SIGNAL RECONSTRUCTION FROM PHASE

by

PRANATHARTHI HARAN

B. Tech, Indian Institute of Technology, Madras, 1981

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

Approved by:

Major Professor

TABLE OF CONTENTS

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

THIS BOOK CONTAINS NUMEROUS PAGES WITH PICTURES THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

LIST OF FIGURES

I. INTRODUCTION

The relationship between the real and imaginary part of the Fourier transform of a causal signal is well known. They are related through the Hilbert transform. However, in general there exists no such relationship between the Fourier transform magnitude and phase. The knowledge of either one is not sufficient to uniquely specify the signal. Under certain conditions, however, these two components are related. For minimum phase signals, the logarithm of the magnitude and the phase form a Hilbert transform pair. Minimum phase signals require that all the poles and zeros of the Z-transform of the signal be within unit circle. This condition is too restrictive in that it reduces the scope of applications. However, using certain windowing techniques, these relationships have been exploited to solve a number of problems in different areas [11].

Recently Hayes et al. [1] have developed a different set of conditions under which the signal can be exactly reconstructed to within a scale factor from the knowledge of either phase or magnitude alone. This ability to reconstruct the signal is useful in a number of applications. For example, in the problem of "blind deconvolution," the observed signal is the convolution of the desired signal with some unknown signal. In the absence of any information about either the desired signal or the distorting signal,

the recovery of the desired signal poses a difficult problem. However, if the distorting signal is known to have a zero phase, then the phase of the desired signal is exactly known. In such cases the signal can be exactly reconstructed from the undistorted phase --e.g., when images are blurred by severely defocused lenses with circular aperture stops [2]. Here, except for phase reversals, the phase of the observed signal is the same as that of the desired signal. Therefore, it is of interest to study the problem of signal reconstruction from phase alone.

In this report we describe the conditions under which the signal can be exactly reconstructed from the phase information alone. We also discuss several algorithms for implementing this reconstruction. One of them is a closed-form solution and it involves the solution of a set of simultaneous equations. There are several iterative methods in which the signal is refined after each iteration. Compared to iterative algorithms, the closed-form solution has the advantage that the desired solution is guaranteed. On the other hand, it requires computing the inverse of an (N-1) x (N-1) matrix, where N is the number of signal points. This may lead to numerical problems and severe round-off errors, particularly as N becomes large. It becomes impractical to use this method for image reconstruction. In such cases an iterative approach is the only viable alternative.

We first illustrate the reconstruction process for some one-dimensional signals using different methods, and make

comparisons of the convergence rate for different methods, for signals of different lengths. Then we apply these techniques to two-dimensional signals, using the two-dimensional transform. We will also reconstruct some images from their phase, and also demonstrate the ability to remove the effect of blurring.

II. UNIQUENESS OF A SEQUENCE OF KNOWN PHASE

In this section the conditions under which a signal can be uniquely specified by its phase are stated. These conditions were developed by Hayes et al. [1], first for one-dimensional signals and later on extended to M dimensions by the same author [3]. In the following discussion, the signals are assumed to be real and their Fourier transforms are assumed to converge, i.e., the region of convergence of their Z-transforms include the unit circle.

Let $x(n)$ and $y(n)$ be two finite length sequences and let $\theta_x(w)$ and $\theta_y(w)$ denote their phase functions. Then we have the following theorem.

Theorem 1: [1]

Let $x(n)$ and $y(n)$ be two finite length sequences whose Z-transforms have no zeros in reciprocal pairs, or on unit circle. If $\theta_x(w) = \theta_y(w)$ for all w, then $x(n) = \beta y(n)$ for some positive constant β. If $\tan \theta_x(w) = \tan \theta_y(w)$ for all w, then $x(n) = \beta y(n)$ for some real constant β.

The conditions of Theorem 1 can be explained as follows. A zero at $Z = Z_0$ and a pole at $Z = 1/Z_0^*$ contribute the same amount to the phase, but different to the magnitude of the Fourier transform. Given phase alone, there is an inherent ambiguity between the zero at $Z = Z_0$ and the pole at $Z = 1/Z_0^*$. We need the finite length sequence condition to remove this ambiguity. The exclusion of zeros occuring in reciprocal

pairs obviates the possibility of zero phase components which can never be recovered from the phase alone.

The dual of Theorem 2, which states the uniqueness conditions for an all pole sequence, is stated next.

Theorem 2: [1]

Let $x(n)$ and $y(n)$ be two sequences whose Z-transforms have no poles in reciprocal pairs and which have finite duration convolution inverses. If $\theta_x(w) = \theta_y(w)$ for all w, then $x(n) = \beta y(n)$ for some positive constant β. If $\tan \theta_x(w) = \tan \theta_y(w)$, then $x(n) = \beta y(n)$ for some real constant β.

If $\tilde{x}(n)$ is the convolution inverse of $x(n)$, then

$$\tilde{x}(n) * x(n) = \delta(n)$$

where $\delta(n)$ is the impulse sequence.

Hence the phase of $\tilde{x}(n)$ is negative of the phase of $x(n)$, and $\tilde{x}(n)$ is uniquely specified by its phase by virtue of Theorem 1. Since $\tilde{x}(n)$ is unique for each $x(n)$, Theorem 2 follows from Theorem 1.

We now describe a conceptual algorithm, which helps to lend insight into Theorems 1 and 2 [1]. It is formulated for Theorem 1, but can be easily extended to Theorem 2.

Let $\theta_x(w)$ and $\hat{\theta}_x(w)$ be the specified phase function (Modulo 2π), and unwrapped[+] phase function respectively. From the conditions related to Theorem 1, $x(z)$ is restricted to be of the form

[+]See appendix A for a description of this term.

$$x(z) = cz^{no} \prod_{k=1}^{N_1} (1-a_k z^{-1}) \prod_{k=1}^{N_2} (1-b_k z) \qquad (1)$$

where c is real, n_o an integer, $|a_k|<1$, $|b_k|<1$ for all k, and $a_k \neq b_l^*$ for all k and l.

The algebraic sign of c is obtained from the value of the phase at w = o, using the fact that c is positive iff $\theta_x(o)$ is zero. The value of n_o is obtained from the unwrapped phase function as

$$n_o = \frac{1}{\pi} [\hat{\theta}_x(\pi) - \hat{\theta}_x(o)] \qquad (2)$$

From the unwrapped phase function and the value of n_o, we form a new phase function $\Phi_x(w)$ as

$$\Phi_x(w) = \hat{\theta}_x(w) - n_o w - \hat{\theta}_x(o) \qquad (3)$$

This corresponds to the transfer function,

$$x_1(z) = |c| \prod_{k=1}^{N_1} (1-a_k z^{-1}) \prod_{k=1}^{N_1} (1-b_k z) \qquad (4)$$

The minimum phase sequence for $x_1(z)$ can be formed by reflecting all zeros outside unit circle as poles within unit circle, and $x_{min}(z)$ is given by

$$x_{min}(z) = |c| \frac{\prod_{k=1}^{N_1} (1-a_k z^{-1})}{\prod_{k=1}^{N_2} (1-b_k^* z^{-1})} \qquad (5)$$

Using the Hilbert transform, $x_{min}(n)$ can be obtained from $\Phi_x(w)$. The a_k and the b_k in (5) are the same as those in (1). Since pole-zero concellations can not occur by virtue of conditions of Theorem 1 $\{a_k \neq b_l^*\}$, the coefficients a_k can

be obtained from the zeros of $x_{min}(z)$ and the b_k can be found from the poles of $x_{min}(z)$.

The conditions of Theorem 1 ensure that pole-zero cancellations do not occur in (5). If the original sequence contained zeros in reciprocal pairs, then the above algorithm may still be applied to recover all but those zeros which occured in reciprocal pairs.

In the above two theorems, we assumed that the phase function was available at all frequencies. In the computer implementation of such algorithms, the phase is known only at a set of N discrete frequencies. As such we state the following theorems, which require that the phase be specified at a sufficient number of discrete frequencies. Here again it is assumed that all sequences are real, and the regions of convergence of their Z-transforms include the unit circle.

Theorem 3: [1]

Let $x(n)$ and $y(n)$ be two finite length sequences which are zero outside the interval $o \leq n \leq N-1$, with Z-transform which have no zeros in reciprocal pairs, or on the unit circle. If $\theta_x(w) = \theta_y(w)$ at N-1 distinct frequencies in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for some positive constant β. If $\tan \theta_x(w) = \tan \theta_y(w)$ at N-1 distinct frequencies in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for a real constant β.

The dual of Theorem 3, for an all pole sequence is the following.

Theorem 4: [1]

Let $x(n)$ and $y(n)$ be two sequences whose Z-transform have no poles in reciprocal pairs and which have convolutional inverses which are zero outside the interval $o \leq n \leq N-1$. If $\theta_x(w) = \theta_y(w)$ at a set of N-1 distinct frequencies in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for some positive constant β. If $\tan \theta_x(w) = \tan \theta_y(w)$ at a set of N-1 distinct frequencies in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for a real constant β.

The theorems we have stated so far imposed certain conditions on the unknown sequence $y(n)$. We would like these conditions to be simpler, so that they can be implemented easily in the reconstruction algorithms. To this end, some additional information about $x(n)$ is needed to guarantee that the sequences obtained from the algorithms satisfy the conditions of Theorem 1. The additional information needed is the first non-zero point of $x(n)$. In such cases, we can show that if $x(n)$ satisfies the conditions of Theorem 1 and if $x(n) = o$ outside the interval $o \leq n \leq N-1$, then $x(n)$ and scaled versions of $x(n)$ are the only sequences which are zero outside the given interval, and have the same phase as $x(n)$.

Theorem 5: [1]

Let $x(n)$ be a sequence which is zero outside the interval

$o \leq n \leq N-1$ with $x(o) \neq o$, and whose Z-transform has no zeros in reciprocal pairs, or on the unit circle. Let $y(n)$ be any sequence which is zero outside the interval $o \leq n \leq N-1$. If $\theta_x(w) = \theta_y(w)$ at N-1 distinct frequencies in the interval $o < w < \pi$, than $x(n) = \beta y(n)$ for some positive constant β. If $\tan \theta_x(w) = \tan \theta_y(w)$ at N-1 distinct frequencies in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for some real constant β.

Theorem 6: [1]

Let $x(n)$ be a sequence whose Z-transform has no poles in reciprocal pairs, or on unit circle, and whose convolution inverse is zero outside the interval $o \leq n \leq N-1$, and non-zero at $n = o$. Let $y(n)$ be any sequence whose convolution inverse is zero outside the interval $o \leq n \leq N-1$. If $\theta_x(w) = \theta_y(w)$ at N-1 distinct frequencies in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for some positive constant β. If $\tan \theta_x(w) = \tan \theta_y(w)$ at N-1 distinct points in the interval $o < w < \pi$, then $x(n) = \beta y(n)$ for some real constant β.

Note that in contrast to Theorem 1 through 4, the above two theorems do not impose any restriction on the zeros of $y(n)$. So $y(n)$ may be any sequence which is zero outside the interval $o \leq n \leq N-1$.

III. CLOSED-FORM SOLUTION

The closed-form solution of x(n) from the given phase function $\theta_x(w)$ follows directly from the definition of $\theta_x(w)$; i.e.,

$$\tan \theta_x(w) = \frac{\text{Imag } [x(w)]}{\text{Real } [x(w)]} \tag{6}$$

Since x(n) is real,

$$FT[x(n)] = \sum_{n=1}^{N-1} x(n) \cos nw - j \sum_{n=1}^{N-1} x(n) \sin nw \tag{7}$$

where FT[·] denotes the Fourier transform.

Thus

$$\tan \theta_x(w) = - \frac{\sum_{n=1}^{N-1} x(n) \sin nw}{\sum_{n=1}^{N-1} x(n) \cos nw} \tag{8}$$

when $\theta_x(w) = \pm\pi/2$, (8) becomes

$$\sum_{n=1}^{N-1} x(n) \cos nw = o \tag{9}$$

When the phase samples are available at N-1 distinct points $w_1, w_2 \ldots w_{N-1}$, where $o < w_k < \pi$ for k = 1,2...N-1, we have,

$$\left[\sum_{n=o}^{N-1} x(n) \cos nw_k \sin \theta_x(w_k)\right] = -\left[\sum_{n=o}^{N-1} x(n) \sin nw_k\right] \cos\theta_x(w_k) \tag{10}$$

or

$$\sum_{n=0}^{N-1} x(n)\{\cos nw_k \cdot \sin \theta_w(w_k) + \sin nw_k \cos \theta_w(w_k)\} = o \qquad (11)$$

That is

$$\sum_{n=1}^{N-1} x(n) \sin [\theta_x(w_k) + nw_k] = -x(o) \sin \theta_x(w_k) \qquad (12)$$

if $w_k \neq \pm \pi/2$

and

$$\sum_{n=1}^{N-1} x(n) \cos nw_k = - x(o) \qquad (13)$$

if $\theta_x(w_k) = \pm \pi/2$.

The above equations represent a set of N-1 linear equations with N-1 unknowns, when the value of x(o) is known. This system of equations could be overdetermined, if the phase is given at a set of M points, where M > N-1. If the value of x(o) is not specified, the set of equations are homogeneous and in any case there will be at least one arbitrary constant, which can be thought of as a scale factor. Equations (12) and (13) can be expressed in matrix form as

$$S\underline{x} = -x(o)\ \underline{b} \qquad (14)$$

where $\underline{x}^T = [x(1)\ x(2)\ \ldots x(N-1)]$

$\underline{b}$ is a column vector where $b_k = \sin \theta_x(w_k)$

and $s_{mn} = \sin [\theta_x(w_m) + nw_m]$ is the element in the mth row and nth column of S.

The solution of (14) is given by

$$\underline{x} = -S^{-1}\, x(o)\, \underline{b} \qquad (16)$$
$$= \alpha S^{-1}\, \underline{b}$$

where $\alpha = -x(o)$.
if S^{-1} exists. If S is overdetermined, the least-squares solution is given by

$$\underline{x} = \alpha (S^T S)^{-1}\, S^T \underline{b} \qquad (17)$$

As stated earlier, this algorithm is useful when the length of the sequence is small. As N becomes large, the problem of round-off errors become severe, and worsens when the matrix $s^T s$ is ill-conditioned. The problems associated with the closed-form solution motivates one to search for alternate algorithms for reconstruction. These algorithms are iterative and usually involve repeated transformations between the time and frequency domains. In all the iterative algorithms we discuss, one typically needs to specify the phase at M points, where $M \geq 2N$, for a sequence of length N. These M points are usually Discrete Fourier transform (DFT) values and the normalized frequency w is between o and 2π. This means that the information in (o, π) is repeated in $(\pi, 2\pi)$, and hence M has to be at least twice as large as N.

IV. ITERATIVE METHODS

We discuss three iterative schemes. In each case the sequence is assumed to be zero outside the interval $o \leq n \leq N-1$ and $x(o) \neq 0$, and we start with an initial estimate $\tilde{x}(n)$ of $x(n)$. The value of $\tilde{x}(n)$ in each iteration is refined. The three methods only differ in the way values of $\tilde{x}(n)$ are updated. They have different convergence rates, and faster convergence usually requires more computation. The first of the iterative methods is the well-known steepest descent method, used to solve a set of linear equations. The second iterative algorithm is similar to the one developed by Gerchberg and Saxton [4], and Fineup [5], for reconstructing a signal from the magnitude information, and also to the algorithm developed by Quatieri [6] for reconstructing a minimum phase signal from only the phase information. The third method is a combination of first two methods and it has an adaptive parameter λ which is chosen to maximize the rate of convergence.

A. Gradient Method

Let $\underline{x}^T = [x(1)\ x(2) \ldots x(N-1)]$ and $\alpha = -x(o)$. Then from (14) we have

$$S\underline{x} = \alpha\underline{b} \tag{18}$$

If the phase is specified at M points, we have M equations and N-1 unknowns. The matrix S is of size M x N-1 and $\underline{b}$ is

a (M x 1) column matrix. The elements of S and $\underline{b}$ are given by

$$b_k = \text{Sin } \theta_x(w_k) \tag{18a}$$

$$Smn = \text{Sin } [\theta_x(w_m) + nw_m] \tag{18b}$$

It is known that the solution of (18) is the minimum point of the quadratic error surface ε, where ε is given as

$$\varepsilon = (S\underline{x} - \alpha\underline{b})^T (S\underline{x} - \alpha\underline{b}) \tag{19}$$

In the gradient method, we start with an initial guess of the solution and update the current solution using the recursion

$$\underline{x}_{k+1} = \underline{x}_k + \mu (-\nabla_x \varepsilon) \tag{20}$$

where μ is a parameter which can be chosen to give the maximum convergence rate and the term within the brackets is the gradient of ε with respect to $\underline{x}$. Using certain matrix manipulations, (20) can be replaced by a set of DFT's which can be computed using Fast Fourier transform (FFT).

