MINIMUM ABSOLUTE ERROR AS AN IMAGE RESTORATION CRITERION

by

Chris Karaguleff

A Thesis Submitted to the Faculty of the

COMMITTEE ON OPTICAL SCIENCES (GRADUATE)

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCES

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 8 1

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interest of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Chris Karagaleff

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Bobby R. Hunt

BOBBY R. HUNT
Professor of Optical Sciences and
Systems and Industrial Engineering

4-16-81

Date

ACKNOWLEDGMENTS

I first want to thank my advisor, Dr. Bobby Hunt, for his help and guidance in selecting the topics considered in the following thesis, and also for introducing me to and exciting my interest in the field of image restoration.

The following individuals, all colleagues of mine, should also receive proper and appropriate attention:

Tony "Suburban" Ervin

Art "The Dart" Gmitro

Tom "Tom" Bruegge

Dave "Fingers" Winker

Were it not for the influences of these people, the thesis that follows would not be in its present form.

I would also like to thank the Burnett Brothers for their tremendous insights.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ABSTRACT

Minimum absolute error (MAE) is used as an image restoration criterion in trying to find an optimal solution to the classical imaging equation with additive noise. This problem is dealt with in the field of mathematics known as linear programming, and is formally known as the ℓ_1 solution to an overdetermined set of linear equations. A computer algorithm employing the simplex method of linear programming is used to find this solution. In cases where the image data is afflicted with occasional and sporadic outlier noise, it is hoped that the MAE criterion will provide more accurate restorations than other more popular (and more analytic) methods such as least squares or minimum mean square error.

CHAPTER 1

INTRODUCTION

The classical model of image formation is characterized by two different processes. The object, represented as a function f(u,v), is first blurred by a known (but not necessarily stationary) point spread function h(x,y;u,v). x and y are image plane coordinates; u and v are object plane coordinates. This blurring is described by the integral (Goodman, 1968):

$$\int f(u,v)h(x,y;u,v)dudv \tag{1.1}$$

The image modeling is completed by adding noise n(x,y) to the above integral:

$$g(x,y) = \int f(u,v)h(x,y;u,v)dudv + n(x,y) \tag{1.2}$$

In the discrete or digital domain the image, object and noise functions (g, f and n, respectively) become vectors via lexicographic stacking (Andrews and Hunt, 1977) of the sampled 2-D data. The point spread function h(x,y;u,v) becomes an mxn matrix H. This all gives

$$\vec{g} = H\vec{f} + \vec{n} , \tag{1.3}$$

where $\vec{g}$ and $\vec{n}$ are mx1, $\vec{f}$ is nx1, and H is mxn.

The problem of image restoration is to seek an object estimate, $\vec{f}'$, that is in some sense a more accurate rendering of the original

object than the available image data. The method of restoration used is strongly dependent on the characteristics of the additive noise. Most commonly the noise is modeled as being Gaussian, uncorrelated and signal independent. Within the modeling procedure the degree of noise is characterized by its standard deviation or a signal to noise ratio. A rather popular restoration method adopted in the presence of such noise is one that is guided by some type of least squares criterion. More specifically, one seeks a solution $\vec{f}'$ that minimizes the error function defined as

$$e(\vec{f}') = (\vec{g} - H\,\vec{f}')^t\,(\vec{g} - H\,\vec{f}')$$

$$= \sum_{i=1}^{m} [g_i - (\sum_{j=1}^{n} H_{ij}\, f_j')]^2 \qquad (1.4)$$

The squaring operation ensures that our error function is positive definite, and it lends itself well to differentiation and, thusly, to an analytic solution.

A problem that is implicitly present whenever such a squaring operation is imposed is that data points having a comparatively large noise value are weighted more heavily than points with less noise magnitude. This can be very detrimental in cases where it is known that the image data is afflicted with outlier noise: substantial "spikes" or "holes" that sporadically afflict only a few data points, due to sampling or transmission errors, within the entire set of image data. During the squaring process such points would contribute largely to the overall value of the error function. The restoration would thus place an

undeserved emphasis on "correcting" those data points associated with the outliers, most likely at the cost of poorly restoring other data points.

A way to avoid this unfair biasing of inaccurate data points would be to consider the absolute value of the error, rather than the square:

$$e(f') = \sum_{i=1}^{m} \left| g_i - \left(\sum_{j=1}^{n} H_{ij} f'_j \right) \right| \qquad (1.5)$$

The subject of this paper is to use an existing algorithm that finds a solution f' which minimizes the absolute error. The degree of success to which the minimum absolute error (MAE) criterion serves to restore the object when differing amounts of image noise (both gaussian and outlier) are present is compared to both least square and minimum mean square error restorations.

As a final note of introduction it should be pointed out that the mathematical techniques and modeling procedures employed throughout this paper are not limited to the domain of image restoration. Spectroscopy, atmospheric physics (where, for example, remote sensing data is inverted to calculate aerosol size distributions) and geophysics (interpretation of seismic data) are examples of different disciplines that must contend with methods of mathematical analysis that are comparable to those employed in image processing.

CHAPTER 2

THEORETICAL BACKGROUND

On the following pages we discuss in greater detail the theoretical development of each of the restoration methods to be employed.

Least-Squares

As mentioned earlier, our image formation model within the discrete regime can be stated as follows:

$$\vec{g} = H \vec{f} + \vec{n} \tag{2.1}$$

Solving for the noise component explicitly we obtain

$$\vec{n} = \vec{g} - H \vec{f} \tag{2.2}$$

The spirit of the least squares criterion is as follows: we first define an error function e,

$$e = \vec{n}^t \vec{n} = \sum_{i=1}^{m} n_i^2 \quad , \tag{2.3}$$

the superscript "t" implies "transpose." If we invoke Eq. (2.2) and substitute $\vec{f}'$ for $\vec{f}$, we can write e as a function of the object estimate:

$$\begin{aligned} e &= (\vec{g} - H \vec{f}')^t \, (\vec{g} - H \vec{f}') \\ &= \sum_{i=1}^{m} \left[g_i - \sum_{j=1}^{n} H_{ij} \, f_j' \right]^2 \end{aligned} \tag{2.4}$$

We now wish to determine an object estimate, $\vec{f}'$, that minimizes e. Mathematically, it is helpful to think of the function e as representing a surface in some n+1 dimensioned space, n being the number of independent components of the object vector $\vec{f}'$. As such, to find $\vec{f}'$ that minimizes e, there will be n independent equations that satisfy the relation $\frac{\partial e}{\partial f'_k} = 0$. More explicitly:

$$\frac{\partial e}{\partial f'_k} = \frac{\partial}{\partial f'_k} \sum_{i=1}^{m} [g_i - \sum_{j=1}^{n} H_{ij} f'_j]^2$$

$$= \sum_{i=1}^{m} \frac{\partial}{\partial f'_k} [g_i - \sum_{j=1}^{n} H_{ij} f'_j]^2$$

$$= \sum_{i=1}^{m} -2[g_i - \sum_{j=1}^{n} H_{ij} f'_j] H_{ik}$$

$$= -2 \sum_{i=1}^{m} H^t_{ki} [g_i - \sum_{j=1}^{n} H_{ij} f'_j] \qquad (\text{since } H_{ik} = H^t_{ki})$$

$$= -2 \sum_{i=1}^{m} H^t_{ki} g_i + 2 \sum_{i=1}^{m} H^t_{ki} [\sum_{j=1}^{n} H_{ij} f'_j] = 0. \qquad (2.5)$$

If we now employ matrix notation we obtain

$$H^t \vec{g} = H^t H \vec{f}' \qquad (2.6)$$

Since H is mxn and H^t is nxm, H^tH is nxn and thus invertible. This allows us to write the solution for $\vec{f}'$ directly as

$$\vec{f}' = (H^tH)^{-1} H^t \vec{g} \qquad (2.7)$$

This equation is known as the pseudo-inverse solution (Pratt, 1978). Stated simply, the least square error approach effectively says to ignore the presence of random noise and to simply invert the blurring process that was effected by the psf.

MMSE - Minimum Mean Square Error

In the least square error approach we sought a solution $\vec{f}'$ that fulfilled the sole criterion of minimizing the total image noise, $\sum_{i=1}^{m} n_i^2$. As the resulting solution suggests, this is one of the simplest and most straightforward approaches that can be taken. That is, we merely effect the inverse of the blurring procedure upon the available image data. The only presumptions inferred are that the object was blurred by a known spread function, and that the image data is contaminated by random, uncorrelated noise.

Realistically, considerably more is known about the image forming process than what is intimated above. For example, one generally has knowledge regarding the degree of noise incurred. That is, we typically know the signal to noise ratio, or noise variance, of our data as effected by the particular form of image data acquisition that was invoked. Furthermore, one often has some notion of what the actual object looks like. For example, the image may be a blurred and noisy rendition of a page of text. Thus we can consider our ensemble of possible object distributions to be pages of text, or rather, evenly spaced rows of varying sequences of letters of the alphabet and punctuation marks. Obviously, the statistical properties of such a class of possible object

distributions differ markedly from such objects as, say, a jet aircraft situated against a uniform background (the sky), or the bone structure of an individual with a broken limb.

The considerations discussed above are invoked in the MMSE restoration method. The available information regarding the ensemble of possible object distributions is characterized by the autocorrelation matrix R_f:

$$R_f = E\{\vec{f}\,\vec{f}^t\} , \tag{2.8}$$

where "E" connotes "expected value." The noise behavior can similarly be characterized by the autocorrelation matrix R_n:

$$R_n = E\{\vec{n}\,\vec{n}^t\} . \tag{2.9}$$

If the noise is uncorrelated, R_n is diagonal, with diagonal elements $(R_n)_{ii}$ equal to σ_n^2, the noise variance.

In light of the above discussion, let us now develop the MMSE restoration method. In the manner of Andrews and Hunt (1977) we begin by defining, almost glibly, the error vector $\vec{e}$ as

$$\vec{e} = \vec{f} - \vec{f}' . \tag{2.10}$$

In a manner similar to the least squares method, we define an error function $e(\vec{f}')$ that is positive definite as

$$e(\vec{f}) = \sum_{i=1}^{m} e_i^2 = \vec{e}^t\,\vec{e} = \mathrm{Tr}(\vec{e}\,\vec{e}^t) , \tag{2.11}$$

where "Tr" means the trace of the corresponding matrix. The problem posed by the MMSE method is to find an object estimate $\vec{f}'$ that minimizes

the expected value of e:

$$\underset{\text{respect to } \vec{f}'}{\text{minimize with}} \quad E\{e(\vec{f}')\} = E\{\mathrm{Tr}(\vec{e}\,\vec{e}^{t})\} \tag{2.12}$$

We next assert that the optimum solution $\vec{f}'$ is obtained via a linear operation, {L}, upon the image data,

$$\vec{f}' = L\,\vec{g}\ . \tag{2.13}$$

Thus,

$$E\{\mathrm{Tr}(\vec{e}\,\vec{e}^{t})\} = E\{\mathrm{Tr}[(\vec{f}-\vec{f}')(\vec{f}-\vec{f}')^{t}]\}$$

$$= E\{\mathrm{Tr}[(\vec{f}-L\,\vec{g})(\vec{f}-L\,\vec{g})^{t}]\} \tag{2.14}$$

Since $\vec{g} = H\,\vec{f} + \vec{n}$, we obtain:

$$E\{\mathrm{Tr}(\vec{e}\,\vec{e}^{t})\} = E\{\mathrm{Tr}[(\vec{f}-L(H\vec{f}+\vec{n}))(\vec{f}-L(H\vec{f}+\vec{n}))^{t}]\}$$

$$= E\{\mathrm{Tr}[\vec{f}\vec{f}^{t} - L(H\vec{f}\vec{f}^{t} + \vec{n}\vec{f}^{t})$$

$$-(\vec{f}\vec{f}^{t}\,H^{t} + \vec{f}\,\vec{n}^{t})L^{t}$$

$$+ L(H\vec{f}\vec{f}^{t}\,H^{t} + \vec{n}\,\vec{f}^{t}\,H^{t} + H\,\vec{f}\,\vec{n}^{t} + \vec{n}\vec{n}^{t})L^{t}$$

(2.15)

We may now interchange the order of trace and expectation operations, since trace is linear. This gives:

$$E\{\mathrm{tr}(\vec{e}\vec{e}^{t})\} = \mathrm{Tr}\Big\{E\{\vec{f}\vec{f}^{t}\} - LE\{H\vec{f}\vec{f}^{t} + \vec{n}\vec{f}^{t}\}$$

$$- E\{\vec{f}\vec{f}^{t}\,H^{t} + \vec{f}\,\vec{n}^{t}\}\,L^{t} \tag{2.16}$$

$$+ LE\{H\vec{f}\vec{f}^{t}\,H^{t} + \vec{n}\vec{f}^{t}\,H^{t} + H\vec{f}\vec{n}^{t} + \vec{n}\vec{n}^{t}\}L^{t}\Big\}$$

We now substitute in Eqs. (2.8) and (2.9), and also note that for uncorrelated and signal independent noise, the expectation value of any terms linear in $\vec{n}$ or $\vec{n}^t$ become zero.

$$E\{tr(\vec{e}\vec{e}^t)\} = Tr\{R_f - 2LHR_f + LHR_f H^t L^t + L R_n L^t\} \qquad (2.17)$$

For convenience, let us define the matrix A:

$$A = R_f - 2LHR_f + LHR_f H^t L^t + L R_n L^t \qquad (2.18)$$

The diagonal elements of A, A_{ii}, are

$$A_{ii} = R_{fii} - 2 \sum_j \sum_k L_{ij} H_{jk} R_{fki}$$

$$+ \sum_j \sum_k \sum_l \sum_m L_{ij} H_{jk} R_{fkl} H_{ml} L_{im}$$

$$+ \sum_j \sum_k L_{ij} R_{njk} L_{ik} \qquad (2.19)$$

Of course, the trace of A is just

$$Tr(A) = \sum_i A_{ii} \qquad (2.20)$$

As per expression (2.12), we want to minimize Tr(A) with respect to $\vec{f}'$. But since we've set $\vec{f}'$ equal to L $\vec{g}$, we can equivalently minimize with respect to each element L_{pq} of L. So, we differentiate (2.20) with respect to L_{pq} and set these derivatives to zero. This gives

$$0 = \frac{d}{dL_{pq}} \mathrm{Tr}(A)$$

$$= -2 \sum_k H_{qk} R_{fkp} + \sum_k \sum_l \sum_m H_{qk} R_{fkl} H_{ml} L_{pm}$$

$$+ \sum_j \sum_k \sum_l L_{pj} H_{jk} R_{fkl} H_{ql} + \sum_k R_{nqk} L_{pk}$$

$$+ \sum_j L_{pj} R_{njq} \tag{2.21}$$

Rewriting this expression in matrix notation we obtain

$$0 = -2HR_f + HR_fH^tL^t + [LHR_fH^t]^t$$

$$+ R_nL^t + [LR_n]^t \tag{2.22}$$

We now transpose the entire expression, noting that since both R_f and R_n are symmetric, $R_f^t = R_f$ and $R_n^t = R_n$. This gives

$$0 = -2R_fH^t + 2LHR_fH^t + 2LR_n \ . \tag{2.23}$$

Continuing, we obtain

$$LHR_fH^t + LR_n = R_fH^t \tag{2.24}$$

$$L[HR_fH^t + R_n] = R_fH^t \tag{2.25}$$

Finally, we obtain for our restoration matrix L:

$$L = R_fH^t[HR_fH^t + R_n]^{-1} \tag{2.26}$$

Let us now develop the results of Eq. (2.26) for the following relatively simple circumstances. Let's presume that all we know about our image data is the signal to noise ratio and the noise variance.

$$\sigma_n^2 = \text{noise variance}$$

$$= \left\{ \sum_{i=1}^{m} n_i^2 \right\} / m \tag{2.27}$$

and

$$SNR = \left[\sum_{i=1}^{m} g_i^2 \Big/ \sum_{i=1}^{m} n_i^2 \right]^{\frac{1}{2}} . \tag{2.28}$$

From these quantities we infer an object variance σ_f^2 to be simply

$$\sigma_f^2 = (SNR)^2 \, \sigma_n^2 \tag{2.29}$$

Since our noise is uncorrelated, R_n is of course diagonal, and is equal to

$$R_n = \sigma_n^2 \, I \tag{2.30}$$

where I is the mxm identity matrix. Furthermore, possessing no greater knowledge of our object ensemble other than its variance, we note that R_f is also diagonal, and:

$$R_f = \sigma_f^2 \, I \tag{2.31}$$

where in this case I is the nxn identity matrix. If we substitute Eqs. (2.31) and (2.30) we obtain:

$$L = \sigma_f^2 I \, H^t [H(\sigma_f^2 I) H^t + \sigma_n^2 I]^{-1}$$

$$= \sigma_f^2 \, H^t [\sigma_f^2 H H^t + \sigma_n^2 I]^{-1} . \tag{2.32}$$

This result is employed experimentally and compared to the results encountered by the least squares and minimum absolute error methods.

One final observation: for the case of zero (or negligible) noise variance σ_n^2, Eq. (2.32) reduces to:

$$L = \sigma_f^2 H^t [\sigma_f^2 HH^t]^{-1}$$

$$= H^t [HH^t]^{-1}$$

$$HL = HH^t [HH^t]^{-1}$$

$$HL = I$$

$$H^t HL = H^t$$

$$L = [H^t H]^{-1} H^t \qquad (2.33)$$

This result is equivalent to the pseudo-inverse solution of Eq. (2.7).

Minimum Absolute Error: Linear Programming and the Simplex Method

We seek a solution, $\vec{f}'$, to the classical image forming equation that minimizes the sum of the absolute value of the error that exists implicitly within each data point. That is

$$\text{minimize } \vec{f}' \qquad e(f') = \sum_{i=1}^{m} \left| g_i - \left(\sum_{j=1}^{n} H_{ij} f_j' \right) \right| \qquad (2.34)$$

Following the approach of Barrodale and Roberts (1973) we will show that Eq. (2.34) can be rewritten in a somewhat different mathematical form. More specifically, Eq. (2.34) will be restated as a standard problem that is dealt with commonly in the field of mathematical study known as linear programming. A solution to Eq. (2.34), in its new form, can be

obtained fairly directly by a well developed technique known as the simplex method.

In the following pages the general problem of linear programming and the simplex method itself will be discussed in some detail. We will then show how the problem stated in Eq. (2.34) can be massaged into a form that is compatible with the simplex method.

Linear programming grew out of the more general field of optimization theory. Linear programming deals with the problem of how to minimize (or maximize) the value an objective function, this latter quantity being a linear function of n non-negative variables, $\{x_i\}$. The objective function is accompanied by m linear equations of the variables x_i. Mathematically we have

$$\begin{array}{l}\text{minimize} \\ \text{(or maximize)}\end{array} \qquad f = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{2.35}$$

subject to

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\ &\quad\vdots \\ &a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i \\ &\quad\vdots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \end{aligned} \tag{2.36}$$

and $x_i \geq 0$ for $i = \{1,2,\ldots,n\}$. It will soon be shown that the system of linear inequalities (2.36) can correspondingly be linear equalities.

The motivation for the development of linear programming techniques arose primarily during the 1930's and 40's (Barsov, 1959). Then, as continues today, economists and mathematicians were becoming very

interested in analyzing the effects that new and constantly improving methods of production were having on the economy. The problem then was still rather academic.

