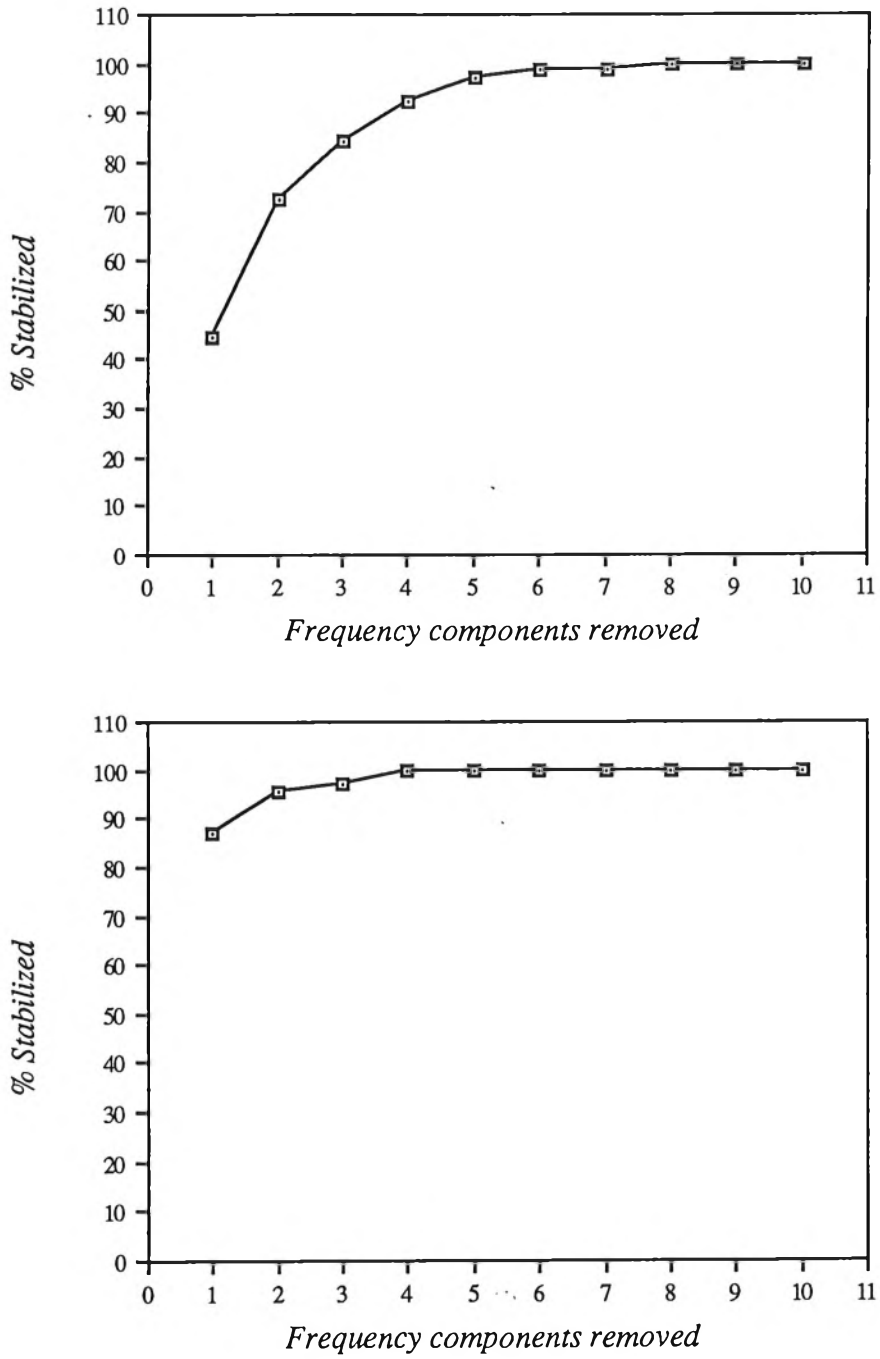


Figure 3.18: *Cumulative histograms showing how many frequency components must be removed before a stable filter is produced.*
top *The woman using the Ordered method.*
bottom *The street using the Ordered method.*

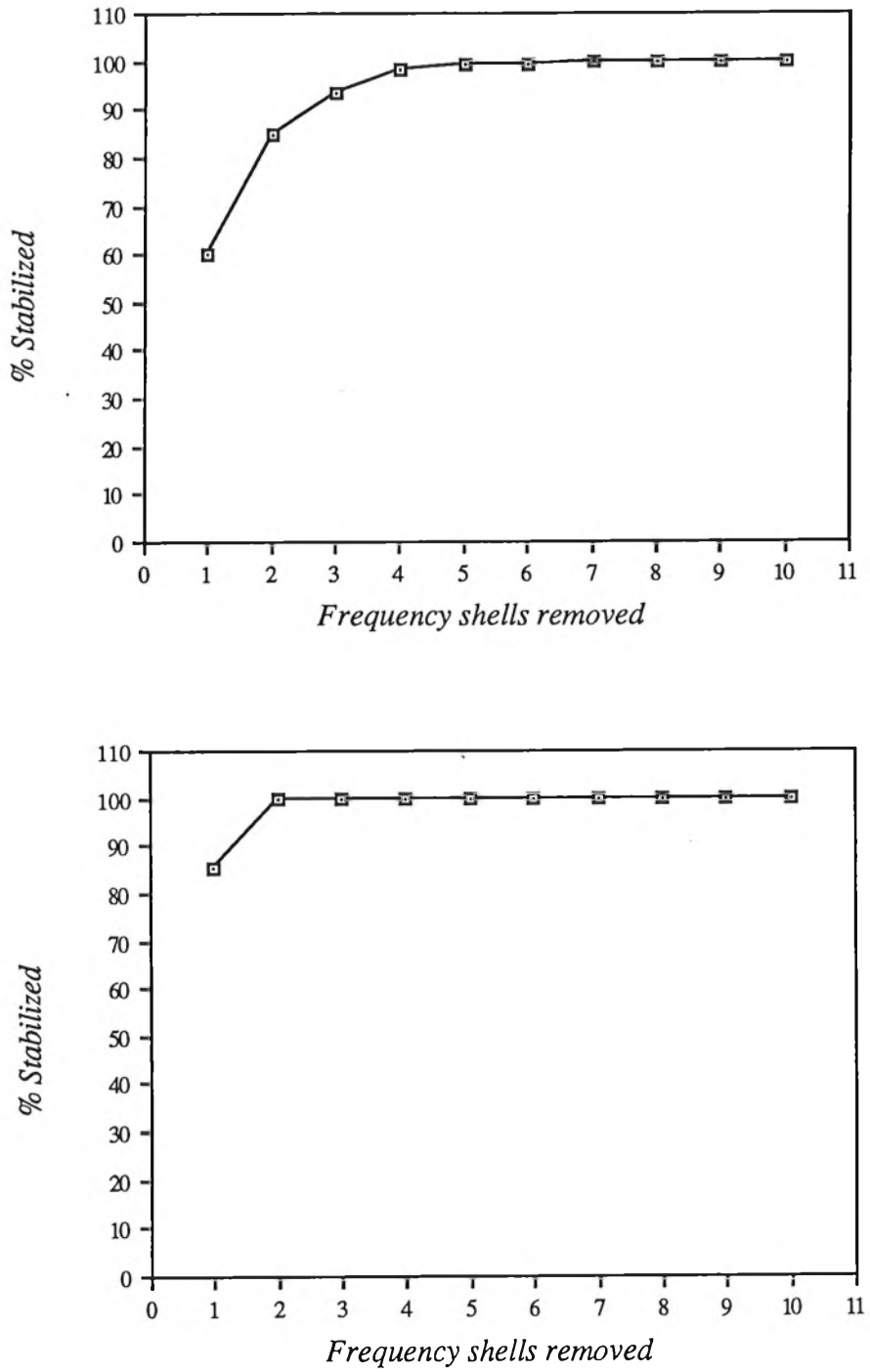


Fig. 3.18 cont'd: *Cumulative histograms showing how many frequency components must be removed before a stable filter is produced.*
top *The woman using the Shell method.*
bottom *The street using the Shell method.*

Both methods are successful in stabilizing the all-pole filter if a sufficient number of components is removed. It should be noted that considerably less components need to be removed for the Ordered method. Figure 3.19 shows 2-dimensional histograms illustrating the distribution of removed components for the two images. Note that, as expected, most of the removed components are from the low-frequency region of the spectrum.

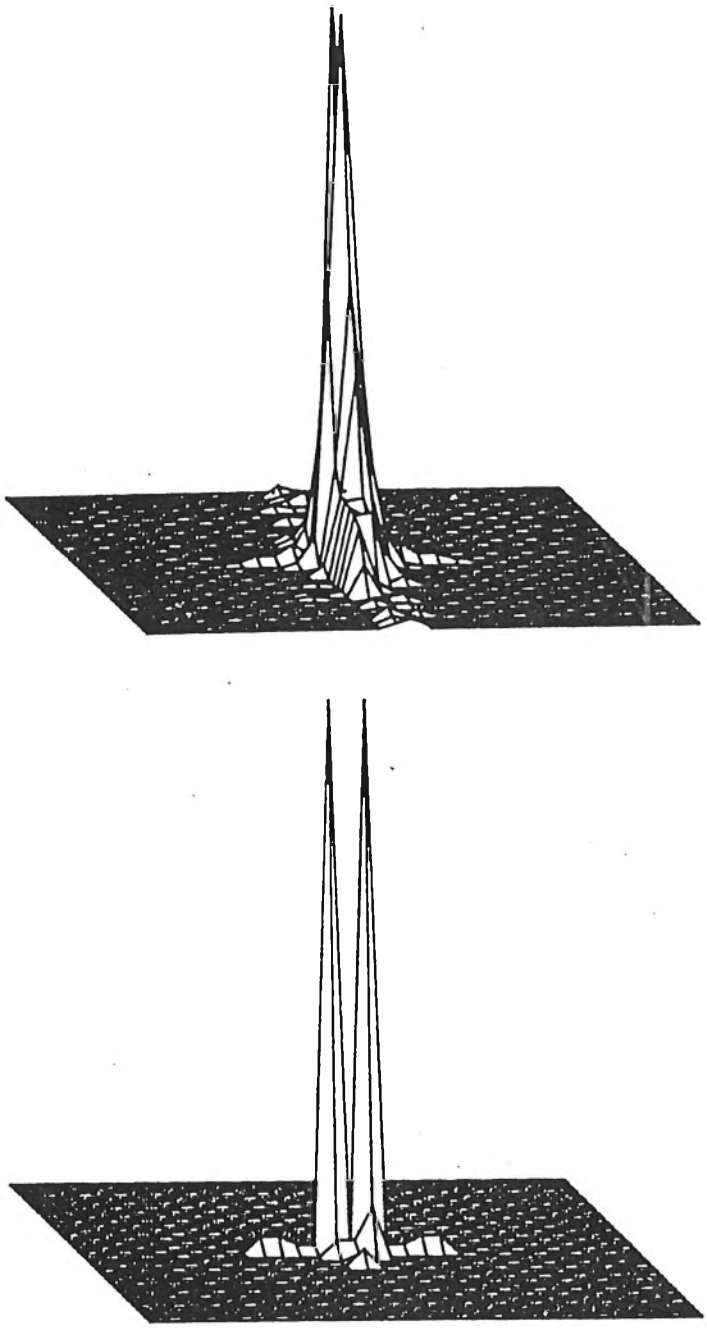


Figure 3.19: *2-D histograms showing the distribution of removed frequency components for the Ordered method.*
top *The Woman.* **bottom** *The Street.*

3.6.2 Sequency Removal

We saw in Section 3.6.1 that correlation reduction may be accomplished satisfactorily with the 2-D Fourier Transform. We are prompted to inquire if there are other faster transforms which may also be applicable. A prime candidate is the Hadamard Transform, which has been successfully applied in the past to picture coding problems [16]. The advantages of using the Hadamard transform result from the fact that it is a 'real' transform and that several 'fast' algorithms exist for its computation. (See Section 2.7.3).

Note that the Hadamard Transform is most often associated with Transform Coding of images [16], along with its attendent short-comings. Specifically, the reconstructed images appear 'blocky' because the signal is truncated in Hadamard space, leaving only the Walsh basis functions with relatively long period. This is not a problem with our technique because even though sequency components are removed from the frame prior to analysis, those sequency components are added back to the resynthesised frame at the receiver.

The Fast Hadamard Transform (FHT) may be computed with either dyadic or sequency ordering (see Section 2.7.1). Our interest lies in a sequency-ordered implementation so that the sequency increases as we move away from the transform origin.

The basic approach to sequency removal is identical to that of the frequency removal technique in the previous section. With sequency removal, the Shell method cannot remove an integral number of Hadamard components, but instead forms a sequence 3,8,15,...,K(K+2) for positive integers K. This situation is illustrated in Figure 3.20 where shell K encloses P=K(K+2) Hadamard components, excluding DC. Moving from shell K to shell (K+1) introduces an extra 2K+3 components.

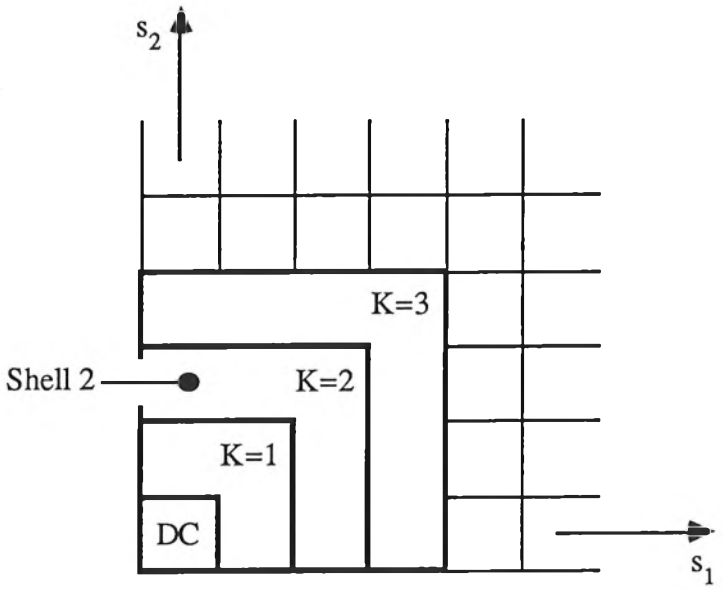


Figure 3.20: *Removing successive shells from Hadamard space for the Shell method.*

Figure 3.21 shows cumulative histograms detailing the number of sequency components which must be removed in order to achieve a specified stabilization rate. Both methods are successful but again the Ordered method requires the removal of fewer components and is therefore more efficient in this respect.

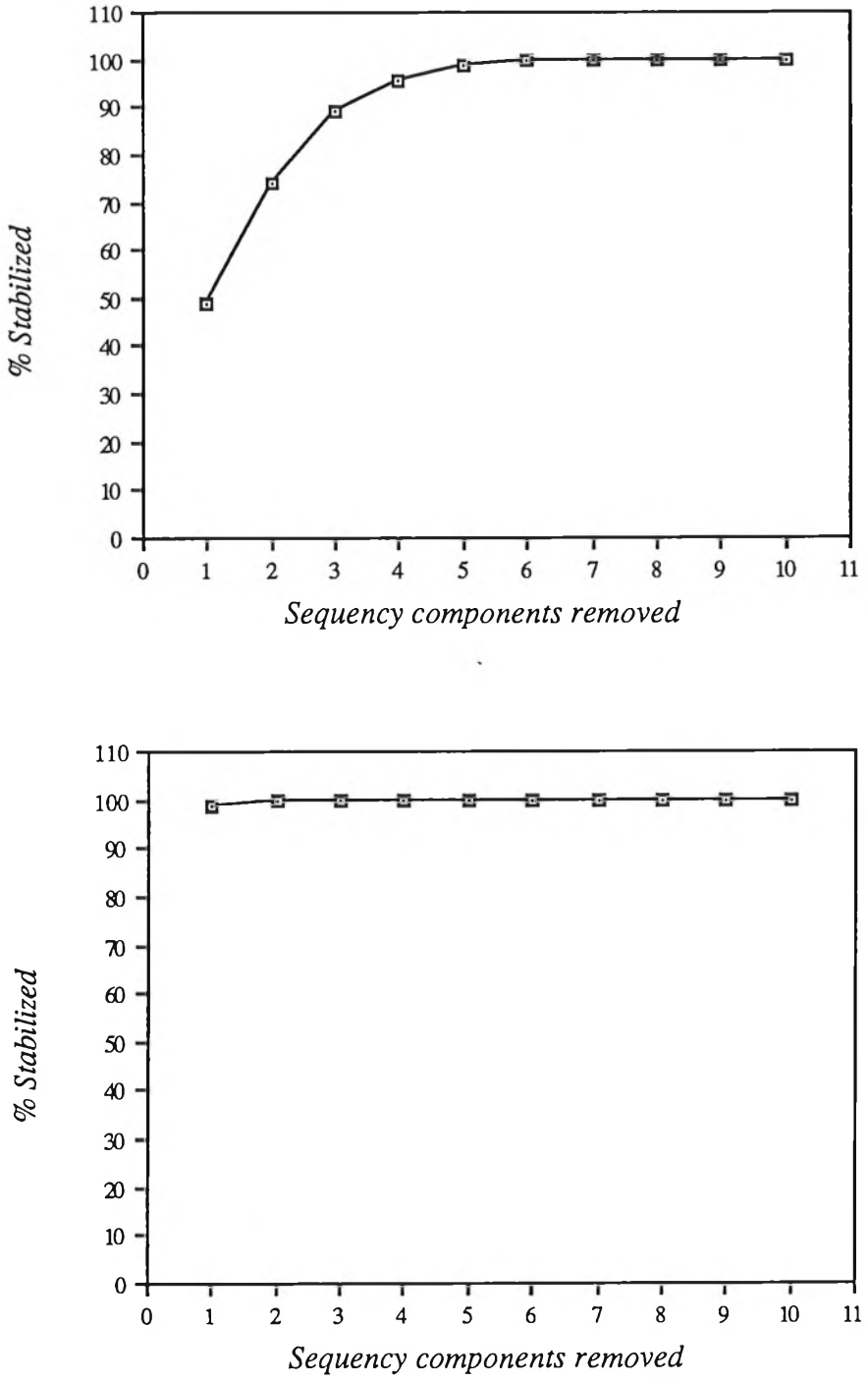


Fig. 3.21: *Cumulative histograms showing how many sequency components must be removed before a stable filter is produced.*
top *The woman using the Ordered method.*
bottom *The street using the Ordered method.*

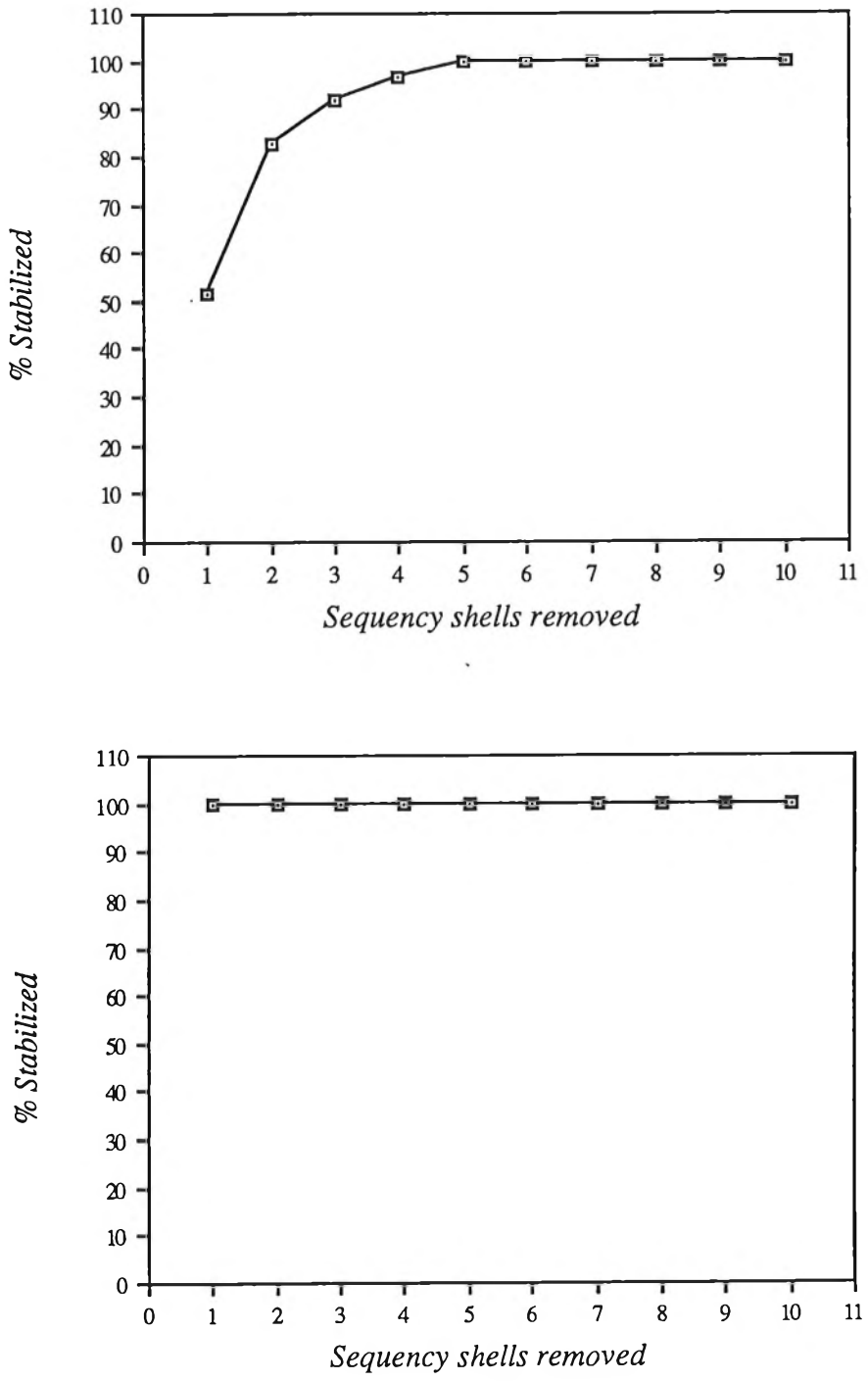


Figure 3.21 cont'd: *Cumulative histogram showing how many sequency components must be removed before a stable filter is produced.*
top*The woman using the Shell method.*
bottom*The street using the Shell method.*

3.6.3 Routes to Stability

In order to show directly how Fourier or Hadamard component removal stabilizes the prediction filter, it is instructive to follow the *route to stability* for each method. That is, we examine the rootmap of the filter† as either frequency or sequency components are removed from the analysis frame. Figure 3.22 illustrates the effect of shell removal in Hadamard space. Stabilization occurs after the removal of 5 shells (35 components). Notice that the right-hand contour in each z-plane has shrunk almost to a point on the real axis, indicating a linear factor in the Z transform of the prediction filter. Figure 3.23 shows a similar trend as ordered Hadamard components are removed. However, in this case, stabilization is achieved much sooner, requiring the removal of only 4 components.

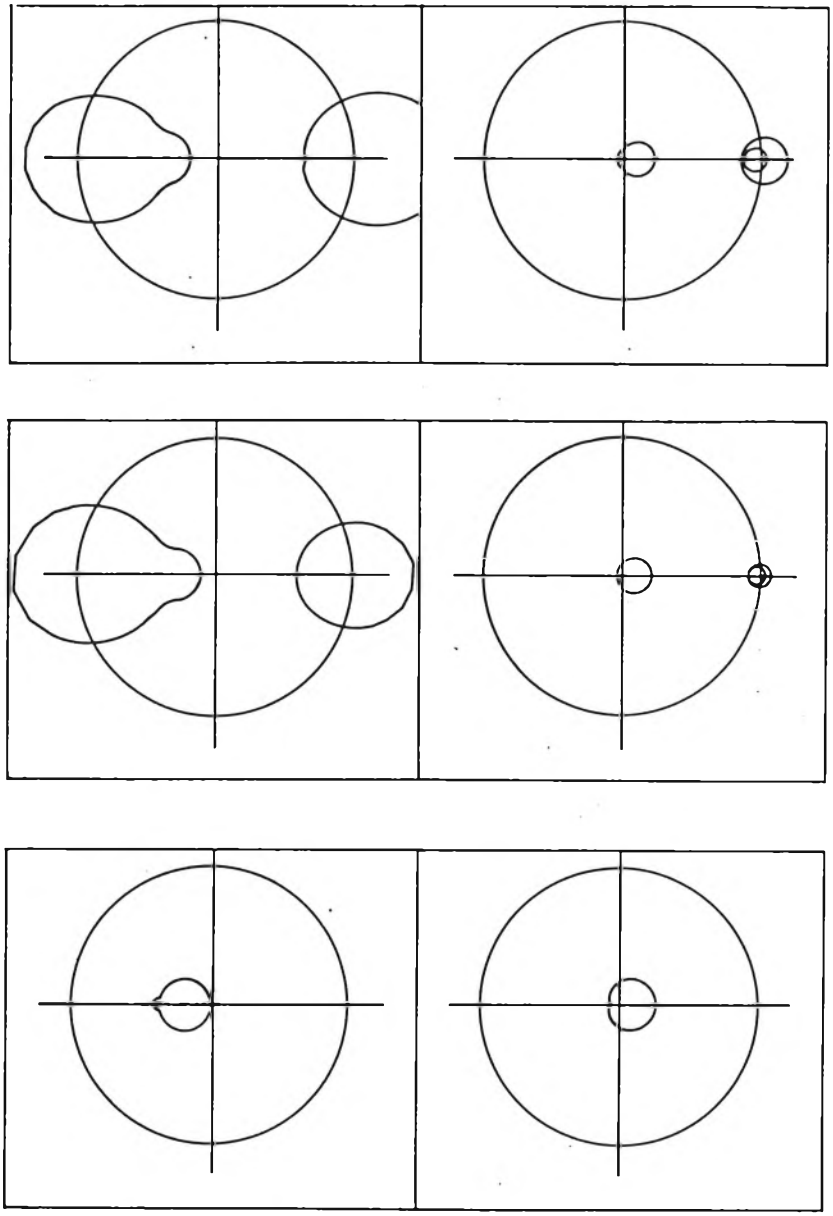
This effect should not be unexpected since we are effectively removing the deterministic part of the signal, leaving behind only the random components. It is well known that the Z transform of the autocorrelation function of a 2-D Markovian random field is separable. That is, the Markov correlation characteristics are completely determined by *separate* horizontal and vertical correlation coefficients. This separability manifests itself in the rootmap as a single point in each z-plane. The latter stages of sequency removal in Figures 3.22 and 3.23 clearly show a tendency towards this situation as the rootmap contracts in on itself, away from the unit bidisc.

In order to contrast the Fourier and Hadamard methods, Figures 3.24 and 3.25 show the routes to stability for Fourier shell removal and ordered frequency removal, respectively. Note that for this particular frame, frequency removal performs poorly compared to sequency removal; even after the removal of 9 shells (180 components), the filter is still unstable. However, removal of only 8 ordered frequency components ensures stability (Figure 3.25).

Comparison of Figures 3.23 and 3.24 with Figures 3.25 and 3.26 indicate that sequency removal is more efficient than the equivalent Fourier method, in terms of the number of components involved. This is not unexpected since the subtraction of Walsh functions (which make up the basis functions of the Hadamard transform) from the analysis frame introduce discontinuities which greatly reduces the interpixel correlation. Obviously the effect is less severe when removing Fourier components since all of the basis functions are continuous.