Starting with the error surface ε, we have,

$$\nabla_{\underline{x}} \varepsilon = 2S^T (S\underline{x} - \alpha\underline{b}) \tag{21}$$

Elements of S and $\underline{b}$ are given by equations (18a) and (18b). If we use the DFT to obtain the phase, then the w_m are equally spaced and equal to $(m-1)w_o$, where w_o is fundamental frequency. That is

$$S_{mn} = \text{Sin}\,[\theta_x(\overline{m-1}w_o) + (m-1)\,nw_o]$$

$$= \text{Imag}[e^{j\theta_x(\overline{m-1}w_o)} \cdot e^{j(m-1)nw_o}]$$

The second term of the product is an element of the DFT matrix. Since each row of the matrix is multiplied by a single factor $e^{j\theta_x(\overline{m-1}w_o)}$, it can be replaced by premultiplication by a diagonal matrix Φ of size (M x M) whose mth diagonal element is $e^{j\theta_x(\overline{m-1}w_o)}$. A selection matrix R can be used to select only columns 2 through N from the product of the diagonal matrix Φ and the DFT matrix W. This has to be done since S is of size M x (N-1) and the columns correspond to column 2 through N, of the product ΦW. Finally S is given as

$$S = \text{Imag}[\Phi WR] \tag{22}$$

where

$$\Phi = \begin{bmatrix} e^{j\Phi_x(o)} & & & \\ & e^{j\Phi_x(w_o)} & & \Phi \\ & & \ddots & \\ & \Phi & & e^{j\Phi_x(\overline{M-1}w_o)} \end{bmatrix}_{(M \times M)}$$

$$W = [\text{ DFT matrix }]_{M \times M}$$

and

$$R = \begin{bmatrix} o & o & \cdots & o \\ 1 & o & \cdots & o \\ o & 1 & \cdots & o \\ \cdot & & & \\ \cdot & & & 1 \\ o & & \cdots & o \\ o & & \cdots & o \end{bmatrix}_{M \times (N-1)} \text{..Nth row}$$

The resulting matrix is seen to be of size M x N-1 and its elements are same as those of S.

Now the gradient term is given by

$$\nabla_{\underline{x}} \varepsilon = 2\, \text{Imag}\{\Phi W R\}^T\, [\text{Imag}\{\Phi W R\}\, \underline{x} - \alpha \underline{b}] \qquad (23)$$

The vector $\underline{b}$ can also be written as a product of Φ and a selection matrix to obtain

$$\underline{b} = \text{Imag}\, [\Phi \gamma]$$

where $\gamma^T = [1\ 0\ 0\ \ldots .0]$, and "Imag" denotes the imaginary
M X 1
part. Thus the terms within the square brackets of (23) can be written as

$$\text{Imag}\, [\Phi\{W\underline{y}\}]$$

where $y^T = [\alpha\ x(1)\ x(2) \ldots x(N-1)\ o\ o\ \ldots\ o]$
(1 x M)

Next the iterative algorithm can be stated as follows.

1) Start with the initial guess $\underline{x}_o = [x_o(1)\ x_o(2)\ \ x_o(N-1)]$
2) Form the vector $\underline{y}$ by appending $\underline{x}$ with zeros until the vector $\underline{y}$ of size M x 1 and set $x(o) = \alpha$, where α is the chosen scale factor
3) Compute the DFT of the appended sequence in step (2)
4) Multiply the kth element of the DFT by $e^{j\theta_x(\overline{k-1}w_o)}$ for k = o through M - 1.
5) Extract the imaginary part.
6) Multiply each element obtained in step (5) by $e^{j\theta_x(\overline{k-1}w_o)}$ as in step 4 for k = o through M - 1.

7) Compute the inverse DFT.

8) The resulting column matrix is proportional to the negative of the gradient of the error with respect to $\underline{x}$. Update $\underline{x}_k$ using

$$\underline{x}_{k+1} = \underline{x}_k + \mu R^T \text{ [gradient vector in step 8]}$$

9) Repeat steps 2 through 8 to complete the recursion. The optimum value of μ that maximizes the convergence rate is given as

$$\mu_k = (g_i^T \gamma_i)/(g_i^T A g_i)$$

where $\gamma_i = \alpha b - s\underline{x}_i$

and g_i is the gradient vector.

It is known that the convergence of the steepest descent method depends on the eigenvalue distribution of the matrix S^TS. If the ratio $\lambda max/\lambda min$ is large, the rate of convergence is usually slow. In the examples we considered it was found that the steepest decent method converges fast in the beginning, but tends to slow down after some time. The steepest decent method did not converge much faster even when techniques were used to accelerate convergence.

B. Non-expansive Convergence Algorithm

In this method, the current estimate of $\underline{x}$ is a nonlinear function of the previous estimate. Thus the update becomes

$$\underline{x}_{k+1} = F(\underline{x}_k)$$

where F is a nonlinear transformation. The algorithm we describe now, falls into a class of mappings known as non-expansive mappings. These mappings have been proven to yield a unique solution and an iterative algorithm with such a mapping always converges to the correct solution [7]. The algorithm is described next.

We let $x(k)$ be the kth DFT component of the M-point DFT of $x(n)$ and write $x(k)$ as

$$x(k) = |x(k)| \, e^{j\theta_x(k)}$$

Step 1: Begin with an initial guess of the unknown DFT magnitude $|x_o(k)|$. Form $x_1(k)$, an estimate of $x(k)$ as

$$x_1(k) = |x_o(k)| \, e^{j\theta_x(k)}$$

Compute the inverse DFT of $x_1(k)$ to obtain the first estimate $x_1(n)$ of $x(n)$. Since an M-point IDFT was used $x_1(n)$ is in general non-zero for $N \leq n \leq M-1$.

Step 2: Use $x_1(n)$ to form another sequence $y_1(n)$ as

$$y_1(n) = \begin{cases} x_1(n) & o \leq n \leq N-1 \\ o & N \leq n \leq M-1 \end{cases}$$

Step 3: Take the M-point DFT of $y_1(n)$. The magnitude $|y_1(k)|$ of $y_1(n)$ is then considered as a new estimate of $|x(k)|$. Then form a new estimate of $x(k)$ as

$$x_2(k) = |y_1(k)| \, e^{j\theta_x(k)}$$

From this a new estimate $x_2(n)$ is formed by computing the IDFT of $x_2(k)$. Repeating steps 2 and 3 sets up the recursion.

In this iterative procedure, the total squared error between $x(n)$ and its estimate $x_k(n)$ is a non-increasing quantity with each iteration [1]. To prove this, let $x_p(n)$ be the estimate at pth iteration and let us define the error E_p as

$$E_p = \sum_{n=o}^{M-1} \{x(n) - x_p(n)\}^2$$

From Parseval's theorem [11]

$$E_p = \frac{1}{M} \sum_{k=o}^{M-1} |x(k) - x_p(k)|^2$$

Since $x(k)$ and $x_p(k)$ have the same phase, E_p becomes

$$E_p = \frac{1}{M} \sum_{k=o}^{M-1} [|x(k)| - |x_p(k)|]^2$$

$$= \frac{1}{M} \sum_{k=o}^{M-1} [|x(k)| - |y_{p-1}(k)|]^2$$

using the triangular inequatity [12]

$$E_p \leq \frac{1}{M} \sum_{k=o}^{M-1} |x(k) - Y_{p-1}(k)|^2$$

Again using Parseval's theorem yields

$$E_p \leq \sum_{k=o}^{M-1} \{x(n) - y_{p-1}(n)\}^2$$

Since $y_{p-1}(n) = \begin{cases} x_{p-1}(n) & o \leq n \leq N-1 \\ o & N \leq n \leq M-1 \end{cases}$

We have,

$$\sum_{n=0}^{M-1} \{x(n) - y_{p-1}(n)\}^2 \leq \sum_{n=0}^{M-1} \{x(n) - x_{p-1}(n)\} = E_{p-1}$$

Thus

$$E_p \leq E_{p-1}$$

with equality holding iff $y_{p-1}(n) = x_{p-1}(n)$. In general, the above condition is not enough to ensure convergence. However, recently it has been shown by Lim et al. [7] that such an iterative scheme always converges. This is achieved by showing that the iterative algorithm falls into a class of non-expansive mappings[2]. Further, using certain properties of such mappings the convergence of the algorithm has been proven. In all the examples that were considered in the current study this algorithm was always found to converge to the correct solution. It was also observed that fewer iterations were needed when larger DFT sizes were used, as illustrated in Fig. 5. However, this does not result in shorter computation time, since larger DFT's need more computations.

C. Adaptive Relaxation [7],[3],[8]

The two methods described above do not converge fast all the time, when the length of the sequence is large. Hence, it is of interest to find ways to accelerate the

---

[2]See Appendix B for a more detailed discussion on this topic.

convergence rate. One way of doing this is to express the new estimate of $\underline{x}$ as a linear combination of the previous estimate and the current estimate; i.e.,

$$\underline{x}_{k+1} = \lambda \underline{x}_k + (1 - \lambda)\, F(\underline{x}_k)$$

where F is non-linear operation on $\underline{x}_k$ to get the current estimate, and λ is the convergence parameter which is chosen adaptively to achieve a faster convergence at each iteration. Now $\underline{x}_k$ is partitioned as

$$\underline{x}_k = \begin{bmatrix} \underline{v}_k^{(1)} \\ \underline{v}_k^{(2)} \end{bmatrix} = \underline{v}_k \qquad (24)$$

where $\underline{v}_k^{(1)}$ consists only of the elements of x(n) for $o \leq n \leq N-1$ and $\underline{v}_k^{(2)}$ contains all x(n) for $n \geq N$.
Next, rewriting the above recursion equation we have

$$\underline{v}_{k+1} = (1 - \lambda_k)\, \underline{v}_k + \lambda_k\, F\, (\underline{v}_k)$$

Next, defining

$$\gamma_k = F(\underline{v}_k) - \underline{v}_k$$

we obtain

$$\underline{v}_{k+1} = \underline{v}_k + \lambda_k\, \underline{\gamma}_k \qquad (25)$$

This is equivalent to the original iterative scheme, when $\lambda_k = 1$.

The objective is to choose λ_k, so that the convergence rate at each iteration is maximized. Rewriting (25) using (24) we get

$$\begin{bmatrix} v_{k+1}^{(1)} \\ \hline v_{k+1}^{(2)} \end{bmatrix} = \begin{bmatrix} v_k^{(1)} \\ \hline v_k^{(2)} \end{bmatrix} + \lambda_k \begin{bmatrix} \gamma_k^{(1)} \\ \hline \gamma_k^{(2)} \end{bmatrix}$$

The phase of $\underline{v}_k$ and $F(\underline{v}_k)$ are same and it is equal to the phase of $\underline{x}$. Therefore, for any λ_k. $\underline{v}_{k+1}$ has the same phase as $\underline{x}$. If $\underline{v}_{k+1}^{(2)}$ goes to zero, then $v_{k+1}^{(1)}$ will be equal to $\underline{x}$. Thus, a reasonable approach to maximize the rate of convergence is to choose λ_k in such a way that the norm of the vector $\underline{v}_{k+1}^{(2)}$ is minimized. That is

$$\left[\frac{d}{d\lambda} \| v_{k+1}^{(2)} \|\right]^2 \Bigg|_{\lambda = \lambda_k^*} = 0$$

which gives the optimum value of λ_k, which is denoted by λ_k^*. Solving for λ_k^* we get

$$\lambda_k^* = - \frac{\langle \underline{v}_k^{(2)}, \underline{\gamma}_k^{(2)} \rangle}{\| \gamma_k^{(2)} \|^2}$$

where <-,-> denotes the inner product between the two vectors.

In the examples considered, this method has the fastest convergence rate. It required significantly smaller number of iterations to achieve a certain mean square error (m.s.e) compared to the first two methods. Figures 2 and 6 clearly demonstrate this fact. The factor λ_k is evaluated using the two vectors $\underline{\gamma}_p^{(2)}$ and $\underline{v}_p^{(2)}$. The elements of these vectors tend to become smaller as we approach the correct solution. Due to machine round-off errors, the value of λ_k obtained is unreliable. Thus, below a certain m.s.e. the solution does

not improve much due to unreliable estimates of λ. However, the m.s.e. at this point is usually so small, that the reconstructed signal closely matches the original signal.

V. EXTENSION TO TWO DIMENSIONS [1],[3]

One usual way to accomplish the extension from one to two dimensions is to stack all rows or columns to form a large column vector, and then apply the one dimensional theorems to it; i.e.,

$$\hat{x}_1 [n_1 N + n_2] = x(n_1, n_2)$$

where $\hat{x}(k)$ is the constructed one-dimensional signal and $x(n_1, n_2)$ is defined for $o \leq n \leq N_1-1$ and $o \leq n \leq N_2-1$ and $N \geq N_1$.
For this case,

$$\hat{x}_1(w) = x(w_1, w_2) \Big|_{w_1 = Nw,\ w_2 = w}$$

But recently, Hayes [3] extended Theorems 1 through 6 to the multi-dimensional case using theorems of algebra of several variables. The constraints on the Z-transforms of the signals are same as that for one-dimensional signals. Before we go on to state the theorems, certain definitions from algebra of several variables are worthwhile considering.

Definitons

An m-dimensional sequence is a function of m integer variables, n_1 through n_m, which is denoted by $x(n_1, n_2 \ldots n_m)$. The Z-transform of the m-dimensional sequence is given by

$$x(z_1, z_2 \ldots z_m) = \sum_{n_1} \cdots \sum_{n_m} x(n_1, n_2 \ldots n_m) z_1^{-n_1}, z_2^{-n_2} \ldots z_m^{-n_m}$$

For ease of notation we will represent $x(n_1,\ldots n_m)$ as $x(\underline{n})$ and $x(z_1, z_2 \ldots z_m)$ by $x(\underline{z})$. With this notation the above equation becomes

$$x(\underline{z}) = \sum_{\underline{n}} x(\underline{n})\ \underline{z}^{-\underline{n}}$$

Throughout the rest of our discussion, all sequences are assumed to have rational Z-transforms with a region of convergence which includes the unit poly disk. That is, the n-dimensional sequence is assumed to have a convergent Fourier transform, which is defined as

$$x(\underline{w}) = \sum_{\underline{n}} x(\underline{n})\ e^{-j\underline{nw}}$$

where $x(\underline{w}) = x(w_1, w_2, \ldots w_n)$.

All the sequences are assumed to have "finite support," i.e., $x(\underline{n})$ is non-zero only for a finite number of values of $\underline{n}$. It is also assumed, without loss of generality, that a sequence with finite support is equal to zero for all $\underline{n} < \underline{o}$. (If $\underline{m}$ and $\underline{n}$ are two vectors, then $\underline{m} < \underline{n}$ means that $m_k < n_k$ for each k).

The Z-transform of a sequence $x(\underline{n})$ is defined to be symmetric if, for some vector $\underline{k}$ of positive integers

$$x(\underline{z}) = \pm\ \underline{z}^{-\underline{k}}\ \ x(\underline{z}^{-1})$$

It is seen from this definition that a one-dimensional signal with all of the zeros on unit circle or in reciprocal pairs is symmetric. With these definitions we state the following theorems.

Theorem 1:

Let $x(\underline{n})$ and $y(\underline{n})$ be two multi-dimensional sequences with finite support and let $\underline{M} \geq 2\underline{N} - 1$. If $x(\underline{z})$ and $y(\underline{z})$ have no nontrivial symmetric factors and if $\Phi_x(\underline{k}) = \Phi_y(\underline{k})$ for all $\underline{k}$ $\underline{o} \leq \underline{k} \leq \underline{M} - \underline{1}$, then $x(\underline{n}) = \beta y(\underline{n})$ for some positive constant β. If $\tan \Phi_x(\underline{k}) = \tan \Phi_y(\underline{k})$ for all $\underline{k}$, $\underline{o} \leq \underline{k} \leq \underline{M}-1$, then $x(\underline{n}) = \beta y(\underline{n})$ for a real constant β.

