MTF for Photographic Films by Parameter Estimation in the Space and Frequency Domains

James H. Clark
MTF FOR PHOTOGRAPHIC FILMS BY PARAMETER ESTIMATION
IN THE SPACE AND FREQUENCY DOMAINS

by

James H. Clark

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in the School of
Photographic Arts and Sciences in the
College of Graphic Arts and Photography
of the Rochester Institute of Technology

June 1978

Author __________ James H. Clark __________
Photographic Science and Instrumentation

Certified by __________ Edward M. Granger _________
Dr. Edward M. Granger
Thesis Advisor

Accepted by __________ John F. Carson _________
Prof. John F. Carson
Coordinator, Graduate Program
MTF FOR PHOTOGRAPHIC FILMS BY PARAMETER ESTIMATION
IN THE SPACE AND FREQUENCY DOMAINS

by

James H. Clark

Submitted to the Photographic Science and Instrumentation Division in partial fulfillment of the requirements for the Master of Science degree at the Rochester Institute of Technology

ABSTRACT

A new method for determining the modulation transfer function (MTF) for photographic films is described. The MTF and corresponding line spread function are approximated by carefully chosen models which are uniquely specified by a few parametric constants. By estimating the parameters in space and frequency simultaneously, the interactions between the two domains result in improved estimates over those provided by either domain alone. A computer algorithm has been written to determine the parameter constants that define the best-fit model for the emulsion MTF.
ACKNOWLEDGEMENTS

The author expresses his appreciation to his thesis advisor, Dr. Edward M. Granger of the Eastman Kodak Company, for suggesting the original topic and for providing guidance at every subsequent step of the project.

In addition, the author wishes to thank Mr. Roland W. Porth of the Rochester Institute of Technology for many fruitful discussions and suggestions.

Finally, special thanks to Kary Clark, who spent many hours working with the author to complete this project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>HISTORICAL BACKGROUND</td>
<td>4</td>
</tr>
<tr>
<td>THEORETICAL BACKGROUND</td>
<td>13</td>
</tr>
<tr>
<td>HYPOTHESES</td>
<td>23</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>38</td>
</tr>
<tr>
<td>DISCUSSION</td>
<td>41</td>
</tr>
<tr>
<td>RECOMMENDATIONS</td>
<td>63</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>66</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>68</td>
</tr>
<tr>
<td>ADDITIONAL REFERENCES</td>
<td>75</td>
</tr>
<tr>
<td>APPENDIX A: MAIN PROGRAMS</td>
<td>79</td>
</tr>
<tr>
<td>EMODX</td>
<td>80</td>
</tr>
<tr>
<td>ERRCON</td>
<td>91</td>
</tr>
<tr>
<td>GAUSS</td>
<td>95</td>
</tr>
</tbody>
</table>
APPENDIX B: SUBROUTINES ... 104

CONTOUR3 ... 105

EDGES ... 107

FFT24 ... 108

FTLIST2 ... 113

FTLIST3 ... 114

INVFFT ... 115

PLOT2 ... 116

RANDNO ... 118

SORT ... 119

UNSORT ... 120
LIST OF FIGURES

Figure 1: Modulation Transfer Function
Figure 2: Point Image, Point Spread Function
Figure 3: Line Image, Line Spread Function
Figure 4: Edge Image, Edge Response Function
Figure 5: Optical Transfer Function
Figure 6: Displacement of an even function introduces odd components.
Figure 7: Trace of an edge showing adjacency effects.
Figure 8: Cumulative Gaussian and its first two derivatives.
Figure 9: Simple model for edge with adjacency effects.
Figure 10: The effects of parameters for controlling the height, h, the width, w, and the position, p, of a function.
Figure 11: Mean Square Error versus parameter value for a single-parameter model.
Figure 12: Partial derivative of MSE with respect to the parameter, w, versus the value of w.
Figure 13: Calculation scheme for Newton's method.
Figure 14: Mean square error surface in space.
Figure 15: Mean square error surface in frequency.
Figure 16: Total mean square error surface.
Figure 17: The mean MTF is greater than or equal to the signal.
Figure 18: Ratio of the standard deviation in computed MTF to mean MTF, (S/M), as a function of frequency for different signal-to-noise ratios (SNR).

vi
Figure 19: Microdensitometer trace of a noisy image.
Figure 20: Discrete edge data from microdensitometer trace.
Figure 21: Synthetic edge data with SNR = 24.
Figure 22: Edge trace data in density.
Figure 23: Edge trace data in transmittance.
Figure 24: Edge trace data in relative exposure.
Figure 25: Original spread function data after centering.
Figure 26: Models for line spread function and modulation transfer function.
Figure 27: Comparison of measured line spread function and best-fit model.
Figure 28: Comparison of measured OTF data and MTF model.
Figure 29: Computed modulation transfer function.
Figure 30: MTF from three different traces across the same edge.
Figure 31: Comparison of mean estimated MTF and manufacturer's data.
INTRODUCTION

Since its first applications to the field of optics in the 1940's, modulation transfer theory has gradually gained acceptance and is now one of the most important tools available for the evaluation of imaging systems. A principal reason for this usefulness is the possibility of computing the modulation transfer function (MTF) for an entire imaging system by simply multiplying the MTF's of the individual components. In photography, where one or more photo-sensitive materials are used to produce the final image, it has thus become necessary to determine the transfer function of the emulsion layers.

In his extensive pioneering work on the subject, Schade discussed the difficulty that arises because the photographic response is inherently non-linear. Neither density nor transmittance is linearly related to exposure, and the optical properties of the unexposed crystal structure of the emulsion are quite different from its developed grain structure. Lamberts redefined the MTF in terms of Frieser's "effective exposure" in the emulsion and found it essentially independent of exposure level and development conditions except for adjacency effects. These effects are nearly always present
to some extent however, so Kelly proposed a chemical transfer function\textsuperscript{7} to describe the influence of development and exposure conditions, including adjacency effects. Defined in terms of effective exposure in the emulsion, the MTF provides the user with an excellent basis for comparing the potential imaging capabilities of different photographic films. Its use has become so wide-spread that film manufacturers now routinely report MTF data along with other specifications for their products.\textsuperscript{8-11}

The MTF is a function of spatial frequency and is difficult to measure directly. It can be derived, however, from either of its counterparts in the spatial domain, the spread function or the edge response. But data from physical experiments is never completely error-free, and measurements made from emulsions are complicated by photographic grain noise. The problem is to estimate the transfer function in one domain from noisy records in the other. The methods currently used to compute MTF's all involve some degree of estimation and smoothing to minimize error. Unfortunately, these attempts to separate the signal from the noise inevitably result in some degradation of the signal.

Whether smoothing is accomplished by convolution with an appropriate function, by regression on a suitable model, or by other schemes, the object is to extract from the data
the best estimate of the true signal. Different techniques, however, result in different estimates, and each may have its advantages. The usual practice in MTF determination is to apply all of the smoothing techniques to the data in either the spatial domain\textsuperscript{12} or the frequency domain,\textsuperscript{13} but not both. The purpose of this research is to develop a method of MTF determination wherein the signal is estimated in the two domains simultaneously, while an interactive feedback mechanism drives the system towards the optimum solution. The functions are approximated by carefully chosen models, which in turn, are uniquely determined by certain parameters. To estimate these parameters a computer algorithm has been written which, even for low signal-to-noise ratios, converges to a "best fit" model for the emulsion MTF.
The scientific study of photographic images on a macroscopic scale began nearly a century ago with the classic work of Ferdinand Hurter and Vero Driffield. It was relatively recently, however, that significant progress was made in the analysis of images on a microscopic scale, and this work had its origins outside the field of photographic science. Working in optics and in electronic communications, Duffieux, Rose, Schade, Shannon and Wiener were among those who first applied Fourier analysis and transfer theory to the problems of information recording and handling. Schade carried the techniques into his studies of motion picture systems, and applied them to the photographic emulsion. Following this lead, significant advances in the analysis of photographic images were made by Fellgett, Linfoot, Jones, Lamberts, Zweig and many others during the 1950's. Nearly everyone used his own terminology when discussing transfer theory. Jones listed at least twelve different names for the transfer function, such as: aperture response, Fourier spectrum, transmission factor and sine-wave response. In 1961, acting on a proposal by Dr. G.C. Higgins, The International Commission for Optics, Subcommittee for Image Assessment Problems, issued recommendations for
standardizing the nomenclature for transfer functions. The various terms are defined and illustrated in the next section of this paper. Adoption of the new terminology was not instantaneous. Some authors, whose work was well known before the recommendations were issued, changed over gradually, sometimes using old and new terms in the same paper. Eventually, however, the proposed nomenclature was universally accepted.

If there were many different names for the modulation transfer function, equally many methods were devised to measure it. In his review of MTF measurement methods, Dainty grouped the various techniques into four classifications: sine-wave methods, Fourier transformation of the line spread function, coherent light processing methods, and calculation from scattering measurements. These categories are rather broad and overlap to some extent, but will serve the purposes of this survey.

Sine-wave techniques involve exposing the emulsion to sinusoidally varying irradiance distributions of known spatial frequency and modulation. The processed image is scanned with a microdensitometer, and the data is taken back through the characteristic curve to determine the effective exposure distribution. The modulation transfer factor for a given frequency is the ratio of the effective
exposure modulation to the input exposure modulation.

The main problem with these techniques is the production of sinusoidal exposures of known modulation. One popular method is to form an image in the emulsion of a varying area test target in such a way that one dimension on the target is integrated or "smeared." This has been accomplished with a cylindrical lens,⁵ ³¹ ³³ ³⁴ a slit pupil,³⁵ and other methods.³⁶ ³⁷ Some workers³⁸ ³⁹ have used rotating polarizers on either side of a half-wave plate to produce a time varying irradiance in the image of a narrow slit. This is scanned across the emulsion to provide exposures varying with position. Other scanning techniques⁴⁰-⁴² have been used, and sinusoidal exposures have also been produced with interference fringes,³⁶ ⁴³-⁴⁶ and defocused Moiré patterns.⁴⁷ Since granularity degrades the accuracy of measurement, a multiple slit can be fitted to the microdensitometer to scan several cycles simultaneously, thereby improving the signal-to-noise ratio.³⁴ An "MTF Meter" has been described which scans sinusoids in a way that minimizes problems associated with conventional microdensitometry.⁴⁸

To avoid making sine-wave test objects, square-wave targets have been substituted. A modulation transfer factor, \( M'(f) \), is determined from the image of a square-wave chart of the type described by Sayce.⁴⁹ The true MTF, \( M(f) \) is
calculated from a formula credited to Coltman: \[ M(f) = (\pi/4)[M'(f) + M'(3f)/3 - M'(5f)/5 + ...] \] (1)

The conditions of exposure, scanning, and data handling must be consistent to reduce variability in the results, but even so this method may be unsatisfactory for other reasons.\[ \text{32} \]

For a linear imaging system the MTF is the modulus of the Fourier transform of the line spread function, \( h(x) \).\[ \text{30} \]

\[ M(f) = \left| \int_{-\infty}^{+\infty} h(x) \exp(-i2\pi fx) dx \right| \] (2)

Optical scattering in the emulsion is a linear process, but adjacency effects cause the effective exposure distribution to be a non-linear response function. As a result, there is not a unique MTF. However, if adjacency effects are minimized, different measurement methods can produce nearly the same result.