During the 50's the advent of the electronic computer allowed the possibility for the actual implementation of the long and arduous calculations involved. In time computational methods were developed that were quite practical and efficient. Such success attracted the interest of industrial, business, and other commercial and governmental concerns, since they routinely encounter the type of optimization problems dealt with in linear programming.

The question may arise as to why traditional methods of optimization involving calculus and partial differentiation aren't used. This is because an optimum solution to the linear programming problem lies on a vertex point of the admissable solution set. Partial derivatives simply don't exist at such non-well behaved locations.

To better understand this latter point as well as other aspects of linear programming, we'll briefly consider an example that is substantially simplified (see, for example, Van de Panne, 1975). The number of variables will be small enough (i.e., two) to allow us to perform a graphic analysis of how an (not necessarily "the") optimum solution is found. Though it wouldn't be feasible to try and solve a real life problem in this manner, where hundreds (perhaps thousands, as may be the case in image restoration problems) of variables may be involved, such an approach will provide a method of visualizing the behavior of the objective function and the accompanying constraint equations.

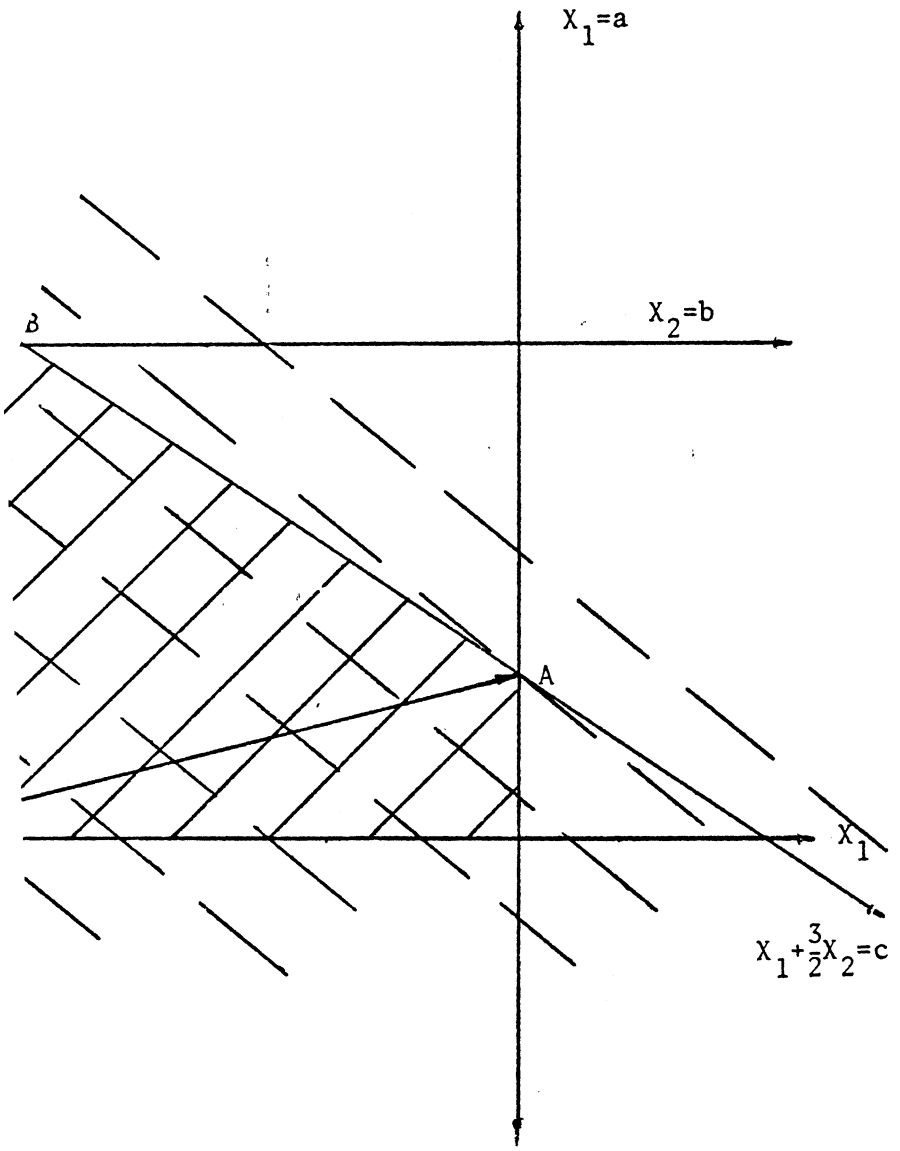
Consider the case of a factory producing two different products, with x_1 and x_2 representing the number produced of each product. If product two garners, say, 25% more profit per unit sold than product one, then the total profit may be written as

$$p = x_1 + 1.25\ x_2\ . \tag{2.37}$$

The problem is to decide how many of each product the factory should produce in order to maximize their profits. Without any constraints, they should of course produce nothing but product two. However, constraints do exist in the form of limitations of materials, the time required to produce each product, etc. For example, for each production run there exists enough materials to produce at most "a" of x_1 and "b" of x_2. Additionally, the man-hours for preparation of product two may be, say, 50% greater than that for product one on a per unit basis. If "c" represents the total number of man-hours effected within one production run we have the constraint $x_1 + \frac{3}{2} x_2 \leq c$. Of course, 2 additional constraints exist, manely that both x_1 and x_2 must be non-negative. Written together our constraints are

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_1 &\leq a \\ x_2 &\leq b \\ x_1 + \tfrac{3}{2} x_2 &\leq c \end{aligned} \tag{2.38}$$

These constraints can be represented graphically as portrayed in Figure 1. The cross hatched region is known as the feasible solution



Graph of region that represents the feasible solution set as determined by constraint equation 2.38.

Vertex points A and B represent basic feasible solutions.

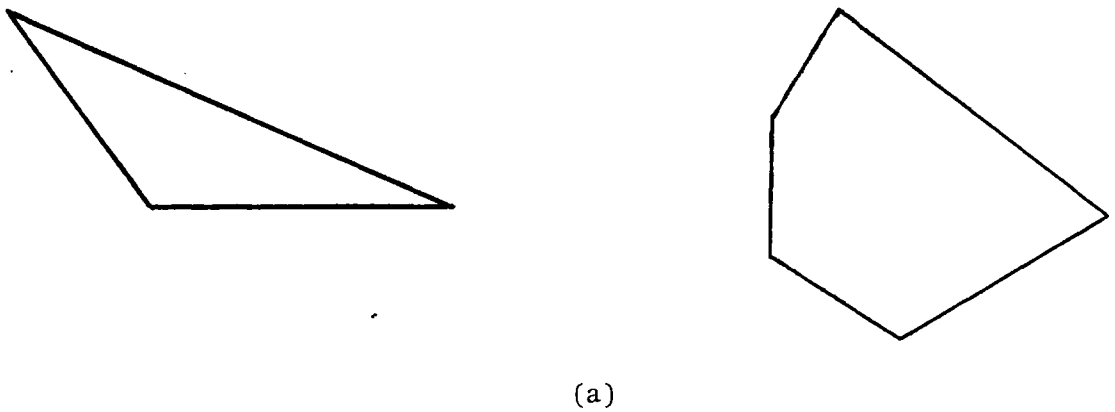
set as determined by the constraint equations. The family of dashed lines represents different values of the objective function (profit) as a function of x_1 and x_2. The optimum solution vector (x_1,x_2) will of course provide maximum profit (or equivalently, the greatest x_2-intercept of the family of objective function lines) and lie within the feasible solution set. For the present example we can readily identify this solution by inspection to be the point A.

One important feature that is depicted in Figure 1 is the convexity of the feasible solution set. A solution set is convex if, for any 2 points within the solution set, a line segment connecting these 2 points is also fully contained within the set. Figure 2 illustrates various convex and non-convex sets. This concept extends to any n-dimensioned set of points, and it can be shown that the set of points defined in n variables by m linear inequalities is necessarily convex (Barsov, 1959) (unless, of course, the set is null, as is the case for a set of inconsistent inequalities). This solution set is easily thought of as a multi-faceted polygon consisting of sides determined by the linear constraint equations, and edges and vertices determined by the intersections of the various hyperplanes.

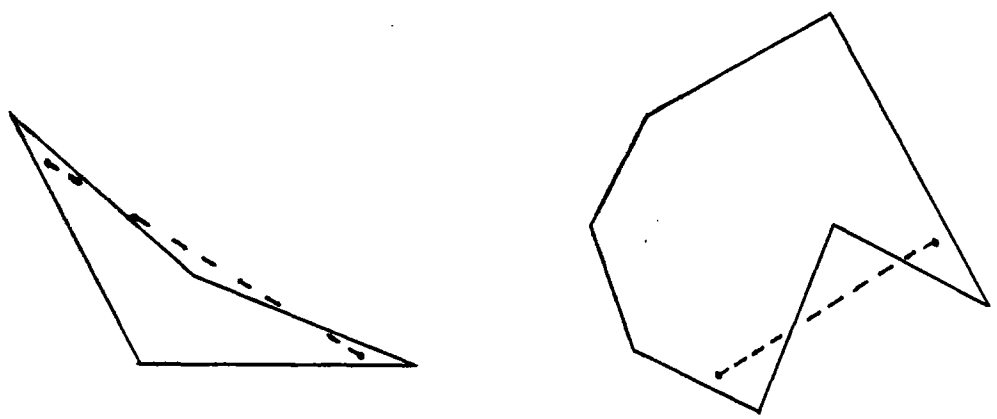
The objective function of our example was represented by a family of parallel lines. Similarly, for the case of n variables our objective function,

$$f = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{2.39}$$

represents a family of parallel hyper-planes in n-space. An increase (decrease) in the value of f corresponds to a hyperplane moving farther



(a)



(b)

Figure 2. Convex and non-convex solutions sets in two dimensions.

(a) Convex solution sets.
(b) Non-convex solutions sets.

from (closer to) the origin of our coordinate system. It should thus be apparent that an optimum solution, i.e., a minimum or maximum of f, occurs when one of these objective function hyper-planes intersects the convex polygon that is our feasible solution set at a vertex of that polygon. This point needn't be unique of course - the objective function hyperplane corresponding to an optimum solution could intersect an entire edge or face of the solution set. In our 2 dimensional example, this would be analogous to the case wherein the line segment AB of Figure 1 is parallel to the family of objective function straight lines.

It should now be apparent that each vertex point within the feasible solution set is potentially an optimum solution vector to the linear programming problem. Thus a method of solving the problem would be to determine each of these vertex points and evaluate the objective function at that point. Such a procedure would be impractical for application to problems with many variables and constraints. A better procedure was developed by Dantzig (1951), which he called the simplex method. In this method an extreme point (vertex) is selected and the objective function is evaluated. It then moves onto a neighboring extreme point, in the direction that increases (or decreases) the objective function. In this manner a maximum (or minimum) is obtained in between m and 2m iterations (Gass, 1958), where m is the number of linear constraints. Such a method is reliant on the implicit convexity of the solution set. If the solution-set "polygon" was concave at any point, then such a point would appear to be a local minimum or maximum, thus halting the iterative best-solution search process.

The first step in applying the simplex procedure is to rewrite our linear inequalities as linear equalities. This is done by introducing m new non-negative variables, where m is the number of inequalities. If such an alteration is performed on Eq. (2.38) of our example above, we must introduce the additional non-negative variables x_3, x_4 and x_5. Our new equations are

$$\begin{aligned} & x_1,\ x_2,\ x_3,\ x_4,\ x_5 \geq 0 \\ & x_1 + x_3 = a \\ & x_2 + x_4 = b \\ & x_1 + \frac{3}{2} x_2 + x_5 = c \end{aligned} \tag{2.40}$$

In this new form, the extreme points of the feasible solution set take on special significance. Letting n=5 represent the number of variables and m=3 represent the number of linear equalities, it can be asserted that the coordinates of any extreme point (which, as we pointed out earlier, is also a potential optimum solution) will contain at most m non-zero coordinates (Gass, 1958). (Equivalently, there can be no more than m variables that are linearly independent.) For example, vertex point A has x_1=a, which implies that x_3=0, and satisfies the equation $x_1 + \frac{3}{2} x_2 = c$, which implies that x_5=0. Thus at the point A

$$\begin{aligned} x_1 &= a \\ x_2 &= \frac{2}{3} (c-a) \\ x_3 &= 0 \\ x_4 &= b - \frac{2}{3} (c-a) \\ x_5 &= 0 \ . \end{aligned} \tag{2.41}$$

Similarly for point B it can be shown that

$$\begin{aligned} x_1 &= c - \frac{3}{2} b \\ x_2 &= b \\ x_3 &= a-c + \frac{3}{2} b \\ x_4 &= 0 \\ x_5 &= 0 \end{aligned} \tag{2.42}$$

More generally, the linear programming problem can be restated as follows: Maximize (or minimize) the objective function

$$f = c_1x_1 + c_2x_2 + \ldots + c_nx_n \ , \tag{2.43}$$

subject to the following constraints:

$$x_j \geq 0 \ , \ j = 1, 2, \ldots, n \tag{2.44}$$

and

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1_n}x_n &= b_1 \\ a_{21}x_1 + \ldots + a_{2_n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + \ldots + a_{mn}x_n &= b_m \end{aligned} \tag{2.45}$$

While it is definitely not as easy to visualize, our feasible solution set as expressed in this new form is still an n-m dimensioned manifold within n-space. Furthermore, as can be proven mathematically (Gass, 1958), it can be shown that the coordinates of any extreme point will consist of at most m non-zero values. Such a point is also known as a basic feasible solution, as distinguished from an optimum feasible solution. These solutions will have the following appearance:

$$(x_1, x_2, \ldots, x_{m-1}, x_m, 0, 0, \ldots, 0)$$

$$(x_1, x_2, \ldots, x_{m-1}, 0, x_{m+1}, 0, \ldots, 0)$$

$$(x_1, x_2, \ldots, 0, x_m, x_{m+1}, 0, 0, \ldots, 0) \qquad (2.46)$$

There will be, at most, $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ such basic solutions (Luenberger, 1973).

Conceptually, there is no great difficulty in determining only one existing basic (but not necessarily minimum) solution. We know that, generally, for a set of m equations in n unknowns, we can pick any one variable x_i and, by either straightforward Gaussian elimination or column and row operations upon the matrix of coefficients, obtain an expression for x_i as a linear function of any desired group of n-m remaining variables. This can be done simultaneously for, say, variables $\{x_i, 1 \leq i \leq m\}$ of Eqs. (2.45). This would provide the following:

$$\begin{aligned}
x_1 &= b_1 - (a_{1,m+1}x_{m+1} + a_{1,m+2}x_{m+2} + \ldots + a_{1,n}x_n) \\
x_2 &= b_2 - (a_{2,m+1}x_{m+1} + a_{2,m+2}x_{m+2} + \ldots + a_{2,n}x_n) \\
&\vdots \\
x_m &= b_m - (a_{m,m+1}x_{m+1} + a_{m,m+2}x_{m+2} + \ldots + a_{m,n}x_n)
\end{aligned} \qquad (2.47)$$

More compactly we have

$$x_i = b_i - \sum_{j=m+1}^{n} a_{ij}x_j , \qquad 1 \leq i \leq m \qquad (2.48)$$

When expressed in this form the variables $\{x_i; 1 \leq m\}$ are known as the basic variables. The remaining n-m x_i are the non-basic variables. Thus, once Eqs. (2.45) have been altered into a form similar to Eqs.

(2.47), a basic feasible solution is available by just setting the non-basic variables $\{x_i, i>m\}$ equal to zero.

The problem with the above procedure is that there is no way of knowing whether the solution we obtain will be positive; that is whether the basic variables x_i satisfy the non-negativity constraint $x_i \geq 0$. To find such positive solutions we begin by rewriting the given constraint equations in the following form:

$$\begin{aligned} b_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ &\vdots \\ b_i &= a_{i1}x_1 + \ldots + a_{in}x_n \\ &\vdots \\ b_m &= a_{m1}x_1 + \ldots + a_{mn}x_n \end{aligned} \tag{2.49}$$

or more compactly,

$$b_i = \sum_{j=1}^{n} a_{ij}x_j \quad , \qquad 1 \leq i \leq m \tag{2.50}$$

We assume that each b_i is non-negative. This is no problem - if any one $b_i<0$, we simply multiply that equation by -1.

For convenience, we construct the matrix of coefficients of Eqs. (2.49), and identify each column by an appropriately numbered vector $\vec{P}_j$.

$$\begin{array}{ccccccc} \vec{P}_0 & \vec{P}_1 & \vec{P}_2 & \ldots & \vec{P}_j & \ldots & \vec{P}_n \\ b_1 & a_{11} & a_{12} & \ldots & a_{1j} & \ldots & a_{1n} \\ b_2 & a_{21} & & & \vdots & & \vdots \\ \vdots & \vdots & & & & & \\ b_i & a_{i1} & & & a_{ij} & & \\ \vdots & \vdots & & & & & \\ b_m & a_{m1} & & & & & a_{mn} \end{array} \tag{2.51}$$

Before we may begin invoking the simplex method as briefly described above, we require an initial non-negative basic solution. With this initial solution we may begin our iterative search through other solutions (representing traveling along adjacent extreme points of our solution set) until we find our optimum solution.

The most straightforward and efficient method to determine such an initial solution is reminiscent of the way in which we transformed a set of linear inequalities to a set of linear equalities. To the set of Eqs. (2.49) we simply introduce m new positive variables x_j, where $n+1 \leq j \leq n+m$, such that Eqs. (2.49) now appear in the following somewhat altered form:

$$\begin{aligned}
b_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1_n}x_n + x_{n+1} \\
b_2 &= a_{21}x_1 + \ldots + a_{2_n}x_n + x_{n+2} \\
&\vdots \\
b_i &= a_{i1}x_1 + \ldots + a_{in}x_n + x_{n+i} \\
&\vdots \\
b_m &= a_{m1}x_1 + \ldots + a_{mn}x_n + x_{n+m}
\end{aligned} \tag{2.52}$$

These new variables are typically called slack variables. The coefficient matrix corresponding to this revised set of constraint equations appears as follows:

$$\begin{array}{cccccccccc}
\vec{P}_0 & \vec{P}_1 & \vec{P}_2 & \ldots & \vec{P}_n & \vec{P}_{n+1} & \vec{P}_{n+2} & \ldots & \vec{P}_{n+m} & \\
b_1 & a_{11} & a_{21} & \ldots & a_{1_n} & 1 & 0 & \ldots & 0 & \\
b_2 & a_{21} & a_{22} & \ldots & a_{2_n} & 0 & 1 & & 0 & \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \} m \\
b_m & a_{m1} & a_{m2} & \ldots & a_{mn} & 0 & 0 & & 1 & \\
 & & & & & \underbrace{\qquad\qquad}_{m} & & & &
\end{array} \tag{2.53}$$

From such a form we can immediately recognize that the variables x_{n+1} through x_{n+m} are basic variables, and as before, a basic feasible solution is realized by merely setting the non-basic variables x_1 through x_n equal to zero.