Theorem 2:

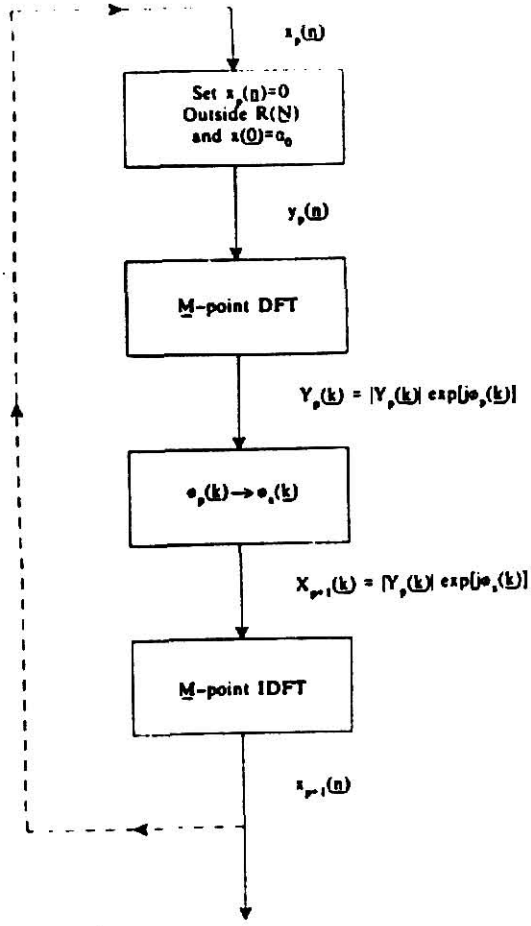
Let $x(\underline{n})$ and $y(\underline{n})$ be two multi-dimensional sequences with finite support and let $\underline{M} \geq 2\underline{N} - 1$. If $x(\underline{z})$ has no symmetric factors and $\Phi_x(\underline{k}) = \Phi_y(\underline{k})$ for all $\underline{k}$, $\underline{o} \leq \underline{k} \leq \underline{M} - \underline{1}$, then $x(\underline{n}) = \beta y(\underline{n})$ for a positive constant β. If $\tan \Phi_x(\underline{k}) = \tan \Phi_y(\underline{k})$ for all $\underline{k}$, $\underline{o} \leq \underline{k} \leq 2\underline{M}-1$, then $x(\underline{n}) = \beta y(\underline{n})$ for a real constant β.

Note that in contrast to Theorem 1, the second theorem does not require any constraints on $y(\underline{n})$, in that it can be any multi-dimensional sequence with support $R(\underline{N})$. The exclusion of trivial symmetric factors excludes any factor of the form $\underline{z}^{\underline{n}_o}$. This implies that either the location of first nonzero term is known, or $\underline{n}_o$ is equal to zero. That is, $x(\underline{z})$ is reducible to the form.

$$x(\underline{z}) = \underline{z}^{-\underline{n}_o} \prod_k x_k(\underline{z})$$

where $\underline{n}_o = \underline{o}$ or $\underline{n}_o$ is known.

For reasons stated earlier, the iterative algorithms are the only way to solve multi-dimensional reconstruction problems. In fact all three algorithms can be easily extended to multi-dimensions. The transforms used will be m-dimensional DFTs instead of one-dimensional DFTs. We illustrate this aspect via the following flow chart related to the non-expansive iterative algorithm.



Block diagram of the phase-only iteration.

This algorithm can be modified to implement the adaptive relaxation algorithm. This is desirable, since this would result in faster convergence and the reconstructed sequences can be obtained in fewer iterations.

VI. SOME EXPERIMENTAL RESULTS

In this section several data sets are used to first demonstrate that reconstruction does occur and that the related m.s.e. decreases as the number of iterations increases. Secondly, we wish to compare the convergence rates of different iterative schemes and evaluate the extra computation required to achieve a faster convergence rate. Next we compare the convergence rates with respect to different DFT sizes used for the iterative methods. For the gradient method we wish to study the effect of larger DFT sizes and longer sequence lengths on the convergence rate. One would expect the matrix S^TS in (17) to become ill conditioned as its size becomes larger. Finally, reconstruction is demonstrated for some image data, using two-dimensional DFTs.

Figure 1 shows that all three methods achieve perfect reconstruction after sufficient number of iterations. For the data used it was observed that the gradient method needed 140 iterations, the nonexpansive iteration needed 300 iterations and the adaptive relaxation needed only 60 iterations to give a m.s.e. of 3. Figure 2 shows the comparison of convergence rates of different methods. It is clear that the adaptive relaxation method converges the fastest and the nonexpansive method the slowest. The gradient method is as fast as the adaptive relaxation in the beginning, but tends to slow down after about 50 iterations. Figure 3 shows the reconstruction of a sequence of length 8 and using a 16-point DFT, at different

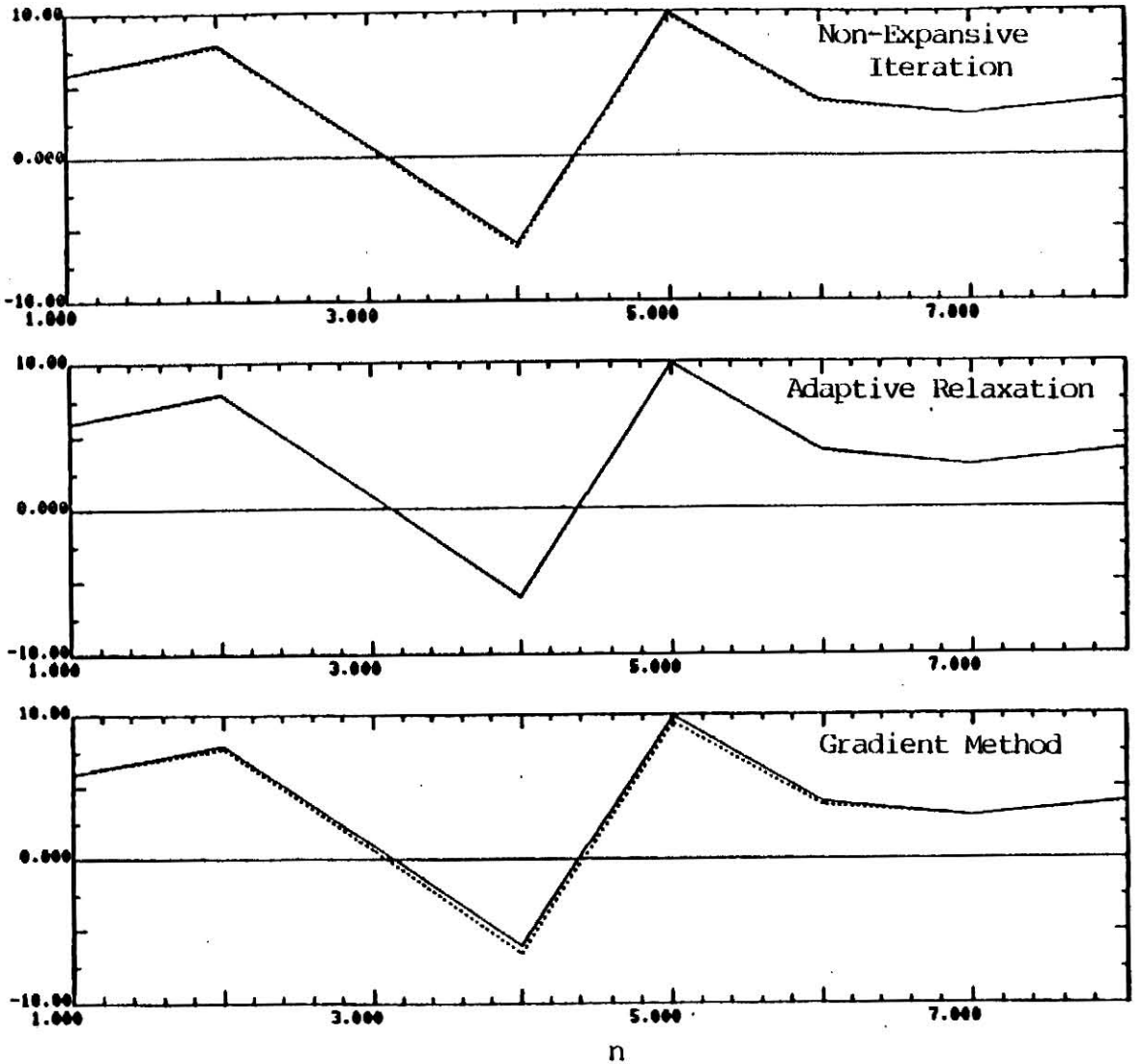


Figure 1: Comparison of the original and the Reconstructed signals
—— original
... Reconstructed

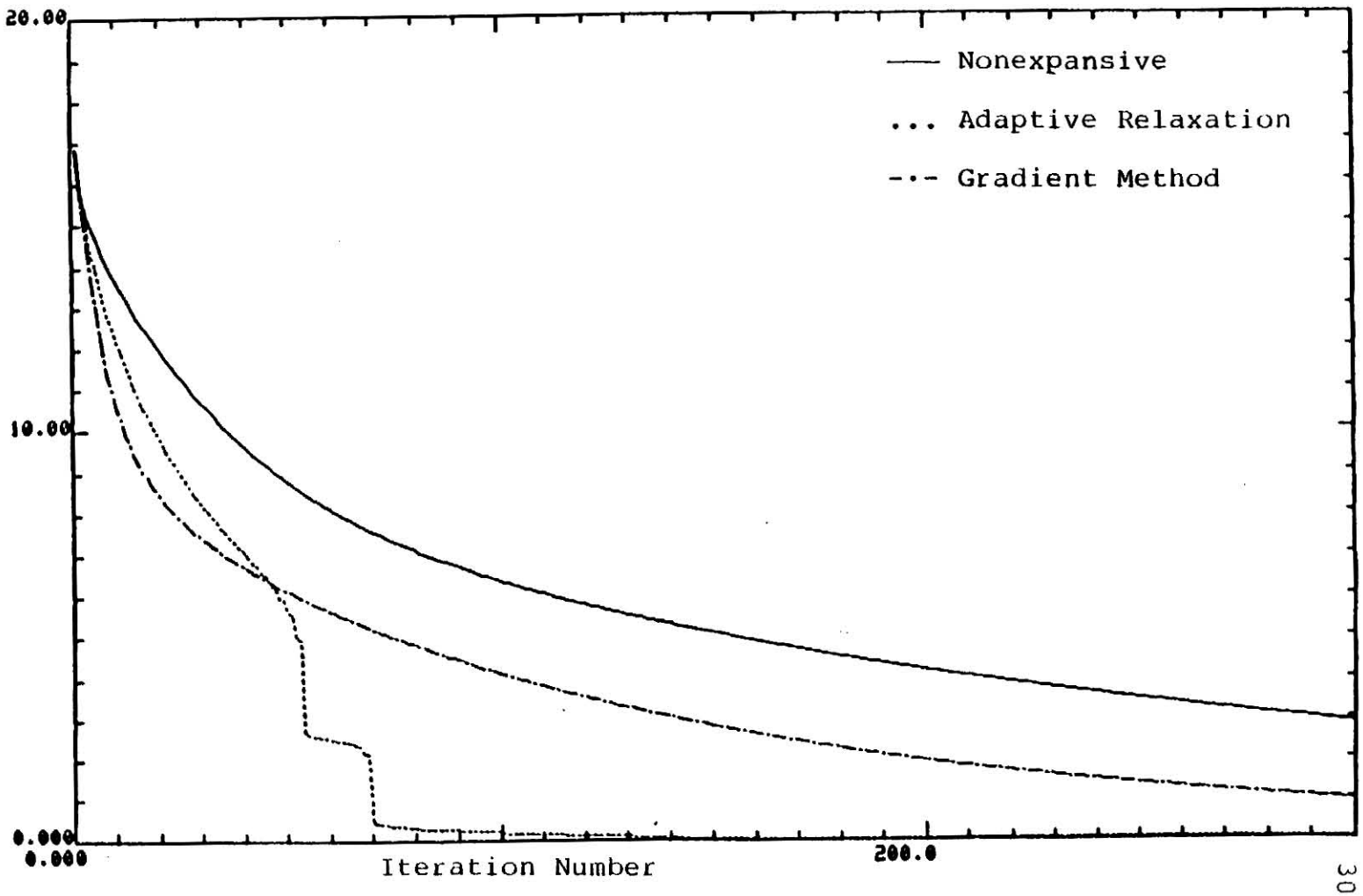


Figure 2: Comparison of Convergence Rates of different Methods. Length of the sequence = 8, FFT = 16.

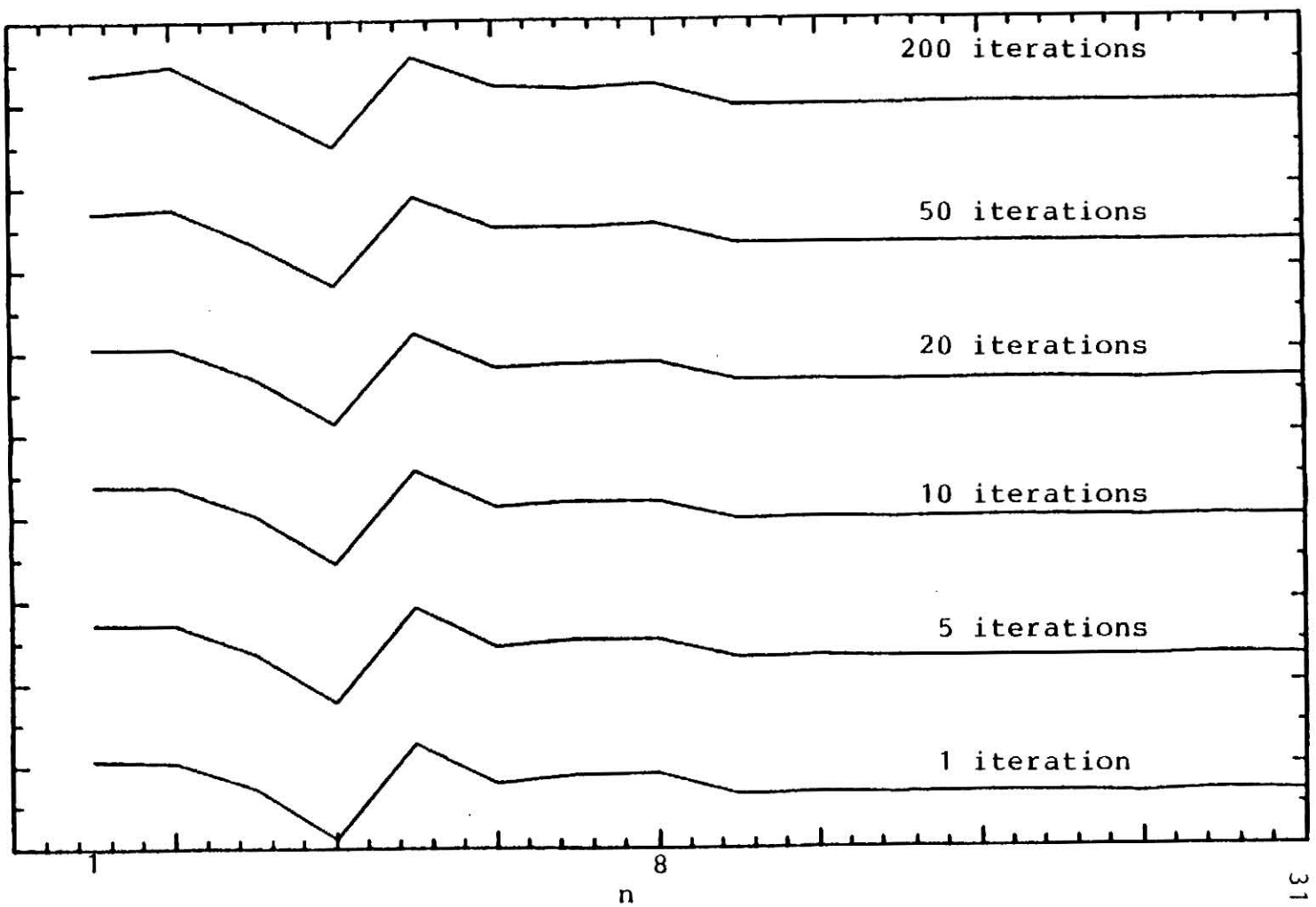


Figure 3: Reconstructed Signal at Different Iterations
Length of sequence = 8, FFT size = 16

iterations, while Figure 4 shows the reconstruction of a sequence of length 32 using a 128-point DFT. These two figures show the reconstructed signal at various iterations.

Figure 5 shows the convergence rates of the same method using two different size FFTs. One would naturally expect the larger size FFT iteration to converge faster, since more information is known about the signal, in terms of its phase at a larger number of points. Experimental results concur with this conclusion. However, larger FFT involves more computation and hence, the computation time required for reconstruction was more or less the same, when different size FFTs were used for reconstruction. Figure 6 shows that for all DFT sizes, the adaptive relaxation method converges the fastest. It involves only 3(M-N) more multiplications and additions than the nonexpansive iterative method, where M is the FFT size and N is the signal length. Since it requires significantly smaller number of iterations than the other two methods, the savings in terms of computer time is significant. Therefore, one can save more computer time by using adaptive relaxation, rather than using a larger FFT for reconstruction.