---

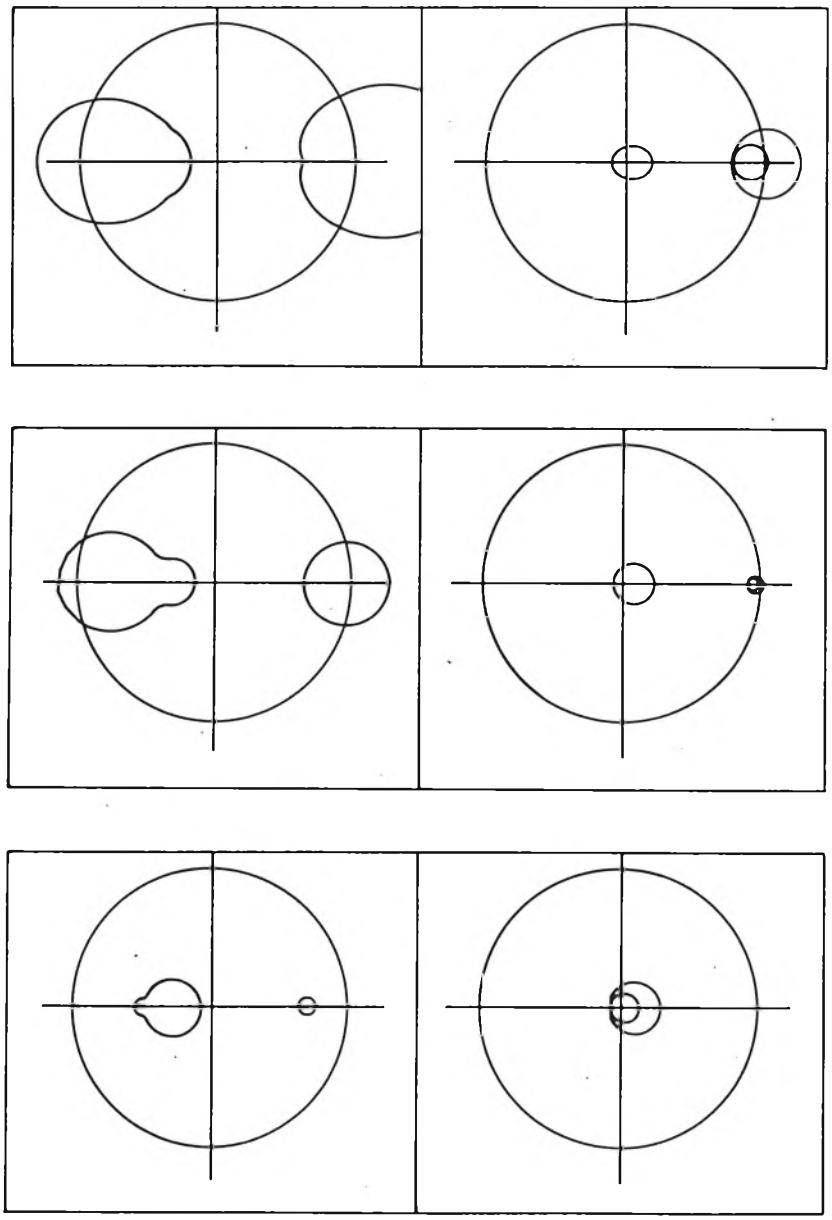
† Section 4.4 discusses the rootmap and its meaning.



z_1 plane z_2 plane

(For ease of viewing, the reciprocal of each rootmap is shown)

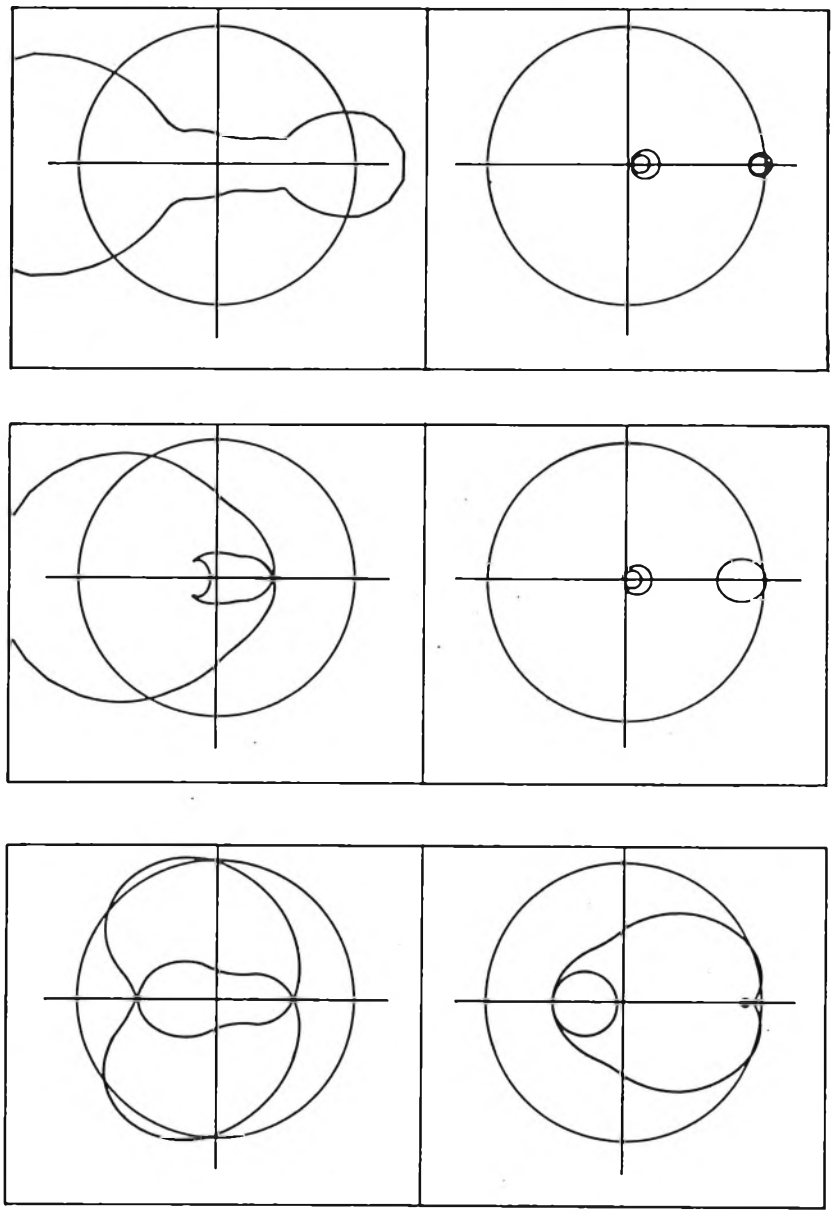
Figure 3.22: *Following the route to stability for Hadamard shell removal*
top *Shell 1 removed* **mid** *Shells 1–3 removed* **bot** *Shells 1–5 removed*



z_1 plane z_2 plane

(For ease of viewing, the reciprocal of each rootmap is shown)

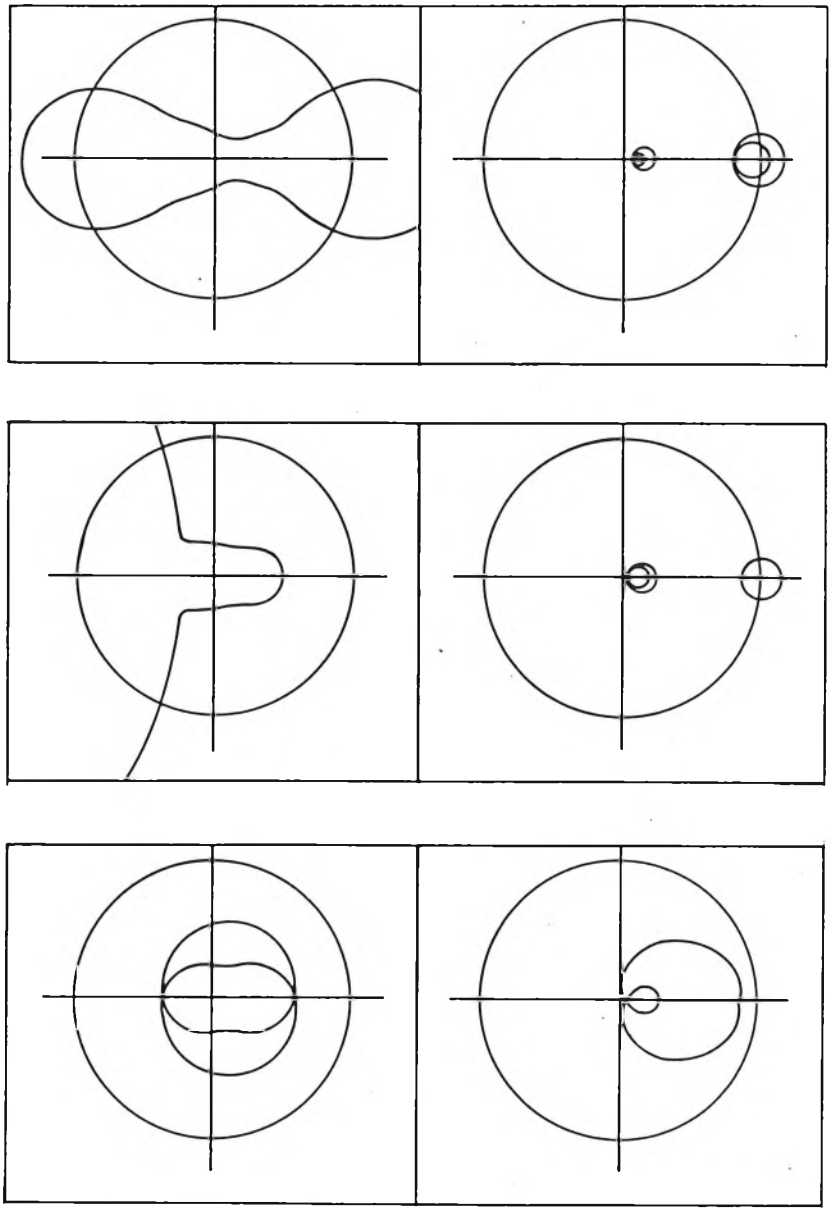
Figure 3.23: *Following the route to stability for ordered sequency removal*
top *1 removed* **mid** *3 removed* **bot** *4 removed*



z_1 plane z_2 plane

(For ease of viewing, the reciprocal of each rootmap is shown)

Figure 3.24: *Following the route to stability for Fourier shell removal*
top *Shell 1 removed* **mid** *Shells 1–5 removed* **bot** *Shells 1–9 removed*



z_1 plane z_2 plane

(For ease of viewing, the reciprocal of each rootmap is shown)

Figure 3.25: *Following the route to stability for ordered frequency removal*
top *1 removed* **mid** *4 removed* **bot** *8 removed*

3.7 Cost Analysis

A coding scheme is viable for real-world application if it can be implemented with minimum algorithm complexity and transmission-rate overhead. Not surprisingly, this almost always leads to a tradeoff between algorithm sophistication (i.e. complexity) and execution speed. Thus, it is of interest to compare the complexity of the four approaches discussed in this chapter. That is, we wish to assess the relative merits of Hadamard versus Fourier transforms, and the Shell versus the Ordered methods. It should be noted that the bit-rate overhead in dealing with an unstable filter is higher for our method compared with that of Maragos *et al* [11]; although experience has shown that the method of Maragos *et al* only detects approximately half of all unstable filters (refer to Chapter 4, Section 4.6 for details).

To judge the complexity of each transform algorithm, we will adopt the standard practice of comparing the number of additions and multiplications for each scheme. Subtraction is considered equivalent to addition.

3.7.1 Transform Calculations

Fourier Transform

The major computational bottleneck for Fourier component removal is calculating the forward and reverse discrete Fourier transform (DFT). The literature is filled with papers relating to the computation of the DFT or new methods of its implementation; however, one of the most efficient algorithms for computing 2-D DFTs is the Vector Radix FFT [3,p.76]. For an LxL analysis frame, evaluation of a single forward or reverse radix-(2x2) FFT requires

$$0.75L^2\log_2 L \quad \text{complex multiplications}$$

and

$$2L^2\log_2 L \quad \text{complex additions}$$

Each complex multiplication consists of 4 real multiplications and 2 real additions. A complex addition requires 2 real additions. Hence, a forward-reverse transform cycle requires

$$6L^2\log_2 L \quad \text{real multiplications}$$

and

$$11L^2\log_2 L \quad \text{real multiplications}$$

If ß is defined as

$$\beta = \frac{\text{time for 1 real multiplication}}{\text{time for 1 real addition}}$$

then the total complexity, in terms of 'add times', is given by

$$C_{freq} = (11+6\beta)L^2\log_2 L \tag{3.46}$$

Hadamard Transform

Several 'divide-and-conquer' strategies exist, similar to those for computing the DFT, for the calculation of the 1-D fast Hadamard transform (FHT or WHT). (For example, see [17]). Use of the row-column method for evaluationg the 2-D FHT requires

$$C_{seq} = 4L^2\log_2 L \quad \text{(real additions)} \tag{3.47}$$

for a complete forward-reverse transform cycle of an LxL analysis frame.

Speed Comparison

Comparison of eqns. (3.46) and (3.47) reveals that the frequency removal method is considerably more time consuming than sequency removal. In fact:

$$\frac{C_{freq}}{C_{seq}} = 2.75 + \beta \tag{3.48}$$

so that the frequency method will always be at least 2.75 times faster than the sequency method.

Theoretically, $\beta \in [1,\infty)$, although typically β ranges from 1 (for digital signal processing chips such as the TMS32020 or the DSP56000) to 10 (for common microprocessors), therefore eqn. (3.48) will be in the range 4.25 to 17.75.

3.7.2 Shell Method versus the Ordered Method

The ordered scheme has the advantage in that only selected components are removed from either Hadamard or Fourier space. The penalty incurred in such an operation is that the addresses of those components must also be transmitted. The shell approach has the advantage of not transmitting any addresses since the receiver will always know the order in which frequency/sequency components emanate from the origin of the respective transform space. We can quantify these observations in terms of the extra bits per pixel required for transmission. Throughout the following discussion, we assume an LxL analysis frame with each pixel (either in the spatial or transform domain), and each parameter, occupying B bits.

Shell Method

Transmission of K_h Hadamard shells requires $K_h(K_h+2)$ components of B bits per component. The Hadamard overhead (HOHD) for the shell method is thus:

$$HOHD_{shell} = K_h(K_h+2)B/L^2 \quad \text{bits per pixel} \tag{3.49}$$

Transmission of K_f Fourier shells requires $2K_f(K_f+1)$ components of 2B bits per component (since each component is complex). Hence the Fourier overhead (FOHD) for the shell method is

$$FOHD_{shell} = 4K_f(K_f+1)B/L^2 \quad \text{bits per pixel} \tag{3.50}$$

Using the worst-case experimental data given in Figure 3.18 and 3.21, we see that typically (for a 95% stabilization rate), $K_f = 4$ and $K_h = 4$. This gives (with B=8)

$$HOHD_{shell} = 0.188 \quad \text{bits per pixel}$$

and

$$FOHD_{shell} = 0.625 \quad \text{bits per pixel.}$$

Ordered Method

Transmission of K_h Hadamard components requires K_hB. The addresses of each component in Hadamard space will occupy at most $2\log_2 L$ bits. Hence, the Hadamard overhead for the ordered method is

$$HOHD_{ord} = K_h(B+2\log_2 L)/L^2 \quad \text{bits per pixel} \tag{3.51}$$

Transmission of K_f Fourier components requires $2K_fB$ bits. Recalling that approximately half of the Fourier components are redundant, the address of each component will occupy at most $(2\log_2 L-1)$ bits. The Fourier overhead for the ordered method is thus

$$FOHD_{ord} = K_f(2B-1+2\log_2 L)/L^2 \quad \text{bits per pixel} \tag{3.52}$$

Using the worst-case experimental data given in Figures 3.18 and 3.21, we see that typically (for a 95% stabilization rate), $K_f = 5$ and $K_h = 4$. This gives (with B=8)

$$HOHD_{ord} = 0.070 \quad \text{bits per pixel}$$

and

$$FOHD_{ord} = 0.122 \quad \text{bits per pixel.}$$

3.8 References

1 **AC Bajpai, LR Mustoe** and **D Walker**: 'Advanced engineering mathematics', Wiley, Great Britain, 1979.

2 **WH Press, BP Flannery, SA Teukolsky** and **WT Vetterling**: 'Numerical recipes – the art of scientific computing' Cambridge University Press, New York, 1987.

3 **DE Dudgeon** and **RM Mersereau**: 'Multidimensional digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

4 **JS Lim**: 'Two-dimensional signal processing. Tutorial B', *Int. Symp. on Sig. Proc. and its Applications*, Brisbane, Australia, August 24-28, 1987.

5 **SL Marple**: 'Digital spectral analysis (with applications)', Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

6 **RP Freese**: 'Optical disks become erasable', *IEEE Spectrum*, Feb. 1988, pp.41-45.

7 **A Ikonomopoulos** and **M Kunt**: 'High compression image coding via directional filtering', Signal Processing, Vol. 8, No. 2, 1985, pp.179-203.

8 **M Kunt, A Ikonomopoulos** and **M Kocher**: 'Second-generation image-coding techniques', *Proc. IEEE*, Vol. 73, No. 4, 1985, pp.549-574.

9 **DT Nguyen** and **M Andrews**: 'Edge detection via direction filtering', *Proc. 1st Int. Conf. on Future Advances in Comp.*, Christchurch, New Zealand, Feb 16-21, 1986.

10 **RN Strickland**: 'Transforming images into block stationary behavior', *Applied Optics*, Vol. 22, No. 10, 15 May 1983, pp.1462-1473.

11 **PA Maragos, RW Schafer** and **RM Mersereau**: 'Two-dimensional linear prediction and its application to adaptive predictive coding of images', *IEEE Trans. Acoustics Speech and Sig.Proc.*, Vol. ASSP-32, No.6, 1984, pp.1213-1229.

12 **A Papoulis**: 'Probability, random variables and stochastic processes', 2nd. Ed., McGraw-Hill, Japan, 1984.

13 **WF Osgood**: 'Topics in the theory of functions of several complex variables', Dover, New York, 1966.

14 **DT Nguyen, DJ Knowles** and **M Andrews**: 'A new technique in hi-low coding of images', *Proc. ISSPA 87 (Int. Symp. on Sig. Proc. and its Applications)*, Brisbane, Australia, August 24-28, 1987, pp.662-666.

15 **M Andrews** and **DT Nguyen**: 'Techniques for improving the stability rate of linear predictive image coding schemes', *IEE Proc. Part E: Computers and Digital Techniques*, Vol. 135, No. 6, 1988, pp.298-306.

16 **WK Pratt, J Kane** and **HC Andrews**: 'Hadamard transform image coding', *Proc. IEEE*, Vol. 57, No. 1, 1969, pp.58-68.

17 **JW Manz**: 'A sequency ordered fast Walsh transform', *IEEE Trans. Audio and Electroacoustics*, Vol. AU-20, No. 3, 1972, pp.204-205.

18 **JG Proakis**: 'Digital communications', McGraw-Hill, Japan, 1983.

19 **NS Jayant** and **P Noll**: 'Digital coding of waveforms (principles and applications to speech and video)', Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

20 **MF Barnsley** and **AD Sloan**: 'A better way to compress images', BYTE, January, 1988, pp.215-223.

Appendix 3A

Directional Filters with Optimal Passband Shaping [1]†

3A.1 Introduction

This appendix is intended to expand on the work presented in Section 3.3.2.3 regarding directional filtering. Specifically, a modification to the filter used in [2] and [10] will be introduced which improves the edge detection abilities of the filter by incorporating an optimal edge detector. Section 3A.2 discusses the question of directional resolution and the parameters involved. Impulse response windowing, an important factor in the implementation phase of the filter, is presented in Section 3A.3. The optimal edge detector, and its integration into the directional filter, is introduced in Section 3A.4, followed by numerical results in Section 3A.5.

3A.2 Directional Resolution

The directional resolution depends on the number of directional filters used. Very fine directional resolution requires a large number of filters which places a limit on the minimum length of an edge which can be detected because of the lowpass bandwidth of the filter. Refering to Figure 3A.1, the lowpass bandwidth depends on the value of ω_1: being relatively narrow for small ω_1 and wider for large ω_1. It seems reasonable to consider the mean lowpass bandwidth, $\Delta\omega_m$, being the average of these two extremes. $\Delta\omega_m$ is related to the angular bandwidth, φ_c, via (see Figure 3A.1)

$$\Delta\omega_m = \frac{1}{2}(\omega_c+\pi)\sin\frac{\pi}{2n} \qquad (3A.1)$$

since $\varphi_c = \pi/(2n)$. An estimate can now be made of the minimum edge length, Δx, which can be detected with a single filter. Using eqn. (3A.1) and the uncertainty principle [3,p.160], which states

$$\Delta x\,\Delta\omega_m \geq \frac{1}{2}, \qquad (3A.2)$$

† Reference numbers refer to the list of references at the end of this appendix.

we obtain

$$\Delta x \geq \frac{1}{(\omega_c+\pi)\sin\frac{\pi}{2n}} \tag{3A.3}$$

Now $\sin[\pi/(2n)] \approx \pi/(2n)$ to within 1% if $|\pi/(2n)| < 0.244$ [4,p.232]. This last condition is satisfied if $n \geq 7$, which is always the case, hence eqn. (3A.3) may be simplified to

$$\Delta x \geq \frac{2n}{\pi(\omega_c+\pi)} \tag{3A.4}$$

The equation above loosely quantifies the statements made at the beginning of this section since it is seen that the minimum edge length which may be detected is linearly proportional to n, the number of directional filters.

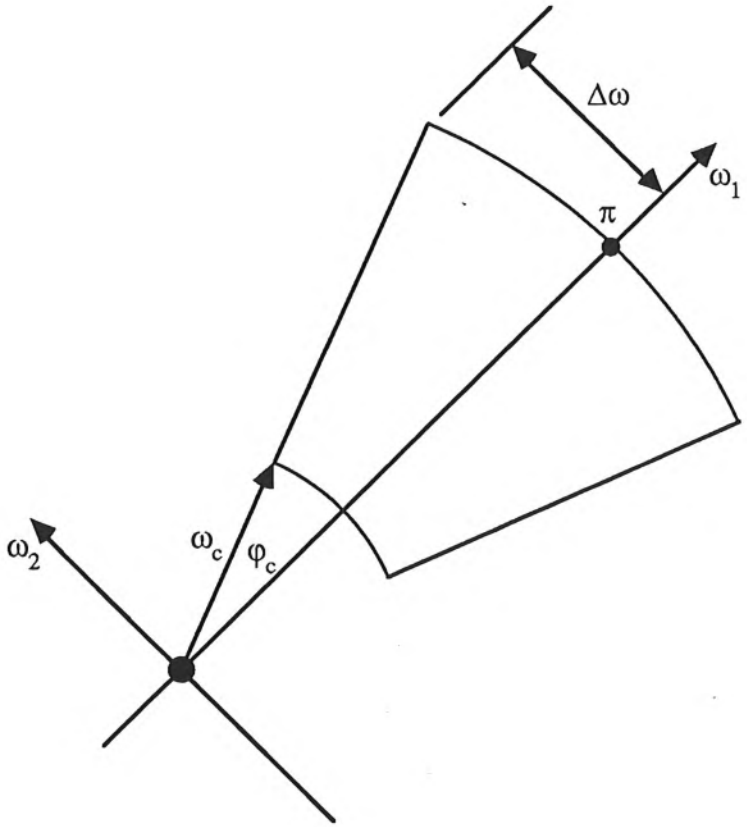


Figure 3A.1: *Relationship between lowpass and angular bandwidths.*

3A.3 Impulse Response Windowing

There are four main issues to address when considering the impulse response window. Firstly, it is desirable to *not* window the impulse response in the interests of directional selectivity. A windowed impulse response will broaden the frequency response in the lowpass direction, thereby

admitting edges in a range greater then $\pm\varphi_c$ (see Figure 3A.1). Secondly, windowing is advantagous because it reduces the region of support of the filter, making an implementation less computationally intensive. Thirdly, windowing reduces the edge extension problem encountered in narrowband lowpass filters, whereby insufficient frequency components remain in the passband to sharply define the end of an edge. Finally, and most importantly, windowing has the effect of reducing the oscillations resulting from the extremely narrow transition band in the principal direction (Gibbs phenomenon).

Bearing these comments in mind, we seek a window which is, on the one hand bandlimited to maintain the directional selectivity of the filter, and on the other hand spatially limited to satisfy the remaining three requirements listed above. Since the only spatially limited and frequency limited signal is the zero signal [5,p.30], a suitable compromise, which minimises the space-bandwidth product, is the circularly symmetric Gaussian window:

$$g(x,y) \equiv g(r) = \exp\left(-r^2/2\sigma_w^2\right), \qquad r^2 = x^2 + y^2 \tag{3A.5}$$

with Fourier transform

$$G(\omega_1,\omega_2) \equiv G(\omega) = K \exp\left(-\sigma_w^2\omega^2/2\right), \qquad \omega^2 = \omega_1^2 + \omega_2^2 \tag{3A.6}$$

The '1/e' bandwidth of $G(\omega)$ is computed from eqn. (3A.6) as

$$\Delta\omega = \sqrt{2}/\sigma_w \tag{3A.7}$$

Similarly, the spatial signal is 1/e times its maximum value when

$$r = \sqrt{2}\,\sigma_w$$

giving an approximate 'spatial extent'

$$\Delta r = 2r = 2\sqrt{2}\,\sigma_w \tag{3A.8}$$

Since the gaussian window in eqn. (3A.5) multiplies the impulse response of the filter, Δr should be larger than the minimum edge length Δx defined in eqn. (3A.4).

3A.4 The Optimal Directional Filter

The main drawback with the filter structure proposed by Ikonomopoulos and Kunt [2] is suboptimality with respect to edge detection. That is, despite impulse response windowing, a single edge in the filter output produces several false edges in the filter output because of ringing. (Recall that a zero crossing supposedly signifies the presense of an edge). In this section, we incorporate the optimal frequency domain function for edge detection (introduced by Shanmugam *et al* in [6]) directly into the frequency response in the principal direction of the directional filters.

The Optimal Edge Detector

The derivation of the optimal edge detector is quite straightforward and relies on some interesting properties of prolate spheroidal wavefunctions. The following definitions are required:

f(x) = image intensity along a line orthogonal to the edge direction.
h(x) = impulse response of the optimal edge detector.
g(x) = output of the edge detector.

Shanmugam's method is to maximise the filter output in some small region $\pm I/2$ of the real edge, which is assumed, for convenience, to be at the origin. Thus the metric

$$\gamma = \frac{\int_{-I/2}^{I/2} |g(x)|^2 \, dx}{\int_{-\infty}^{\infty} |g(x)|^2 \, dx} \tag{3A.9}$$

is a maximum given f(x) and the optimal impulse response h(x). In order to find g(x) (and hence h(x)), we expand g(x) as an infinite sum of prolate spheroidal wavefunctions (PSWs), $\psi_n(x)$:

$$g(x) = \sum_{n=0}^{\infty} a_n \psi_n(x) \tag{3A.10}$$

The PSWs have the remarkable property of being orthonormal on the real number line

$$\int_{-\infty}^{\infty} \psi_m(x) \psi_n(x) \, dx = \delta_{mn} \tag{3A.11}$$

(where δ_{mn} is the Kronecker delta function [7,p.5]) and orthogonal over any finite interval

$$\int_{-I/2}^{I/2} \psi_m(x) \psi_n(x) \, dx = \lambda_m \delta_{mn} \tag{3A.12}$$

where λ_m is a function of $I/2$, and each λ_m is ordered such that

$$\lambda_0 > \lambda_1 > \cdots > 0 \tag{3A.13}$$

Using eqns. (3A.10), (3A.11) and (3A.12), eqn. (3A.9) may be written as

$$\gamma = \frac{\sum_{n=1}^{\infty} \lambda_n |a_n|^2}{\sum_{n=1}^{\infty} |a_n|^2} \leq \lambda_1 \tag{3A.14}$$

The inequality in eqn. (3A.14) is obtained by arguing that g(x) must be an odd function, and then invoking eqn. (3A.13). The maximum value of γ is achieved when $a_n = 0, n \geq 2$. Hence

$$g_{opt}(x) = a_1 \psi_1(x) \tag{3A.15}$$

By assuming that f(x) is a step input, that is

$$f(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

so that

$$F(\omega) = \frac{1}{j\omega} + \pi \delta(\omega)$$

Shanmugam *et al* show that

$$H_{opt}(\omega) = K\omega \psi_1(\omega I/2\Omega), \quad |\omega| \leq \Omega \tag{3A.16}$$

where Ω is the bandwidth of the PSWs. A convenient asymptotic version of eqn. (3A.16) was shown by Lunscher [8] to be

$$H(\omega) = K\omega^2 \exp(-I\omega^2/4\Omega), \quad |\omega| \leq \Omega \tag{3A.17}$$

(A similar expression derived by Shanmugam *et al* [6] contains a dimension error). The asymptotic version of eqn. (3A.16) is identical to an edge detector developed by Marr and Hildreth [9], who decided on the optimality of eqn. (3A.17) by alternative means.

