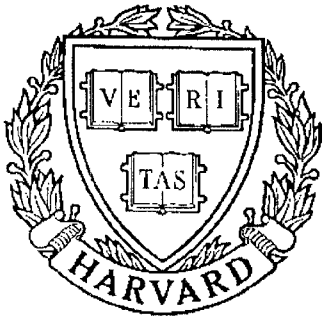
THESIS REPORT
Master's Degree

SYSTEMS
RESEARCH
CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Image Deconvolution Using Multiple Sensors

by N.D. Sidiropoulos
Advisor: J.S. Baras

M.S. 90-6

ABSTRACT

Title of Thesis:	Image Deconvolution Using Multiple Sensors
Name of Degree Candidate:	Nicholaos D. Sidiropoulos
Degree and Year:	Master of Science, 1990
Thesis directed by:	Dr. J. S. Baras
	Professor
	Department of Electrical Engineering
	Dr. C. A. Berenstein
	Professor
	Department of Mathematics

We consider the two dimensional Analytic Bezout Equation (ABE) and investigate the properties of a particular solution, based on certain conditions imposed on the convolution kernels. We use a family of suitably chosen sensors which besides being strongly coprime also satisfies additional technical conditions. The results permit the reconstruction of the original signal with arbitrarily good resolution, i.e. achieving arbitrarily large bandwidths, depending solely upon computational resources.

The theoretical foundations of this technique provide a rigorous mathematical framework, and simulation results have been promising. The feasibility of constructing reasonably good discrete-time, finite-bandwidth approximations has been established , and efficient Data Parallel Grid layouts that perform the required computation have been designed. A number of implementation problems arising out of the need to approximate a basically infinite computation have been addressed.

Image Deconvolution Using Multiple Sensors

by

Nicholaos D. Sidiropoulos

Thesis submitted to the Faculty of The Graduate School of The University of Maryland in partial fulfillment of the requirements for the degree of Master of Science

May 1990

Advisory Committee:

Dr. J. S. Baras	Chairman/Advisor
Dr. C. A. Berenstein	Advisor
Dr. P. Krishnaprasad	

DEDICATION

To my parents Dimitrios and Maria

ACKNOWLEDGMENTS

I am indebted to both my research advisors, Dr. Carlos Berenstein and Dr. John Baras, for their continous support, encouragement and contribution throughout the course of this work. I am especially thankful to Dr. Carlos Berenstein for his extensive tutoring, which greatly helped develop my understanding of the subject. Dr. John Baras has been instrumental in pointing out new research directions and providing his experience and judjement.

Part of the original work which this thesis has been built upon is due to Dr. Alain Yger of the University of Bordeaux, France. I am grateful to him for the fruitful discussions we had during the early stages of this research. Dr. Vince Patrick has pursued a different approach to the problem, and I have benefited from his work.

I would like to thank the System Research Center for the financial support it provided to me in the form of a research assistantship (NSF grant NSFD CDR 8803012). I would also like to thank the UMIACS for the use of its Connection Machine. Finally I would like to express my gratitude to Yagyensh Pati for his help and support in the typesetting of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

NOTATION

ABE	Analytic Bezout Equation
AGWN	Additive Gaussian White Noise
BW	BandWidth
CM	Connection Machine
FT	Fourier Transform
DFT	Discrete time Fourier Transform
IDFT	Inverse Discrete time Fourier Transform
LTI	Linear Time Invariant
S/N	Signal to Noise power ratio
χ_K	Characteristic Function over the compact set $K \subset R^n$

CHAPTER
ONE

Introduction and Problem Setup

1.1 A brief overview

Signal deconvolution is a fundamental problem related to a variety of scientific and engineering disciplines, notably communications and signal processing, control theory, remote sensing, pure and applied mathematics, and others. The problem can be initially stated as follows: We are given data that consist of local averages of a particular signal $I(\cdot)$ and wish to synthesize $I(\cdot)$ itself based on these averages. As stated the problem is a very general one and does not admit a unique solution, unless certain constraints are imposed on $I(\cdot)$ or, alternatively, strong a priori assumptions are made. Clearly another fundamental question has to be answered first: How much data is adequate? Equivalently, what is the minimal dimensionality of the data set in order for the problem to have a (unique) solution? These and other questions are addressed later. For now let us turn to a more concrete and specific model that will prove instructive.

Let $L_2(-\infty, +\infty)$ denote the space of square-integrable functions defined

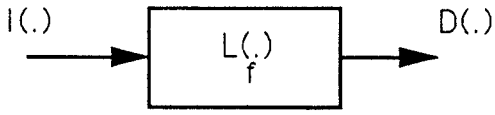


Figure 1.1: Block diagram of a LTI convolutional operator

over the real line. Consider the following *Linear time-invariant* map:

$$\mathcal{L}_f : L_2(-\infty, +\infty) \mapsto L_2(-\infty, +\infty) \tag{1.1.1}$$

Given by

$$[\mathcal{L}_f(I(\cdot))](t) \triangleq (f * I)(t) = \int_{-\infty}^{\infty} I(\tau) \cdot f(t-\tau) d\tau \tag{1.1.2}$$

i.e. $\mathcal{L}_f$ performs convolution with kernel $f(.) \in L_2(-\infty, +\infty)$. This operation is schematically depicted in figure 1.1. Here, D(t) denotes the observed data. In this setting the answer seems straightforward: if the inverse operator $\mathcal{L}_f^{-1}(\cdot)$ exists apply it to the observed data to obtain $I(\cdot)$. Then the problem reduces to the question of existence of the inverse operator. The implications of this are more profound in the frequency domain. We can write:

$$\widehat{D}(\omega) = \widehat{f}(\omega)\widehat{I}(\omega) \tag{1.1.3}$$

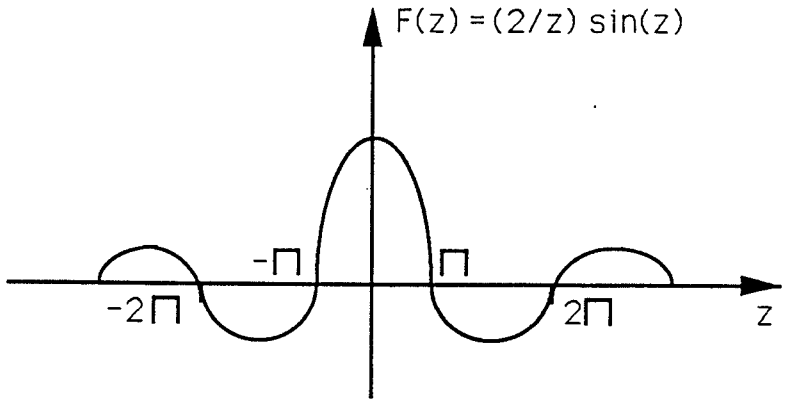


Figure 1.2: Fourier transform of characteristic function over [-1,+1]

where $\wedge$ denotes Fourier Transform, as usual. Then naivelly:

$$\hat{I}(\omega) = \frac{\widehat{D}(\omega)}{\hat{f}(\omega)} \tag{1.1.4}$$

which of course is not well-defined unless $\hat{f}(\omega)$ is nonzero everywhere. Now most physical systems of engineering interest are either band-limited or have a countable infinity of zeros in the frequency domain. Thus $\frac{1}{\hat{f}(\omega)}$ is not well defined (blows up) at a countable infinity of points. Therefore the inverse operator $(\cdot)$ *does not exist*, i.e. the convolution operation *is not invertible.*

As an example let us consider: $f(t) = \chi_{[-1,+1]}(t)$: the characteristic function of the closed interval [-1, +1]. Then $\hat{f}(\omega) = \frac{2}{\omega}\sin\omega$. This is depicted in figure 1.2. Then it is obvious that we loose all information contained in frequencies that are a multiple of π. We therefore conclude that the given problem is ill-posed. A usual approach towards overcoming this ill-posedness has been

via regularization. For the above set-up efforts have focused on the following techniques [14] :

(1) Restrict $I(\cdot)$ to be in the "observable" subspace of $\mathcal{L}(\cdot)_f$. (i.e. $I(\cdot)$ does not contain frequency components located at the zeros of $\widehat{f}(\cdot)$).

(2) Use a priori information about $I(\cdot)$. (i.e. Suppose *we know that* $I(\cdot)$ contains impulses at odd multiples of π and is zero at even multiples of π).

(3) Use some other regularization scheme (*e.g.*: Maximum entropy method).

All these approaches have definite advantages and a common drawback: the imposition of conditions on the input, i.e. the restriction of the input space.

A more natural strategy is to go back to the original problem statement and ask "how much data is enough?" In general the linear model given above is misleading because it assumes that we have data available from one linear operator only (i.e. one sensor). This need not be the case though; suppose we have a family of m sensors $\mathcal{L}_{f_1}(\cdot), \cdots, \mathcal{L}_{f_m}(\cdot)$ with the property that: $\forall\omega\ \exists i$, $i = 1, \cdots, m : \widehat{f}_i(\omega) \neq 0$ Then, clearly, we have moved from an ill-posed problem to a (possibly) overdetermined one. It is a well known fact that the above condition is not enough for the construction of an explicit solution. We shall introduce certain qualifications later. For the moment consider the system of figure 1.3.

Now the natural question that comes up is: what is the minimum possible m and what conditions should the f_i's satisfy so that we can uniquely determine $I(\cdot)$ from the D_i's? An equivalent formulation is: We are looking for a family of

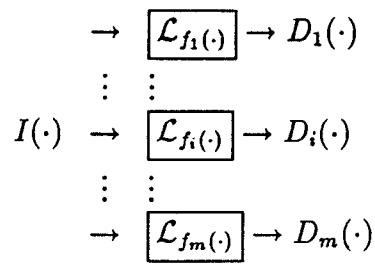


Figure 1.3: Multiple convolutional operators operating on a single input

deconvolvers $\widehat{h}_i(\cdot)$, $i = 1, \cdots, m$ such that:

$$\widehat{D}_1\widehat{h}_1 + \cdots + \widehat{D}_m\widehat{h}_m = \widehat{I} \tag{1.1.5}$$

i.e. since:

$$\widehat{D}_i = \widehat{I} \cdot \widehat{f}_i, \; i = 1, \ldots m \tag{1.1.6}$$

Equation 1.1.5 above is equivalent to:

$$\widehat{f}_1\widehat{h}_1 + \cdots + \widehat{f}_m\widehat{h}_m = 1 \tag{1.1.7}$$

The later equation is known as the Analytic Bezout Equation (ABE). It is a well-known fact that the existence of a family of deconvolvers, $\{\widehat{h}_1, \cdots, \widehat{h}_m\}$ that solves the Bezout Equation is completely equivalent to a coprimeness condition on the part of the f_i's. Let us formalize by introducing some necessary notation and presenting the very important Paley-Wiener Theorem.

Definition 1.1.1 *Let $\mathcal{E}'_{\mathcal{R}^n}$ denote the space of all distributions of compact support defined over $\mathcal{R}^n$. For convenience we drop the index $\mathcal{R}^n$ when the underlying space is obvious from the context.*

Let:

$$\widehat{f}(\omega) \triangleq < f, e^{-i\omega t} > \tag{1.1.8}$$

$$\widehat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{1.1.9}$$

Definition 1.1.2 *Let:* $\widehat{\mathcal{E}}'_{\mathcal{R}^n} \triangleq \{\widehat{f} : f \in \mathcal{E}'_{\mathcal{R}^n}\}$.

Definition 1.1.3 *Let* $\omega = (\omega_1, \cdots, \omega_n) \in \mathcal{C}^n$. *Then:* $Im\omega = (Im\omega_1, \cdots, Im\omega_n)$, *and we define the function* $p(\omega)$ *as follows:*

$$p(\omega) \triangleq |Im\omega| + log(1 + |\omega|) \tag{1.1.10}$$

Definition 1.1.4 *(Paley-Wiener Space). Let* $\widehat{\mathcal{E}}'_{\mathcal{R}^n}$ *denote the space of all functions* $\widehat{f}$ *which are analytic in* $\mathcal{C}^n$ *and have the property that for some constants* A, $B > 0$ *the following inequality holds:*

$$|f(\omega)| \leq Ae^{Bp(\omega)} \tag{1.1.11}$$

Theorem 1.1 *(Paley - Wiener) [17, page 21]. The mapping* $\mathcal{E}'_{\mathcal{R}^n} \mapsto \widehat{\mathcal{E}}'_{\mathcal{R}^n}$ *given by equation 1.1.8 for all* $f \in \mathcal{E}'_{\mathcal{R}^n}$, *is 1-1 and onto the Paley-Wiener Space* $\widehat{\mathcal{E}}'_{\mathcal{R}^n}$. *For convenience we drop the index* $\mathcal{R}^n$ *at all instances where the underlying space is clear from context.*

Theorem 1.2 *[16] A family of functions* $\{\widehat{h}_1, \cdots, \widehat{h}_m\}$ *in* $\widehat{\mathcal{E}}'$ *that solves the Bezout Equation exists iff the family of entire functions* $\{\widehat{f}_1 \cdots, \widehat{f}_m\}$ *in* $\widehat{\mathcal{E}}'$ *is strongly coprime, i.e.* **iff:** $\sum_{j=1}^{m} |\widehat{f}_j(\omega)|^2 \geq e^{-cp(\omega)}, \forall \omega \in C^n$, *for some constant* c. *The function* $p(\omega)$ *is the one in definition 1.1.3.*

Thus the problem of existence of an inverse reduces to the problem of finding suitable distributions $\{f_1, \cdots, f_m\}$ or, alternatively, entire functions $\{\widehat{f}_1, \cdots, \widehat{f}_m\}$ in $\widehat{\mathcal{E}}'$ such that the coprimeness condition is satisfied *for the smallest possible* m. The reason why we are interested in the smallest possible m is that these distributions translate to actual devices, and it is usually desired to achieve reconstruction using the minimum possible number of devices.

Remarks:

1. The Theorem 1.2 above guarantees *existence*; it does not provide a procedure to construct suitable deconvolvers.

2. Uniqueness of the family $\{h_i\}_{i=1}^m$ (or $\{\widehat{h}_i\}_{i=1}^m$) is not implied by 1.2. In fact if a particular family $\{\widehat{h}_i\}_{i=1}^m$ exists that solves the ABE then it is easy to show that other families exist too.

Assuming that the f_i's are indeed strongly coprime one particular solution of the ABE evidently manifests itself. Let:

$$\widehat{h}_i \triangleq \frac{\widehat{f}_i^*}{\sum_{i=1}^m |\widehat{f}_i|^2} \tag{1.1.12}$$

then:

$$\begin{aligned} \widehat{f}_1\widehat{h}_1 + \cdots + \widehat{f}_m\widehat{h}_m &= \frac{1}{\sum_{i=1}^m |\widehat{f}_i|^2}\left(\widehat{f}_1\widehat{f}_1^* + \cdots + \widehat{f}_m\widehat{f}_m^*\right) \\ &= \frac{\sum_{i=1}^m \widehat{f}_i\widehat{f}i^*}{\sum^m{}_{i=1}|\widehat{f}_i|^2} \end{aligned}$$

$$= \frac{\sum_{i=1}^{m} |\widehat{f}_i|^2}{\sum_{i=1}^{m} |\widehat{f}_i|^2} = 1.$$

Hence the ABE is satisfied.

Clearly the $\widehat{h}_i's$ are well defined everywhere (since the coprimeness condition guarantees that $\sum_{i=1}^{m} |\widehat{f}_i|^2$ is bounded away from zero). These are known as Wiener deconvolvers. They have been proven to be optimal with respect to Signal to Noise performance [2]. One undesirable feature of these deconvolvers is that they are not compactly supported. Thus in order to be realized (with finite delay) they have to be truncated, resulting in ripple effects and loss of resolution [30].

1.2 Theoretical Foundations

Another approach has been proposed by C. Berenstein and A. Yger in a series of papers [9,2,3,6,7,8]. The core of their research restricted to our present scope is summarized in a theorem that we will introduce shortly, but before that let us give a number of definitions:

Definition 1.2.5 *Let K be a compact subset of $\mathcal{R}^n$. Define the* **supporting function** *of K as follows:*

$$H_K(\xi) \triangleq max\{x \cdot \xi | x \in K\} \tag{1.2.1}$$

Where $\cdot$ denotes inner product and $\xi \in \mathcal{R}^n$.

Now let $T = \chi_K$. Then for $\omega = \eta + i\xi$ we have that:

$$\widehat{T}(\omega) = \int_K e^{-i\omega t} dt \tag{1.2.2}$$

$$= \int_K e^{t\cdot\xi} e^{-it\cdot\eta} dt \tag{1.2.3}$$

Using the definition of a supporting function 1.2.5 we can obtain the following bound:

$$|\widehat{T}(\omega)| \leq \int_K |e^{t\cdot\xi} e^{-it\cdot\eta}| dt \tag{1.2.4}$$

$$= \int_K |e^{t\cdot\xi}| dt \tag{1.2.5}$$

$$\leq |e^{H_K(\xi)}| \int_K dt \tag{1.2.6}$$

$$= |e^{H_K(\xi)}| m(K) \tag{1.2.7}$$

where $m(K)$ is the volume of the set K.

Definition 1.2.6 *A family of n distributions* $\{f_1, \ldots, f_n\}$ *of compact support in* $\mathcal{R}^n$ *is* **well behaved** *if there exist positive constants A,B,N,K,C and two supporting functions* H_o, H_1, *such that* $0 \leq H_o \leq H_1$, *and such that the common zero set,* $\mathcal{Z}$, *of the functions* $\{\widehat{f}_1, \cdots, \widehat{f}_n\}$ *is almost real i.e.* $\forall \omega \in \mathcal{Z} : |Im\omega| \leq C\, log(2 + |\omega|)$, *and the number of zeros in* $\mathcal{Z}$ *included in an open ball of radius r grows like* r^A *:* $n(\mathcal{Z}, r) = O(r^A)$. *Furthermore, denoting:*

$$|\widehat{f}(z)| \triangleq \left[\Sigma_{i=1}^n |\widehat{f}_i(z)|^2\right]^{1/2} \tag{1.2.8}$$

the following inequality holds:

$$|\widehat{f}(z)| \geq \frac{B d(z, \mathcal{Z})^K e^{H_o(Imz)}}{(1 + |z|)^N} \tag{1.2.9}$$

Where $d(z, \mathcal{Z})$ is the Euclidean distance of the point z from the set $\mathcal{Z}$.

Definition 1.2.7 *A well-behaved family* $\{f_1, \cdots, f_n\}$ *is* **very well behaved** *if there exist constants* $M, C_1 > O$: $\forall \zeta \in \mathcal{Z}$ *we have that:*

$$|J(\zeta)| \triangleq |det[\frac{\partial \widehat{f}_j}{\partial z_i}(\zeta)]_{ij}| \geq C_1(1 + |\zeta|)^{-M} \tag{1.2.10}$$

This last condition guarantees that the points in $\mathcal{Z}$ (the set of common zeros of the family $\{\widehat{f}_1, \cdots, \widehat{f}_n\}$) are simple and "adequately spaced" i.e. they are not accumulated in any neighborhood around any common zero. Now we give a restricted version of a Theorem of C. Berenstein and A. Yger that is sufficient for our purposes and is simpler in the sense that it does not involve cumbersome notation, thus helping to keep things in perspective: Suppose $n = 2$ (i.e. we have distributions defined over $\mathcal{R}^2$) and $m = n+1 = 3$ (i.e. we use 3 convolvers). Then:

Theorem 1.3 *[9] Let* $\{f_1, f_2, f_3\}$ *be a strongly coprime family of compactly supported distributions over* $\mathcal{R}^2$. *Suppose that the subfamily* $\{f_1, f_2\}$ *is very well behaved. Suppose* f_3 *is the "best" kernel in the sense that it has the smallest support of all three. Let* H_0, H_1 *be as in definition 1.2.6 for the subfamily* $\{f_1, f_2\}$. *Define*

$$H_2(\theta) = max_{1 \leq j \leq 3} max\{x \cdot \theta \ : x \in suppf_j\}, \theta \in \mathcal{R}^2$$

, where · denotes inner product, and suppose $H_2 \leq 2H_1$. *Furthermore suppose* $\exists\, r_o > 0 : r_o|\theta| \leq 4H_o(\theta) - 2H_1(\theta) - H_2(\theta)$ *(these conditions control the support of* f_3 *vs. the support of* f_1, f_2*). Then for any* $u \in C_o^\infty(\mathcal{R}^2)$ *compactly supported and with "small" support:*

supp $u \subseteq \{x \in \mathcal{R}^n : |x| \leq r_0$*, one can write:*

$$\widehat{u}(z) = \sum_{\zeta \in \mathcal{Z}} \frac{\widehat{u}(\zeta)}{J(\zeta)\widehat{f}_3(\zeta)} \begin{vmatrix} g_1^1(z,\zeta) & g_1^2(z,\zeta) & g_1^3(z,\zeta) \\ g_2^1(z,\zeta) & g_2^2(z,\zeta) & g_2^3(z,\zeta) \\ \widehat{f}_1(z) & \widehat{f}_2(z) & \widehat{f}_3(z) \end{vmatrix} \tag{1.2.11}$$

where: $z = (z_1, z_2)$, $\zeta = (\zeta_1, \zeta_2)$, *both in* C^2, *and:*

$$g_1^i(z,\zeta) \triangleq \frac{\widehat{f}_i(z_1,\zeta_2) - \widehat{f}_i(\zeta_1,\zeta_2)}{z_1 - \zeta_1} \tag{1.2.12}$$

$$g_2^i(z,\zeta) \triangleq \frac{\widehat{f}_i(z_1,z_2) - \widehat{f}_i(z_1,\zeta_2)}{z_2 - \zeta_2} \tag{1.2.13}$$

and: $J(\zeta) = det(M(z))|_{z=\zeta}$, *where the Jacobian matrix M(z) is defined as:*

$$M(z) \triangleq \begin{bmatrix} \frac{\partial \widehat{f}_1}{\partial z_1} & \frac{\partial \widehat{f}_2}{\partial z_1} \\ \frac{\partial \widehat{f}_1}{\partial z_2} & \frac{\partial \widehat{f}_2}{\partial z_2} \end{bmatrix} \tag{1.2.14}$$

and:

$$\mathcal{Z} = \{z \in C^2 : \widehat{f}_1(z) = \widehat{f}_2(z) = 0\} \tag{1.2.15}$$

Remarks: Obviously the result extends to higher-dimensional spaces, see [9]. Now, clearly, equation 1.2.11 is an interpolation formula since it constructs

the entire function $\widehat{u}(\cdot)$ based on distinct point values of $\widehat{u}(\cdot)$ at a discrete set of isolated points in the plane. Now equation 1.2.11 is obtained using a *limiting argument.* This fact has implications that will be of profound interest later on. For the moment let us show why equation 1.2.11 is a very important result, since it may not be immediately apparent. Equation 1.2.11 can be rewritten in the form:

$$\widehat{u}(z) = \widehat{h_1}(z)\widehat{f_1}(z) + \widehat{h_2}(z)\widehat{f_2}(z) + \widehat{h_3}(z)\widehat{f_3}(z) \tag{1.2.16}$$

And since $u(x)$ is of sufficiently small support (we can certainly shrink the support of u below r_o) then $\widehat{u}(z) \cong 1$ and:

$$1 \cong \widehat{h_1}(z)\widehat{f_1}(z) + \widehat{h_2}(z)\widehat{f_2}(z) + \widehat{h_3}(z)\widehat{f_3}(z)$$

i.e. $\{\widehat{h}_1(z), \widehat{h}_2(z), \widehat{h}_3\}$ give an approximate solution to the ABE! In principle we can set $u = \delta$ thus getting $\widehat{u}(\cdot) = 1$ and obtain *exact* deconvolvers. Notice that $\widehat{u}(\cdot)$ *is not* compactly supported (because u *is* compactly supported). Observe that since a realistic computation will truncate the sum over $\mathcal{Z}$ in 1.2.11 to make it finite, it will force $\widehat{u}(\cdot)$ to be compactly supported (because setting $\widehat{u}(\zeta) = 0$ for $\|\zeta\| > \zeta_0$ is equivalent to truncating the sum in equation 1.2.11. Therefore, u will not be compactly supported, thus we will be violating the corresponding assumption of therorem 1.3. We will discuss this point in detail later on. For the moment let us turn to a specific example.