Figure 7 compares the convergence rates of gradient method and the nonexpansive method at different DFT sizes. It is seen that even for larger DFTs the gradient method converges faster than the nonexpansive method for a short sequence. Hence, one may conclude that the condition number ($\lambda max/\lambda min$) of the matrix S^TS is not affected even when

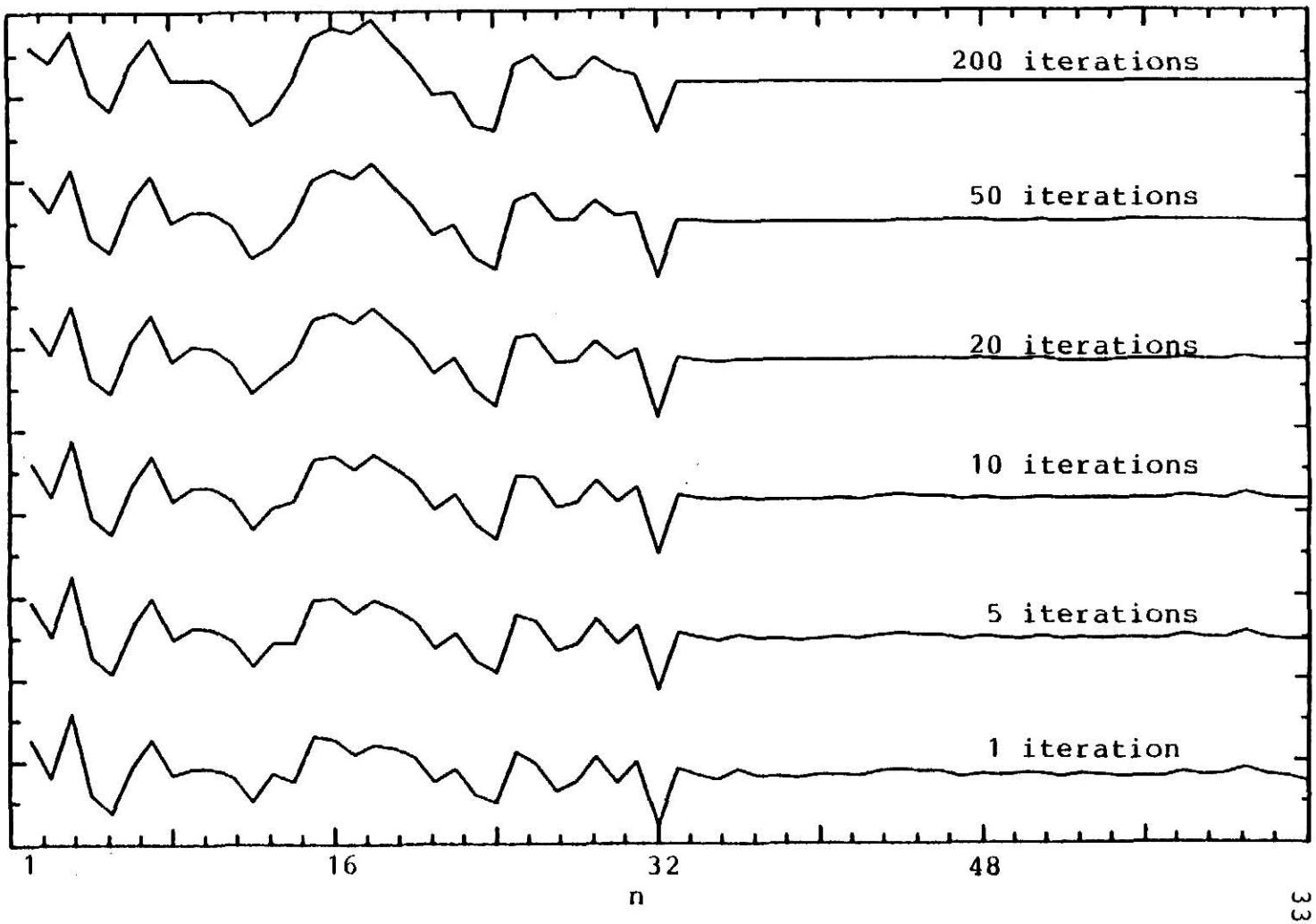


Figure 4: Reconstructed signal at different iterations
Length of sequence = 32, FFT size = 64

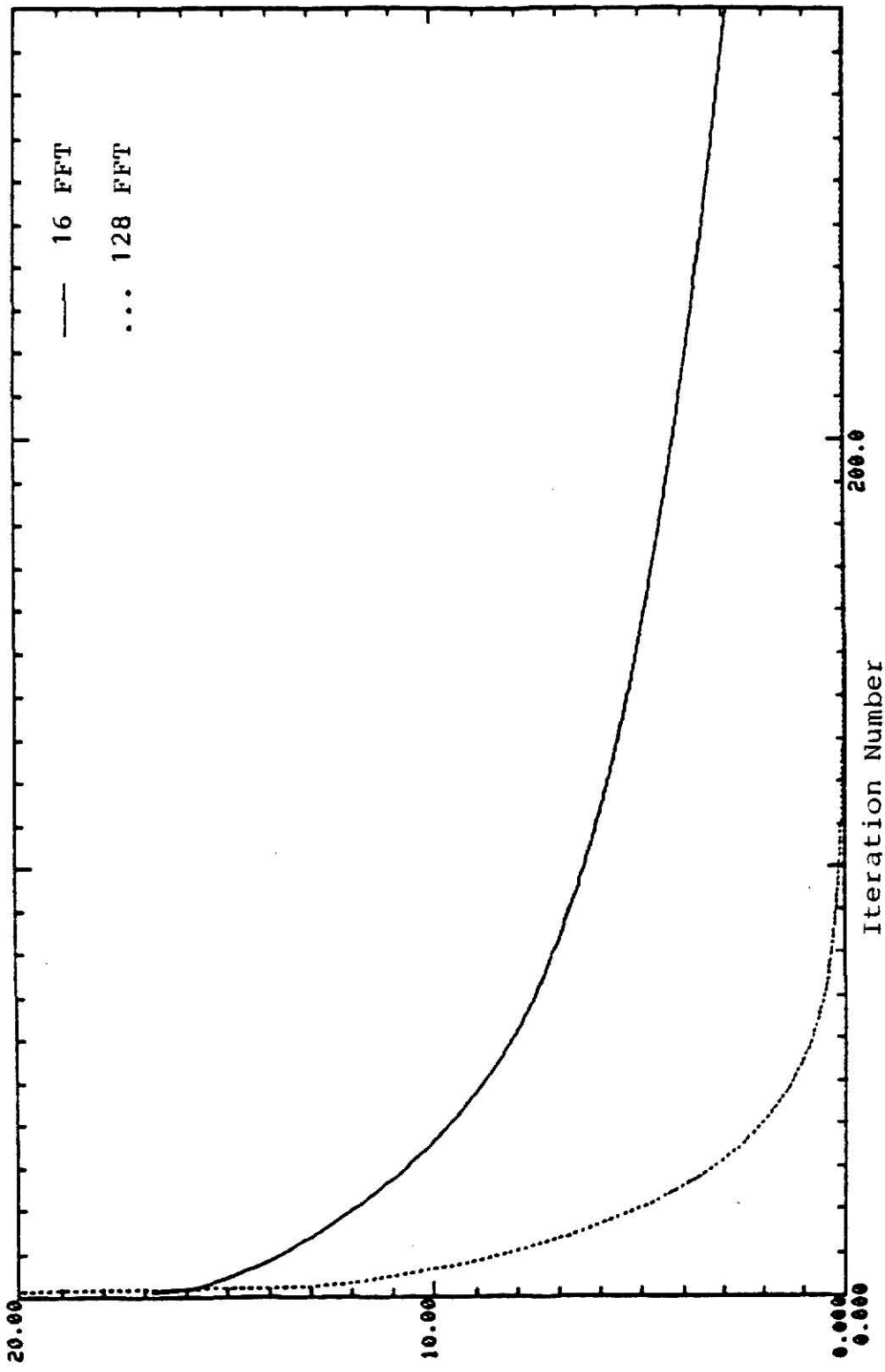


Figure 5: Comparison of Convergence rates of different FFT sizes. Length of the sequence = 8.

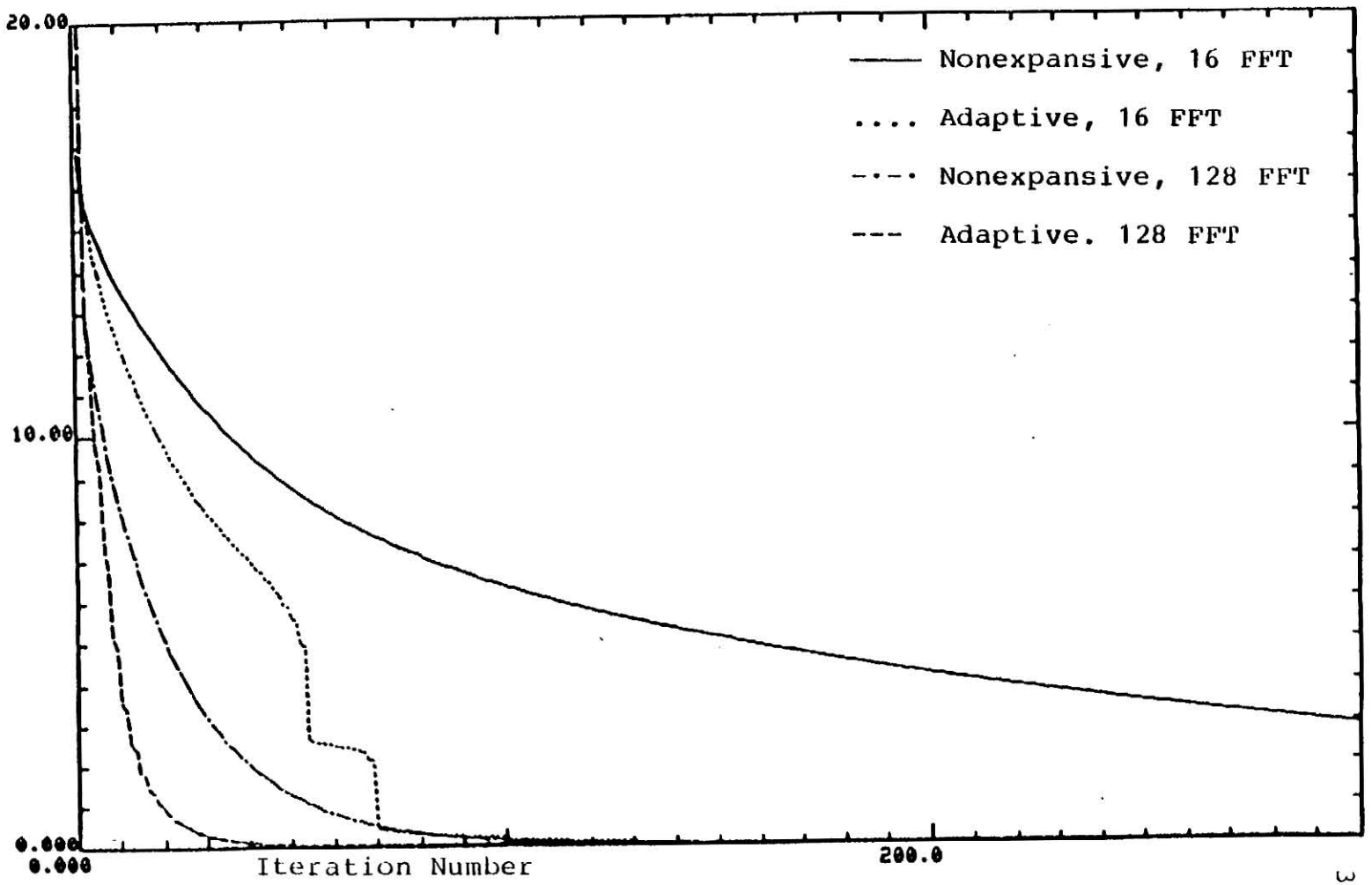


Figure 6: Comparison of Adaptive relaxation and Nonexpansive iterative method
Length of the sequence = 8

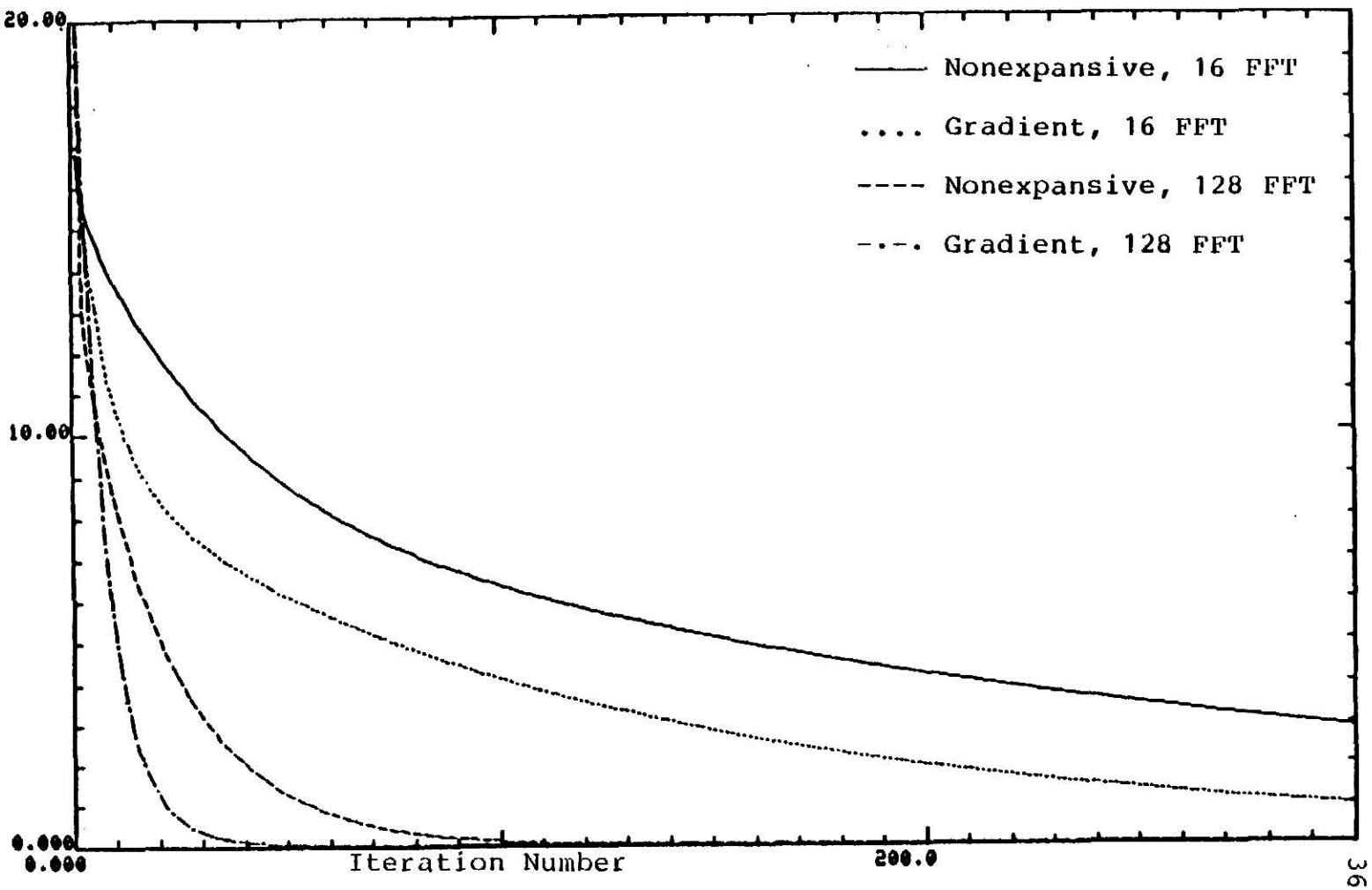
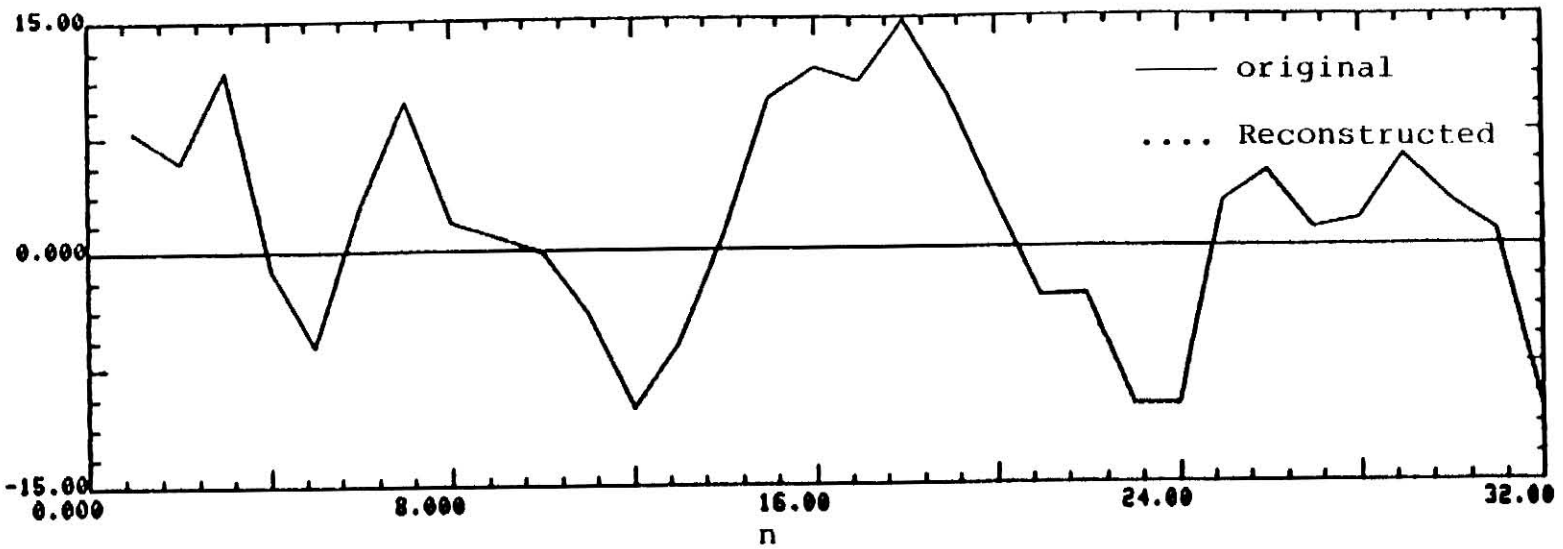