Integration into the Directional Filter

Integration of eqn. (3A.17) into the directional filter is now extremely simple since H(ω) represents the filter frequency response in the principal direction of the filter. Before we do so, some simplifications are possible. For digital signal processing applications we have $\Omega = \pi$. Also, H(ω) has a local minimum when

$$\omega_0^2 = \frac{4\Omega}{I} = \frac{4\pi}{I} \tag{3A.18}$$

at which point

$$H(\omega_0) = \omega_0^2 K e^{-1} \tag{3A.19}$$

Substituting eqns. (3A.18) and (3A.19) into (3A.17), and rearranging, we obtain the normalised frequency response

$$H_{norm}(\omega) = \frac{H(\omega)}{H(\omega_0)} = (\omega/\omega_0)^2 \exp\left[1-(\omega/\omega_0)^2\right] \tag{3A.20}$$

Examination of eqn. (3A.20) reveals that the task of designing $H_{norm}(\omega)$ for a particular application

reduces to choosing a suitable position for the filter maximum, ω_0. For example, ω_0 may be constrained in the following way. In the interests of noise suppression and minimising the effects of aliasing at the Nyquist frequency ($\omega = \pi$), we require $H_{norm}(\omega)$ to be some appropriately small number, say ε. Using

$$H_{norm}(x) = x^2 \exp(1-x^2), \quad x = \omega/\omega_0$$

with $\varepsilon = 0.001$ (i.e. 0.1%), a quick numerical search shows that $x = \pm 0.019184$ and ± 3.198971. Discarding the roots of small moduli, the required value of ω_0 (with $\omega = \pi$) is

$$\omega_0 = 0.982064 \tag{3A.21}$$

Since ω_0 and the extent of energy localization are related through eqn. (3A.18), we find

$$I = 13.029 \tag{3A.22}$$

Thus, edges which span approximately 13 pixels are detected optimally for ω_0 defined in eqn. (3A.21).

3A.5 Results

The optimal edge detector frequency response (in the principal direction) is shown in Figure 3A.2. The frequency response of the filter before windowing is shown in Figure 3A.3. Figure 3A.4 shows the step response of the filter used by Ikonomopoulos and Kunt [2]. Note the ringing about the true edge location. The amplitude of the oscillations is such that a local threshold, necessary to discriminate between true and spurious edges, needs to be set quite high. This results in edges with gradual slope being overlooked during the thresholding process. The step response of the optimal directional filter is shown in Figure 3A.5. The most important feature to note in this response is the reduced ringing about the true edge. This means that edges are detected more reliably, and also a greater range of edges may be successfully detected since the threshold can be set to a lower value.

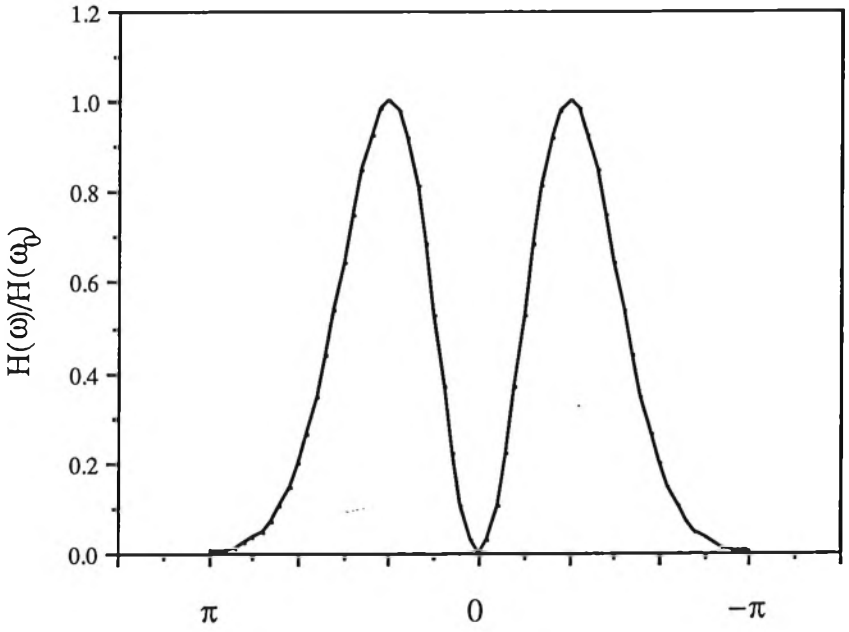


Figure 3A.2: *Optimal edge detector for I = 13 and $\omega_0 = 0.982$.*

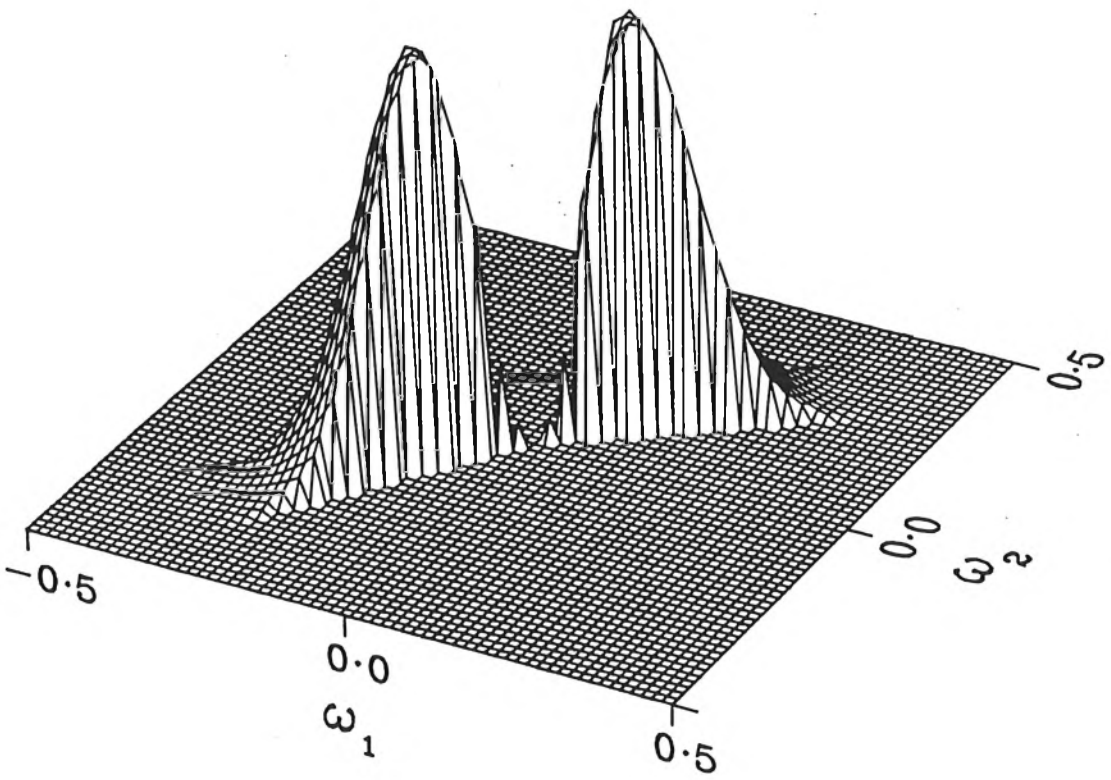


Figure 3A.3: *Directional filter frequency response before windowing.*

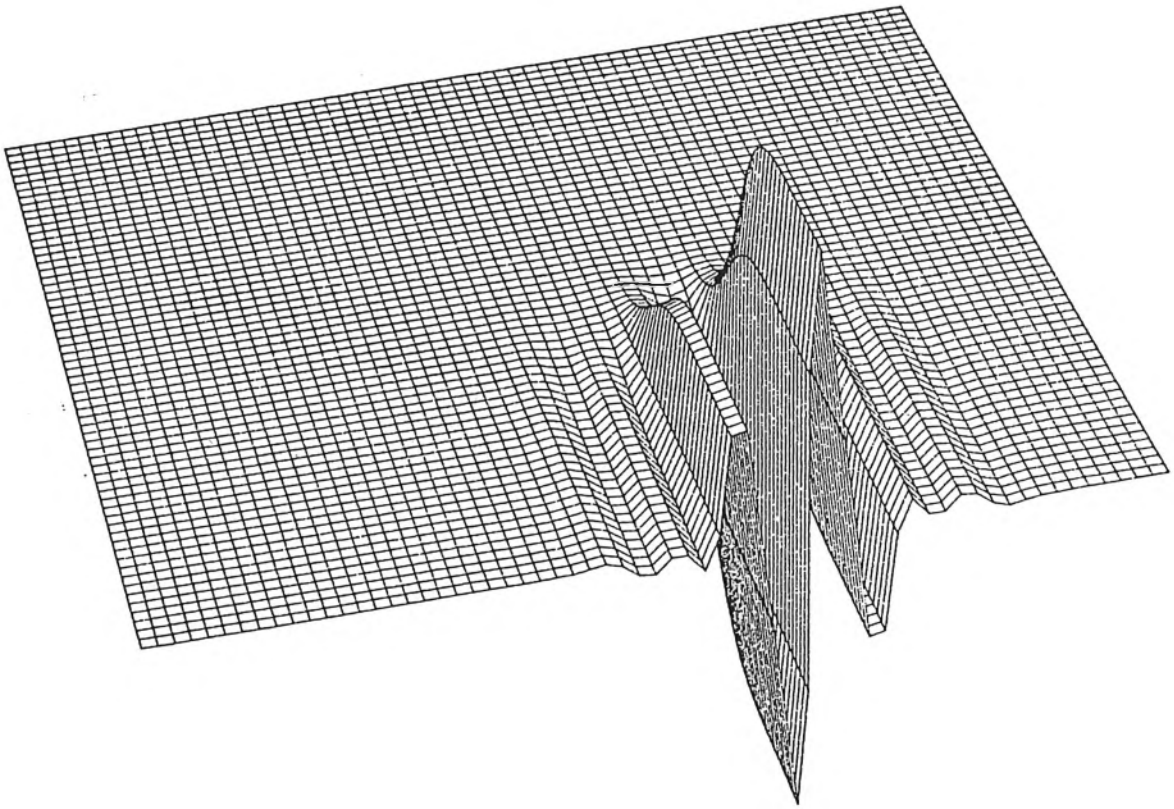


Figure 3A.4: *Step response of the filter used in [2]. Note the severe ringing about the true edge position.*

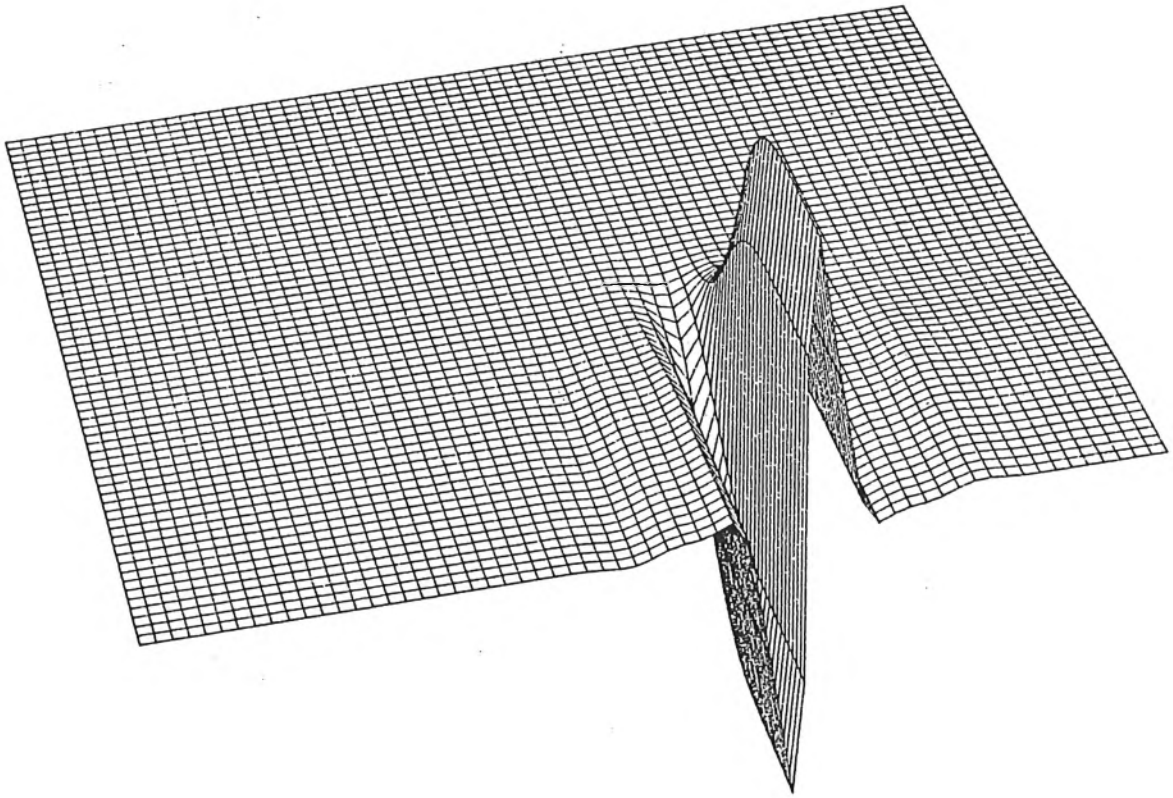


Figure 3A.5: *Step response of the optimal directional filter. Note the reduced ringing about the true edge position.*

3A.6 References

1 **DT Nguyen** and **M Andrews**: 'Edge detection via direction filtering', *Proc. 1st Int. Conf. on Future Advances in Comp.*, Feb. 16-21, Christchurch, New Zealand, 1986.

2 **A Ikonomopoulos** and **M Kunt**: 'High compression image coding via directional filtering', *Signal Processing*, Vol. 8, No. 2, April 1985, pp.179-203.

3 **R Bracewell**: 'The Fourier transform and its applications', McGraw-Hill, New York, 1979.

4 **JJ Tuma**: 'Engineering mathematics handbook', 2nd Ed., McGraw-Hill, New York, 1979.

5 **P Henrici**: 'Applied and computational complex analysis. Vol. 3', Wiley, New York, 1986.

6 **KS Shanmugam, FM Dickey** and **JA Green**: 'An optimal frequency domain filter for edge detection in digital pictures', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 1, 1979, pp.37-49.

7 **AC Bajpai, LR Mustoe** and **D Walker**: 'Advanced engineering mathematics', Wiley, Great Britain, 1979.

8 **WHHJ Lunscher**: 'The asymptotic optimal frequency domain filter for edge detection', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 6, 1983, pp.678-680.

9 **D Marr** and **E Hildreth**: 'Theory of edge detection', *Proc. Royal Soc. Lond.*, **B207**, 1980, pp.187-217.

10 **M Kunt, A Ikonomopoulos** and **M Kocher**: 'Second-generation image-coding techniques', *Proc. IEEE*, Vol. 73, No. 4, 1985, pp.549-574.

Digital Filter Stability, Stability Tests and Stabilization Methods

4.1 Introduction

This chapter examines the problem of determining the stability condition of a multidimensional digital system or filter. It transpires that the question of stability is much more complicated when the number of independent variables is greater than one. This is directly attributable to the important differences that exist between multidimensional polynomials and the more familiar one-dimensional polynomials.

The basic theory of multidimensional systems is presented in Section 4.2 along with the concept of bounded-input bounded-output (BIBO) stability. Section 4.3 presents the classical necessary and sufficient conditions required for stability, and highlights some of the problems involved in implementing such tests. The graphical device known as the *rootmap* is introduced in Section 4.4 followed by an efficient and robust technique for its calculation in Section 4.5. Section 4.6 critically examines a necessary test used by Maragos *et al* in the context of 2-D linear predictive coding. Finally, Sections 4.7 and 4.8 introduce a new and improved suite of necessary stability tests, and examines the test's performance relative to both Maragos' test and 'sure fire' methods.

4.2 Multidimensional Systems and BIBO Stability

Consider the general difference equation relating the input sequence $x(\mathbf{n})$ to an output sequence $y(\mathbf{n})$, for the N-dimensional digital system illustrated in Figure 4.1:

$$y(\mathbf{n}) = \sum_{\mathbf{k}\in R_a} a(\mathbf{k})\, y(\mathbf{n}-\mathbf{k}) \; + \sum_{\mathbf{k}\in R_b} b(\mathbf{k})\, x(\mathbf{n}-\mathbf{k}) \; ; \quad \mathbf{0} \notin R_a \qquad (4\text{-}1)$$

The N-vector $\mathbf{m} = (m_1, m_2, \ldots, m_N)$ locates the co-ordinate of a particular sample of a,b,x or y. R_a specifies the region of support of the output mask $\{a(\mathbf{k})\}$ while R_b specifies the corresponding region of support of the input mask $\{b(\mathbf{k})\}$ (Figure 4.2). The N-dimensional filter is completely defined by the coefficients $\{a(\mathbf{k})\}$, $\{b(\mathbf{k})\}$ and the support regions R_a and R_b. Note that if $\mathbf{0} \in R_a$

then the output y(**n**) is no longer recursively computable but must be evaluated using iterative techniques [1,p.224]. The current work will consider only systems expressible via recursively computable equations.

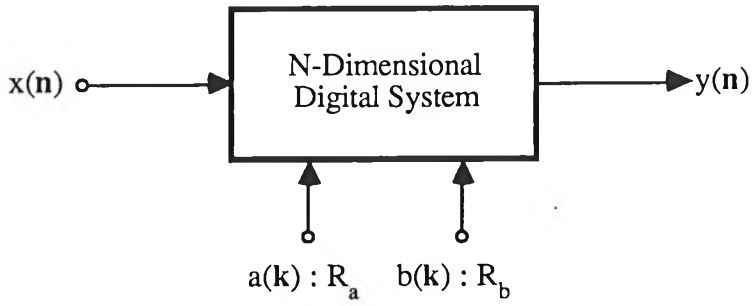


Figure 4.1: *Single-input, single-output N-dimensional digital system. The characteristics of the system are determined by the parameters a(**k**) and b(**k**).*

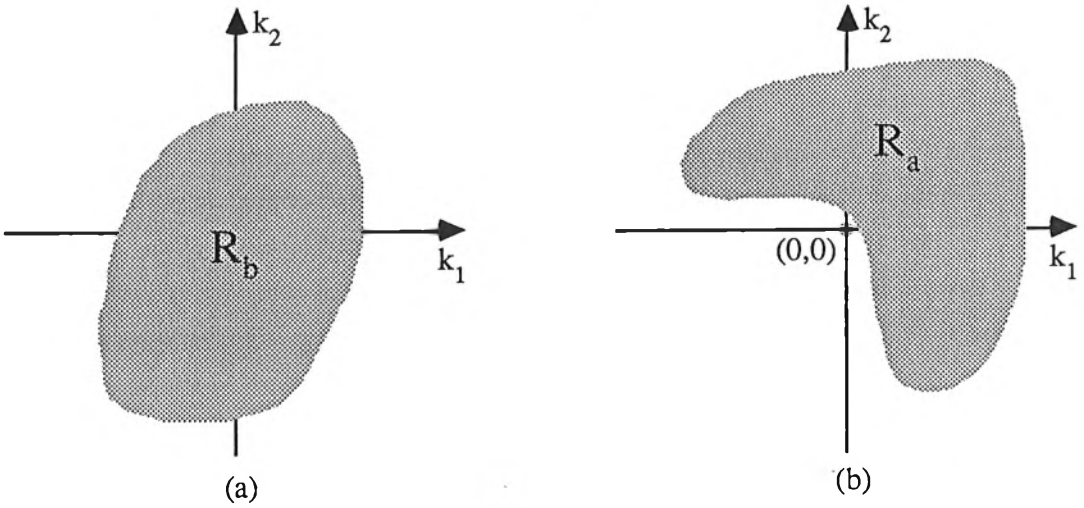


Figure 4.2: a *Region of support for the input mask. i.e. b(**k**)≠0 if **k**∈R_b.*

b *Region of support for the output mask. i.e. a(**k**)≠0 if **k**∈R_a.*

Taking the N-D Z-Transform of eqn. (4.1), and using the notation of Section 2.2, we have

$$Y(\mathbf{z}) = \sum_{\mathbf{k}\in R_a} a(\mathbf{k})Y(\mathbf{z})\mathbf{z}^{\mathbf{k}} + \sum_{\mathbf{k}\in R_b} b(\mathbf{k})X(\mathbf{z})\mathbf{z}^{\mathbf{k}}$$

and form the transfer function, or system function, H(**z**) defined as

$$H(\mathbf{z}) = \frac{Y(\mathbf{z})}{X(\mathbf{z})} = \frac{\sum_{\mathbf{k}\in R_b} b(\mathbf{k})\mathbf{z}^{\mathbf{k}}}{1-\sum_{\mathbf{k}\in R_a} a(\mathbf{k})\mathbf{z}^{\mathbf{k}}}$$

$$= \frac{\sum_{\mathbf{k}\in R_b} b(\mathbf{k})\mathbf{z}^{\mathbf{k}}}{\sum_{\mathbf{k}\in R_c} c(\mathbf{k})\mathbf{z}^{\mathbf{k}}} = \frac{N(\mathbf{z})}{D(\mathbf{z})} \qquad (4\text{-}2)$$

where $c(\mathbf{k}) = -a(\mathbf{k})$ for $\mathbf{k} \in R_a$, $c(\mathbf{0}) = 1$ and $R_c = R_a \cup \mathbf{0}$.

If $H(\mathbf{z})$ is analytic in some neighbourhood of $\mathbf{0}$ (i.e. $D(\mathbf{z}) \neq 0 \ \forall \ z_i$: $|z_i| < \varepsilon$, i=1...N, where ε is arbitrarily small) then we may expand $H(\mathbf{z})$ about $\mathbf{0}$ as locally convergent Taylor series [3,p.5]:

$$H(\mathbf{z}) = \sum_{\mathbf{k}\in R_h} h(\mathbf{k})\mathbf{z}^{\mathbf{k}} \qquad (4\text{-}3)$$

where R_h is the region of support of $h(\mathbf{k})$, the impulse response of the system. Note that even though $\eta(R_c) < \infty$ and $\eta(R_b) < \infty$, we usually have $\eta(R_h) = \infty$.

If we put $Y(\mathbf{z}) = H(\mathbf{z})\, X(\mathbf{z})$, then using eqn. (4.3), we have

$$Y(\mathbf{z}) = \sum_{\mathbf{k}\in R_h} h(\mathbf{k})X(\mathbf{z})\mathbf{z}^{\mathbf{k}}$$

and taking the inverse Z-Transform:

$$y(\mathbf{n}) = \sum_{\mathbf{k}\in R_h} h(\mathbf{k})\, x(\mathbf{n}-\mathbf{k}) \qquad (4\text{-}4)$$

we obtain the well-known convolutional input-output summation.

If we now apply a bounded input sequence $x(\mathbf{n})$, such that $|x(\mathbf{n})| \leq P$ (with $P \geq 0$), what can be said about the modulus of the output sequence $y(\mathbf{n})$? Assuming $x(\mathbf{n}) = P$, then using eqn. (4.4) we have

$$|y(\mathbf{n})| = \left| \sum_{\mathbf{k}\in R_h} h(\mathbf{k})\, P \right|$$

$$\leq P \sum_{\mathbf{k}\in R_h} |h(\mathbf{k})|$$

Since P is finite, the output sequence will be bounded, if and only if

$$\sum_{\mathbf{k}\in R_h} |h(\mathbf{k})| < \infty \tag{4-5}$$

If a sequence $s(\mathbf{n})$ is absolutely summable, (as in eqn. (4.5)), then we say that $s(\mathbf{n})\in l_1$. If $s(\mathbf{n})$ satisfies the weaker notion of square summability:

$$\sum_{\mathbf{k}\in R_h} |s(\mathbf{k})|^2 < \infty$$

then we say that $s(\mathbf{n})\in l_2$. Hence our digital system given in eqn. (4.1) is BIBO stable if, and only if, $h(\mathbf{n})\in l_1$. If $\eta(R_h) < \infty$ then $y(\mathbf{n})$ is always bounded, since a finite sum of finite terms is always finite. If $\eta(R_h) = \infty$, then $y(\mathbf{n})$ may or may not be bounded.

Unfortunately, eqn. (4.5) is of little practical use since R_h is usually semi-infinite in extent and there are no finite algorithms for evaluating an infinite sum of arbitrary terms. A more insightful formulation results if we examine the effect eqn. (4.5) has on the singularities of $H(\mathbf{z})$.

From Chapter 2, Section 2.5.3, we saw that

$$h(\mathbf{k}) = (j2\pi)^{-N} \int_{C_1} \cdots (N) \int_{C_N} H(\mathbf{z}) \mathbf{z}^{-\mathbf{k}-1} \, d\mathbf{z} \tag{4-6}$$

where $d\mathbf{z}$ is the elemental *complex volume* $dz_1 dz_2 \ldots dz_N$ and the closed contours $C_1,\ldots,C_N$ all lie completely within the region of convergence of $H(\mathbf{z})$. Consequently

$$|h(\mathbf{k})| = (2\pi)^{-N} \left| \int_{C_1} \cdots (N) \int_{C_N} H(\mathbf{z}) \mathbf{z}^{-\mathbf{k}-1} \, d\mathbf{z} \right|$$

or

$$|h(\mathbf{k})| \le (2\pi)^{-N} \int_{C_1} \cdots (N) \int_{C_N} |H(\mathbf{z})\mathbf{z}^{-1}| . |\mathbf{z}^{-\mathbf{k}}| \, d\mathbf{z} \tag{4-7}$$

If we restrict our attention to $h(\mathbf{k})$ with support in the generalized first quadrant (this formulation is the most useful), that is

$$h(\mathbf{k}) = 0 \qquad \forall\, k_i < 0 \quad \text{for} \quad i = 1,2,\ldots N$$

so that R_h is simply the N-fold cartesian product of the interval $[0,\infty)$ with itself, then eqn. (4.5) becomes

$$\sum_{\mathbf{k}\in R_h} |h(\mathbf{k})| = \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} \cdots \sum_{k_N=0}^{\infty} |h(k_1,k_2,\cdots,k_N)| < \infty$$

which, when eqn. (4.7) is incorporated, requires

$$\sum_{\mathbf{k}\in R_h} |h(\mathbf{k})| \le (2\pi)^{-N} \int_{C_1}(N) \int_{C_N} |H(\mathbf{z})\mathbf{z}^{-1}| \sum_{k_1=0}^{\infty} \cdots \sum_{k_N=0}^{\infty} |\mathbf{z}|^{-k} d\mathbf{z} < \infty \qquad (4\text{-}8)$$

for BIBO stability. The N summations

$$\sum_{k_1=0}^{\infty} \cdots \sum_{k_N=0}^{\infty} |\mathbf{z}|^{-k}$$

are only convergent for $\mathbf{z}$ outside $\overline{\mathbf{U}}^N$, and since the contours $C_1,...,C_N$ must lie within the region of convergence of $H(\mathbf{z})$, $H(\mathbf{z})$ must be analytic on and within the unit polydisc for a stable system. Formally this is stated as

$$h(\mathbf{n}) \in l_1 \quad \text{if and only if} \quad D(\mathbf{z}) \neq 0, \ \mathbf{z} \in \overline{\mathbf{U}}^N \qquad (4\text{-}9)$$

The stability condition embodied in eqn. (4.9) is the original criterion established by Shanks [4]. Conceptually the result is straightforward and is simply an extension of the well known result for 1-D systems. However, implementing the result in a useful algorithm is far from simple and is considerably more complicated than the 1-D case.

For N=1, eqn. (4.9) states

$$h(n) \in l_1 \quad \text{if and only if} \quad D(z) \neq 0 \quad \text{for} \quad |z| \le 1$$

Thus H(z) must be analytic on and inside the unit circle. Thus it is a straightforward matter to ascertain the membership of h(n) in l_1 by checking to see if any of the zeros of D(z) lie within the unit circle. This may be done by invoking the Fundamental Theorem of Algebra [5,p.137] and actually finding all of the zero positions (or finding their moduli, which is faster), or by using one of the many established tests for stability. The latter method is to be favoured since it relies on simple arithmetic operations on the coefficients of D(z). In essence, the geometry of the zeros of D(z) is somehow elegantly reflected in the polynomial coefficients. Unfortunately the situation is considerably more complicated for N>1. The root of the problem lies in the fact that there is no Fundamental Theorem of Algebra for polynomials with dimension greater than 1. Thus, while we can always write

$$\sum_{k=0}^{K} a_k z^k = a_K \prod_{k=1}^{K} (z-b_k) \qquad (4\text{-}10)$$

for 1-D polynomials, no such factorization for N≥2 is possible. Note that the K+1 degrees of freedom of the coefficients a_k on the LHS of eqn. (4.10) is matched by exactly K+1 degrees of freedom of the zero locations b_k, and a_K, on the RHS.