Having obtained an initial basic solution, we now describe the method by which one obtains a new extreme point solution. We select one of the m equations, say the m^{th}, that contains a positive coefficient a_{mj}. For simplicity let us assume $a_{m\ell}$ is such a positive quantity. Having decided on an appropriate coefficient, we sort through all positive $a_{i\ell}$'s within the ℓ^{th} column of the coefficient matrix until we locate the appropriate row that minimizes the quantity $b_i/a_{i\ell}$. If this is the k^{th} row, then $a_{k\ell}$ is the pivotal element. We then solve the k^{th} equation for the variable x_ℓ, and this expression is substituted into the remaining equations. That is

$$b_k = a_{k1}x_1 + \ldots + a_{k\ell}x_\ell + \ldots + a_n x_n + x_{n+k} \tag{2.54}$$

and thus

$$x_\ell = \frac{b_k}{a_{k\ell}} - \frac{1}{a_{k\ell}} x_{n+k} - \frac{1}{a_{k\ell}} \sum_{\substack{j=1 \\ j\neq\ell}}^{n} a_{kj}x_j \tag{2.55}$$

Substituting Eq. (2.55) into the i^{th} equation we obtain

$$b_i = a_{i1}x_1 + \ldots + \frac{a_{i\ell}}{a_{k\ell}} \left[b_k - x_{n+k} - \sum_{\substack{j=1 \\ j\neq\ell}}^{n} a_{kj}x_j\right]$$

$$+ \ldots + a_{in}x_n + a_{i,n+i}x_{n+i} \tag{2.56}$$

This gives for the new i^{th} equation

$$b_i - a_{i\ell}\frac{b_k}{a_{k\ell}} = \sum_{\substack{j=1 \\ j\neq\ell}}^{n} (a_{ij} - \frac{a_{i\ell}}{a_{k\ell}} a_{kj})x_j$$

$$- \frac{a_{i\ell}}{a_{k\ell}} x_{n+k} + a_{i,n+i}x_{n+i} \qquad (2.57)$$

Upon performing these operations, we note that the ℓ^{th} column of the new updated coefficient matrix will be a new basis vector; that is, it will contain all zeroes, except for the k^{th} entry $a_{k\ell}$, which is of course one. Likewise, substituting Eq. (2.55) into the other equations will introduce into them a non-zero term in the, until now, basic variable x_{n+k}. The new coefficient matrix will have the following general appearance:

$$\begin{array}{ccccccccccc}
\vec{p}_0 & \vec{p}_1 & \cdots & \vec{p}_\ell & \cdots & \vec{p}_n & \vec{p}_{n+1} & \cdots & \vec{p}_{n+k} & \cdots & \vec{p}_{n+m} \\
b_1' & a_{11}' & \cdots & 0 & \cdots & a_{1n}' & 1 & \cdots & a_{1,n+k}' & \cdots & 0 \\
b_2' & a_{21}' & & 0 & & a_{2n}' & 0 & & a_{2,n+k}' & \cdots & 0 \\
\vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots \\
b_k' & a_{k1}' & & 1 & & a_{kn}' & 0 & & a_{k,n+k}' & & 0 \\
\vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots \\
b_m' & a_{m1}' & & 0 & & a_{mn}' & 0 & & a_{k,n+k}' & & 1
\end{array} \qquad (2.58)$$

Furthermore, it is particularly important to note from Eq. (2.57) the values of the new b_i's. That the pivotal element $a_{k\ell}$ was chosen so as to minimize the quantity $b_k/a_{k\ell}$ for i ranging from 1 to m is assurance that the new b_i's, i.e.,

$$b_i - a_{i\ell} \frac{b_k}{a_{k\ell}} \quad , \tag{2.59}$$

are all non-negative.

The process described above is known as a simplex transformation. It can also be thought of as a change of basis between adjacent basis sets. Just as with the previous set of basic and non-basic variables, we can set the non-basic variables equal to zero in order to obtain an extreme point solution. The positivity of this solution is assured by the pivotal element criterion.

We now consider how the above technique is employed to find an optimum extreme point solution. That is, one that minimizes our objective function.

As before, we begin by introducing the appropriate m slack variables to our set of constraint equations. We construct the corresponding coefficient matrix as in (2.12), and then augment this matrix with yet one more row of figures - namely, the coefficients from our objective function. This gives as our starting matrix:

$$\begin{array}{cccccccc}
\vec{P}_0 & \vec{P}_1 & \vec{P}_2 & \cdots & \vec{P}_n & \vec{P}_{n+1} & \vec{P}_{n+2} & \cdots & \vec{P}_{n+m} \\
b_1 & a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\
b_2 & a_{21} & & & \vdots & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & & & \vdots & \vdots & \vdots & & \vdots \\
b_m & a_{m1} & \cdots\cdots\cdots & & a_{mn} & 0 & 0 & & 1 \\
f & c_1 & c_2 & \cdots & c_n & 0 & 0 & \cdots & 0
\end{array} \tag{2.60}$$

To find a minimum feasible solution, one begins by seeking the first column, say the ℓ^{th}, that contains both a positive objective

function coefficient, c_ℓ, and a positive $a_{i\ell}$. Then, in the manner described above, a suitable pivotal element is found among the $a_{i\ell}$'s, that corresponding equation is solved for x_ℓ, and the proper change of basis is effected as per the simplex method. This process is repeated until there no longer exists a positive coefficient c_j within the bottom row. The optimum value of the objective function residing in the position f at the completion of the final simplex iteration and the minimum feasible solution itself are promptly obtained by substituting zeroes in for the non-basic variables remaining in the m constraint equations.

It should be pointed out that within the above discussion there were many mathematical subtleties that were well glossed over. Many assertions and generalities were made on the basis of analogy (for example with 2 and 3 dimensional cases), and sometimes just glibly written down, for the sake of brevity and clarity. Also, there exist certain pathological situations that can occur during the iterative process, e.g. cycles, and degeneracy (Barsov, 1959), that require special attention that weren't discussed. It is hoped that any individual interested in the mathematical rigor accompanying linear programming will find the works listed in the references to their liking.

Having obtained at least a conceptual understanding of the simplex method and how it works, we shall now discuss how it can be used to obtain a minimum absolute error solution to the classical imaging equation (Barrodale and Roberts, 1973). (Recall Eq. (1.1).) First, non-negative variables u_i and v_i for $i \leq m$ are introduced such that

$$g_i - \sum_{j=1}^{m} H_{ij} f'_j = u_i - v_i \tag{2.61}$$

Though neither u_i or v_i can be negative, their difference of course can. Next, we introduce yet another set of non-negative variables, a_j and b_j for $j \leq n$ such that $f'_j = a_j - b_j$. This last step is made in order to accommodate "object" values that are negative. This provides the algorithm with greater versatility, and allows it to be applied to a larger class of data restoration problems. For the purposes of image restoration, however, there is typically no physical relevance in allowing negative object values. As such, some greater computational efficiency within the algorithm could be realized by foregoing this last step, and instead just effect the ensuing simplex computations upon the variables u_i, v_i and f'_j. In addition to somewhat enhanced computational efficiency, such a process might also result in more accurate restorations due to invoking the positivity constraints placed on the values f'_j. This would be subject matter for further study.

Having made such substitutions, we now rewrite the MAE restoration problem, originally stated in Eq. (1.5), as follows:

$$\text{minimize} \quad e = \sum_{i=1}^{m} u_i - v_i \tag{2.62}$$

subject to the following m constraints,

$$g_i = \sum_{j=1}^{n} (a_j - b_j) H_{ij} + u_i - v_i \tag{2.63}$$

Thus, we have succeeded in converting the rather cumbersome problem of Eq. (1.5) into a form that lends itself to analysis via the methods of linear programming described earlier.

There still exists one additional alteration to be made that will ease the computational procedure. Barrodale and Roberts (1970) have shown that the problem stated above is equivalent to solving the following problem with a slightly altered objective function:

$$\text{minimize} \quad e = \sum_{i=1}^{m} u_i + v_i \tag{2.64}$$

subject to the (same) following m constraints,

$$g_i = \sum_{j=1}^{n} (a_j - b_j) H_{ij} + u_i - v_i \tag{2.65}$$

This latter statement of the problem turns out to be less unwieldly to solve that the previous one, allowing certain shortcuts to be taken within the ensuing simplex procedure (Barrodale and Roberts, 1973).

Median Filter

Median filtering is a non-linear method of preparing the image data that is well suited for eliminating outliers (Pratt, 1978). The method is rather straightforward: a window which samples an odd number of data points, NWIND, at a time is moved along the stream of data one point at a time. At any one time, a designated array will contain these NWIND sampled points. From this array the median value of the pixel values stored is determined, and this value is chosen as the output

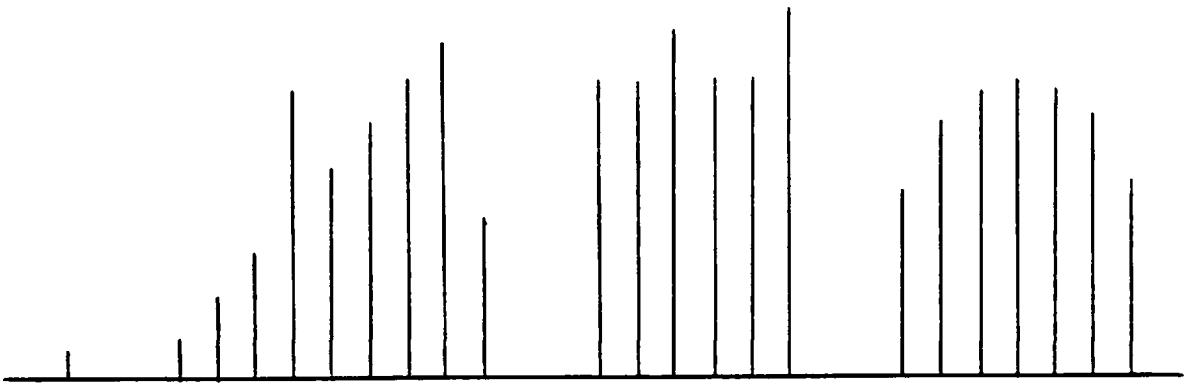
pixel value. Thus, any image point that is severely afflicted with outlier noise will be much higher, or lower, than any other points contained within a particular array of sampled points.

Figure 3 shows how a given set of image data, afflicted with sporadic outliers, is altered upon median filtering. The sampling window used was of length NWIND=3. Several effects are observed that distinguish median filtering from a linear process such as convolving the data with, say, a rectangle function, where it may be hoped that the outliers will average out among the neighboring points. This latter operation will clearly have a smoothing effect upon the image data. The (undesired) information represented by the outlier is still present, just smoothed and made somewhat less conspicuous.

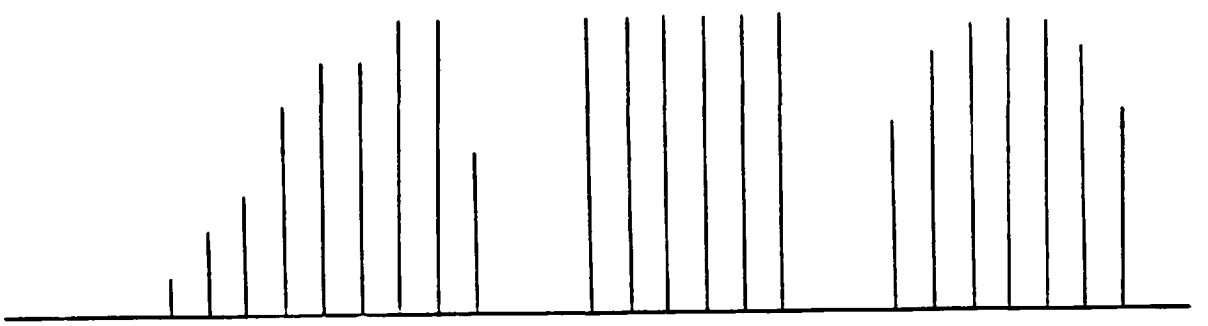
Median filtering also effects some degree of smoothing. Portions of the image data that contain local maxima (minima), like the top of the ramp and the sinusoidal-like hump, will be slightly flattened at the top (bottom). Also, any highly localized or impulse type structure, as on the far left of Figure 3a, will be altogether deleted if its spatial extent is of less than (NWIND-1)/2.

It is this latter effect that makes median filtering especially desirable and effective in picking out and deleting outliers. Remember, the image data represents the object distribution after having been smoothed by the appropriate point spread function. This means that there will be some characteristic limit to the amount of high spatial frequency content within the image data. Any such high frequency structure that is present can be attributed to noise. Outliers in particular

represent such structure and obviously must represent a noise extremum or sampling error at that point.



(a)



(b)

Figure 3. Effects of median filtering.

(a) "Image" data with outliers.
(b) Data from part a above after median filtering. NWIND=3.

CHAPTER 3

EXPERIMENTAL PROCEDURE

An appropriate experimental procedure must be developed that will provide a fair and consistent comparison of the different restoration methods employed. We are of course interested in comparing how well the different restoration methods can cope with varying degrees of gaussian noise versus varying degrees of outlier contamination, as well as combinations of the two. Thus we want to be able to control the amount of each type of noise that is added to the image data.

Other considerations exist as well. We'd like to observe how well restorations behave for varying degrees of blurring. Thus, we want to be able to vary the size and shape of the point spread function.

An added consideration exists in deciding on the type of object distribution it is we wish to restore. For example, it is well known that certain restoration techniques are better suited to restoring rapidly varying impulse type objects than smoother, more slowly varying object scenes. To this end we wish to design an object distribution that contains a good portion of both high and low frequency content.

The following procedure was adopted to meet the above requirements. A program OBJGEN was written to create an initial image file. The user supplies N, the number of object points; NPSF, the number of points in the point spread function; and finally the values of the point

spread function. OBJGEN first produces a set of N object points that fit the object distribution pictured in Figure 4. Such an object distribution seems to be a good combination of high and low frequency components. N was consistently set equal to 124. This decision was made largely on practical considerations - the larger N is, the longer one has to wait for their results, and the more expensive it is. If n is small, severe limitations are encountered when the user tries to vary the size of the psf. For example, if N is 50, then a 5 point psf represents a blur function that is 10% of the total field.

After calculating the object vector, OBJGEN assembles the mxn blurring matrix H, using the user supplied psf values. (Before doing this, OBJGEN normalizes the psf values such that the area under the psf equals one.) This matrix is then multiplied by the object vector, giving the noise - free image data. The object vector, the blur matrix H, and the noise free image are then written into a file. This file is typically retained in the user's directory so that subsequently it can be read by the program NOISE, which adds varying amounts of noise to the image vector.

The program NOISE gives the user the opportunity to add Gaussian noise and outlier noise to the image data. The values produced by the random number generator of the computer follow a flat probability distribution, located between zero and one. In order to obtain random numbers adhering to a Gaussian probability distribution, the following transformation is performed (Frieden, 1980):

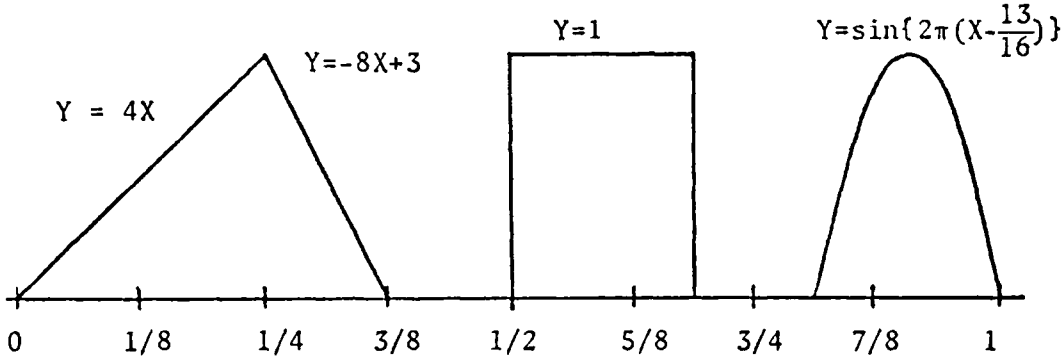


Figure 4. Diagram of object distribution, f(X), used in OBJGEN.

Expressed mathematically:

$$f(X) = \begin{cases} 4X \text{ for } 0 \leq X < .25 \\ -8X + 3 \text{ for } .25 \leq X < .375 \\ 0 \text{ for } .375 \leq X < .5 \\ 1 \text{ for } .5 \leq X < .6875 \\ 0 \text{ for } .6875 \leq X < .8125 \\ \sin\left\{2\pi\left(X - \frac{13}{16}\right)\right\} \text{ for } .8125 \leq X \leq 1.0 \end{cases}$$

$$X = \langle m \rangle + \sigma(-2\ln Y_1)^{\frac{1}{2}}\cos 2\pi Y_2 \qquad (3.1)$$

The quantities Y_1 and Y_2 represent the "random" numbers supplied by the computer's system subroutine, and are characterized by the above mentioned rectangular probability distribution function. The quantity X will have the desired Gaussian probability distribution of mean "m," and of standard deviation σ. Both of these latter quantities are user supplied. "m" was always zero and σ, known in NOISE as SIGMA, was varied.

After adding the appropriate Gaussian noise to the image vector, NOISE asks the user whether outlier noise is to be added to the image vector. If so, the user supplies the indices and the corresponding outlier noise magnitude for the image points that are to be perturbed.

NOISE also calculates the noise standard deviation and SNR of the data, and appends them at the end of the image file. These two quantities are determined just after the Gaussian noise has been added, before considering the matter of outliers. In other words, the outlier fluctuations don't enter into the determination of SNR of noise variance since these two quantities represent a priori knowledge regarding the noise characteristics of whatever signal acquisition process may have been involved. In contrast, outliers represent sampling or transmission errors, about which no information is known. As such it would be unrealistic to include their effect in the assessment of the noise characteristics.

The following three programs were written to effect the restoration methods discussed earlier.

MINABS: Performs the minimum absolute error restoration as per the simplex method of Barrodale and Roberts (Barrodale and Roberts, 1974).

INVMAT: Achieves a least square solution via the pseudo-inverse method.

NOCORR: Performs the MMSE restoration assuming an uncorrelated object ensemble.

The matrix inversion required in both INVMAT and NOCORR was achieved by a subroutine which employed Jordan's method of successive elementary transformations. (See, for example, Hoffman and Kunze, 1961.)

Each restoration method was applied to image data degraded by various amounts of both Gaussian and outlier noise. The amount of Gaussian noise added was varied so as to achieve SNR's ranging from roughly 740 to roughly 78. For the sake of fairness and consistency the degree of outlier noise was handled in the following manner. In each trial, the same 3 image points - the eighth, sixteenth and twentyfourth - had added to them some amount of outlier noise. For different trials, the magnitude of the outlier noise was increased or decreased in order to test each method's ability to cope with it.

The restoration quality of each method was judged by calculating the root mean square discrepancy, d_{rms}, between the object and its corresponding estimate:

$$d_{rms} = \left\{ \sum_{i=1}^{n} (f_i - f_i')^2/n \right\}^{\frac{1}{2}} . \qquad (3.2)$$

The values of d_{rms} for each restoration are tabulated as a function of the amount of Gaussian noise (characterized by the SNR) and outlier noise (characterized by the outlier noise magnitude, ONM) added to the image data points 8, 16 and 24.

One additional program, STARGEN, was written in order to investigate whether or not any of the restoration methods employed showed a marked ability in restoring object distributions that consist mainly of impulse type structures. We are alluding, of course, to astronomical applications, where the image is of a group of stars located in some portion of the sky. A typical problem that arises with such images is to determine whether a particular "spot" or "blur" is due to a single star or perhaps an unresolved pair of stars.

STARGEN is an alteration of OBJGEN. The object distribution created is a sequence of pairs of impulses ("stars") separated by varying distances (see Figure 5). This object is then blurred by a user supplied point spread function in a manner identical to that of OBJGEN. The data file created by STARGEN is identical in form to that of OBJGEN in order that it may be read by NOISE, wherein the user can impair the image vector with the desired amount of noise.

The experiments described above were performed on a VAX 11/780 minicomputer, manufactured by the Digital Equipment Corporation. The computer was outfitted with a VMS operating system.