Figure 7: Comparison of the Gradient and Nonexpansive Methods
Lengths of the sequence = 8

larger DTSs are used; i.e., it depends only on the length of the sequence. Figure 8 shows the reconstruction of a larger image. Figure 9 shows that for a larger sequence the gradient method performs poorer compared to the nonexpansive method. Figure 10 shows that the adaptive relaxation converges the fastest even for long sequences. It is seen that about 50 iterations are sufficient to achieve a good reconstruction using this method. Of the three methods studied so far, the adaptive relaxation method was found to be the most effective. With very little additional computation, it schieves a much faster convergence.

A 8 x 8 image was also used for reconstruction (see Figure 11a). A 64 x 64 DFT was used at each iteration and the reconstructed signal is shown after 50 iterations (Figure 11b). The reconstructed signal compares closely with the original signal, even though each point is off by a small fraction. Figure 11c shows the image, when rounding to the nearest integer was used. This compares exactly with the orginal signal. The inference is that for standard digital images, it does not require too many iterations to achieve a S/N ratio in the same order as the related quantization noise. Therefore, it appears that about 50 iterations would be adequate to get a fairly good picture as a reconstruction image.

Figure 12a through 12 f show the original and the reconstructed image at various iterations. It is a 64 x 64 image and a 128 x 128 DFT was used. It is seen that the reconstructed image is of comparable quality to the original image. Figures



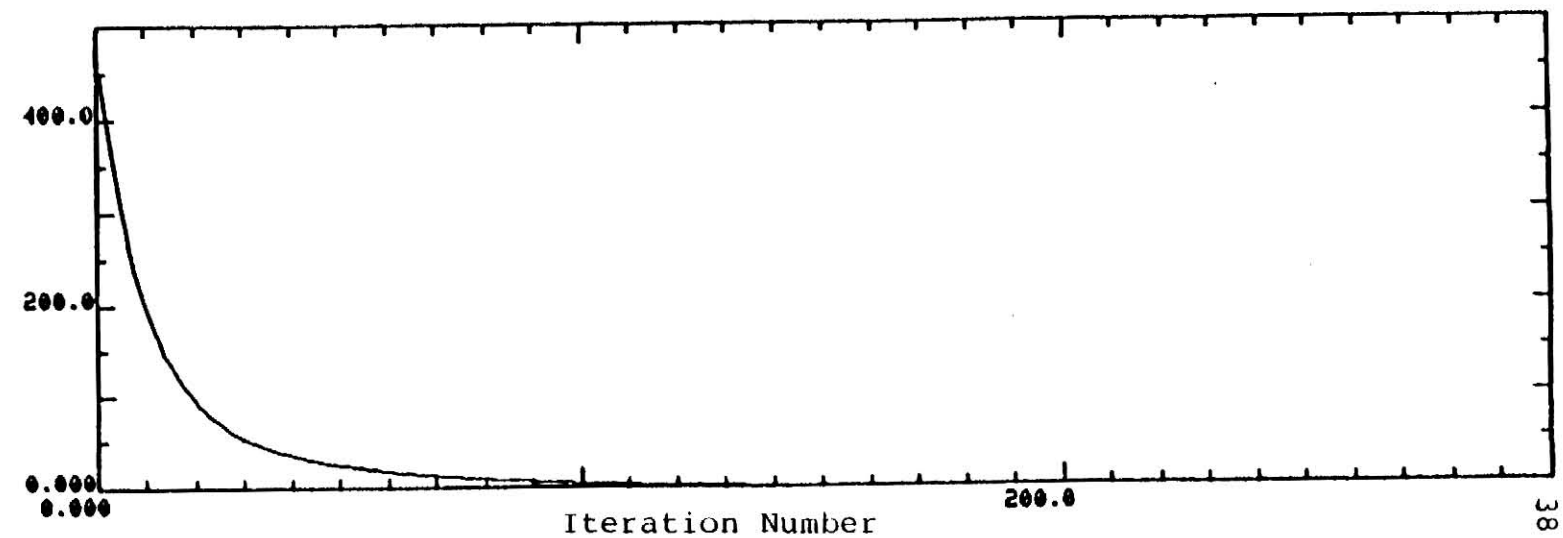


Figure 8: Reconstruction of a long sequence
Length of the sequence = 32, FFT size = 64

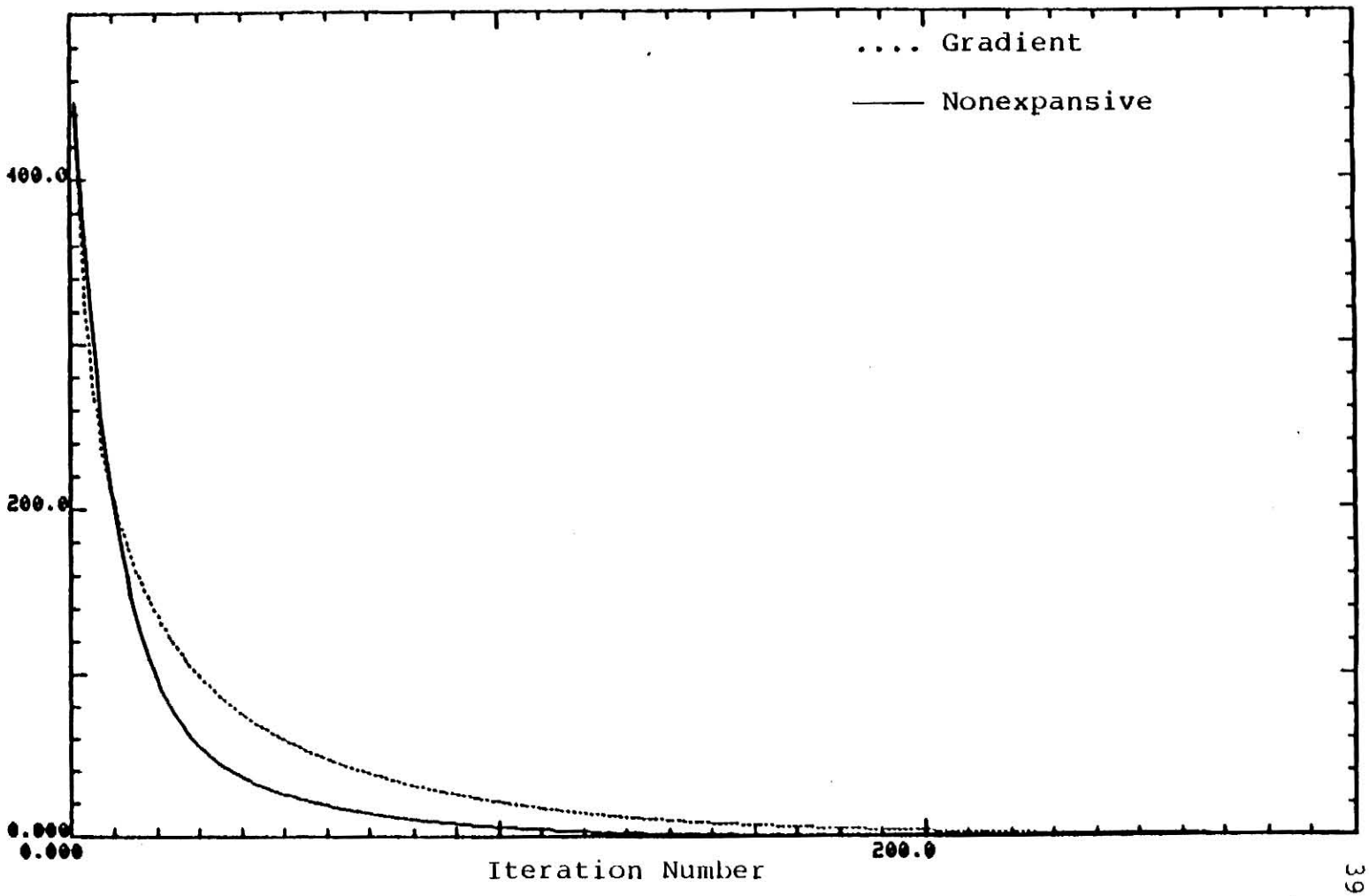


Figure 9: Comparison of Gradient and Nonexpansive Methods for a long sequence
Length of the sequence = 32, FFT size = 128

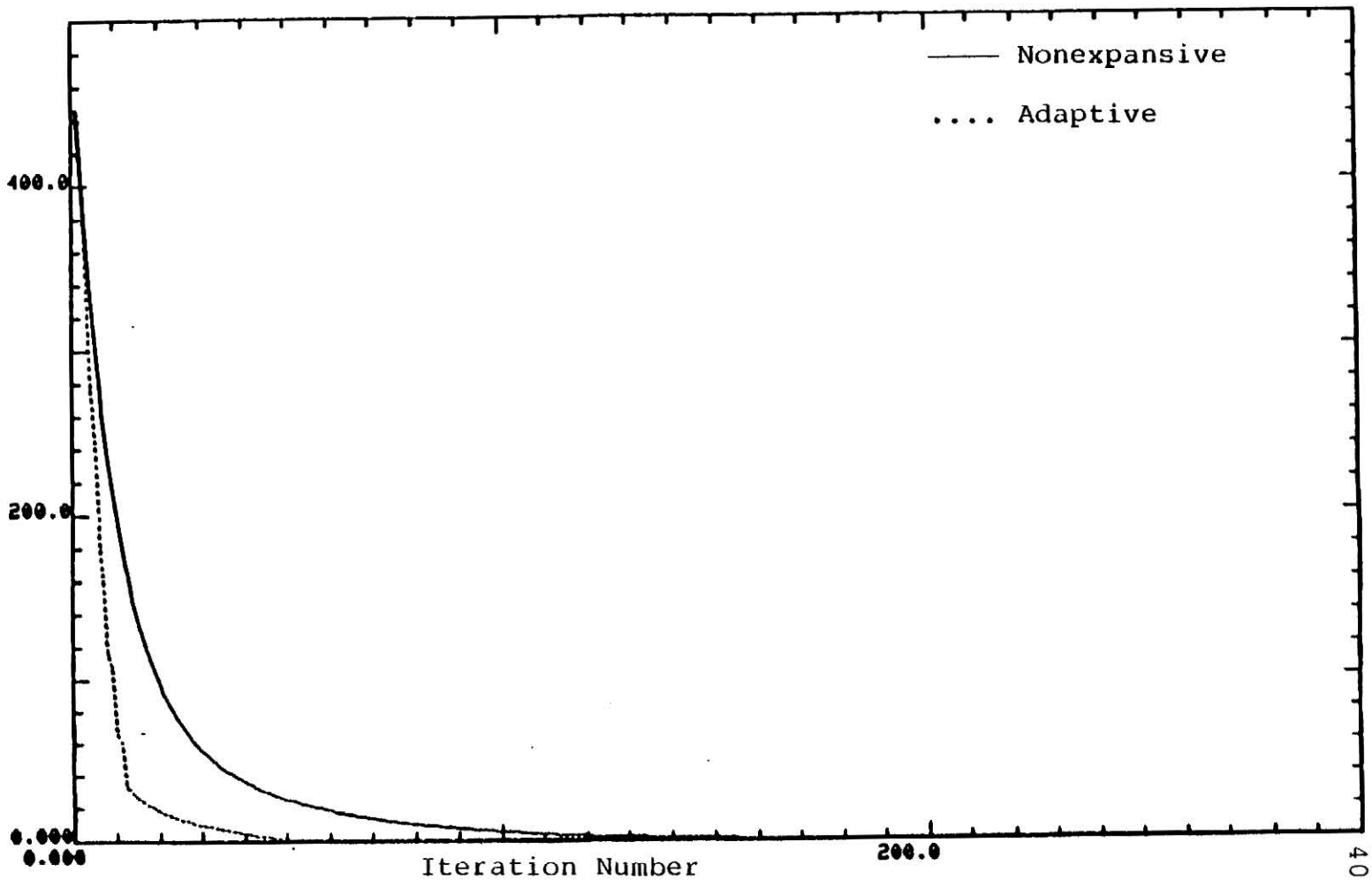


Figure 10: Comparison of Nonexpansive and Adaptive Methods for a long sequence
Sequence length = 32, FFT size = 128

1	3	4	5	6	4	2	1
4	2	1	1	4	7	3	4
7	6	9	4	7	6	5	7
8	11	13	15	7	8	9	6
4	3	12	6	6	6	6	6
6	6	6	8	9	7	5	4
9	11	7	9	8	4	3	5
11	8	8	5	4	6	7	3

(a)

1.000	3.079	4.069	5.098	6.114	4.072	2.054	1.001
4.084	2.040	1.007	1.027	4.051	7.155	3.033	4.091
7.148	6.072	9.201	4.029	7.182	6.091	5.069	7.161
8.150	11.24	13.15	15.38	7.080	8.129	9.202	6.092
4.068	3.010	12.30	6.009	6.139	6.120	6.082	6.113
6.095	6.126	6.084	8.150	9.156	7.138	5.100	4.054
9.132	11.26	7.092	9.186	8.159	4.057	3.034	5.125
11.27	8.074	8.195	5.095	4.054	6.109	7.167	3.033

(b)

0	3	4	5	6	4	2	1
4	2	1	1	4	7	3	4
7	6	9	4	7	6	5	7
8	11	13	15	7	8	9	6
4	3	12	6	6	6	6	6
6	6	6	8	9	7	5	4
9	11	7	9	8	4	3	5
11	8	8	5	4	6	7	3

(c)

Figure 11: Image Reconstruction, 8 x 8 image.
a) Original
b) Reconstructed image after 50 iterations
c) Reconstructed image when rounding off is done

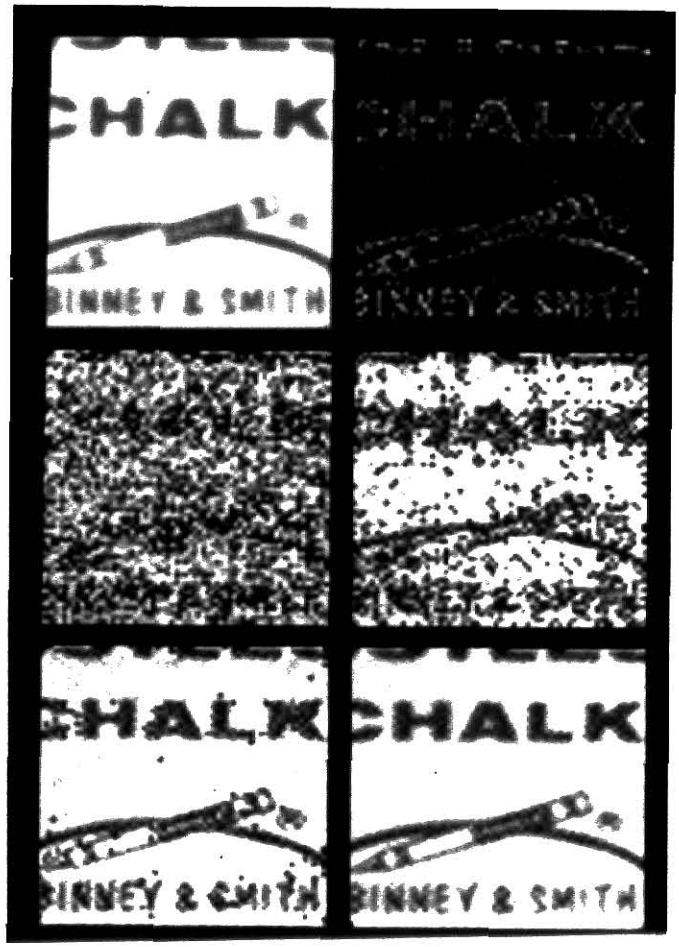


Fig. 12. Phase only reconstruction of a 64 x 64 image.

a) original b) after 1 iteration

c) 5 iterations d) 10 iterations

e) 20 iterations f) 50 iterations

13a through 13f show the reconstruction for a larger image. A 128 x 128 image was the test data and a 256 x 256 DFT was used at each iteration. Here again near perfect reconstruction is achieved.