Practical considerations make it more common to obtain the line spread function from the image of an edge than from the image of a line. Ideal line images are difficult to produce and measure. Edges, on the other hand, are common objects, and exposures can be made in the field as well as in the laboratory. The processed image is then scanned with a microdensitometer. To convert from instrument response to effective exposure, the use of witchcraft has been suggested,\[ \text{51} \]
but the usual practice is to work through the characteristic curve. The resultant edge response function is differentiated to obtain the line spread function, and then Fourier transformed to produce the MTF. Many variations of this basic scheme have been used, and somewhere along the line, provision must be made for removal of noise and instrument effects. Jones\textsuperscript{12} devised a technique employing a digitizing microdensitometer and a computer to perform all of the required steps automatically, including the non-linear conversion from transmittance to effective exposure. The edges and sensitometric exposures are scanned and read directly into the computer. A microdensitometer correction function is convoluted with the edge data, then a polynomial least squares fit to the transmittance-exposure curve is used in the conversion to effective exposure. Differentiation and smoothing are done simultaneously by convolution with a special function. This yields the noise-free line spread function which is Fourier transformed numerically to give the MTF. Problems from truncation\textsuperscript{52,53} of the data set appear when numerical methods are applied, and photographic grain noise distorts the phase of the measured transfer function, introducing uncertainty into the computed MTF.\textsuperscript{54,55}

Alternative schemes have been developed which avoid the need for numerical differentiation and Fourier transformation. Scott, Scott, and Shack\textsuperscript{56} described a method whereby
they smooth the edge trace by hand and extract first the square-wave response, then the sine-wave response by simply taking finite sums and differences. The MTF of the micro-densitometer is divided out at the end. Tatian\textsuperscript{57} expressed the transfer function as a trigonometric series whose coefficients are proportional to sampled values of the edge response function, and Barakat\textsuperscript{58} derived the MTF directly from the edge response by inversion of a Fredholm integral equation of the first kind.

The coherent light processing methods of determining the MTF of a photographic material either study the diffraction pattern of a periodic image in the emulsion, or analyze the power spectrum of an image produced by exposure to a laser speckle pattern.\textsuperscript{59} Ooue\textsuperscript{60} has described the use of a Fraunhofer diffractometer to compare the image of a square-wave grating to the grating itself. The modulation transfer factor for a particular frequency is related to the ratio of intensities of one of the diffraction orders. Although the measurement is rapid and relatively unaffected by granularity, the method assumes a linear relationship between amplitude transmittance and exposure. This is valid only for small differences in transmittance.\textsuperscript{32} Diffraction techniques are useful for measuring the performance of high resolution materials at very high frequencies.\textsuperscript{61,62} In fact, results have been claimed\textsuperscript{63} at spatial frequencies out to
Calculation of the MTF from scattering measurements is of theoretical interest, but not very practical. The method is based on physical characteristics of the emulsion, such as thickness, light absorption, and scattering. Frieser\textsuperscript{64,65} worked on this problem, and more recently, Monte Carlo techniques have been applied.\textsuperscript{66-68} It is necessary to know the probability that a photon will be absorbed upon collision with a silver halide crystal, as well as the probability distributions for scattering angles and distances between collisions. Although results have not closely matched those from other methods,\textsuperscript{67} these techniques may be useful for predicting the effect, for example, of adding acutance dyes to an emulsion.

Regardless of the method of MTF measurement, experimenters have searched for decades to find an analytical function which closely fits their data. Since the MTF and the line spread function are a Fourier transform pair, it is equally useful to find a mathematical model for the latter.

As early as 1935, Frieser\textsuperscript{6} fit an exponential curve to the line spread function, $h(x)$, which can be expressed as:

$$h(x) = (2.3/k)\exp(-4.6|x|/k)$$

(3)
Finding that a single parameter was inadequate at higher frequencies, he expanded his model\textsuperscript{64} to an equivalent form of:

\[ h(x) = (2.3\beta/k_1)\exp(-4.6|x|/k_1) + (2.3(1-\beta)/k_2)\exp(-4.6|x|/k_2) \]  

(4)

Citing problems with the above formulas, Lewis\textsuperscript{69} proposed a single parameter, fourth-power cosine function:

\[ h(x) = 2\cos^*[\arctan(x/t)]/\pi t \]

(5)

\[ = 2/\pi t[1 + (x/t)^2]^2 \]

(6)

The author speculated on the possible significance of \( t \) being approximately equal to the emulsion thickness. Gaussian spread functions have been discussed by several workers, including Schade\textsuperscript{70} and Hendeberg:\textsuperscript{71}

\[ h(x) = (1/\sqrt{2\pi} s)\exp(-x^2/2s^2) \]

(7)

Jones\textsuperscript{72} suggested a model which corresponds to an exponential MTF:

\[ h(x) = a/[\pi(a^2 + x^2)] \]

(8)

And Sayanagi\textsuperscript{73} tried a first order Bessel function of the second kind:

\[ h(x) = (1/\pi a)(|x|/a)K_1(|x|/a) \]

(9)

In general, an effort has been made to minimize the number of variables, but computers have made practical the use of
multiple-parameter models. For example, Smith$^{74}$ expanded the line spread function into a series of Hermite polynomials and then integrated term by term to get an expression for the edge response function. A least squares estimation of the coefficients adjusted the model to fit the microdensitometer data.

The list of MTF measurement methods presented here is by no means to be considered exhaustive. It is, however, representative of the work that has been done in the past thirty years. Except for light scattering calculations, all of the techniques involve deriving the function from grainy photographic records. Methods for removing the noise have ranged from drawing a smooth line over the microdensitometer trace by hand,$^{56}$ to spatially filtering the diffraction pattern of the image.$^{62}$ But in none of these works have the spread function and the transfer function been estimated simultaneously, as is done in this project.
THEORETICAL BACKGROUND

The modulation transfer function of an incoherent image-forming system indicates the extent to which the modulation of a sinusoidally varying irradiance distribution is changed by image spreading as a function of spatial frequency. (Figure 1) Image spreading can be represented by the point spread function, which is the two-dimensional irradiance distribution in the image of an idealized point

Figure 1
radiation source. (Figure 2) Since the photographic emulsion is statistically isotropic, its point spread function is radially symmetric and can therefore be integrated along any axis to yield the one-dimensional line spread function, with no loss of information. (Figure 3) Consider now a semi-infinite planar source of radiation, bounded on one edge by a perfectly straight line. This source can be decomposed into an infinite array of line sources, each parallel to the edge and each imaged by the photographic material as the line spread function. Since the total irradiance for any line in the image is the sum of the contributions from the spread functions of all the lines in the source, the irradiance distribution in the image of the half-plane is simply the cumulative integral of the line spread function, and is called the edge response function. (Figure 4)
LINE IMAGE

LINE SPREAD FUNCTION

Figure 3

EDGE IMAGE

EDGE RESPONSE FUNCTION

Figure 4
In an isoplanatic linear imaging system the irradiance distribution in the image can be represented as a mathematical convolution of the object distribution with the system spread function.\textsuperscript{75} If the object radiance varies sinusoidally along one dimension, then the image will be a sinusoid of the same frequency; only the modulation and phase are affected by the system spread function.\textsuperscript{26} The modulation transfer function (MTF) and the phase transfer function (PTF) show the extent to which the modulation and phase of the input are modified as a function of spatial frequency. The MTF and PTF are the modulus and phase respectively of the complex optical transfer function (OTF). The OTF is the Fourier transform of the line spread function, $h(x)$.\textsuperscript{30} If $H(f)$ is the OTF, then

$$H(f) = \int_{-\infty}^{\infty} h(x) \exp(-i2\pi fx) \, dx \quad (10)$$

Applying Euler's formula,\textsuperscript{76} equation 10 becomes

$$H(f) = H_c(f) - iH_S(f) \quad (11)$$

where

$$H_c(f) = \int_{-\infty}^{\infty} h(x) \cos(2\pi fx) \, dx \quad (12)$$

and

$$H_S(f) = \int_{-\infty}^{\infty} h(x) \sin(2\pi fx) \, dx \quad (13)$$

Equations 12 and 13 are the Fourier cosine and sine transforms of the line spread function. If $M(f)$ is the MTF and $\phi(f)$ is the PTF, then

$$M(f) = [H_c(f)^2 + H_S(f)^2]^{\frac{1}{2}} \quad (14)$$
and
\[ s(f) = \arctan[-H_g(f)/H_c(f)] \]  
\hfill (15)

These relationships are shown graphically in Figure 5.

OPTICAL TRANSFER FUNCTION

Figure 5

It is clear that if \( H_s(f) = 0 \) for all \( f \), then \( s(f) = 0 \) and the OTF is real, not complex. \( H_s(f) \) will be zero if the integrand of equation 13 is odd. A function, \( g_o(x) \), is odd if

\[ g_o(-x) = -g_o(x), \quad -\infty < x < +\infty \]  
\hfill (16)

A function, \( g_e(x) \), is even if

\[ g_e(-x) = g_e(x), \quad -\infty < x < +\infty \]  
\hfill (17)
Products of symmetric functions are also symmetric:

\((\text{odd})(\text{odd}) = \text{even}, (\text{even})(\text{even}) = \text{even}, \text{ and } (\text{even})(\text{odd}) = \text{odd}\).

Since the sine function is odd, \(h(x)\) must be even to make \(H_s(f)\) equal zero. This is the case for photographic emulsions which have symmetric spread functions. Thus an emulsion OTF is real and equal to the MTF, an important property which can be used when estimating the function.

Because of the difficulties involved with producing and scanning point and line images, it is frequently more practical to obtain image structure data from edge traces, than from spread functions.\(^{56}\) The edge response function, \(e(x)\), can be differentiated to get the line spread function, \(h(x)\), and then transformed to yield the OTF. Alternately, the Fourier Derivative Theorem\(^{78}\) permits \(e(x)\) to be transformed directly and then multiplied by \((i2\pi f)\) to give the OTF:

\[
H(f) = (i2\pi f)F[e(x)]
\]  

(18)

where \(F[\ ]\) is the Fourier transform operator.

The edge response function, like any real function can be decomposed into its even and odd parts. For a general function, \(u(x)\), the even and odd components are given by

\[
u_e(x) = \frac{1}{2}[u(x) + u(-x)]
\]

(19)

\[
u_o(x) = \frac{1}{2}[u(x) - u(-x)]
\]

(20)
Note that \( u_e(x) \) and \( u_o(x) \) satisfy equations 16 and 17 respectively, and that

\[
u(x) = u_e(x) + u_o(x) \tag{21}\]

The edge response is the sum of an odd function with the same shape as the edge, and a constant which represents the average value of the curve. Subtracting the constant leaves a function which is odd only when properly centered about the origin. In general, displacement of a function destroys its symmetry. Shifting introduces odd components into an even function, and even components into an odd function. (Figure 6) Since the edge trace data from a microdensitometer do not include the location of the center of the function,
minimization of the even part will properly position the edge for subsequent manipulation of the data in the computer.