If we now choose N=3, we can almost never (except in contrived situations) obtain a factorization similar to eqn. (4.10):

$$\sum_{r=0}^{R} \sum_{s=0}^{S} \sum_{t=0}^{T} \alpha_{rst} z_1^r z_2^s z_3^t \neq \alpha_{RST} \prod_{r=1}^{R} (z-\beta_r) \prod_{s=1}^{S} (z-\gamma_s) \prod_{t=1}^{T} (z-\delta_t) \qquad (4\text{-}11)$$

The isolated zero locations β_r, γ_s, and δ_t, possess insufficient degrees of freedom to represent every possible polynomial expressible on the LHS of eqn. (4.11). Note that the LHS of eqn.(4.11) has (R+1)(S+1)(T+1) degrees of freedom whereas the RHS has R+S+T+1, and we always have (R+1)(S+1)(T+1) > R+S+T+1 except for R=S=T=0. Consideration of eqn. (4.11) may lead us to suspect a factorization based on non-linear terms. However, it transpires that in general, polynomials with N≥2 are *irreducible*. It has recently been shown that the set of polynomials which are reducible over $\mathbf{C}^N$ (N>1) has zero measure [6]. That is, for arbitrarily chosen coefficients, the polynomial will almost never possess factors of any order. Even if it was possible to factorize the polynomial into, say quadratics, it would still be of little use to us as an aid to stability testing, since the zero-sets for N≥2 form continuous surfaces. In fact, for an N-D polynomial defined over the complex field $\mathbf{C}^N$, the zero-set forms a (2N-2)-dimensional surface embedded in 2N-dimensional space; i.e. $\mathbf{C}^N$ or $\mathbf{R}^{2N}$ [7,p.52]. For example, consider the 3-D polynomial

$$G(x,y,z) = xy + z - b$$

defined over $\mathbf{C}^3$ (i.e. 6 dimensions). The condition G(x,y,z)=0 specifies the zero-set of G as

$$z = b - xy \tag{4.12}$$

Note that x, y and z are no longer independent but are related by eqn. (4.12). Since x, y and z are complex, the dimensionality of the space in which the zero-set exists has been reduced by 2, to 4. Note further that we are free to choose any value for x or y in eqn. (4.12), and obtain the corresponding z value for which G(x,y,z)=0. Thus the zero-set of G(x,y,z) is a continuum of points, extending out to infinity in a precisely defined fashion.

It should be obvious from the foregoing discussion that it is exceedingly difficult to imagine the zero-set of an N-D polynomial (N≥2) in the same way that we visualise the familiar pole-zero diagram of 1-D systems. This fact makes the study of multidimensional systems both difficult and fascinating.

An interesting phenomenon in multidimensional systems which has no analogue in 1 dimension is the so-called *non-essential singularity of the second kind,* [7,p.51][1,p.196], hereafter called singularity of the second kind. Such a singularity results when two mutually prime polynomials P(**z**) and Q(**z**) define a rational function H(**z**) = P(**z**)/Q(**z**), but there is a point $\mathbf{z}^{(a)}$, for which $P(\mathbf{z}^{(a)}) = Q(\mathbf{z}^{(a)}) = 0$, and thus H(**z**) is undefined at $\mathbf{z}^{(a)}$. (For example $H(z_1,z_2) = z_1/z_2$ has a singularity of the second kind at **z** = (0,0)). It is a remarkable fact that singularities of this kind can affect (and even *effect*) the stability of multidimensional digital filters. Goodman [8] has shown (for the 2-D case) that Shanks' Theorem (eqn. (4.9)) is insufficient if $H(z_1,z_2)$ has singularities of the second kind on $\mathbf{T}^2$. Specifically, eqn. (4.9) should be restated as

> if $h(n_1,n_2) \in l_1$ then $D(z_1,z_2) \neq 0$ for $(z_1,z_2) \in \overline{\mathbf{U}}^2$
> and $N(z_1,z_2)/D(z_1,z_2)$ has no singularities of the second
> kind on $\overline{\mathbf{U}}^2$, except possibly on $\mathbf{T}^2$ (4-13)

The proof of eqn. (4.13) hinges on the fact that if $H(z_1,z_2)$ has a singularity of the second kind in $\overline{\mathbf{U}}^2$-$\mathbf{T}^2$, then $H(z_1,z_2)$ has a singularity of the first kind (i.e. a pole) in $\mathbf{U}^2$. Goodman [8] has presented 2 different rational functions, each with a singularity of the second kind on $\mathbf{T}^2$, and established the BIBO stability of one and the BIBO instability of the other. In an attempt to extend Goodman's findings to higher dimensions, Swamy *et al* [9] have found that the result in eqn. (4.13) is too restrictive for $H(\mathbf{z})$ defined over $\mathbf{C}^N$, $N\geq 3$. In particular, they demonstrated two $H(\mathbf{z})$ (defined over $\mathbf{C}^3$), each with a singularity of the second kind in $\overline{\mathbf{U}}^3$-$\mathbf{U}^3$, one of which was stable and the other unstable. The important difference between these results is that for two dimensions, it is possible to have singularities of the second kind on $\mathbf{T}^2$ (which is small subset of the boundary $\partial\mathbf{U}^2=\overline{\mathbf{U}}^2-\mathbf{U}^2$) and the filter may be stable; while for three dimensions, the filter may be stable even when singularities of the second kind exist not only on $\mathbf{T}^3$, but in the much larger region $\partial\mathbf{U}^3=\overline{\mathbf{U}}^3-\mathbf{U}^3$. It is suggested by this author that such a richness of behaviour should not be unexpected since, for N=2, any singularities of the second kind must be isolated, whereas for N=3, a 2-D locus of singularities of the second kind exists [7,p.52].

Finally, it may appear from the simplicity of Goodman's contrived equations (to illustrate the effect of numerator zeros on stability), that singularites of the second kind will be very rare in "real" situations. However, it has been recently shown that the set of polynomials with zeros on $\mathbf{T}^2$ has positive measure [10]. This suggests that, in practice, the frequency of occurrence of filters with second-kind singularities may not be negligible.

4.3 Necessary and Sufficient Tests for Stability

Notwithstanding the concluding remarks of the previous section, filters which are devoid of numerator zeros form a large and important class of recursive digital filters. The theorems presented in this section, which will be quoted for N=2, assume that the numerator polynomial is a constant, in which case the results are necessary and sufficient for BIBO stability. If the numerator is not a constant and singularities of the second kind exist on $\mathbf{T}^2$, then the results are simply necessary conditions.

Shanks [4] developed the earliest stability theorem, emboddied in eqn. (4.9) and is restated for completeness:

Shanks' Stability Theorem

Let $H(z_1,z_2) = 1/D(z_1,z_2)$ be a recursive filter with support in the first quadrant. $H(z_1,z_2)$ represents a BIBO-stable filter if and only if

$$D(z_1,z_2) \neq 0, \quad |z_1| \leq 1 \text{ and } |z_2| \leq 1 \tag{4-14}$$

While the above theorem is easy to understand, it is difficult to use because the whole interior of the closed unit bidisc must be searched for zeros. An improvement on Shanks' original theorem was given by Huang [11][12][13]:

Huang's Stability Theorem

Let $H(z_1,z_2) = 1/D(z_1,z_2)$ be a recursive filter with support in the first quadrant. $H(z_1,z_2)$ represents a BIBO-stable filter if and only if the following conditions are satisfied

$$1.\quad D(z_1,0) \neq 0, \quad |z_1| \leq 1 \tag{4-15}$$

$$2.\quad D(z_1,z_2) \neq 0, \quad |z_1| \leq 1 \quad \text{and} \quad |z_2| = 1 \tag{4-16}$$

Note that the first part of Huang's theorem is actually a 1-D stability test. That is, no roots of the 1-D polynomial $D(z_1,0)$ must have magnitudes less than 1. This condition is very easy to test. The second part of the theorem is a great improvement over Shanks' result because the region of C^2 which must be searched for zeros is reduced from $\overline{U}^2$ to $\overline{U}^2$-U_2 (where $U_2 = \{ z_2: |z_2| < 1 \}$). Equation (4.16) may be interpreted as restricting the "image" of the z_2 unit circle to outside the z_1 unit circle under the mapping $D(z_1,z_2) = 0$.

As a further improvement, Strintzis [14] and DeCarlo *et al* [15] independently showed that Huang's theorem could be simplified as follows:

DeCarlo-Strintzis Stability Theorem

Let $H(z_1,z_2) = 1/D(z_1,z_2)$ be a recursive filter with support in the first quadrant. $H(z_1,z_2)$ represents a BIBO-stable filter if and only if the following conditions are satisfied

$$1.\quad D(\alpha,z_2) \neq 0, \quad |z_2| \leq 1 \quad \text{for some } \alpha\text{: } |\alpha| = 1 \tag{4-17}$$

$$2.\quad D(z_1,\beta) \neq 0, \quad |z_1| \leq 1 \quad \text{for some } \beta\text{: } |\beta| = 1 \tag{4-18}$$

$$3.\quad D(z_1,z_2) \neq 0, \quad |z_1| = 1 \quad \text{and} \quad |z_2| = 1 \tag{4-19}$$

Conditions 1 and 2 in the above theorem are clearly 1-D stability conditions and therefore easy to test. Condition 3 is an improvement over Condition 2 of Huang's theorem because now the region which must be searched for zeros has reduced from $\overline{U}^2$-U_2 to T^2. This is equivalent to requiring that the 2-D Fourier Transform of the coefficients d(m,n) of $D(z_1,z_2)$ be free of zeros.

The important point to note from the three foregoing theorems is that, while the "hard" part (conceptually) of the stability test has been reduced from a search for zeros in all of $\overline{U}^2$, to only T^2, it is very difficult to say anything similar about a practical implementation. In principle, a numerical implementation of any one of the theorems requires an infinite amount of computation. For example, eqn. (4.19), which has the least "area" concerned, is usually evaluated using the Discrete

Fourier Transform. If the frequency plane is not sampled often enough, the zeros of $D(z_1,z_2)$ may cross the unit bidisc, and back again, in between DFT samples.

4.4 The Rootmap

Given the 2-D polynomial $G(z_1,z_2)$ defined as

$$G(z_1,z_2) = \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} g(k_1,k_2) z_1^{k_1} z_2^{k_2} \tag{4-20}$$

the condition

$$G(z_1,z_2) = 0 \tag{4-21}$$

defines an implicit mapping between the z_1 and z_2 planes. It is a non-trivial task to extract the explicit algebraic functions

$$z_2 = \zeta_2(z_1) \tag{4-22a}$$

$$z_1 = \zeta_1(z_2) \tag{4-22b}$$

which satisify eqn. (4.21), and except for extremely simple cases, eqn (4.22) must be determined numerically. Such an algorithm is the subject of this section.

The rootmap of a 2-D polynomial $G(z_1,z_2)$ can be represented as a special case of eqn. (4.22) where we restrict each independent z variable to its unit circle. That is

$$z_2 = \zeta_2(e^{j\omega_1}) \equiv \zeta_2(\omega_1) \tag{4-23a}$$

$$z_1 = \zeta_1(e^{j\omega_2}) \equiv \zeta_1(\omega_2) \tag{4-23b}$$

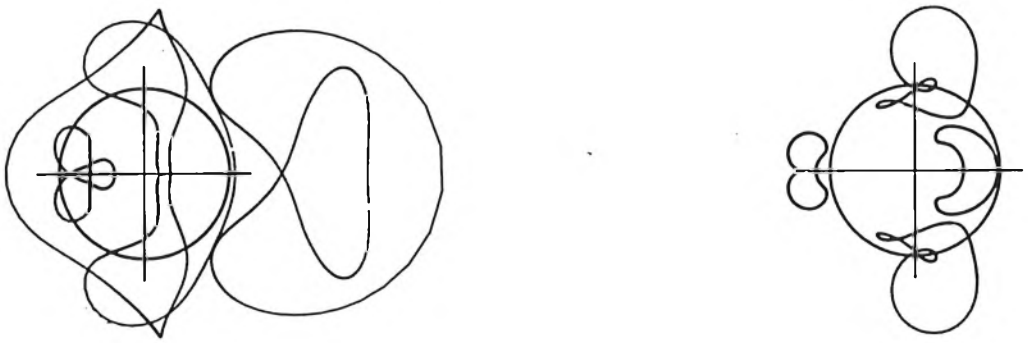
Thus eqn. (4.23a) should be interpreted as the image of the z_1 unit circle in the z_2 plane. Similarly, eqn. (4.23b) is the image of the z_2 unit circle in the z_1 plane. Examination of the two rootmaps informs us immediately of the stability condition of the filter $H(z_1,z_2) = 1/G(z_1,z_2)$ since, according to the DeCarlo-Strintzis Theorem, the filter will be unstable if any rootmap contour is such that

$$|\zeta_2(\omega_1)| \le 1 \quad \text{or} \quad |\zeta_1(\omega_2)| \le 1 \tag{4-24}$$

Note that since the rootmap is a graphical device, it also allows us to ascertain the stability margin of a filter simply by examining the proximity of the contours to each unit circle. Some typical rootmap plots are shown in Figures 4.3, 4.4 and 4.5, for 3x3-, 5x5- and 7x7-degree denominator polynomials, respectively. (The filter coefficients used in Figures 4.3, 4.4 and 4.5 are provided in Appendix 4A).

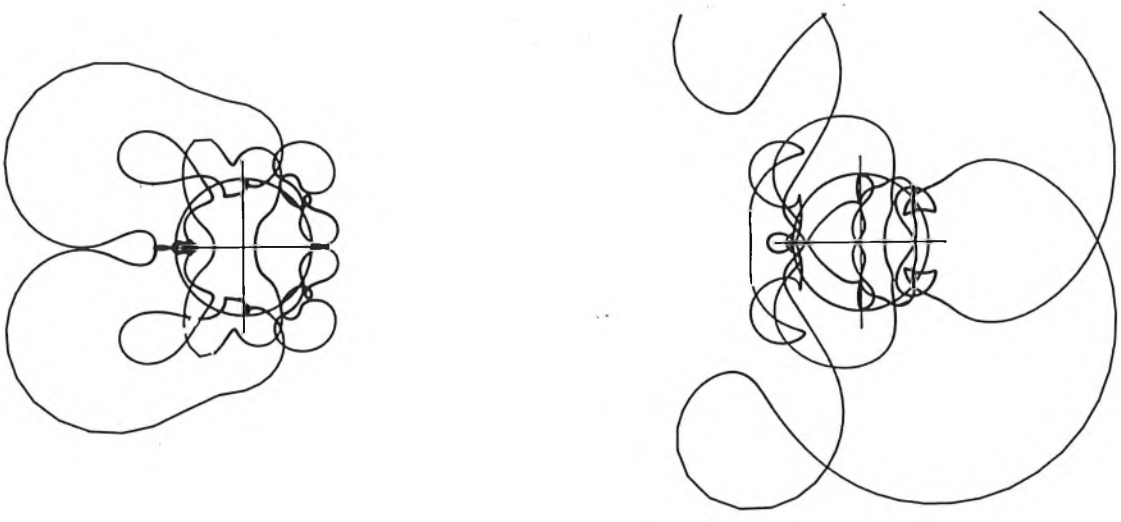
z_1 plane z_2 plane

Figure 4.3: *Typical rootmap for a 3 by 3 denominator polynomial*



z_1 plane z_2 plane

Figure 4.4: *Typical rootmap for a 5 by 5 denominator polynomial*



z_1 plane z_2 plane

Figure 4.5: *Typical rootmap for a 7 by 7 denominator polynomial*

4.5 Robust Rootmap Calculation [16]

If we restrict z_1 to the unit circle, we may write eqn. (4.20) as

$$G(\omega_1,z_2) = \sum_{k_2=0}^{M_2} c_{k_2}(\omega_1) z_2^{k_2} = 0 \tag{4-25}$$

where

$$c_{k_2}(\omega_1) = \sum_{k_1=0}^{M_1} g(k_1,k_2)\, e^{-jk_1\omega_1} \tag{4-26}$$

In the interests of clarity, we will drop the multiple subscripts, by putting $M=M_2$, $m=k_2$, $N=M_1$, $n=k_1$, and consider instead the equations

$$G(\omega_1,z_2) = \sum_{m=0}^{M} c_m(\omega_1)\, z_2^{m} = 0 \tag{4-27}$$

where

$$c_m(\omega_1) = \sum_{n=0}^{N} g(n,m)\, e^{-jn\omega_1} \tag{4-28}$$

Clearly $G(\omega_1,z_2)$ may be treated as a 1-D polynomial in z_2 where the coefficients $c_m(\omega_1)$ are given parametrically in terms of ω_1, and eqn. (4.28) should be recognised as the Discrete Fourier Transform of the rows of g(n,m) and may be efficiently evaluated using the 1-D Fast Fourier Transform. A similar expression could be obtained by treating ω_2 as a parameter and forming $G(z_1, \omega_2)$. For a fixed value of ω_1, the Fundamental Theorem of Algebra ensures that eqn. (4.27) has exactly M zeros. Thus, $G(\omega_1,z_2)$ may be written as

$$G(\omega_1,z_2) = c_M \prod_{m=1}^{M} (1-\zeta_{2(m)}(\omega_1)\cdot z_2) \tag{4-29}$$

where the M zeros $\zeta_{2(m)}$ necessarily depend on ω_1 and hence correspond to M points on the rootmap $z_2 = \zeta_2(z_1)$. It should be noted that, due to the generally irreducible nature of 2-dimensional polynomials, there is only *one* zero surface for $G(z_1,z_2)$ and thus all the zeros are "connected" via the zero sheet. Consequently, even though the M zeros $\zeta_{2(m)}$ seem distinct, they will usually be connected for values of z_1 off the unit circle. In order to trace out the entire rootmap, we simply increment ω_1 and trace the movement of each zero location accordingly.

A procedure for calculating a complete rootmap in the z_2 plane is as follows:

- **Begin**
- Calculate the initial zero-set $\{ z_2^{(i)} \}_0$, i=1,...,M, corresponding to $\omega_1=\omega_{1(initial)}$ (which is usually zero or a random number $\in (-\pi,\pi]$).
- **Repeat**

  Increment ω_1 by a small amount $\delta\omega_1$ and follow the change in the location of each zero $z_2^{(i)}$.

  Until each $z_2^{(i)}$ has traced out a closed contour
- **End**

4.5.1 Implementation

Step1: Calculating the initial zero-set

Since possibly thousands of zeros need to be located as the rootmap contour is mapped, finding the initial zeros does not have to be especially fast. However, since we have no *a priori* information regarding their location, a robust and globally convergent algorithm is required. Many such algorithms exist, such as the method of Laguerre [17,p.380][18,p.263] or the Jenkins-Traub method [17,p.383][19][20]. A bug-free implementation of the Jenkins-Traub method, suitable for a VAX/VMS system, may be found in [21,p.335].

Step 2: Zero Tracking

Once the initial zero-set $\{z_2^{(i)}\}_0$ has been found, ω_1 is incremented by $\delta\omega_1$, causing a small change in each $z_2^{(i)}$ due to continuity in the zero surface. At first, we may be tempted to simply re-apply the global algorithm used in Step 1, however, this will lead to major difficulties. The reason is that, given a particular zero in the new zero-set, say $z_2^{(r)}{}_{(1)}$, it is very difficult to identify the corresponding zero $z_2^{(r)}{}_{(0)}$ in the initial zero-set. This is because the rootmaps are usually very complicated, with the contours crossing each other often. It is therefore pointless to simply look at the relative positions of each zero in the zero-set as most globally convergent algorithms calculate the zeros in increasing order of magnitude. As the rootmap is traced, the modulus of each contour increases and decreases, thereby shifting its position in the ordered set many times.

Since the zeros of a polynomial are continuous functions of the coefficients [22,p.281], the problem outlined above may be solved by using a locally convergent algorithm for calculating the remaining zeros. That is, we can use $\{z_2^{(i)}\}_0$ as an estimate $\{\hat{z}_2^{(i)}\}_1$ of the true zero-set $\{z_2^{(i)}\}_1$ corresponding to $\omega_1+\delta\omega_1$, and refine the estimate to within a specified accuracy of the true zeros. Continuity in the zero surface guarantees that this is possible provided that the increment $\delta\omega_1$ around the unit circle is sufficiently small. This process is repeated until the sequence $\{\}_0,\{\}_1,\ldots,\{\}_r,\ldots,\{\}_0$ completely maps out M contours in the z_2-plane.

A simple and fast locally convergent zero-finder is the Newton-Raphson method [18,p.254], at the heart of which are the relations

$$z_2^{(i)} = \hat{z}_2^{(i)} - \frac{G(\omega_1,\hat{z}_2^{(i)})}{G'(\omega_1,\hat{z}_2^{(i)})} \qquad (4\text{-}30a)$$

$$\hat{z}_2^{(i)} \leftarrow z_2^{(i)} \qquad (4\text{-}30b)$$

where the derivative $G'(\omega_1,z_2^{(i)})$ is taken with respect to $z_2^{(i)}$. Equation (4.30) is iterated until the relative error

$$e_2^{(i)} = \frac{z_2^{(i)} - \hat{z}_2^{(i)}}{z_2^{(i)}}$$

falls below a specified tolerance. Providing the estimate $\hat{z}_2^{(i)}$ is "close enough" to the true zero location, eqn. (4.30) converges quadratically on the correct solution. In general, $e_2^{(i)}$ is complex, so it makes sense to require that $e_2^{(i)}$ lies within the circle $|e_2^{(i)}| < \varepsilon$ (Figure 4.6a). However, as calculation of the modulus $|e_2^{(i)}|$ is time consuming, (2 multiplcations, 1 addition and 1 square root) a slightly modified criterion for halting the iterations may be adopted. A technique which works well in practice is to stop when the relative error lies within the square (see Figure 4.6b):

$$|\mathrm{Re}\, e_2^{(i)}| < \varepsilon \quad \cap \quad |\mathrm{Im}\, e_2^{(i)}| < \varepsilon$$

It should be clear that, given a sufficiently close estimate of the true zero location, the major computational bottleneck is the evaluation of eqn. (4.30a). It is therefore desirable to have available a method for efficient calculation of $G'(\omega_1,z_2)$ and $G(\omega_1,z_2)$. Brute-force evaluation of $G(\omega_1,z_2)$ requires 2M-1 complex multiplications and M complex additions, and 2M-2 complex multiplications and M-1 complex additions for $G'(\omega_1,z_2)$. A much more efficient algorithm, due to Horner [22,p.433], reduces the computational requirements to M complex multiplcations and M complex additions. To see how Horner's scheme works, rewrite $G(\omega_1,z_2)$ in nested form

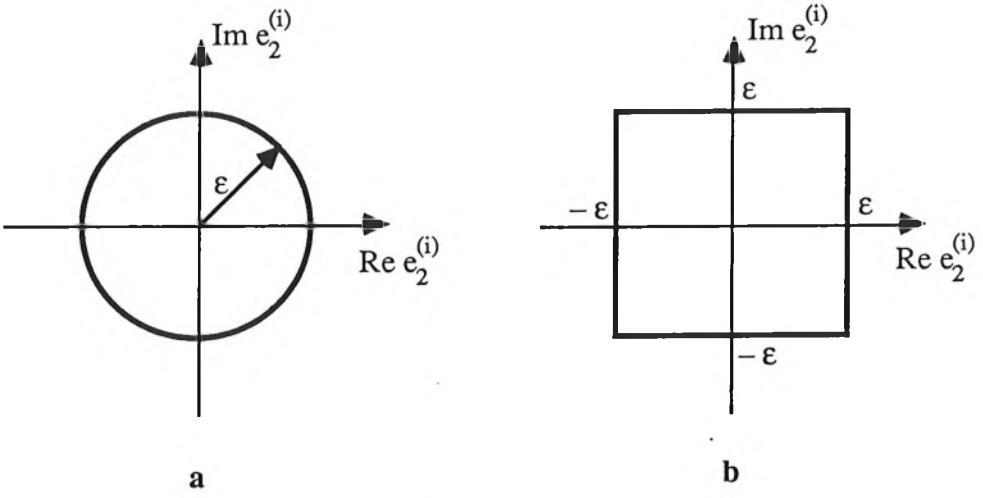


Figure 4.6: a *Complex tolerance circle for* $e_2^{(i)}$.

b *Complex tolerance square for* $e_2^{(i)}$.

$$G(\omega_1,z_2) = c_0(\omega_1)+z_2[c_1(\omega_1)+z_2[c_2(\omega_1)+z_2[\ldots z_2[c_{M-1}(\omega_1)+z_2c_M(\omega_1)]]\ldots] \quad (4\text{-}31)$$

This representation can be realized using the following recurrence relationships

$$\alpha_0 = c_M(\omega_1)$$

$$\alpha_1 = z_2\alpha_0 + c_{M-1}(\omega_1)$$

$$\alpha_2 = z_2\alpha_1 + c_{M-2}(\omega_1) \ \ldots$$

$$\alpha_k = z_2\alpha_{k-1} + c_{M-k}(\omega_1) \quad (4\text{-}32)$$

and obviously

$$\alpha_M = G(\omega_1,z_2)$$

In order to evaluate $G'(\omega_1,z_2)$ using Horner's scheme, we operate on each α_k in a similar manner to that given in eqn. (4.32), forming the following recurrence equations

$$\beta_0 = \alpha_0$$

$$\beta_1 = z_2\beta_0 + \alpha_1$$

$$\beta_2 = z_2\beta_1 + \alpha_2 \ \ldots$$

$$\beta_k = z_2\beta_{k-1} + \alpha_k \quad (4\text{-}33)$$

With a bit of effort, it is possible to see that

$$\beta_{M-1} = G'(\omega_1,z_2)$$

The recurrence formulae (4.32) and (4.33) may be evaluated concurrently, as shown in Algorithm 1. When the "**for**" loop terminates, after 2M complex multiplications and 2M complex additions, α contains $G(\omega_1,z_2)$ and β contains $G'(\omega_1,z_2)$.

$\alpha = c_M(\omega_1)$

$\beta = 0$

For k = M-1 **Down To** 0 **Do**

Begin

$\beta = \beta.z_2 + \alpha$

$\alpha = \alpha.z_2 + c_k(\omega_1)$

End

Algorithm 1: *Horner's method for evaluating $G(\omega_1,z_2)$ and $G'(\omega_1,z_2)$.*

Step 3: Algorithm Termination

Examination of eqn. (4.28) shows that the coefficients $c_m(\omega_1)$ are periodic in ω_1. Thus the zero contours of $G(\omega_1,z_2)$ must eventually close on themselves as ω_1 is incremented indefinitely. A straightforward way of detecting this event is by recording the zero-spacing distance at the beginning of the algorithm. That is, if $z_2^{(i)}{}_{(0)}$ is an initial zero and $z_2^{(i)}{}_{(1)}$ is the same zero after ω_1 has been incremented for the first time, then we choose $s^{(i)} > | z_2^{(i)}{}_{(1)} - z_2^{(i)}{}_{(0)} |$, i=1,...,M. As each zero-set $\{z_2^{(i)}\}_r$ is determined, we check if $| z_2^{(i)}{}_{(r)} - z_2^{(i)}{}_{(0)} | < s^{(i)}$ for i=1,...,M. If these conditions are satisfied, then the contours are closed.

A note regarding the Z-Transform definition

Several researchers define the 2-D Z-Transform in negative powers of **z**, e.g. [1,p.174][25]. In this case the denominator polynomial takes the form

$$W(z_1,z_2) = \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} w(k_1,k_2)\, z_1^{-k_1} z_2^{-k_2}$$

and if $w(k_1,k_2)=g(k_1,k_2)$, the rootmap of $W(z_1,z_2)$ is clearly the inverse of eqn. (4.23). Rather than

explicitly computing this inverse, $W(z_1,z_2)$ may be written as

$$W(z_1,z_2) = z_1^{-M_1} z_2^{-M_2} \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} w(M_1-k_1,M_2-k_2)\, z_1^{k_1} z_2^{k_2}$$

which is of the same form as eqn. (4.20). Hence we may use the preceeding method to compute the rootmap of $W(z_1,z_2)$, provided we first permute the coefficients according to the relation $g(k_1,k_2)=w(M_1-k_1,M_2-k_2)$.

4.5.2 Failure of the Basic Algorithm

Unless the bidisc increment, $\delta\omega_1$, is very small, it often happens that the algorithm will fail in the sense that contours appear to never close on themselves, but instead exhibit severe discontinuities. The reason is that some polynomials are ill-conditioned; that is, a small change in ω_1 (and hence $c_m(\omega_1)$) can cause quite large variations in the zero-set [23]. The failure mechanism is illustrated below in Figure 4.7.