CHAPTER
TWO

Model Problem

2.1 Formulation and derived expressions

Let us consider the following family of convolution kernels:

$$f_1(t_1,t_2) = \chi_{[-\sqrt{3},\sqrt{3}]\times[-\sqrt{3},\sqrt{3}]}(t_1,t_2) \tag{2.1.1}$$

$$f_2(t_1,t_2) = \chi_{[-\sqrt{2},\sqrt{2}]\times[-\sqrt{2},\sqrt{2}]}(t_1,t_2) \tag{2.1.2}$$

$$f_3(t_1,t_2) = \chi_{[-1,1]\times[-1,1]}(t_1,t_2) \tag{2.1.3}$$

with:

$$\widehat{f}_1(z_1,z_2) = \frac{4}{z_1 z_2}\sin(\sqrt{3}z_1)\sin(\sqrt{3}z_2) \tag{2.1.4}$$

$$\widehat{f}_2(z_1,z_2) = \frac{4}{z_1 z_2}\sin(\sqrt{2}z_1)\sin(\sqrt{2}z_2) \tag{2.1.5}$$

$$\widehat{f}_3(z_1,z_2) = \frac{4}{z_1 z_2}\sin(z_1)\sin(z_2) \tag{2.1.6}$$

Then it is easy to verify that $\{f_1, f_2, f_3\}$ satisfy all conditions of theorem 1.3.

Here:

$$\mathcal{Z} = \left\{ \left(\frac{j\pi}{\sqrt{3}}, \frac{k\pi}{\sqrt{2}} \right) \ , \ k,j = \pm 1, \pm 2, \cdots \right\} \cup \left\{ \left(\frac{j\pi}{\sqrt{2}}, \frac{k\pi}{\sqrt{3}} \right) \ , \ k,j = \pm 1, \pm 2, \cdots \right\} \tag{2.1.7}$$

$$J(\zeta) = \begin{vmatrix} \frac{4\sin(\sqrt{3}\zeta_2)}{\zeta_1^2\zeta_2} r_1(\zeta_1) & \frac{4\sin(\sqrt{2}\zeta_2)}{\zeta_1^2\zeta_2} r_2(\zeta_1) \\ \\ \frac{4\sin(\sqrt{3}\zeta_1)}{\zeta_2^2\zeta_1} r_1(\zeta_2) & \frac{4\sin(\sqrt{2}\zeta_1)}{\zeta_2^2\zeta_1} r_2(\zeta_2) \end{vmatrix} \tag{2.1.8}$$

with:

$$r_1(x) \triangleq \sqrt{3}x\cos(\sqrt{3}x) - \sin(\sqrt{3}x) \tag{2.1.9}$$

$$r_2(x) \triangleq \sqrt{2}x\cos(\sqrt{2}x) - \sin(\sqrt{2}x) \tag{2.1.10}$$

and after a series of transformations we can rewrite 1.2.11 in the form 1.2.16 and then read out the expressions for $\hat{h}_1$, $\hat{h}_2$, $\hat{h}_3$. These are given by the infinite sums:

$$\hat{h}_i(z_1, z_2) = \sum_{\zeta \in \mathcal{Z}} \frac{\hat{u}(\zeta)}{J(\zeta)\hat{f}_3(\zeta)} \cdot \frac{C_i(z,\zeta)}{(z_1 - \zeta_1)(z_2 - \zeta_2)} \tag{2.1.11}$$

with:

$$\begin{aligned} C_1(z,\zeta) \ \triangleq \ & \hat{f}_2(z_1,\zeta_2)\left[\hat{f}_3(z_1,z_2) - \hat{f}_3(\zeta_1,\zeta_2)\right] - \\ & \hat{f}_2(z_1,z_2)\left[\hat{f}_3(z_1,\zeta_2) - \hat{f}_3(\zeta_1,\zeta_2)\right] \end{aligned} \tag{2.1.12}$$

$$C_2(z,\zeta) \triangleq \widehat{f}_1(z_1,z_2)\left[\widehat{f}_3(z_1,\zeta_2)-\widehat{f}_3(\zeta_1,\zeta_2)\right]- \widehat{f}_1(z_1,\zeta_2)\left[\widehat{f}_3(\zeta_1,\zeta_2)-\widehat{f}_3(z_1,z_2)\right] \quad (2.1.13)$$

$$C_3(z,\zeta) \triangleq \widehat{f}_1(z_1,\zeta_2)\widehat{f}_2(z_1,z_2)-\widehat{f}_1(z_1,z_2)\widehat{f}_2(z_1,\zeta_2) \quad (2.1.14)$$

Definition 2.1.8 *The function $\mathcal{F}(\cdot,\cdot)$ is $\frac{\pi}{2}$ rotation-invariant ($\frac{\pi}{2}-ri$) iff:*

$\mathcal{F}(z_1,z_2)=\mathcal{F}(z_2,z_1)$, $\forall z_1,z_2\in\mathcal{C}^2$.

Notice that $\widehat{f}_1$, $\widehat{f}_2$, $\widehat{f}_3$ are all $\frac{\pi}{2}$-ri. Nevertheless the C_i's *are not*; for example $C_1(z_1,z_2)\neq C_1(z_2,z_1)$. Hence every *finite* approximation to $\widehat{h}_1$ is bound to exhibit asymmetry i.e. it will not be $\frac{\pi}{2}$-ri. In the limiting case we expect this asymmetry to die out because of cancellations. Similar remarks hold for $\widehat{h}_2,\widehat{h}_3$. This fact will prove annoying for applications. A final remark is in place here: The distributions $h_i = F.T.^{-1}\{\widehat{h}_i\}$ $i=1,2,3$, where the $\widehat{h}_i's$ are those obtained using theorem 1.3 , are *compactly supported* (in fact of support comparable to that of the kernels f_i) . Thus, once obtained, they are easily realized *exactly* with finite delay (exactly refers to the fact that there is no need for truncation of their duration; sampling and finite word length Arithmetic errors can be controlled to meet the design goals [30]. This property of these deconvolvers is their most desirable feature when compared to Wiener deconvolvers.

The rest of this chapter is organized as follows: We first examine the one dimensional case which highlights much of what is involved while keeping notational and computational burden with a minimum; it also helps to strengthen

intuition and pinpoint potential trouble spots. Next we consider the exploitation of symmetries to help reduce the computational complexity of the problem and improve error performance; then we discuss two alternative ways to compute the deconvolution kernels (in the time domain or in the frequency domain). Finally, we investigate proper practical frequency windowing and energy spreading techniques.

2.2 One-Dimensional Case

Here we have $n = 1$, $m = n + 1 = 2$. (Everything defined over $\mathcal{R}$, use 2 convolvers) and formula (1.2.3) of Theorem (1.2.1) reduces to:

$$\widehat{u}(z) = \sum_{\zeta \in \mathcal{Z}} \frac{\widehat{u}(\zeta)}{J(\zeta)\widehat{f}_2(\zeta)} \begin{vmatrix} g_1^1(z,\zeta) & g_1^2(z,\zeta) \\ \widehat{f}_1(z) & \widehat{f}_2(z) \end{vmatrix} \tag{2.2.1}$$

with: $\mathcal{Z}$ = set of zeros of $\widehat{f}_1(z)$, and:

$$g_1^1(z,\zeta) \triangleq \frac{\widehat{f}_1(z) - \widehat{f}_1(\zeta)}{z - \zeta} \tag{2.2.2}$$

$$g_1^2(z,\zeta) \triangleq \frac{\widehat{f}_2(z) - \widehat{f}_2(\zeta)}{z - \zeta} \tag{2.2.3}$$

So that:

$$\begin{vmatrix} g_1^1(z,\zeta) & g_1^2(z,\zeta) \\ \widehat{f}_1(z) & \widehat{f}_2(z) \end{vmatrix} = \begin{vmatrix} \frac{\widehat{f}_1(z)-\widehat{f}_1(\zeta)}{z-\zeta} & \frac{\widehat{f}_2(z)-\widehat{f}_2(\zeta)}{z-\zeta} \\ \widehat{f}_1(z) & \widehat{f}_2(z) \end{vmatrix}$$

$$= \frac{\widehat{f}_1(z)\widehat{f}_2(\zeta) - \widehat{f}_1(\zeta)\widehat{f}_2(z)}{z - \zeta} \tag{2.2.4}$$

Also observe that here $J(\zeta) = \left.\frac{d\hat{f}_1(z)}{dz}\right|_{z=\zeta}$ so that we can expand $\hat{u}(z)$ as follows:

$$\hat{u}(z) = \hat{h}_1(z) \cdot \hat{f}_1(z) + \hat{h}_2(z) \cdot \hat{f}_2(z) \tag{2.2.5}$$

With $\hat{h}_1, \hat{h}_2$ given by:

$$\hat{h}_1(z) = \sum_{\zeta \in \mathcal{Z}} \frac{\hat{u}(\zeta)}{J(\zeta)(z-\zeta)} \tag{2.2.6}$$

$$\hat{h}_2(z) = -\sum_{\zeta \in \mathcal{Z}} \frac{\hat{u}(\zeta)}{J(\zeta)(z-\zeta)} \cdot \frac{\hat{f}_1(\zeta)}{\hat{f}_2(\zeta)} \tag{2.2.7}$$

Now let $u = \delta, \mapsto \hat{u}(\zeta) = 1, \forall \zeta$. Then:

$$\hat{h}_1(z) = \sum_{\zeta \in \mathcal{Z}} \frac{1}{J(\zeta)(z-\zeta)} \tag{2.2.8}$$

$$\hat{h}_2(z) = -\sum_{\zeta \in \mathcal{Z}} \frac{1}{J(\zeta)(z-\zeta)} \cdot \frac{\hat{f}_1(\zeta)}{\hat{f}_2(\zeta)} \tag{2.2.9}$$

Now notice that $\mathcal{Z}$ is the set of zeros of $\hat{f}_1$. Thus $\hat{f}_1(\zeta) = 0$, $\forall \zeta \in \mathcal{Z}$ and: $J(\zeta) = \left.\frac{d\hat{f}_1(z)}{dz}\right|_{z=\zeta} \neq 0, \forall \zeta \in \mathcal{Z}$, since $\hat{f}_1$ is very well behaved. Also: $\hat{f}_2(\zeta) \neq 0, \forall \zeta \in \mathcal{Z}$, by strong coprimeness of the family $\{\hat{f}_1, \hat{f}_2\}$. Thus, given z, the only term that possibly survives in equation 2.2.9 is the one for $\zeta = z$. Therefore $\hat{h}_2(z)$ is identically equal to 0 everywhere, except possibly at points $z \in \mathcal{Z}$. Consider the factor:

$$\left.\frac{\hat{f}_1(\zeta)}{(z-\zeta)}\right|_{\zeta=z} = \frac{\left.\frac{d\hat{f}_1(\zeta)}{d\zeta}\right|_{\zeta=z}}{\left.\frac{d(z-\zeta)}{d\zeta}\right|_{\zeta=z}}$$

$$= \frac{J(z)}{-1} = -J(z) \tag{2.2.10}$$

Hence:

$$\widehat{h}_2(z) = \begin{cases} \frac{1}{\widehat{f}_2(z)} & , for z \in \mathcal{Z} \\ \\ 0 & , elsewhere \end{cases} \tag{2.2.11}$$

And:

$$\widehat{h}_1(z) = \sum_{\zeta \in \mathcal{Z}} \frac{1}{\frac{d\widehat{f}_1(\omega)}{d\omega}\Big|_{\omega=\zeta} \cdot (z-\zeta)} \tag{2.2.12}$$

Some remarks are in order: These results are compatible with one's intuitive feeling of how to attack the problem at hand: Use one of the two convolvers "almost surely" (i.e. everywhere except at a countable set of points where its F.T. vanishes) and resort to the other one for these very points where the F.T. of the former one vanishes. By strong coprimeness, $\widehat{f}_2(\cdot)$ will be nonzero at all these points and we are done. Observe that the Fourier transform of the second deconvolution kernel has a support set of measure zero, i.e. it is defined to be nonzero only at a countable set of isolated points. Hence we will need to approximate it with a kernel whose Fourier transform has a support set of nonzero measure.

A somewhat suprizing fact is that the Fourier transform of the first deconvolution kernel is not equal to the intuitively obvious choice $\frac{1}{\widehat{f}_1(z)}$. Therefore we need to investigate how close and in what sense the Fourier transform given by equation 2.2.12 approximates $\frac{1}{\widehat{f}_1(z)}$. Let us elaborate on this. Consider equa-

tion 2.2.12 . Under our assumptions the derivative $\frac{d\widehat{f}_1(\omega)}{d\omega}$ exists and it is nonzero $\forall \zeta \in \mathcal{Z}$, since f_1 is very well behaved (for if this derivative were zero at some point $\zeta \in \mathcal{Z}$ then about this point the first order contribution of infinitessimaly small displacements in terms of ω would be zero , thus the set $\mathcal{Z}$ would include a subset of non-isolated points). Furthermore the derivative is a well behaved function. Then, clearly, the contribution of the summation term corresponding to $\zeta = p$ diminishes as the difference $|z - p|$ goes to zero. Therefore, under some more (quite general) technical assumptions on the rate of $\widehat{f}_1$ the dominant term in 2.2.12 is the one corresponding to the nullpoint ζ^* that satisfies:

$$\zeta^* = argmin_{\zeta \in \mathcal{Z}} ||z - \zeta|| \tag{2.2.13}$$

where $||\cdot||$ denotes the Euclidean norm, i.e. absolute value for the one dimensional case. Hence we can write:

$$\widehat{h}_1(z) \cong \frac{1}{\frac{d\widehat{f}_1(\omega)}{d\omega}\Big|_{\omega=\zeta^*}(z - \zeta^*)} = \frac{1}{D(z)} \tag{2.2.14}$$

Where D(z) is a first order Taylor series approximation to $\widehat{f}_1(z)$ about the point ζ^*. Therefore for points z sufficiently close to some nullpoint ζ^*, $\widehat{h}_1(z)$ essentially approximates one over the Taylor series expansion of $\widehat{f}_1(z)$ about the point ζ^*, which in turn approximates $\frac{1}{\widehat{f}_1(z)}$, assuming that the zeros form a sufficiently dense grid. Then it becomes clear that a brute-force truncation of the nullset $\mathcal{Z}$ will severely distort $\widehat{h}_1(z)$ at all points outside the truncated nullset

grid. Hence *the nullset grid must extend at least up to the desired bandwidth* in order to get an approximation that is good at least up to the first order.

In the general case formula 2.2.12 estimates $\frac{1}{\widehat{f}_1(z)}$ as follows: Every point $\zeta \in \mathcal{Z}$ makes a linear estimate of $\widehat{f}_1(z)$ by using a Taylor series extrapolation of the first order. Then this is inverted and an estimate of $\frac{1}{\widehat{f}_1(z)}$ is formed. Finally all estimates are summed up to produce $\widehat{h}_1(z)$. If the nullset grid is dense enough and *regular* (as is usually the case) then points ζ located away from z estimate terms corresponding to higher order derivatives.

The above discussion gives the reader an intuitive feeling about how finite approximations work and points out some interesting aspects of the solution under consideration. A rigorous proof of the fact that this solution indeed converges to $\frac{1}{\widehat{f}_1(z)}$ for points z not in $\mathcal{Z}$ will not be given here, as it is a direct consequence of Theorem 1.3. The persistent reader is reffered to the original paper [9], by C. Berenstein and A. Yger.

We close this section with a discussion of how we go along computing expressions for h_1, h_2. We have two options: either work in the frequency domain, calculate approximations to 2.2.12 , test convergence (in the Cauchy sense) and then use an $IDFT$ algorithm to get approximations to the inverted spectrums or we can try to invert 2.2.12 analytically. In the former case, care should be exercised in sampling the spectra so that all points $z \in \mathcal{Z}$ are in the sampling grid, otherwise $\widehat{f}_1(z)$ is completely lost. A study of the errors involved in this discretization and approximate inversion procedure can be found in [1].

In the later case we are faced with the problem of how to go about finding an approximation of $\widehat{f}_1(z)$ that has a support set of nonzero measure; an obvious choice is:

$$\widetilde{h}_2(z) = \widehat{\xi} * \widehat{\theta}(z) \tag{2.2.15}$$

with:

$$\widehat{\xi}(z) \triangleq \frac{1}{\widehat{f}_2(z)} \cdot \sum_{\zeta \in Z} \delta(z - \zeta) \tag{2.2.16}$$

where * denotes convolution, $\delta(\cdot)$ is the delta mass, and $\widehat{\theta}(z)$ is a frequency domain mollifier of sufficiently compact support, smooth, nonnegative, and integrates to unity. Now let:

$$\widehat{\phi}(z) \triangleq \frac{1}{\widehat{f}_2(z)} \tag{2.2.17}$$

and:

$$\widehat{p}(z) \triangleq \sum_{\zeta \in Z} \delta(z - \zeta) \tag{2.2.18}$$

Then:

$$\xi(t) = (\phi * p)(t) \tag{2.2.19}$$

and one reasonable approximation to equation 2.2.11 that has support set of nonzero measure in frequency is given by:

$$\widetilde{h}_2(t) = \xi(t) \cdot \theta(t) \tag{2.2.20}$$

For $\widehat{h}_1(z)$ things are quite straightforward:

$$h_1(t) \;=\; FT^{-1}\left\{\widehat{h}_1(z)\right\}$$

$$= FT^{-1}\left\{\sum_{\zeta\in Z}\frac{1}{J(\zeta)(z-\zeta)}\right\}$$

$$= \sum_{\zeta\in Z}\frac{1}{J(\zeta)}FT^{-1}\left\{\frac{1}{z-\zeta}\right\} \qquad (2.2.21)$$

Recall that:

$$e^{j\zeta t}\cdot\frac{j}{2}\cdot sgn(t) \overset{FT}{\longleftrightarrow} \frac{1}{z-\zeta}$$

where sgn(t) denotes the sign function,

$$sgn(t) = \begin{cases} 1 & ,t>0 \\ 0 & ,t=0 \\ -1 & ,t<0 \end{cases}$$

Therefore:

$$h_1(t) = \sum_{\zeta\in Z}\frac{1}{J(\zeta)}\cdot e^{j\zeta t}\cdot\frac{j}{2}\cdot sgn(t)$$

$$= \frac{j}{2}sgn(t)\sum_{\zeta\in}\frac{e^{j\zeta t}}{J(\zeta)}$$

$$= \frac{j}{2}sgn(t)\sum_{\zeta\in Z}\frac{\cos(\zeta t)+j\sin(\zeta t)}{J(\zeta)}$$

$$= -\frac{sgn(t)}{2}\sum_{\zeta\in Z}\frac{\sin(\zeta t)}{J(\zeta)}+\frac{j\cdot sgn(t)}{2}\sum_{\zeta\in Z}\frac{\cos(\zeta t)}{J(\zeta)} \qquad (2.2.22)$$

Again this expression involves infinite sums and has to be approximated. The advantage of this approach is that there is no need for approximate inversion,

thus a class of errors is eliminated. The problem is that the analytic inversion of (2.2.7) is difficult in general, even if simple analytic expressions for the convolvers are available. For the 2-Dimensional case it becomes almost impossible for any nontrivial set of convolvers. Recent developments in the field of symbolic manipulation packages (such as MACSYMA and MATHEMATICA) may provide a valuable helping hand but at the present time a lot of the work has to be done by hand, mainly because even if expressions like the one above are machine-deduced, numerical computation requires careful reshuffling of terms and exploitation of symmetries to reduce the complexity as much as possible. As an example consider the one-dimensional case and the following specific set of convolvers:

$$f_1(t) = \chi_{[-\sqrt{2},\sqrt{2}]}(t) \overset{FT}{\longleftrightarrow} \widehat{f}_1(z) = \frac{2}{z} \cdot \sin(\sqrt{2}z) \tag{2.2.23}$$

$$f_2(t) = \chi_{[-1,1]}(t) \overset{FT}{\longleftrightarrow} \widehat{f}_1(z) = \frac{2}{z} \cdot \sin(z) \tag{2.2.24}$$

So frequency-domain computation requires *real* arithmetic, whereas time-domain computation requires *complex* arithmetic. Therefore, in most cases, frequency-domain computations are preferable, especially if filtering is going to be performed in frequency. We now return to the 2-Dimensional case and our model problem.

2.3 Exploitation of symmetries in the frequency domain

We are interested in ways to exploit inherent problem symmetry in order to reduce the computational complexity and improve arithmetic error performance. A lot of problems of practical engineering interest exhibit strong symmetries or can be approximated as such. We shall show how one can accomplish such savings, using our model problem. Consider the result obtained in equation 2.1.11 , which we rewrite here making the number of frequency variables explicit:

$$\widehat{h}_1(z_1,z_2) = \sum_{(\zeta_1,\zeta_2)\in\mathcal{Z}} \frac{\widehat{u}(\zeta_1,\zeta_2)}{J(\zeta_1,\zeta_2)\cdot\widehat{f}_3(\zeta_1,\zeta_2)} \cdot \frac{C_1(z_1,z_2,\zeta_1,\zeta_2)}{(z_1-\zeta_1)\cdot(z_2-\zeta_2)} \tag{2.3.1}$$

Assume that $\widehat{u}(\zeta_1,\zeta_2)$ is an even function of both its arguments. We will use the following notation to denote this: $\widehat{u}(\zeta_1,\zeta_2) \sim e(\zeta_1,\zeta_2)$. Similarly odd functions are denoted by $\sim o(\cdot)$. We are interested in symmetries in terms of (ζ_1,ζ_2) of the summation term in equation 2.3.1 .