Displacement of a function in space corresponds to changing the phase of its transform in proportion to the frequency. If the transform is imaginary, a real component will appear, and vice versa. Therefore, minimizing the unwanted component of the transform provides an alternative method of centering a symmetric or nearly symmetric function in the space domain.

Up to this point it has been assumed that the functions in question were continuous. However, for analysis with a digital computer, the functions must be sampled at finite intervals and represented as a set of discrete data points. If the values of a function vary gradually from one point to the next, then it would seem possible, at least to some degree of approximation, to recover the original function with relatively few data points. If, on the other hand, the function varies rapidly, then many samples will have to be taken very close together for the data to reasonably represent the function. Rapid changes in a function are due to high-frequency components. If a frequency, $f_c$, exists, beyond which the Fourier transform is zero, then the function is band limited and $f_c$ is the cut-off frequency.
The Sampling Theorem states that a band-limited function can be fully and accurately described by a properly chosen sample set. The prescribed interval between samples must not be greater than $1/2f_c$, which corresponds to the Nyquist sampling frequency. In most practical situations there exists an effective cut-off frequency beyond which spectral contributions are negligible. But if the transform never goes to zero absolutely, choosing a practical cut-off frequency results in truncation of the spectrum. This is equivalent to convolution with a sinc function in the spatial domain, where $\text{sinc}(x) = \sin(\pi x)/\pi x$. When a function is not truly band-limited, it is necessary to estimate the amount of error introduced by truncation of the transform.

The simple fact that only a finite number of samples can be taken for analysis, means that functions are usually truncated in the spatial domain as well. This produces error in the computed MTF, especially at low frequencies. The effects of truncation, and correction techniques have been examined by Rabedeau and Tatian.52 53

Another problem arises when, for example, the physical limitations of the measuring equipment do not allow sampling at the Nyquist frequency. Under-sampling results in high-frequency components of the transform appearing at the lower frequencies, and the effect is known as "aliasing."
For sampling spread functions, guide lines have been published which suggest sampling rates for limiting the absolute error in the computed MTF to less than 0.005.\textsuperscript{83}

The theoretical and practical considerations introduced here are necessary for understanding the experimental approach used in this project. The relevance of these concepts to this particular problem will be discussed in the following section.
HYPOTHESES

The purpose of this research is to develop a new method of MTF determination by fitting mathematical models, in the space and frequency domains simultaneously, to the functions which characterize the image structure of the photographic emulsion. These include the edge response function, the line spread function, and the MTF.

Consider the edge trace shown in Figure 7, which is typical of many film-developer combinations showing mild adjacency effects. To a first approximation, this data could be represented by a cumulative Gaussian function or some similar curve as shown in Figure 8a. A better fit might result from a linear combination of this curve and its second derivative, Figure 8c. Let \( g_0(x) \) represent the first function and \( g_n(x) \), its \( n \)th derivative. If \( e(x) \) is the edge response function and \( m(x) \) is the mathematical model, then so far,

\[
m(x) = a_0 g_0(x) + a_1 g_1(x) + a_2 g_2(x)
\]

where coefficient \( a_1 \) is zero and \( a_2 \) is negative, as shown in Figure 9. By continuing in this way, a model is constructed
Figure 7: Trace of an edge showing adjacency effects.

Figure 8: Cumulative Gaussian and its first two derivatives.
Figure 9: Simple model for edge with adjacency effects.

which approximates the true function:

\[ e(x) = \sum a_i g_i(x) \]  \hspace{1cm} (23)

The coefficients, \( a_i \), are used here to change the height of the model. It may also be necessary to alter the width of the model, or its position along the axis. Figure 10 illustrates the effects of parameters for controlling the height, width and position of a function.

The parameters which determine the shape of the spread function model in the space domain, also appear in the frequency domain MTF model. In fact, the two models are an exact Fourier transform pair, and conform to all of the rules of transform analysis.\(^{84,85}\) The Fourier theorems which apply directly to this project are summarized in Table 1.
Figure 10: The effects of parameters for controlling the height, \( h \), the width, \( w \), and the position, \( p \), of a function. In each case \( h = \frac{1}{2} \), \( w = 2 \), \( p = \frac{1}{2} \).

Table 1 Properties of the Fourier Transform

<table>
<thead>
<tr>
<th>Property</th>
<th>Space Domain</th>
<th>Frequency Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform Pair</td>
<td>( g(x) )</td>
<td>( G(f) )</td>
</tr>
<tr>
<td>Similarity</td>
<td>( g(x/a) )</td>
<td>(</td>
</tr>
<tr>
<td>Shift</td>
<td>( g(x-a) )</td>
<td>( \exp(-i2\pi af)G(f) )</td>
</tr>
<tr>
<td>Linearity</td>
<td>( a \cdot g(x) + b \cdot h(x) )</td>
<td>( a \cdot G(f) + b \cdot H(f) )</td>
</tr>
<tr>
<td>Convolution</td>
<td>( g(x) \ast h(x) )</td>
<td>( G(f) \cdot H(f) )</td>
</tr>
<tr>
<td>Derivative</td>
<td>( \frac{d[g(x)]}{dx} )</td>
<td>( (i2\pi f)G(f) )</td>
</tr>
</tbody>
</table>
The first spread function model to be investigated is the Gaussian curve. It was chosen because of its many convenient properties which include having Gaussian derivatives and Fourier transform. The latter is given below.

\[ \exp[-\pi x^2] \overset{F}{\rightarrow} \exp[-\pi f^2] \] (24)

It is a function which is easily manipulated mathematically, and thus is well suited as a basis on which to build the parameter estimation program. In addition, past workers have found it to be a reasonable model for some emulsion spread functions.

A Gaussian curve can be completely described by three parameters: height, width and position. If \( m(x) \) is the spread function model, then

\[ m(x) = h \cdot \exp[(x-p)^2/w^2] \] (25)

By applying the appropriate Fourier transform theorems to equation 24, one obtains the MTF model, \( M(f) \).

\[ M(f) = (hw\sqrt{\pi})\exp[-(\piwf)^2]\exp[-i2\pi pf] \] (26)

Thus both models are fully described by the same three parameters. Turning that idea around is the key to this entire project. That is, a single set of unknown parameters can be estimated in either of two domains, or in both domains simultaneously.
All of the parameter values are estimated by a method of least squares. This means minimizing the mean square error (MSE) between the model, \( m(x) \), and the data, \( u(x) \):

\[
MSE = \frac{\sum[m(x_i) - u(x_i)]^2}{N}
\]

(27)

where \( N \) is the number of data points. Least square regression techniques work on the assumption that the mean square error is clearly at a minimum for the correct value of the parameter. A plot of MSE versus the parameter value may resemble Figure 11. The minimum can be found by locating the point at which the slope of the curve equals zero. Let \( g(w) \) equal the partial derivative of the MSE with respect to the parameter, \( w \). A typical plot of \( g(w) \) versus \( w \) is shown in Figure 12. The desired value of the parameter is at the point where the curve crosses the axis.

For linear coefficients, such as \( h \) in equation 25, the partial derivative expression can be set equal to zero and solved in closed form for the parameter value in question. Non-linear factors, like \( p \) and \( w \) in equation 25, usually require an iterative solution. One powerful numerical technique for finding the roots of non-linear equations is Newton's method. Briefly, the technique is to rewrite the equation as a function of one variable and start with an estimate of that variable not far from the root. Then extend the tangent to the curve at that point until it meets
Figure 11: Mean Square Error versus parameter value for a single-parameter model.

Figure 12: Partial derivative of MSE with respect to the parameter, \( w \), versus the value of \( w \).
the axis of the abscissa. The value of the parameter at the intersection is the next estimate. Repeat the process until successive iterates are sufficiently close, or until the value of the function is sufficiently near zero.

The calculation scheme for Newton's method is illustrated in Figure 13, and works as follows:

$$\tan(\alpha) = g'(w_1) = \frac{g(w_1)}{(w_1-w_2)} \quad (28)$$

where $g'(w_1)$ is the derivative of $g(w)$ evaluated at $w_1$. By simply rearranging the terms,

$$w_2 = w_1 - \frac{g(w_1)}{g'(w_1)} \quad (29)$$

Figure 13: Calculation scheme for Newton's method. $w_2 = w_1 - \frac{g(w_1)}{g'(w_1)}$
The algorithm can be repeated indefinitely in its more general form:

\[ w_{n+1} = w_n - \frac{g(w_n)}{g'(w_n)} \tag{30} \]

For multiple-parameter models, it is necessary to compute partial derivatives of the mean square error expression with respect to each of the parameters in question, and then create separate algorithms for estimating each of the unknowns.

As an example of what is involved in estimating just one non-linear factor in two domains simultaneously, consider \( w \), the width scaling parameter, in the Gaussian models described earlier. The mean square error is a function of all three parameters:

\[
\text{MSE}(h, p, w) = \frac{\sum (m(x_i) - u(x_i))^2}{N} + \frac{\sum (M(f_i) - U(f_i))^2}{N} \tag{31}
\]

where \( m(x_i) \) and \( M(f_i) \) are given by equations 25 and 26, \( u(x_i) \) is the spread function data, and \( U(f_i) \) is the discrete Fourier transform of \( u(x_i) \). Let \( g(w) \) equal the partial derivative of \( \text{MSE}(h, p, w) \) with respect to \( w \). After substituting for \( m \) and \( M \), squaring, differentiating and rearranging, \( g(w) \) emerges as follows:
Next Newton's method is employed to find which value(s) of $w$ will make $g(w)$ equal zero. Ordinarily this would require taking the derivative of $g(w)$ for use in equation 30. However, in cases such as this it is desireable to modify Newton's method by using an approximation to the derivative:

$$g'(w) \approx \frac{g(w+\Delta w) - g(w)}{\Delta w}$$  \hspace{1cm} (33)$$

where $\Delta w$ is a small increment in $w$. This results in a new algorithm for computing successive estimates of $w$:

$$w_{n+1} = w_n - \Delta w/g(w_n + \Delta w) - 1]$$  \hspace{1cm} (34)$$

Each time a new estimate for one of the parameters is computed, this value is used to re-evaluate the other parameters, and so on until no improvement beyond a predetermined level of accuracy is achieved. Thus the entire parameter estimation scheme is repeated many times, and each repetition calls for many iterations of the modified Newton's method. Each of these iterations requires the partial derivative expression (equation 32) to be evaluated twice, and both of those calculations involve the evaluation and summation of complicated expressions for each of the $N$ data points in the
two domains.

Clearly this task is for a digital computer, and so all of the steps described above were programmed in Xerox Extended Fortran IV for execution on R.I.T.'s Sigma - 9 computer. A listing of the program used to fit the Gaussian model is included in Appendix A under the title "Gauss." The results of this program will be detailed in the next section of this paper.