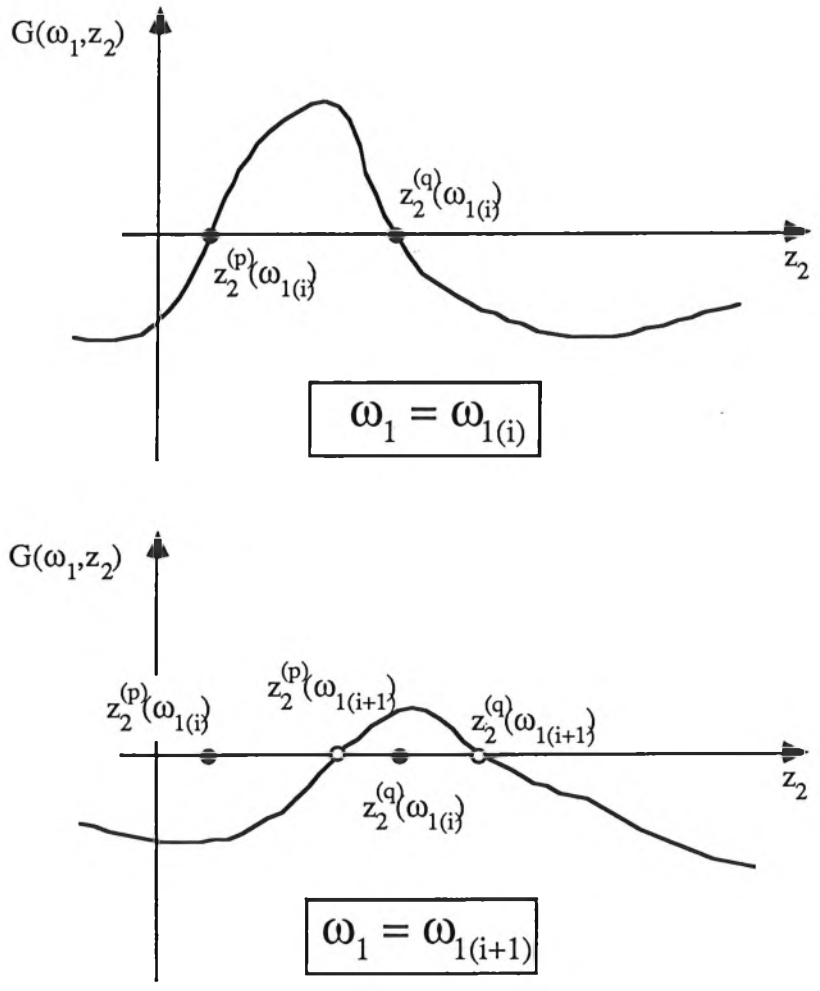


Figure 4.7: *Pertaining to the failure of the basic zero-tracking algorithm.*

For $\omega_1 = \omega_{1(i)}$, two zeros are located: $z_2^{(p)}$ and $z_2^{(q)}$. We then increment $\omega_{1(i)}$ to get $\omega_{1(i+1)}$. According to our method, we use $z_2^{(p)}(\omega_{1(i)})$ as an estimate of $z_2^{(p)}(\omega_{1(i+1)})$, and $z_2^{(q)}(\omega_{1(i)})$ as an estimate of $z_2^{(q)}(\omega_{1(i+1)})$. However, since $z_2^{(q)}(\omega_{1(i)})$ is closer to $z_2^{(p)}(\omega_{1(i+1)})$ than $z_2^{(q)}(\omega_{1(i+1)})$,the algorithm will erroneously "lock-on" and track $z_2^{(p)}$ twice, rather than follow the distinct zeros $z_2^{(p)}$ and $z_2^{(q)}$. For example, Figure 4.8a shows the effect of an erroneous lock-on. By decreasing $\delta\omega_1$ from 0.141 to 0.025, the algorithm terminates successfully, as shown in Figure 4.8b.

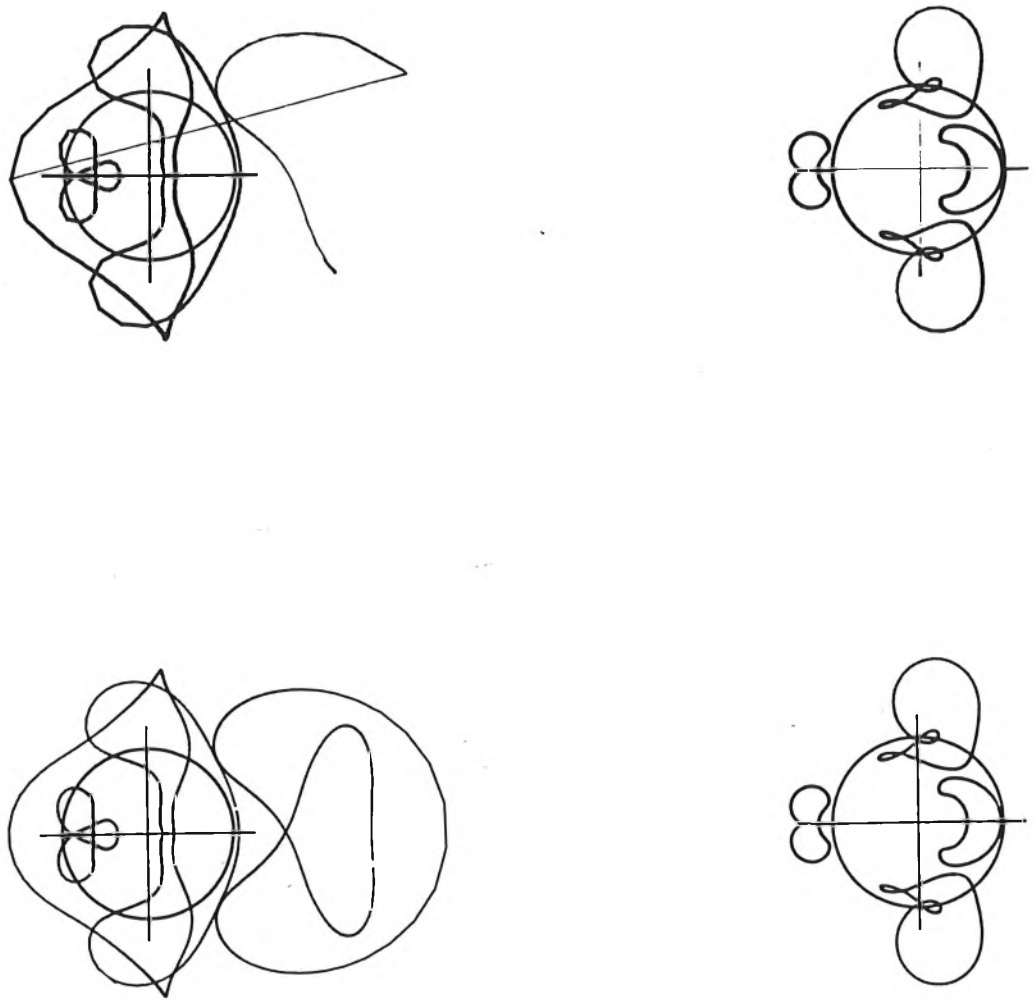


Figure 4.8: top *Zero tracking failure for $\delta\omega = 0.141$*

bot *Zero tracking success for $\delta\omega = 0.025$*

4.5.3 Zero Prediction

We saw in the previous section that ill-conditioned polynomials (i.e. large values of $\partial z_2^{(i)}/\partial c_m(\omega_1)$) have variations in zero location which are too large to track successfully with the basic Newton-Raphson method. An estimate of the likely location reduces the the chance of this happening, without having to greatly reduce the sample interval $\delta\omega_1$.

4.5.3.1 Taylor Series Method

The change in the zero location as a function of the coefficients $c_k(\omega_1)$ can be expanded as a Taylor series

$$\delta z_2^{(i)} = \sum_{k=0}^{M} \frac{\partial z_2^{(i)}}{\partial c_k(\omega_1)} \delta c_k(\omega_1) + \frac{1}{2!} \sum_{k=0}^{M} \frac{\partial^2 z_2^{(i)}}{\partial c_k^2(\omega_1)} \delta c_k^2(\omega_1) + \sum_{\substack{p=0 \\ p\neq q}}^{M} \sum_{q=0}^{M} \frac{\partial^2 z_2^{(i)}}{\partial c_p(\omega_1)\partial c_q(\omega_1)} \delta c_p(\omega_1)\delta c_q(\omega_1) + \cdots \tag{4-34}$$

where $\delta c_k(\omega_1) = c_k(\omega_1+\delta\omega_1) - c_k(\omega_1)$, and the infinite series is truncated to obtain a useful approximation for each $\delta z_2^{(i)}$. The partial derivatives in eqn. (4.34) may be calculated from eqn. (4.27) as

$$\frac{\partial z_2^{(i)}}{\partial c_m(\omega_1)} = \frac{-[z_2^{(i)}]^m}{G'(\omega_1, z_2^{(i)})} \tag{4-35}$$

$$\frac{\partial^2 z_2^{(i)}}{\partial c_m^2(\omega_1)} = \frac{[z_2^{(i)}]^{2m-1}}{[G'(\omega_1, z_2^{(i)})]^3} \left[mG'(\omega_1, z_2^{(i)}) - z_2^{(i)} G''(\omega_1, z_2^{(i)}) \right] \tag{4-36}$$

$$\frac{\partial^2 z_2^{(i)}}{\partial c_p(\omega_1)\partial c_q(\omega_1)} = \frac{[z_2^{(i)}]^{p+q-1}}{[G'(\omega_1, z_2^{(i)})]^3} \left[pG'(\omega_1, z_2^{(i)}) - z_2^{(i)} G''(\omega_1, z_2^{(i)}) \right] \tag{4-37}$$

The desired accuracy of $\delta z_2^{(i)}$ depends on the number of terms used in eqn. (4.34). For example, with the first two terms of eqn. (4.34), and using eqns. (4.35) and (4.36), we have

$$\delta z_2^{(i)} \cong \frac{-1}{G'(\omega_1, z_2^{(i)})} \sum_{k=0}^{M} \delta c_k(\omega_1) [z_2^{(i)}]^k - \frac{G''(\omega_1, z_2^{(i)})}{2[G'(\omega_1, z_2^{(i)})]^3} \sum_{k=0}^{M} \delta c_k^2(\omega_1) [z_2^{(i)}]^{2k} +$$

$$\frac{z_2^{(i)}}{2[G'(\omega_1,z_2^{(i)})]^2}\sum_{k=1}^{M}\delta c_k^2(\omega_1)k\,[z_2^{(i)}]^{2(k-1)} \qquad (4\text{-}38)$$

The predicted zero location is then $\hat{z}_2^{(i)} = z_2^{(i)} + \delta z_2^{(i)}$. Note that $G'(\omega_1,z_2^{(i)})$ is available at the last iteration of eqn. (4.30) for calculating $z_2^{(i)}$, and $G''(\omega_1,z_2^{(i)})$ is computed concurrently by making a simple modification to Algorithm 1. The first summation in eqn. (4.38) is evaluated using Horner's scheme since it is simply a polynomial in $z_2^{(i)}$ with coefficients $\delta c_k(\omega_1)$. Similarly, the second and third summations may be computed concurrently since they are, respectively, a polynomial and its derivative, in $[z_2^{(i)}]^2$ with coefficients $\delta c_k^2(\omega_1)$.

4.5.3.2 Polynomial fits to Re z_i and Im z_i

Prediction of the likely position of the next zero via a Taylor series expansion involves a fair amount of calculation. As the zeros weave their way around each z-plane, it seems reasonable, over a sufficiently small interval, to be able to fit a low-order polynomial through previous zero locations. Thus we are still relying on zero-surface continuity, but we base our prediction on previous zero locations rather than changes in the polynomial coefficients. Two prediction methods have been examined: linear and quadratic polynomials. In order to simplify the discussion, the following symbols are defined:

$z \equiv$ either z variable z_1 or z_2

$z_0 \equiv z(\omega)$ for a given value of ω

$z_1 \equiv z(\omega+\delta\omega)$

$z_n \equiv z(\omega+n\delta\omega)$, n an integer

$x \equiv$ Re z, $y \equiv$ Im z

Linear estimate of the next zero

A linear estimate of the next zero requires the 2 previous zero locations, as illustrated in Figure 4.9. Given (x_0,y_0) and (x_1,y_1) it is easy to show that

$$a = \frac{y_1-y_0}{x_1-x_0} \text{ and } b = y_1-ax_1$$

The problem is that we do not know the value of x_2 with which to predict y_2 (or vice versa). If $|a| \leq 1$ it makes sense to put $x_2 = x_1 + \Delta x$ and calculate y_2. If $|a| > 1$ then choose $y_2 = y_1 + \Delta y$ and calculate x_2. Such a scheme will tend to minimise the error between the estimate and the true zero location.

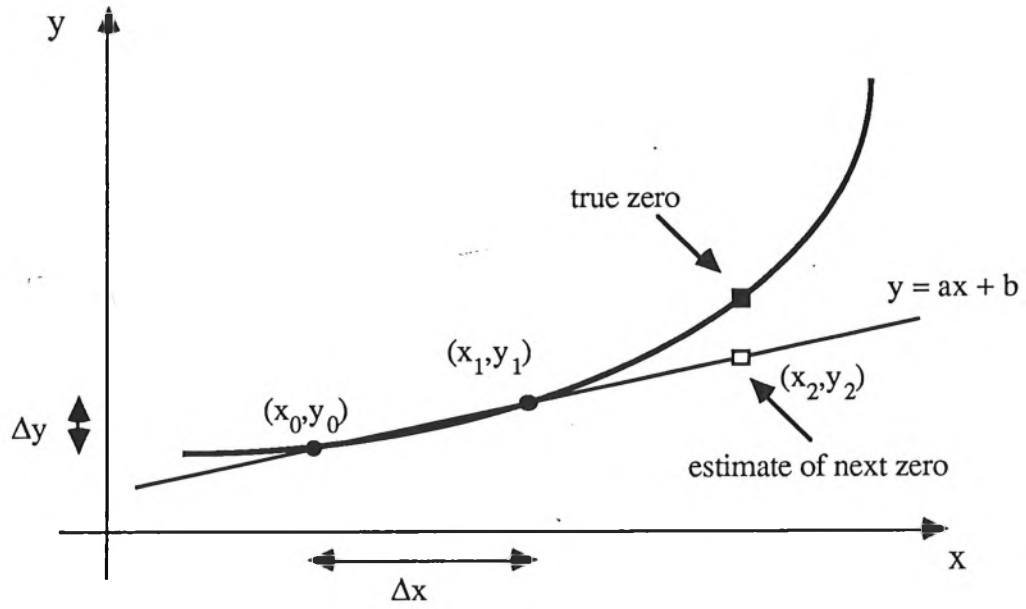


Figure 4.9: *Linear estimate of the next rootmap zero.*

Quadratic estimate of the next zero

A quadratic estimate of the next zero requires the 3 previous zero locations: z_0, z_1 and z_2, as shown in Figure 4.10. In order to simplify calculation of a,b and c (by solving a 2x2 rather than a 3x3 system of equations), we translate the co-ordinates (x_0,y_0), (x_1,y_1) and (x_2,y_2) via $u = x - x_0$ and $v = y - y_0$. See Figure 4.11. Obviously $v = au^2 + bu$, from which

$$a = \left[\frac{v_2}{u_2} - \frac{v_1}{u_1}\right]\frac{1}{u_2 - u_1} \qquad \text{and} \qquad b = \frac{v_1}{u_1} - a \cdot u_1$$

(Note that the *a* and *b* in Figure 4.11 are different to those in Figure 4.10.) If u_1 and u_2 are close to each other (or close to zero), then the modulus of 'a' will be very large, which may cause overestimates of (x_3,y_3). The problem and its solution are shown in Figure 4.12. In this case, choosing $u = a'v^2 + b'v$ rather than $v = au^2 + bu$ gets us closer to the correct zero. In general, if $|a| \leq |a'|$ then put $u_3 = u_2 + \Delta u$ and calculate v_3. If $|a| > |a'|$ then put $v_3 = v_2 + \Delta v$ and calculate u_3.

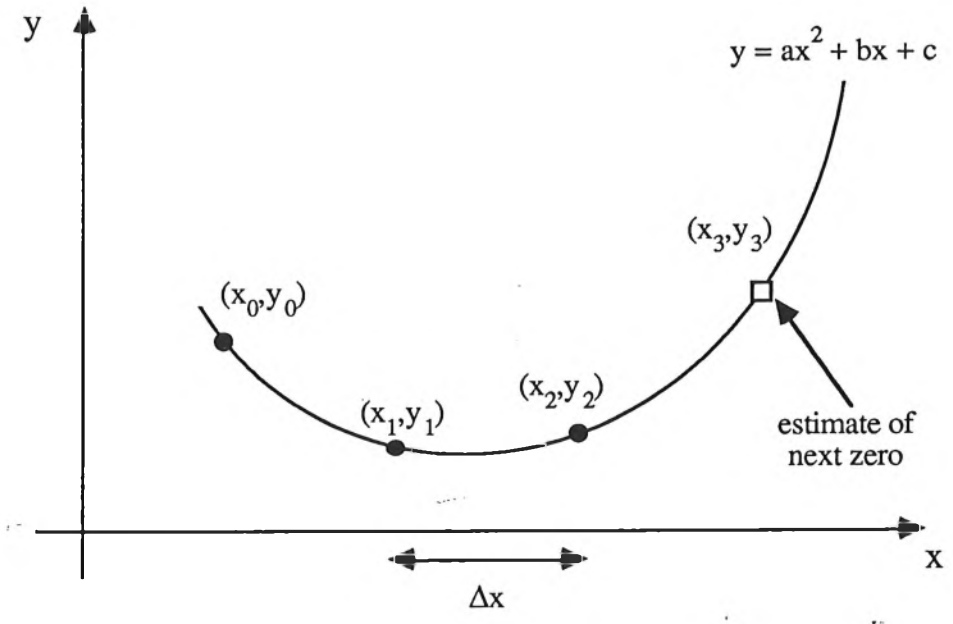


Figure 4.10: *Quadratic estimate of the next rootmap zero.*

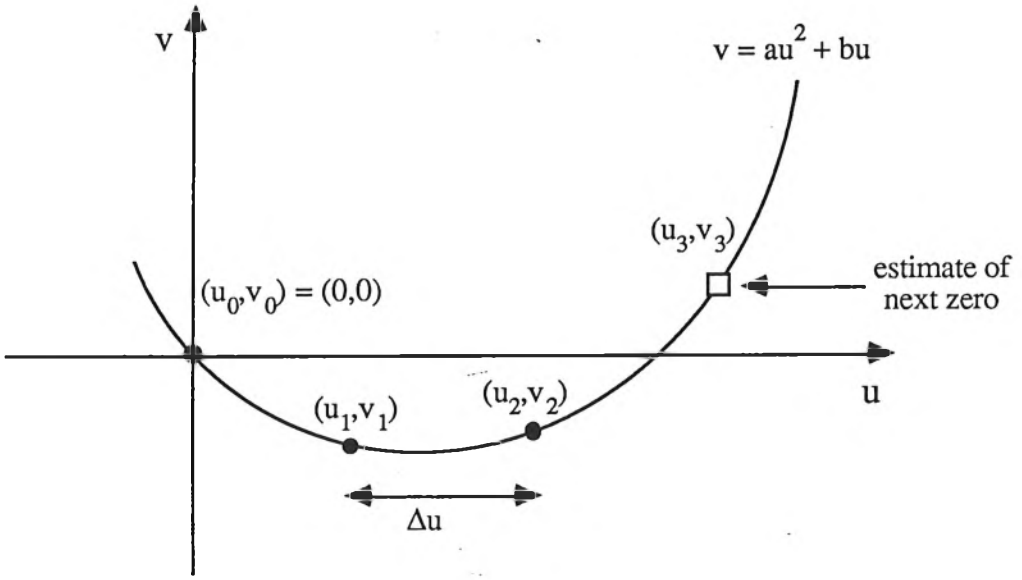


Figure 4.11: *Translated co-ordinate system for the quadratic zero prediction method.*

4.5.3.3 Comparison of the Taylor, Linear and Quadratic Methods

Raw speed trials of rootmap calculation, based on linear, quadratic and Taylor series zero prediction, are shown in Figures 4.13 and 4.14. As might be expected, the execution time increases linearly as the number of bidisc samples increases. (Not shown by these figures is the fact that for a fixed bidisc sampling rate, the execution time increases with filter order). Furthermore, the execution time decreases if a zero predictor is used. The linear and quadratic methods are approximately equal in speed and both are faster than the Taylor series method. All methods are faster than no zero prediction at all.

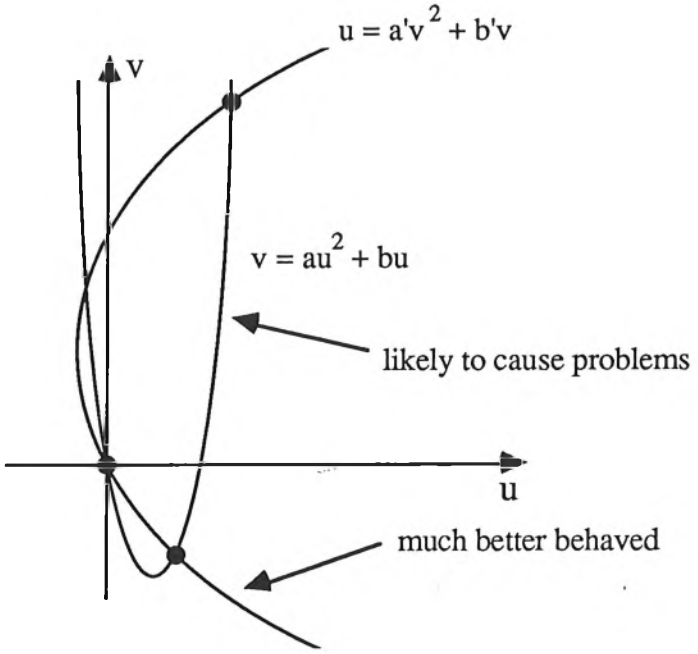


Figure 4.12: *Over-estimate of the quadratically predicted zero position.*

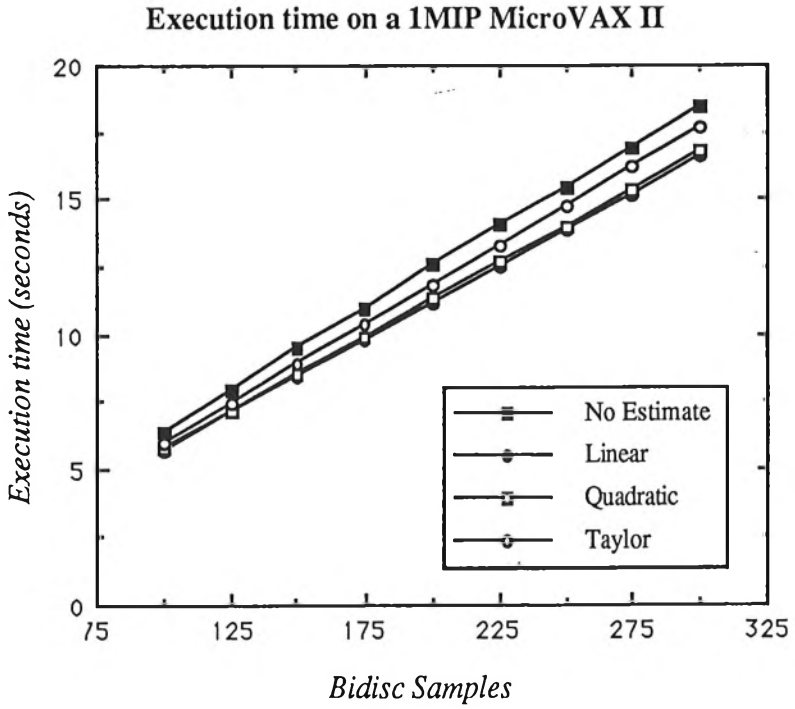


Figure 4.13: *Execution time versus bidisc sampling rate for a 3 by 3 mask.*

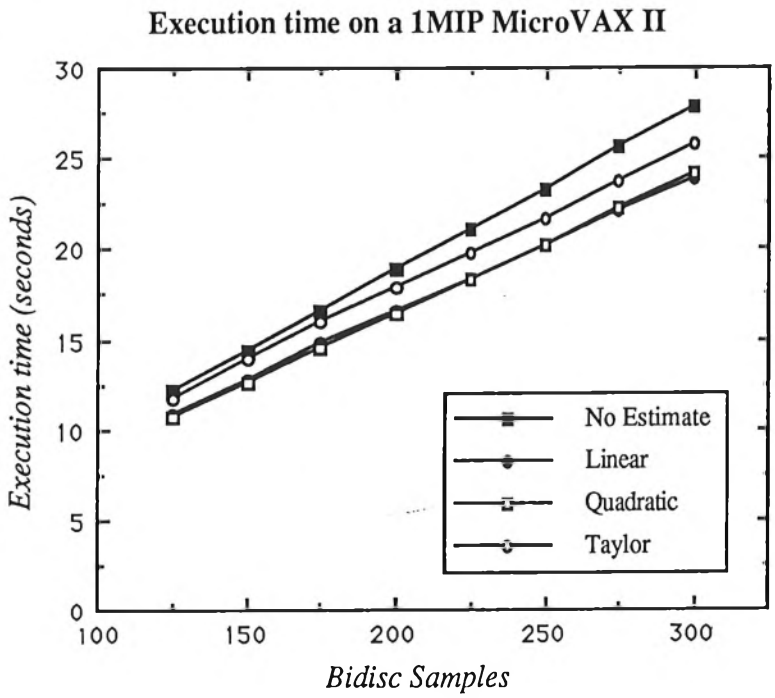


Figure 4.14: *Execution time versus bidisc sampling rate for a 5 by 5 mask.*

Despite the fact that zero prediction involves extra calculation, execution speed increases because less time is spent iterating eqn. (4.30). Figure 4.15 shows a histogram illustrating how this happens. *Frequency* refers to the average number of times eqn. (4.30) is iterated per bidisc sample. For example, referring to the top graph in Figure 4.15, we see that with no zero estimate eqn. (4.30) requires 4 iterations most of the time. With linear zero prediction (or quadratic or Taylor prediction), only 3 iterations are required. In general, with any form of zero prediction, the iteration count reduces from between 4 and 6, to between 3 (mainly) and 4, and even down to 2 in some cases. As expected, a high bidisc sampling rate results in fewer iterations 'across the board' since the previous zero will be much closer to the next zero. Figures 4.16 and 4.17 show a similar trend as the mask size increases from 5 by 5 to 7 by 7.