Define:

$$W(z_1,z_2,\zeta_1,\zeta_2) \triangleq \frac{\widehat{u}(\zeta_1,\zeta_2)\cdot C_1(z_1,z_2,\zeta_1,\zeta_2)}{J(\zeta_1,\zeta_2)\cdot\widehat{f}_3(\zeta_1,\zeta_2)} \tag{2.3.2}$$

Next recall that:

$$\mathcal{Z} = \left\{ \left(\frac{j\pi}{\sqrt{3}},\frac{k\pi}{\sqrt{2}}\right) \ , \ k,j = \pm1,\pm2,\cdots \right\} \cup \left\{ \left(\frac{j\pi}{\sqrt{2}},\frac{k\pi}{\sqrt{3}}\right) \ , \ k,j = \pm1,\pm2,\cdots \right\} \tag{2.3.3}$$

and observe that:

$$(n_1,n_2)\in\mathcal{Z} \longrightarrow (-n_1,n_2),\ (n_1,-n_2),\ (-n_1,-n_2)\in\mathcal{Z}$$

Now, for z fixed: $C_1(z_1, z_2, \zeta_1, \zeta_2) \sim e(\zeta_1, \zeta_2)$ and $J(\zeta_1, \zeta_2) \sim o(\zeta_1)o(\zeta_2)$. Furthermore: $\widehat{f}_3(\zeta_1, \zeta_2) \sim e(\zeta_1)e(\zeta_2)$. Thus , for fixed z:

$W(z_1, z_2, \zeta_1, \zeta_2) \sim o(\zeta_1)o(\zeta_2)$.

Therefore, for fixed z, the pointwise contribution of the four zeros:

$$(n_1, n_2),\ (-n_1, n_2),\ (n_1, -n_2),\ (-n_1, -n_2)$$

can be combined as follows:

$$W(z_1, z_2, n_1, n_2) \cdot \left[\frac{1}{(z_1 - n_1)\cdot(z_2 - n_2)} - \frac{1}{(z_1 + n_1)\cdot(z_2 - n_2)}\right] -$$

$$W(z_1, z_2, n_1, n_2) \cdot \left[\frac{1}{(z_1 - n_1)\cdot(z_2 + n_2)} - \frac{1}{(z_1 + n_1)\cdot(z_2 + n_2)}\right] =$$

$$= W(z_1, z_2, n_1, n_2) \cdot \frac{4 \cdot n_1 \cdot n_2}{(z_1^2 - n_1^2)\cdot(z_2^2 - n_2^2)}$$

Therefore we can combine the four individual contributions into a single term. Furthermore, notice that since for fixed z, $C_1(z_1, z_2, \zeta_1, \zeta_2) \sim e(z_1, z_2)$ this implies that $W(z_1, z_2, \zeta_1, \zeta_2) \sim e(z_1, z_2)$, thus the combined term above is $\sim e(z_1, z_2)$. Hence $\widehat{h}_1(z_1, z_2) \sim e(z_1, z_2)$, as the sum of even functions of both z_1 and z_2. Therefore we only need to compute the upper right quadrant of the Fourier transform plane. Notice that we have achieved combined savings (in the number of terms that need to be computed by any digital approximation scheme) of 1/16 ! This can make an enormous difference in any practical computation.

2.4 Actual Computation of Approximate Deconvolution Kernels

As we briefly mentioned before we basically have two ways to go about computing discrete approximations to the deconvolution kernels: either perform a frequency-domain computation (which seems much more straightforward) and then invert the resulting spectra using an IDFT procedure, or invert the formulas analytically (symbolically) and directly compute the kernels in the time domain.

It has been remarked that time domain computation can be overwhelming! For simple problems, like the model we're considering, it is within one's limits, although it requires a considerable number of manipulations. Basically one is trying to invert the summation term in the expressions for the F.T. of the deconvolvers. This has been done and the results are given in the next few pages. It is assumed that $\widehat{u}(\zeta_1, \zeta_2) \equiv 1$, but any other $\widehat{u}$ can be accommodated by replacing the unit in the nominator of the summation term with $\widehat{u}(\zeta_1, \zeta_2)$. Also notice that only the formulas for $h_1(t_1, t_2)$ and $h_3(t_1, t_2)$ are given, since (due to similarity) the expression for $h_2(t_1, t_2)$ can be easily obtained by replacing $\sqrt{2}$ by $\sqrt{3}$ everywhere the former appears in the formulas for $h_1(t_1, t_2)$. Notice that we verify claims made earlier for the support of the deconvolution kernels (it is bounded above independently of $\widehat{u}$). The results follow:

$$h_1(r_1, r_2) = \sum_{\zeta \in Z} \frac{1}{J(\zeta) \cdot \hat{f}_3(\zeta)} \cdot \eta_\zeta^{(1)}(r_1, r_2) \tag{2.4.1}$$

with:

$$\eta_\zeta^{(1)}(r_1,r_2)=\begin{cases}
-\sqrt{2}<r_2\le -1: & -1-\sqrt{2}<r_1\le -\sqrt{2}: & -R_2^-\cdot S_2\\
& -\sqrt{2}<r_1\le 1-\sqrt{2}: & -R_2^-\cdot S_3\\
& 1-\sqrt{2}<r_1\le \sqrt{2}-1: & -R_2^-\cdot S_4\\
& \sqrt{2}-1<r_1\le \sqrt{2}: & -R_2^-\cdot S_5\\
& \sqrt{2}<r_1\le 1+\sqrt{2}: & -R_2^-\cdot S_6\\
-1<r_2\le 0: & -1-\sqrt{2}<r_1\le -\sqrt{2}: & P_2^-\cdot Q_2-R_2^-\cdot S_2\\
& -\sqrt{2}<r_1\le 1-\sqrt{2}: & P_2^-\cdot Q_2-R_2^-\cdot S_3\\
& 1-\sqrt{2}<r_1\le \sqrt{2}-1: & P_2^-\cdot Q_3-R_2^-\cdot S_4\\
& \sqrt{2}-1<r_1\le \sqrt{2}: & P_2^-\cdot Q_4-R_2^-\cdot S_5\\
& \sqrt{2}<r_1\le 1+\sqrt{2}: & P_2^-\cdot Q_4-R_2^-\cdot S_6\\
0<r_2\le 1: & -1-\sqrt{2}<r_1\le -\sqrt{2}: & P_2^+\cdot Q_2-R_2^+\cdot S_2\\
& -\sqrt{2}<r_1\le 1-\sqrt{2}: & P_2^+\cdot Q_2-R_2^+\cdot S_3\\
& 1-\sqrt{2}<r_1\le \sqrt{2}-1: & P_2^+\cdot Q_3-R_2^+\cdot S_4\\
& \sqrt{2}-1<r_1\le \sqrt{2}: & P_2^+\cdot Q_4-R_2^+\cdot S_5\\
& \sqrt{2}<r_1\le 1+\sqrt{2}: & P_2^+\cdot Q_4-R_2^+\cdot S_6\\
1<r_2\le \sqrt{2}: & -1-\sqrt{2}<r_1\le -\sqrt{2}: & -R_2^+\cdot S_2\\
& -\sqrt{2}<r_1\le 1-\sqrt{2}: & -R_2^+\cdot S_3\\
& 1-\sqrt{2}<r_1\le \sqrt{2}-1: & -R_2^+\cdot S_4\\
& \sqrt{2}-1<r_1\le \sqrt{2}: & -R_2^+\cdot S_5\\
& \sqrt{2}<r_1\le 1+\sqrt{2}: & -R_2^+\cdot S_6\\
\textit{elsewhere}: & 0 &
\end{cases}\tag{2.4.2}$$

With:

$$P_2^- = \left\{\frac{1}{\zeta_2}\left[\cos(\zeta_2(r_2+1))-1\right]\right\} + j\left\{\frac{1}{\zeta_2}\cdot\sin(\zeta_2(r_2+1))\right\}$$

$$P_2^+ = \left\{\frac{1}{\zeta_2}\left[\cos(\zeta_2(r_2-1))-1\right]\right\} + j\left\{\frac{1}{\zeta_2}\sin(\zeta_2(r_2-1))\right\}$$

$$Q_2 = \left\{\frac{2\sin(\sqrt{2}\zeta_2)}{\zeta_1\zeta_2}\left[\frac{\cos(\sqrt{2}\zeta_1)\sin(\zeta_1(r_1+1))}{\zeta_1} - 1 - \sqrt{2} - r_1\right]\right\} +$$

$$+ \; j\left\{\frac{2\sin(\sqrt{2}\zeta_2)\cos(\sqrt{2}\zeta_1)}{\zeta_1^2\zeta_2}\left[\cos(\sqrt{2}\zeta_1) - \cos(\zeta_1(r_1+1))\right]\right\}$$

$$Q_3 = \left\{\frac{4\sin(\sqrt{2}\zeta_2)}{\zeta_1\zeta_2}\left[\frac{\sin(\zeta_1)\cos(\sqrt{2}\zeta_1)\cos(\zeta_1 r_1)}{\zeta_1} - 1\right]\right\} +$$

$$+ \; j\left\{\frac{4\sin(\sqrt{2}\zeta_2)\sin(\zeta_1)\cos(\sqrt{2}\zeta_1)\sin(\zeta_1 r_1)}{\zeta_1^2\zeta_2}\right\}$$

$$Q_4 = \left\{\frac{2\sin(\sqrt{2}\zeta_2)}{\zeta_1\zeta_2}\left[\frac{\cos(\sqrt{2}\zeta_1)\left(\sin(\zeta_1\sqrt{2}) - \sin(\zeta_1(r_1-1))\right)}{\zeta_1}\right.\right.$$

$$\left.\left. - \; 1 - \sqrt{2} + r_1\right]\right\} +$$

$$+ \; j\left\{\frac{2\sin(\sqrt{2}\zeta_2)\cos(\sqrt{2}\zeta_1)}{\zeta_1^2\zeta_2}\cdot\left[\cos(\zeta_1(r_1-1)) - \cos(\zeta_1\sqrt{2})\right]\right\}$$

$$R_2^- = \left\{ \frac{1}{\zeta_2} \cos(\zeta_2(r_2 + \sqrt{2})) \right\} + j \left\{ \frac{1}{\zeta_2} \cdot \sin(\zeta_2(r_2 + \sqrt{2})) \right\}$$

$$R_2^+ = \left\{ \frac{1}{\zeta_2} \cos(\zeta_2(r_2 - \sqrt{2})) \right\} + j \left\{ \frac{1}{\zeta_2} \cdot \sin(\zeta_2(r_2 - \sqrt{2})) \right\}$$

$$S_2 = \left\{ \frac{2\sin(\zeta_2)}{\zeta_1\zeta_2} \left[\frac{\sin(\zeta_1(r_1 + \sqrt{2} + 1))}{\zeta_1} - 1 - \sqrt{2} - r_1 \right] \right\} +$$

$$+ \; j \left\{ \frac{2\sin(\zeta_2)}{\zeta_1^2\zeta_2} \left[1 - \cos(\zeta_1(r_1 + \sqrt{2} + 1)) \right] \right\}$$

The third kernel is given by:

$$h_3(r_1, r_2) = \sum_{\zeta \in Z} \frac{1}{J(\zeta) \cdot f_3(\zeta)} \cdot \eta_\zeta^{(3)}(r_1, r_2) \tag{2.4.3}$$

with:

$$\eta_{\zeta}^{(3)}(r_1,r_2)=\begin{cases}
-\sqrt{3}<r_2\le-\sqrt{2}: & -\sqrt{3}-\sqrt{2}<r_1\le-\sqrt{3}+\sqrt{2}: & -C_2^-\cdot D_2\\
 & -\sqrt{3}+\sqrt{2}<r_1\le-\sqrt{2}+\sqrt{3}: & -C_2^-\cdot D_3\\
 & -\sqrt{2}+\sqrt{3}<r_1\le\sqrt{3}+\sqrt{2}: & -C_2^-\cdot D_4\\
-\sqrt{2}<r_2\le 0: & -\sqrt{3}-\sqrt{2}<r_1\le-\sqrt{3}+\sqrt{2}: & G_1\\
 & -\sqrt{3}+\sqrt{2}<r_1\le-\sqrt{2}+\sqrt{3}: & G_2\\
 & -\sqrt{2}+\sqrt{3}<r_1\le\sqrt{3}+\sqrt{2}: & G_3\\
0<r_2\le\sqrt{2}: & -\sqrt{3}-\sqrt{2}<r_1\le-\sqrt{3}+\sqrt{2}: & G_4\\
 & -\sqrt{3}+\sqrt{2}<r_1\le-\sqrt{2}+\sqrt{3}: & G_5\\
 & -\sqrt{2}+\sqrt{3}<r_1\le\sqrt{3}+\sqrt{2}: & G_6\\
\sqrt{2}<r_2\le\sqrt{3}: & -\sqrt{3}-\sqrt{2}<r_1\le-\sqrt{3}+\sqrt{2}: & -C_2^+\cdot D_2\\
 & -\sqrt{3}+\sqrt{2}<r_1\le-\sqrt{2}+\sqrt{3}: & -C_2^+\cdot D_3\\
 & -\sqrt{2}+\sqrt{3}<r_1\le\sqrt{3}+\sqrt{2}: & -C_2^+\cdot D_4\\
0, & \textit{elsewhere.} &
\end{cases} \tag{2.4.4}$$

with:

$$G_1=A_2^-\cdot B_2-C_2^-\cdot D_2 \tag{2.4.5}$$

$$G_2=A_2^-\cdot B_3-C_2^-\cdot D_3 \tag{2.4.6}$$

$$G_3=A_2^-\cdot B_4-C_2^-\cdot D_4 \tag{2.4.7}$$

$$G_4 = A_2^+ \cdot B_2 - C_2^+ \cdot D_2 \tag{2.4.8}$$

$$G_5 = A_2^+ \cdot B_3 - C_2^+ \cdot D_3 \tag{2.4.9}$$

$$G_6 = A_2^+ \cdot B_4 - C_2^+ \cdot D_4 \tag{2.4.10}$$

and:

$$A_2^- = \left\{\frac{1}{\zeta_2} \cdot \cos(\zeta_2(r_2 + \sqrt{2}))\right\} + j\left\{\frac{1}{\zeta_2} \cdot \sin(\zeta_1(r_2 + \sqrt{2}))\right\}$$

$$A_2^+ = \left\{\frac{1}{\zeta_2} \cdot \cos(\zeta_2(r_2 - \sqrt{2}))\right\} + j\left\{\frac{1}{\zeta_2} \cdot \sin(\zeta_1(r_2 - \sqrt{2}))\right\}$$

$$B_2 = \left\{\frac{2\sin(\zeta_2\sqrt{3})}{\zeta_1\zeta_2} \cdot \left[\frac{\cos(\zeta_1\sqrt{3})\sin(\zeta_1(r_1 + \sqrt{2}))}{\zeta_1} - r_1 - \sqrt{2} - \sqrt{3}\right]\right\} +$$

$$+ \; j\left\{\frac{2\sin(\zeta_2\sqrt{3})\cos(\zeta_1\sqrt{3})}{\zeta_1^2\zeta_2} \cdot \left[\cos(\zeta_1\sqrt{3}) - \cos(\zeta_1(r_1 + \sqrt{2}))\right]\right\}$$

$$B_3 = \left\{\frac{4\sin(\zeta_2\sqrt{3})}{\zeta_1\zeta_2} \cdot \left[\frac{\cos(\zeta_1\sqrt{3})\sin(\zeta_1\sqrt{2})\cos(\zeta_1 r_1)}{\zeta_1} - \sqrt{2}\right]\right\} +$$

$$+ \; j\left\{\frac{4\cos(\zeta_1\sqrt{3})\sin(\zeta_1\sqrt{2})\sin(\zeta_2\sqrt{3})\sin(\zeta_1 r_1)}{\zeta_1^2\zeta_2}\right\}$$

$$B_4 = \left\{\frac{2\sin(\zeta_2\sqrt{3})}{\zeta_1\zeta_2}\cdot\left[-\frac{\cos(\zeta_1\sqrt{3})\sin(\zeta_1(r_1-\sqrt{2}))}{\zeta_1} - r_1 - \sqrt{2} - \sqrt{3}\right]\right\} +$$

$$+ \; j\left\{\frac{2\cos(\zeta_1\sqrt{3})\sin(\zeta_1\sqrt{3})}{\zeta_1^2\zeta_2}\cdot\left[\cos(\zeta_1(r_1-\sqrt{2})) - \cos(\zeta_1\sqrt{3})\right]\right\}$$

$$C_2^- = \left\{\frac{1}{\zeta_2}\cdot\cos(\zeta_2(r_2+\sqrt{3}))\right\} + j\left\{\frac{1}{\zeta_2}\cdot\sin(\zeta_1(r_2+\sqrt{3}))\right\}$$

$$C_2^+ = \left\{\frac{1}{\zeta_2}\cdot\cos(\zeta_2(r_2-\sqrt{3}))\right\} + j\left\{\frac{1}{\zeta_2}\cdot\sin(\zeta_1(r_2-\sqrt{3}))\right\}$$

$$D_2 = \left\{\frac{2\sin(\zeta_2\sqrt{2})}{\zeta_1\zeta_2}\cdot\left[\frac{\cos(\zeta_1\sqrt{2})\sin(\zeta_1(r_1+\sqrt{3}))}{\zeta_1} - r_1 - \sqrt{3} - \sqrt{2}\right]\right\} +$$

$$+ \; j\left\{\frac{2\sin(\zeta_2\sqrt{2})\cos(\zeta_1\sqrt{2})}{\zeta_1^2\zeta_2}\cdot\left[\cos(\zeta_1\sqrt{2}) - \cos(\zeta_1(r_1+\sqrt{3}))\right]\right\}$$

$$D_3 = -\frac{4\sqrt{2}\sin(\zeta_2\sqrt{2})}{\zeta_1\zeta_2}$$

$$D_4 = \left\{\frac{2\sin(\zeta_2\sqrt{2})}{\zeta_1\zeta_2}\cdot\left[-\frac{\cos(\zeta_1\sqrt{2})\sin(\zeta_1(r_1-\sqrt{3}))}{\zeta_1} + r_1 - \sqrt{3} - \sqrt{2}\right]\right\} +$$

$$+ \; j\left\{\frac{2\sin(\zeta_2\sqrt{2})\cos(\zeta_1\sqrt{2})}{\zeta_1^2\zeta_2}\cdot\left[\cos(\zeta_1(r_1-\sqrt{3})) - \cos(\zeta_1\sqrt{2})\right]\right\}$$

The reader should be convinced by now that this approach is costly. We further remark that numerical evaluation of these formulas requires complex arithmetic in contrast with the original frequency-domain formulas that can be evaluated using real arithmetic. Therefore we focus on frequency domain computation and more on the necessary windowing. (Windowing also applies to time-domain computation; nevertheless, its effects are more readily visualized in the frequency domain).

2.5 Windowing

Our goal is the pointwise evaluation of the F.T. of approximate deconvolution kernels over a suitably chosen finite grid. For ease of reference we reproduce the formula obtained earlier:

$$\widehat{h}_i(z_1, z_2) = \sum_{(\zeta_1,\zeta_2)\in\mathcal{Z}} \frac{\widehat{u}(\zeta_1, \zeta_2)}{J(\zeta_1, \zeta_2) \cdot \widehat{f}_3(\zeta_1, \zeta_2)} \cdot \frac{C_i(z_1, z_2, \zeta_1, \zeta_2)}{(z_1 - \zeta_1) \cdot (z_2 - \zeta_2)} \tag{2.5.1}$$

Clearly this involves the pointwise (in z) computation of an infinite sum. Therefore, one way or another, we will have to truncate this sum at some point. This is equivalent to forcing $\widehat{u}(\zeta)$ to be band-limited, thus forcing $u(t)$ to be non-compactly supported and therefore violating the corresponding condition of Theorem 1. Hence, we must strike a balance between computational feasibility and support of u(t) in order to obtain meaningful results. Notice that in principle $u(t)$ can by anything as long as it is sufficiently compactly supported. Also note that *u(t) is the impulse response of the overall system that we realize* and thus

$\widehat{u}(z)$ is the frequency response of the overall system. Clearly we would like u(t) to be as close to $\delta(t)$ as possible (in the sense of distributions) or, equivalently, $\widehat{u}(z)$ to be as close to unity as possible. This means that we would like to include as many terms as possible in 2.5.1. On the other hand, noise considerations dictate a *smooth* choice of u(t) which in turn implies a fast decay of $\widehat{u}(z)$ at infinity. Therefore we have to accommodate conflicting interests. In practice we are not trying to achieve infinite Bandwidth but rather a large *finite* bandwidth. This is due to practical limitations and noise considerations. More precisely stated, we seek a compromise in the choice of $\widehat{u}(z)$ that will account for the following:

1. $u(t)$ is sufficiently compactly supported (i.e. the condition of Theorem 1.3 is approximately satisfied).

2. $\widehat{u}(z)$ is close to the identity over some bounded region of interest (i.e. bandwidth requirements are being met).

3. $u(t)$ is sufficiently smooth (Noise averaging)

These requirements are clearly interrelated. Now it is not at all clear what is a proper choice for $\widehat{u}$. The following have been considered, for different reasons, and with different degrees of success:

(i) Use a Hamming-like window in place of $\widehat{u}(z)$. This has been motivated by (1), (2), (3) and the fact that it exhibits fairly good ripple characteristics [30] .

(ii) Use a Butterworth type filter in place of $\widehat{u}(z)$. This is similar to the above. In addition $\widehat{u}(z)$ is "maximally flat" resulting in good noise performance [30] .

(iii) Use the theory of Prolate Spheroidal functions. This is motivated by the work of D. Slepian, H. Pollak who proved that these functions strike an optimal balance between support in time and support in frequency [28,18,27] .

(iv) Use functions of the form:

$$\widehat{u}(z) = \left(\prod_{i=1}^{2} \frac{\sin(\frac{c_i}{N} z_i)}{\frac{c_i}{N} z_i} \right)^N \cdot P_r(z) \tag{2.5.2}$$

with:

$$p_r(z) = \begin{cases} 1, & |z_i| \leq r \ , \ i = 1,2 \\ 0, & elsewhere \end{cases} \tag{2.5.3}$$

This approach comes from an attempt to control the residue in 2.5.1 due to the non-compactness of the support of $u(t)$. We shall consider (i) and (iv) in some detail.