Finally to demonstrate the ability to remove blur, the image is severely lowpass filtered and the blurred picture is shown in Figure 14b. The original is shown in 14a. The reconstructed images at 20 and 40 iterations are shown in Figures 14c and 14d, respectively. It is apparent that the reconstructed images are much sharper and are of comparable quality to the original image.

In the reconstruction alogrithms, constraints such as positivity can be imposed. Such constraints can be shown to be a class of nonexpansive mappings and the algorithms can be shown to converge. In fact such constraints make the iterative schemes converge faster [7].

Fig. 13. Phase only reconstruction (128 x 128)

a) original b) After 5 iterations

c) 15 iterations d) 30 iterations

e) 40 iterations f) 50 iterations

Fig. 14. Reconstruction of an LPF image.

a) original b) LPF

c) After 20 iterations d) 40 iterations

VII. CONCLUSIONS

In this report we have demonstrated that a signal can be uniquely reconstructed from the phase of the Fourier transform. The conditions for uniqueness were developed for one-dimensional and multi-dimensional signals. Different algorithms to do the reconstruction were also developed. Reconstruction was demonstrated for some one-dimensional as well as two-dimensional signals. Convergence rates of different algorithms and the effect of signal length on the convergence rate were studied. The ability remove blur for zerophase distortions was demonstrated. Recently this phase only reconstruction algorithm has been used as a means of implementing Hilbert transformers for minimum phase signal [9]. The nonexpansive iterative algorithms have been used to implement two-dimensional filters iteratively [10].

Of the reconstruction algorithms that were considered in this study, the adaptive relaxation method was found to be the most effective. This assessment was made on the basis of the m.s.e. incurred in reconstruction, convergence rate, and computational considerations.

Certain problems need further investigation. One of them is the sensitivity of the reconstructed signal due to changes in phase information; i.e., how does the reconstructed signal change due to inaccurate specification of phase? Another problem of interest is to develop similar theorems for reconstructing the signal from the magnitude information

alone. Such problems are of interest in optics where only the magnitude information alone is available in the form of diffraction plane pictures.

Computer programs used to carry out the experimental results reported here are given in Appendix C.

REFERENCES

[1] M.H. Hayes, J. S. Lim, and A. V. Oppenheim, "Signal Reconstruction from Phase or Magnitude," IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP-28, pp. 672-680. Dec. 1980.

[2] H. C. Andrews and B. R. Hunt, Digital Image Restoration. Englewood Cliffs, NJ: Prentic Hall, 1977.

[3] M. H. Hayes, "The Reconstruction of a Multi-dimensional sequence from the Phase or Magnitude of it's Fourier Transform," IEEE Trans. Acoustic, Speech and Signal Processing, Vol. ASSP-30, pp. 140-154, Apr. 1982.

[4] R. W. Gerchberg and W. O. Saxton, "A Practical Algorithm for the Determination of Phase from Image Diffraction Plane Pictures", Optik, Vol. 35, pp. 237-246, 1972.

[5] J. R. Fineup, "Reconstruction of an Object from the Modulus of it's Fourier Transform", Opt. Letters, Vol. 3, pp. 27-29, 1978.

[6] T. Quatieri, "Phase Estimation with Applications to Speech Analysis and Synthesis", Sc. D. dissertation, Dept. of Elect. Eng., M.I.T., 1979.

[7] V. T. Tom, T. F. Quatieri, M. H. Hayes and J. H. McClellan, "Convergence of Iterative Nonexpansive Signal Reconstruction Algorithms", IEEE Trans. Acoustic, Speech and Signal Processing, Vol. ASSP-29, pp. 1052-1058, Oct 1981.

[8] V. T. Tom, "Constrained Iterative Signal Reconstruction Algorithms", Sc. D. dissertation, Dept. of Elec. Eng., M.I.T., Cambridge, 1981.

[9] T. F. Quatieri and A. V. Oppenheim, "Iterative Techniques for Minimum Phase Signal Reconstruction from Phase or Magnitude", IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP-29, pp. 1187-1193, Dec. 1981.

[10] T. F. Quatieri and D. E. Dudgeon,"Implementation of 2-D Digital Filters by Iterative Methods", IEEE Trans. Acoust. Speech and Signal Processing, Vol. ASSP-30, pp. 473-487, Jun. 1982.

[11] A. V. Oppenheim, and R. W. Shafer, "Digital Signal Processing, Englewood Cliffs, NJ: Prentice Hall, 1975.

[12] R. V. Churchil, J. W. Brown and R. F. Verhey, Complex Variables and Applications, McGraw-Hill Book Company, New York, 1960.

[13] J. S. Lim, J. C. Anderson and C. L. Searle,"Signal Reconstruction from Cosine Transform Magnitude", Proceedings of the IEEE, Vol. 70, pp. 1460-1462, Dec. 1982.

[14] J. R. Fineup, "Space Object Imaging through the Turbulent Atmosphere", Opt. Engg., pp. 529-534, Sept. 1979.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank Dr. Nasir Ahmed, Dr. Donald Hummels and Dr. Rodney Bates for being members of my graduate committee. In particular I wish to thank Dr. Ahmed for his encouragement and guidance. Thanks are also due to Marcus Junod for his assistance in taking the pictures. I also wish to thank the Department of Electrictrial Engineering for their financial support.

APPENDIX A

Phase Unwrapping

From the definition of the phase of the Fourier transform we have,

$$\theta_x(w) = \tan^{-1}\left[\frac{\text{Imag}[x(w)]}{\text{Real}[x(w)]}\right]$$

where $\theta_x(w)$ is the phase of $x(n)$ and $x(w)$ is the Fourier transform of $x(n)$. Since arctan function is a periodic function, there are infinitely many values that $\theta_x(w)$ can have at each value of w. One simple way to avoid this problem is to limit $\theta_x(w)$ between $-\pi$ and $-\pi$. A typical phase function of this type appears as shown in Figure A-1-a.

It is seen that the phase function thus computed has a finite number of discontinuities, where the values of $\theta_x(w)$ jump from $+\pi$ to $-\pi$ and vice versa. In some problems like homorphic filtering, one requires that $\theta_x(w)$ be a continuous function for all w. To circumvent this problem, we add an appropriate interger multiples of 2π to the value of $\theta_x(w)$ obtained as shown above. Figure A-1-b shows the correction to be added and Figure A-1-c shows the resulting phase that is continuous for all w. A practical way of accomplishing this would be to compute the phase with modulo 2π, detect the discontinuities, and add 2π to the phase if it is needed. To detect the discontinuities correctly, one should calculate the phase at sufficiently large number of points. This is

generally not a severe limitation because of the existence of FFT algorithms.

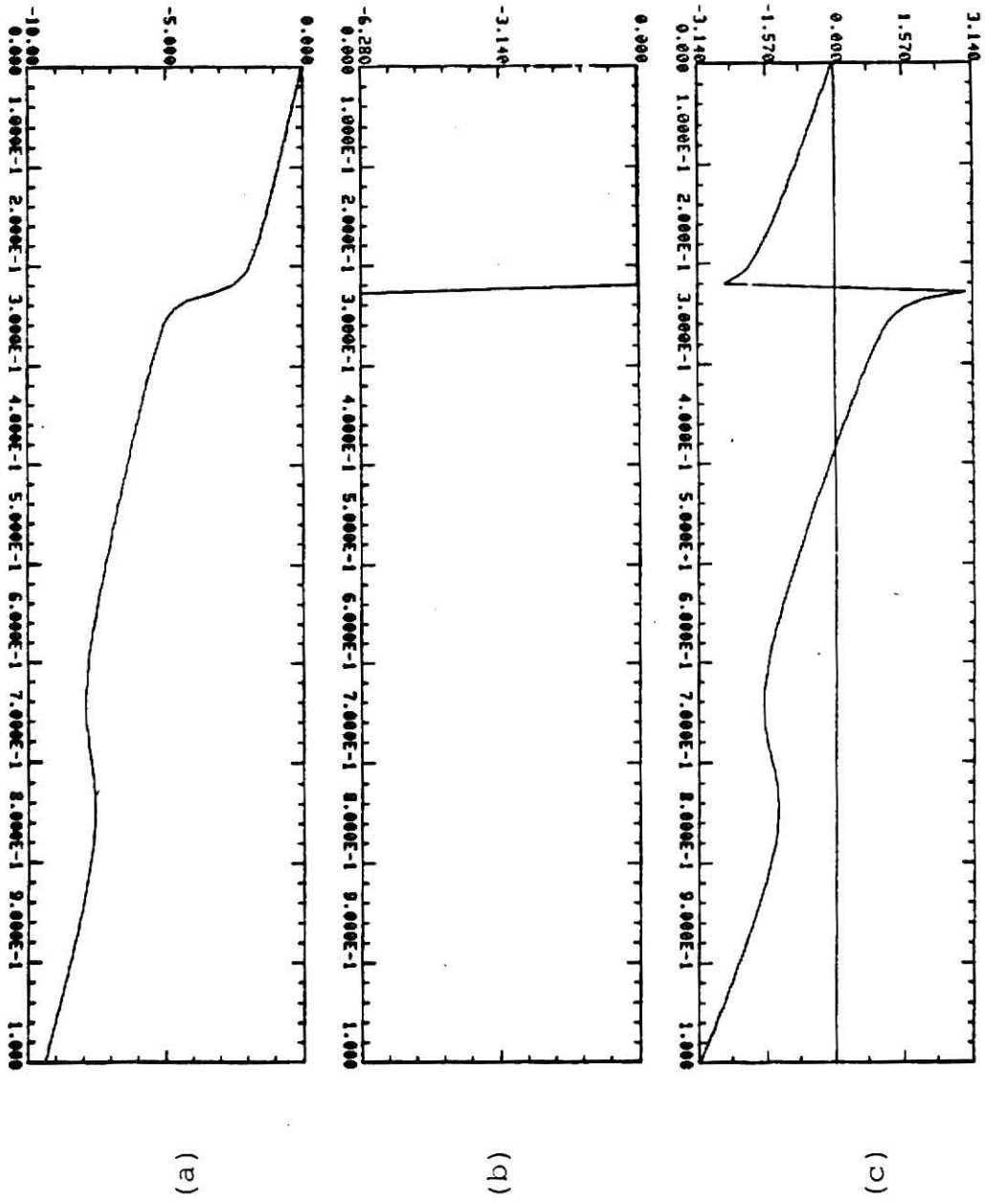


Figure A-1. Phase Unwrapping.

a) Modulo 2π phase b) Correction
c) Unwrapped phase

APPENDIX B

Non-Expansive Mappings [+]

A metric space consists of a set X along with metric or a distance function d defined on X. We simply refer to X as a metric space and assume that an underlying metric d has been defined on X. The distance function d is defined as follows.

$$d(x,y) = \left[\sum_{n=o}^{N-1} [x(n) - y(n)]^2\right]^{\frac{1}{2}} \qquad (A-1)$$

A point $x \in R^N$ is an N-dimensional vector but will also refer to an N-point sequence x(n). In this context, we assume that x(n) is defined for all n with x(n) equal to zero outside the interval [o, N-1].

A mapping F from a subset A of metric space X into X will be represented notationally as $F:A \subset X \to X$. First we define what is meant by a contraction mapping.

Let A be a subset of metrix space X and let F be a mapping which maps A into itself. Then F is said to be contraction mapping if there is a constant α, $o \leq \alpha < 1$, such that for all $x,y \in A$.

$$d(Fx, Fy) \leq \alpha\, d(x,y) \qquad (A-2)$$

A wider class of mappings results if we allow α equal to one in the definition of contraction mapping. In this case, the mapping F: A c X → X is said to be nonexpansive if

[+]The material in this section is taken directly from [7].