Up to this point, the discussion has centered around fitting a mathematical model to some experimental data, under the assumption that this data is an accurate representation of the signal being measured. Unfortunately, physical measurements always contain some error or noise, which may have been acquired at any point in the experiment. Error can be systematic or entirely random, additive or multiplicative, signal-dependent or independent, or any combination of the above. In photography, many factors can degrade the quality of images recorded on film, but one of the most important is grain. Photographic emulsion grain is a source of signal-dependent additive noise which is inseparable from the signal because, in fact, it is the signal. Without grain, there would be no image. Since grain noise can not be eliminated, its effects must be considered whenever information is extracted from images on film.
Let $n(x)$ represent the noise which, when added to the true signal, $s(x)$, gives the measured values of the data, $u(x)$. Thus,

$$u(x) = s(x) + n(x)$$  \hspace{1cm} (35)$$

The signal may be either the edge response or the line spread function. In the frequency domain, the signal is either the OTF or the MTF, and the corresponding formula is

$$U(f) = S(f) + N(f)$$  \hspace{1cm} (36)$$

The problem, of course, is that the true values of the signal and noise are unknown. Actually, one assumes some apriori knowledge of the nature of the signal. Otherwise there would be no reason to suspect that it differs in any way from the measured data. Theoretical predictions and practical experience tell what to expect and allow intelligent choices to be made among possible mathematical models to represent the underlying signal. How well the model fits is determined for the most part by the choice of model, and by the signal-to-noise ratio of the data.

A considerable improvement in the signal-to-noise ratio is made possible by taking advantage of the isotropy of photographic emulsions. This characteristic results in symmetric edge response and spread functions. The measured data, however, will rarely exhibit symmetry. Assume some
data have been experimentally determined for the spread function, \( h(x) \):

\[ u(x) = h(x) + n(x) \]  \hspace{1cm} (37)

By use of equations 19 and 20, \( u(x) \), \( h(x) \) and \( n(x) \) can be decomposed into their even and odd parts. Since \( h(x) \) is even and has no odd component, the odd part of \( u(x) \) is entirely noise. That is,

\[ u_e(x) = h(x) + n_e(x) \]  \hspace{1cm} (38)

\[ u_o(x) = n_o(x) \]  \hspace{1cm} (39)

Being purely random, the noise is approximately equally comprised of even and odd components. Thus by removing the odd part of the spread function data, the signal-to-noise ratio is significantly improved.

In practice, it is not necessary to actually remove the odd component of the spread function data. It is possible to carry out all of the calculations of the parameter estimation routine in such a way that the odd noise has no effect at all.

Assume that the center of the spread function has been located and that the entire data set has been shifted to locate the center at \( x = 0 \). Consider now the two data points
at \( x = \pm a \).

\[
\begin{align*}
    u(a) & = h(a) + n(a) \quad (40) \\
    u(-a) & = h(-a) + n(-a) \quad (41)
\end{align*}
\]

After the noise has been separated into its even and odd parts, equations 16 and 17 are applied to yield:

\[
\begin{align*}
    u(a) & = h(a) + n_e(a) + n_o(a) \quad (42) \\
    u(-a) & = h(a) + n_e(a) - n_o(a) \quad (43)
\end{align*}
\]

Finally, the arithmetic mean of \( u(a) \) and \( u(-a) \) is computed:

\[
\overline{u(a)} = \frac{[u(a) + u(-a)]}{2} = h(a) + n_e(a) \quad (44)
\]

All that remains is the signal plus even noise. Since the model, \( m(x) \), and \( u(x) \) are both even functions, the mean square error can be determined using only the positive values of \( x \), and will reflect only the even noise contained in the original data.

When the spread function data are Fourier transformed to become the OTF data, the odd noise becomes a pure imaginary function. The signal plus even noise remains real and even. Thus when the MSE is computed in the frequency domain, only the real part of the OTF data should be
considered. Using the modulus only inflates the estimate, especially where the signal is low.

In summary, the working hypothesis of this research project is that buried within noisy experimental data for an emulsion spread function lies a well-behaved function which can be closely approximated by a properly chosen mathematical model. This model can be fully described by as few as two or three parametric constants. The same parameters determine the shape of the model's Fourier transform, and thus there are effectively two data sets from which to extract estimates of the parameters. The advantage is that the noise in each domain affects the estimate in different ways, because the error at a given point in one domain is distributed over all the points in the other domain. The models are fit to the data by a method of least squares, and where non-linear coefficients are involved, Newton's method is employed. The development of a computer program to perform all of the tasks detailed in this chapter is described in the following section.
PROCEDURE

The creation of a complete computer program for fitting mathematical models to experimental data was a gradual process. Many separate input, output and computational tasks had to be performed, which for convenience and flexibility, were isolated and programmed as individual subroutines. During the course of the work, fourteen such subroutines were devised for performing particular operations. Only one program by another author was used. FORTRAN subroutine FFT24 by R.P. Brumbach was called upon to compute Fast Fourier Transforms. Twenty additional programs were written to test subroutines, compare different computational schemes, simulate conditions of use, and study preliminary results. These provided the foundation upon which the project's primary work was based.

Ultimately the research produced three main computer programs. Of these, GAUSS and EMODX fit mathematical models to noisy spread function data by parameter estimation in two domains. The third, ERRCON, computes the error surfaces in each domain as a function of the height and width of a Gaussian model, plots the surface contours for the mean square error in space, in frequency and in both domains
together, and locates their respective minima. ERRCON was run at varying signal-to-noise levels, and used to determine whether the minimum error actually occurred at the correct parameter values, and whether any greater reliability was predicted by estimating parameters in two domains.

GAUSS was written first, for the reasons stated earlier. During its development, there were frequent problems of non-convergence with the parameter estimating routine, so ERRCON was created to study the nature of the error surface and to show whether it was theoretically feasible to locate the minimum. GAUSS was then extensively revised many times until it could successfully fit a mathematical model to artificial spread function data. At that point several portions were rewritten to change from a Gaussian to an exponential model. This version became EMODX, which continued to evolve by having several new sections added and others modified.

In its final form, EMODX reads in edge trace data in density, converts to transmittance, computes via the linear portion of the micro-characteristic curve the effective exposure distribution, and differentiates the edge to produce the line spread function (LSF) data. The program plots out the edge data in density, transmittance and relative exposure as a function of position. The center of the spread
function is then located and the entire data set is shifted to put it at the center of the space domain. The LSF data is plotted out and then Fourier transformed to produce the OTF data. The numerical values of both functions are printed out for each point in the two domains. After making initial estimates of the parameters, the program fits the exponential model and its transform to the two data sets simultaneously. The values of these models at every point in each domain are calculated, normalized, tabulated and plotted with the original data for visual comparison. Finally the MTF values are listed and plotted for the appropriate range of frequencies. Then, if the user so desires, the program smooths the edge data in density by convolution with a narrow triangular function, and starts over at the beginning. This is repeated as many times as requested by the user. For test purposes, the program is also capable of synthesizing its own LSF data with given parameters and signal-to-noise ratio.

Listings for programs GAUSS, ERRCON and EMODX are included in Appendix A. All of the subroutines called by the final versions of the main programs can be found in Appendix B. Discussion of the program results, including typical output from EMODX, follows in the next section of this paper.
DISCUSSION

Long before either parameter estimation program had been completed, program ERRCON was running and providing information about the nature of the error surfaces in each domain. Based on the Gaussian spread function and MTF models (Equations 25 and 26), it computed the mean square error (Equation 27) for every combination of height and width as both parameters varied a specified amount from the correct values. Essentially, ERRCON was testing the hypothesis that the best-fit model in two domains together is closer to the true function than the best fit in either domain separately. Had this not turned out to be true, or at least usually true, there would have been little reason to proceed with the project. What ERRCON showed was that the error surfaces in space, frequency and both domains together are smooth, continuous surfaces with a single distinct minimum in the area of interest, and that for all but the noisiest data, the two domain estimate was as good as or better than the result from estimating the parameters in either domain alone.

Consider Figures 14-16, which are error surface contours printed out by ERRCON. In each case, the symbol, ●, indicates
Figure 14: Mean square error surface in space.
43

**Figure 15:** Mean square error surface in frequency.
Figure 16: Total mean square error surface.
the point on the surface that results when the exact parameter values are used in the models. The symbol, $\mathbf{M}$, marks the minimum point in the surface. In each of the two domains, the estimates of the parameters that result from minimizing the mean square error are incorrect. But the errors are in different directions, and when considered together, tend to cancel. Thus the two-domain error surface is the best choice for use in estimating the true parameters.

This result is entirely consistent with what is known about the error produced by photographic grain. Grain noise is difficult to handle because it is not necessarily distributed evenly about the signal. In fact, for areas receiving little or no exposure, some grains are still developable and so the noise is entirely positive. An emulsion spread function, therefore, appears wider than it actually is because some density is measured, even beyond the practical limits of the function.

If the estimate of the spread function is too wide, then by the similarity property of Fourier transforms, the corresponding MTF will be too narrow. But noise in the frequency domain also causes a positive bias in the estimate. Figure 17 shows a signal vector of magnitude, $S$, in a complex field. Added to it is a noise vector of magnitude, $N$. The measured MTF, $U$, is the magnitude of the sum of the other
two vectors. All possible values of $U$ are defined by the distance between point $O$ and every point on the noise circle. If all phase angles for noise are equiprobable, then the mean MTF is always greater than or equal to $S$ because more than half the noise circle is outside the signal circle. It is clear that if the signal-to-noise ratio is low, the mean MTF is significantly larger than the true signal.

The low MTF estimate that results from noise in space is therefore counteracted by the high estimate due to noise in frequency, when both domains are considered simultaneously. This hypothesis is central to the project and is supported by the results of program ERRCON.
ERRCON's original purpose was somewhat less ambitious than to test the fundamental premise on which this work is based. Program GAUSS, for fitting Gaussian spread function and MTF models, was not running properly, and the source of the difficulty was unknown. Either the mathematical routines for locating the minimum error were not correctly applied, or the error surface itself had some peculiar characteristics which made those methods inappropriate. Plotting out the error surfaces seemed the most expedient way to settle the question. Having determined that the surface was smooth and free of local minima and maxima in the neighborhood of the desired point, attention was turned to making GAUSS converge towards that point. Success came not from any major breakthrough, but rather gradually, as sections were modified to give better starting values, restrict increments, and test for divergence. As soon as GAUSS was performing consistently, it was revised to fit an exponential model similar to Equation 3, and renamed EMODX.