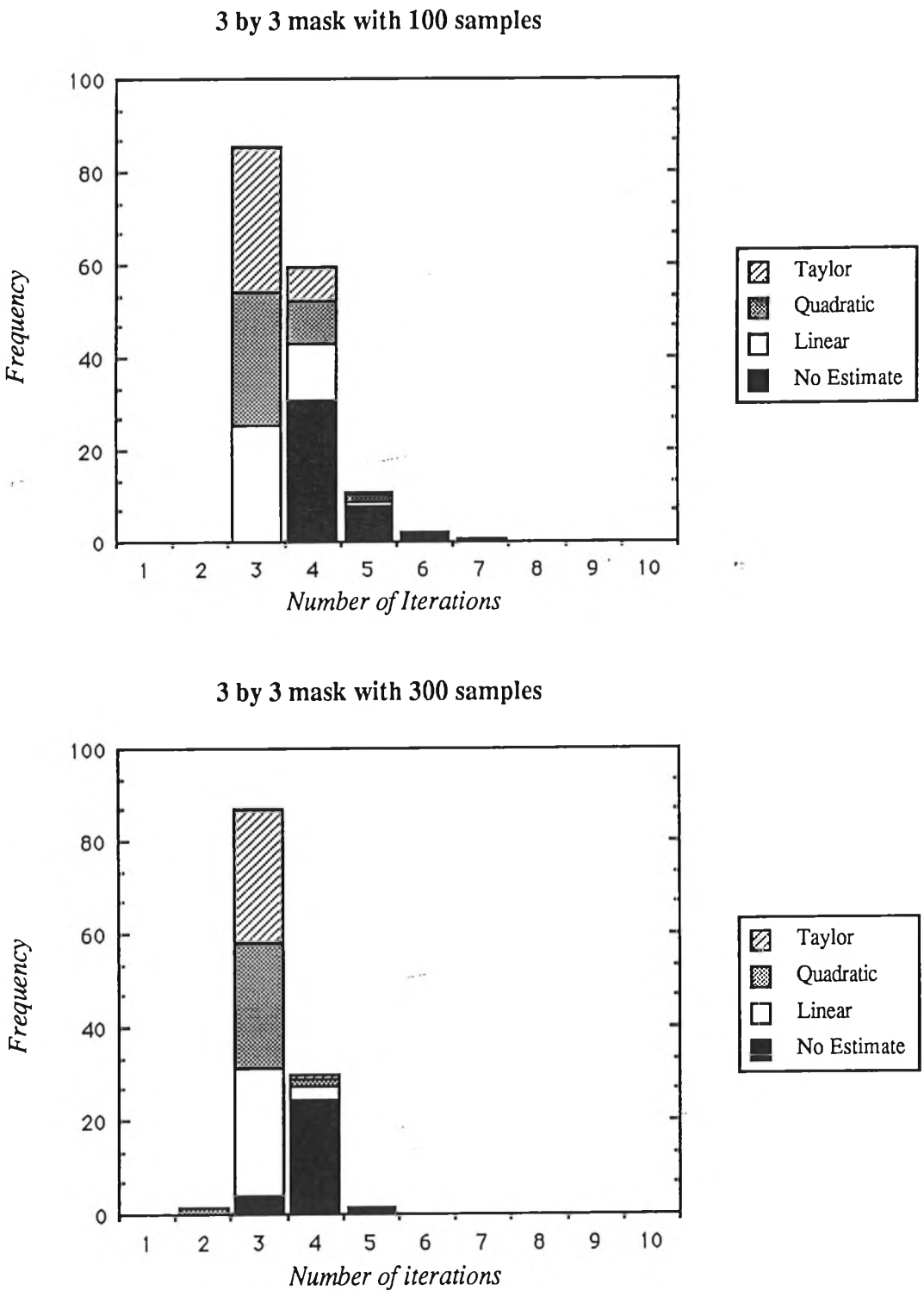


Figure 4.15: *Illustrating Newton-Raphson iteration-count reduction for the 3 by 3 mask.*
top *100 bidisc samples*
bot *300 bidisc samples*

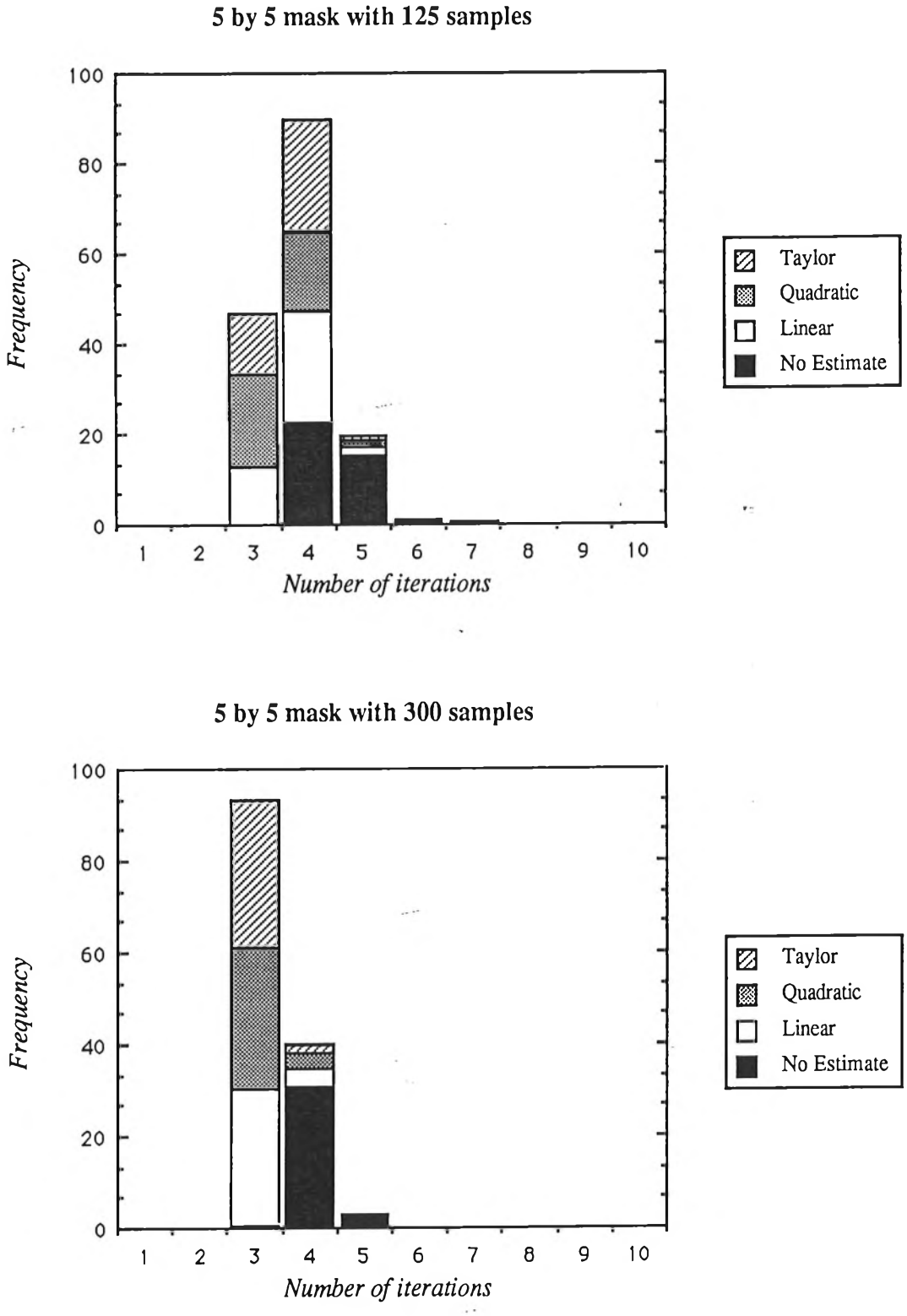


Figure 4.16: *Illustrating Newton-Raphson iteration-count reduction for the 5 by 5 mask.*
top *125 bidisc samples*
bot *300 bidisc samples*

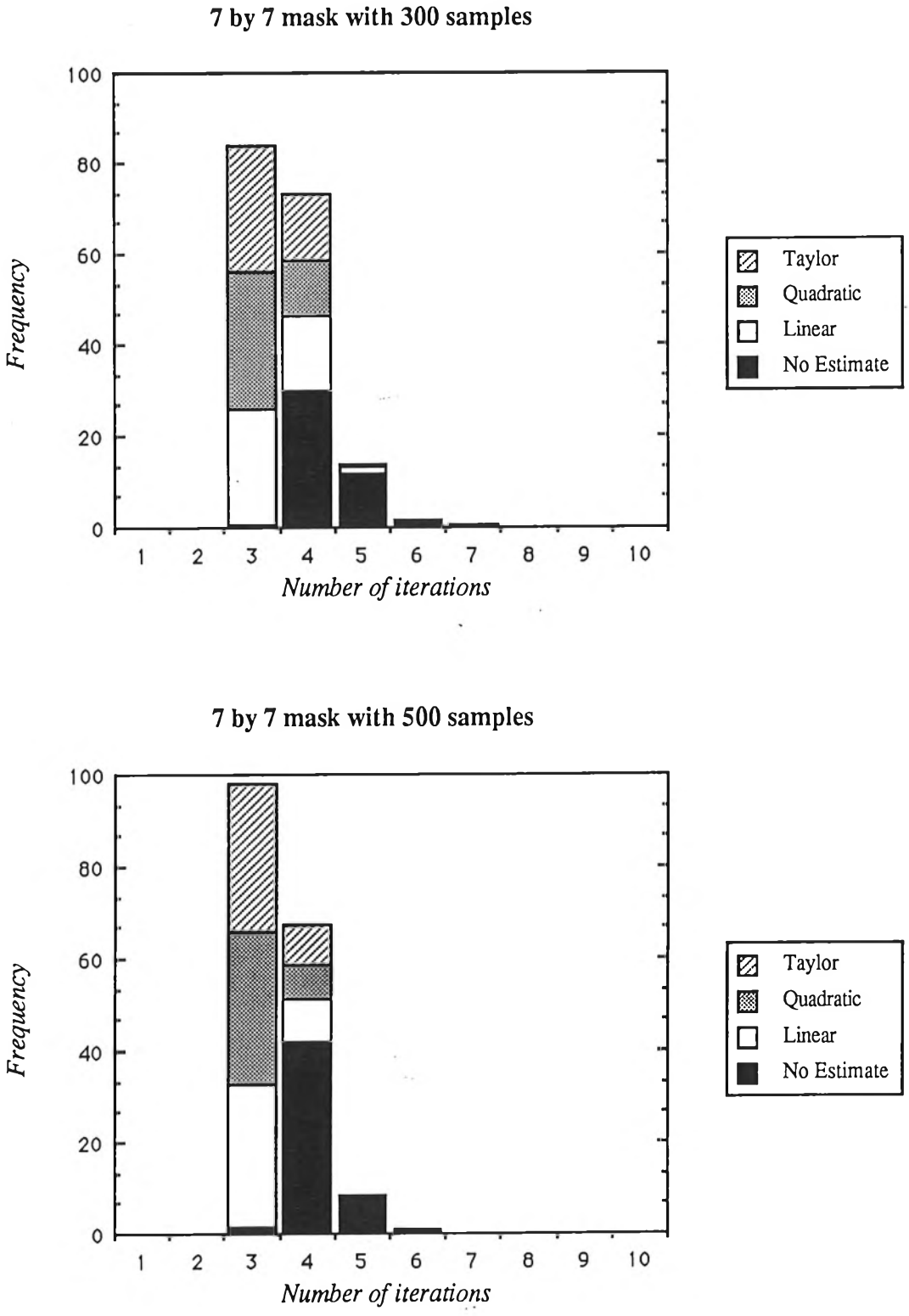


Figure 4.17: *Illustrating Newton-Raphson iteration-count reduction for the 7 by 7 mask.*
top *300 bidisc samples*
bot *500 bidisc samples*

4.6 The Stability Test of Maragos, Schafer and Mersereau [25]

In a recent paper, Maragos *et al* [25], propose a very simple test used in conjunction with a 2-D linear predictive image coding scheme. This writer maintains that the proposed stability criterion is impractical in a realistic situation.

Theory

Define the prediction filter as

$$P(z_1,z_2) = \sum_{\substack{k \\ (k,l)\in\pi}} \sum_{l} a(k,l)\, z_1^{-k}\, z_2^{-l} \tag{4-39}$$

where

$$\pi = \{\ (k,l):\ 0 \le k,l \le Q\ ;\quad (k,l) \neq (0,0)\ \}$$

If the filter

$$H(z_1,z_2) = \frac{1}{1-P(z_1,z_2)}$$

is stable, then

$$P(1,1) = \sum_{\substack{k \\ (k,l)\in\pi}} \sum_{l} a(k,l) \ < \ 1 \tag{4-40}$$

(Note that $P(z_1,z_2)$ is defined in inverse powers of z_1 and z_2). Equation (4.40) is the stability test proposed by Maragos *et al.*

We will now analyse the limitations of Maragos' method. Define

$$B(z_1,z_2) = 1 - P(z_1,z_2) \ = \sum_{\substack{k \\ (k,l)\in R}} \sum_{l} b(k,l)\, z_1^{-k}\, z_2^{-l} \tag{4-41}$$

where

$$R = \pi \cup (0,0)$$

and $b(k,l) = -a(k,l)$ if $(k,l) \in \pi$ and $b(0,0) = 1$, then the condition (4.40) is equivalently expressed as

$$B(1,1) = \sum_{\substack{k \\ (k,l)\in R}} \sum_{l} b(k,l) \ < \ 0 \tag{4-42}$$

We now write $B(z_1,z_2)$ as

$$B(z_1,z_2) \ = \ b(0,0) \prod_{k=1}^{Q} (1-\zeta_k(z_1).z_2^{-1}) \tag{4-43}$$

where we are treating z_1 as a parameter. At the point $(z_1,z_2) = (1,1)$ we have

$$B(1,1) = b(0,0)\prod_{k=1}^{Q}(1-\zeta_k(1)) \tag{4-44}$$

and, noting that b(0,0) = 1,

$$B(1,1) < 0 \text{ if and only if } \prod_{k=1}^{Q}(1-\zeta_k(1)) < 0.$$

$\zeta_k(1)$ may be real or complex, but since B(1,1) is real (for real b(k,l)), any complex $\zeta_k(1)$ must occur in conjugate pairs. Thus eqn. (4.44) may be expressed as

$$B(1,1) = b(0,0)\prod_{k=1}^{Q-2M}(1-\zeta_k(1))\prod_{m=1}^{M}(1-\zeta_m(1))(1-\zeta_m^*(1))$$

or

$$B(1,1) = b(0,0)\prod_{k=1}^{Q-2M}(1-\zeta_k(1))\prod_{m=1}^{M}(1-2\mathrm{Re}\,\zeta_m(1) + |\zeta_m(1)|^2) \tag{4-45}$$

where we assume that there are 2M complex $\zeta_k(1)$ and Q-2M real $\zeta_k(1)$. It is easy to show that

$$1 - 2\mathrm{Re}\,\zeta_m(1) + |\zeta_m(1)|^2 \geq 0 \qquad \forall\, \zeta_m(1)$$

hence **B(1,1) can only be negative if an *odd number* of *real* $\zeta_k(1)$ intersect the real axis for $\mathrm{Re}(z_2) > 1$.** A similar statement can be made about the z_1 plane by modifying eqn. (4.43) so that z_2 is treated as a parameter.

Effectiveness of Maragos' Test

In order to gauge the effectiveness of Maragos' test, it is necessary to have a very large number of recursive filters available. As it is impractical to have such a number, (mainly because of the extremely long design times required for 'real' filters), a simple compromise is to generate the filter coefficients randomly. To this end, Figure 4.18 shows the success rate (i.e. percentage of detected unstable filters versus truly unstable filters), as a function of coefficient dynamic range and filter order. For each filter order and dynamic range, 100 filters were randomly generated, but with b(0,0) fixed at 1.0. The general trend is clear: for filters of any order, those with coefficients occupying a large dynamic range are more likely to be detected. This result is intuitively obvious since, with b(0,0) fixed, the zeros of B(**z**) are likely to take on large values, increasing the chance of an odd number of rootmap contours intersecting the real axis for Re $z_1 > 1$ or Re $z_2 > 1$.

Another interesting effect is that for any dynamic range, unstable filters of higher order are more likely to be detected than those of low order. Again this result seems conceptually agreeable since, as the filter order increases, the rootmap contours become wildly convoluted (refer to Figure 4.5), increasing the chance of intersection with the real axis. In any event, Maragos' test does not perform well. For a 3x3 mask with large dynamic range, less than half the unstable filters are detected. Even when the filter order is increased to 7x7, only 56% of unstable filters are successfully identified.

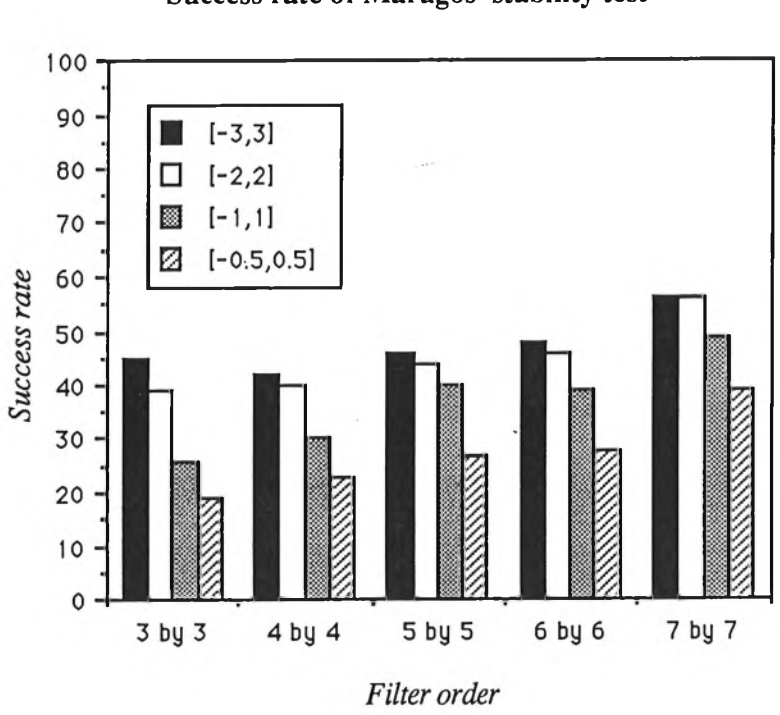


Figure 4.18: *Success rate of Maragos' test as a function of coefficient dynamic range and filter order.*

[True instability was detected by monitoring the (densely sampled) rootmap of each filter. If the modulus of any rootmap contour dropped to (or below) unity, the filter was deemed unstable]

4.7 New Necessary Tests for Stability and a Simple Stabilization Strategy [24]

When designing 2-D recursive digital filters, the design process does not always incorporate stability tests in the design process, or guarantee that a stable filter will be generated. Thus, to be of any use, the filter must be tested for stability prior to the implementation phase, and if unstable, stabilized in some fashion. Unfortunately, stability testing is usually quite involved and time consuming (as evidenced in Section 4.3) and to date, the highest filter order for which a closed-form stability test exists is a 3x3 denominator polynomial [27]. Furthermore, no truly satisfactory stabilization technique has been proposed. This point is particularly important for model-based image coding schemes (such as 2-D linear predictive coding [25][26]) where it is imperative to have real-time testing and stabilization methods available. This section introduces a set of simple necessary tests for stability accompanied by a simple stabilization method suitable for real-time implementaion.

4.7.1 New Necessary Tests

One Dimensional Results

Consider the polynomial

$$D(z) = \sum_{k=0}^{N} a_k z^k \tag{4-46}$$

which can always be factored as

$$D(z) = a_0 \prod_{k=1}^{[N/2]} (q_k z^2 + p_k z + 1) = a_0 \prod_{k=1}^{[N/2]} D_k(z) \tag{4-47}$$

where [N/2] is the smallest integer $\geq N/2$. If N is odd then one q_k will be zero. The filter defined by H(z)=1/D(z) will be stable provided that each subfilter, $D_k(z)$, has no zeros on or within the unit disc. This is assured if, and only if, for every k:

$$|q_k| < 1 \tag{4-48a}$$

$$q_k + p_k + 1 > 0 \quad \text{or} \quad D_k(1) > 0 \tag{4-48b}$$

$$q_k - p_k + 1 > 0 \quad \text{or} \quad D_k(-1) > 0 \tag{4-48c}$$

Using eqn. (4.48) in eqn. (4.47), we find that if H(z) is stable, then

$$|a_0| > |a_N| \tag{4-49a}$$

$$a_0 D(1) > 0 \tag{4-49b}$$

$$a_0 D(-1) > 0 \tag{4-49c}$$

Two Dimensional Results

Given the bivariate polynomial

$$D(z_1,z_2) = \sum_{n=0}^{N} \sum_{m=0}^{M} d(m,n)\, z_1^m z_2^n \tag{4-50}$$

Huang's theorem (Section 4.3) states that $H(z_1,z_2) = 1/D(z_1,z_2)$ represents a stable filter if, and only if

$$D(z_1,0) \neq 0, \quad |z_1| \leq 1 \tag{4-51a}$$

$$D(z_1,z_2) \neq 0, \quad |z_1| \leq 1 \quad \text{and} \quad |z_2| = 1 \tag{4-51b}$$

Note that the role of z_1 and z_2 in eqn. (4.51) may be interchanged, and we do so in the following development.

Using eqns. (4.49a) and (4.51a), we find that if $H(z_1,z_2)$ is stable, then

$$|d(0,0)| > |d(M,0)| \quad \text{and} \quad |d(0,0)| > |d(0,N)| \qquad \text{(test 1)}$$

where to get the second result, z_1 and z_2 have been swapped in eqn. (4.51a). Similarly, using equations (4.49b,c) and (4.51a) we have

$$d(0,0)\sum_{m=0}^{M} d(m,0) > 0 \quad \text{and} \quad d(0,0)\sum_{n=0}^{N} d(0,n) > 0 \qquad \text{(test 2)}$$

and

$$d(0,0)\sum_{n=0}^{N} (-1)^n d(0,n) > 0 \quad \text{and} \quad d(0,0)\sum_{m=0}^{M} (-1)^m d(m,0) > 0 \qquad \text{(test 3)}$$

Choosing $z_2=1$ in eqn. (4.51b), and again swapping z_1 and z_2, application of eqn. (4.49a) yields

$$\left|\sum_{n=0}^{N} d(0,n)\right| > \left|\sum_{n=0}^{N} d(M,n)\right| \text{ and } \left|\sum_{m=0}^{M} d(m,0)\right| > \left|\sum_{m=0}^{M} d(m,N)\right| \qquad \text{(test 4)}$$

and now putting $z_2=-1$, the same combination of equations gives:

$$\left|\sum_{n=0}^{N} (-1)^n d(0,n)\right| > \left|\sum_{n=0}^{N} (-1)^n d(M,n)\right|$$

$$\text{and} \quad \left|\sum_{m=0}^{M} (-1)^m d(m,0)\right| > \left|\sum_{m=0}^{M} (-1)^m d(m,N)\right| \qquad \text{(test 5)}$$

Finally, choosing $z_2=\pm 1$, application of eqns. (4.49b,c) to eqn. (4.51b) gives the following

$$D(1,1)\sum_{n=0}^{N} d(0,n) > 0 \quad \text{and} \quad D(1,1)\sum_{m=0}^{M} d(m,0) > 0$$

$$D(-1,1)\sum_{n=0}^{N} (-1)^n d(0,n) > 0 \quad \text{and} \quad D(-1,1)\sum_{m=0}^{M} d(m,0) > 0$$

$$D(1,-1)\sum_{n=0}^{N} d(0,n) > 0 \quad \text{and} \quad D(1,-1)\sum_{m=0}^{M} (-1)^m d(m,0) > 0$$

$$D(-1,-1)\sum_{n=0}^{N} (-1)^n d(0,n) > 0 \quad \text{and} \quad D(-1,-1)\sum_{m=0}^{M} (-1)^m d(m,0) > 0$$

When combined with tests 2 and 3, the foregoing inequalities reduce to the simple tests below

$$d(0,0)D(1,1) > 0 \qquad \text{(test 6)}$$

$$d(0,0)D(-1,1) > 0 \qquad \text{(test 7)}$$

$$d(0,0)D(1,-1) > 0 \qquad \text{(test 8)}$$

$$d(0,0)D(-1,-1) > 0 \qquad \text{(test 9)}$$

It is interesting to note that tests 6-9 are simply the inverse of the frequency response of $H(z_1,z_2)$ at

the corners of the first quadrant in the (ω_1,ω_1)-plane. For a bandlimited signal, we often have

$$D(1,1) \leq (D(-1,1) \text{ or } D(1,-1)) \leq D(-1,-1)$$

and thus tests 7,8 and 9 are usually not necessary. Note that the only test proposed by Maragos *et al* is test 6. As a numerical example of the tests described above, consider the (3x3) denominator polynomial $D(z_1,z_2)$ defined by

$$d(m,n) = \begin{bmatrix} 1.000 & -0.628 & -0.036 \\ -0.851 & 0.708 & -0.378 \\ 0.137 & -0.070 & 0.366 \end{bmatrix} \tag{4-52}$$

Applying tests 1-9, we find that while Maragos' test 6 is satisfied, the second part of test 4 fails, indicating an unstable filter as shown in Figure 4.19.

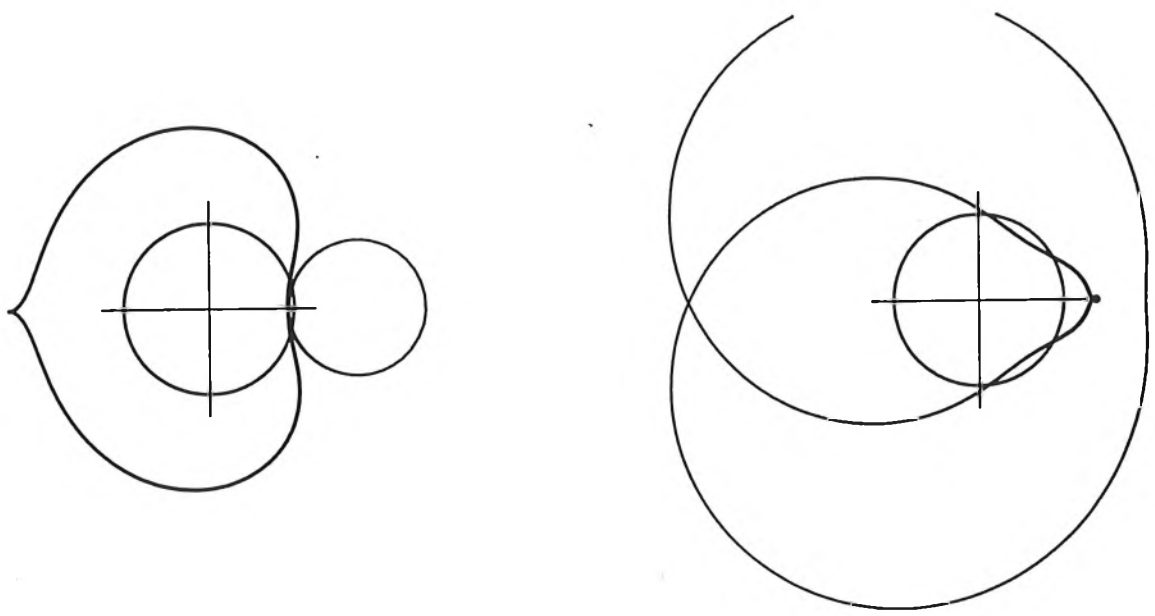


Figure 4.19: *Unstable denominator polynomial given by eqn. (4.52).*

4.7.2 A Stabilization Strategy

For a recursively computable implementation of the filter $1/D(z_1,z_2)$, it is usual to write eqn. (4.50) so that d(0,0)=1. In view of tests 1-9 above, a particularly simple stabilization strategy is to scale the filter coefficients in question until the offending test condition is satisfied. For example, if the first part of test 3 fails, then choose a scale factor ß, such that

$$1 + \beta \sum_{n=1}^{N} (-1)^n \, d(0,n) \; > \; 0$$

Similarly, if the second part of test 4 fails, choose a value for ß which satisfies

$$\left| 1 + \beta \sum_{m=1}^{M} d(m,0) \right| > \left| \beta \sum_{m=0}^{M} d(m,N) \right| \tag{4-53}$$

For example, using the unstable filter given by eqn. (4.52) and result (4.53), we require a value of ß such that ß < 0.911. Applying this factor to the coefficients d(m,n) yields a set of stabilized coefficients

$$d_{stable} = \begin{bmatrix} 1.000 & -0.572 & -0.033 \\ -0.775 & 0.645 & -0.344 \\ 0.125 & -0.064 & 0.333 \end{bmatrix} \tag{4-54}$$

Examination of the rootmap of eqn. (4.54) in Figure 4.20 shows that the filter has been stabilized.

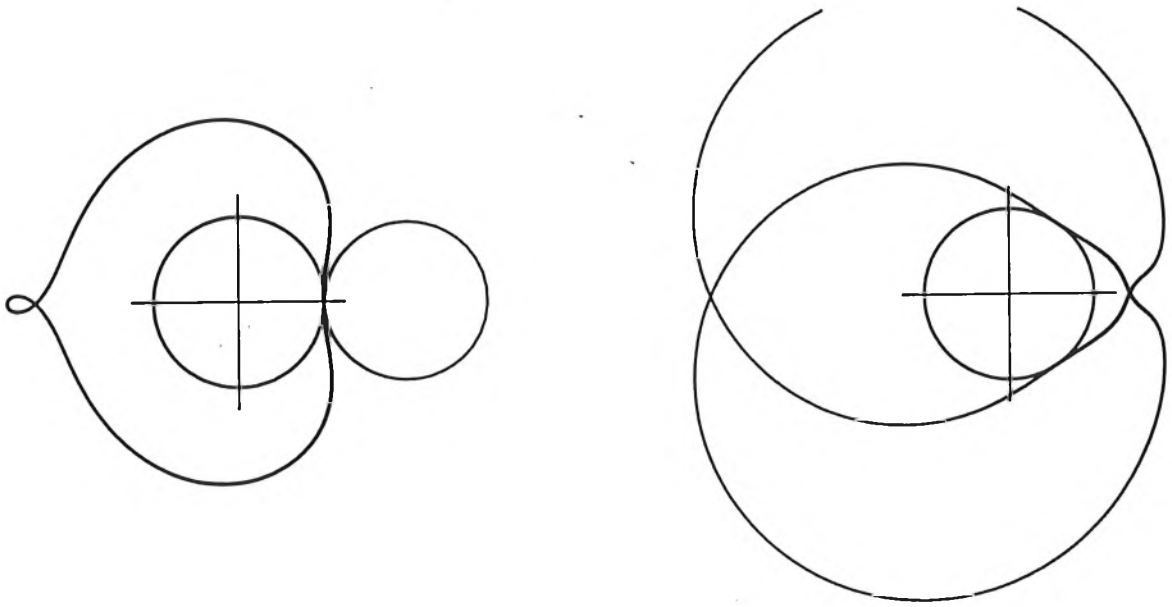


Figure 4.20: *Stabilized denominator polynomial given by eqn. (4.54).*

4.8 Stability Test Performance

In order to measure the effectiveness of the proposed stability tests, we may use the procedure detailed in Section 4.6.1 whereby random filters are generated, of a specified order and prescribed maximum dynamic range. The success rate, as a function of maximum dynamic range and filter order, is shown in Figure 4.21. The salient points to note are: for sufficiently large dynamic range (between [-1,1] and [-2,2]) the detection rate appears to be 100%; as the dynamic range decreases, so does the detection rate; for sufficiently small dynamic range, the detection rate is better for higher order filters.

A breakdown of the performance of each test is shown in Figures 4.22–4.24. The general trend is that for large dynamic ranges, tests 1–5 tend to dominate; that is, they are successful at detecting unstable filters more often than tests 6–9. As the dynamic range decreases, the effectiveness of tests 1–3 drops off rather rapidly, and the success of the suite of tests as a whole, depends on tests 4–9.