$$d(Fx, Fy) \leq d(x,y) \qquad \text{(A-3)}$$

for all $x,y \in A$. The mapping F is strictly nonexpansive if strict inequality holds in (A-3) when over $x \neq y$. The proofs of convergence and details on the properties of these mappings can be found in [7].

```
C*************************************************************************
C
C       COMPUTATION OF PHASE
C
C       DG FORTRAN 5 SOURCE FILENAME:          PHASE.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING   KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                   PROGRAMMER
C       --------        ----                   ----------
C       00.0            JAN 12, 1983           P.HARAN
C
C*************************************************************************
C
C       PURPOSE
C
C       THIS ROUTINE COMPUTES THE PHASE OF THE GIVEN
C       ONE DIMENSIONAL SIGNAL. IF DESIRED, PHASE
C       UNWRAPPING CAN BE DONE.
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               CFFT
C
C*************************************************************************
C
        DIMENSION X(512)
        COMPLEX X
        LOGICAL SIGN,LARGE
C
C       INITIALIZATION
C
        SIGN=.FALSE.
        LARGE=.FALSE.
        PHA1=0.0
        THETA=0.0
        I=1
C
C       GET THE INPUTS NEEDED
C
        ACCEPT ' HOW MANY POINTS ',N
        CALL QUERY(' INPUT BY HAND OR DISKFILE ',IANS)
        IF(IANS.LT.0)GO TO 300
C
C       INPUT THROUGH TERMINAL
C
        DO 10 I=1,N
        WRITE(10,1) I
1       FORMAT(' INPUT X(',I3,')-> ',Z)
        ACCEPT ' ',B
        X(I)=CMPLX(B,0.0)
10      CONTINUE
        GO TO 200
C
C       INPUT THROUGH DISK FILE
C
300     CALL OPENR(0,' INPUT FILENAME ? ',4,F1)
```

```
100     CALL READRW(0,I,A,1,IER)
        CALL CHECK(IER)
        X(I)=CMPLX(A,0.0)
        I=I+1
        IF(I.GT.N)GO TO 200
        GO TO 100
200     CONTINUE
C
C       CHOOSE THE DESIRED FFT SIZE
C
        ACCEPT ' HOW MANY POINT FFT ',M
C
C       APPEND WITH ZEROS FOR THE DESIRED FFT SIZE
C
        DO 40 I=N+1,M
        X(I)=(0.0,0.0)
40      CONTINUE
        K=M
C
C       COMPUTE THE FFT
C
        CALL FFT(X,K,0)
        CALL OPENW(1,' PHASE FILENAME ? ',4,F2)
        CALL QUERY(' WNWRAP PHASE ? ',IANS)
        IF(IANS.LE.0)GO TO 500
C
C       THEORITICALLY THIS IS 2*3.1415
C
        ACCEPT ' DISCONTINUITY SIZE ',DEL
500     CONTINUE
        DO 20 J=1,M
        D=REAL(X(J))
        C=AIMAG(X(J))
C
C       CALCULATE THE PHASE
C
        Y=C/D
        PHA=ATAN(Y)
C
C       FIND OUT WHICH QUADRANT THE COMPLEX VECTOR IS IN.
C
        IF(C.LT.0.AND.D.LT.0)PHA=-3.14+PHA
        IF(C.GT.0.AND.D.LT.0)PHA=3.14+PHA
C
C       UNWRAP PHASE IF DESIRED
C
        IF(IANS.LT.0)GO TO 600
        IF(PHA1*PHA.LT.0)SIGN=.TRUE.
        IF(ABS(PHA1-PHA).GT.DEL)LARGE=.TRUE.
        UNWRAP=0.0
        IF(LARGE.AND.SIGN)UNWRAP=6.28
        IF(PHA.GT.0)UNWRAP=-UNWRAP
        THETA=THETA+UNWRAP
        PHA1=PHA
        PHA=PHA+THETA
        SIGN=.FALSE.
        LARGE=.FALSE.
```

```
C
C       OUTPUT THE PHASE FUNCTION
C
600     CALL WRITRW(1,J,PHA,1,IER)
        CALL CHECK(IER)
20      CONTINUE
        STOP
```

```
C*********************************************************************
C
C       SIGNAL RECONSTRUCTION FROM PHASE
C
C       DG FORTRAN 5 SOURCE FILENAME:           MAG.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            JAN 12, 1983            P.HARAN
C
C*********************************************************************
C
C       PURPOSE
C
C       THIS PROGRAM RECONSTRUCTS THE SEQUENCE FROM THE GIVEN
C       PHASE FUNCTION USING THE NONEXPANSIVE ITERATIVE METHOD.
C
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               DFFT
C               OPENW
C               OPENR
C
C*********************************************************************
C
        DIMENSION XPHASE(512),XMAG(512),X(512),Y(512),ERR(1000)/1000*0.0/
        COMPLEX X
        DATA XMAG/512*1.0/,Y/512*0.0/
C
C       READ THE PHASE FUNCTION
C
        ACCEPT ' HOW MANY POINTS IN PHASE FILE ',N
        CALL OPENR(0,' PHASE FILENAME ? ',4,F1)
        CALL READRW(0,1,XPHASE,N,IC,IER)
        CALL CHECK(IER)
C
C       GET ALL OTHER INPUTS NEEDED
C
        ACCEPT ' LENGTH OF SEQUENCE ',M
        ACCEPT ' MAXIMUM # OF ITERATIONS ',NITER
        ACCEPT ' SCALE FACTOR ',SCALE
        ACCEPT ' OUTPUT INTERVAL ',NOUT
        CALL OPENW(1,' OUTPUT SEQUENCE FILENAME ? ',4,F2)
        CALL OPENR(3,'ACTUAL SEQUENCE FILENAME ? ',4,F1)
        CALL READR(3,1,Y,M,IC,IER)
        CALL CHECK(IER)
C
C       INITIALIZATION
C
        NDUMP=0
        ITER=0
        IREC=1
C
C       THE PROGRAMMING LOOP BEGINS HERE
```

```
C
C
C       CALCULATE X(w) FROM THE PHASE AND THE MAGNITUDE
C
100     DO 10 I=1,N
        XREAL=XMAG(I)*COS(XPHASE(I))
        XIMAG=XMAG(I)*SIN(XPHASE(I))
        X(I)=CMPLX(XREAL,XIMAG)
10      CONTINUE
C
C       COMPUTE THE INVERSE FFT
C
        CALL FFT(X,N,1)
C
C       OUT PUT THE CURRENT ESTIMATE OF THE SEQUENCE
C       IF DESIRED.
C
        IF(NNDUMP.NE.NOUT)GO TO 300
        C=REAL(X(1))
        DO 40 I=1,N
        XREAL=REAL(X(I))*SCALE/C
        CALL WRITRW(1,IREC,XREAL,1,IER)
        IREC=IREC+1
40      CONTINUE
        NDUMP=0
300     NDUMP=NDUMP+1
C
C       MULTIPLY THE SEQUENCE BY THE SCALE FACTOR.
C
        C=REAL(X(1))
        DO 60 I=1,M
        XREAL=REAL(X(I))
        XREAL=XREAL/C*SCALE
        XERR=Y(I)-XREAL
C
C       CALCULATE THE MEAN SQUARE ERROR
C
        ERR(ITER+1)=ERR(ITER+1)+XERR*XERR
        X(I)=CMPLX(XREAL,0.0)
60      CONTINUE
C
C       SET x(n)=0 OUTSIDE THE INTERVAL 0<n<N
C
        DO 20 I=M+1,N
        X(I)=(0.0,0.0)
20      CONTINUE
        WRITE(10,1) ITER
1       FORMAT(' ITERATION # ',I3,'  IS COMPLETED ')
        IF(ITER.GE.NITER)GO TO 400
        ITER=ITER+1
C
C       COMPUTE THE FFT
C
        CALL FFT(X,N,0)
C
C       ESTIMATE THE CURRENT MAGNITUDE
C
```

```
      DO 30 I=1,N
      XREAL=REAL(X(I))
      XIMAG=AIMAG(X(I))
      XMAG(I)=XREAL*XREAL+XIMAG*XIMAG
      XMAG(I)=SQRT(XMAG(I))
30    CONTINUE
      GO TO 100
400   CALL OPENW(2,' FINAL OUTPUT FILENAME ',4,IER)
      IREC=1
C
C     OUTPUT THE ERROR FUNCTION
C
      CALL OPENW(4,'ERROR FILENAME ? ',4,F1)
      CALL WRITR(4,1,ERR,NITER,IER)
      CALL CHECK(IER)
C
C     OUTPUT THE RECONSTRUCTED SEQUENCE
C
      DO 50 I=1,M
      XREAL=REAL(X(I))
      CALL WRITRW(2,IREC,XREAL,1,IER)
      CALL CHECK(IER)
      IREC=IREC+1
50    CONTINUE
      STOP
      END
```

```
C*************************************************************************
C
C       SIGNAL RECONSTRUCTION FROM PHASE                                62
C
C       DG FORTRAN 5 SOURCE FILENAME:           NEWMAG.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            JAN 12, 1983            P.HARAN
C
C*************************************************************************
C
C       PURPOSE
C       THIS PROGRAM RECONSTRUCTS A SEQUENCE FROM THE GIVEN
C       PHASE FUNCTION USING THE ADAPTIVE RELAXATION METHOD.
C
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               OPENR
C               OPENW
C               DFFT
C
C*************************************************************************
C
        DIMENSION XPHASE(512),XMAG(512),X(512),XR(512)/512*0.0/
        DIMENSION ERR(0:511)/512*0.0/,X1(512)/512*0.0/
        COMPLEX X
        DATA XMAG/512*1.0/
C
C       READ THE PHASE FUNCTION
C
        ACCEPT ' HOW MANY POINTS IN PHASE FILE ',N
        CALL OPENR(0,' PHASE FILENAME ? ',4,F1)
        CALL READRW(0,1,XPHASE,N,IC,IER)
        CALL CHECK(IER)
C
C       GET ALL OTHER INPUTS NEEDED
C
        ACCEPT ' LENGTH OF SEQUENCE ',M
        CALL OPENR(3,'ACTUAL SEQUENCE FILENAME ? ',4,F1)
        CALL READR(3,1,XR,M,IC,IER)
        CALL CHECK(IER)
        ACCEPT ' MAXIMUM # OF ITERATIONS ',NITER
        ACCEPT ' SCALE FACTOR ',SCALE
C
C       INIALIZATION
C
        ITER=0
        IREC=1
C
C       THE MAIN PROGRAMMING LOOP BEGINS HERE
C
C
C       CALCULATE X(w) FROM THE PHASE AND THE MAGNITUDE
```

```
C
100     DO 10 I=1,N
        XREAL=XMAG(I)*COS(XPHASE(I))
        XIMAG=XMAG(I)*SIN(XPHASE(I))
        X(I)=CMPLX(XREAL,XIMAG)
10      CONTINUE
C
C       COMPUTE THE INVERSE FFT
C
        CALL FFT(X,N,1)
        C=REAL(X(1))
        XNMR=0.0
        DENM=0.0
        FACT=SCALE/C
C
C       CALCULATE THE ADAPTIVE PARAMETR LAMDA
C
        DO 70 I=M+1,N
        XREAL=REAL(X(I))*FACT
        A=X1(I)*X1(I)
        B=X1(I)*XREAL
        XNMR=XNMR+B-A
        DENM=DENM+(XREAL-X1(I))*(XREAL-X1(I))
70      CONTINUE
        XLAM=-XNMR/DENM
        IF(XLAM.EQ.0.0)XLAM=1.0
        DO 80 I=1,N
        X(I)=XLAM*FACT*X(I)+(1.0-XLAM)*X1(I)
        IF(I.EQ.1)C=REAL(X(I))
        X1(I)=REAL(X(I))/C*SCALE
80      CONTINUE
        C=REAL(X(1))
C
C       CALCULATE THE MEAN SQUARE ERROR
C
        DO 90 I=1,M
        XREAL=REAL(X(I))*SCALE/C
        ERR(ITER)=ERR(ITER)+(XREAL-XR(I))**2
        X(I)=CMPLX(XREAL,0.0)
90      CONTINUE
C
C       SET x(n)=0 OUTSIDE THE INTERVAL 0<n<N
C
        DO 20 I=M+1,N
        X(I)=(0.0,0.0)
20      CONTINUE
        WRITE(10,1) ITER
1       FORMAT(' ITERATION # ',I3,'  IS COMPLETED ')
        IF(ITER.GE.NITER)GO TO 400
        ITER=ITER+1
C
C       COMPUTE THE FFT
C
        CALL FFT(X,N,0)
C
C       CALCULATE THE CURRENT ESTIMATE OF THE MAGNITUDE
C
```

```
      DO 30 I=1,N
      XREAL=REAL(X(I))
      XIMAG=AIMAG(X(I))
      XMAG(I)=XREAL*XREAL+XIMAG*XIMAG
      XMAG(I)=SQRT(XMAG(I))
30    CONTINUE
      GO TO 100
C
C
400   CALL OPENW(2,' FINAL OUTPUT FILENAME ',4,IER)
C
C     OUTPUT THE ERROR FUNCTION
C
      CALL OPENW(4,'ERROR OUTPUT FILENAME ? ',4,F1)
      CALL WRITR(4,1,ERR,NITER,IER)
      CALL CHECK(IER)
C
C     OUTPUT THE RECONSTRUCTED SEQUENCE
C
      IREC=1
      DO 50 I=1,M
      XREAL=REAL(X(I))
      CALL WRITRW(2,IREC,XREAL,1,IER)
      CALL CHECK(IER)
      IREC=IREC+1
50    CONTINUE
      STOP
      END
```

```
C*************************************************************************
C
C       SIGNAL RECONSTRUCTION FROM PHASE
C
C       DG FORTRAN 5 SOURCE FILENAME:           NEW.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            JAN 12, 1982            P.HARAN
C
C*************************************************************************
C
C       PURPOSE
C
C       THIS PROGRAM RECONSTRUCTS THE SIGNAL FROM THE GIVEN
C       PHASE FUNCTION USING THE GRADIENT METHOD.
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               OPENR
C               OPENW
C               CFFT
C
C*************************************************************************
C
        DIMENSION X(0:511),ERR(512),PHASE(512),XA(0:255),XERR(1000)
        COMPLEX ERR,PHASE,CTEMP,ATEMP(512)
        REAL NMR
        DATA XA/256*0.0/,XERR/1000*0.0/
        ITER=0
C
C       GET THE PHASE FUNCTION
C
        ACCEPT ' HOW MANY POINTS IN PHASE FILE ? ',N
        CALL OPENR(0,' PHASE FILENAME ? ',N*4,F1)
        CALL READR(0,1,X,1,IC,IER)
        CALL CHECK(IER)
C
C       FORM THE PHASE AS A COMPLEX ARRAY
C
        DO 10 I=0,N-1
        PHREAL=COS(X(I))
        PHIMAG=SIN(X(I))
        PHASE(I+1)=CMPLX(PHREAL,PHIMAG)
10      CONTINUE
C
C       READ IN THE ACTUAL SEQUENCE
C
        ACCEPT ' HOW MANY POINTS IN THE SEQUENCE ? ',M
        CALL OPENR(1,' ACTUAL SEQUENCE FILENAME ? ',M*4,F1)
        CALL READR(1,1,XA,1,IC,IER)
        CALL CHECK(IER)
C
C       INPUT ALL PARAMETERS
C
```

```
      ACCEPT ' HOW MANY ITERATIONS ? ',NITER
      ACCEPT ' CONVERGENCE PARAMETER MU ',XMU
C
C     START WITH UNIT MAGNITUDE AND ESTIMATE
C     THE STARTING SIGNAL VECTOR.
C
200   DO 20 I=1,N
20    ERR(I)=PHASE(I)
C
C     COMPUTE THE INVERSE FFT
C
      CALL FFT(ERR,N,1)
C
C     SCALE THE SEQUENCE
C
      SCALE=XA(0)
      C=REAL(ERR(1))
      C=SCALE/C
      DO 30 I=1,M
      X(I-1)=REAL(ERR(I))*C
30    CONTINUE
C
C     THE MAIN PROGRAM LOOP BEGINS HERE
C
C
C     CALCULATE THE MEAN SQUARE ERROR
C
100   DO 40 I=0,M-1
      XERR(ITER+1)=XERR(ITER+1)+(XA(I)-X(I))**2
40    CONTINUE
C
C     FORM THE VECTOR Y
C
      ERR(1)=CMPLX(XA(0),0.0)
      DO 50 I=2,M
      ERR(I)=CMPLX(X(I-1),0.0)
50    CONTINUE
C
C     THIS PART APPENDS THE VECTOR x WITH ZEROS
C
      DO 60 I=M+1,N
60    ERR(I)=CMPLX(0.0,0.0)
C
C     COMPUTE THE INVERSE FFT
C
      CALL FFT(ERR,N,1)
C
C
      DO 70 I=1,N
C
C     MULTIPLY BY THE PHI MATRIX
C
      ERR(I)=ERR(I)*PHASE(I)
C
C     EXTRACT THE IMAGINARY PART
C
      TEMP=AIMAG(ERR(I))
```

```
        CTEMP=CMPLX(TEMP,0.0)
C
C       MULTIPLY BY THE PHI MATRIX AGAIN                                  67
C
        ERR(I)=CTEMP*PHASE(I)
70      CONTINUE
C
C       COMPUTE THE INVERSE FFT
C
        CALL FFT(ERR,N,1)
        NMR=0.0
C
C       CALCULATE THE NUMERATOR PART OF MU
C
        DO 110 I=2,M
        NMR=NMR+AIMAG(ERR(I))**2
110     CONTINUE
        ATEMP(1)=(0.0,0.0)
C
C       CALCULATE THE DENOMINATOR PART OF MU
C
        DO 120 I=2,M
        TEMP=AIMAG(ERR(I))
        ATEMP(I)=CMPLX(TEMP,0.0)
120     CONTINUE
        DO 130 I=M+1,N
130     ATEMP(I)=(0.0,0.0)
        CALL FFT(ATEMP,N,1)
        DENM=0.0
        DO 140 I=1,N
        CTEMP=ATEMP(I)*PHASE(I)
        TEMP=AIMAG(CTEMP)
        DENM=DENM+TEMP*TEMP
140     CONTINUE
C
C       CALCULATE MU
C
        XMU=NMR/DENM
C
C       UPDATE THE ESTIMATE OF THE SIGNAL
C
        DO 80 I=1,M-1
        TEMP=AIMAG(ERR(I+1))
        X(I)=X(I)-XMU*TEMP
80      CONTINUE
        ITER=ITER+1
C
C       OUTPUT THE CURRENT ERROR
C
        WRITE(10,1) ITER,XERR(ITER)
1       FORMAT(' ITERATION # = ',I6,' ERROR = ',G16.8)
        IF(ITER.LE.NITER)GO TO 100
C
C       OUTPUT THE RECONSTRUCTED SEQUENCE
C
        CALL OPENW(2,'FINAL OUTPUT FILENAME ? ',4,F1)
        CALL WRITR(2,1,X,M,IER)
```

```
      CALL CHECK(IER)
C
C     OUTPUT THE ERROR FUNCTION                                        68