(i) - Hamming-like windowing [30] .

Let:

$$\widehat{u}(\zeta_1, \zeta_2) \triangleq \prod_{i=1}^{2} \left(0.5 + 0.5 \cos(\frac{\pi \zeta_i}{r}) \right) \cdot p_r(\zeta_i) \tag{2.5.4}$$

with:

$$p_r(\zeta_i) = \begin{cases} 1, & |\zeta_i| \leq r \\ 0, & elsewhere \end{cases} \tag{2.5.5}$$

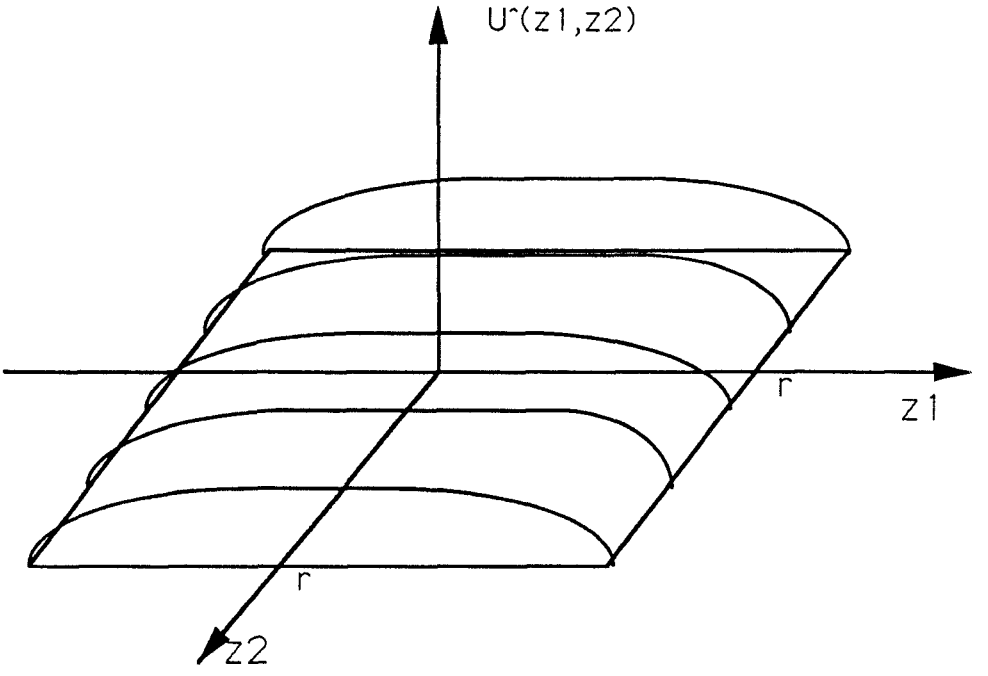


Figure 2.1: Hamming-like window

This is depicted in figure 2.1 . The parameter r controls the *size of the nullset* that we consider since any term due to a nullpoint outside the support of $\widehat{u}$ will be forced to zero. Notice that now the sum in 2.5.1 is well-defined and finite.

With this choice of $\widehat{u}$ we have:

$$u(t_1,t_2) = \prod_{i=1}^{2} \sin(t_i r) \left[\frac{1}{2\pi t_i} + \frac{r^2/2\pi}{\pi^2 - r^2 t_i^2} \cdot t_i \right] \tag{2.5.6}$$

This is depicted in figure 2.2.

The energy content of u is easily obtained as:

$$E_u = \int_{-r}^{r} \int_{-r}^{r} [\widehat{u}(\zeta_1,\zeta_2)]^2 \, d\zeta_1 d\zeta_2$$

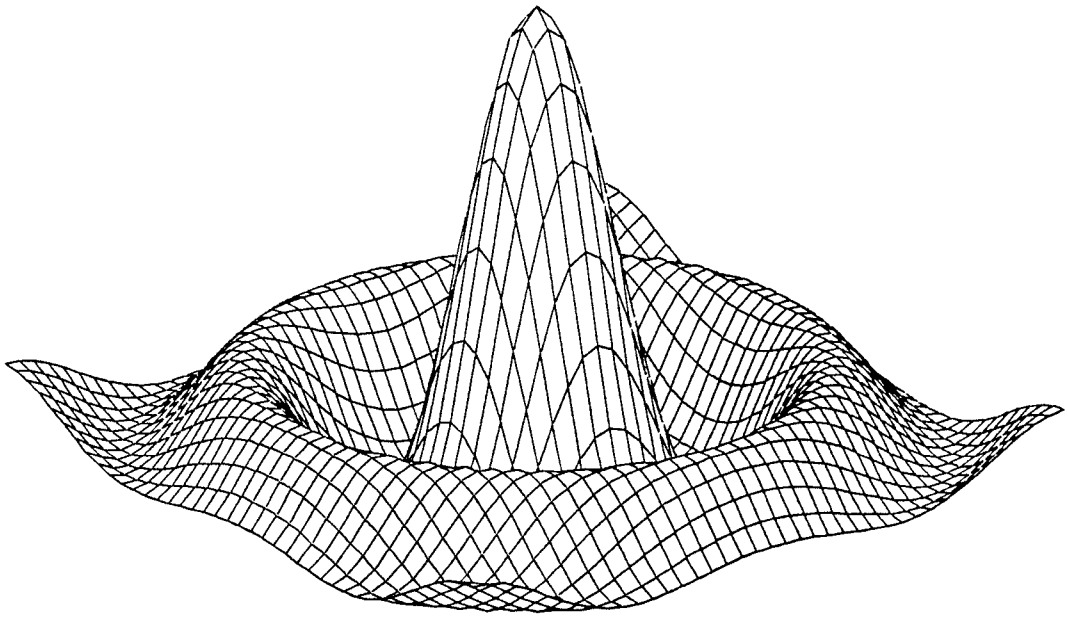


Figure 2.2: Inverse FT of Hamming-like window

$$= \left(\frac{3r}{4}\right)^2$$

Whereas the volume under $u(t_1, t_2)$, V_u, is given by:

$$V_u = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(t_1,t_2)e^{-j(z_1t_1+z_2t_2)}dt_1dt_2\Big|_{z_1=z_2=0}$$

$$= \widehat{u}(0,0) = 1$$

Thus by increasing r we put more energy into u, keep the volume under $u(t_1, t_2)$ fixed to 1, the peak of $u(t_1, t_2)$ is pushed up, and its main lobe narrows.

This type of window is simple and it requires the choice of only one parameter, namely r. Furthermore $\widehat{u}$ approaches the identity (i.e. u approaches δ) as

$r \longrightarrow \infty$. Hence we should be able to find a suitable r to fit our needs. Simulations have proven that only partially good results can be expected for any reasonable choice of r. This is due to the fact that for any finite r, $u(t_1, t_2)$ is not compactly supported and convergence is not achieved. We *do get* a much wider bandwidth; but because of the asymmetry present (cf. our previous discussion of $\pi/2$-rotation-invariance) we are doing much better along one frequency (z_2) than along the other, i.e. along z_1 convergence is much slower (and in fact we never achieve bandwidth r for all reasonable values of r). Typical of this case is that the solution is gradually converging radially about the origin. A final remark for Hamming windowing: Assuming presence of AGWN the computation of output noise psd is particularly easy, and output noise power can easily be trimmed within specifications by picking r smaller if necessary.

Next we turn to the use of powers of 2-dimensional sinc-like functions, which have proven to be more appropriate than anything else we have considered.

(iv) - Sinc-like Windowing.

We consider functions of the form:

$$\widehat{u}(\zeta_1, \zeta_2) = \left(\prod_{j=1}^{2} \frac{\sin(\frac{\epsilon}{N} z_j)}{\frac{\epsilon}{N} z_j} \right)^N \cdot p_r(\zeta_1) p_r(\zeta_2) \tag{2.5.7}$$

where ϵ is a small parameter, N is a small positive integer, and r is a sufficiently large parameter.

For a moment let us forget about the cut-off at r forced by the product $p_r(\zeta_1)p_r(\zeta_2)$ and concentrate on the first factor.

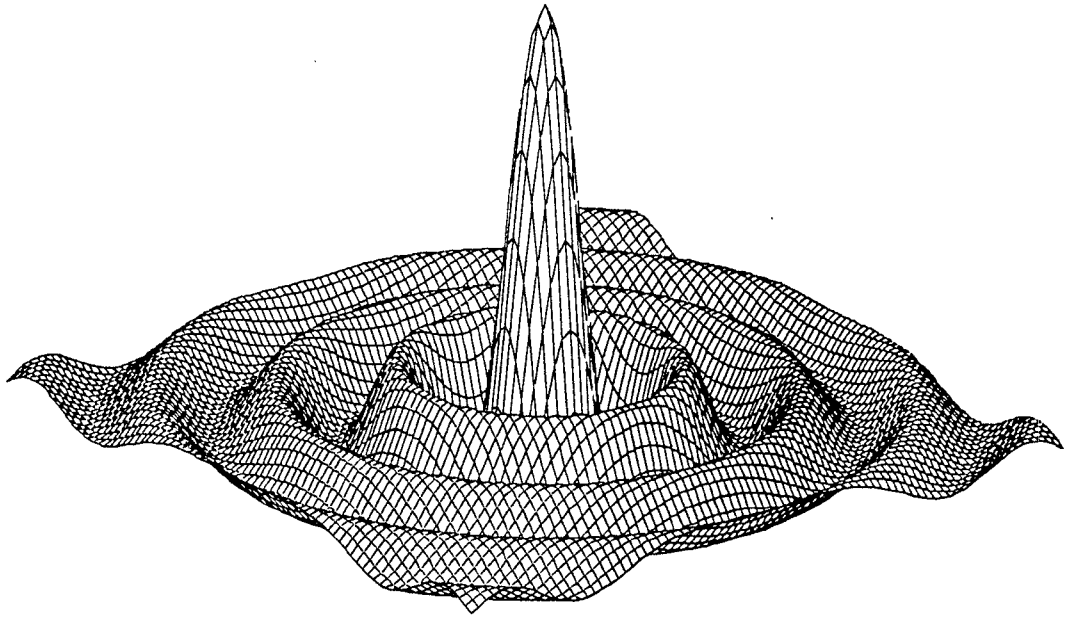


Figure 2.3: Two Dimensional sinc-like function

Let N = 1

$$\widehat{u}(\zeta_1, \zeta_2) = \frac{\sin(\epsilon\zeta_1)\sin(\epsilon\zeta_2)}{\epsilon\zeta_1\epsilon\zeta_2}$$

$$= \left(\frac{1}{4\epsilon^2}\right) \cdot \frac{4}{\zeta_1\zeta_2}\sin(\epsilon\zeta_1)\sin(\epsilon\zeta_2) \qquad (2.5.8)$$

This is a sinc-like function with peak amplitude one and zeros along both frequency variables located at multiples of $\frac{\pi}{\epsilon}$. It is depicted in figure 2.3.

Therefore we can infer that $u(t_1, t_2)$ is $\frac{1}{4\epsilon^2}$ times the characteristic function over a square of side 2ϵ. The characteristic function over a square of side a is depicted in figure 2.4.

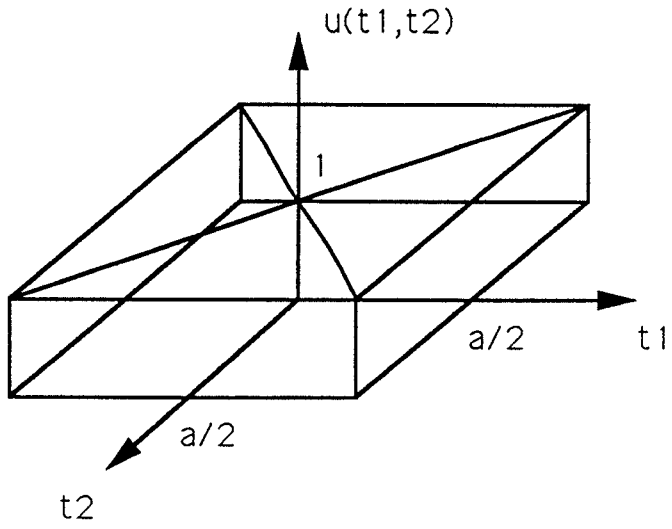


Figure 2.4: Characteristic Function over square of side a

Let N = 2

$$\widehat{u}(\zeta_1,\zeta_2) = \left(\frac{\sin(\frac{\epsilon}{2}\zeta_1)\sin(\frac{\epsilon}{2}\zeta_2)}{\frac{\epsilon}{2}\zeta_1\frac{\epsilon}{2}\zeta_2}\right)^2$$

$$= (\widehat{u}_1(\zeta_1,\zeta_2))^2 \tag{2.5.9}$$

Observe that $\widehat{u}_1$ is a sync-like function with peak amplitude equal to one and zeros along both frequency variables located at multiples of $2\frac{\pi}{\epsilon}$. Therefore $u_1(t_1,t_2)$ equals $\frac{1}{\epsilon^2}$ times the characteristic function over a square of side ϵ. Hence $\widehat{u}(\zeta_1,\zeta_2) = (\widehat{u}_1(\zeta_1,\zeta_2))^2$ is a squared sinc-like function, nonnegative with the same zeros as $\widehat{u}_1$.

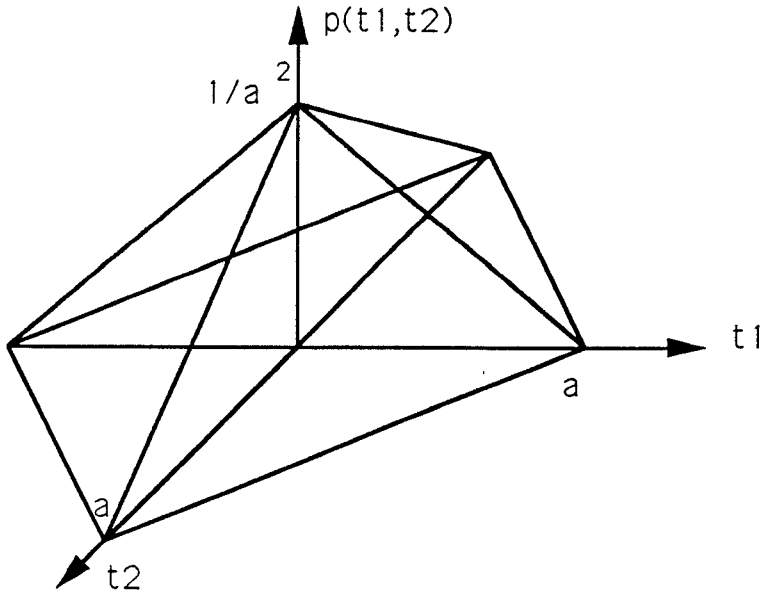


Figure 2.5: Pyramid of diagonal 2a and peak $1/a^2$

Therefore, since:

$$u(t_1, t_2) = (u_1 * u_1)(t_1, t_2)$$

the function $u(t_1, t_2)$ is a pyramid-shaped surface, of diagonal 2ϵ and peak $\frac{1}{\epsilon^2}$, depicted in figure 2.5 for $\epsilon = a$.

So by keeping ϵ fixed and increasing N:

- We keep the support of $u(t_1, t_2)$ fixed

- The peak of $u(t_1, t_2)$ gets taller

- $u(t_1, t_2)$ gets smoother

- The main lobe of $\widehat{u}(\zeta_1, \zeta_2)$ gets wider

- The peak of $\widehat{u}(\zeta_1, \zeta_2)$ remains fixed to 1

- The amplitude within the main lobe of $\widehat{u}(\zeta_1, \zeta_2)$ flattens out

On the other hand, if we keep N fixed and decrease ϵ:

- The support of $u(t_1, t_2)$ narrows

- The peak of $u(t_1, t_2)$ gets taller

- $u(t_1, t_2)$ does not get smoother (we assume that N has been chosen large enough so that $u(t_1, t_2)$ is sufficiently smooth)

- The main lobe of $\widehat{u}(\zeta_1, \zeta_2)$ widens

- Its peak stays fixed to 1

- The main lobe amplitudes are pushed up towards unity

A reasonable choice of the parameters involved is: $\epsilon = 0.1$ (to achieve 1/10 of the support of the best convolution kernel), $N = 3$ (to make u smooth enough). With this choice of parameters we get:

$$\widehat{u}(\zeta_1, \zeta_2) = \left(\prod_{i=1}^{2} \frac{\sin(\frac{\zeta_i}{30})}{\frac{\zeta_i}{30}} \right)^3 \cdot p_r(\zeta_1) p_r(\zeta_2) \tag{2.5.10}$$

where the parameter r (forced cutoff in rads/sec) is to be chosen sufficiently large to include all main features of the first factor (the main lobe and the principal sidelobes at least) , while keeping the size of the computation reasonable.

This particular choice has proven to be the best of all thus far. It helps achieve wide bandwidth, while reducing the magnitude of the asymmetry present. Furthermore it makes the convergence uniform in z_2 (i.e. convergence is not any more achieved radially about the origin but is essentially independent of z_2). Simulation results are given at the end of this chapter . The main drawback is that again we have a high degree of energy concentration along a ribbon-like neighborhood of the z_2 axis, while amplitudes everywhere else are attenuated by at least an order of magnitude. Thus it is quite natural to consider techniques that can effectively spread out the energy content of the overall system response and result in a relatively flat spectrum.

2.6 Energy Spreading via Directional Frequency Windowing and Averaging

In this section we consider the use of a direction-selective asymmetric window to compensate for energy concentration. Since sinc-like mollifiers have demonstrated good behavior consider a modification of the form:

$$\widehat{u}(\zeta_1,\zeta_2) = \left(\prod_{i=1}^{2} \frac{\sin(\frac{\epsilon_i\zeta_i}{N})}{\frac{\epsilon_i\zeta_i}{N}}\right)^N \cdot p_r(\zeta_1)p_r(\zeta_2) \tag{2.6.1}$$

where $\epsilon_1 \neq \epsilon_2$. Now since we achieve better bandwidth characteristics along ζ_2 we choose ϵ_1 smaller than ϵ_2 by half an order of magnitude (observe that the magnitude of the main lobe is $\sim \frac{1}{\epsilon}$). This will compensate for the inherent asymmetry by essentially spreading out the energy originally concentrated along

the neighborhood of z_2.

Simulation results are given at the end of this chapter for the following choice of parameters: $N = 3,\ \epsilon_1 = 0.1,\ \epsilon_2 = 0.5$.

The preceeding discussion has made clear that the asymmetry resulting from finite approximations of the proposed deconvolution kernels actually manifests itself in a profound way and can severely distort the overall system spectrum even at frequencies in the vicinity of the origin. There exists no a priori reason for the appearance of such asymmetries (for the particular model at hand is completely symmetric) but rather the cause can be traced back to a somewhat arbitrary choice between two distinct possibilities in writing down interpolation formulas. Before we discuss this very important point let us give a partial "a posteriori" solution: if the transforms of the convolver kernels are symmetric and $\frac{\pi}{2}$-rotation invariant then a simple solution would be the following:

Let: $\widehat{h}_{i,n}(z_1, z_2),\ i = 1, 2, 3$ denote the obtained approximations of the Fourier Transforms of the deconvolution kernels, where n denotes the cardinality of the nullset, $\mathcal{Z}_n$, over which we sum. Next for $i = 1, 2, 3$ define:

$$\widehat{h}^L_{i,n}(z_1, z_2) \triangleq \widehat{h}_{i,n}(z_2, z_1) \tag{2.6.2}$$

and:

$$\widehat{h}^*_{i,n}(z_1, z_2) \triangleq \frac{\widehat{h}_{i,n}(z_1, z_2) + \widehat{h}^L_{i,n}(z_1, z_2)}{2} \tag{2.6.3}$$

By definition $\widehat{h}^*_{i,n}(\cdot, \cdot)$ is $\frac{\pi}{2}$ rotation invariant since:

$$\widehat{h}^*_{i,n}(z_2, z_1) \triangleq \frac{\widehat{h}_{i,n}(z_2, z_1) + \widehat{h}^L_{i,n}(z_2, z_1)}{2}$$

$$= \frac{\widehat{h}^L_{i,n}(z_1, z_2) + \widehat{h}_{i,n}(z_1, z_2)}{2}$$

$$= \widehat{h}^*_{i,n}(z_1, z_2) \tag{2.6.4}$$

Let $\widetilde{h}_i,\ i = 1, 2, 3$ denote the Fourier transforms of the exact deconvolution kernels. Then by theorem 1.3 we have that:

$$\widehat{h}_{i,n} \longrightarrow \widetilde{h}_i,\ as\ n \longrightarrow \infty,\ for\ i = 1, 2, 3 \tag{2.6.5}$$

Thus, by symmetry and $\frac{\pi}{2}$ rotation invariance of the solution:

$$\widehat{h}^L_{i,n} \longrightarrow \widetilde{h}^L_i \equiv \widetilde{h}_i,\ as\ n \longrightarrow \infty,\ for\ i = 1, 2, 3 \tag{2.6.6}$$

Hence the same is true for their average, i.e. the family $\{\widehat{h}^*_{i,n};\ i = 1, 2, 3\}$ constitutes a converging solution. Furthermore for all finite n the later family behaves better because it acquires bandwidth in a radially increasing fashion. Simulation results for this averaged solution are given at the end of this chapter.