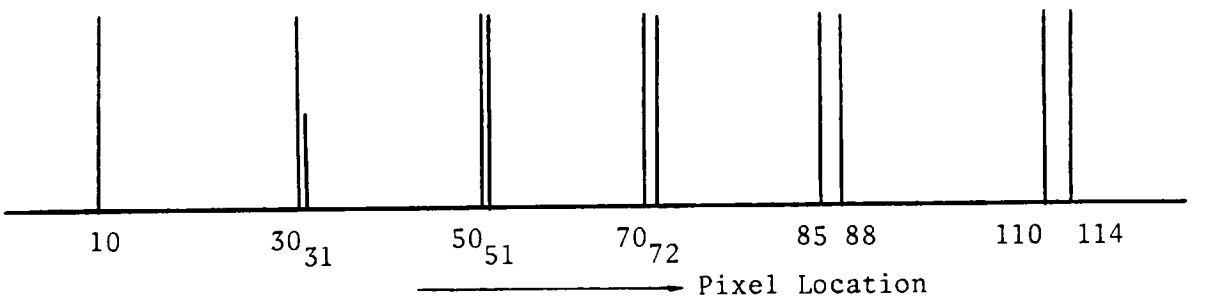


Figure 5. Object distribution used in STARGEN.

CHAPTER 4

EXPERIMENTAL RESULTS

On the following pages are tabulated the root mean square discrepancies, d_{rms}, between the original object distribution and the object estimate as achieved by each of the three restoration methods employed. NOCORR indicates the result obtained via the MMSE criterion with an uncorrelated object ensemble. PI refers to the pseudo-inverse, or least-squares method of INVMAT, and MAE stands for the minimum absolute error solution obtained by MINABS.

Each table shows the results for a specific point spread function used in blurring the object, and for different combinations of Gaussian noise and outlier noise magnitude, ONM. The standard deviation of the Gaussian noise added, σ_n, was varied by altering the user defined variable SIGMA. The outlier noise magnitude ONM corresponds to the amount of error added to the 8^{th}, 16^{th}, and 24^{th} points of the image vector.

The point spread function used to blur the object distribution is pictured in the upper right of each table. Consistent with the nomenclature employed in earlier discussions, the quantities N, NPSF and M are:

N = Number of Object Points

NPSF = Number of Points ("width") of the PSF

M = Number of Image Points, where M=N+NPSF-1

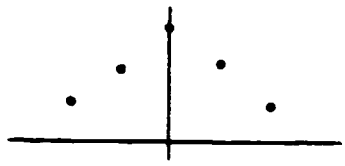
The results depicted in Tables 7 and 8 illustrate the effects that median filtering the image data before submitting it to a restoration algorithm can have on the restoration accuracy. Median filtering was performed on image data with and without outlier noise contamination in order to observe its effects on the restoration process.

Following the numerical results presented in Tables 1 through 8 are Figures 6 through 12. These were obtained from photographs of CRT displays of the various object, image, and restoration data for different combinations of noise and blurring.

Table 1. RMS discrepancies.

PSF = $\mathrm{tri}\left(\frac{X}{5/2}\right)$

N = 124 NPSF = 5 M = 128

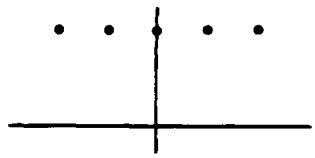


	ONM=0	ONM=.1	ONM=.5
σ_n=0		NOCORR 2.89 Pseud In 2.89 MAE 2.46	NOCORR 14.42 PI 14.42 MAE 12.32
σ_n=7.9x10^{-4} SNR=772.8 SIGMA=.0005	NOCORR .063 PI .021 MAE .370	NOCORR 1.131 PI 2.860 MAE 2.449	NOCORR 5.637 PI 14.384 MAE 12.280
σ_n=1.6x10^{-3} SNR=386.4 SIGMA=.001	NOCORR .082 PI .443 MAE .738	NOCORR .731 PI 2.830 MAE 2.494	NOCORR 3.567 PI 14.327 MAE 12.259
σ_n=7.9x10^{-5} SNR=77.3 SIGMA=.005	NOCORR .136 PI 2.22 MAE 4.21	NOCORR .310 PI 3.298 MAE 4.773	NOCORR 1.298 PI 14.336 MAE 12.311
σ_n=1.6x10^{-2} SNR=38.6 SIGMA=.01	NOCORR .169 PI 4.41 MAE 7.38	NOCORR .242 PI 4.769 MAE 7.563	NOCORR .810 PI 14.206 MAE 13.706
σ_n=7.9x10^{-2} SNR=7.7 SIGMA=.05		NOCORR .207	
σ_n=1.6x10^{-1} SNR=3.9 SIGMA=.1		NOCORR .205	
σ_n=1.58 SNR=.4 SIGMA=1.		NOCORR .566	

Table 2. RMS discrepancies.

PSF = rect $\left(\frac{X}{5}\right)$

N = 124 NPSF = 5 M = 128'

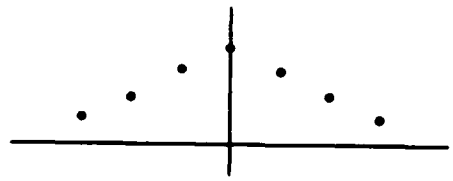


		ONM=0		ONM=.1		ONM=.5
σ_n=0			NOCORR	.230	NOCORR	1.154
			PI	.230	PI	1.154
			MAE	.118	MAE	.674
σ_n=7.9x10^{-4}	NOCORR	.014	NOCORR	.232	NOCORR	1.149
SNR=768.8	PI	.014	PI	.234	PI	1.157
SIGMA=.0005	MAE	.014	MAE	.254	MAE	1.344
σ_n=1.6x10^{-3}	NOCORR	.0283	NOCORR	.231	NOCORR	1.128
SNR=384.4	PI	.0283	PI	.237	PI	1.209
SIGMA=.001	MAE	.0283	MAE	.141	MAE	1.327
σ_n=7.9x10^{-3}	NOCORR	.0984	NOCORR	.185	NOCORR	.780
SNR=76.9	PI	.1414	PI	.292	PI	1.258
SIGMA=.005	MAE	.1597	MAE	.589	MAE	1.309
σ_n=1.6x10^{-2}	NOCORR	.139	NOCORR	.1785	NOCORR	.567
SNR=38.4	PI	.283	PI	.3969	PI	1.538
SIGMA=.01	MAE	.347	MAE	.2835	MAE	1.240

Table 3. RMS discrepancies.

PSF = tri $\left(\frac{X}{7/2}\right)$

N = 124 NPSF = 7 M = 130

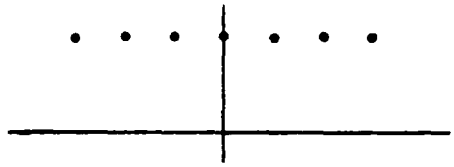


	ONM=0		ONM=.1		ONM=.5	
$\sigma_n=0$			NOCORR	7.78	NOCORR	***
			PI	7.78	PI	***
			MAE	.482	MAE	***
$\sigma_n=7.9\text{x}10^{-4}$	NOCORR	.089	NOCORR	1.39		
SNR=761.1	PI	.176	PI	7.76		
SIGMA=.0005	MAE	.455	MAE	4.54		
$\sigma_n=1.6\text{x}10^{-3}$	NOCORR	.102	NOCORR	.707		
SNR=380.5	PI	.338	PI	7.75		
SIGMA=.001	MAE	.987	MAE	2.16		

Table 4. RMS discrepancies.

PSF = rect $\left(\frac{X}{7}\right)$

N = 124 NPSF = 7 M = 130

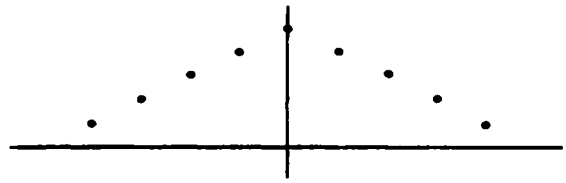


		ONM=0	ONM=.1		ONM=.5	
$\sigma_n=0$			NOCORR	.205	NOCORR	1.026
			PI	.205	PI	1.026
			MAE	.407	MAE	1.833
$\sigma_n=7.9\times10^{-4}$	NOCORR	.014	NOCORR	.205	NOCORR	1.022
SNR=754	PI	.014	PI	.206	PI	1.026
SIGMA=.0005	MAE	.020	MAE	.370	MAE	2.123
$\sigma_n=1.6\times10^{-3}$	NOCORR	.028	NOCORR	.204	NOCORR	1.011
SNR=377.0	PI	.028	PI	.207	PI	1.027
SIGMA=.001	MAE	.045	MAE	.366	MAE	2.161
$\sigma_n=7.9\times10^{-3}$	NOCORR	.112	NOCORR	.194	NOCORR	.823
SNR=75.4	PI	.142	PI	.251	PI	1.038
SIGMA=.005	MAE	.214	MAE	.445	MAE	2.379

Table 5. RMS discrepancies.

PSF = $\mathrm{tri}\left(\frac{X}{9/2}\right)$

N = 124 NPSF = 9 M = 132

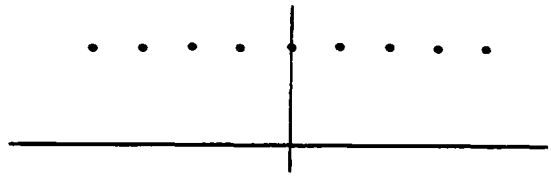


	ONM=0		ONM=.1		ONM=.5	
σ_n=0			NOCORR	6.355	NOCORR	***
			PI	6.355	PI	***
			MAE	5.818	MAE	***
$\sigma_n=7.9\times10^{-3}$	NOCORR	.099	NOCORR	1.14	NOCORR	5.669
SNR=747.5	PI	.318	PI	6.333	PI	***
SIGMA=.0005	MAE	.543	MAE	5.354	MAE	***
$\sigma_n=1.6\times10^{-3}$	NOCORR	.118	NOCORR	.964	NOCORR	4.701
SNR=373.8	PI	.608	PI	6.325		
SIGMA=.001	MAE	1.420	MAE	6.003		

Table 6. RMS discrepancies.

PSF = rect $\left(\frac{X}{9}\right)$

N = 124 NPSF = 9 M = 132



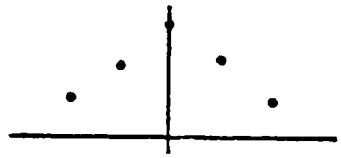
	ONM=0		ONM=.1		ONM=.5	
σ_n=0			NOCORR	.244	NOCORR	1.217
			PI	.244	PI	1.217
			MAE	.412	MAE	1.512
σ_n=7.9x10^{-3}	NOCORR	.0164	NOCORR	.580	NOCORR	1.216
SNR=736.8	PI	.010	PI	.246	PI	1.220
SIGMA=.0005	MAE	.0245	MAE	.405	MAE	1.967
σ_n=1.6x10^{-3}	NOCORR	.032	NOCORR	.245	NOCORR	1.204
SNR=368.4	PI	.024	PI	.248	PI	1.222
SIGMA=.001	MAE	.050	MAE	.379	MAE	1.954
σ_n=7.9x10^{-2}	NOCORR	.110				
SNR=73.7	PI	.124				
SIGMA=.005	MAE	.249				

Table 7. RMS discrepancies.

Restorations preceded by median filtering of image data. NWINDOW=3.

PSF = $\mathrm{tri}\left(\frac{X}{5/2}\right)$

N = 124 NPSF = 5 M = 128



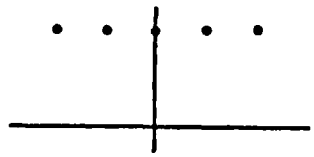
	ONM=0		ONM=.1		ONM=.5	
σ_n=0	NOCORR	.057	NOCORR	.918	NOCORR	.918
	PI	.057	PI	.918	PI	.918
	MAE	.045	MAE	.772	MAE	.772
$\sigma_n=7.9\times10^{-3}$	NOCORR	.050	NOCORR	.301	NOCORR	.301
SNR=772.8	PI	.235	PI	.733	PI	.733
SIGMA=.0005	MAE	.453	MAE	.573	MAE	.573
$\sigma_n=1.6\times10^{-3}$	NOCORR	.057	NOCORR	.187	NOCORR	.187
SNR=386.4	PI	.480	PI	.572	PI	.572
SIGMA=.001	MAE	.922	MAE	.690	MAE	.690

Table 8. RMS discrepancies.

Restorations preceded by median filtering of image data. NWINDOW=3.

PSF = rect $\left(\frac{X}{5}\right)$

N = 124 NPSF = 5 M = 128



	ONM=0		ONM=.1		ONM=.5	
$\sigma_n^2=0$	NOCORR	.009	NOCORR	.060	NOCORR	.060
	PI	.009	PI	.060	PI	.060
	MAE	.010	MAE	.092	MAE	.092
$\sigma_n^2=7.9\text{x}10^{-3}$	NOCORR	.017	NOCORR	.055	NOCORR	.055
SNR=768.8	PI	.017	PI	.055	PI	.055
SIGMA=.0005	MAE	.017	MAE	.032	MAE	.032
$\sigma_n^2=1.6\text{x}10^{-3}$	NOCORR	.028	NOCORR	.053	NOCORR	.053
SNR=384	PI	.029	PI	.054	PI	.054
SIGMA=.001	MAE	.041	MAE	.047	MAE	.047
$\sigma_n^2=7.9\text{x}10^{-2}$	NOCORR	.114	NOCORR	.112	NOCORR	.112
SNR=38.4	PI	.205	PI	.177	PI	.177
SIGMA=.01	MAE	.388	MAE	.299	MAE	.299

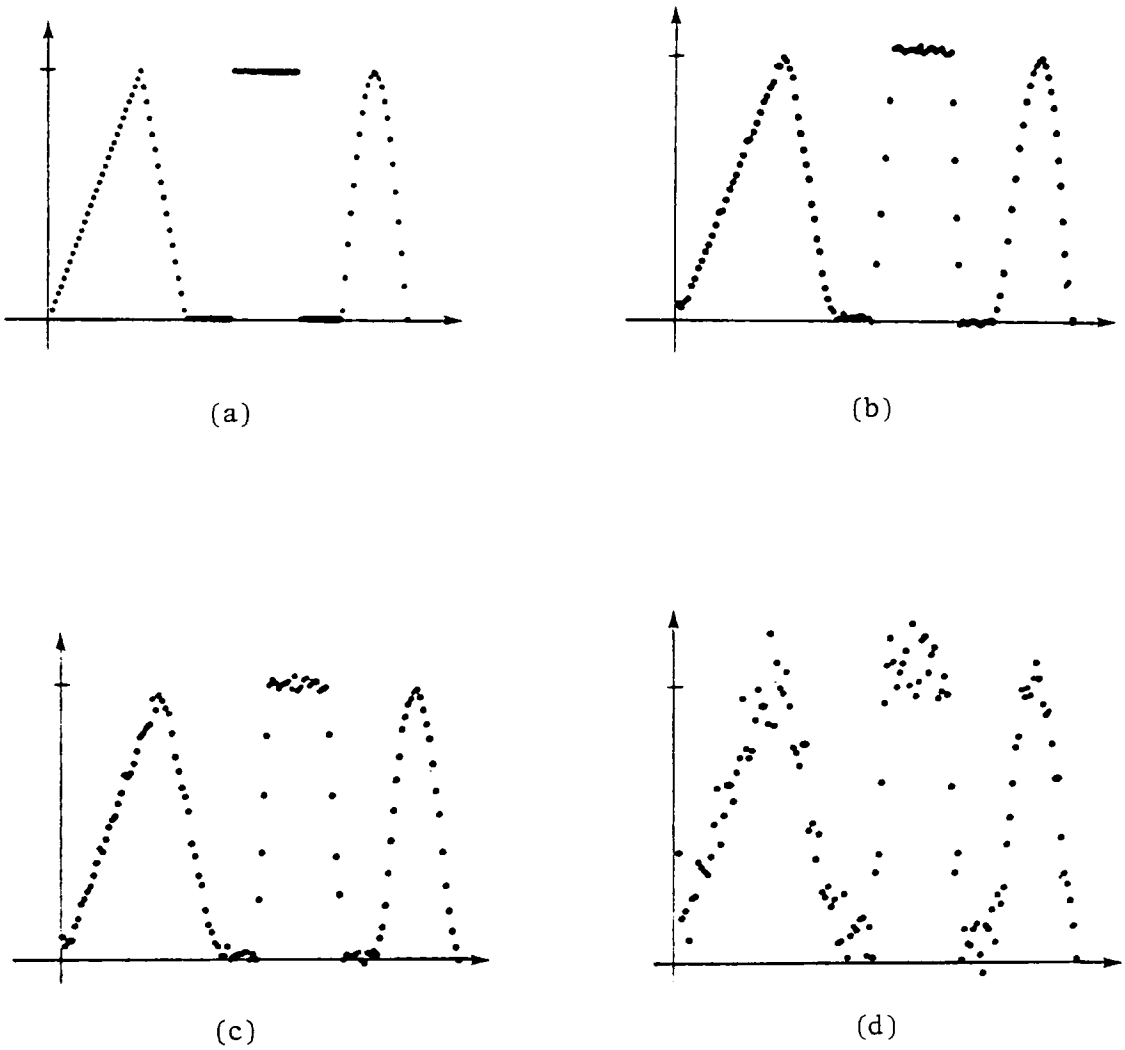


Figure 6. Original object and images with varying degrees of Gaussian noise.

PSF in all cases is rect $\left(\frac{x}{5}\right)$. No outliers.

(a) Original object.
(b) Image, SIGMA=.005, SNR=76.9.
(c) Image, SIGMA=.01, SNR=38.4.
(d) Image, SIGMA=.05, SNR=7.7.