Up to this point the program had always been run using line spread function data created internally. The advantage to using synthetic data was that the exact parameters were known, and the signal-to-noise ratio (SNR) could be set to any desired level. To give a measure of the program's performance, it was run repeatedly with different data while varying the noise level. Then for each signal-to-noise
Figure 18: Ratio of standard deviation in computed MTF to mean MTF, \( (S/M) \), as a function of frequency for different signal-to-noise ratios (SNR).

ratio, the standard deviation in computed MTF divided by the mean MTF was plotted as a function of spatial frequency. Figure 18 shows three such curves. Since the model is always 1.0 at zero frequency, the deviation there is always zero. For middle frequencies, the curves reach a plateau, and at higher frequencies they fall back to zero because ultimately the models all go to zero. For SNR below 16, the variability in computed MTF's is frequently as high as 100\%, and thus not very reliable. However, as SNR rises above 16, the MTF estimates quickly settle down.
SNR is defined here as the ratio of the peak signal to the root-mean-square noise, and so an edge trace with SNR equal to 16 is actually a very noisy trace. The distinguishing feature of this MTF estimation program is that it performs quite well with very noisy data. To demonstrate this, an edge was imaged onto Kodak 2485 High Speed Recording Film and scanned on an Ansco Model 4 Automatic Recording Microdensitometer under conditions which would accentuate the grain structure of the image. Figure 19 is an actual trace of the edge image produced. The image was scanned at three

Figure 19: Microdensitometer trace of a noisy image.
different positions along the edge, and from each trace 64 points were read at regular intervals. Figure 20 shows the discrete data as a function of position for one of the traces, and for comparison, Figure 21 is one of the synthetic edges with SNR equal to 24. All three sets of the real edge trace data were run through program EMODX. Figures 22 through 29 show a portion of the output from one such run. The MTF models for the three traces are plotted against log frequency in Figure 30, and in Figure 31 their mean is compared to the film manufacturer's data. The estimated MTF agrees closely with Kodak's curve except at low frequencies. The difference is due to the fact that the model always goes through 1.0 at zero frequency, while the published MTF curve scarcely rises above 80%. At all other frequencies, however, this fit is remarkably good, especially when one considers that the input data to the parameter estimation portion of the program looked like that shown in Equation 25.
Figure 20: Discrete edge data from microdensitometer trace.

Figure 21: Synthetic edge data with SNR = 24.
Figure 22: Edge trace data in density.
Figure 23: Edge trace data in transmittance.
Figure 24: Edge trace data in relative exposure.
Figure 25: Original spread function data after centering.
<table>
<thead>
<tr>
<th>DISTANCE (MICRONS)</th>
<th>NORMALIZED LSF</th>
<th>DISTANCE (MICRONS)</th>
<th>NORMALIZED MTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>-242.4</td>
<td>.000</td>
<td>.000</td>
<td>1.000</td>
</tr>
<tr>
<td>-234.6</td>
<td>.000</td>
<td>1.000</td>
<td>.993</td>
</tr>
<tr>
<td>-226.8</td>
<td>.000</td>
<td>2.000</td>
<td>.977</td>
</tr>
<tr>
<td>-219.0</td>
<td>.000</td>
<td>3.000</td>
<td>.941</td>
</tr>
<tr>
<td>-211.1</td>
<td>.000</td>
<td>4.000</td>
<td>.900</td>
</tr>
<tr>
<td>-203.3</td>
<td>.000</td>
<td>5.000</td>
<td>.852</td>
</tr>
<tr>
<td>-195.5</td>
<td>.000</td>
<td>6.000</td>
<td>.803</td>
</tr>
<tr>
<td>-187.7</td>
<td>.000</td>
<td>7.000</td>
<td>.746</td>
</tr>
<tr>
<td>-179.9</td>
<td>.000</td>
<td>8.000</td>
<td>.689</td>
</tr>
<tr>
<td>-172.0</td>
<td>.000</td>
<td>9.000</td>
<td>.629</td>
</tr>
<tr>
<td>-164.2</td>
<td>.000</td>
<td>10.000</td>
<td>.569</td>
</tr>
<tr>
<td>-156.4</td>
<td>.000</td>
<td>11.000</td>
<td>.503</td>
</tr>
<tr>
<td>-148.6</td>
<td>.000</td>
<td>12.000</td>
<td>.439</td>
</tr>
<tr>
<td>-140.8</td>
<td>.000</td>
<td>13.000</td>
<td>.377</td>
</tr>
<tr>
<td>-132.9</td>
<td>.000</td>
<td>14.000</td>
<td>.316</td>
</tr>
<tr>
<td>-125.1</td>
<td>.000</td>
<td>15.000</td>
<td>.256</td>
</tr>
<tr>
<td>-117.3</td>
<td>.005</td>
<td>16.000</td>
<td>.197</td>
</tr>
<tr>
<td>-109.5</td>
<td>.010</td>
<td>17.000</td>
<td>.139</td>
</tr>
<tr>
<td>-101.7</td>
<td>.018</td>
<td>18.000</td>
<td>.083</td>
</tr>
<tr>
<td>-93.9</td>
<td>.032</td>
<td>19.000</td>
<td>.032</td>
</tr>
<tr>
<td>-86.0</td>
<td>.056</td>
<td>20.000</td>
<td>.015</td>
</tr>
<tr>
<td>-78.2</td>
<td>.106</td>
<td>21.000</td>
<td>.000</td>
</tr>
<tr>
<td>-70.4</td>
<td>.168</td>
<td>22.000</td>
<td></td>
</tr>
<tr>
<td>-62.6</td>
<td>.239</td>
<td>23.000</td>
<td></td>
</tr>
<tr>
<td>-54.7</td>
<td>.411</td>
<td>24.000</td>
<td></td>
</tr>
<tr>
<td>-46.9</td>
<td>1.100</td>
<td>25.000</td>
<td></td>
</tr>
<tr>
<td>-39.1</td>
<td>1.953</td>
<td>26.000</td>
<td></td>
</tr>
<tr>
<td>-31.3</td>
<td>3.872</td>
<td>27.000</td>
<td></td>
</tr>
<tr>
<td>-23.5</td>
<td>6.426</td>
<td>28.000</td>
<td></td>
</tr>
<tr>
<td>-15.6</td>
<td>11.596</td>
<td>29.000</td>
<td></td>
</tr>
<tr>
<td>-7.6</td>
<td>20.893</td>
<td>30.000</td>
<td></td>
</tr>
<tr>
<td>.0</td>
<td>37.464</td>
<td>31.000</td>
<td></td>
</tr>
<tr>
<td>7.8</td>
<td>20.893</td>
<td>32.000</td>
<td></td>
</tr>
<tr>
<td>15.6</td>
<td>11.596</td>
<td>33.000</td>
<td></td>
</tr>
<tr>
<td>23.3</td>
<td>6.426</td>
<td>34.000</td>
<td></td>
</tr>
<tr>
<td>31.3</td>
<td>3.872</td>
<td>35.000</td>
<td></td>
</tr>
<tr>
<td>39.1</td>
<td>1.953</td>
<td>36.000</td>
<td></td>
</tr>
<tr>
<td>46.9</td>
<td>1.100</td>
<td>37.000</td>
<td></td>
</tr>
<tr>
<td>54.7</td>
<td>.411</td>
<td>38.000</td>
<td></td>
</tr>
<tr>
<td>62.6</td>
<td>.239</td>
<td>39.000</td>
<td></td>
</tr>
<tr>
<td>70.4</td>
<td>.168</td>
<td>40.000</td>
<td></td>
</tr>
<tr>
<td>78.2</td>
<td>.106</td>
<td>41.000</td>
<td></td>
</tr>
<tr>
<td>86.0</td>
<td>.058</td>
<td>42.000</td>
<td></td>
</tr>
<tr>
<td>93.8</td>
<td>.032</td>
<td>43.000</td>
<td></td>
</tr>
<tr>
<td>101.7</td>
<td>.018</td>
<td>44.000</td>
<td></td>
</tr>
<tr>
<td>109.5</td>
<td>.010</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>117.2</td>
<td>.005</td>
<td>46.000</td>
<td></td>
</tr>
<tr>
<td>125.1</td>
<td>.003</td>
<td>47.000</td>
<td></td>
</tr>
<tr>
<td>132.9</td>
<td>.002</td>
<td>48.000</td>
<td></td>
</tr>
<tr>
<td>140.6</td>
<td>.001</td>
<td>49.000</td>
<td></td>
</tr>
<tr>
<td>148.6</td>
<td>.000</td>
<td>50.000</td>
<td></td>
</tr>
<tr>
<td>156.4</td>
<td>.000</td>
<td>51.000</td>
<td></td>
</tr>
<tr>
<td>164.2</td>
<td>.000</td>
<td>52.000</td>
<td></td>
</tr>
<tr>
<td>172.0</td>
<td>.000</td>
<td>53.000</td>
<td></td>
</tr>
<tr>
<td>179.9</td>
<td>.000</td>
<td>54.000</td>
<td></td>
</tr>
<tr>
<td>187.7</td>
<td>.000</td>
<td>55.000</td>
<td></td>
</tr>
<tr>
<td>195.5</td>
<td>.000</td>
<td>56.000</td>
<td></td>
</tr>
<tr>
<td>203.3</td>
<td>.000</td>
<td>57.000</td>
<td></td>
</tr>
<tr>
<td>211.1</td>
<td>.000</td>
<td>58.000</td>
<td></td>
</tr>
<tr>
<td>219.0</td>
<td>.000</td>
<td>59.000</td>
<td></td>
</tr>
<tr>
<td>226.8</td>
<td>.000</td>
<td>60.000</td>
<td></td>
</tr>
<tr>
<td>234.6</td>
<td>.000</td>
<td>61.000</td>
<td></td>
</tr>
<tr>
<td>242.4</td>
<td>.000</td>
<td>62.000</td>
<td></td>
</tr>
<tr>
<td>250.2</td>
<td>.000</td>
<td>63.000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 26: Models for line spread function and modulation transfer function.
Figure 27: Comparison of measured line spread function and best-fit model.
Figure 28: Comparison of measured OTF data and MTF model.
Figure 29: Computed modulation transfer function.
Figure 30: MTF from three different traces across the same edge.
Figure 31: Comparison of mean estimated MTF and manufacturer's data.
At the onset of this work it was hypothesized that some small but noticeable improvement in MTF estimates from noisy data would result from working with the data in two domains. The limited testing permitted by the available time and resources, has supported that original premise to a greater degree than expected. The method does indeed work, even with extremely noisy data. The author feels, however, that there remains considerable room for improvement and additional testing. Recommendations for further work follow in the next chapter.
RECOMMENDATIONS

The primary recommendation for future work on this topic is that the edge response function be modeled rather than the line spread function. The reason becomes obvious when one compares Figures 24 and 25. Whereas the former is clearly recognizable as an edge, the latter is hardly suggestive of a typical spread function. The great difference in noise levels is the result of numerical differentiation. By avoiding that step, even better MTF estimates should result.

To get OTF data directly from an edge, the edge transform must be multiplied by $i2\pi f$. The relatively constant noise level in the space domain thus becomes proportional to frequency in the other domain. To compensate for this it may be necessary to weight the parameter estimation routine in favor of the space domain.