We now turn to the general case where the set of convolvers is not symmetric. In this case it is not necessarily true that $\widetilde{h}^L_i \equiv \widetilde{h}_i$ and the remedy above fails. In fact *we expect* the exact solutions to be asymmetric too. Nevertheless we have to account for asymmetries introduced by the need to come up with a finite computation because otherwise our results will be severely distorted. We now investigate the cause of these asymmetries and proceed to propose a definitive solution. Consider the determinant involved in the interpolation formula 1.2.11

of Theorem 1.3 .

$$d = \begin{vmatrix} g_1^1(z,\zeta) & g_1^2(z,\zeta) & g_1^3(z,\zeta) \\ g_2^1(z,\zeta) & g_2^2(z,\zeta) & g_2^3(z,\zeta) \\ \widehat{f}_1(z) & \widehat{f}_2(z) & \widehat{f}_3(z) \end{vmatrix} \tag{2.6.7}$$

where the $g_j^i(z,\zeta)$, $i = 1,2,3$, $j = 1,2$ are holomorphic functions given by:

$$g_1^i(z,\zeta) = \frac{\widehat{f}_i(z_1,\zeta_2) - \widehat{f}_i(\zeta_1,\zeta_2)}{z_1 - \zeta_1} \tag{2.6.8}$$

$$g_2^i(z,\zeta) = \frac{\widehat{f}_i(z_1,z_2) - \widehat{f}_i(z_1,\zeta_2)}{z_2 - \zeta_2} \tag{2.6.9}$$

Now consider the first column of d. The idea is to write [9] :

$$\widehat{f}_1(z_1,z_2) - \widehat{f}_1(\zeta_1,\zeta_2) = (z_1 - \zeta_1)\cdot g_1^1(z,\zeta) + (z_2 - \zeta_2)\cdot g_2^1(z,\zeta) \tag{2.6.10}$$

Quite clearly this can also be achieved via:

$$\widehat{f}_1(z_1,z_2) - \widehat{f}_1(\zeta_1,\zeta_2) = (z_1 - \zeta_1)\cdot \bar{g}_1^1(z,\zeta) + (z_2 - \zeta_2)\cdot \bar{g}_2^1(z,\zeta) \tag{2.6.11}$$

Where $\bar{g}_1^1(z,\zeta)$ and $\bar{g}_2^1(z,\zeta)$ are defined as follows:

$$\bar{g}_1^1(z,\zeta) \triangleq \frac{\widehat{f}_1(z_1,z_2) - \widehat{f}_1(\zeta_1,z_2)}{z_1 - \zeta_1} \tag{2.6.12}$$

$$\bar{g}_2^1(z,\zeta) \triangleq \frac{\widehat{f}_1(\zeta_1,z_2) - \widehat{f}_1(\zeta_1,\zeta_2)}{z_2 - \zeta_2} \tag{2.6.13}$$

There is no *a priori* reason for choosing any particular holomorphic form; either will do. Nevertheless *some* choice has to be made. Although in the limit this choice makes no difference, it is the hidden cause of spurious asymmetries

for all finite $n = \#\mathcal{Z}$, because *either* expansion pair of holomorphic functions is biased towards one of the two frequency variables. Therefore we can define:

$$\bar{d} = \begin{vmatrix} \bar{g}_1^1(z,\zeta) & \bar{g}_1^2(z,\zeta) & \bar{g}_1^3(z,\zeta) \\ \bar{g}_2^1(z,\zeta) & \bar{g}_2^2(z,\zeta) & \bar{g}_2^3(z,\zeta) \\ \hat{f}_1(z) & \hat{f}_2(z) & \hat{f}_3(z) \end{vmatrix} \tag{2.6.14}$$

with $\bar{g}_1^2$, $\bar{g}_2^2$, $\bar{g}_1^3$, $\bar{g}_2^3$ defined as follows:

$$\bar{g}_1^2(z,\zeta) = \frac{\hat{f}_2(z_1,z_2) - \hat{f}_2(\zeta_1,z_2)}{z_1 - \zeta_1} \tag{2.6.15}$$

$$\bar{g}_2^2(z,\zeta) = \frac{\hat{f}_2(\zeta_1,z_2) - \hat{f}_2(\zeta_1,\zeta_2)}{z_2 - \zeta_2} \tag{2.6.16}$$

$$\bar{g}_1^3(z,\zeta) = \frac{\hat{f}_3(z_1,z_2) - \hat{f}_3(\zeta_1,z_2)}{z_1 - \zeta_1} \tag{2.6.17}$$

$$\bar{g}_2^3(z,\zeta) = \frac{\hat{f}_3(\zeta_1,z_2) - \hat{f}_3(\zeta_1,\zeta_2)}{z_2 - \zeta_2} \tag{2.6.18}$$

Now we need to use equation 1.2.11 to obtain two sets of solutions: one using the original d, and one using $\bar{d}$ in place of d. Again since both solutions converge to the exact deconvolvers as $n \longrightarrow \infty$ the same is true for their average. Furthermore, the bias is canceled out and does not appear in the overall system spectrum. Observe that if the transforms of the convolvers are symmetric and $\frac{\pi}{2}$ rotation invariant this approach reduces to the much simpler *a posteriori* remedy discussed earlier, which requires about half as much computation. In

any case directional windowing is beneficial because we are generally interested in a completely symmetrical frequency response of the overall system, i.e. a system response that is ϕ - rotation invariant for arbitrary $\phi \in [0, 2\pi]$.

Simulation results are presented in the sequence of figures that follows. A nullset cardinality $n = \#\mathcal{Z} = 3200$ points, a frequency step of 0.1718 rads/sec and a frequency resolution of $256x256$ points is adopted throughout the whole sequence of simulations. The parameter t denotes threshold value. The magnitude of the Fourier Transform of the convolution kernels for the model problem is plotted in figures 2.6, 2.7, 2.8. The magnitude of the Fourier Transform of the overall system using sinc like frequency windowing with parameters $\epsilon = 0.1$, $N = 3$ is plotted in figures 2.9, 2.10, 2.11 for various threshold levels. The following three figures, 2.12, 2.13, 2.14 depict the magnitude of the Fourier Transform of the overall system using directional frequency windowing with parameters $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$, $N = 3$. The last four figures, 2.15 ,2.16, 2.17, 2.18 depict the magnitude of the Fourier transform of the overall system using frequency averaging of the resulting deconvolution kernels depicted in figures 2.12 up to 2.14.

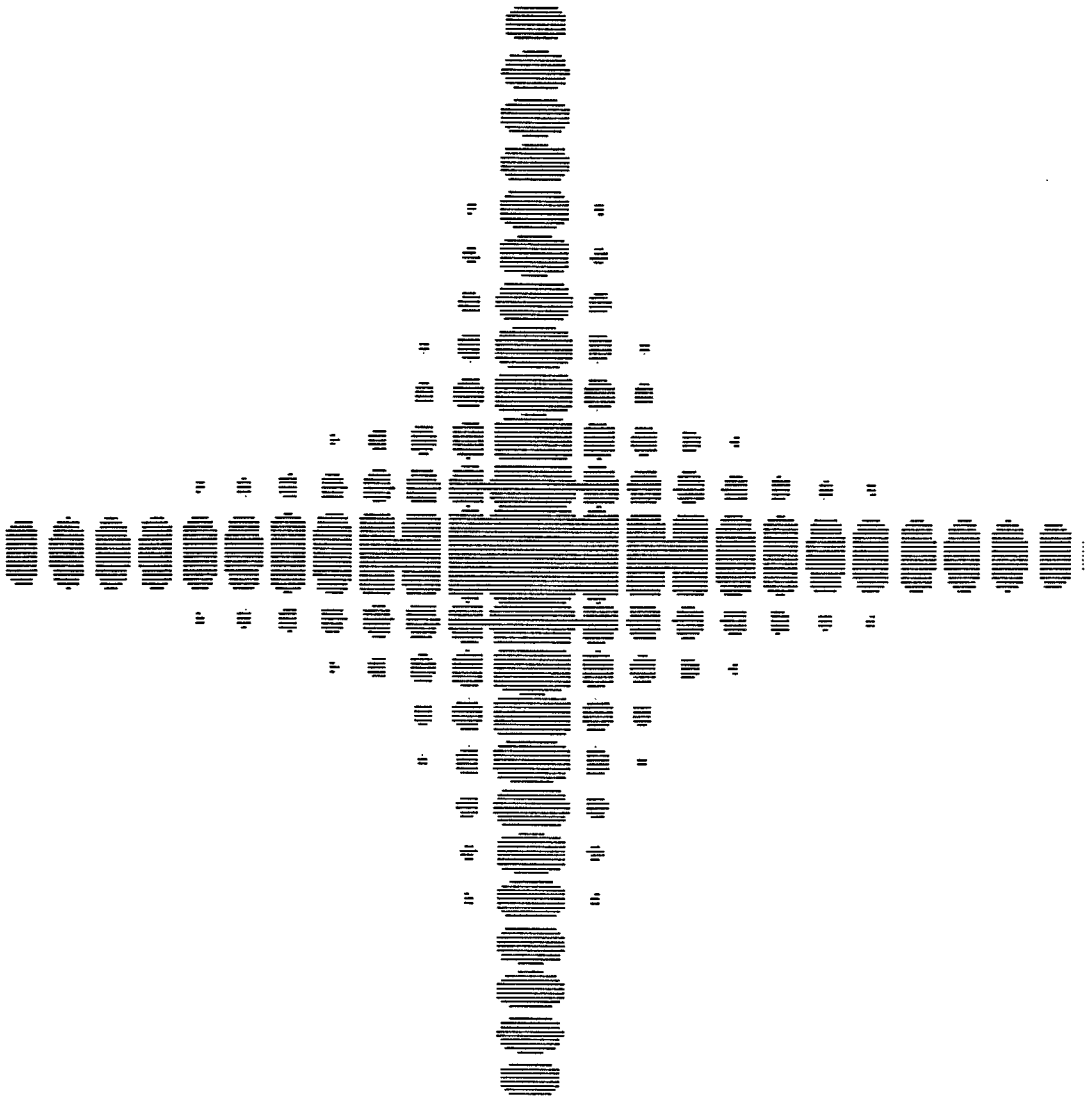


Figure 2.6: Magnitude of FT of convolver 1, t=0.1

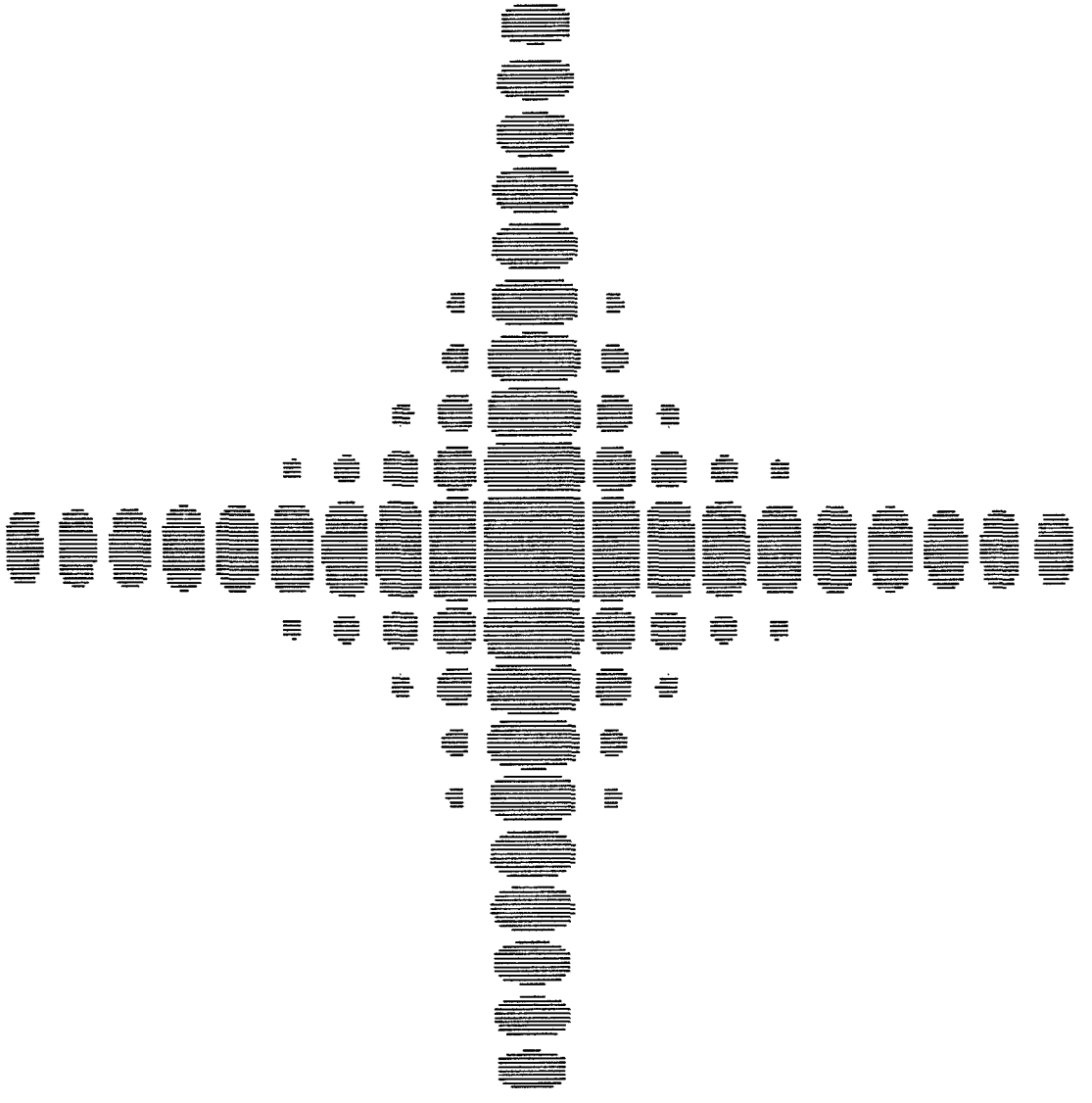


Figure 2.7: Magnitude of FT of convolver 2, t=0.1

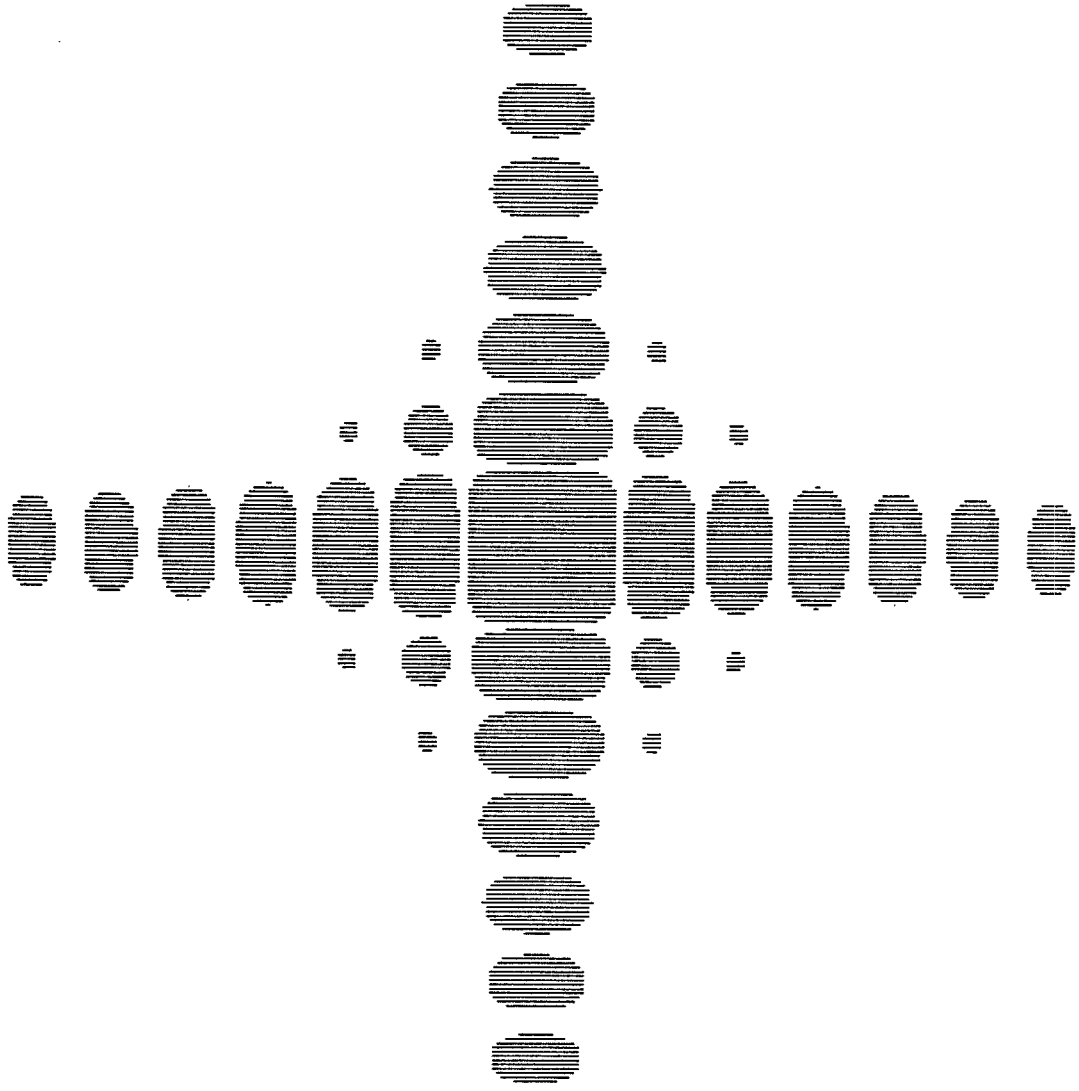


Figure 2.8: Magnitude of FT of convolver 3 (best one), t=0.1

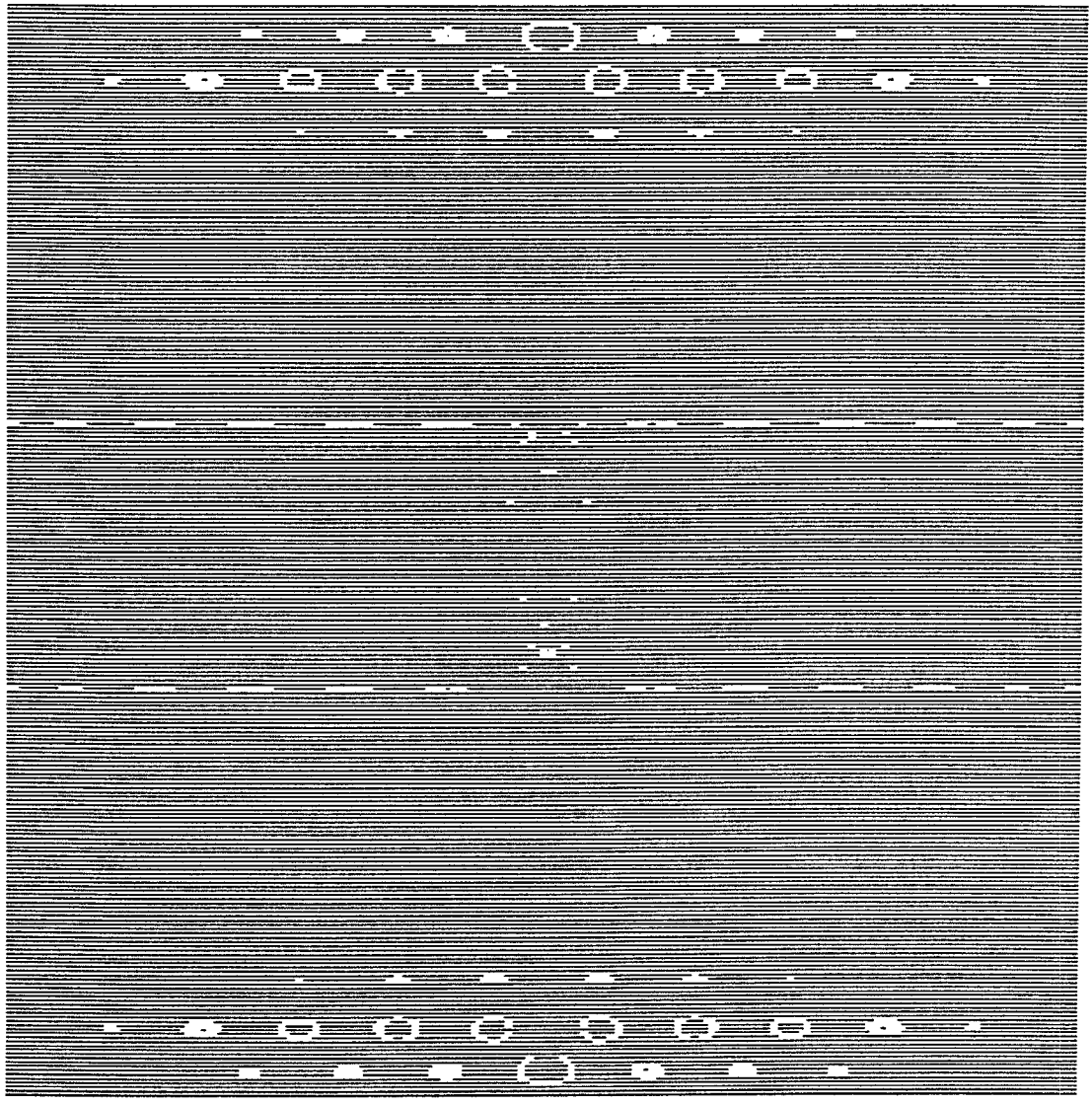


Figure 2.9: Magnitude of FT of overall system, $\epsilon = 0.1,\ N = 3,\ t = 0.1$

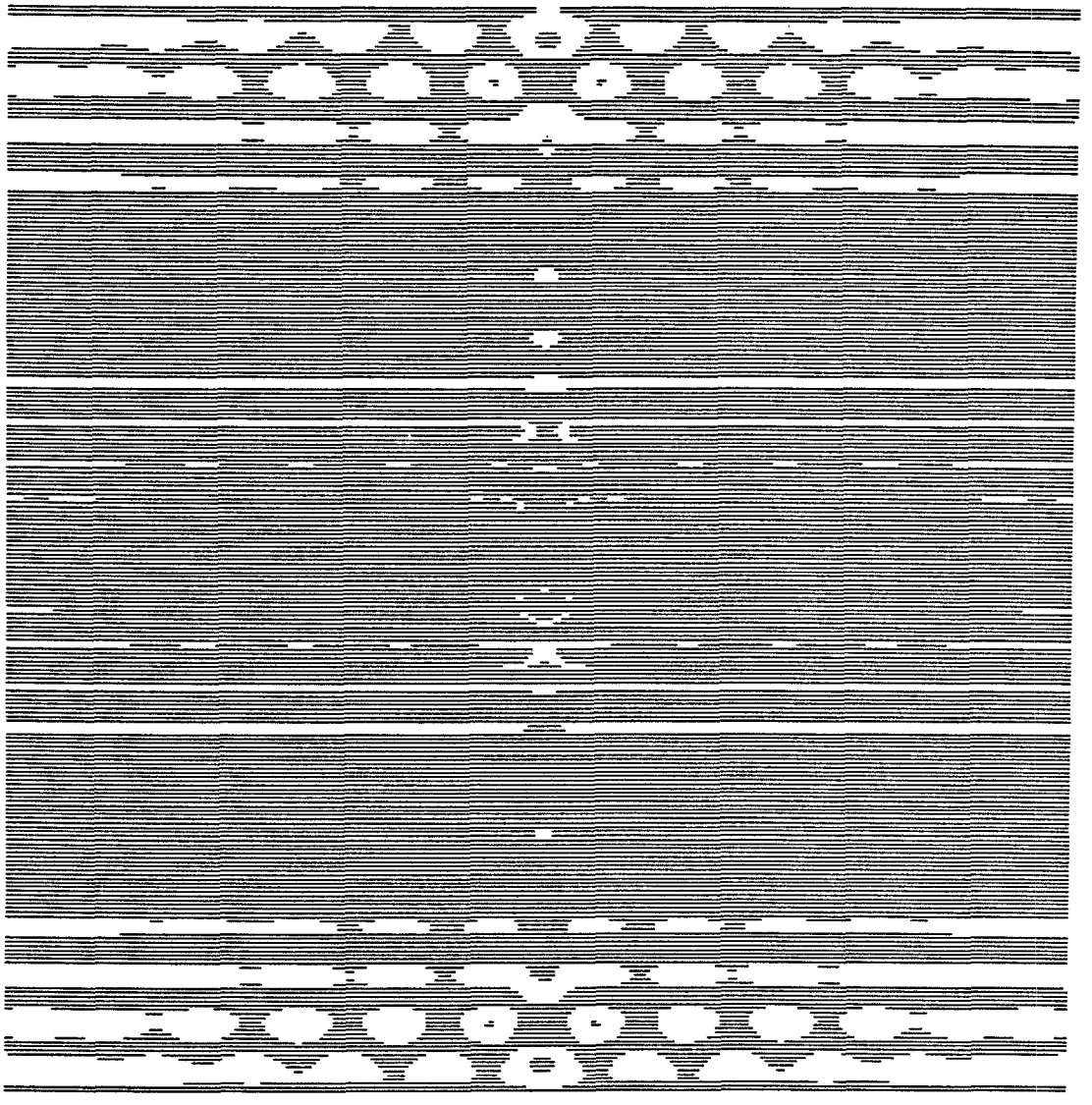


Figure 2.10: Magnitude of FT of overall system, $\epsilon = 0.1,\ N = 3,\ t = 0.5$