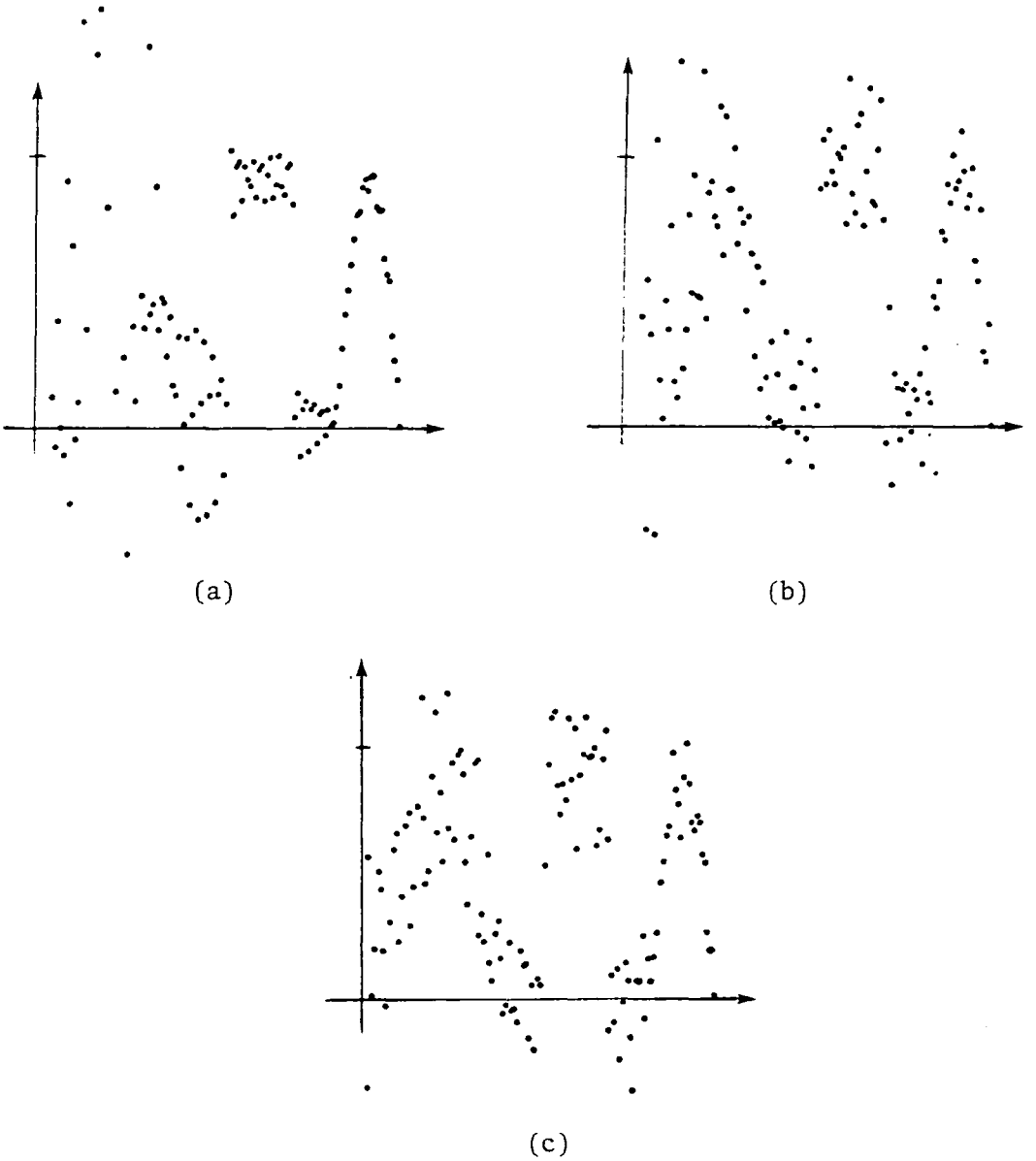


Figure 7. NOCORR restorations for different amounts of Gaussian Noise.

In all above cases, psf = tri $\left(\frac{x}{5/2}\right)$, and ONM=.1.

(a) SIGMA=.001, SNR=386.4, d_{rms}=.731.
(b) SIGMA=.01, SNR=38.6, d_{rms}=.731.
(c) SIGMA=.1, SNR=3.9, d_{rms}=.205.

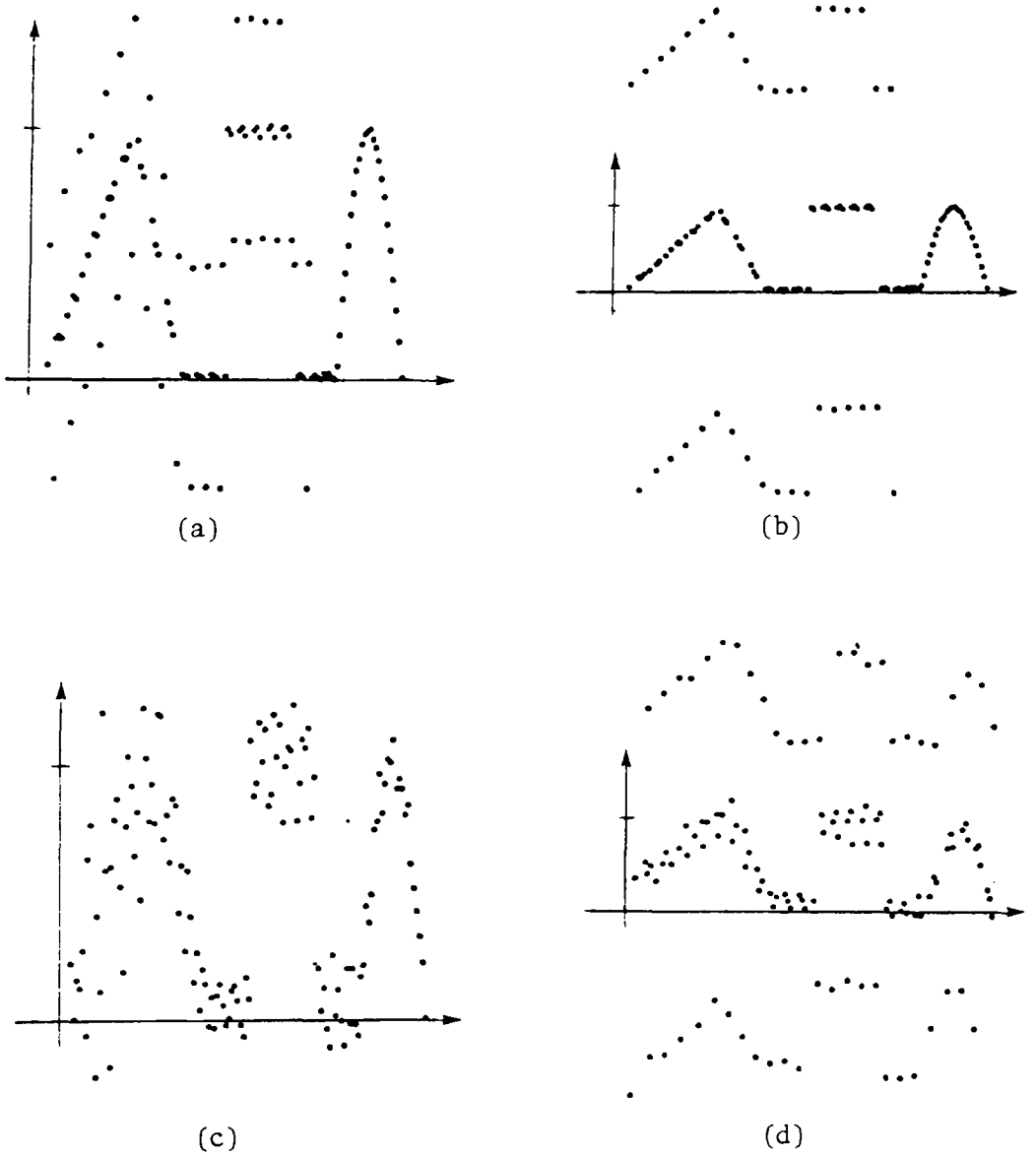


Figure 8. MAE restorations for varying amounts of Gaussian and outlier noise.

PSF = rect $\left(\frac{x}{5}\right)$.

(a) SIGMA=.0005, SNR=768.8, ONM=.1, d_{rms}=.254.
(b) SIGMA=.0005, SNR=768.8, ONM=.5, d_{rms}=1.344.
(c) SIGMA=.005, SNR=76.9, ONM=.1, d_{rms}=.589.
(d) SIGMA=.005, SNR=76.9, ONM=.5, d_{rms}=1.309.

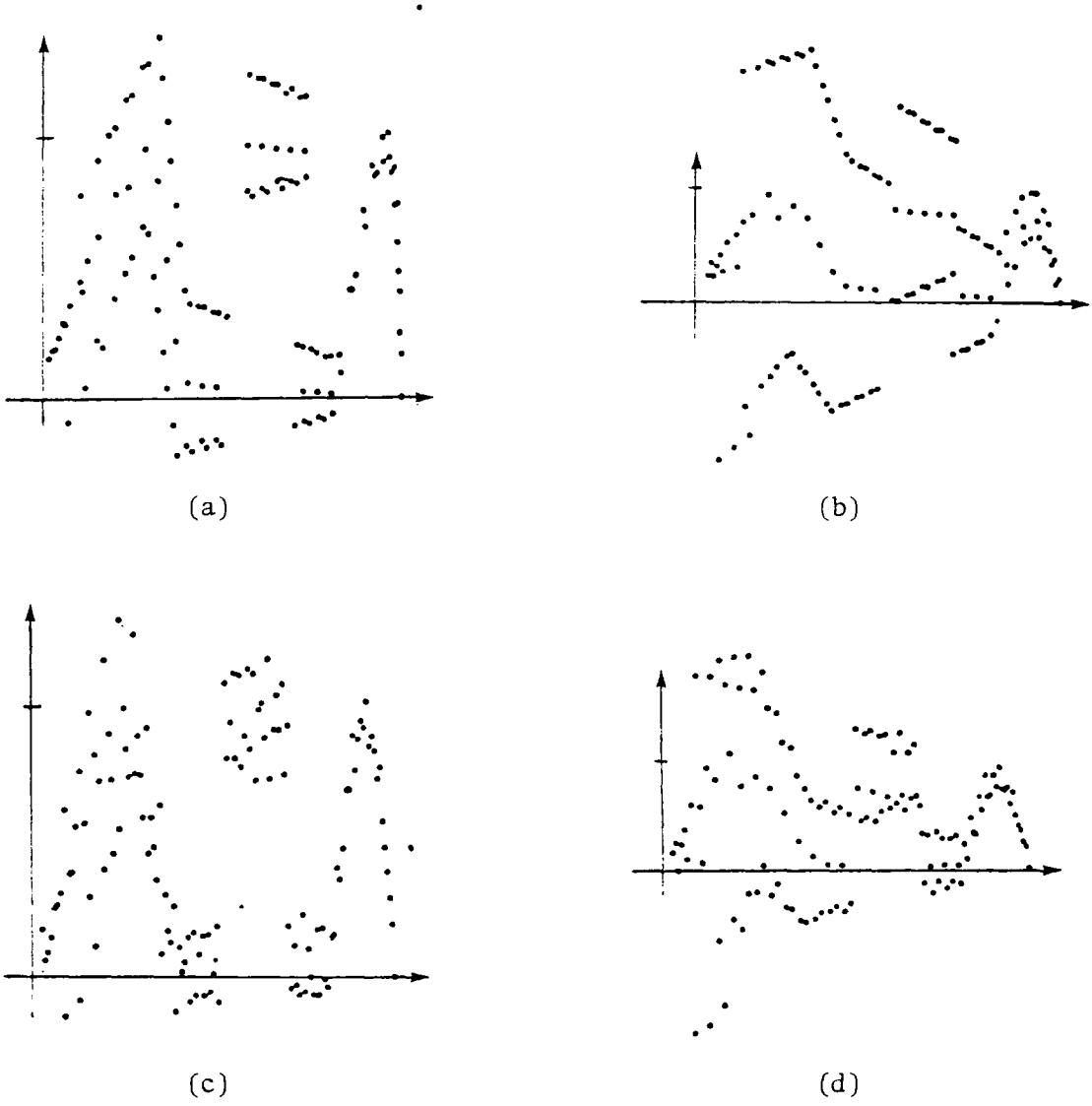


Figure 9. NOCORR restorations for varying amounts of Gaussian and outlier noise.

PSF = rect $\left(\frac{x}{5}\right)$.

(a) SIGMA=.0005, SNR=768.8, ONM=.1, d_{rms}=.232.
(b) SIGMA=.0005, SNR=768.8, ONM=.5, d_{rms}=1.149.
(c) SIGMA=.005, SNR=76.9, ONM=.1, d_{rms}=.185.
(d) SIGMA=.005, SNR=76.9, ONM=.5, d_{rms}=.780.

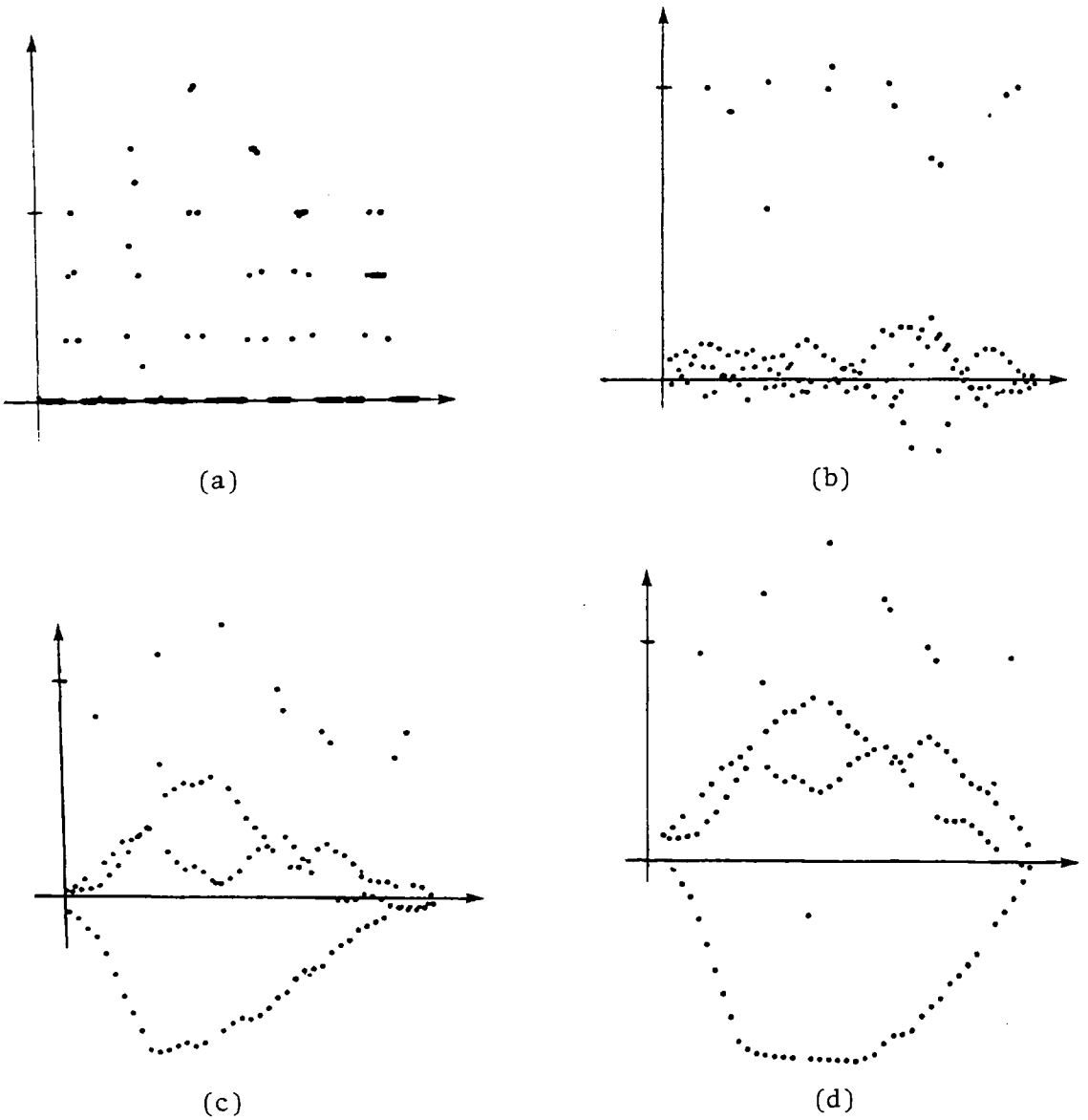
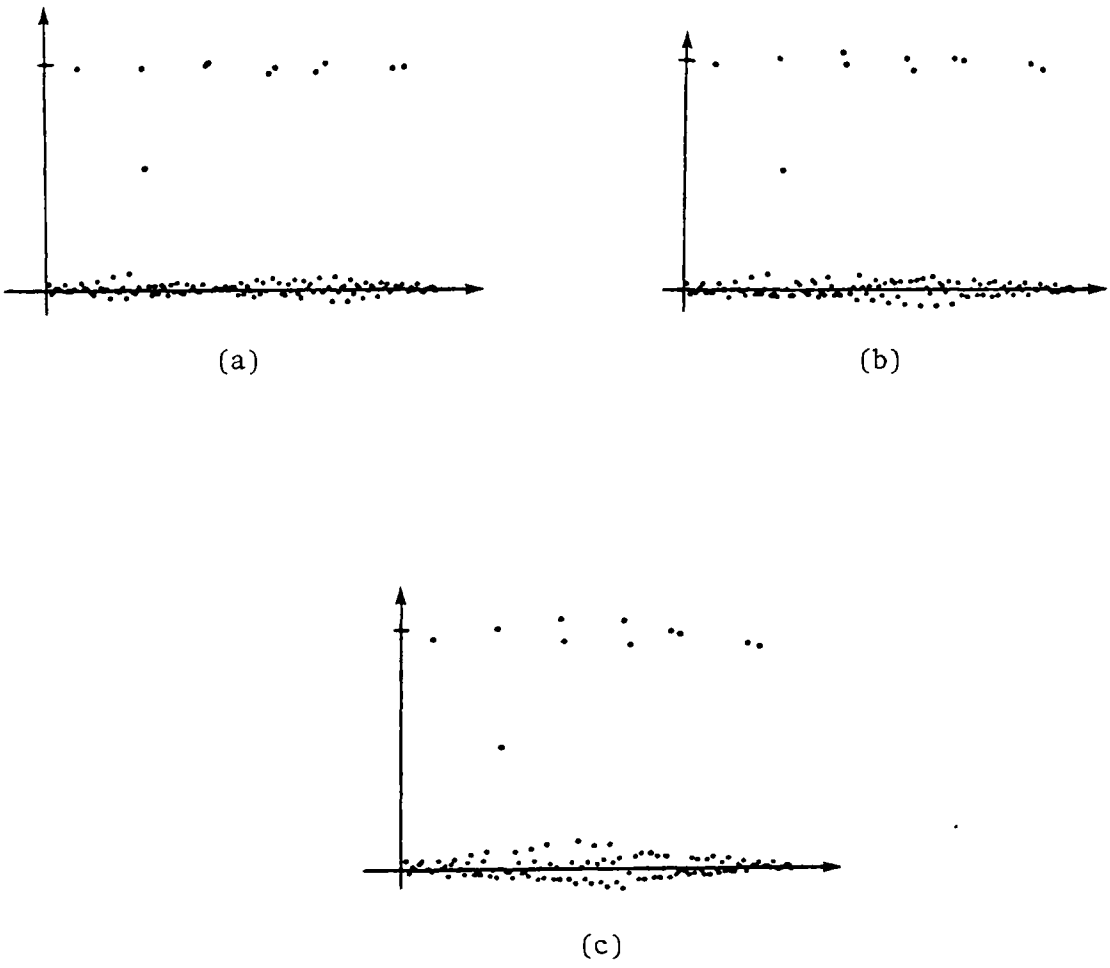


Figure 10. Restorations of STARGEN image blurred by tri $\left(\frac{x}{5/2}\right)$.

In all cases SIGMA=.001, SNR=101.8, and ONM=0.0.

(a) Image from STARGEN.
(b) NOCORR restoration. d_{rms}=.0822.
(c) INVMAT restoration. d_{rms}=.4416.
(d) MINAMS restoration. d_{rms}=.738.



(a) (b)

(c)

Figure 11. Restorations of STARGEN image blurred by rect $\left(\frac{x}{5}\right)$.

In all cases SIGMA=.001, SNR=96.3, and ONM=0.0.

(a) NOCORR restoration. d_{rms}=.0230.
(b) INVMAT restoration. d_{rms}=.0283.
(c) MINABS restoration. d_{rms}=.0412.