Alternately, one may choose to filter the OTF. In this context, filtering simply means multiplying the transform of a function by a suitable filter, usually for the purpose of modifying or eliminating certain frequencies. The optimal filter would be one that eliminates the noise
transform, \( N(f) \), entirely while leaving the signal transform, \( S(f) \), unchanged. Since the data is the sum of signal plus noise (Equation 35), the optimal filter, \( T(f) \), is:

\[
T(f) = \frac{S(f)}{[S(f) + N(f)]}
\]  

(45)

Obviously, if \( T(f) \) is multiplied by \( U(f) \), (Equation 36), then,

\[
U(f)T(f) = S(f)
\]  

(46)

Unfortunately, \( S(f) \) is unknown, but if \( S(f) \) can be approximated by \( S'(f) \), then the optimal filter is:

\[
T(f) = \frac{S'(f)}{[S(f) + N(f)]}
\]  

(47)

If filtering were applied during the parameter estimation routine, then \( T(f) \) could be up-dated with each new estimate of \( S(f) \). This is adaptive optimal filtering, and may be ideally suited to the type of problem addressed by this paper.

Edge modeling and adaptive optimal filtering should not be regarded as finishing touches on an otherwise complete research effort. Instead, each topic represents a starting point for major new investigations into the relatively unexplored field of two-domain function estimation. This project could have been continued indefinitely, but ends here only because the original hypothesis; that parameter estimation in two domains can be better than in either
domain alone; has been supported with qualitative, if not quantitative results. As is often the case, the search for answers has uncovered as many questions, and further inquiries will be limited only by the experimenter's imagination.
SUMMARY

The working hypothesis behind this research project is that buried within noisy experimental data for an emulsion spread function lies a well-behaved function which can be closely approximated by a properly chosen mathematical model. This model can be fully described by as few as two or three parametric constants. The same parameters determine the shape of the model's Fourier transform, the MTF model, and thus there are effectively two data sets from which to extract estimates of the parameters. Fitting two functions simultaneously benefits from the fact that the noise in each domain affects the estimate in different ways, because the error at a given point in one domain is distributed over all the points in the other domain.

The models are fit to the data by a method of least squares, and where non-linear coefficients are involved, Newton's method is employed. Two FORTRAN computer programs were written to fit Gaussian and exponential spread function models respectively, and compute the corresponding MTF values. A third program was created to analyze and plot out the mean square error surfaces in each domain separately, and in both domains together. The contour plotting program
showed that not only were the error surfaces smooth, with single distinct minima, but also that the two-domain error surface promised to yield the best parameter estimates.

Of the two MTF estimating programs, EMODX, for fitting an exponential spread function, evolved the furthest, and was tested using both synthetic spread function data and real edge trace data. The artificial data provided information about deviation in computed MTF as a function of frequency and signal-to-noise ratio (SNR). It was found that even for moderately low SNR the program succeeded in converging to MTF models varying by only a few percent. With three sets of extremely noisy edge data from microdensitometer traces, the program computed MTF values reasonably close to the film manufacturer's curves.

The basic premise; that parameter estimation in two domains can be better than either domain alone; received considerable support from the experimental results. Two new areas were recommended for investigations which may lead to even better MTF estimates from noisy data.
REFERENCES


4. Ibid., p.613


8. Fujicolor F-II 400, Film Data Sheet No. AF3-89E, Fuji Film Co., Ltd., Tokyo, 1976


51. Blackman, E.S.: Reference 13, p.105


79. Ibid., p.189ff


82. Ibid., p.203


84. Bracewell, R.N.: Reference 78, Chapter 6

85. Brigham, E.O.: Reference 80, Chapter 3


87. Brumbach, R.P.: FFT24, a FORTRAN subroutine for computing the radix 4+2 Fast Fourier Transform of an equispaced, well ordered complex sequence. Sea Operations Department, AC Electronics Defence Research Labs, Santa Barbara, California. c. 1970

88. After Blackman, E.S.: Reference 54, p.245

ADDitional references


APPENDIX A: MAIN PROGRAMS

EMODX
ERRCON
GAUSS
THIS PROGRAM FINDS THE VALUES OF THE PARAMETERS,

\[ \text{LSF}(x) = H \cdot \exp\left(-\frac{\text{ABS}\left((x-P)/W\right)}{W}\right) \]

AND SIMULTANEOUSLY IN THE FREQUENCY DOMAIN MODEL:

\[ \text{MTF}(f) = 2H \cdot \frac{W}{(1+(2\pi W f)^2)^2} \]

WHICH RESULT IN A LEAST SQUARE FIT TO THE NOISY LINE

SPREAD FUNCTION DATA, PARTIAL DERIVATIVES, WITH

RESPECT TO EACH PARAMETER, OF AN EXPRESSION FOR

THE MEAN SQUARE ERROR, ARE SET EQUAL TO ZERO

SOLVED THROUGH SUCCESSIVE APPROXIMATIONS BY A

MODIFIED NEWTON'S METHOD. SUBROUTINES USED:

'FFT24', 'FTLIST3', 'INVFFT', 'PL0T2', 'RANDN0',
'SORT' AND 'UNSORT'.

LAST MODIFIED ... 15:05 05AUG78 JHC

EQUIVALENCE (RP*RW*, (CH,CP,CW,CE), (PDP,WDW)
EQUIVALENCE (FM,RELH),(FM2,TRAN)
REAL FM2(64),AM(64),PH(64)
REAL RN(64),G(64),XR(64),XI(64),XM(64)
REAL FR(64),FI(64),FM(64),DENS(64),TRAN(64),RELH(64)
EXTERNAL DEDP,DEDW1,DEDW2
DOUBLE PRECISION H,H1,H2,H3,H4,CH,EX1,EX2,PI
DOUBLE PRECISION DEDW1,DEDW2,VI,CI,DI,VDW,WDW,RW,TOL
DOUBLE PRECISION ERR,ERR1,ERRF,ERRX,ERRTOL,CE,DERV
DATA NID,NØD,NMAX/101,108,300/

SYMBOLS AND CHARACTERS FOR PLOTS
DATA KR,KU,KM,KD,KT,KH/HHR,1H+,1H#,1HD,1HT,1HH/
PI=3.14159265358979323846

NOTE: DATA IS READ IN FROM INPUT DEVICE NUMBER 101.
IN BATCH ... !ASSIGN F:101, (FILE,XXXXX), (IN)
OR ... !ASSIGN F:101, (DEVICE,CR)
ON LINE .... !SET F:101 DC/XXXXX; IN
WHERE XXXX IS THE NAME OF THE INPUT FILE, IF USED.

READ(NID,1) NRUNS,NW,KDATA
1 FORMAT(12,IX,13,IX,11)
NRUNS IS THE NUMBER OF TIMES THE PROGRAM REPEATS.
NW IS THE WIDTH OF THE PAGE FOR PLOTS BY 'PL0T2'.
IF KDATA = 1, USER'S EDGE TRACE DATA IN DENSITY
IS READ IN VIA INPUT DEVICE NUMBER 102.
IF KDATA = 0, THE PROGRAM CREATES ITS OWN DATA.
C  TOLERANCES
   READ(NID,3) DP,DV,TOL,ERRTOL
   FORMAT(4(G10.8,1X))
C  
   L00P=L00P2=0
   G0 TO 99
98 WRITE(N0D,360)
99 L00P=L00P+1
C  
   IF(KDATA.LE.0) G0 TO 11
C  NUMBER OF DATA POINTS = 2**M
C  IPEX = USER'S ESTIMATE OF THE NUMBER OF POINTS FROM THE
   CENTER OF THE EDGE TO THE CENTER OF THE SPACE.
C  LIM = MARGIN OF ERROR FOR IPEX. THAT IS, PROGRAM
   WILL EXPECT TO FIND CENTER OF EDGE AT
   (IPEX + OR - LIM) FROM CENTER OF SPACE.
C  NC0NV = NUMBER OF TIMES PROGRAM REPEATS AFTER FIRST
   SMOOTHING THE EDGE BY CONVOLUTION WITH A TRIANGLE.
C  FIRST RUN IS ALWAYS UNSMOOOTED. EACH SUCCESSIVE
   RUN IS SMOOTHER THAN THE LAST. IF NC0NV IS
   GREATER THAN 0, BETTER INCREASE YOUR PAGE LIMIT!
C  DX = DISTANCE BETWEEN SAMPLES IN MILLIMETRES.
C  GAM = SLOPE OF MICRO-CURVE.
C  HO = INTERSECTION OF LINEAR PORTION OF CURVE WITH
   DENSITY AXIS.
   READ(102,17) N,IPEX,LIM,NC0NV
   READ(102,18) DX,GAM,H0
   F0RMAT(4(12,1X))
17   F0RMAT(3(G10.6,1X))
   N=2**M
   AN=FL0AT(N)
   DF=1.0/(AN*DX)
   NMAX=MAX0(2*N,NMAX)
   PEX=FL0AT(IPEX)*DX
   HT=SIX=XBAR=STDV=0.0
   NSD=NSD1=0
   DO 26 I=1,N/8
   J=8*I-7
26   READ(102,14) (DENS(K),K=J,J+7)
   F0RMAT(8(F4.2,1X))
   HO=10.0**(-H0)
   DMAX=DENS(1)
   DO 71 I=1,N
   DMAX=AMAX1(DENS(I),DMAX)
   TRAN(I)=10.0**(-DENS(I))
   RELH(I)=(H0/TRAN(I))**(1.0/GAM)
   XR(1)=0.0
   D0 72 I=2,N
   XR(I)=RELH(I)-RELH(I-1)
C * PLOUT OUT EDGE TRACE DATA.
WRITE(N0D,350)
WRITE(N0D,380) DMAX,GAM
CALL PLOT2(DENS,KD,DENS,KD,N,NW,N0D,-1)
WRITE(N0D,350)
WRITE(N0D,390)
CALL PLOT2(TAN,KT,TAN,KT,N,NW,N0D,-1)
WRITE(N0D,350)
WRITE(N0D,400)
CALL PLOT2(RELH,KH,RELH,KH,N,NW,N0D,-1)

X  G0 TO 16
G0 TO 19

C * THIS BLOCK OF STATEMENTS IS EXECUTED ONLY IF USER
C * HAS NO DATA AND WANTS PROGRAM TO SYNTHESIZE DATA.