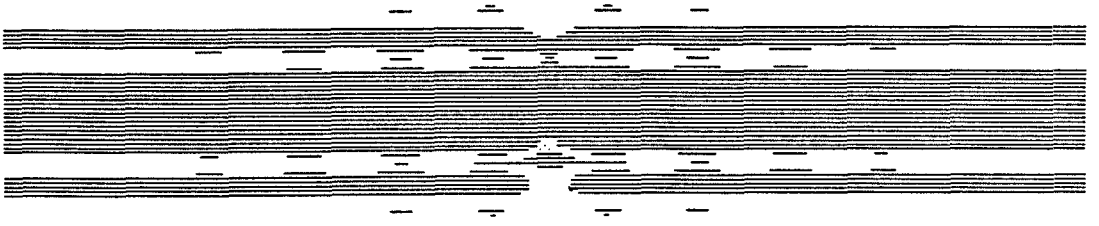


Figure 2.11: Magnitude of FT of overall system, $\epsilon = 0.1,\ N = 3,\ t = 5.0$

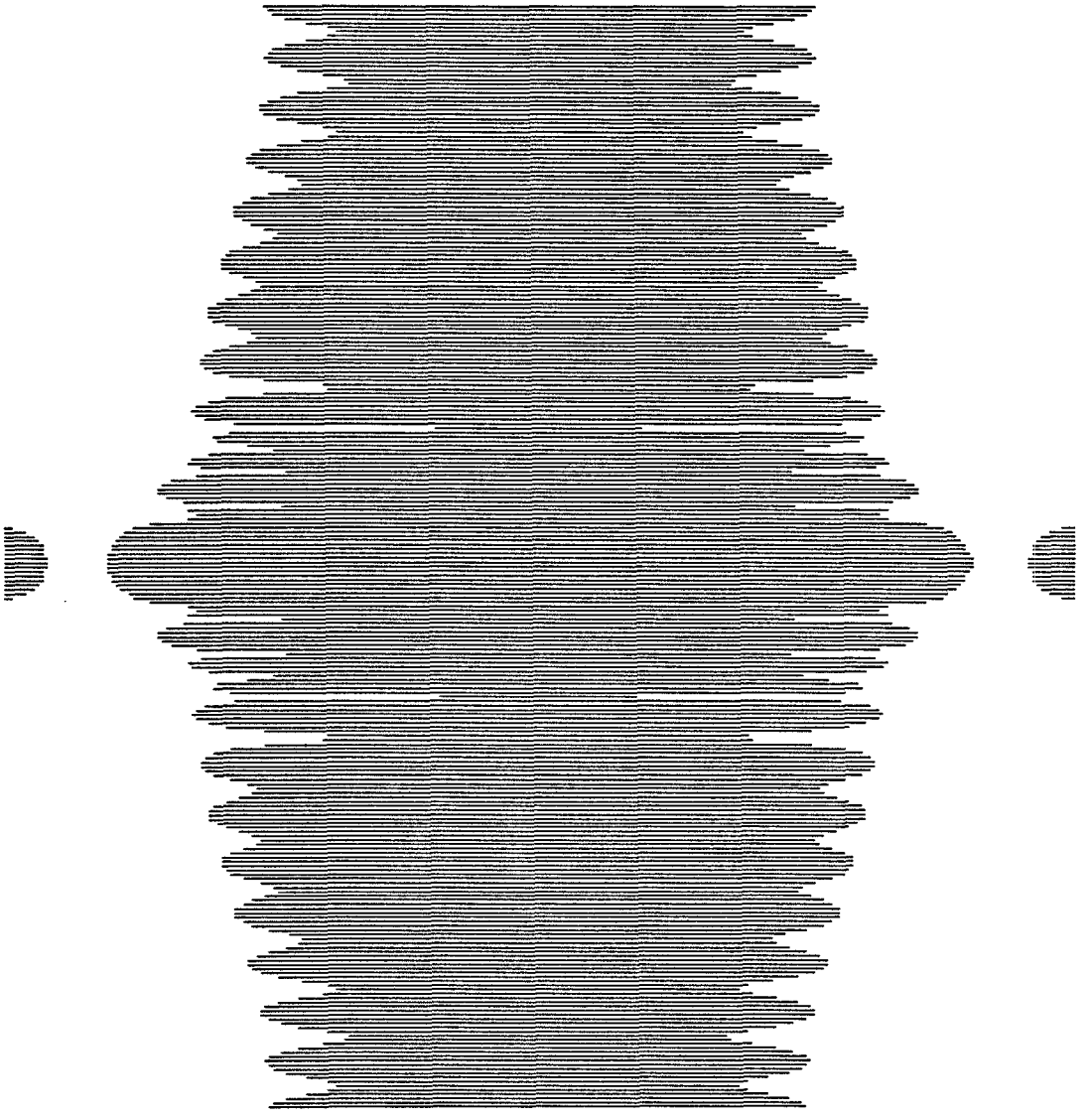


Figure 2.12: Magnitude of FT of overall system, $\epsilon_1 = 0.1,\ \epsilon_2 = 0.5,\ N = 3, t = 0.1$

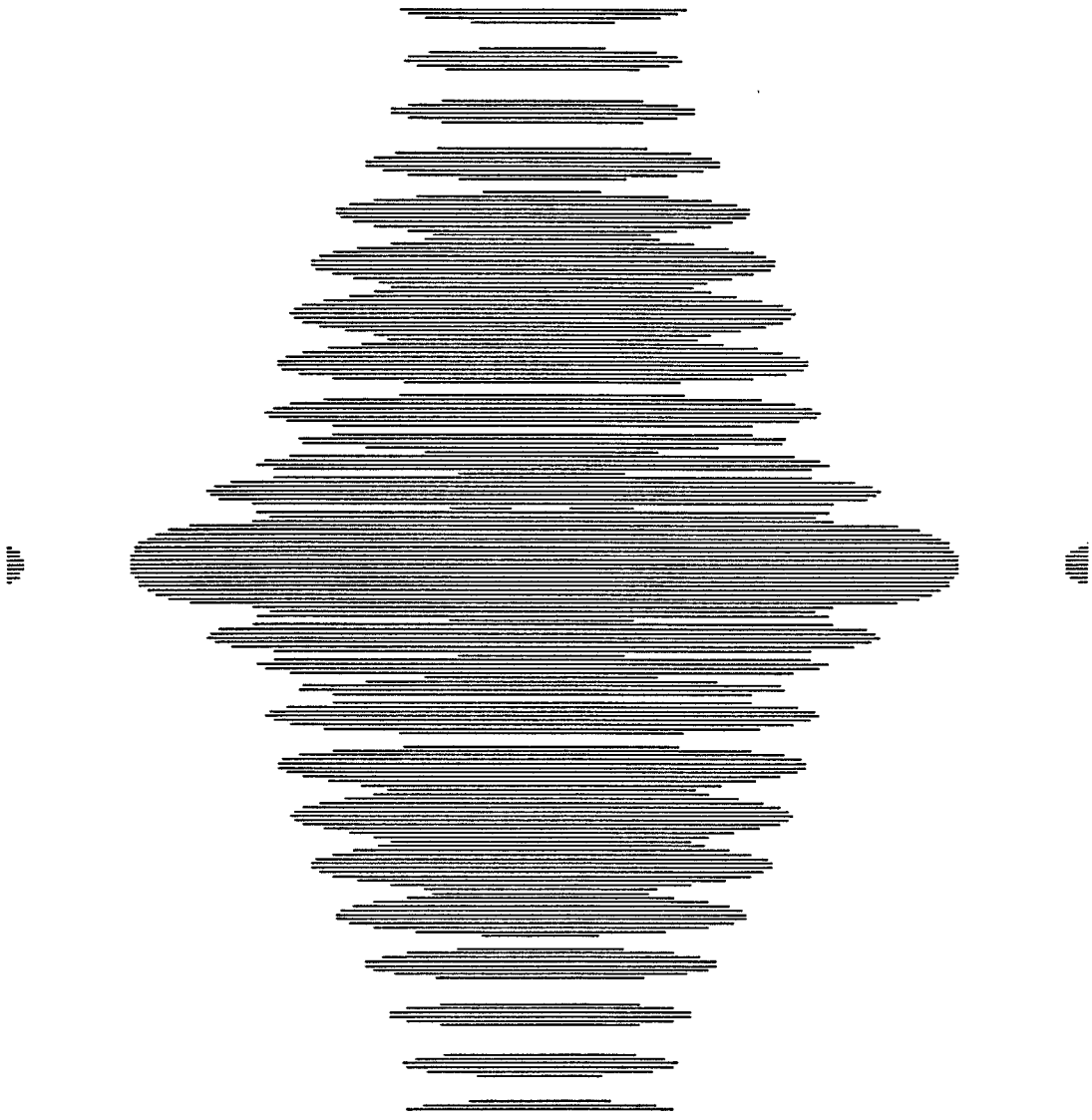


Figure 2.13: Magnitude of FT of overall system, $\epsilon_1 = 0.1,\ \epsilon_2 = 0.5,\ N = 3, t = 0.5$

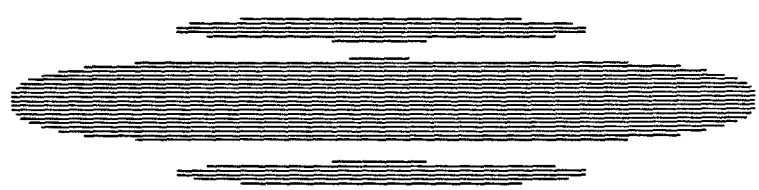


Figure 2.14: Magnitude of FT of overall system, $\epsilon_1 = 0.1,\ \epsilon_2 = 0.5,\ N = 3, t = 5.0$

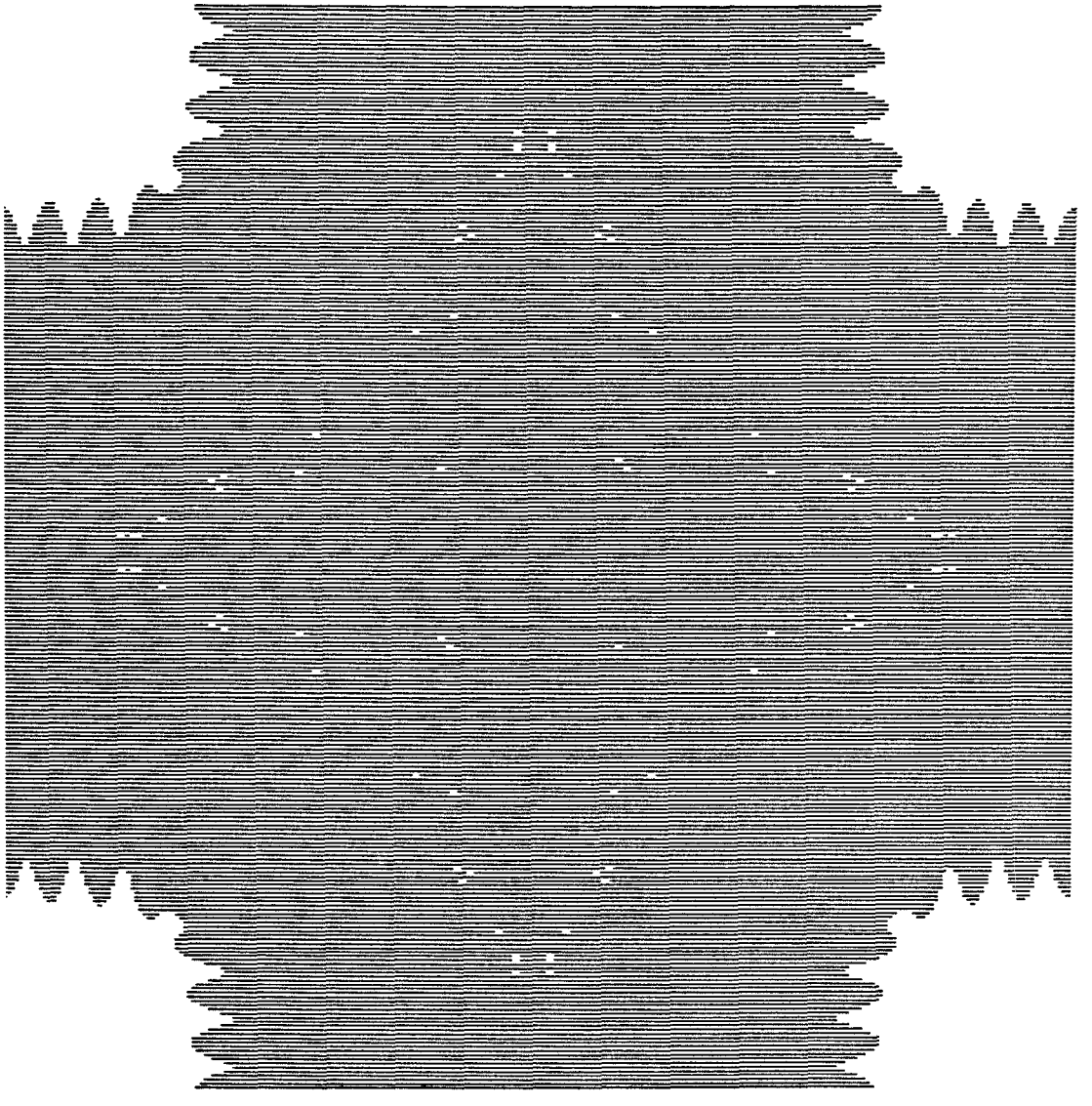


Figure 2.15: Magnitude of FT of overall system, $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$, $N = 3$, *averaged*, $t = 0.01$

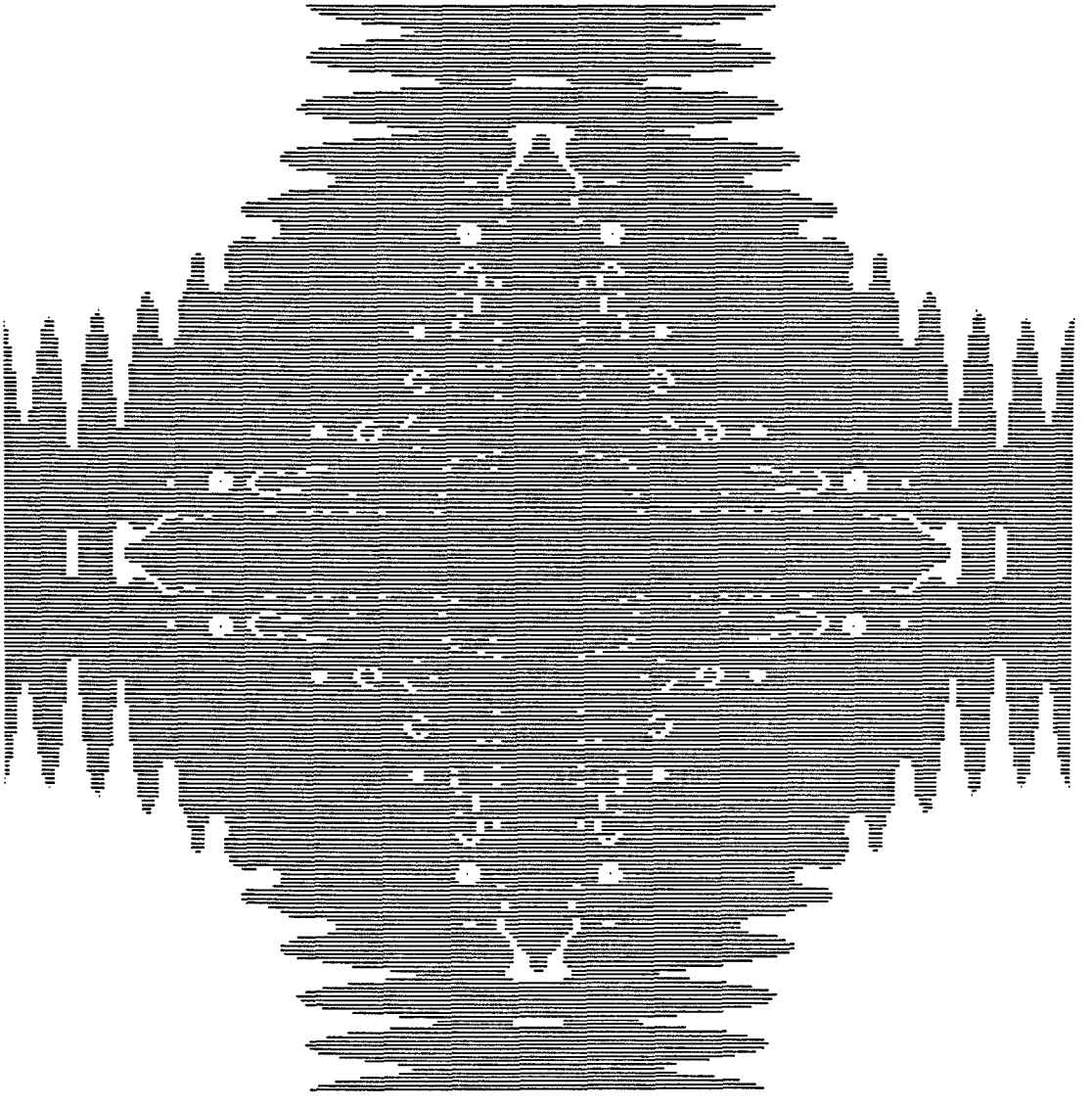


Figure 2.16: Magnitude of FT of overall system, $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$, $N = 3$, *averaged*, $t = 0.1$

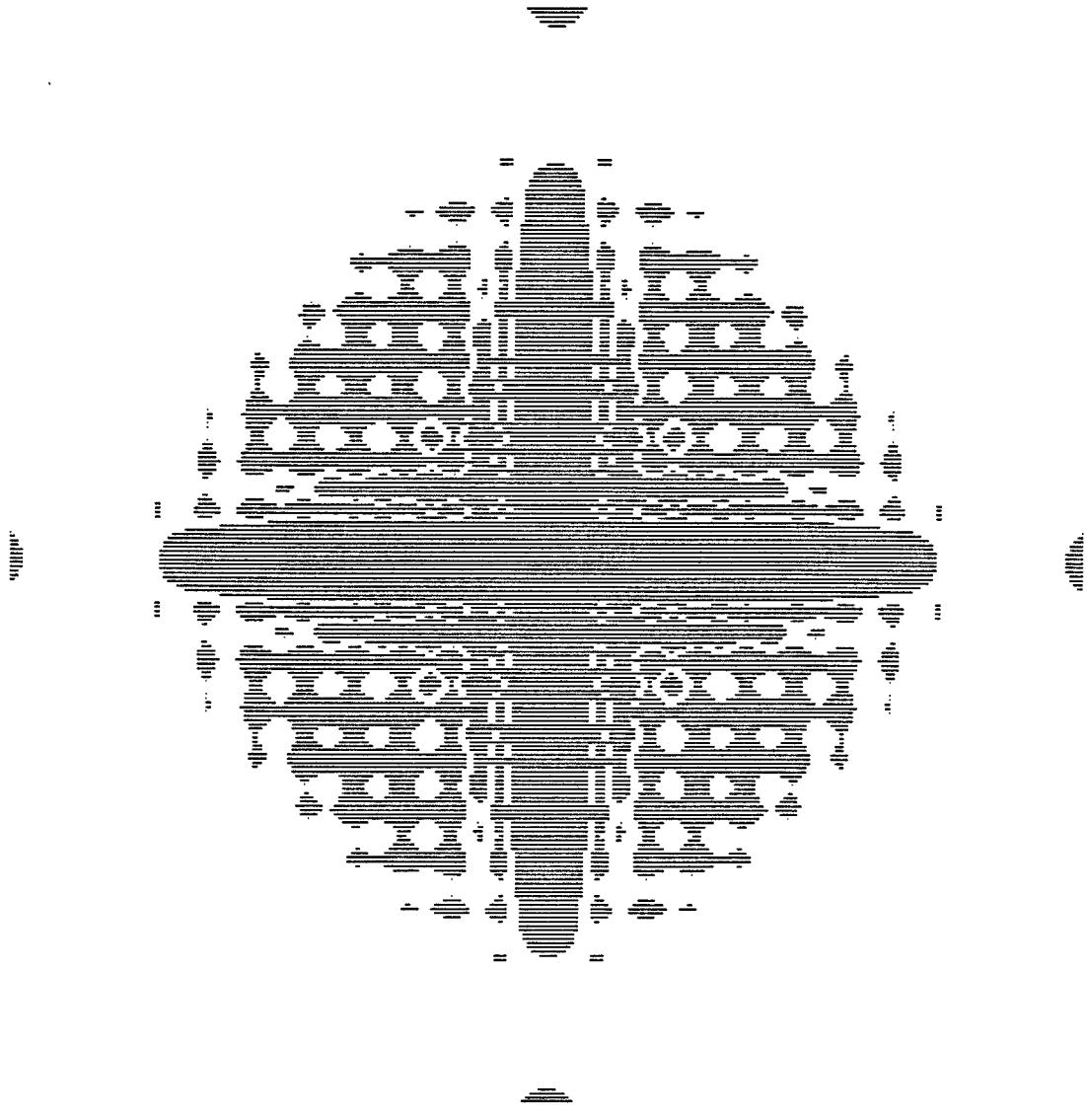


Figure 2.17: Magnitude of FT of overall system, $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$, $N = 3$, *averaged*, $t = 0.5$

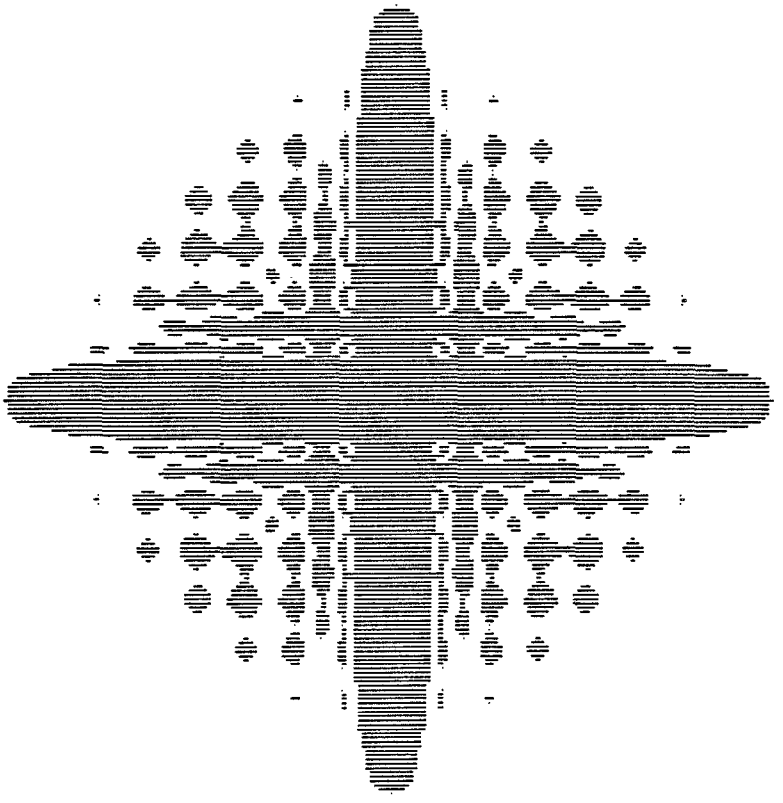


Figure 2.18: Magnitude of FT of overall system, $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$, $N = 3$, *averaged*, $t = 1.0$

CHAPTER
THREE

Efficient Computation

3.1 Preliminary Discussion

A very important issue (at least from the point of view of applications) is how efficiently can "reasonably good" approximate deconvolution kernels be computed from raw data such as samples of the frequency response of the convolvers, or, ideally, analytic expressions for the convolution kernels or their Fourier transforms. There are several complex issues involved here, including the basic questions of modeling, data availability, accuracy of measurements (resolution) and others. We will not discuss these issues here but we will assume that a sufficiently accurate model of the convolvers is available. Since we will work with *samples* of the Fourier transforms of the convolution kernels the data set will eventually be discrete. We remark however that the algorithm requires precise knowledge of the common zeros of the very-well behaved subfamily (and is in fact sensitive to errors in the location of these common zeros).