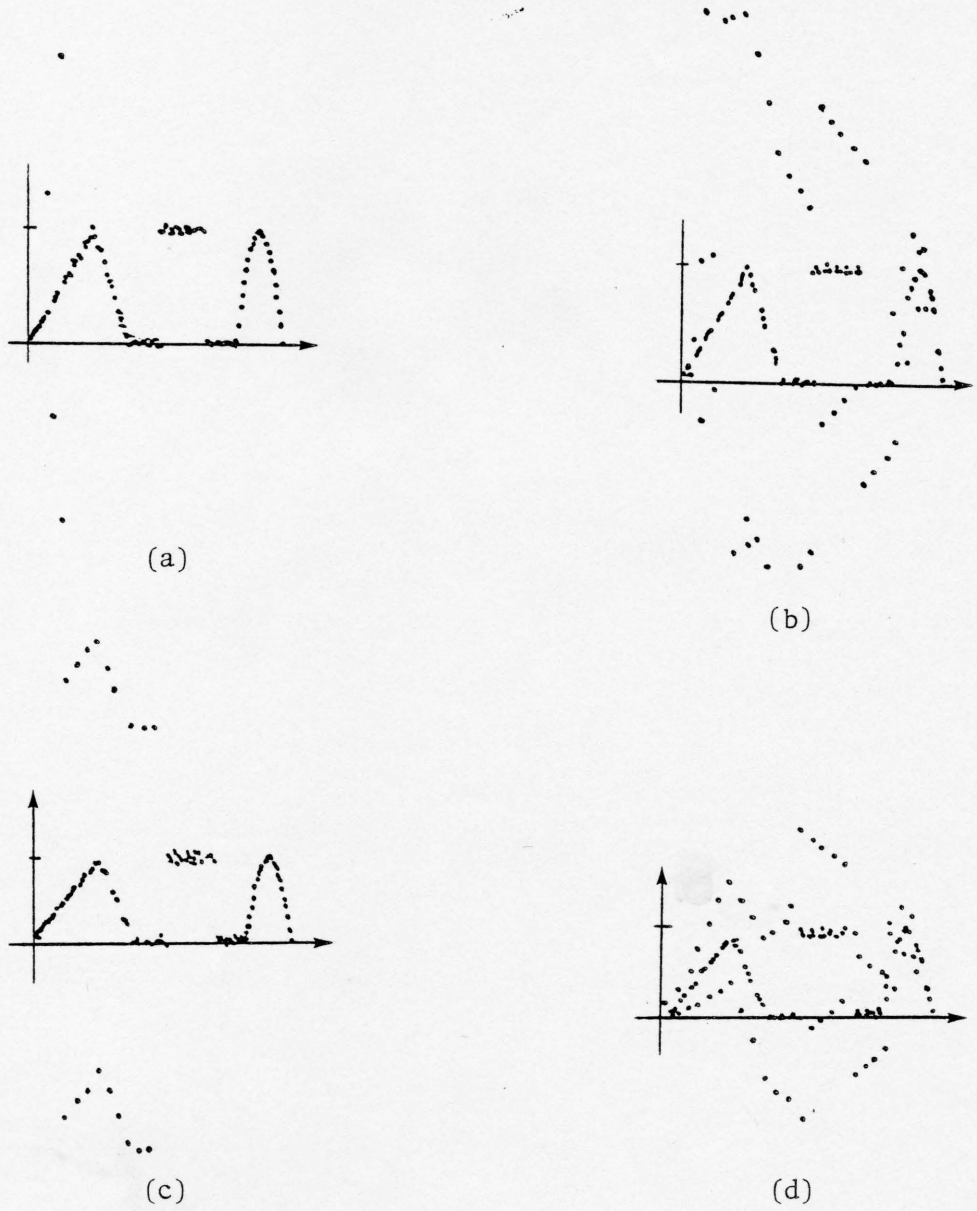


Figure 12. MAE and MMSE restorations for different amounts of outlier contamination.

In all cases, psf = rect $\left(\frac{x}{5}\right)$.

(a) MINABS. SIGMA=.0005. Outliers: .2@10, .4@20. $d_{\quad}$=.290.
(b) NOCORR. SIGMA=.0005. Outliers: .2@10, .4@20. d_{rms}=.877.
(c) MINABS. SIGMA=.001. Outliers: .5@60. d_{rms}=.981.
(c) NOCORR. SIGMA=.001. Outliers: .5@60. d_{rms}=.460.

CHAPTER 5

DISCUSSION AND CONCLUSIONS

The results summarized in Tables 1 through 6 indicate that the minimum absolute error criterion as effected by the method of Barrodale and Roberts (1973) provided results that are quite comparable to the least-squares criterion invoked by the pseudo-inverse method. As was originally hoped for, the MAE method does seem to provide slightly better restoration accuracy under the influence of outlier contamination. Likewise, in the presence of only Gaussian noise, with no outliers, the pseudo-inverse method was consistently better than the MAE approach. Most conspicuous of all, though, is the immense success of NOCORR, which employed the MMSE and signal to noise criteria. NOCORR succeeded in out-performing both MINABS and INVMAT in nearly every restoration problem considered.

Overall, it would have to be judged that INVMAR, or equivalently, the least-squares criterion, was generally more successful than MINABS. In the presence of outliers, and with only a small amount of Gaussian background noise, MINABS occasionally would restore somewhat more accurately than INVMAT. This slight advantage, however, would dissipate as the Gaussian noise was allowed to increase, as though, with sufficient background noise, the outliers became less conspicuous and dominated the error function to a lesser degree.

The results depicted in Figure 12, however, demand that radically altered appraisals of the different restoration schemes be made. When a seemingly modest change in the outlier noise added was made, MINABS unfurled a restoration prowess far superior to either INVMAT or NOCORR. Obviously, the MAE approach is extremely sensitive to the number of outliers present.

The specific manner in which outliers affect the restored object is peculiar (see Figures 8, 9 and 12) in all three restoration methods, but it is most intriguing with MINABS. The so-called ringing effect that is prominent in many of the restored objects is well known and has been long observed in the field of image restoration (Frieden, 1977). It is the symmetrical nature of the ringing, or erroneous points of the MAE restorations that is interesting (see Figure 8). The MAE method also displays a significant localization sensitivity in its restorations. That is, the bad or inaccurate points within the MAE object estimate seem to generally be within the same locale as the outlier contaminated data points of the image vector. This is quite apparent in the MAE restorations of Figure 12. Here, image points 10 and 20 were contaminated with outliers of .2 and .4, respectively. Correspondingly, points within this same vicinity of the object estimate are what seem to suffer most. In contrast, the detrimental effects that outliers have on pseudo-inverse or MMSE restorations extend more throughout the entire object estimate.

Despite the appealing nature of the above observations, the fate of the usefulness of MAE restorations pales somewhat upon considering the effectiveness of simply median filtering the image data before

submitting it to restoration. The MAE method is most effective when the background Gaussian noise is quite low. In such instances, outliers are readily picked out and done away with when subjected to median filtering. Then, an MMSE, or even a pseudo-inverse (for suitably high SNR) restoration will provide restorations superior to that of the MAE method. The assertions are born out in Tables 7 and 8.

For the most part, the MMSE criterion employed by NOCORR was the superior restoration technique. As described earlier in Chapter 2, it was the only one of the three methods used to invoke any kind of constraint during the restoration process that regarded some knowledge of what the object estimate should look like. We see first hand how effective it can be to utilize even such seemingly sparse information as a SNR in directing the restoration.

It is well known that the MMSE criterion in effect imposes a smoothness constraint upon the restoration. This is best understood in the Fourier domain, wherein the MMSE criterion leads to the Weiner function (Helstrom, 1967).

$$W(f_x) = \frac{H^*(f_x)}{|H(f_x)|^2 + \phi_n(f_x)/\phi_o(f_x)} \tag{5.1}$$

For uncorrelated noise and object ensemble, we have

$$W(f_x) = \frac{H^*(f_x)}{|H(f_x)|^2 + (1/SNR)^2} \tag{5.2}$$

From this latter expression we observe that the lower the SNR, the greater the attenuation of high frequency components of the image data.

Equivalently, this suppresses the acquisition of wildly varying points in the object estimate. In constrast, neither the MAE or least-squares method impose such constraints. As such, the restorations are subject to such oscillations. Perhaps the best display of the MMSE qualities is in the restorations of the "star" patterns depicted in Figures 10 and 11.

An apparent anomaly seems to exist with regard to the results of NOCORR restorations as given in the tables. In the cases where the image is contaminated with both Gaussian and outlier noise, it seems to occur with rather perplexing consistency that as the level of Gaussian noise increases, the NOCORR restoration accuracy improves. Weird, huh?

An explanation of this phenomena could lie in the manner in which the SNR and noise standard deviation, σ_n, are specified. Early on, an ethic was adopted wherein statements of SNR's and σ_n's would not reflect the effect of any outliers present, since their existence is not known of. Let us now violate this ethic and see how our NOCORR restorations behave if we acknowledge these outliers when assessing SNR and σ_n.

By definition

$$\sigma_n = \left[\sum_{i=1}^{M} n_i^2/M \right]^{\frac{1}{2}} \tag{5.3}$$

$$\sigma_f = \left[\sum_{i=1}^{M} \left(\sum_{j=1}^{N} H_{ij} f_j \right)^2 /M \right]^{\frac{1}{2}} \tag{5.4}$$

$$SNR = \sigma_f/\sigma_n \tag{5.5}$$

where n_i is the Gaussian noise contribution to the i^{th} image point. Let us now determine how σ_n and SNR will be altered if we consider the effects of 3 outliers, each of magnitude .1.

$$\sigma_{no} \equiv \text{noise standard deviation, with outlier contribution} \tag{5.6}$$

$$\cong \left[\frac{\sum_{i=1}^{M} n_i^2 + .1^2 + .1^2 + .1^2}{M} \right]^{\frac{1}{2}} \tag{5.7}$$

$$= \left[\sigma_n^2 + \frac{.03}{M} \right]^{\frac{1}{2}} \tag{5.8}$$

Additionally,

$$SNR_o \equiv \text{signal to noise ratio, with outlier contribution} \tag{5.9}$$

$$= \sigma_f / \sigma_{no} \tag{5.10}$$

From (5.5) we have

$$\sigma_f = SNR\ \sigma_n \tag{5.11}$$

Thus, combining (5.10) and (5.11),

$$SNR_o = SNR\ \frac{\sigma_n}{\sigma_{no}} \tag{5.12}$$

Furthermore

$$\delta SNR = SNR - SNR_o \tag{5.13}$$

$$= SNR(1 - \sigma_n/\sigma_{no}) \tag{5.14}$$

and

$$\frac{\delta SNR}{SNR} + \frac{\sigma_{no} - \sigma_n}{\sigma_{no}} \quad . \tag{5.15}$$

These altered standard deviations and signal to noise ratios were calculated for the noise levels pertinent to the various restorations that were performed. The results are contained in Table 9. These modified values for σ_n and SNR were then employed in NOCORR, and the d_{rms} achieved using these altered values are compared to the d_{rms} using the unaltered noise values. The results are in Table 10.

From Table 9 we observe a significant improvement in the MMSE restoration quality when the outlier contributions to the noise statistics are considered. Apparently the MMSE restoration method performs more accurately when it is provided with a more accurate assessment of the image vector variance.

One final observation to be made is also the least understood. Throughout the entire experiment it was consistently found that images that had been blurred with a triangularly shaped psf would simply not yield restorations as accurately as images that had been blurred by rectangular psf's of the same width, despite having identical noise and outlier contamination. If any such difference was to be anticipated, one would think that it would be the reverse of the observed outcome; i.e., that images blurred with a triangle would provide better restorations than images blurred with a rectangle. This is because the rectangle effectively blurs more than a triangle of the same width. In any event,

Table 9. Effects on noise statistics when outlier contribution is considered.

The outliers were each of magnitude .1, located at the 8^{th}, 16^{th} and 24^{th} image points.

SIGMA	σ_n	σ_{no}	SNR	SNR_o	$\frac{\delta SNR}{SNR}$
.0005	.00079	.0153	770	39.8	.95
.001	.0016	.0154	385.4	40.04	.90
.005	.0079	.0172	77.2	35.5	.54
.01	.016	.0220	38.5	28	.28
.05	.079	.0805	7.7	7.6	.02
.1	.16	.161	3.9	3.9	.005

Table 10. Comparison of NOCORR restoration RMS discrepancies for image statistics that neglect outliers vs. statistics that consider outliers.

Outlier distribution is the same as for Table 9.

SIGMA	σ_n	SNR	d_{rms}	σ_{no}	SNR_o	d_{rms}
.0005	.00079	768.8	.232	.0153	39.8	.113
.001	.0016	384.4	.231	.0154	40.04	.114
.005	.0079	76.9	.185	.0172	35.5	.127
.01	.016	38.4	.178	.0220	28.0	.154
.05	.079	7.7	.247	.0805	7.6	.244

this rather perplexing result has been observed by other individuals who have worked in image restoration (personal discussions).

In conclusion, it is felt that the MAE method of restoration may hold some promise in application to restoring data that is sparsely afflicted with outliers, and with low overall background noise. Of course, we have seen that median filtering can be extremely effective in deleting such points. But it is also true that median filtering itself perturbs and smooths the image data. If the inherent SNR of the data is sufficiently high to begin with, it can occur that the effects of median filtering can do more harm than good.

For the more general case, though, when the data is not blessed with such low levels of background noise, the MMSE method, used with or without median filtering (as circumstances dictate), seems to be the overall favored method.