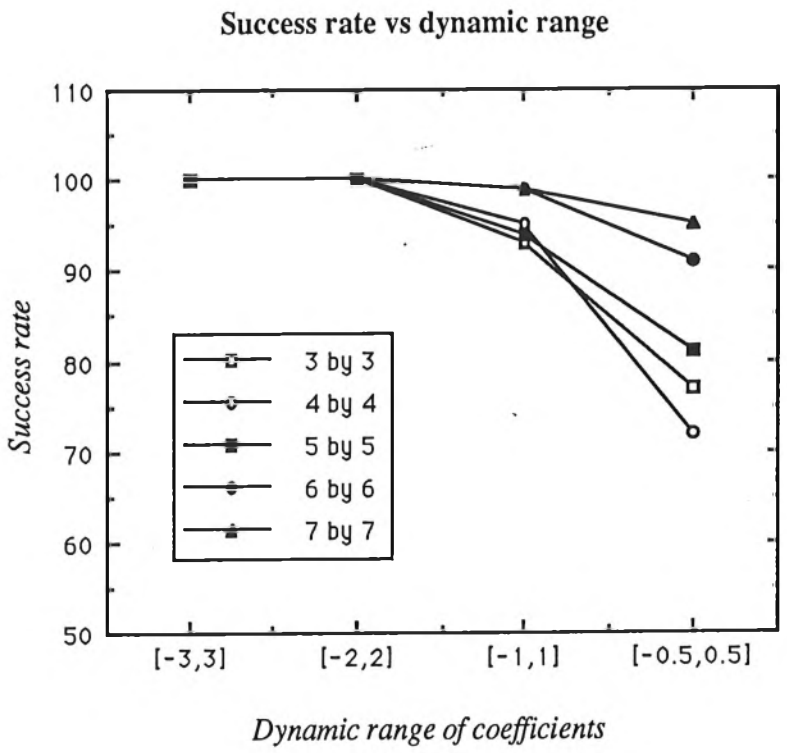


Figure 4.21: *Success rate of the new tests as a function of coefficient dynamic range and filter order.*

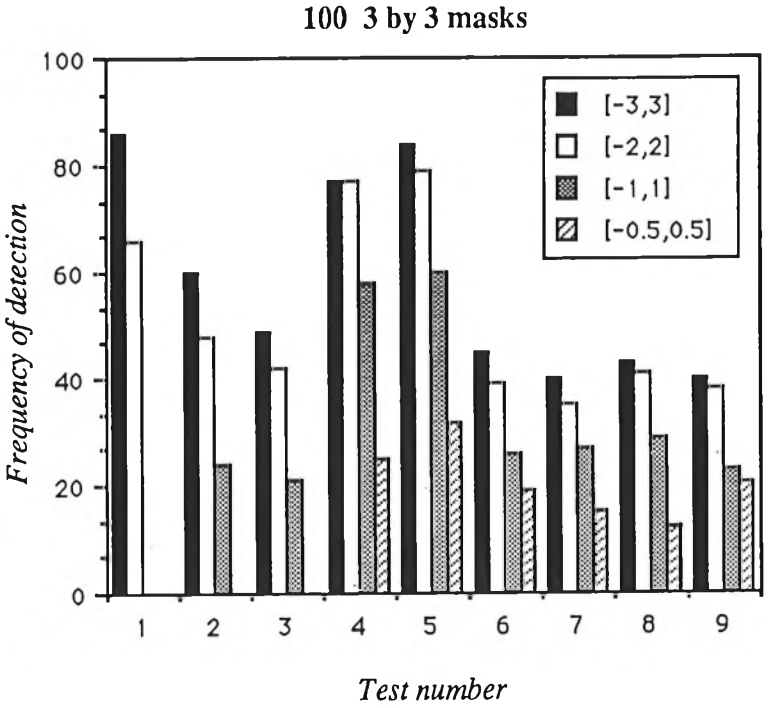


Figure 4.22: *Breakdown of specific test condition violations for 100 randomly generated filters. (3 by 3 mask).*

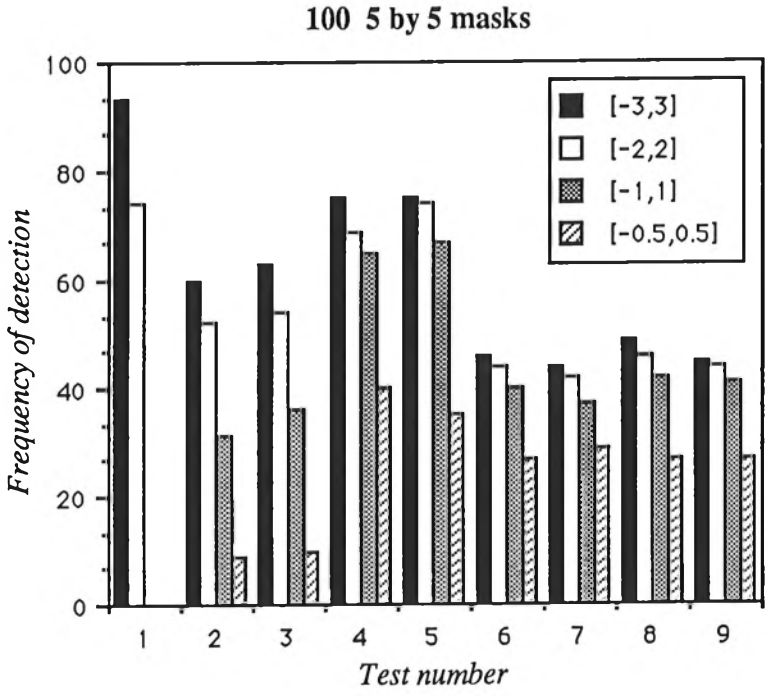


Figure 4.23: *Breakdown of specific test condition violations for 100 randomly generated filters. (5 by 5 mask).*

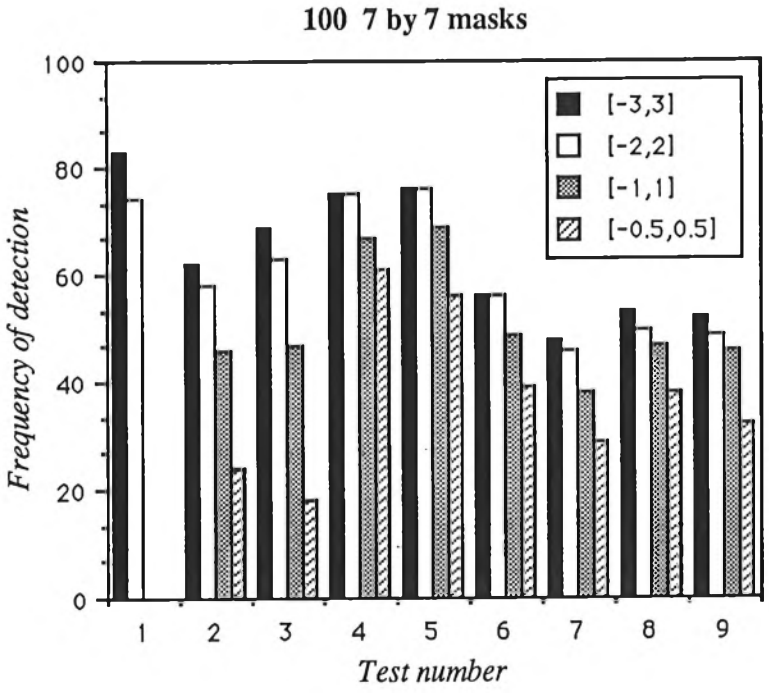


Figure 4.24: *Breakdown of specific test condition violations for 100 randomly generated filters. (7 by 7 mask).*

4.9 References

1 **DE Dudgeon** and **RM Mersereau**: 'Multidimensional digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

2 **R Narasimhan**: 'Several complex variables', The University of Chicago Press, 1971.

3 **W Rudin**: 'Function theory in polydiscs', W. A. Benjamin, Inc., New York, 1969.

4 **JL Shanks, S Treital**, and **JH Justice**: 'Stability and synthesis of two-dimensional recursive filters', *IEEE Trans. Audio and Electroacoustics*, Vol. AU-20, No. 2, June 1972, pp.115-128.

5 **RV Churchill, JW Brown** and **RF Verhey**: 'Complex variables and applications', McGraw-Hill, 3rd Ed., New York, 1974.

6 **MH Hayes** and **JH McClellan**: 'Reducible polynomials in more than one variable', *Proc. IEEE*, Vol. 70, No. 2, 1982, pp.197-198.

7 **WF Osgood**: 'Topics in the theory of several complex variables', Dover Publications, New York, 1966.

8 **D Goodman**: 'Some stability properties of two-dimensional linear shift-invariant digital filters', *IEEE Trans. Circuits and Systems*, Vol. CAS-24, No. 4, April 1977, pp.201-208.

9 **MNS Swamy, L Roytman** and **EI Plotkin**: 'On stability properties of three- and higher dimensional linear shift-invariant filters', *IEEE Trans. Circuits and Systems*, Vol. CAS-32, No. 9, 1985, pp.888-891.

10 **M-H Pee, PP Khargonekar**, and **EB Lee**: 'Further results on possible root locations of 2-D polynomials', *IEEE Trans. Circuits and Systems*, Vol. CAS-33, No. 5, May 1986, pp.566-569.

11 **TS Huang**: 'Stability of two-dimensional recursive filters', *IEEE Trans. Audio and Electroacoustics*, Vol. AU-20, No. 2, June 1972, pp.158-163.

12 **D Goodman**: 'An alternate proof of Huang's stability theorem', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-24, No. 5, 1976, pp.426-427.

13 **DL Davis**: 'A correct proof of Huang's theorem on stability', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-24, No. 5, 1976, pp.425-426.

14 **MG Strintzis**: 'Tests of stability of multidimensional filters', *IEEE Trans. Circuits and Systems*, Vol. CAS-24, No. 8, 1977, pp.432-437.

15 **R DeCarlo, J Murray** and **R Saeks**: 'Multivariate Nyquist theory', *Int. J. Control*, 25, 1977, pp.657-675.

16 **M Andrews**: 'Robust rootmap calculation', *Electronics Letters*, Vol. 24, No. 15, 21 July 1988, pp.968-969.

17 **A Ralston** and **P Rabinowitz**: 'A first course in numerical analysis', 2nd Ed., McGraw-Hill, New York, 1978.

18 **WH Press, BP Flannery, SA Teukolsky** and **WT Vetterling**: 'Numerical recipes: The art of scientific computing', Cambridge University Press, New York, 1986.

19 **MA Jenkins** and **JF Traub**: 'Algorithm 419: Zeros of a complex polynomial [C2]', *Communications of the ACM*, Vol. 15, No. 2, 1972, pp.97-99.

20 **DH Withers**: 'Remark on Algorithm 419 [C2]', *Communications of the ACM*, Vol. 17, No. 3, 1974, p.20.

21 **SL Marple Jr.**: 'Digital spectral analysis with applications', Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

22 **P Henrici**: 'Applied and computational complex analysis. Vol. 1', Wiley, New York, 1974.

23 **JH Wilkinson**: 'The evaluation of the zeros of ill-conditioned polynomials, Part I', *Numerische Mathematik*, Vol. 1, 1959, pp.150-166.

24 **M Andrews** and **DT Nguyen**: 'Necessary conditions for stability and a simple stabilization scheme for two-dimensional digital filters', *Electronics Letters*, Vol. 24, No. 14, 7 July 1988, pp.843-844.

25 **PA Maragos, RW Schafer** and **RM Mersereau**: 'Two-dimensional linear prediction and its application to adaptive predictive coding of images', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol ASSP-32, No. 32, 1984, pp.1213-1229.

26 **DT Nguyen, DJ Knowles** and **M Andrews**: 'A new technique in hi-low coding of images', *Proceedings 1st IASTED Int. Symp. on Signal Processing and its Applications (ISSPA-87)*, Brisbane, Australia, August 24-28, 1987, pp.662-660.

27 **M-Y Zou** and **R Unbehauen**:
'Stability Tests for
$B(z_1,z_2)=\sum_{m=0}^{2}\sum_{n=0}^{2} b_{mn} z_1^{m} z_2^{n}$
in closed form', *Electronics Letters*, Vol. 22, No. 11, 22 May 1986, pp.618-619.

Appendix 4A

Filter Coefficients

This appendix contains the denominator polynomial coefficients used in Figures 4.3, 4.4 and 4.5. The constant coefficient is the top left hand element in each case.

3 by 3 Polynomial:

-0.2850	-0.1746	0.2228
-0.1920	-0.5016	-0.8516
0.9928	-0.3338	-0.1587

5 by 5 Polynomial:

-0.2522	-0.2960	0.9947	0.2798	0.2146
0.5514	-0.7788	-0.9136	0.5919	-0.4384
0.7962	0.4699	-0.0971	0.6855	-0.3920
-0.2786	0.0871	-0.4289	0.4871	0.1808
-0.3939	-0.7469	-0.6626	0.5821	-0.3440

7 by 7 Polynomial:

-0.4869	-0.6388	0.0710	-0.8165	0.2687	0.1408	-0.1107
0.9517	0.7394	0.3163	0.4760	0.1592	0.9807	0.5996
-0.9975	0.4798	0.7738	-0.8200	0.5818	0.2048	-0.2703
-0.9861	-0.1790	0.1727	0.7182	0.4795	-0.2548	0.6720
0.5923	0.9902	0.8145	0.6640	-0.3229	0.6613	0.2507
0.1666	0.9894	0.6227	-0.6062	0.1444	-0.3397	0.0718
0.3063	-0.2519	0.5264	-0.3390	0.3445	0.2480	-0.6398

Future Research and Conclusions

5.1 Introduction

Almost any course of research raises more questions than are answered, or suggests future areas of promising research. In order that a thesis does in fact get submitted, it is impossible that all avenues are persued to their end. Consequently, this chapter outlines several topics worthy of study by future students of multidimensional digital signal processing. Section 5.2 lays the groundwork for a 2-D recursive filter design algorithm which incorporates a new measure of stability. This will allow iterative design procedures to explicitly cater for a stable, as well as optimal, design. The theoretical work in Section 5.2 suggests an unforseen application in finding the zeros of a polynomial in a complex variable. A suitable algorithm is described in Section 5.3, along with its similarities to an existing algorithm by Lehmer and Schur. Finally, future work in image coding research is mentioned in Section 5.4, followed by the major conclusions in Section 5.5.

5.2 2-D IIR Filter Design with a New Stability Metric

It should be apparent from Chapter 4 that the question of stability in a multidimensional digital filter is a difficult one. To recapitulate, given an N-D set of coefficients it is a non-trivial task to determine whether or not the filter that the coefficients describe is a stable one. (In a break with the definition adopted throughout most of this thesis, the stability metric is most conveniently defined if the Z transform is expanded in negative powers of $\mathbf{z}$. This will require the rootmap of a stable 2-D filter to be completely contained within the unit bidisc). For those versed in 1-D digital filter design, this has important ramifications when considering the design of N-D filters. The reason is that IIR filter designs usually proceed in an iterative fashion [1,p.267],[2,p.204],[3]–[9]. Specifically, we wish to design a filter of the form

$$A(\mathbf{z}) = \frac{N(\mathbf{z})}{D(\mathbf{z})} \tag{5.1}$$

where

$$D(\mathbf{z}) = \sum_{\mathbf{n}} a(\mathbf{n})\, \mathbf{z}^{-\mathbf{n}}, \quad \mathbf{n} \in R_a \tag{5.2}$$

which matches (in some way) a specified frequency response

$$S(\mathbf{z}), \quad \mathbf{z} \in \mathbf{T}^2 \tag{5.3}$$

All design algorithms attempt to choose the set $\{a(\mathbf{n})\}$ so that the error

$$E^p = \iint_{\mathbf{z} \in \mathbf{T}^2} |H(\mathbf{z}) - S(\mathbf{z})|^p \, d\mathbf{z} \tag{5.4}$$

is minimised. In the interests of mathematical tractability, the norm p is usually 2, although it may be argued that $p = \infty$ gives the "best" result by minimising the maximum error on $\mathbf{T}^2$. Some researchers have used large values of p (say, 20) to retain mathematical simplicity but at the same time approximating the case $p = \infty$. This is a direct result of the monotonicity of the p-norms [10] In any case, the problem is that most design algorithms cannot ensure that $H(\mathbf{z})$ is analytic ($D(\mathbf{z})$ free of zeros) outside the bidisc.

As each iteration of the design algorithm is executed, the proto-filter should be converging on the desired solution, and just as importantly, converging on a stable solution. The inherent difficulty in assessing the absolute stability, let alone a stability margin, usually means that no check is made on the stability of 2-D (or N-D) IIR filters during the design phase, although some methods impose a filter topology which guarantees stability at the expense of generality [12]. To date, and to the authors knowledge, only one stability metric has been proposed and integrated into 2-D digital filter design algorithms. The technique relies on certain properties of the 2-D complex cepstrum [13],[8], and can best be described as extremely computationally intensive. The purpose of this section is to propose a new stability metric, based on experience with the 2-D rootmap, and show how it may be incorporated into a contemporary filter design algorithm.

5.2.1 The Proposal

A portion of a typical (unstable) rootmap in the z_2–plane is shown in Figure 5.1.

The stability metric is extremely simple: ρ (which is the maximum rootmap modulus) is a measure of the stability or instability of the filter, so we may use this to drive or steer the design algorithm towards a stable solution. It is only necessary to consider one rootmap since if a rootmap contour crosses the z_1–plane unit circle, then the other rootmap *must* cross the z_2–plane. A simplified design algorithm is of the form:

Step 1: Estimate starting values for the filter coefficients

Step 2: Compute the maximum rootmap modulus

Step 3: Compute the maximum rootmap modulus derivatives with respect to the filter coefficients

Step 4: Estimate the new filter coefficients which drive the proto-filter towards a smaller error

Step 5: If the error is small enough, stop; otherwise go to *Step 2*

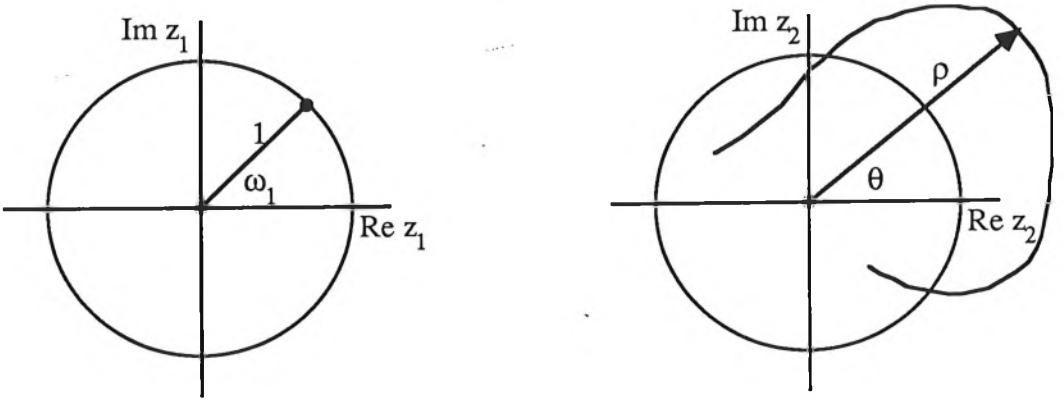


Figure 5.1: *Using the maximum rootmap modulus as a stability metric.*

ρ is the maximum rootmap modulus.

θ is the argument of the maximum rootmap zero.

ω_1 is the argument of z_1 for which $|z_2| = \rho$.

5.2.2 Step 1: Initial Estimates

The design algorithm will obviously converge more rapidly on the correct solution if the starting values for the unknown parameters are close to the true (optimal) values. More importantly, non-linear optimization routines are often highly sensitive to starting values for the unknown parameters, with a result that, at best, settles in a local rather than a global minimum; and worst, may diverge from the solution. An estimate that is often used in the literature is as follows. From eqn. (5.3), compute the inverse Fourier transform of $1/S(\omega_1,\omega_2)$ using the 2-D inverse FFT:

$$s(n_1,n_2) = FFT^{-1}\left[\frac{1}{S(\omega_1,\omega_2) + \varepsilon_s}\right] \tag{5.5}$$

The (ω_1,ω_2)-plane is sampled at some suitably high rate for the inverse FFT, and ε_s is a small

positive constant to ensure that $1/S(\omega_1,\omega_2)$ is never undefined. Assuming $N(\mathbf{z}) = 1$, we can match eqn. (5.2) to $s(n_1,n_2)$ to give

$$a(n_1,n_2) = s(n_1,n_2) \quad \text{for} \quad (n_1,n_2) \in R_a \tag{5.6}$$

Equation (5.6) gives the starting values for the unknown parameter set $\{a(\mathbf{n})\}$.

5.2.3 Step 2: Computing the Maximum Rootmap Modulus

Given the set of coefficients $\{a(m,n)\}$, we require the modulus of the "largest" zero in the rootmap. That is, if

$$z_2 = \zeta_2(\omega_1) \tag{5.7}$$

is the rootmap in the z_2–plane, then we seek

$$\rho = \max_{\omega_1} |\zeta_2(\omega_1)| \tag{5.8}$$

It is also necessary to know ω_1 for which eqn. (5.8) is true, and θ, the argument of z_2 for which $|z_2|=\rho$. Obviously it is extremely wasteful to compute the entire rootmap simply to find ω_1,θ and ρ. Hence we need an algorithm which will find the radius of the smallest circle which just encloses $\zeta_2(\omega_1)$.

There are many theorems relating the distribution of polynomial zeros to the polynomial coefficients, although a particularly useful result for the current work depends on the Schur transform [14,p.493] of a polynomial.

Let f(z) be the polynomial

$$f(z) = a_0 + a_1 z + a_2 z^2 + \cdots + a_N z^N \tag{5.9}$$

then the *reciprocal polynomial* [14,p.492] $f^*(z)$ is defined as

$$f^*(z) = z^N \overline{f(1/\bar{z})} \tag{5.10}$$

The Schur transform of f(z), denoted T[f(z)], is given by

$$\begin{aligned} T[f(z)] &= \bar{a}_0 f(z) - a_N f^*(z) \\ &= \sum_{k=0}^{N-1} c_k z^k \end{aligned} \tag{5.11}$$

where

$$c_k = \bar{a}_0 a_k - a_N \overline{a_{N-k}}.$$

Note that if f(z) is of degree N, then T[f(z)] is of degree N–1. The iterated Schur transform T^k of f(z) is

$$T^k[f(z)] = T[T^{k-1}[f(z)]] \tag{5.12}$$

and it is of use to define the sequence of numbers

$$\gamma_k = T^k[f(0)], \qquad k = 1,2,...,N \tag{5.13}$$

The main result of interest is the following: f(z) is free of zeros in the unit disc if and only if [14,p.493]

$$\gamma_k > 0, \qquad k = 1,2,...,N \tag{5.14}$$

A linear transformation of the argument z allows us to test for the absence of zeros in other discs. For example, if f(z) has a zero inside $|z| = r$, then f(rz) has a zero inside the unit circle.

Returning now to the rootmap, for a fixed value of $z_1 \in \mathbf{T}$, eqn. (5.2) may be written as

$$D(\omega_1,z_2) = \sum_{n=0}^{N} d_n(\omega_1)\, z_2^{-n} \tag{5.15}$$

where

$$d_n(\omega_1) = \sum_{m=0}^{N} a(m,n)\, e^{-jm\omega_1} \tag{5.16}$$

If we form the sequence

$$\gamma_k = T^k[D(\omega_1,0)] \tag{5.17}$$

then $D(\omega_1,z_2)$ has all of its zeros contained within $|z_2| = 1$ (for a given ω_1) provided

$$\gamma_k > 0, \qquad k = 1,2,...,N$$

This statement is consistent with the result in eqn. (5.14) since $D(\omega_1,z_2)$ is defined in negative powers of z_2. In order to find the circle of minimum radius which contains all N zeros of eqn. (5.15), we choose r as a test radius, and form the transforms

$$T^k[D(\omega_1,z_2/r)] \tag{5.18}$$

and then generate the sequence

$$\gamma_k = T^k[D(\omega_1,0)] \tag{5.19}$$

If

$$\gamma_k > 0, \qquad k = 1,2,...,N$$

then all the zeros of D lie within $|z_2| = r$, and r may be reduced accordingly. Similarly, if $\exists k$: $\gamma_k < 0$,

then $|z_2| = r$ does not contain all of the zeros of D and r must be increased. It may appear that eqn. (5.19) is identical to eqn. (5.17). This is not the case because the scale factor r in eqn. (5.18) is absorbed into the coefficients of D before setting $z_2 = 0$. The search for r such that $r = \rho$ may be accelerated by having a reasonable idea where the modulus of the zeros will lie on the real line. Fortunately, there are some especially simple results from analysis which will aid our search for ρ. In particular [15,p.88,90]

$$r_{min} = \rho \geq \max \left\{ \left|\frac{d_N}{d_{N-1}}\right|, 2\left|\frac{d_{N-1}}{d_{N-2}}\right|, \ \ldots \ , 2\left|\frac{d_1}{d_0}\right| \right\} \tag{5.20a}$$

and

$$r_{max} = \rho \leq \max_{0 \leq \nu \leq N-1} \left\{ \frac{(N-\nu)!\ \nu!}{N!} \left|\frac{d_\nu}{d_0}\right| \right\}^{\frac{1}{N-\nu}} \tag{5.20b}$$

The interval $[r_{min}, r_{max}]$ may then be searched (preferably via a binary search) until ρ is found. The foregoing discussion is summarized in Algorithm 1. After each pass through Algorithm 1, ω_1 is incremented , and if the new value of ρ is greater than the last value, it is recorded as *the* maximum rootmap modulus (along with ω_1). This process continues until all values of ω_1 have been used.

- Specify a(l,k), ω_1 and ε (the tolerance for accepting r as ρ)
- Calculate r_{min} and r_{max}
- **Repeat**

  $r = (r_{min} + r_{max})/2$

  Form $T^k[D(\omega_1, z_2/r)]$ for k = 1,2,...,N

  $\gamma_k = T^k[D(\omega_1, 0)]$ for k = 1,2,...,N

  If $\gamma_k > 0 \ \forall k$ **then**

  $r_{max} = r$

  else

  $r_{min} = r$

Until $(r_{max} - r_{min})/r < \varepsilon$

Algorithm 1: Finding ρ using Schur transforms and a binary search

Calculating the Argument of the Zero of Maximum Modulus

Anticipating the business of the next section, where it is necessary to evaluate the derivatives of the zero of maximum modulus with respect to the filter coefficients, it is necessary to know the argument of the zero of maximum modulus: θ. Algorithm 1 (above) simply told us on which circle we could find the zero. It should be noted that although Algorithm 1 is executed many times (for each value of ω_1), the argument θ is computed only *once*.

Writing

$$z_2 = \rho \exp(j\varphi) \tag{5.21}$$

eqn. (5.15) becomes

$$D(\omega_1,\varphi) = \sum_{n=0}^{N} d_n(\omega_1)\, \rho^{-n} \exp(-jn\varphi)$$

and if we absorb ρ into $d_n(\omega_1)$:

$$D(\omega_1,\varphi) = \sum_{n=0}^{N} g_n(\omega_1) \exp(-jn\varphi) \tag{5.22}$$

where $g_n(\omega_1) = d_n(\omega_1)\rho^{-n}$.

For a fixed value of ω_1, eqn. (5.22) is simply the discrete Fourier transform of the sequence $\{g_n(\omega_1)\}$ and may be computed efficiently using the 1-D FFT. Thus, θ occurs when†

$$(\,\mathrm{Re\ FFT}\{g_n(\omega_1)\} = 0\,) \;\cap\; (\,\mathrm{Im\ FFT}\{g_n(\omega_1)\} = 0\,) \tag{5.23}$$

Note that the DFT will only return a finite number of samples on $z_2 = \rho \exp(j\varphi)$, and in general θ will not fall exactly on a sample. However, if the zero falls between samples, a simple linear interpolator (see Figure 5.2) will improve the estimate of θ. In practice, the interpolated estimates of θ will probably be different, so a better estimate is

$$\theta_{better} = \frac{1}{2}\left[\theta^{(re)} + \theta^{(im)}\right] \tag{5.24}$$

The appearance of N in eqn. (5.22) should not suggest that we are restricted to N samples of φ.

---

† There are degenerate cases where Re FFT{g} or Im FFT{g} have roots of even multiplicity making eqn. (5.23) useless. However, preliminary experimental results suggest that such cases will be rare in practice.