It is important that care is exercised in order to minimize the uncertainty

with regard to the location of zeros due to imperfect sampling or measurement noise. Of course these problems are eliminated once analytic expressions are known.

As soon as a sufficient model has been established the actual computation seems quite straightforward; one basically needs to calculate pointwise approximations to an infinite sum, that is, approximate an infinite sum for every point over a finite two-dimensional grid of frequencies. For reasonable resolution and degree of approximation this computation can simply be overwhelming! A typical set-up for simulations throughout this work has been as follows: 256×256 individual frequencies and 3200 terms/point for each frequency. The computation of each term requires a relatively large number of floating point operations by itself. Therefore the task becomes time-consuming for an ordinary sequential machine.

One usually needs to apply a trial-and-error approach and modify the window parameters (or the window itself) in order to optimize various system characteristics such as S/N ratio vs. BW. This practice is an integral part of common engineering trade. The possibility of having to re-run the whole computation even for a few times is certainly alarming! Alas, it does not have to be; the problem at hand is inherently massively parallel and can be very efficiently solved using a special machine architecture that already exists commercially. The key observation is that the computation for each frequency point is independent from the corresponding computation for every other point.

3.2 The Data Parallel Architecture

The data parallel architecture is a combination of data parallel software and hardware that supports parallel data-element-wise processing of large *uniform* data structures, such as arrays or fields. It differs from conventional (sequential) architectures in some fundamental aspects while it retains much of the familiar underlying structure of a sequential computing machine, such as uniform address space, centralized program control and layered protocol communications.

Simply put, Data Parallel computing associates one processor with each data element. Upon instruction from the supervising central control unit, each processor operates on its associated data element, i.e. all processors execute the *same* sequence of operations each on *its own* data element in a *synchronous* fashion. This computing style exploits the natural computational parallelism inherent in many data-intensive problems. It can significantly decrease the execution time of a problem and simplify the design of numerical algorithms. In the best case (and our problem is in this class of problems) execution time can be reduced in proportion to the number of data elements in the computation; programming effort can be reduced in proportion to the complexity of expressing a naturally parallel problem in a sequential manner. The final program code becomes extremely compact, natural and expressive. As an example, matrix addition can be coded simply as: A = B + C. The execution of a statement of this form is performed in one addition cycle; all index loops can be eliminated.

Commercially available machines that support the data parallel model typically offer extensive inter-processor communication capabilities usually via an elaborate interconnection network that is synchronized to various extends. While this kind of service is critical for many applications it is *not necessary* for the problem under consideration. Global resource allocation and synchronization is always a must for all data parallel applications and thus a mechanism that offers such services is considered to be an integral part of the data parallel machine.

The particular machine used throughout this work has been the Thinking Machines Corp. Connection Machine CM-2 system. Next we move on to a brief overview of this particular machine. This will make our discussion concrete and give the reader a flavor of the actual computation.

3.2.1 The Connection Machine System

Thinking Machines Corporation Connection Machine model CM-2 is an advanced, highly sophisticated machine that supports the Data Parallel Computing model. Data parallel operations are implemented directly in hardware. In its full configuration the CM-2 system parallel processing unit contains 65,536 data processors, logically subdivided into four clusters of 16,384 processors each. Each data processor contains a separate Arithmetic and Logic Unit (ALU), 256 Kbits of bit-addressable local memory, an optional floating point accelerator, an I/O interface as well as a number of interprocessor communication interfaces. Each cluster (or segment) of 16,384 data processors is attached to a sequencer

that controls the segment. Up to four front-end computers can be attached to the system, one front end computer to each individual sequencer. Alternatively two or four sequencers can be assigned to a single front end machine. A 4 $\times$ 4 crossbar switch (front-end Nexus) reconfigures the front-end to sequencer assignments.

Arbitrary point-to-point communications are permitted by means of special-purpose hardware, called *The Router.* Message passing can occur in parallel; all processors can simultaneously send and receive messages via mailboxes that reside in their local memories. A finer system called *The NEWS Grid* implements a nearest-neighbor communication scheme that is much more efficient than the general router mechanism. The NEWS Grid is realized in hardware too. The Grid operates via a permutation circuit. This permutation circuit has another mode of operation, known as *Direct Hypercube Access* that facilitates the design of rather complex but quite regular communication patterns. The overall architecture of the Connection Machine can be seen in figure 3.1.

The data structure to be operated upon is uniformly spread over the data processor grid of one, two or four clusters. Each data element of the structure resides in the local memory of a data processor. If the data structure is bigger than the actual number of processors available (which is the normal case), a virtual processor mechanism becomes active. As a result each data processor is timeshared between two or more tasks and its associated local memory is sliced into a proportional number of equal-length segments. From the user point of

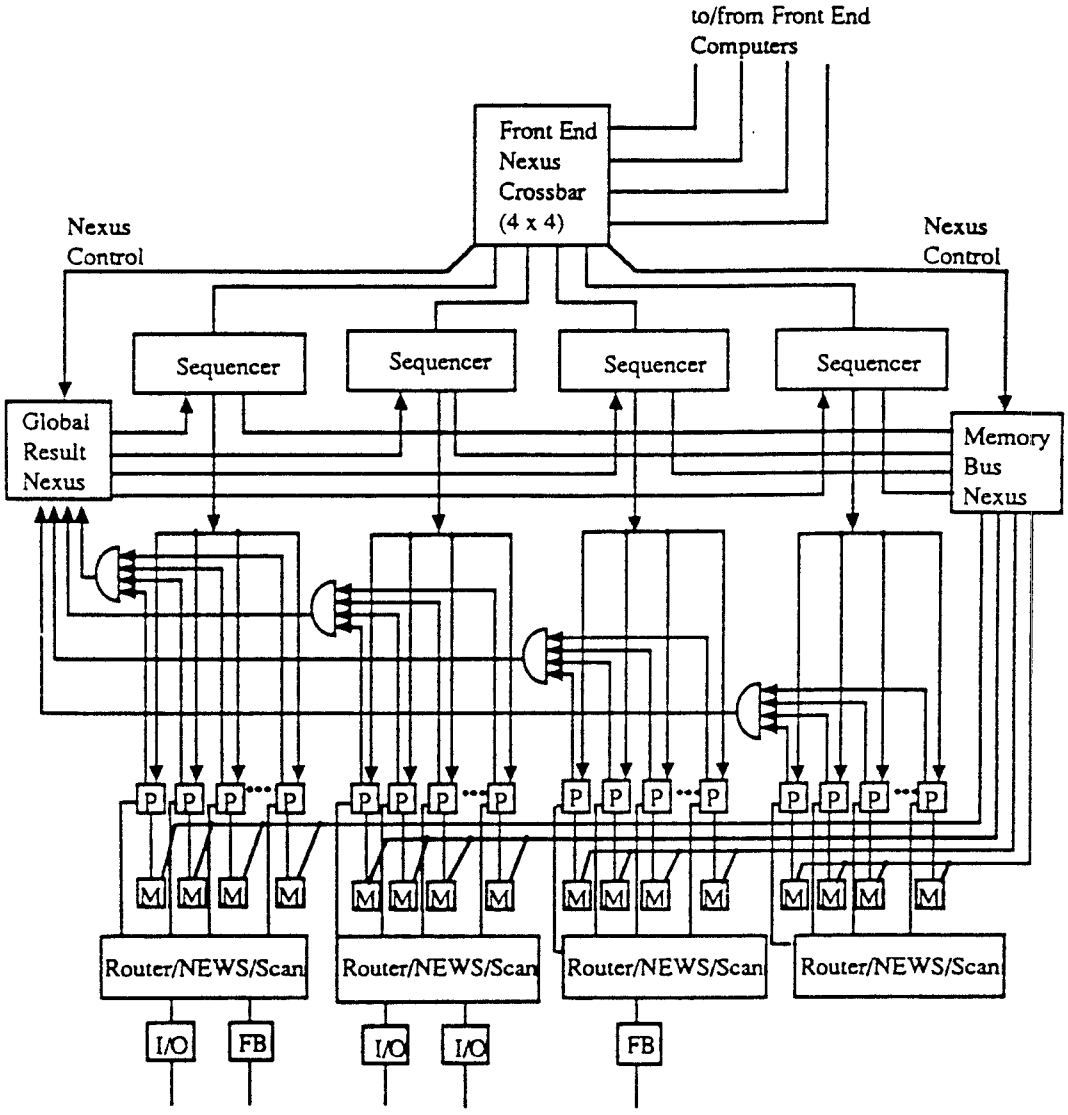


Figure 3.1: Connection Machine model CM-2 Architecture

view this process is transparent , except for the obvious system slowdown. The net effect is that the user is always presented with enough data processors to accommodate his application.

All program code resides in the memory of the front-end computer. Instructions pertaining to the CM-2 unit are broadcasted as needed, from the front-end to the sequencer and onto the individual processors. All serial code is executed by the front-end machine. Input-Output transactions between the processors and special mass storage devices are handled by the interprocessor communication services and each cluster has its own I/O port and associated frame buffer.

Application programs are developed on the front-end machine using familiar tools (editors, debuggers, etc.). The front end can either be a DEC VAX machine running UNIX or a LISP SYMBOLICS machine. Parallel extensions of the popular programming languages C, Fortran and Lisp are supported, in addition to a low-level language known as *PARIS* (PARallel Instruction Set). The actual language extensions are minimal, depending on context to distinguish scalar from parallel operations. "Loaded" versions of all familiar functions and constructs are provided. These enable the use of a rather abstract programming style that does not require explicit identification of parallel operations.

The most critical part of any parallel algorithm is *synchronization.* The solution of any non-trivial task synchronization problem constitutes a formidable exercise in its own right. Typical approaches concentrate on the use of special synchronization flags, called *semaphors.* This can place a considerable load on

the part of the programmer and severely complicate programs that are otherwise quite straightforward. It must be noted that even if interprocessor communications during runtime are not required, global data access and resource utilization must be organized (consider for example what will happen if two or more processors try to read/write data from global memory; the results can be totally unpredictable).

Quite fortunately *all* synchronization is handled by the Connection Machine itself and is transparent to the user. Furthermore the following rule of thumb applies: portions of parallel code are executed in a *"as-if-serial"* manner. That is, although the actual operations are executed in parallel, the net results are completely predictable, as if the operations were executed in a sequential (but unknown) order. This is important when one executes a many-to-one operation. For example, consider the following case: one needs to compute the sum of the entries of an N-vector V. Suppose that the entries are spread across N data processors. Then one could instruct each processor to read a predefined memory location, add its associated entry to it and store the result back to the predefined memory location. With no synchronization one could end up with anything from the correct sum to just one element of V, with everything in between. The concept of "as-if-serial" execution guarantees that one will get the correct sum.

The preceeding discussion should have made clear that this is a rather user-friendly machine, especially when one considers its nature and complexity. The Connection Machine virtually behaves as a very powerful extension of the front-

end sequential machine, each executing these parts of the code that it is well suited for.

The programming language used for this work has been C*, the parallel extension of the standard C draft supported by the Connection Machine System. This dialect of the language closely resembles a relatively unknown but quite powerful earlier variation of standard C, known as C^{++}. C* itself is elegant and noteworthy for its compactness and clarity of expression. We will very briefly review only those elements of C^* which are directly needed for the problem at hand. The reader is referred to the Connection Machine literature for more details [29] .

3.2.2 The C* Language

The C* model is an extension of the plain C model. C* extends C by having many processors instead of one, all executing the same instruction stream. The C* model may be summarized as follows: C* allows the programmer to use *lots of processors* of an otherwise *conventional nature*, operating within a *uniform address space* in a *synchronous execution mode.* Except for the fact that no code is stored in the memory of a data processor, local memory layout of each data processor is conventional.

Data processor memory layout can be informally described as a C struct. At any given time the active set of processors within a particular cluster has a uniform memory layout. In C* a structure type that describes the memory

layout of a data processor is called a *domain.* The notion of a domain is an outgrowth of the *class* structure of C^{++}. We can declare *functions* as members of a domain in much the same way we declare variables within the domain. In C*, all code is divided into two kinds: serial and parallel. Code that belongs to a domain is parallel and is executed by many data processors *at once*; all other code is serial and is executed sequentially by the front-end machine. All data is also divided into two kinds: scalar and parallel. These are described using two new keywords: *mono* and *poly* respectively. Within a code segment that pertains to a domain the default is poly; within serial code it is mono. A scalar value is automatically replicated where necessary to form a parallel value.

Typically, a large number of processors within a cluster are configured with the same memory layout. One simple example follows:

```
domain processor {
                    float x;
                    float y;
                }; domain processor PROC[1000];
```

The above sequence configures a thousand processors according to the domain processor structure. Each processor PROC[i], $i = 1, \cdots, 1000$, constitutes an *instance* of the domain processor. Parallel computation can be initiated using the *selection statement*:

$$[\textit{domain processor}] \cdot \textit{substatement}$$

The substatement appearing right of the dot is executed for all instances of the domain processor (i.e. all 1000 processors) *at once*, i.e. a selection statement activates all instances of the specified domain and then simultaneously executes a substatement. If the substatement contains conditional constructs such as:

if (expression) statement *else* statement

or *while* (expression) statement

then the expression is treated as a *poly* value, so that each active domain instance has its own value for the test. Instances that calculate the value zero become inactive; instances that calculate a non-zero value execute the substatement and then loop. The *while* loop completes if and when the set of active instances becomes empty.

An example program follows: Suppose we want to compute the sample values of the function $f(x) = x\sin(x)$ at 1000 points. This can be done very efficiently by assigning each x value to an individual processor and then computing all y values *at once* using a selection statement. The complete code follows and it is self explanatory:

```
# include <math.hs>


domain processor {
                    float x,y;
                  }
```

```
domain processor PROC[1000];


void processor::f()
{ y=x*sin(x); }


void setupvalues()
{ int i; float t;
  t=0.0;
  for (i=0;i<=999;i++) {
    t+=0.001;
    PROC[i].x=t;
    }
}


main()
{ setupvalues();
  [domain processor].{f();} /* init. parallel processing*/
}
```

The declaration: void processor::f(), specifies the function f to operate on *specific instances* of the domain processor. Thus the variables x and y that appear within the body of $f()$ implicitly refer to the specific instance that f operates upon (recall the as-if-sequential rule). After the program completes its

execution the y values reside in the local memories of the processors and can be accessed as members of an array of struct simply as: $PROC[i].y$.

The preceeding example serves a double purpose; it familiarizes the reader with the fundamentals of C* programming and it basically provides the skeleton of what is needed for the coding of our target computation.

3.3 Optimizing the computation of the deconvolution kernels - Grid Layouts

With the Connection Machine Data Parallel Architecture in mind, the next task is to optimize the target computation with respect to various efficiency considerations. Next we stress the most important facts that need to be taken into account, based on our model problem.

1. Symmetry of Deconvolution kernels in the transform domain. As a result we only need compute the pointwise values of the corresponding Fourier Transforms in the upper-right frequency quadrant.

2. For each frequency pair, (z_1, z_2), careful reshuffling of summation terms (over the nullset $\mathcal{Z}$) can result in a reduced number of required floating point operations.

3. For each frequency pair, (z_1, z_2), and each nullset point (ζ_1, ζ_2) the corresponding terms for the Fourier Transforms of the three deconvolvers *share a common factor.* Therefore this factor need only be computed once for

all three uses. Recall that for $i = 1, 2, 3$ we have that:

$$\widehat{h}_i(z_1, z_2) = \sum_{(\zeta_1,\zeta_2)\in\mathcal{Z}} \frac{\widehat{u}(\zeta_1, \zeta_2)}{J(\zeta_1, \zeta_2)\widehat{f}_3(\zeta_1, \zeta_2)} \cdot \frac{1}{(z_1 - \zeta_1)(z_2 - \zeta_2)} \cdot C_i(z_1, z_2, \zeta_1, \zeta_2) \tag{3.3.1}$$

It is worth noting that this common factor carries out much of the computation involved. Hence *for a given point* (z_1, z_2) the computations $\widehat{h}_1(z_1, z_2)$, $\widehat{h}_2(z_1, z_2)$, $\widehat{h}_3(z_1, z_2)$ are *strongly related*; this fact can be effectively used to our advantage.

Fact (1) reduces the complexity by $\frac{1}{4}$. Fact (2) by a factor close to $\frac{1}{4}$. Observation (3) can result in reductions of up to $\frac{2}{3}$, depending on the floating point function library used. Hence the total gain is in the order of $\frac{2}{48} = \frac{1}{24}$!

Returning to the specific computing model and assuming "enough" processors what would be the most efficient thing to do? One can employ parallelism at various different levels. A very efficient way to attack the problem would be as follows: For each pair of frequencies, (z_1, z_2), in the upper-right transform quadrant assign *one data processor to each nullpoint* (ζ_1, ζ_2), and use the NEWS GRID nearest neighbor communication facility to compute partial sums rowwise along the grid. This scheme should be replicated for all three kernels and for all frequency pairs (z_1, z_2) in the upper right transform quadrant. The data processor grid layout would be as in figure 3.2.

In figure 3.2 • denotes a data processor, and + denotes floating point addition. This grid configuration is *not optimal* in terms of time efficiency because it does not make use of observation (3). It is nevertheless useful because it

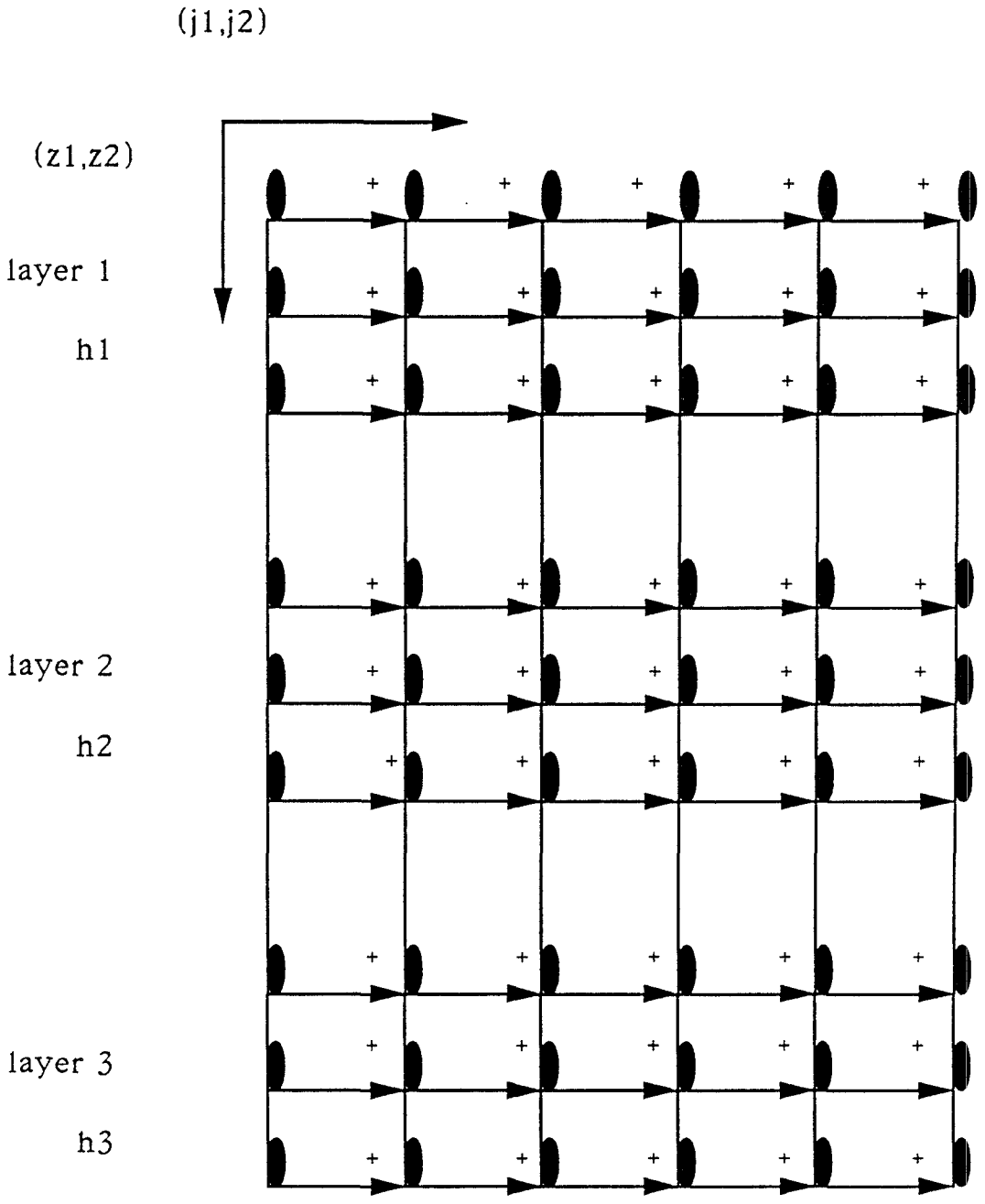


Figure 3.2: Layered grid configuration for distributed computation, Type I

requires a smaller number of cells (data processors) than the one that follows. Next we add a fourth computing layer that calculates the factor $\frac{\widehat{u}(\zeta_1,\zeta_2)}{J(\zeta_1,\zeta_2)\widehat{f_3}(\zeta_1,\zeta_2)}$ for each pair (ζ_1, ζ_2) of the nullpoint set and feeds the result to the appropriate computing cells. This scheme makes *partial use* of observation (3). This time we need to introduce *vertical cell communications* to enable this new type of transaction to take place. This (type II) grid setup is depicted in figure 3.3.