APPENDIX A

LISTINGS OF COMPUTER PROGRAMS

```
      PROGRAM OBJGEN
C     THIS PROGRAM GENERATES AN OBJECT VECTOR OF N POINTS, N BEING
C     USER SUPPLIED. THE OBJECT IS A RAMP, RECT, AND THE POSITIVE
C     LOBE OF HALF A SINE WAVE. THE USER ALSO SUPPLIES THE PSF,
C     FROM WHICH THE PSF-MATRIX IS ASSEMBLED AND MULTIPLIED BY
C     THE OBJECT VECTOR.

      DIMENSION Y(256),PSF(9),OBJ(256),H(256,256)
      CHARACTER OUTFIL*16
      WRITE(5,5)
5     FORMAT(' ENTER THE NUMBER OF PTS. IN THE PSF, FORMAT(I2)')
      READ(5,10)NPSF
10    FORMAT(I2)
      WRITE(5,15)
15    FORMAT(' ENTER VALUES OF PSF, FORMAT(8F10.0)')
      READ(5,20)(PSF(K),K=1,NPSF)
20    FORMAT(8F10.0)

C     NORMALIZE PSF VALUES SUCH THAT SUM OF ALL PSF(I)'S=1.0
      SUMPSF=0.
      DO 21 I=1,NPSF
21    SUMPSF=SUMPSF+PSF(I)
      DO 22 I=1,NPSF
22    PSF(I)=PSF(I)/SUMPSF
      WRITE(5,25)
25    FORMAT(' ENTER NO. OF OBJECT POINTS AND BIAS, FORMAT(I3,F10.0)')
      READ(5,30)N,BIAS
30    FORMAT(I3,F10.0)
      PI2F=2.*8.*ACOS(-1.)/3.
      DO 20000 I=1,N
      X=FLOAT(I)/FLOAT(N)
      IF(X.LE.1.)OBJ(I)=SIN(PI2F*(X-.8125))+BIAS
      IF(X.LE..8125)OBJ(I)=BIAS
      IF(X.LE..6875)OBJ(I)=1.0+BIAS
      IF(X.LE..5)OBJ(I)=BIAS
      IF(X.LE..375)OBJ(I)=-8.*X+3.+BIAS
      IF(X.LE..25)OBJ(I)=4.0*X
20000 CONTINUE
40    FORMAT(8F10.0)

C     CALCULATE M, NO. OF IMAGE POINTS

      M=N+NPSF-1

C     DETERMINE ELEMENTS OF PSF MATRIX H((I,J). BLOCK CIRCULANT.

      DO 50 I=1,M
      DO 45 J=1,N
      IF (I.EQ.J)GO TO 55
      IF(J.GT.I)GO TO 60
      IJDEL=I-J
      LPSF=NPSF-IJDEL
      IF(LPSF.LT.1)GO TO 60
      H(I,J)=PSF(LPSF)
      GO TO 45
55    H(I,J)=PSF(NPSF)
      GO TO 45
60    H(I,J)=0.
      GO TO 45
45    CONTINUE
50    CONTINUE
```

```
C       NOW CALCULATE IMAGE VALUES
C       INITIALIZE
        DO 65 I=1,M
65      Y(I)=Ø.

        DO 7Ø I=1,M
        DO 75 J=1,N
        Y(I)=Y(I)+H(I,J)*OBJ(J)
75      CONTINUE
7Ø      CONTINUE
        WRITE(5,8Ø)
8Ø      FORMAT(' ENTER NAME OF OUTFIL')
        READ(5,85)OUTFIL
85      FORMAT(A16)
        OPEN(UNIT=1,NAME=OUTFIL,TYPE='NEW')
        WRITE(1,9Ø)(Y(I),I=1,M)
9Ø      FORMAT(1ØF1Ø.6)
        WRITE(1,9Ø)(OBJ(I),I=1,N)
        DO 1ØØ I=1,M
        WRITE(1,9Ø)(H(I,J),J=1,N)
1ØØ     CONTINUE
        STOP
        END
```

```
      PROGRAM NOISE
C     NOISE ADDS GAUSSIAN AND OUTLIER NOISE TO THE IMAGE VECTOR.
C     GAUSSIAN NOISE IS GENERATED FROM THE FLATLY DISTRIBUTED
C     RANDOM NUMBERS SUPPLIED BY THE COMPUTER'S RANDOM NUMBER
C     ROUTINE VIA THE TRANSFORMATION DESCRIBED ON PAGE 3-88
C     OF ROY FRIEDEN'S CLASS NOTES. TO ADD OUTLIERS THE USER
C     SPECIFIES THE APPROPRIATE INDEX AND THE CORRESPONDING
C     OUTLIER TO BE ADDED TO THAT IMAGE POINT.
      CHARACTER INFIL*16,OUTFIL*16
      REAL MAGNOI
      DIMENSION Y(256),H(256,256),OUTLY(2Ø),INDEX(2Ø),OBJ(256)
      WRITE(5,1Ø)
1Ø    FORMAT(' ENTER NAME OF FILE TO BE READ, FORMAT(A16)')
      READ(5,2Ø)INFIL
2Ø    FORMAT(A16)
      OPEN(UNIT=1,NAME=INFIL,TYPE='OLD')
      WRITE(5,25)
25    FORMAT(' ENTER FILE NAME TO BE WRITTEN, FORMAT(A16)')
      READ(5,3Ø)OUTFIL
3Ø    FORMAT(A16)
      OPEN(UNIT=2,NAME=OUTFIL,TYPE='NEW')
      WRITE(5,35)
35    FORMAT(' ENTER M,N AND SIGMA, FORMAT(2I3,F1Ø.Ø)')
      READ(5,4Ø)M,N,SIGMA
4Ø    FORMAT(2I3,F1Ø.Ø)
      TWOPI=2.Ø*ACOS(-1.Ø)
      SUMNOI=Ø.
      POWNOI=Ø.
      POWSIG=Ø.
C     READ, ADD NOISE, WRITE.
      ISEED1=Ø
      READ(1,6Ø)(Y(I),I=1,M)
      READ(1,6Ø)(OBJ(I),I=1,N)
      DO 75 I=1,M
      READ(1,6Ø)(H(I,J),J=1,N)
75    CONTINUE
      Y1=RAN(ISEED1)
      DO 5Ø I=1,M
      Y1=RAN(ISEED1)
      Y2=RAN(ISEED1)
      X1=-4.6Ø6*LOG(Y1)
      X1=SQRT(X1)*SIGMA*COS(TWOPI*Y2)
      SUMNOI=SUMNOI+X1
      POWNOI=POWNOI+X1*X1
      POWSIG=POWSIG+Y(I)*Y(I)
      Y(I)=Y(I)+X1
5Ø    CONTINUE
      WRITE(5,95)
95    FORMAT(' WANT TO ADD OUTLIER NOISE? NO=Ø,YES=1.FORMAT(I1')
      READ(5,1ØØ)NOUT
1ØØ   FORMAT(I1)
      IF(NOUT.EQ.Ø)GO TO 1Ø5
      WRITE(5,11Ø)
11Ø   FORMAT(' HOW MANY IMAGE POINTS DO YOU WISH TO ADD OUTLIER NOISE
     1TO - FORMAT(I3?')
      READ(5,115)NOUT
115   FORMAT(I3)
      WRITE(5,12Ø)
12Ø   FORMAT(' ENTER INDICES OF IMAGE POINTS & NOISE TO BE ADDED.
     1FORMAT(I3,F1Ø.Ø')
      READ(5,125)(INDEX(I),OUTLY(I),I=1,NOUT)
```

```
125     FORMAT(I3,F10.0)
        DO 130 I=1,NOUT
130     Y(INDEX(I))=Y(INDEX(I))+OUTLY(I)
105     WRITE(2,60)(Y(I),I=1,M)
60      FORMAT(10F10.6)
        WRITE(2,60)(OBJ(I),I=1,N)
        DO 80 I=1,M
        WRITE(2,60)(H(I,J),J=1,N)
80      CONTINUE
        IF(POWNOI.EQ.0.)GO TO 81
        SNR=SQRT(POWSIG/POWNOI)
        GO TO 83
81      SNR=-1.0
        GO TO 83
83      AVENOI=SUMNOI/M
        STANDV=(POWNOI/M)**.5
        WRITE(5,85)SNR,AVENOI,STANDV
        WRITE(2,90)SNR,AVENOI,STANDV
85      FORMAT(' SNR=',F10.1,/,' AVERAGE NOISE VALUE=',F10.8,
        1/,' NOISE STANDARD DEVIATION =',F10.8)
90      FORMAT(F10.1,F10.8,F10.8)
        STOP
        END
```

```
      PROGRAM MINABS
      DIMENSION OBJ(256),RESID(256),IWRKSP(256),OBJEST(256)
      CHARACTER PLOTFIL*16,INFIL*16,OUTFIL*16
      COMMON Y(256),H(256,256)
      TOL=1Ø.Ø**(-1Ø)
      WRITE(5,6)
6     FORMAT(' ENTER N, NO. OF OBJ PTS., AND NPSF, THE NO. OF
     1 PSF PTS., FORMAT(2I3)')
      READ(5,11)N,NPSF
11    FORMAT(2I3)
      M=N+NPSF-1
      M2=M+2
      N2=N+2
      WRITE(5,4Ø)
4Ø    FORMAT(' ENTER FILENAME TO BE READ')
      READ(5,45)INFIL
45    FORMAT(A16)
      OPEN(UNIT=1,NAME=INFIL,TYPE='OLD')
      WRITE(5,41)
41    FORMAT(' ENTER NAME OF OUTFILE, FORMAT(A16):')
      READ(5,45)OUTFIL
      OPEN(UNIT=2,NAME=OUTFIL,TYPE='NEW')
      WRITE(2,5)
5     FORMAT(' IMAGE DATA IS:'/)
      READ(1,1Ø)(Y(I),I=1,M)
1Ø    FORMAT(1ØF1Ø.6)
      WRITE(2,1Ø)(Y(I),I=1,M)
      WRITE(2,5Ø)
5Ø    FORMAT(' OBJECT DATA IS:'/)
      READ(1,1Ø)(OBJ(I),I=1,N)
      WRITE(2,1Ø)(OBJ(I),I=1,N)


      DO 6Ø I=1,M
      READ(1,1Ø)(H(I,J),J=1,N)
6Ø    CONTINUE
      READ(1,52)SNR,AVENOI,STANDV
52    FORMAT(F1Ø.1,2F1Ø.6)
      WRITE(5,54)SNR,AVENOI,STANDV
      WRITE(2,54)SNR,AVENOI,STANDV
54    FORMAT(/,' SNR=',F1Ø.1,/,' AVERAGE NOISE VALUE=',F1Ø.6,
     1/,' NOISE STANDARD DEVIATION=',F1Ø.6)

      CALL L1(M,N,M2,N2,TOL,OBJEST,RESID,IWRKSP)

      WRITE(5,25)H(M+1,N+1),H(M+1,N2),H(M2,N+1),H(M2,N2)
      WRITE(2,25)H(M+1,N+1),H(M+1,N2),H(M2,N+1),H(M2,N2)
25    FORMAT(' MINIMUM SUM OF ABS(RESID(I))=',F8.3,/,
     1' RANK OF H(I,J)=',F4.Ø,/,' EXIT CODE=',F4.Ø,/,
     2' NO. OF SIMPLEX ITERATIONS=',F4.Ø)

      WRITE(2,65)
65    FORMAT(' OBJECT ESTIMATE IS:',/)
      WRITE(2,1Ø)(OBJEST(I),I=1,N)

      RMSERR=Ø.
      DO 33 I=1,N
33    RMSERR=RMSERR+(OBJEST(I)-OBJ(I))**2
      RMSERR=(RMSERR/FLOAT(N))**.5
      WRITE(5,43)RMSERR
      WRITE(2,43)RMSERR
43    FORMAT(' ROOT MEAN SQUARE DISCREPANCY=',F1Ø.6)

      WRITE(2,7Ø)
7Ø    FORMAT(' RESIDUALS ARE:',/)
      WRITE(2,1Ø)(RESID(I),I=1,N)
```

```
      WRITE(5,1ØØ)
1ØØ   FORMAT(' DO YOU WANT A FILE WRITTEN FOR PLOTTING? YES=1,NO=Ø')
      READ(5,1Ø5)IDEC
1Ø5   FORMAT(I1)
      IF(IDEC.EQ.Ø)GO TO 1Ø1
      WRITE(5,11Ø)
11Ø   FORMAT(' ENTER 1 FOR OBJECT,2 FOR OBJECT ESTIMATE,3 FOR IMAGE')
      READ(5,1Ø5)IDEC
      WRITE(5,115)
115   FORMAT(' ENTER FILENAME OF DATA TO BE PLOTTED, FORMATA16')
      READ(5,45)PLOTFIL
      OPEN(UNIT=4,NAME=PLOTFIL,TYPE='NEW')
      IF(IDEC.EQ.1)WRITE(4,1Ø)(OBJ(I),I=1,N)
      IF(IDEC.EQ.2)WRITE(4,1Ø)(OBJEST(I),I=1,N)
      IF(IDEC.EQ.3)WRITE(4,1Ø)(Y(I),I=1,M)
1Ø1   STOP
      END
      SUBROUTINE L1(M,N,M2,N2,TOLER,X,E,S)
C
C     THIS SUBROUTINE USES A MODIFICATION OF THE SIMPLEX METHOD
C     OF LINEAR PROGRAMMING TO CALCULATE AN L1 SOLUTION TO AN
C     OVER-DETERMINED SYSTEM OF LINEAR EQUATIONS.
C     DESCRIPTION OF PARAMETERS.
C     M     NUMBER OF EQUATIONS.
C     N     NUMBER OF UNKNOWNS 8M.GE.N)
C     M2       SET EQUAL TO M+2 FOR ADJUSTABLE DIMENSIONS.
C     N2 SET EQUAL TO N+2 FOR ADJUSTABLE DIMENSIONS.
C     A     TWO DIMENSIONAL REAL ARRY OF SIZE (M2,N2)
C     ON ENTRY, THE COEFFICIENTS OF THE MATRIX MUST BE
C     STORED IN THE FIRST M ROWS AND N COLUMNS OF A.
C     THESE VALUE ARE DESTROYED BY THE SUBROUTINE.
C     B     ONE DIMENSIONAL REAL ARRY OF SIZE M. ON ENTRY, B

C     MUST CONTAIN THE RIGHT HAND SIDE OF THE EQUATIONS.
C     THESE VALUES ARE DESTROYED BY THE SUBROUTINE.
C     TOLER A SMALL POSITIVE TOLERANCE.  EMPIRICAL EVIDENCE
C     SUGGESTS TOLER=1Ø**(-D*2/3) WHERE D REPRESENTS
C     THE NUMBER OF DECIMAL DIGITS OF ACURACY AVAILABLE
C     (SEE DESCRIPTION).
C     X     ONE DIMENSIONAL REAL ARRY OF SIZE N. ON EXIT, THIS
C     ARRAY CONTAINS A SOLUTION TO THE L1 PROBLEM.
C     E     ONE DIMENSIONAL REAL ARRAY OF SIZE M. ON EXIT, THIS
C     ARRAY CONTAINS THE RESIDUALS IN THE EQUATIONS.
C     S     INTEGER ARRAY OF SIZE M USED FOR WORKSPACE.
C     ON EXIT FROM THE SUBROUTINE, THE ARRAY A CONTAINS THE
C     A(M+1,N+1)  THE MINIMUM SUM OF THE ABSOLUTE VALUES OF
C     THE RESIDUALS
C     A(M=1,N+2) THE RANK OF THE MATRIX OF COEFFICIENTS.
C     A(M+2,N+1) EXIT CODE WITH VALUES.
C     Ø - OPTIMAL SOLUTION WHICH IS PROBABLY NON-
C     UNIQUE (SEE DESCRIPTION)
C     1 - UNIQUE OPTIMAL SOLUTION
C     2 - CALCULATIONS TERMINATED PREMATURELY DUE TO
C     ROUNDING ERRORS
C     A(M+2,N+2) NUMBER OF SIMPLEX ITERATIONS PERFORMED.
      COMMON B(256),A(256,256)
      DOUBLE PRECISION SUM
      REAL MIN,MAX,E(M),X(N)
      INTEGER OUT,S(M)
      LOGICAL STAGE, TEST
C
C     BIG MUST BE SET EQUAL TO ANY VERY LARGE REAL CONSTANT.
C     ITS VALUE HERE IS APPROPRIATE FOR THE IBM 37Ø
C
      DATA BIG/1.E38/
C
C     INITIALIZATION.
C
```

```
      M1=M+1
      N1=N+1
      DO 1Ø J=1,N
      A(M2,J)=J
      X(J)=Ø.
1Ø    CONTINUE
      DO 4Ø I=1,M
      A(I,N2)=N+I
      A(I,N1)=B(I)
      IF (B(I).GE.Ø.) GO TO 3Ø
      DO 2Ø J=1,N2
      A(I,J)=-A(I,J)
2Ø    CONTINUE
3Ø    E(I)=Ø.
4Ø    CONTINUE
C
C     COMPUTE THE MARGINAL COSTS.
C
      DO 6Ø J=1,N1
      SUM=Ø.ØØ
      DO 5Ø I=1,M
      SUM=SUM+A(I,J)
5Ø    CONTINUE
      A(M1,J)=SUM
6Ø    CONTINUE
C
C     STAGE 1
C     DETERMINE THE VECTOR TO ENTER THE BASIS.
C
      STAGE=.TRUE.
      KOUNT=Ø
      KR=1
      KL=1
7Ø    MAX=-1.
      DO 8Ø J=KR,N
      IF (ABS(A(M2,J)).GT.N) GO TO 8Ø
      D=ABS(A(M1,J))
      IF (D.LE.MAX) GO TO 8Ø
      MAX=D
      IN=J
8Ø    CONTINUE
      IF (A(M1,IN).GE.Ø.) GO TO 1ØØ
      DO 9Ø I=1,M2
      A(I,IN)=-A(I,IN)
9Ø    CONTINUE
C     DETERMINE THE VECTOR TO LEAVE THE BASIS
1ØØ   K=Ø
      DO 11Ø I=KL,M
      D=A(I,IN)
      IF (D.LE.TOLER) GO TO 11Ø
      K=K+1
      B(K)=A(I,N1)/D
      S(K)=I
      TEST=.TRUE.
11Ø   CONTINUE
12Ø   IF (K.GT.Ø) GO TO 13Ø
      TEST=.FALSE.
      GO TO 15Ø
13Ø   MIN=BIG
      DO 14Ø I=1,K
      IF (B(I).GE.MIN) GO TO 14Ø
      J=I
      MIN=B(I)
      OUT=S(I)
14Ø   CONTINUE
      B(J)=B(K)
      S(J)=S(K)
      K=K-1
```

```
C       CHECK FOR LINEAR DEPENDENCE IN STAGE I.
C
15Ø     IF (TEST.OR..NOT.STAGE) GO TO 17Ø
        DO 16Ø I=1,M2
        D=A(I,KR)
        A(I,KR)=A(I,IN)
        A(I,IN)=D
16Ø     CONTINUE
        KR=KR+1
        GO TO 26Ø
17Ø     IF (TEST) GO TO 18Ø
        A(M2,N1)=2.
        GO TO 35Ø
18Ø     PIVOT=A(OUT,IN)
        IF (A(M1,IN)-PIVOT-PIVOT.LE.TOLER) GO TO 2ØØ
        DO 19Ø J=KR,N1
        D=A(OUT,J)
        A(M1,J)=A(M1,J)-D-D
        A(OUT,J)=-D
19Ø     CONTINUE
        A(OUT,N2)=-A(OUT,N2)
        GO TO 12Ø
C
C       PIVOT ON A(OUT,IN).
C
2ØØ     DO 21Ø J=KR,N1
        IF (J.EQ.IN) GO TO 21Ø
        A(OUT,J)=A(OUT,J)/PIVOT
21Ø     CONTINUE
        DO 23Ø I=1,M1
        IF (I.EQ.OUT) GO TO 23Ø
        D=A(I,IN)
        DO 22Ø J=KR,N1
        IF (J.EQ.IN) GO TO 22Ø
        A(I,J)=A(I,J)-D*A(OUT,J)
22Ø     CONTINUE
23Ø     CONTINUE
        DO 24Ø I=1,M1
        IF (I.EQ.OUT) GO TO 24Ø




        A(I,IN)=-A(I,IN)/PIVOT
24Ø     CONTINUE
        A(OUT,IN)=1./PIVOT
        D=A(OUT,N2)
        A(OUT,N2)=A(M2,IN)
        A(M2,IN)=D
        KOUNT=KOUNT+1
        IF (.NOT.STAGE) GO TO 27Ø
C
C       INTERCHANGE ROWS IN STAGE 1.
C
        KL=KL+1
        DO 25Ø J=KR,N2
        D=A(OUT,J)
        A(OUT,J)=A(KOUNT,J)
        A(KOUNT,J)=D
25Ø     CONTINUE
26Ø     IF (KOUNT+KR.NE.N1) GO TO 7Ø
C
C       STAGE II.
C
        STAGE=.FALSE.
C
C       DETERMINE THE VECTOR TO ENTER THE BASIS.
```

```
27Ø     MAX=-BIG
        DO 29Ø J=KR,N
        D=A(M1,J)
        IF (D.GE.Ø.) GO TO 28Ø
        IF (D.GT.(-2.)) GO TO 29Ø
        D=-D-2.
28Ø     IF (D.LE.MAX) GO TO 29Ø
        MAX=D
        IN=J
29Ø     CONTINUE
        IF (MAX.LE.TOLER) GO TO 31Ø
        IF (A(M1,IN).GT.Ø.) GO TO 1ØØ
        DO 3ØØ I=1,M2
        A(I,IN)=-A(I,IN)
3ØØ     CONTINUE
        A(M1,IN)=A(M1,IN)-2.
        GO TO 1ØØ
C
C       PREPARE OUTPUT.
C
31Ø     L=KL-1
        DO 33Ø I=1,L
        IF (A(I,N1).GE.Ø.) GO TO 33Ø
        DO 32Ø J=KR,N2
        A(I,J)=-A(I,J)
32Ø     CONTINUE
33Ø     CONTINUE
        A(M2,N1)=Ø.
        IF (KR.NE.1) GO TO 35Ø
        DO 34Ø J=1,N
        D=ABS(A(M1,J))
        IF (D.LE.TOLER.OR.2.-D.LE.TOLER) GO TO 35Ø
34Ø     CONTINUE
        A(M2,N1)=1.
35Ø     DO 38Ø I=1,M
        K=A(I,N2)
        D=A(I,N1)



        IF (K.GT.Ø) GO TO 36Ø
        K=-K
        D=-D
36Ø     IF (I.GE.KL) GO TO 37Ø
        X(K)=D
        GO TO 38Ø
37Ø     K=K-N
        E(K)=D
38Ø     CONTINUE
        A(M2,N2)=KOUNT
        A(M1,N2)=N1-KR
        SUM=Ø.DØ
        DO 39Ø I=KL,M
        SUM=SUM+A(I,N1)
39Ø     CONTINUE
        A(M1,N1)=SUM
        RETURN
C
        END
```

```
      SUBROUTINE MATINV(A,N,B,M,DETERM,IPIVOT,INDEX,NMAX,ISCALE)
      DIMENSION IPIVOT(N),A(NMAX,N),B(NMAX,M),INDEX(NMAX,2)
      EQUIVALENCE (IROW,JROW),(ICOLUM,JCOLUM),(AMAX,T,SWAP)

C     INITIALIZATION

5     ISCALE=Ø
6     R1=1Ø.ØE36
7     R2=1.Ø/R1
1Ø    DETERM=1.Ø
15    DO 2Ø J=1,N
2Ø    IPIVOT(J)=Ø
3Ø    DO 55Ø I=1,N

C     SEARCH FOR PIVOT ELEMENT

4Ø    AMAX=Ø.
45    DO 1Ø5 J=1,N
5Ø    IF(IPIVOT(J)-1)6Ø, 1Ø5, 6Ø
6Ø    DO 1ØØ K=1,N
7Ø    IF(IPIVOT(K)-1)8Ø, 1ØØ, 74Ø
8Ø    IF(ABS(AMAX)-ABS(A(J,K)))85,1ØØ,1ØØ
85    IROW=J
9Ø    ICOLUM=K
95    AMAX=A(J,K)
1ØØ   CONTINUE
1Ø5   CONTINUE
      IF (AMAX)11Ø,1Ø6,11Ø
1Ø6   DETERM=Ø.
      ISCALE=Ø
      GO TO 74Ø
11Ø   IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1

C     INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL

13Ø   IF (IROW-ICOLUM)14Ø,26Ø,14Ø
14Ø   DETERM=-DETERM
15Ø   DO 2ØØ L=1,N
16Ø   SWAP=A(IROW,L)
17Ø   A(IROW,L)=A(ICOLUM,L)
2ØØ   A(ICOLUM,L)=SWAP
2Ø5   IF(M) 26Ø,26Ø,21Ø
21Ø   DO 25Ø L=1,M
22Ø   SWAP=B(IROW,L)
23Ø   B(IROW,L)=B(ICOLUM,L)
25Ø   B(ICOLUM,L)=SWAP
26Ø   INDEX(I,1)=IROW
27Ø   INDEX(I,2)=ICOLUM
31Ø   PIVOT=A(ICOLUM,ICOLUM)

C     SCALE THE DETERMINANT
1ØØØ  PIVOTI=PIVOT
1ØØ5  IF(ABS(DETERM)-R1)1Ø3Ø,1Ø1Ø,1Ø1Ø
1Ø1Ø   DETERM=DETERM/R1
      ISCALE=ISCALE+1
      IF(ABS(DETERM)-R1)1Ø6Ø,1Ø2Ø,1Ø2Ø
1Ø2Ø   DETERM=DETERM/R1
      ISCALE=ISCALE+1
      GO TO 1Ø6Ø
1Ø3Ø   IF(ABS(DETERM)-R2)1Ø4Ø,1Ø4Ø,1Ø6Ø
1Ø4Ø   DETERM=DETERM*R1
      ISCALE=ISCALE-1
      IF(ABS(DETERM)-R2)1Ø5Ø,1Ø5Ø,1Ø6Ø
1Ø5Ø   DETERM=DETERM*R1
      ISCALE=ISCALE-1
1Ø6Ø   IF(ABS(PIVOTI)-R1)1Ø9Ø,1Ø7Ø,1Ø7Ø
1Ø7Ø   PIVOTI=PIVOTI/R1
```

```
      ISCALE=ISCALE+1
      IF(ABS(PIVOTI)-R1)32Ø,1Ø8Ø,1Ø8Ø
1Ø8Ø  PIVOTI=PIVOTI/R1
      ISCALE=ISCALE+1
      GO TO 32Ø
1Ø9Ø  IF(ABS(PIVOTI)-R2)2ØØØ,2ØØØ,32Ø
2ØØØ  PIVOTI=PIVOTI*R1
      ISCALE=ISCALE-1
      IF(ABS(PIVOTI)-R2)2Ø1Ø,2Ø1Ø,32Ø
2Ø1Ø  PIVOTI=PIVOTI*R1
      ISCALE=ISCALE-1
32Ø   DETERM=DETERM*PIVOTI

C     DIVIDE PIVOT ROW BY PIVOT ELEMENT

33Ø   A(ICOLUM,ICOLUM)=1.
34Ø   DO 35Ø L=1,N
35Ø   A(ICOLUM,L)=A(ICOLUM,L)/PIVOT
355   IF(M)38Ø,38Ø,36Ø
36Ø   DO 37Ø L=1,M
37Ø   B(ICOLUM,L)=B(ICOLUM,L)/PIVOT

C     REDUCE NON-PIVOT ROWS

38Ø   DO 55Ø L1=1,N
39Ø   IF(L1-ICOLUM)4ØØ,55Ø,4ØØ
4ØØ   T=A(L1,ICOLUM)
42Ø   A(L1,ICOLUM)=Ø.
43Ø   DO 45Ø L=1,N
45Ø   A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
455   IF(M)55Ø,55Ø,46Ø
46Ø   DO 5ØØ L=1,M
5ØØ   B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
55Ø   CONTINUE

C     INTERCHANGE COLUMNS

6ØØ   DO 71Ø I=1,N
61Ø   L=N+1-I
62Ø   IF (INDEX(L,1)-INDEX(L,2))63Ø,71Ø,63Ø
63Ø   JROW=INDEX(L,1)
64Ø   JCOLUM=INDEX(L,2)
65Ø   DO 7Ø5 K=1,N
66Ø   SWAP=A(K,JROW)
67Ø   A(K,JROW)=A(K,JCOLUM)
7ØØ   A(K,JCOLUM)=SWAP
7Ø5   CONTINUE
71Ø   CONTINUE
74Ø   RETURN
      END
```

```
      PROGRAM INVMAT
C     THIS PROGRAM OBTAINS THE LEAST SQUARES SOLUTION TO THE
C     CLASSICAL IMAGING EQUATION BY DETERMINING THE PSEUDO-INVERSE
C     OF THE PSF-MATRIX THAT WAS INITIALLY USED TO BLUR THE OBJECT.
      DIMENSION Y(256),OBJ(256),H(256,256),HTH(256,256),IPIVOT(256)
      DIMENSION INDEX(256,2),DATA(256)
      CHARACTER PLOTFIL*16,INFIL*16,OUTFIL*16
      WRITE(5,5)
5     FORMAT(' ENTER INFIL, FORMAT(A16)')
      READ(5,10)INFIL
10    FORMAT(A16)
      WRITE(5,15)
15    FORMAT(' ENTER OUTFIL')
      READ(5,10)OUTFIL
      WRITE(5,20)
20    FORMAT(' ENTER N(# OF OBJ PTS.), NPSF(# OF PSF PTS.),FORMAT2I3')
      READ(5,25)N,NPSF
25    FORMAT(2I3)

      OPEN(UNIT=1,NAME=INFIL,TYPE='OLD')
      OPEN(UNIT=2,NAME=OUTFIL,TYPE='NEW')
      M=N+NPSF-1
      READ(1,30)(Y(I),I=1,M)
30    FORMAT(10F10.6)
      READ(1,30)(OBJ(I),I=1,N)
      DO 35 I=1,M
      READ(1,30)(H(I,J),J=1,N)
35    CONTINUE
      READ(1,36)SNR,AVENOI,STANDV
36    FORMAT(F10.1,2F10.6)
      NC=N
      NR=M
C     H IS M BY N. HT (H-TRANSPOSE) IS N BY M. HTH IS THUS N BY N.
C     COMPUTE HTH.
C     INITIALIZE.
      DO 40 I=1,N
      DO 45 J=1,N
      HTH(I,J)=0.
45    CONTINUE
      DATA(I)=0.
40    CONTINUE
      DO 50 I=1,N
      DO 55 K=1,N
      DO 60 J=1,M
      HTH(I,K)=HTH(I,K)+H(J,I)*H(J,K)
60    CONTINUE
55    CONTINUE
50    CONTINUE
C     COMPUTE DATA(I)=HT(I,J)*Y(J)=H(J,I)*Y(J)
      DO 51 I=1,N
      DO 53 J=1,M
      DATA(I)=DATA(I)+H(J,I)*Y(J)
53    CONTINUE
51    CONTINUE

      CALL MATINV(HTH,N,DATA,1,DETERM,IPIVOT,INDEX,256,ISCALE)
      WRITE(2,85)
85    FORMAT(' LEAST SQUARES RESULTS VIA PSEUDO-INVERSE METHOD')
      WRITE(2,65)
65    FORMAT(' IMAGE DATA:')
      WRITE(2,70)(Y(I),I=1,M)
```

```
7Ø      FORMAT(12F1Ø.4)
        WRITE(2,71)SNR,AVENOI,STANDV
        WRITE(5,71)SNR,AVENOI,STANDV
71      FORMAT(' SNR=',F1Ø.1,/,' AVERAGE NOISE VALUE=',F1Ø.6,/,
        1' NOISE STANDARD DEVIATION=',F1Ø.6)
        WRITE(2,75)
75      FORMAT(' ORIGINAL OBJECT:')
        WRITE(2,7Ø)(OBJ(I),I=1,N)
        WRITE(2,8Ø)
8Ø      FORMAT('OBJECT ESTIMATE VIA LEAST SQUARES')
        WRITE(2,7Ø)(DATA(I),I=1,N)
        RMSERR=Ø.
        DO 9Ø I=1,N
9Ø      RMSERR=RMSERR+(OBJ(I)-DATA(I))**2
        RMSERR=(RMSERR/FLOAT(N))**.5
        WRITE(5,95)RMSERR
        WRITE(2,95)RMSERR
95      FORMAT(' ROOT MEAN SQUARE DISCREPANCY=',F1Ø.6)
        WRITE(5,1ØØ)
1ØØ     FORMAT(' DO YOU WANT A FILE WRITTEN FOR PLOTTING? YES=1,NO=Ø')
        READ(5,1Ø5)IDEC
1Ø5     FORMAT(I1)
        IF(IDEC.EQ.Ø)GO TO 1Ø1
        WRITE(5,11Ø)
11Ø     FORMAT(' ENTER 1 FOR OBJECT,2 FOR OBJECT ESTIMATE,3 FOR IMAGE')
        READ(5,1Ø5)IDEC
        WRITE(5,115)
115     FORMAT(' ENTER FILENAME OF DATA TO BE PLOTTED-FORMATA16')
        READ(5,1Ø)PLOTFIL
        OPEN(UNIT=4,NAME=PLOTFIL,TYPE='NEW')
        IF(IDEC.EQ.1)WRITE(4,3Ø)(OBJ(I),I=1,N)
        IF(IDEC.EQ.2)WRITE(4,3Ø)(DATA(I),I=1,N)
        IF(IDEC.EQ.3)WRITE(4,3Ø)(Y(I),I=1,M)
1Ø1     STOP
        END
```

```
      PROGRAM NOCORR
C     THIS PROGRAM OBTAINS THE MMSE SOLUTION TO THE
C     CLASSICAL IMAGING EQUATION. THE OBJECT ENSEMBLE IS ASSUMED
C     UNCORRELATED, GIVING A DIAGONAL AUTO-CORRELATION MATRIX.
      DIMENSION Y(256),OBJ(256),H(256,256),RF(256,256),IPIVOT(256)
      DIMENSION INDEX(256,2),DATA(256)
      DIMENSION HMESS(256,256)
      CHARACTER PLOTFIL*16,INFIL*16,OUTFIL*16
      WRITE(5,5)
5     FORMAT(' ENTER INFIL, FORMAT(A16)')
      READ(5,10)INFIL
10    FORMAT(A16)
      WRITE(5,15)
15    FORMAT(' ENTER OUTFIL')
      READ(5,10)OUTFIL
      WRITE(5,20)
20    FORMAT(' ENTER N(# OF OBJ PTS.), NPSF(# OF PSF PTS.),FORMAT2I3')
      READ(5,25)N,NPSF
25    FORMAT(2I3)

      OPEN(UNIT=1,NAME=INFIL,TYPE='OLD')
      OPEN(UNIT=2,NAME=OUTFIL,TYPE='NEW')
      M=N+NPSF-1
      READ(1,30)(Y(I),I=1,M)
30    FORMAT(10F10.6)
      READ(1,30)(OBJ(I),I=1,N)
      DO 35 I=1,M
      READ(1,30)(H(I,J),J=1,N)
35    CONTINUE
      READ(1,36)SNR,AVENOI,STANDV
36    FORMAT(F10.1,2F10.8)
      WRITE(5,400)
400   FORMAT(' DO YOU WISH TO ALTER SNR AND STANDV? YES=1,NO=0')
      READ(5,105)ICHNG
      IF(ICHNG.EQ.0)GO TO 420
      WRITE(5,440)
440   FORMAT(' ENTER STANDV,SNR. FORMAT(F10.2,F10.8)')
      READ(5,460)STANDV,SNR
460   FORMAT(F10.2,F10.8)
420   VARIAN=STANDV*STANDV
      SIGF2=VARIAN*SNR*SNR
      IF(SNR.EQ.-1.)SIGF2=1.
C     INITIALIZE HMESS
      DO 291 I=1,M
      DO 292 J=1,M
      HMESS(I,J)=0.
292   CONTINUE
291   CONTINUE
C     FORM HMESS=H*RFHT+RN
      DO 300 I=1,M
      DO 310 K=1,M
      DO 320 J=1,N
320   HMESS(I,K)=HMESS(I,K)+H(I,J)*H(K,J)
      HMESS(I,K)=HMESS(I,K)*SIGF2
      IF(I.EQ.K)HMESS(I,K)=HMESS(I,K)+VARIAN
310   CONTINUE
300   CONTINUE
      WRITE(2,65)
65    FORMAT(' IMAGE DATA:')
      WRITE(2,70)(Y(I),I=1,M)
70    FORMAT(12F10.4)
```

```
      CALL MATINV(HMESS,M,Y,1,DETERM,IPIVOT,INDEX,256,ISCALE)
15Ø   FORMAT(8F1Ø.6)
C     Y NOW CONTAINS (HMESS)^-1*Y. SO, NOW WANT
C     DATA=RFHT*Y, WHERE DATA IS FINAL RESTORATION.
C     INITIALIZE.
      DO 33Ø I=1,N
33Ø   DATA(I)=Ø.
      DO 34Ø  I=1,N
      DO 35Ø J=1,M
35Ø   DATA(I)=DATA(I)+H(J,I)*Y(J)*SIGF2
34Ø   CONTINUE
      WRITE(2,71)SNR,AVENOI,STANDV
      WRITE(5,71)SNR,AVENOI,STANDV
71    FORMAT(' SNR=',F1Ø.1,/,' AVERAGE NOISE VALUE=',F1Ø.8,/,
      1' NOISE STANDARD DEVIATION=',F1Ø.8)
      WRITE(2,75)
75    FORMAT(' ORIGINAL OBJECT:')
      WRITE(2,7Ø)(OBJ(I),I=1,N)
      WRITE(2,8Ø)
8Ø    FORMAT('OBJECT ESTIMATE VIA MMSE, UNCORRELATED OBJECT ENSEMBLE')
      WRITE(2,7Ø)(DATA(I),I=1,N)
      RMSERR=Ø.
      DO 9Ø I=1,N
9Ø    RMSERR=RMSERR+(OBJ(I)-DATA(I))**2
      RMSERR=(RMSERR/FLOAT(N))**.5
      WRITE(5,95)RMSERR
      WRITE(2,95)RMSERR
95    FORMAT(' ROOT MEAN SQUARE DISCREPANCY=',F1Ø.6)
      WRITE(5,1ØØ)
1ØØ   FORMAT(' DO YOU WANT A FILE WRITTEN FOR PLOTTING? YES=1,NO=Ø')
      READ(5,1Ø5)IDEC
1Ø5   FORMAT(I1)
      IF(IDEC.EQ.Ø)GO TO 1Ø1
      WRITE(5,11Ø)
11Ø   FORMAT(' ENTER 1 FOR OBJECT,2 FOR OBJECT ESTIMATE,3 FOR IMAGE')
      READ(5,1Ø5)IDEC
      WRITE(5,115)
115   FORMAT(' ENTER FILENAME OF DATA TO BE PLOTTED-FORMATA16')
      READ(5,1Ø)PLOTFIL
      OPEN(UNIT=4,NAME=PLOTFIL,TYPE='NEW')
      IF(IDEC.EQ.1)WRITE(4,3Ø)(OBJ(I),I=1,N)
      IF(IDEC.EQ.2)WRITE(4,3Ø)(DATA(I),I=1,N)
      IF(IDEC.EQ.3)WRITE(4,3Ø)(Y(I),I=1,M)
1Ø1   STOP
      END
```

```
      PROGRAM STARGEN
C     STARGEN IS SIMILAR TO OBJGEN. INSTEAD OF THE RAMP-RECT-SINE
C     OBJECT PRODUCED BY OBJGEN, STARGEN CREATES AN OBJECT THAT
C     SIMULATES VARIOUS IMPULSE OBJECTS ('STARS') SEPARATED BY
C     1, 2, OR 3 PIXELS.

      DIMENSION Y(256),PSF(9),OBJ(256),H(256,256)
      CHARACTER OUTFIL*16
      WRITE(5,5)
5     FORMAT(' ENTER THE NUMBER OF PTS. IN THE PSF, FORMAT(I2)')
      READ(5,1Ø)NPSF
1Ø    FORMAT(I2)
      WRITE(5,15)
15    FORMAT(' ENTER VALUES OF PSF, FORMAT(8F1Ø.Ø)')
      READ(5,2Ø)(PSF(K),K=1,NPSF)
2Ø    FORMAT(8F1Ø.Ø)

C     NORMALIZE PSF VALUES SUCH THAT SUM OF ALL PSF(I)'S=1.Ø
      SUMPSF=Ø.
      DO 21 I=1,NPSF
21    SUMPSF=SUMPSF+PSF(I)
      DO 22 I=1,NPSF
22    PSF(I)=PSF(I)/SUMPSF
      N=124
      DO 2ØØØ I=1,N
2ØØØ  OBJ(I)=Ø.
C     INSERT 'STARS'
      OBJ(1Ø)=1.
      OBJ(3Ø)=1.
      OBJ(31)=.5
      OBJ(5Ø)=1.
      OBJ(51)=1.
      OBJ(7Ø)=1.
      OBJ(72)=1.
      OBJ(85)=1.
      OBJ(88)=1.
      OBJ(11Ø)=1.
      OBJ(114)=1.

C     CALCULATE M, NO. OF IMAGE POINTS

      M=N+NPSF-1

C     DETERMINE ELEMENTS OF PSF MATRIX H(I,J). BLOCK CIRCULANT.

      DO 5Ø I=1,M
      DO 45 J=1,N
      IF (I.EQ.J)GO TO 55
      IF(J.GT.I)GO TO 6Ø
      IJDEL=I-J
      LPSF=NPSF-IJDEL
      IF(LPSF.LT.1)GO TO 6Ø
      H(I,J)=PSF(LPSF)
      GO TO 45
55    H(I,J)=PSF(NPSF)
      GO TO 45
6Ø    H(I,J)=Ø.
      GO TO 45
45    CONTINUE
5Ø    CONTINUE
```

```
C       NOW CALCULATE IMAGE VALUES
C       INITIALIZE
        DO 65 I=1,M
65      Y(I)=Ø.

        DO 7Ø I=1,M
        DO 75 J=1,N
        Y(I)=Y(I)+H(I,J)*OBJ(J)
75      CONTINUE
7Ø      CONTINUE
        WRITE(5,8Ø)
8Ø      FORMAT(' ENTER NAME OF OUTFIL')
        READ(5,85)OUTFIL
85      FORMAT(A16)
        OPEN(UNIT=1,NAME=OUTFIL,TYPE='NEW')
        WRITE(1,9Ø)(Y(I),I=1,M)
9Ø      FORMAT(1ØF1Ø.6)
        WRITE(1,9Ø)(OBJ(I),I=1,N)
        DO 1ØØ I=1,M
        WRITE(1,9Ø)(H(I,J),J=1,N)
1ØØ     CONTINUE
        STOP
        END
```