C * SPREAD FUNCTION DATA.
11 READ(NID,2) M,DX,PEX,HT,SIG
2 FORMAT(I1,2(X,G10.8),2(I,G8.4))
N=2**M
AN=FLOAT(N)
DF=1.0/(AN*DX)
NMAX=MAX0(2*N,NMAX)
IPEX=INT(PEX/DX+0.5)
LIM=N/4

C * NOISE STATISTICS. SNR = (PEAK SIGNAL)/(RMS NOISE)
READ(NID,4) SNR,NSD
4 FORMAT(F5.2,2,13)
XBAR=ERR=0.0
STDV=HT/SNR
NSD1=NSD

C * THIS SET OF STATEMENTS PRODUCES AN EXPONENTIAL
C * FUNCTION OF KNOWN PARAMETERS, WITH RANDOM NOISE
C * ADDED. THESE DATA CAN BE USED IN LIEU OF ACTUAL
C * SPREAD FUNCTION DATA TO TEST THE PARAMETER
C * ESTIMATION ROUTINE.
CTR=FLOAT(N/2)*DX
CALL RANDN0(RN,XBAR,STDV,N,NSD)

X WRITE(N0D,350)
X WRITE(N0D,140)
D0 10 I=1,N
XA=DX*FLOAT(I)
XA=(XA-CTR-PEX)/SIG
G(I)=HT*EXP(-ABS(XA))
XR(I)=ABS(G(I)+RN(I))
XI(I)=FI(I)=0.0

X WRITE(N0D,150) I,G(I),RN(I),XR(I)
10 CONTINUE

C * THE FIRST ESTIMATE OF THE SHIFT PARAMETER, P, IS
C * THE USER'S ESTIMATE.
19 WRITE(N0D,360)
 16 FP=PEX
     P=DBLE(FP)

C * THE FIRST ESTIMATE OF THE WIDTH SCALING PARAMETER, W,
C * IS THE DISTANCE, DX, BETWEEN 2 ADJACENT DATA POINTS.
     W=DBLE(DX)

C * THE FIRST ESTIMATE OF THE HEIGHT SCALING PARAMETER, H,
C * IS THE HIGHEST POINT ON THE SPREAD FUNCTION AFTER
C * PARTIAL SMOOTHING THROUGH A RUNNING AVERAGE (3 TERMS).
     FH=0.0
     DO 15 I=2,N-1
       RUN=(XR(I-1)+XR(I)+XR(I+1))/3.0
     15 FH=AMAX1(FH,RUN)
     H=DBLE(FH)

C * COMPUTE NEXT ESTIMATE OF W FROM LSF DATA
X WRITE(N0D,350)
X WRITE(N0D,190)
     ITERH=ITERP=ITERW=0
22 ITERW=ITERW+1
     WDW=W+DW
     DERV=DEDW1(H,PZ,W,DX,XR,N)
     RP=DEDW1(H,PZ,WDW,DX,XR,N)/DERV
     W1=W+DW/(1.0-RW)
C * COMPUTE CHANGE IN W
     CV=W1-W
     24 CW=CW/2.0
     IF(DABS(CW).GT.DBLE(DX)) G0 T0 24
     W=W+CW
     IF(V.LT.DBLE(DX)) W=DBLE(FLOAT(ITERW)*DX)
X WRITE(N0D,220) ITERW,W,CW,DERV
     IF(ITERW.GE.NMAX) W=W-CW*0.5; WRITE(N0D,240); G0 T0 40
     IF(DABS(DERV).LT.TOL) G0 T0 22
     IF(SNR.LT.11.0) W=W*0.8
     IF(SNR.LT.8.0) W=W*0.6
     FW=SNGL(W)
X WRITE(N0D,240)
C * COMPUTE NEXT ESTIMATE OF P FROM LSF DATA
40 ITERP=ITERP+1
     PDP=P+DP
     RP=DEDP(H,PDP,W,DX,XR,N)
     RP=RP/DEDP(H,P,W,DX,XR,N)
     Pl=P+DP/(1.0-RP)
C * COMPUTE CHANGE IN P
CP=(P�-P)/2.0
P=P+CP
WRITE(N0D,200) ITERP,P,CP,DERV
IF(DABS(P).LE.(1.0E-8)) WRITE(N0D,240); P=0.0; G0 T0 39
IF(ITERP.GE.NMAX)WRITE(N0D,320)ITERP;P=DBLE(P0X);G0 T0 39
IF(DABS(DERV).GT.TOL) G0 T0 40
WRITE(N0D,240)
C *
C * SHİFT ALL DATA P0İNTS T0 CENTER THE FUNCTION AT N/2
39 IF(P.LT.0.0) IP=-INT(0.5-P/DX)
IF(P.GE.0.0) IP=INT(0.5+P/DX)
IF(IABS(IP).GT.LIM) IP=IP0X; WRITE(N0D,300)
30 IF(IP.EQ.0) G0 T0 37
31 XR(I)=XR(I+IP)
D0 32 I=(N+1-IP),N
32 XR(I)=0.0
G0 T0 37
33 D0 34 I=1,(N+IP)
34 XR(N+1-I)=XR(N+1-I+IP)
D0 36 I=1,-IP
36 XR(I)=0.0
37 PS=P-DBLE(FL0AT(IP)*DX)
P2=0.0
D0 38 I=1,N
38 FR(I)=XR(I)
WRITE(N0D,310) -IP,PS
WRITE(N0D,370)
C *
C * PLO T OUT SPREAD FUNCTION DATA
WRITE(N0D,350)
WRITE(N0D,160)
CALL PLO T2(XR,KU,XR,KU,N,NW,N0D,-1)
C *
C * FOURİER TRANSFORM LSF DATA T0 GET ØTF DATA
CALL S0RT(FR,N)
CALL FFT24(FR,FI,N)
CALL UNS0RT(FR,N)
CALL UNS0RT(FI,N)
C *
C * SCALE FREQUENCY DØMAIN DATA. SCALE FACT0R = DX.
D0 12 I=1,N
FI(I)=FI(I)*DX
12 FR(I)=FR(I)*DX
C *
C * PRINT ØUT SPREAD FUNCTION DATA AND TRANSFORM
WRITE(N0D,350)
WRITE(N0D,280) DX*1000.0,N,DF
CALL FTLIST3(XR, XI, FR, FI, AM, PH, N, NOD, 1)
FR(N/2) = 0.0

C * COMPUTE NEXT ESTIMATE OF W FROM LSF AND OTF DATA
X WRITE(NOD, 350)
X WRITE(NOD, 190)
35 ITERW = ITERW + 1
WDW = W + DW
DERV = DEDW2(H, PZ, V, DX, DF, XR, FR, N)
RW = DEDW2(H, PZ, VDW, DX, DF, XR, FR, N) / DERV
W1 = W + DW / (1.0 - RW)
C * COMPUTE CHANGE IN W
CW = W1 - W
44 CW = CW / 2.0
IF(DABS(CW) .GT. DBLE(DX)) GO TO 44
V = DABS(W + CW)
X WRITE(NOD, 220) ITERW, W, CW, DERV
IF(ITERW .GE. NMAX) GO TO 45
IF((DABS(DERV) .GT. TOL) .AND. (DABS(CW / W) .GT. TOL * 0.01)) GO TO 35
X WRITE(NOD, 240)
C * COMPUTE NEXT ESTIMATE OF H FROM LSF AND OTF DATA
45 ITERH = ITERH + 1
H1 = H2 = H3 = H4 = 0.0
D0 50 I = 1, (N/2 - 1)
XT = 0.5 * DBLE(XR(N/2 + I) + XR(N/2 - I))
FT = 0.5 * DBLE(FR(N/2 + I) + FR(N/2 - I))
EX1 = DBLE(FLOAT(I) * DX)
EX1 = H * DEXP(-DABS(EX1 / W))
EX2 = DBLE(FLOAT(I) * DF)
EX2 = 4.0 * W / (1.0 + (2.0 * PI * W * EX2) ** 2)
H1 = H1 + 2.0 * XT * EX1
H2 = H2 + FT * EX2
H3 = H3 + 2.0 * EX1 ** 2
H4 = H4 + 0.5 * EX2 ** 2
50 CONTINUE
H1 = (H1 + H2) / (H3 + H4)
C * COMPUTE CHANGE IN H
CH = (H1 - H) * 0.5
H = H + CH
X WRITE(NOD, 210) ITERH, H, CH, PZ
C * COMPUTE THE MEAN SQUARE ERRØR
ERRX = ERF = 0.0
D0 55 I = 1, (N/2 - 1)
XT = 0.5 * DBLE(XR(N/2 + I) + XR(N/2 - I))
FT = 0.5 * DBLE(FR(N/2 + I) + FR(N/2 - I))
EX1 = DBLE(FLOAT(I) * DX)
EX1 = H * DEXP(-DABS(EX1 / W))
EX2=DFLE(FL0AT(I)*DF)
EX2=2.0*H*W/(1.0+(2.0*PI*W*EX2)**2)
ERRX=ERRX*(EX1-XT)**2
ERRF=ERRF+(EX2-FT)**2
55 CONTINUE
ERR1=(ERRX+ERRF)/DFL0AT(N/2-1)
C * COMPUTE CHANGE IN ERR0R
CE=ERR1-ERR
ERR=ERR1
X WRITE(N0D,230) ERR,CE
IF(ITERW.GT.NMAX) G0 T0 60
IF(ITERP.GT.NMAX) G0 T0 60
IF(ABS(CE).GT.(ERR*ERRTOL*0.01)) G0 T0 35
60 IF(ITERW.GT.NMAX) WRITE(N0D,330) ITERW
C * PRINT OUT INPUT DATA
WRITE(N0D,350)
WRITE(N0D,250) N,DX,HT,SIG
WRITE(N0D,260) XBAR,STDV,NSD1
WRITE(N0D,270) TOL,ERRTOL,DP,DW
C * PRINT OUT PARAMETERS
WRITE(N0D,350)
WRITE(N0D,100)
WRITE(N0D,110) PEX,FP,P
WRITE(N0D,120) SIG,FW,W
WRITE(N0D,130) HT,TH,H
C * COMPUTE AND PRINT VALUES FOR MODELS.
HA=SNGL(H);WA=SNGL(W)
OMAX=2.0*HA*WA
FMAX=DF*AN/2.0
DELTAF=FL0AT(INT(FMAX/AN+0.5))
DELTAF=AMAX1(DELTAF,1.0)
WRITE(N0D,350)
WRITE(N0D,290)
D0 70 I=1,N
XR(I)=XR(I)/OMAX
AM(I)=AM(I)/OMAX
XA=FL0AT(I-N/2)*DX
XM(I)=EXP(-ABS(XA/WA))/(2.0*WA)
XA=XA*1000.0
FA=FL0AT(I-N/2)*DF
FM(I)=1.0/(1.0+(2.0*PI*WA*FA)**2)
F2=FL0AT(I-1)*DELTAF
F2(M(I))=1.0/(1.0+(2.0*PI*WA*F2)**2)
70 WRITE(N0D,410) XA,XM(I),F2,FM2(I)
C *
C * PLOT OUT DATA AND MODELS
WRITE(N0D,350)
WRITE(NOD,180)
CALL PLOT2(XR,KU,XM,KM,N,NW,NOD,-1)
WRITE(NOD,350)
WRITE(NOD,170)
CALL PLOT2(AM,KU,FM,KM,N,NW,NOD,-1)
WRITE(NOD,350)
WRITE(NOD,420) DELTAF
CALL PLOT2(FM2,KM,FM2,KM,N,NW,NOD,1)