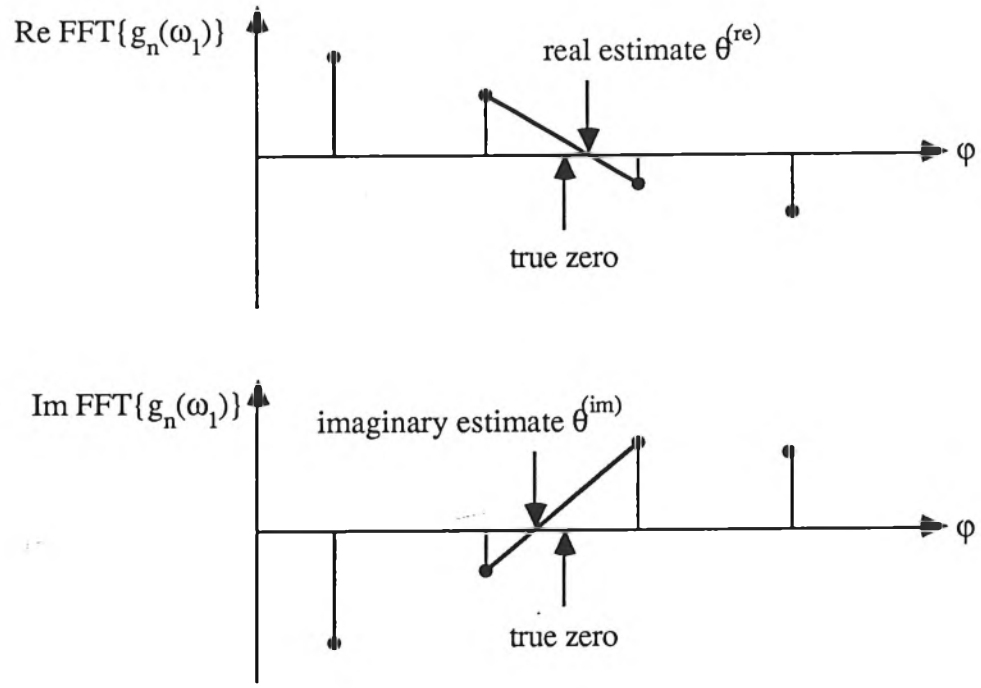


Figure 5.2:*Finding the argument of the zero of maximum modulus using the fast Fourier transform.*

Indeed we may pad out the sequence $\{g_n(\omega_1)\}$ with zeros and compute the FFT with M samples (M>N). The advantage in using eqn. (5.23) to find θ is that we know from the outset that θ will be computed with an error less than $2\pi/M$ (see eqn. (5.24)) in approximately $M\log_2 M$ complex multiplications.

5.2.4 Step 3: Evaluating the Rootmap Modulus Derivatives

The derivatives of ρ with respect to the filter coefficients are required for the optimization phase. Once ρ, θ and ω_1 are known, it is quite straightforward to compute the derivatives

$$\frac{\partial \rho}{\partial a(m,n)}, \quad (m,n) \in R_a \tag{5.25}$$

It is convenient to put

$$z_2 = \rho \exp(j\theta) \tag{5.26}$$

so that

$$\frac{\partial z_2}{\partial a(m,n)} = \frac{\partial \rho}{\partial a(m,n)} \exp(j\theta) + j\rho \frac{\partial \theta}{\partial a(m,n)} \exp(j\theta)$$

from which

$$\frac{\partial \rho}{\partial a(m,n)} = \mathrm{Re}\left\{ \frac{\partial z_2}{\partial a(m,n)} \exp(-j\theta) \right\} \tag{5.27}$$

and eqn. (5.25) follows immediately.

Using eqns. (5.15) and (5.16) we have

$$D(\omega_1, z_2) = \sum_{m=0}^{N} \sum_{n=0}^{N} a(m,n)\, z_2^{-n} \exp(-jm\omega_1) \tag{5.28}$$

from which

$$\frac{\partial z_2}{\partial a(p,q)} = -\frac{z_2^{-q} \exp(-jp\omega_1)}{D'(\omega_1, z_2)} \tag{5.29}$$

where the denominator derivative is taken with respect to z_2. Substituting eqn. (5.29) into eqn. (5.27) yields

$$\frac{\partial \rho}{\partial a(p,q)} = \mathrm{Re}\left\{ -\frac{\rho^{-q} \exp[-j(p\omega_1 + (q+1)\theta)]}{D'(\omega_1, z_2)} \right\} \tag{5.30}$$

Despite the complexity of eqn. (5.30), an efficient recursive implementation exists which relates the derivatives for various values of (p,q). From eqn. (5.29), with (p,q)=(0,0) we have

$$\frac{\partial z_2}{\partial a(0,0)} = \frac{-1}{D'(\omega_1, z_2)} \tag{5.31}$$

and it is easily verified that

$$\frac{\partial z_2}{\partial a(p,q+1)} = z_2^{-1} \frac{\partial z_2}{\partial a(p,q)} \tag{5.32}$$

$$\frac{\partial z_2}{\partial a(p+1,q)} = \exp(-j\omega_1) \frac{\partial z_2}{\partial a(p,q)} \tag{5.33}$$

and

$$\frac{\partial z_2}{\partial a(p+1,q+1)} = z_2^{-1} \exp(-j\omega_1) \frac{\partial z_2}{\partial a(p,q)} \tag{5.34}$$

(Equation (5.34) is essentially redundant given eqns. (5.32) and (5.33)). Once eqn. (5.31) has been evaluated, a single complex multiplication or division yields successive derivatives. Another complex multiplication in eqn. (5.27) gives the final result.

5.2.5 Step 4: The Next Best Estimate

As explained in Section 5.2.1, the optimal filter coefficients are obtained iteratively. Using the results in the previous sections, we are now in a position to estimate a better approximation to the optimal parameters. However, the error measure defined in eqn. (5.4) is inadequate as it does not include the stability metric, ρ, introduced in eqn. (5.8). Accordingly, we modify the error measure as follows. Let the desired frequency response be

$$S(\omega_1,\omega_2) \tag{5.35}$$

and the actual frequency response be

$$A(\omega_1,\omega_2) = \frac{N(\omega_1,\omega_2)}{D(\omega_1,\omega_2)} \tag{5.36}$$

where

$$N(\omega_1,\omega_2) = \sum_m \sum_n b(m,n) \exp(-jm\omega_1) \exp(-jn\omega_2), \quad (m,n) \in R_b \tag{5.37}$$

$$D(\omega_1,\omega_2) = \sum_m \sum_n a(m,n) \exp(-jm\omega_1) \exp(-jn\omega_2), \quad (m,n) \in R_a \tag{5.38}$$

Now define a parameter vector

$$\phi = \left[\mathbf{a} \mid \mathbf{b}\right]^T \tag{5.39}$$

where the partitions **a** and **b** are vectors formed from the coefficient sets $\{a(\mathbf{n})\}$ and $\{b(\mathbf{n})\}$ when ordered lexicographically. The total error to be minimised is now

$$E(\phi) = E_A(\phi) + E_S(\phi) \tag{5.40}$$

where $E_A(\phi)$ is the mean square error in approximating eqn. (5.35) by eqn. (5.36):

$$E_A(\phi) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} |S(\omega_1,\omega_2) - A(\omega_1,\omega_2)|^2 \, d\omega_1 \, d\omega_2 \tag{5.41}$$

In practice $E_A(\phi)$ is usually approximated by a summation over a dense grid of frequencies in the (ω_1,ω_1)-plane. The second error term in eqn. (5.40), $E_S(\phi)$, represents the *stability error*. Thus $E_S(\phi)$ is numerically large if the parameter vector ϕ represents an unstable (or marginally stable) filter and remains small for stable filters. $E_S(\phi)$ may take several forms, for example

$$E_S(\phi) = \begin{cases} \alpha\rho & \text{if } \rho \geq \rho_{thresh} \\ 0 & \text{if } \rho < \rho_{thresh} \end{cases} \tag{5.42}$$

or

$$E_S(\phi) = \alpha\rho^2 \tag{5.43}$$

Here ρ is implicitly a function of ϕ and α is a suitable scalar to weight the stability error relative to the approximation error. In eqn. (5.42) ρ_{thresh} may be chosen in the order of say, 0.8, so that rootmap contours approaching the unit bidisc are penalized heavily.

If we have a value of ϕ, which we now call ϕ_{old}, sufficiently close to the optimum value so that $E(\phi)$ may be approximated by the quadratic form

$$E(\phi_{old}+\Delta\phi) \approx E(\phi_{old}) + \Delta\phi^T \nabla E(\phi_{old}) + \frac{1}{2}\Delta\phi^T \mathbf{H}(\phi_{old})\,\Delta\phi \tag{5.44}$$

it is a simple matter to find the correction vector $\Delta\phi$ such that eqn. (5.44) is a minimum. The important new terms in eqn. (5.44) are:

$\nabla E(\phi_{old}) \equiv$ the gradient vector of $E(\phi)$ evaluated at $\phi = \phi_{old}$.

$\mathbf{H}(\phi_{old}) \equiv$ the Hessian matrix of $E(\phi)$ evaluated at $\phi = \phi_{old}$.

In order to find $\Delta\phi$ such that $E(\phi_{old}+\Delta\phi)$ is a minimum, differentiate eqn. (5.44) with respect to the vector $\Delta\phi$:

$$\frac{\partial E(\phi_{old}+\Delta\phi)}{\partial\Delta\phi} = \nabla E(\phi_{old}) + \mathbf{H}(\phi_{old})\,\Delta\phi \tag{5.45}$$

and for a local minimum we must have

$$\Delta\phi = -\,\mathbf{H}(\phi_{old})^{-1}\,\nabla E(\phi_{old}) \tag{5.46}$$

in which case the minimum value of eqn. (5.44) occurs when

$$\phi = \phi_{old} + \Delta\phi \tag{5.47}$$

In reality $E(\phi)$ is only approximately quadratic so several applications of eqns. (5.47) and (5.46) are necessary to find the true minimum. Furthermore, to evaluate eqn. (5.46) we require the gradient vector ∇E and the Hessian $\mathbf{H}$. $\nabla E(\phi)$ may be written as

$$\nabla E(\phi) = \nabla E_A(\phi) + \nabla E_S(\phi) \tag{5.48}$$

The gradient of the stability error was computed in Section 5.2.4 as

$$\nabla E_S(\phi) = \alpha\left[\ \cdots\ \frac{\partial\rho}{\partial a(m,n)}\ \cdots\ \right]^T, \quad (m,n) \in R_a \tag{5.49}$$

where each element of $\nabla E_S(\phi)$ is computed from eqn. (5.27) and eqns. (5.31)–(5.33). (Note that in eqn. (5.49) we have chosen the form of $E_S(\phi)$ given by eqn. (5.42)). The gradient of the approximation error, $\nabla E_A(\phi)$, is computed from eqn. (5.41). $\nabla E_A(\phi)$ will not be evaluated explicitly in this section because several formulations are possible, depending on the criterion established for measuring the approximation error. For example, some researchers have considered the error in the real and imaginary parts separately (for example, [9]), while other have opted for a specified magnitude and group delay response (for example, [6]).

The system of linear equations in eqn. (5.45) requires the Hessian matrix of $E(\phi)$ in order to find the unknown parameter correction vector, $\Delta\phi$. $\mathbf{H}(\phi)$ is given by

$$\mathbf{H}(\phi) = \mathbf{H}_A(\phi) + \mathbf{H}_S(\phi) \tag{5.50}$$

since the Hessian may be viewed as a linear operator. Again, $\mathbf{H}_A(\phi)$ depends on the exact form of $E_A(\phi)$ and therefore will not be discussed. $\mathbf{H}_S(\phi)$ may be computed in a similar manner to $\nabla E_S(\phi)$, however, it may be argued that if the model fits badly (eqn. (5.36)), the exact second partial derivatives in $\mathbf{H}(\phi)$ may have a destabilizing effect [16,p.523]. For this reason, $\mathbf{H}_S(\phi)$ may be approximated as

$$\mathbf{H}_S(\phi) = \left[\frac{\partial \rho}{\partial \phi_k} \cdot \frac{\partial \rho}{\partial \phi_l} \right] \tag{5.51}$$

5.2.6 General Remarks

Without experimental results, it is not clear how well the proposed algorithm will perform relative to the cepstral technique. However, the proposed method appears promising since all steps in the algorithm are 'one-dimensional', whereas spectral factorization using the 2-D cepstrum typically requires over 100 2-D FFTs per iteration [7],[9].

The convergence of ϕ on the optimal solution may be made more robust by using the Levenburg-Marquardt algorithm [16,p.523], which appears to be the algorithm of choice in the literature (for example, [5],[7],[9]). Essentially, Marquardts method involves a minor modification to eqn. (5.46); namely

$$(\mathbf{H}(\phi) + \lambda \mathbf{I})\, \Delta\phi = -\nabla E(\phi) \tag{5.52}$$

where λ is a positive scalar and $\mathbf{I}$ is the identity matrix. For small values of λ, eqn. (5.52)

minimises the quadratic form of eqn. (5.44) since the Hessian matrix dominates the identity matrix. If λ is very large the reverse is true: **I** swamps **H** and eqn. (5.52) corresponds to the steepest descent method [19,p.119] for minimising $E(\phi)$. Thus by varying λ, we may continuously switch between the slow-but-sure steepest descent algorithm and the fast (but locally convergent) inverse Hessian method.

5.3 A New Method for Determining the Zeros of a Polynomial

The results of Section 5.2.3 concerning the zero of maximum modulus in the rootmap may be applied to the general problem of determining the complex zeros of a polynomial with (possibly) complex coefficients. Since the technique uses the Schur transform, which is a 'sharp' test for the inclusion of zeros within a general disc, the new method could be viewed as a modification of the well-known Lehmer-Schur algorithm for finding polynomial zeros. It is instructive to first review the operation of Lehmer's method before detailing the proposed new method.

5.3.1 The Lehmer-Schur Algorithm [17,p.355]

The Schur transform is central to Lehmer's method. It is convenient to consider the transform embedded in a decision element such as shown in Figure 5.3. Basically, the Schur transform is used to determine if a polynomial g(z) has a zero inside $|z| = 1$. If the polynomial of interest is

$$f(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_N z^N \tag{5.53}$$

then f(z) has a zero inside $|z - c| = r$ if $g(z) = f(rz+c)$ has a zero inside the unit circle. The basic philosophy is that by choosing r and c judiciously, we may cover the z plane with overlapping circles until one or more zeros are contained within a specific circle. The algorithm then 'homes in' on one of the zeros and removes it by synthetic division. The search strategy is as follows [17,p.356]:

Step 1

Put $r = 2^k$ where $k = 0,1,2,\ldots$ and find the first value of k for which $f(|rz|)$ contains a zero†. Therefore at least one zero is contained within the annulus

$$R = 2^{k-1} \leq |z| \leq 2^k = 2R \tag{5.54}$$

---

† If $f(|z|)$ contains a zero then halve r.

Step 2

The annulus in eqn. (5.54) may be covered by 8 overlapping circles as shown in Figure 5.4. Each circle has a radius 4R/5 and a centre at

$$c_k = \frac{3R}{2\cos \pi/8} \exp(jk\pi/4), \quad k = 0,1,...,7 \tag{5.55}$$

We may now use the decision element in Figure 5.3 to find which one of these discs contains a zero.

Step 3

Having found the c_k for which a zero is included, we halve the radius (starting with R_1=4R/5) until an annulus is found which contains a zero. This annulus may now be covered with 8 smaller overlapping circles in a manner similar to ***Step 2***. The procedure is repeated in an obvious way until a zero has been found to sufficient accuracy.

Step 4

The zero is removed by synthetic division and the whole process is repeated, starting with the annulus given by eqn. (5.54) since only one zero contained within |z| = 2R will have been found.

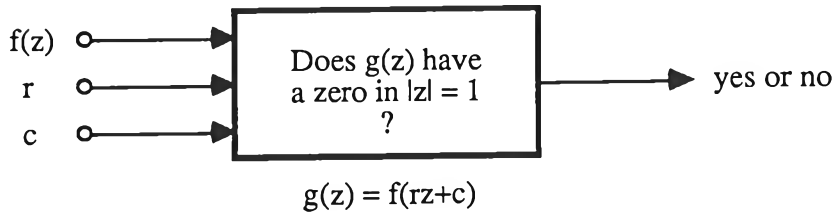
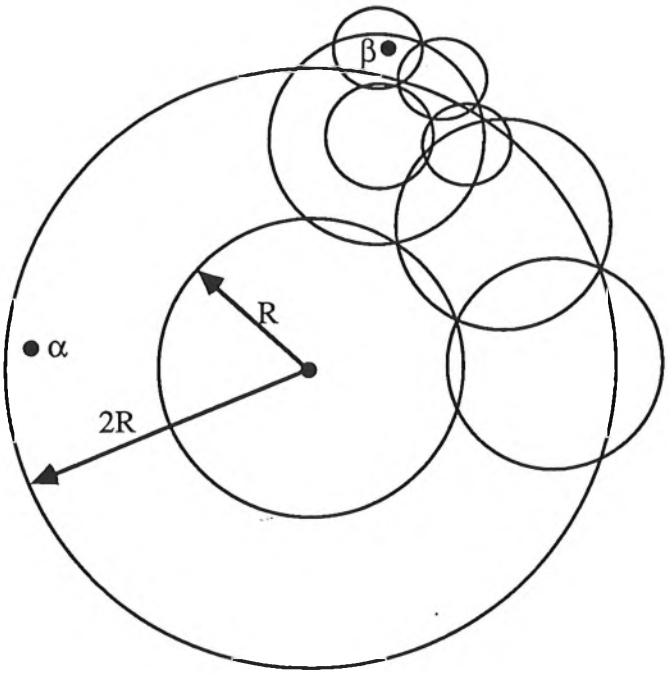


Figure 5.3:*A decision element based on the Schur transform. If g(z) as a zero inside |z| = 1 then f(z) has a zero inside |z – c| = r.*

5.3.2 A New Algorithm using the Schur Transform

The only similarity that the proposed method has with Lehmer's technique is that it also has the Schur transform at the heart of its operation. Basically, we use a repeated application of the transform to find the modulus of the smallest zero, and then evaluate the polynomial on that circle using the fast Fourier transform. The zeros of the transform then give us the phase of all zeros with that modulus. Those zeros are removed by synthetic division and the search continues for the zero of next smallest modulus.



α = zero which first satisfied eqn. (5.54)

β = zero actually 'homed in' on

Figure 5.4:*Lehmer's method of covering the z-plane with overlapping circles in order to find polynomial zeros. (After [17,p.356]).*

Initial Bounds on the Polynomial Zeros

Lower and upper bounds on the modulus of the polynomial zeros will speed the binary search for the zero of smallest modulus. Equation (5.20) provides such a starting point. The lower bound will be labelled α and upper bound will be labelled β. The degree of uncertainty associated with these bounds is strongly coefficient-dependent, but they are finite bounds nevertheless.

Determining the Modulus of the Smallest Zero

Given the bounds $[\alpha,\beta]$ on the polynomial zeros, we wish to find the smallest circle which just encloses the zero of smallest modulus. This task will be accomplished using the Schur transform and a binary search algorithm. A suitable search strategy is shown in Algorithm 2 below. Note that the test to see if f(z) has a zero inside $|z| = R$ is executed inside the 'black box' shown in Figure 5.3 with $c = 0$ and $r = R$. The successive bisection of each interval continues until the relative change in the zero modulus changes by less than ε, a suitably small number. After P passes

through the **Repeat-Until** loop, the zero modulus is contained within an annulus approximately $(\beta-\alpha)2^{-P}$ units wide, thus the maximum iteration count may be specified prior to execution.

$R_{min} = \alpha$

$R_{max} = \beta$

Repeat

$R = (R_{min}+R_{max})/2$

if f(z) has a zero in $|z| = R$ **then**

$R_{max} = R$

else

$R_{min} = R$

Until $(R_{max}-R_{min})/R < \varepsilon$

Algorithm 2: Finding the modulus of the smallest zero

Determining the Phase of the Smallest Zero

If we specify the zero of smallest modulus as

$$z_0 = R \exp(j\phi), \quad \phi \in [0,2\pi) \tag{5.56}$$

then

$$f(z_0) \equiv f(\phi) = \sum_{k=0}^{N} a_k R^k \exp(jk\phi) = \sum_{k=0}^{N} b_k \exp(jk\phi) \tag{5.57}$$

where

$$b_k = a_k R^k$$

and all that remains is to find ϕ such that $f(\phi) = 0$. Now discretize ϕ by putting

$$\phi = \frac{2\pi n}{M}, \quad \text{where} \quad M > N \quad \text{and} \quad n = 0,1,...,M-1 \tag{5.58}$$

so that eqn. (5.57) becomes

$$f(\phi) \equiv f(n) = \sum_{k=0}^{M-1} b_k \exp(j2\pi kn/M) \tag{5.59}$$

where $b_k = 0$ for k = N+1,...,M–1. If M is a power of 2, then eqn. (5.59) may be evaluated using the radix-2 fast Fourier transform. Finally, the M points of f(n) are searched for zeros, that is, record the values of n for which

$$\{ \text{Re } f(n) = 0 \} \cap \{ \text{Im } f(n) = 0 \} \tag{5.60}$$

In practice, adjacent samples of f(n) will span a zero so that a sign change in Re f(n) and Im f(n) is detected rather than a true zero (see Figure 5.2). The phase of the zero may now be estimated using eqn. (5.58).

Polishing the Zero Estimate

Both the modulus and phase of the zero can be obtained to an arbitrary accuracy by specifying an extremely small value of ε in Algorithm 2, and choosing a very large value of M in eqn. (5.58). However, this course of action is taken at the expense of execution time. A more realistic approach is to tolerate a moderate error in R and ϕ, and then switch to a fast locally convergent routine such as the Newton-Raphson method.

5.4 Suggestions for Image Coding Research

Despite volumes of results relating to image coding in the literature, there is no underlying theoretical structure that unifies the various branches of what is called signal coding. Because of this deficiency, there is a certain ad-hoc nature about many techniques, but it does have the advantage that new methods and ideas are only limited by the researcher's imagination; and a desire to investigate the practicalities of his/her brain-child. During the research which culminated in the material presented in Chapter 3 regarding analysis-frame decorrelation, the following ideas suggested themselves (or were suggested) as possible improvements on the basic techniques considered.

5.4.1 Choice of Transform Method

It was found that both the Fourier and the Hadamard transforms worked effectively at decorrelating subimages prior to linear predictive analysis. The Hadamard transform had 'the edge' over its continuous counterpart because of its speed of computation and severe decorrelating properties. The next logical objective is to find the fastest of all orthogonal transforms which perform as well as the Walsh-Hadamard transform. Intuitively, the simpler the transform kernel, the faster will be the execution of the transform. An obvious candidate for investigation is the Haar transform

[18,p.399], which requires fewer arithmetic operations than the Hadamard transform because of the sparseness of its kernel matrix. It is also possible to 'design' ones own transform or even experiment with non-transform methods of de-correlation.

5.4.2 Other methods

Depending on the specific application of linear prediction, it may be possible to re-do an analysis or simply ignore the results of an unstable predictor. For example, imagine that a particular analysis frame yields an unstable prediction filter, and there is time to re-do the analysis. If the analysis frame size is increased (say by 50% on a side), then that may introduce image features which reduce the frame-wide correlation, hopefully resulting in a stable prediction filter. The penalty one must pay is sub-optimality in the model (i.e. a higher error bit-rate) since smaller analysis frames adapt more efficiently to spatial variations in the image statistics.

If the area of application is TV or video coding, isolated unstable prediction filters may be discarded; the offending analysis frame being replaced with the corresponding frame from the previous image (in temporal sequence). Such a scheme will result in minimal transmission rate overheads and generate little subjective annoyance if the rate of change of events in successive images is small.

5.5 Major Conclusions

Several problems have been examined relating to two-dimensional recursive filters and their application to linear predictive image coding. It was found that for natural images (that is, faces, objects etc) a significant proportion of analysis frames produced unstable prediction filters. The instability was attributed to two main causes. Firstly, for regions with high inter-pixel correlation, filters tended towards instability because they were essentially trying to predict non-decaying phenomena: marginal stability at best. Secondly, for an analysis frame which spans an image frame which looks unbounded (if continued indefinitely beyond the frame boundaries) the prediction analysis is 'fooled' into thinking that it is indeed observing a portion of an unbounded signal. An effective counter to both of these problems was to decorrelate the analysis frame prior to analysis. De-correlation obviously reduced interpixel correlation but also produced sufficient undulation in the signal form to prevent the mathematics from incorrectly assuming it was given a 'window' on an unbounded signal. Both the Fourier and Hadamard transforms proved effective at reducing the correlation in the analysis frame, although based on speed and transmission-rate overhead, the Hadamard transform method with sequency ordering emerged as the preferred method. Given the compression ratios currently possible with linear prediction schemes, the extra bandwidth required to transmit the information relating to the sequency components was acceptable.

The problem of what to do with an unstable prediction filter is symptomatic of the more general problem of detecting unstable recursive filters and trying to stabilize them after detection. A useful and informative device for displaying the stability condition of a two-dimensional recursive filter is the rootmap. A simple and robust technique for its computation was presented and it is hoped that this under-rated tool will find more application in the future. An understanding of the 'mechanics' of rootmap calculation lead to a deeper appreciation of the inner workings of the stability test of Maragos *et al.* In particular, that the simplicity of their test, while attractive from a computational point of view, allowed too many unstable filters to pass undetected. The inadequacies of Maragos' test prompted an investigation and development of a new suite of necessary stability tests, suitable for real-time implementation. A broad range of tests assured that a much higher percentage of unstable filters were detected. The exact form of each test suggested a simple stabilization strategy, depending on which test(s) failed. These methods, (and those of the preceding paragraph), have been demonstrated by experiment, but further tests will be required to quantify the performance advantage of each method.

Finally, some new avenues of future research were suggested. The most promising idea appears to be a new stability metric which, incorporated into a contemporary recursive filter design algorithm, should ensure that the final filter satisfies both optimality and stability constraints. On the face of the theoretical groundwork developed so far, the proposed technique has the potential to be much faster than the current method of incorporating a stability constraint: that of the two-dimensional complex cepstrum. The proposed method is characterised by many one-dimensional operations whereas the cepstral method requires several hundred two-dimensional operations. The theoretical work on maximum rootmap modulus also spawned an idea for a new method of determining the zeros of a polynomial. The technique finds the zeros in order of magnitude, then phase, starting with the zero of smallest modulus. Finally, some ad-hoc improvements to analysis-frame de-correlation were suggested which will almost certainly lead to incremental improvements in the methods examined in this thesis.

5.6 References

1 **LR Rabiner** and **B Gold**: 'Theory and application of digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

2 **DE Dudgeon** and **RM Mersereau**: 'Multidimensional digital signal processing', Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

3 **MS Bertran**: 'Approximation of digital filters in one and two dimensions', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-23, No. 5, 1975, pp.438-443.

4 **JA Cadzow**: 'Recursive filter synthesis via gradient based algorithms', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-24, No. 5, 1976, pp.349-355.

5 **JH Lodge** and **MM Fahmy**: 'An optimization technique for the design of half-plane 2-D recursive digital filters', *IEEE Trans. Circuits and Systems,* Vol. CAS-27, No. 8, 1980, pp.721-724.

6 **SAH Aly** and **MM Fahmy**: 'Design of two-dimensional recursive digital filters with specified magnitude and group delay characteristics', *IEEE Trans. Circuits and Systems,* Vol. CAS-25, No. 11, 1978, pp.908-915.

7 **JW Woods, J-H Lee** and **I Paul**: 'Two-dimensional IIR filter design with magnitude and phase error criterion', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-31, No. 4, 1983, pp.886-893.

8 **MP Ekstrom, RE Twogood** and **JW Woods**: 'Two-dimensional recursive filter design – a spectral factorization approach', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-28, No. 1, 1980, pp.16-26.

9 **J-H Lee** and **Y-M Chen**: 'A new method for the design of two-dimensional recursive digital filters', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-36, No. 4, 1988, pp.589-598.

10 **SY Hwang**: 'On monotonicity of L_p and l_p norms', *IEEE Trans. Acoustics, Speech and Signal Processing*, December, 1975, pp.593-594.

12 **H Chang** and **JK Aggarwal**: 'Design of two-dimensional semi-causal recursive filters', *IEEE Trans. Circuits and Systems,* Vol. CAS-25, No. 12, 1978, pp.1051-1059.

13 **MP Ekstrom** and **JW Woods**: 'Two-dimensional spectral factorization with applications in recursive digital filtering', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-24, No. 2, 1976, pp.115-128.

14 **P Henrici**: 'Applied and computational complex analysis. Vol. 1', Wiley, New York, 1974.

15 **HS Wilf**: 'Mathematics for the physical sciences', Wiley, New York, 1962.

16 **WH Press, BP Flannery, SA Teukolsky** and **WT Vetterling**: 'Numerical recipes – the art of scientific computing', Cambridge University Press, New York, 1987.

17 **A Ralston**: 'A first course in numerical analysis', McGraw-Hill, New York, 1965.

18 **DF Elliott** and **KR Rao**: 'Fast transforms: algorithms, analyses, applications', Academic Press, Orlando, 1982.

19 **AC Bajpai, LR Mustoe** and **D Walker**: 'Advanced engineering mathematics', Wiley, Great Britain, 1979.