REFERENCES

Andrews, H. C., and B. R. Hunt, Digital Image Restoration, Prentice-Hall, New Jersey, 1977.

Barrodale, I., and F. D. K. Roberts, "Applications of Mathematical Programming to ℓ_p Approximation," in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian and K. Ritter, eds., Academic Press, New York, p. 447-464, 1970.

Barrodale, I., and F. D. K. Roberts, "An Improved Algorithm for Discrete ℓ_1 Linear Approximation," SIAM Journal of Numerical Analysis 10, 839-848 (1973).

Barrodale, I., and F. D. K. Roberts, "Solution of an Over-Determined System of Equations in the ℓ_1 Norm," Communications of the ACM 17, 319-320 (1974).

Barsov, A. S., What is Linear Programming, D. C. Heath & Co., Boston, 1959.

Dantzig, G. D., "Maximization of a Linear Function of Variables Subject to Linear Inequalities," in Cowles Commission Monograph 13, T. C. Koopmans, ed., Wiley & Sons, New York, 1951.

Frieden, B. R., "Image Enhancement and Restoration," in Picture Processing and Digital Filtering, Springer-Verlag, New York, 1977.

Frieden, B. R., "Computational Methods of Probability and Statistics," in Topics in Applied Physics, Vol. 41, Springer-Verlag, Berlin, 1980.

Gass, S. I., Linear Programming: Methods and Applications, McGraw-Hill, USA, 1958.

Goodman, J. W., Introduction to Fourier Optics, McGraw-Hill, USA, 1968.

Helstrom, C. W., "Image Restoration by the Method of Least Squares," JOSA 57, 297-313 (1967).

Hoffman, K., and R. Kunze, Linear Algebra, Prentice-Hall, New Jersey, 1961.

Koopmans, T. C., "Activity Analysis of Production and Allocation," in Cowles Commission Monograph 13, T. C. Koopmans, ed., Wiley & Sons, New York, 1951.

Luenberger, D. G., Introduction to Linear and Non-Linear Programming, Addison-Wesley, Reading, Mass., 1973.

Pratt, W. K., Digital Image Processing, Wiley & Sons, New York, 1978.

Van de Panne, C., Methods for Linear and Quadratic Programming, North-Holland, Amsterdam, Oxford, 1975.