C
      CALL OPENW(3,' ERROR OUTPUT FILENAME ? ',4,F1)
      CALL WRITR(3,1,XERR,NITER,IER)
      CALL CHECK(IER)
      STOP
      END
```

```
C*************************************************************************
C
C       TWO DIMENSIONAL SIGNAL RECONSTRUCTION FROM PHASE              69
C
C       DG FORTRAN 5 SOURCE FILENAME:           TWOD.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            JAN 12, 1983            P.HARAN
C
C*************************************************************************
C
C       PURPOSE
C
C       THIS PROGRAM RECONSTRUCTS A TWO DIMENSIONAL SIGNAL
C       FROM ITS GIVEN PHASE FUNCTION USING THE NONEXPANSIVE
C       ITERATIVE SCHEME. THIS USES TWO-DIMENSIONAL FFTS
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               OPENR
C               OPENW
C               TDOMN
C               FDOMN
C               FFT2D
C
C*************************************************************************
C
        INTEGER IOFILE(13),PHFILE(13),IUFD(18)
        COMMON /OPN/  IUFD,IOFILE
        ITER=0
C
C       GET THE PHASE FILENAME
C
        CALL OPENR(0,' PHASEFILENAME ? ',4,F1)
        DO 10 I=1,13
10      PHFILE(I)=IOFILE(I)
        CALL CLOSE(0,IER)
C
C       GET THE DATA FILENAME
C
        CALL OPENW(1,' OUTPUT FILENAME ',4,F1)
        CALL CLOSE(1,IER)
C
C       GET ALL INPUT PARAMETERS
C
        ACCEPT ' IMAGE SIZE ',N
        ACCEPT ' FFT SIZE ',M
        ACCEPT ' VALUE OF THE IMAGE AT ORIGIN ',XO
        ACCEPT ' MAXIMUM NUMBER OF ITERATIONS ',NITER
        ACCEPT ' PAUSE AT ?  ',INTER
C
C       THE MAIN PROGRAM LOOP BEGINS HERE
C
C
```

```
C       APPLY FREQUENCY DOMAIN CONSTRAINTS
C
100     CALL FDOMN(IOFILE,PHFILE,M)
C
C       COMPUTE THE 2-D IFFT
C
        CALL DFT2D(IOFILE,M,1)
C
C       APPLY THE TIME DOMAIN CONSTRAINTS
C
        CALL TDOMN(IOFILE,M,N,XO)
C
C       CHECK IF PAUSING IS DESIRED
C
        IF(ITER.NE.INTER)GO TO 300
C
C       THIS IS DONE TO GET THE CURRENT
C       RECONSTRUCTED SEQUENCE
C
        PAUSE ' PAUSING TO CHECK RESULTS '
        ACCEPT ' NEXT POINT TO PAUSE ',INTER
300     IF(ITER.GE.NITER) GO TO 200
C
C       COMPUTE THE 2-D FFT
C
        CALL DFT2D(IOFILE,M,0)
        ITER=ITER+1
        WRITE(10,1) ITER
1       FORMAT(//,'ITERATION NUMBER ',I3,' JUST COMPLETED ',//)
        GO TO 100
200     STOP
```

```
C*******************************************************************************
C
C       FREQUENCY DOMAIN CONSTRAINTS
C
C       DG FORTRAN 5 SOURCE FILENAME:           name of file on disk
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            JAN 12, 1983            P.HARAN
C
C*******************************************************************************
C
C       PURPOSE
C
C       THIS PROGRAM IMPLEMENTS THE FREQUENCY DOMAIN
C       CONSTRAINTS. THIS READS THE GIVEN ESTIMATE
C       OF THE SIGNAL SPECTRUM AND IMPLENTS THE
C       GIVEN PHASE CONDITIONS.
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               NONE
C
C
C       ARGUMENTS REQUIRED FROM THE CALLING ROUTINE
C
C       IOFILE -> 26 BYTE DATA CONTAINING THE NAME OF
C                 THE DATA FILE
C
C       PHFILE -> 26 BYTE DATA CONTAINING THE NAME OF
C                 THE PHASE FILE
C
C       M      -> SIZE OF THE FFT USED
C
C*******************************************************************************
C
        SUBROUTINE FDOMN(IOFILE,PHFILE,M)
        DIMENSION IOFILE(13),PHFILE(13),PH(512)
        INTEGER PHFILE
        COMPLEX X(512)
        COMMON /DUM/ X
        DATA ITER/1/,XMAG/1.0/
        IREC1=M*4
        IREC2=M*8
C
C       THIS IS INCLUDED TO CONTINUE THE ITERATION
C       FROM WHERE IT LAST STOPPED.
C
        IF(ITER.EQ.1)TYPE * INPUT 1 IF YOU ARE*
        IF(ITER.EQ.1)ACCEPT* IN THE MIDDLE OF ITERATION ',IANS
        IF(IANS.EQ.1)ITER=2
C
C       OPEN DATA FILE
C
        CALL OPEN(6,IOFILE,2,IREC2,IER)
```

```
      CALL CHECK(IER)
C
C     OPEN PHASE FILE
C
      CALL OPEN(7,PHFILE,2,IREC1,IER)
      CALL CHECK(IER)
C
C
      DO 10 I=1,M
      IF(ITER.EQ.1)GO TO 200
C
C     READ ONE ROW OF DATA
C
      CALL READR(6,I,X,1,IC,IER)
      CALL CHECK(IER)
C
C     READ ONE ROW OF PHASE
C
200   CALL READR(7,I,PH,1,IC,IER)
      CALL CHECK(IER)
C
C
      DO 20 J=1,M
      IF(ITER.EQ.1)GO TO 100
C
C     CALCULATE THE CURRENT MAGNITUDE
C
      XMAG=CABS(X(J))
C
C     APPLY THE PHASE CONDITIONS
C
100   XREAL=XMAG*COS(PH(J))
      XIMAG=XMAG*SIN(PH(J))
      X(J)=CMPLX(XREAL,XIMAG)
C
C
20    CONTINUE
C
C     OUTPUT THE NEW ESTIMATE OF THE FOURIER TRANS FORM
C
      CALL WRITR(6,I,X,1,IER)
      CALL CHECK(IER)
10    CONTINUE
C
C     CLOSE ALL FILES
C
      CALL CLOSE(6,IER)
      CALL CLOSE(7,IER)
      ITER=10
      RETURN
```

```
C*******************************************************************************
C
C       TIME DOMAIN CONSTRINTS
C
C       DG FORTRAN 5 SOURCE FILENAME:           TDOMN.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            JAN 12, 1983            P.HARAN
C
C*******************************************************************************
C
C       PURPOSE
C
C       THIS PROGRAM IMPLEMENTS THE TIME DOMAIN CONSTRINTS
C       IN THE PHASE ONLY ITERATION ALGORITHMS. THIS MAKES
C       THE SIGNAL ZERO OUTSIDE THE DEFINED INTERVAL AND
C       ALSO CORRECTS FOR ANY NEGETIVE VALUES FOR PEL
C       VALUES.
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               NONE
C
C
C       ARGUMENTS REQUIRED FROM THE CALLING ROUTINE
C
C       IOFILE -> 26 BYTE DATA CONTAINING THE NAME OF THE
C                 DATA FILE
C
C       M       -> FFT SIZE USED
C
C       N       -> SIZE OF THE SEQUENCE
C
C       X0      -> SCALE FACTOR
C
C*******************************************************************************
C
        SUBROUTINE TDOMN(IOFILE,M,N,X0)
        DIMENSION IOFILE(13),X(512)
        COMPLEX X
        COMMON /DUM/ X
        IREC=M*8
C
C       OPEN THE DATA FILE
C
        CALL OPEN(6,IOFILE,2,IREC,IER)
        CALL CHECK(IER)
C
C
        DO 10 I=1,N
C
C       READ ONE ROW OF DATA
C
        CALL READR(6,I,X,1,IC,IER)
```

```
C
C       FIND THE SCALE FACTOR
C
        IF(I.EQ.1)X1=X(1)
        CALL CHECK(IER)
C
C       SCALING IS DONE BY THE FOLLOWING LOOP
C
        DO 50 KK=1,N
        X(KK)=X(KK)/X1*X0
        A=X(KK)
C
C       CHECK FOR NEGETIVE VALUES AND CORRECT
C
        IF(A.LT.0)X(KK)=0.0
50      CONTINUE
C
C       SET THE SIGNAL EQUAL TO ZERO OUTSIDE THE
C       DEFINED INTERVAL.
C
        DO 20 J=N+1,M
        X(J)=(0.0,0.0)
20      CONTINUE
C
C       OUTPUT ONE ROW OF DATA
C
        CALL WRITR(6,I,X,1,IER)
        CALL CHECK(IER)
C
C
10      CONTINUE
C
C       SET THE SIGNAL EQUAL TO ZERO OUTSIDE THE
C       DEFINED INTERVAL
C
        DO 40 I=1,M
        X(I)=(0.0,0.0)
40      CONTINUE
C
C       OUTPUT THE DATA
C
        DO 30 I=N+1,M
        CALL WRITR(6,I,X,1,IER)
        CALL CHECK(IER)
30      CONTINUE
C
C       CLOSE THE DATA FILE
C
        CALL CLOSE(6,IERR)
        RETURN
```

```
C*************************************************************************
C
C       TWO DIMENSIONAL FFT PROGRAM                                    75
C
C       DG FORTRAN 5 SOURCE FILENAME:          FFT2D.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING   KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                   PROGRAMMER
C       --------        ----                   ----------
C       00.0            JAN 12, 1983           P.HARAN
C
C*************************************************************************
C
C       PURPOSE
C
C       THIS PROGRAM COMPUTES THE TWO DIMENSIONAL FFT USING
C       ONE-DIMENSIONAL FFTS. THIS IS DONE BY COMPUTING 1-D
C       FFT FIRST FOR ROWS AND THEN FOR COLUMNS.
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               DFFT
C
C
C       ARGUMENTS REQUIRED FROM THE CALLING ROUTINE
C
C       IOFILE -> 26 BYTE DATA CONTAINING THE NAME
C                 OF THE DATA FILE
C
C       M      -> SIZE OF THE FFT NEEDED
C
C       INV    -> 0 MEANS FFT AND 1 MEANS INVERSE FFT
C
C*************************************************************************
C
        SUBROUTINE DFT2D(IOFILE,M,INV)
        DIMENSION X(512),X2(512,8),BLK(8),IOFILE(13)
        COMPLEX X,X2,BLK
        COMMON /DUM/ X
        EQUIVALENCE (X(1),X2(1,1))
        IREC=M*8
C
C       OPEN THE DATA FILE
C
        CALL OPEN(6,IOFILE,2,IREC,IER)
        CALL CHECK(IER)
C
C       COMPUTE THE FFT ALONG ROWS
C
        DO 10 I=1,M
        CALL READR(6,I,X,1,IC,IER)
        CALL CHECK(IER)
        CALL FFT(X,M,INV)
C
C       WRITE TEMPRORARY DATA ONTO DISK
C
```

```
      CALL WRITR(6,I,X,1,IER)
      CALL CHECK(IER)
10    CONTINUE
      CALL CLOSE(6,IER)
      CALL CHECK(IER)
C
C     NOW THE PROBLEM IS COMPUTE THE FFT ALONG
C     COLUMNS. THIS IS DONE BY READING 8 COLUMNS
C     AT A TIME AND DO 8 FFTS. THE POINTER IREC
C     POINTS TO THE CURRENT RECORD BEING READ.
C
      LENGTH=64
      CALL OPEN(6,IOFILE,2,LENGTH,IER)
      CALL CHECK(IER)
      N=M/8
      DO 20 I=1,N
      DO 30 J=1,M
      IREC=N*(J-1)+I
150   CALL READR(6,IREC,BLK,1,IC,IER)
250   CALL CHECK(IER)
C
C     STORE THE 8 COLUMNS OF DATA IN THE ARRAY X2
C
      DO 40 K=1,8
      X2(J,K)=BLK(K)
40    CONTINUE
30    CONTINUE
C
C     TAKE 8 FFTS ALONG COLUMNS
C
      DO 50 K=1,8
      CALL FFT(X2(1,K),M,INV)
50    CONTINUE
C
C     OUTPUT 8 COLUMNS OF DATA
C
      DO 60 J=1,M
      DO 70 K=1,8
      BLK(K)=X2(J,K)/IFIX(M)
70    CONTINUE
      IREC=(J-1)*N+I
      CALL WRITR(6,IREC,BLK,1,IER)
      CALL CHECK(IER)
60    CONTINUE
      WRITE(10,1) I*8
1     FORMAT('EIGHT COLUMN DFTS ENDING WITH',I4,'  JUST FINISHED')
20    CONTINUE
C
C     CLOSE THE FILES
C
      CALL CLOSE(6,IER)
      CALL CHECK(IER)
      RETURN
```

```
C*************************************************************************
C
C       TWO DIMENSION FILTERING PROGRAM                                 77
C
C       DG FORTRAN 5 SOURCE FILENAME:           2DFIL.FR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER
C       --------        ----                    ----------
C       00.0            FEB 11, 1980            MYRON FLICKNER
C
C*************************************************************************
C
C       PURPOSE
C
C               THE ROUTINE DOES TWO DIMENSIONAL FILTERING OF
C               AN IMAGE INPUT FILE.  THE FILTERING IS DONE
C               IN THE FREQUENCY DOMAIN VIA THE 2 DIMENSIONAL FFT.
C               SEVERAL FILTER TYPE CAN BE IMPLEMENTED.
C
C       ROUTINE(S) CALLED BY THIS ROUTINE
C
C               BLPF
C               DFT5
C               OPENR
C               OPENW
C               READR
C               WRITR
C
C*************************************************************************
C
C       use this space for added information unique to this routine
C
C*************************************************************************
C
        COMPLEX IMAGE(64,64),YLINE(64)
        REAL XLINE(64)
        INTEGER ILINE(64)
        LOGICAL HP
        ACCEPT ' IMAGE SIZE (MUST BE A POWER OF TWO) ?  ',M
        CALL ASK( ' HIGH PASS FILTER ? ', HP )
        IF( .NOT. HP ) TYPE ' LOW PASS FILTER SELECTED '
        ACCEPT ' VALUE OF DO (CUTOFF FREQUENCY) ? ', DO
        ACCEPT ' ORDER OF FILTER ?  ',N
        CALL OPENR(0,' INPUT IMAGE FILENAME ? ',M,FSIZE)
        CALL OPENW(1,' OUTPUT FLOATING POINT FILENAME ? ',M*4,F)
C
C               GET BYTE DATA BY ROWS AND CONVERT TO COMPLEX
C               THEN TAKE FORWARD FFT BY ROWS
C
                DO 1000 NLINE=1,M
                CALL READR(0,NLINE,ILINE,1,ICNT,IERR)
C
                        DO 1005 NELEM=1,M
                        IBYTE=BYTE(ILINE,NELEM)*((-1)**(NELEM+NLINE))
1005                    IMAGE(NELEM,NLINE)=CMPLX(FLOAT(IBYTE),0.0)
```

```
C
                CALL DFT5(IMAGE(1,NLINE),M,0)
1000            CONTINUE
C
                DO 1010 NELEM=1,M
C
C               GET A COLUMN OF DATA
C
                        DO 1015 NLINE=1,M
1015                    YLINE(NLINE)=IMAGE(NELEM,NLINE)
C
C               TAKE FFT OF THE COLUMN
C
                CALL DFT5(YLINE,M,0)
C
C               MULTIPLY THE RESULT BY THE TRANFER FUNCTION
C
                IF( HP ) CALL BHPF(YLINE,M,NELEM,D0,N)
                IF( .NOT. HP )  CALL BLPF(YLINE,M,NELEM,D0,N)
C
C               TAKE INVERSE TRANSFORM OF THE COLUMN
C
                CALL DFT5(YLINE,M,1)
C
C               REPLACE COLUMN IN IMAGE MATRIX
C
                        DO 1020 NLINE=1,M
1020                    IMAGE(NELEM,NLINE)=YLINE(NLINE)
C
1010            CONTINUE
C
C               NOW TAKE INVERSE FFT OF ROWS
C
                DO 1025 NLINE=1,M
                CALL DFT5(IMAGE(1,NLINE),M,1)
C
C               MOVE THE DATA INTO REAL OUTPUT BUFFER
C
                        DO 1030 NELEM=1,M
                        R = REAL( IMAGE(NELEM,NLINE) )
                        AI = AIMAG( IMAGE(NELEM,NLINE) )
                        XLINE(NELEM) = SQRT( R*R + AI*AI )
1030                    CONTINUE
C
C               WRITE OUT THE OUTPUT BUFFER TO DISK
C
                CALL WRITR(1,NLINE,XLINE,1,IERR)
1025            CONTINUE
C
C       CLOSE ALL OPEN FILES
C
        CALL RESET
        STOP ' NORMAL TERMINATION '
        END
```

SIGNAL RECONSTRUCTION FROM PHASE

by

PRANATHARTHI HARAN

B. Tech, Indian Institute of Technology, Madras, 1981

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

ABSTRACT

This is a tutorial paper on Signal Reconstruction from phase. The conditions for uniqueness of a signal of known phase is discussed. Algorithms for the reconstruction are developed and a comparison of the different methods shows that Adaptive Relaxation is the best method. Image reconstruction using phase information is shown and the ability to remove blur is demonstrated.