In figure 3.3 • denotes a data processor, $\rightarrow$ denotes interprocessor communication, and $+$ denotes floating point addition. The grid operates as follows: While Layer O computes the factors $\frac{\widehat{u}(\zeta_1,\zeta_2)}{J(\zeta_1,\zeta_2)\widehat{f_3}(\zeta_1,\zeta_2)}$ the cells of Layers 1-3 compute the factors $\frac{C_i(z_1,z_2,\zeta_1,\zeta_2)}{(z_1-\zeta_1)(z_2-\zeta_2)}$. When both computations are completed a *column-wise parallel fetch transaction*, takes place after which each Layer 1-3 cell multiplies the fetched value with its result and then a *row-wise* $+$ *transaction* takes place. Results rest in right most column. Notice that this scheme actually hits the bottom line in parallelism; We cannot *generically* reduce each computing cell unless we move into single-instruction level parallelism.

Both these strategies are highly efficient but very much out of reach of current technology and the Connection Machine capabilities in particular, at least for reasonable resolution and accuracy of approximation. Assuming a resolution of 256×256 frequency points we need to compute $3 \times \frac{256\times256}{4}$ point values (3 kernels, only need upper-right transform quadrant). A minimal nullset cardinality is of the order of 200. Using fact (2) we can reduce this by 1/4 resulting in 50 nullpoints. Then both grids require approximately 2.5×10^6 processors!

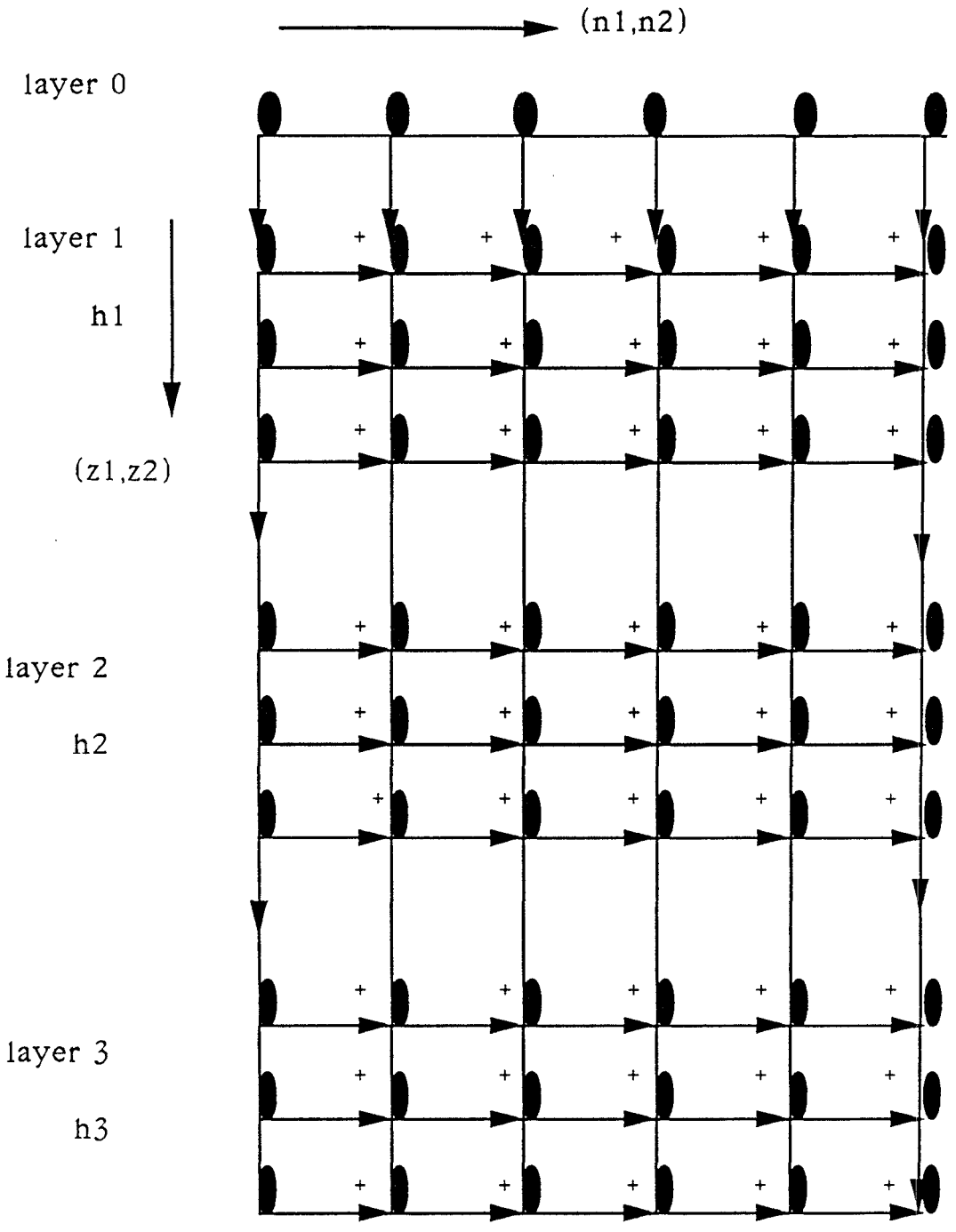


Figure 3.3: Layered grid configuration for distributed computation, Type II

While this level of parallelism may be reachable in the future, it is out of the question now. Therefore one needs to relax the computing power requirements. The proposed grid configurations, although impractical, have demonstrated the massively parallel nature of the problem. We now turn to a more modest strategy.

In its full configuration the Connection Machine model CM-2 employs 64K processors. With a frequency resolution of 512×512 points we can simply assign each processor the task of computing one point value for all three kernels. This requires $\frac{512 \times 512}{4} = 64$K processors. This way *no interprocessor communication is needed* and the size of the nullset is of little practical importance, because it does not affect the number of processors required (only affects the execution time). Thus quite large nullsets can be easily accommodated. Notice that since each processor computes a specific point value *for all three kernels*, observation (3) can easily be exploited.

The appendix lists the C* source code that implements this grid setup for the model problem that we have considered (characteristic functions over squares), a resolution of 256×256 frequency points, a frequency step of $0.171 rads/sec$, and a nullset of 3200 points. For simplicity purposes the whole spectrum is computed (i.e. all four frequency quadrants are independently computed – this facilitates subsequent storage). A directional frequency window is used, of the type discussed in pp. 28-29 with $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$ and N=3. In a fully configured Connection Machine with 64K processors the run time (excluding I/O) is

around four minutes (for the upper right hand quadrant only). Simulation results throughout this work have been produced using this program as a skeleton. The code itself is fairly straightforward and the experienced C user will have no problems understanding it. Finally note that this program does not average the solutions, and this has to be done separately.

3.4 Contributions and Future Research

We have considered the problem of signal deconvolution, and the properties of a specific solution to the two dimensional Analytic Bezout Equation in particular. We have focused on discrete time, finite bandwidth approximations. The major results are summarized here, and an outline for future research is given in what follows.

The one-dimensional case has been studied and general analytic expressions for the deconvolvers have been devised, both in the frequency domain and in the space domain. This important case serves as a vehicle for understanding the more involved two dimensional case and provides useful insight into much of the trade-off involved.

A specific example for the two dimensional case serves as a model problem upon which we build subsequent developments. We have chosen the case of characteristic functions of three squares of suitably chosen size, over $\mathcal{R}^2$. This family can be easily proven to satisfy the strong coprimeness and very-well behavior conditions required by Theorem 1.3. Analytic expressions for the deconvolution

kernels for this model problem have been devised, both in the frequency domain and in the time domain. The properties of these solutions have been investigated. Support claims have been verified. It has been pointed out that any finite computation may cripple the performance due to spurious asymmetries that die out very slowly. The cause of this (not immediately apparent) problem has been traced back to the original interpolation formulas and ways to deal with this problem have been proposed. The use of appropriate frequency windowing has been proposed as a means of forcing the residue term to converge faster while achieving other desirable properties such as noise robustness and good ripple characteristics.

The computability of reasonably good approximate deconvolvers has been demonstrated. Efficiency considerations have been addressed and a number of different Data Parallel grid layouts have been devised that allow for interactive design and trade-off in close to real time.

These ideas have an extremely wide range of applications. The ultimate goal would be to connect this framework with image understanding and machine perception ideas. It is interesting to note the fact that work in these two inherently interacting fields has in fact followed parallel, and, in some sense, diverging paths. The goal of low-level signal processing is to achieve wide bandwidth and good quantitative end-user characteristics, whereas any image understanding system (and pattern recognition scheme in particular) is usually rejecting much of the high frequency information content of the input, and concentrates on

particular image features, partly through the use of hearistic rules. Therefore one wonders whether there exists any need to recover such information to start with! Clearly a "Contraction point" must be revealed, and this should be made an essential design goal.

Towards this end, a promising approach would be to use the concepts and principles of Minkowski set theory and Mathematical Morphology, which provide a valuable link between low-level signal processing and shape representation and pattern classification ideas. The incorporation of such mathematical devices into the present context of signal reconstruction ideas should constitute a major step forward towards a unified approach to signal processing and understanding via translation-invariant systems [19,20].

The flexibility of morphological operators has been established in a variety of applications. Finally it has to be remarked that such operations are nonlinear and thus are more suited to attack the general inverse problem rather than the (linear) deconvolution problem. Hence a top-down approach will be most appropriate [26,21].

APPENDIX

List of C* Source Code

```
#include <stdio.hs>

#include <math.hs>


#define riza3 1.732050808

#define riza2 1.414213562

#define pi 3.141592654

#define fstep 0.171805848;



domain point {

  float z1, z2, h1, h2, h3;

};


domain point frame[256][256];


float point::f1()
```

```
{ return((4/(z1*z2))*(sin(riza3*z1))*(sin(riza3*z2))); }


float point::f2()

{ return((4/(z1*z2))*(sin(riza2*z1))*(sin(riza2*z2))); }


float point::f3()

{ return((4/(z1*z2))*(sin(z1))*(sin(z2))); }


float point::jf3(s1,s2)

poly float s1, s2;

{ poly float a, b1, b2, b3, b, c1, c2, c3, c, d, g;

  a = 16/(s1*s1*s1*s2*s2*s2);

  b1 = (sin(riza3*s2))*(sin(riza2*s1));

  b2 = riza3*s1*cos(riza3*s1) - sin(riza3*s1);

  b3 = riza2*s2*cos(riza2*s2) - sin(riza2*s2);

  b = b1*b2*b3;

  c1 = (sin(riza3*s1))*(sin(riza2*s2));

  c2 = riza3*s2*cos(riza3*s2) - sin(riza3*s2);

  c3 = riza2*s1*cos(riza2*s1) - sin(riza2*s1);

  c = c1*c2*c3;

  d = a*(b-c);

  g = d*(4/(s1*s2))*(sin(s1))*(sin(s2));
```

```
  return(g);
}


float point::c1(s1,s2)
poly float s1,s2;
{ poly float a, b, c, d;
  a = (4/(z1*s2))*(sin(riza2*z1))*(sin(riza2*s2));
  b = f3() - (4/(s1*s2))*(sin(s1))*(sin(s2));
  c = f2();
  d = (4/(z1*s2))*(sin(z1))*(sin(s2))
      - (4/(s1*s2))*(sin(s1))*(sin(s2));
  return(a*b-c*d);
}


float point::c2(s1,s2)
poly float s1, s2;
{ poly float a, b, c, d;
  a = f1();
  b = (4/(z1*s2))*(sin(z1))*(sin(s2))
      - (4/(s1*s2))*(sin(s1))*(sin(s2));
  c = (4/(z1*s2))*(sin(riza3*z1))*(sin(riza3*s2));
  d = (4/(s1*s2))*(sin(s1))*(sin(s2)) - f3();
```

```
  return(a*b+c*d);
}

float point::c3(s1,s2)
poly float s1, s2;
{ poly float a, b, c, d;
  a = (4/(z1*s2))*(sin(riza3*z1))*(sin(riza3*s2));
  b = f2();
  c = f1();
  d = (4/(z1*s2))*(sin(riza2*z1))*(sin(riza2*s2));
  return(a*b-c*d);
}

float point::uhat(s1,s2)
poly float s1, s2;
{ poly float uh;
  /* epsilon1 = 0.1, epsilon2 = 0.5, N = 3 for this realization */
  uh = (sin(s1/30.0))*(sin(s2/6.0))*(180.0/(s1*s2));
  uh = uh*uh*uh;
  return(uh);
 }
```

```
void point::h()      /* Parallel pointwise computation */
{ poly float s1, s2, u, w;
  poly int j,k;
  h1 = 0.0; h2 = 0.0; h3 = 0.0;
  for (j=1;j<=20;j++) {
    for (k=1;k<=20;k++) {
      s1 = (j*pi)/riza3;
      s2 = (k*pi)/riza2;
      w = (1/jf3(s1,s2))*(4*s1*s2/((z1*z1-s1*s1)*(z2*z2-s2*s2)));
      u = uhat(s1,s2);
      h1+=w*c1(s1,s2)*u;
      h2+=w*c2(s1,s2)*u;
      h3+=w*c3(s1,s2)*u;
      s1 = (j*pi)/riza2;
      s2 = (k*pi)/riza3;
      w = (1/jf3(s1,s2))*(4*s1*s2/((z1*z1-s1*s1)*(z2*z2-s2*s2)));
      u = uhat(s1,s2);
      h1+=w*c1(s1,s2)*u;
      h2+=w*c2(s1,s2)*u;
      h3+=w*c3(s1,s2)*u;
      }
    }
```

```
}



void set_up_frame()
{ int i, j;
  float l1, l2;
  l1 = -21.81934273 - fstep;
  l2 = -21.81934273 - fstep;
  for (i=0;i<=255;i++) {
    l1 = l1 + fstep;
    for (j=0;j<=255;j++) {
      l2 = l2 + fstep;
      frame[i][j].z1 = l1;
      frame[i][j].z2 = l2;
      }
    }
}


void store()
{ int i, j, dummy, ioerr;
  FILE *fp1, *fp2, *fp3, *fopen();
  fp1 = fopen("h13200.dat","w");
```

```
fp2 = fopen("h23200.dat","w");
fp3 = fopen("h33200.dat","w");
for (i=0;i<=255;i++) {
  for (j=0;j<=255;j++) {
    fprintf(fp1,"%f\n",frame[i][j].h1);
    fprintf(fp2,"%f\n",frame[i][j].h2);
    fprintf(fp3,"%f\n",frame[i][j].h3);
    }
  /* flush buffers */

  do {
       ioerr = fflush(fp1);
       if (ioerr) { for (dummy=0;dummy<10000;dummy++) {} }
     }
  while(ioerr);
  do {
       ioerr = fflush(fp2);
       if (ioerr) { for (dummy=0;dummy<10000;dummy++) {} }
     }
  while(ioerr);
  do {
       ioerr = fflush(fp3);
```

```
      if (ioerr) { for (dummy=0;dummy<10000;dummy++) {} }
   }
while(ioerr);

/* pad imag. part with zeros */

for (j=0;j<=255;j++) { fprintf(fp1,"%f\n",0.0);
                       fprintf(fp2,"%f\n",0.0);
                       fprintf(fp3,"%f\n",0.0);
                     }

/* flush buffers */

do {
      ioerr = fflush(fp1);
      if (ioerr) { for (dummy=0;dummy<10000;dummy++) {} }
   }
while(ioerr);
do {
      ioerr = fflush(fp2);
      if (ioerr) { for (dummy=0;dummy<10000;dummy++) {} }
   }
```

```
    while(ioerr);
    do {
         ioerr = fflush(fp3);
         if (ioerr) { for (dummy=0;dummy<10000;dummy++) {} }
       }
    while(ioerr);
    }
 fclose(fp1); fclose(fp2); fclose(fp3);
}


main() {
  printf("set_up_frame...\n");
  set_up_frame(); /* set up frequencies in frame */
  printf("start parallel processing...\n");
  [domain point].{ h(); } /* initiate parallel processing */
  printf("...printing output files\n");
  store();
  printf(".Completed.\n");
}
```

BIBLIOGRAPHY

[1] L. Auslander and F. A. Grunbaum. The Fourier Transform and the Discrete Fourier Transform. *Inverse Problems*, 5(2):149–164, 1989.

[2] C. A. Berenstein, P. S. Krishnaprasad, and B. A. Taylor. Deconvolution methods for multisensors. Technical Report DTIC ADA 152351, University of Maryland, 1984.

[3] C. A. Berenstein and D. Struppa. On explicit solutions to the Bezout equation. *Systems Control Letters*, 4:33–39, 1984.

[4] C. A. Berenstein and D. Struppa. 1-Inverses for polynomial matrices of non constant rank. *Systems Control Letters*, pages 309–314, 1986.

[5] C. A. Berenstein and D. Struppa. Small degree solutions for the polynomial Bezout equation. *Linear Algebra Appl.*, 98:41–56, 1988.

[6] C. A. Berenstein, B. A. Taylor, and A. Yger. Sur quelques formules explicites de deconvolution. *J. Opt*, 14:75–82, 1983.

[7] C. A. Berenstein and A. Yger. Le probleme de la deconvolution. *J. Funct. Anal.*, 54:113–160, 1983.

[8] C. A. Berenstein and A. Yger. Ideals generated by exponential polynomials. *Adv. in Math.*, 60:1–80, 1986.

[9] C. A. Berenstein and A. Yger. Analytic Bezout Identities. *Advances in Applied Mathematics*, 10:51–74, 1989.

[10] B. Berndtsson. A formula for interpolation and division in C^n. *Math. Anal.*, 263:395–418, 1983.

[11] N. K. Bose. *Applied Multidimensional Systems Theory.* Reidel, Dordrecht, 1984.

[12] W. D. Brownawell. Bounds for the degrees in the Nullstellensatz. *Ann. of Math.*, 126:577–591, 1987.

[13] B. Buchberger. An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory.* Reidel, Dordrecht, 1985.

[14] A. R. Davies, M. Iqbal, K. Maleknejad, and T. C. Redshaw. A comparison of statistical regularization and Fourier extrapolation methods for numerical deconvolution. In P. Deuflhard and E. Hainer, editors, *Proc. of Int. Workshop on Numerical Treatment of Inverse Problems in Differential and Integral Equations, Heidelberg FRG, Aug 30 - Sept 3, 1982*, Stutgart, 1983. Birkhauser.

[15] L. Ehrenpreis. *Fourier Analysis in Several Complex Variables.* Interscience. Wiley, New York, 1970.

[16] L. Hormander. Generators for some rings of analytic functions. *Bull. Amer. Math. Soc.*, 73:943–949, 1967.

[17] L. Hormander. *Linear Partial Differential Operators.* Springer-Verlag, Berlin, Heidelberg, New York, third edition, 1969.

[18] H. J. Landau and H. O. Pollak. Prolate Spheroidal Wave Functions, Fourier Analysis and Uncertainty-II. *Bell System Tech. J.*, pages 65–84, Jan. 1961.

[19] P. Maragos and R. W. Schafer. Morphological filters - part I : Their set-theoretic analysis and relations to linear shift-invariant filters. *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-35(8):1153–1169, Aug. 1987.

[20] P. Maragos and R. W. Schafer. Morphological filters - part II : Their relations to median, order-statistic, and stack filters. *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-35(8):1170–1184, Aug. 1987.

[21] G. Matheron. *Random Sets and Integral Geometry.* Wiley, New York, 1975.

[22] R. E. Molzon, B. Shiffman, and N. Sibony. Average growth estimates for hyperplane sections of entire analytic sets. *Math. Ann.*, 257:43–59, 1981.

[23] V. P. Palamodov. *Linear Differential Equations with Constant Coefficients.* Springer-Verlag, New York, 1970.

[24] E. V. Patrick. Deconvolution for the case of multiple characteristic functions of cubes in R^n. Technical report, Systems Research Center, University of Maryland, 1986.

[25] L. Schwartz. *Theorie des Distributions.* Herman, Paris, 1950.

[26] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[27] D. Slepian. On Bandwidth. *Proc. of the IEEE*, 64(3), March 1976.

[28] D. Slepian and H. O. Pollak. Prolate Spheroidal Wave Functions, Fourier Analysis and Uncertainty-I. *Bell System Tech. J.*, pages 43–63, Jan. 1961.

[29] Thinking Machines Corporation, Cambridge, Massachusetts. *Connection Machine model CM-2 Reference and Technical Summary*, May 1989.

[30] S. A. Tretter. *Introduction to Discrete time Signal Processing.* Wiley, New York, 1976.