C THIS SET OF STATEMENTS SMOOTHES THE EDGE IN DENSITY BY
C CONVOLUTION WITH A TRIANGLE FUNCTION WITH BASE = 4*DX.
X IF(KDATA.LE.0) GO TO 96
IF(KDATA.LE.0) GO TO 97
IF(L00P2.GE.NC0NV) GO TO 97
L00P2=L00P2+1
D1=DENS(1)
D0 85 I=2,(N-1)
D2=(2.0*DENS(I)+DENS(I+1)+D1)/4.0
D1=DENS(I)
85 DENS(I)=D2
DENS(N)=(DENS(N-1)+DENS(N)*3.0)/4.0
G0 TO 26
C X 96 IF(L00P.LT.NRUNS) GO TO 99
97 IF(L00P.LT.NRUNS) GO TO 98

C 100 FORMAT(1HO,T23,'RESULTS OF PARAMETER ESTIMATION R0'
1,'UTINE'///T31,'EXACT'T44,'INITIAL',T59,'FINAL','
2///T28,'PARAMETERS'ESTIMATES')
110 FORMAT///T19,'SHIFT':3X,3(F8.4,6X))
120 FORMAT///T19,'WIDTH':3X,3(F8.4,6X))
130 FORMAT///T18,'HEIGHT':3X,3(F8.4,6X))
X 140 FORMAT(1HI,T12,'ORIGINAL SPREAD FUNCTION '
X 1'DATA BEFORE CENTERING'///T6,'N',T17,
X 2'SIGNAL',T34,'NOISE',T50,'ABS(S+N)',/)
X 150 FORMAT(2X,14,5X,F12.6,5X,F12.6,5X,F12.6)
160 FORMAT(1HI,T8,'ORIGINAL SPREAD FUNCTION DATA '
1'AFTER CENTERING'///)
170 FORMAT(1HI,T8,'M0DULUS OF OTF DATA .. + + + + +',
1///T8,'MTF M0DEL ... # # # # ',/)
180 FORMAT(1HI,T8,'MEASURED LINE SPREAD FUNCTION ...',
1'+ + + + +'///T8,'LSF M0DEL ... # # # # ',/)
X 190 FORMAT(1HI,T21,'CURRENT',T42,'AMOUNT',T63,'PARTIAL',/)
X 1,T3,'ITERATION',T17,'PARAMETER VALUE',T37,
X 2'PARAMETER CHANGED',T62,'DERIVATIVE',/)
X 200 FORMAT(T4,14,T18,'P =',F10.6,T37,'DP =',E12.4,
X 1T58,'DEDP =',E12.4)
X 210 FORMAT(T4,14,T18,'H =',F10.6,T37,'DH =',E12.4,
X 1T58,'DEDH =',E12.4)

WRITE(N0D,180)
_CALL PLOT2(XR,KU,XM,KM,N,NW,N0D,-1)
WRITE(N0D,350)
WRITE(N0D,170)
_CALL PLOT2(AM,KU,FM,KM,N,NW,N0D,-1)
WRITE(N0D,350)
_WRITE(N0D,420) DELTAF
_CALL PLOT2(FM2,KM,FM2,KM,N,NW,N0D,1)

C THIS SET OF STATEMENTS SMOOTHES THE EDGE IN DENSITY BY
C CONVOLUTION WITH A TRIANGLE FUNCTION WITH BASE = 4*DX.
X IF(KDATA.LE.0) G0 T0 96
IF(KDATA.LE.0) G0 T0 97
IF(L00P2.GE.NC0NV) G0 T0 97
L00P2=L00P2+1
D1=DENS(1)
D0 85 I=2,(N-1)
D2=(2.0*DENS(I)+DENS(I+1)+D1)/4.0
D1=DENS(I)
85 DENS(I)=D2
DENS(N)=(DENS(N-1)+DENS(N)*3.0)/4.0
G0 T0 26
C X 96 IF(L00P.LT.NRUNS) G0 T0 99
97 IF(L00P.LT.NRUNS) G0 T0 98

C 100 FORMAT(1HO,T23,'RESULTS OF PARAMETER ESTIMATION R0'
1,'UTINE'///T31,'EXACT'T44,'INITIAL',T59,'FINAL','
2///T28,'PARAMETERS'ESTIMATES')
110 FORMAT///T19,'SHIFT':3X,3(F8.4,6X))
120 FORMAT///T19,'WIDTH':3X,3(F8.4,6X))
130 FORMAT///T18,'HEIGHT':3X,3(F8.4,6X))
X 140 FORMAT(1HI,T12,'ORIGINAL SPREAD FUNCTION '
X 1'DATA BEFORE CENTERING'///T6,'N',T17,
X 2'SIGNAL',T34,'NOISE',T50,'ABS(S+N)',/)
X 150 FORMAT(2X,14,5X,F12.6,5X,F12.6,5X,F12.6)
160 FORMAT(1HI,T8,'ORIGINAL SPREAD FUNCTION DATA '
1'AFTER CENTERING'///)
170 FORMAT(1HI,T8,'M0DULUS OF OTF DATA .. + + + + +',
1///T8,'MTF M0DEL ... # # # # ',/)
180 FORMAT(1HI,T8,'MEASURED LINE SPREAD FUNCTION ...',
1'+ + + + +'///T8,'LSF M0DEL ... # # # # ',/)
X 190 FORMAT(1HI,T21,'CURRENT',T42,'AMOUNT',T63,'PARTIAL',/)
X 1,T3,'ITERATION',T17,'PARAMETER VALUE',T37,
X 2'PARAMETER CHANGED',T62,'DERIVATIVE',/)
X 200 FORMAT(T4,14,T18,'P =',F10.6,T37,'DP =',E12.4,
X 1T58,'DEDP =',E12.4)
X 210 FORMAT(T4,14,T18,'H =',F10.6,T37,'DH =',E12.4,
X 1T58,'DEDH =',E12.4)
X 220 FORMAT(T4,I4,T18,'W = ',F10.6,T37,'DW = ',E12.4)
X 230 FORMAT(IHO,T7,'MEAN SQR. ERRØR = ',F18.14,T47)
X 1 'CHANGE = ',F18.14/
240 FORMAT(IH1)
250 FORMAT(IH1,T15,'DATA: ',10X,'N = ',14//T15,'DX = ',F8.6,
15X,'HEIGHT = ',F7.4,5X,'WIDTH = ',F7.4)
260 FORMAT(IHO,T15,'RANDOM NOISE: ',T15,'MEAN = ',F6.4,
15X,'STD.DEV. = ',F7.4,5X,'SEED = ',I4)
270 FORMAT(IH1,T15,'TOLERANCES: ',T15,'FØR P AND W ... ',
1'DERIVATIVE = ',F8.6,'T15,'FØR ERRØR, ... 'T7.4,
2' PERCENT', T15,'DELTA P = ',F10.8,6X,'DELTA W = ',
3F10.8///)
280 FORMAT(1H1*T20,'SPREAD FUNCTION DATA AND SCALES ',
1'TRANSFORM',//T15,'DX = ',F6.2,' MICRØNS',T38,'N = ',
214/T52,'DF = ',F6.2,' L/MM'//)
290 FORMAT(IH1,T14,'MODEL FØR',T50,'MODEL FØR',//T9,
1'LINE SPREAD FUNCTION',T41,'MODULATION TRANSFER FUNC',
2'T10N',//T8,'DISTANCE NØRMALED',T43,'FREQUEN',
3'CY NØRMALED',//T7,'(MICRØNS) LSF',T42,
4'(CYCLES/MM)'MTF'//)
300 FORMAT(//T6,'CENTER ØF FUNCTION HAS NOT BEEN LØCATED',
1//'T6,'SHIFT HAS BEEN SET EQUAL TO USERS ESTIMATE',//)
310 FORMAT(//T6,'BEFORE BEING TRANSFORMED TO THE ENTIRE ',
1'DATA SET WAS SHIFTED( ',13,'') UNITS',//T6,'TØ CENTER',
2,' THE FUNCTION AT (N/2) IN THE SPACE DOMAIN',//T6,
3,'CENTERING HAS AN ABSOLUTE ERROR ØF LESS THAN (DX/2)',
4//'T6,'AFTER CENTERING, THE SHIFT PARAMETER, P = ',F9.6)
320 FORMAT(//T6,'AFTER ( ',13,' ) ITERATIONS, SHIFT ESTIM',
1'ATING ROUTINE DID NOT CONVERGE',//T6,'SHIFT PARAM',
2'ETER, P, HAS BEEN SET EQUAL TO INITIAL ESTIMATE',//)
330 FORMAT(//T6,'AFTER ( ',13,') ITERATIONS, WIDTH ESTIM',
1'ATING ROUTINE DID NOT CONVERGE',//T6,'WIDTH PARAM',
2'ETER, W, HAS BEEN SET EQUAL TO THE AVERAGE ØF ITS ',
3'LAST',//T6,'TUØ VALUES',//)
X 340 FORMAT(IH1,T21,'LINE SPREAD FUNCTION MODEL IN SPACE ',
X 1'DØMAIN',//T27,'MTF MODEL IN FREQUENCY DOMAIN',
X 27//T16,'DX = ',F9.6,T38,'N = ',I4,T53,'DF = ',F9.6,,/
350 FORMAT(//)
360 FORMAT(IH1)
370 FORMAT(//T6,'THIS PARAMETER WAS THEN RESET TO Ø ZERO',//)
380 FORMAT(IH1,T8,'EDGE TRACE DATA IN DENSITY',10X,
1'DMAX = ',F5.2,'T8,'GAMMA FØR MICRØ-CHARACTERISTIC',
2' CURVE = ',F5.2,///)
390 FORMAT(IH1,T8,'EDGE TRACE DATA IN TRANSMITTANCE',//)
400 FORMAT(IH1,T8,'EDGE TRACE DATA IN RELATIVE ',
1'EXPOSURE',//)
410 FORMAT(T7,F8.1,T19,F8.3,T44,F6.1,T58,F6.3)
420 FORMAT(IH1,T8,'MODULATION TRANSFER FUNCTION',//
1T8,'DELTA F = ',F4.1,' CYCLES/MM',//)
STOP
END

C *
FUNCTION DEDW1(H,P,W,DX,XR,N)
C * COMPUTES THE PARTIAL DERIVATIVE OF E, THE MEAN SQUARE
C * ERROR WITH RESPECT TO W, THE WIDTH SCALING PARAMETER.
DIMENSION XR(N)
DOUBLE PRECISION DEDW1,H,P,W,V1,W2,X,XCTR,EX
W2=0.0
XCTR=DBLE(FLOAT(N/2)*DX)
D0 10 I=1,N
X=DBLE(DX*FLOAT(I))
X=X-XCTR-P/W
EX=DEXP(-X)
W1=H*EX**2-DBLE(XR(I))*EX
10 W2=W2+X*W1
DEDW1=(2.0*H/W)*W2/DFLOAT(N)
RETURN
END

C *
FUNCTION DEDP(H,P,W,DX,XR,N)
C * COMPUTES THE PARTIAL DERIVATIVE OF THE MEAN SQUARE
C * ERROR WITH RESPECT TO P, THE SHIFT PARAMETER.
DIMENSION XR(N)
DOUBLE PRECISION DEDP,P,P1,P2,H,V,X,EX,XCTR
P2=0.0
XCTR=DBLE(FLOAT(N/2)*DX)
D0 10 I=1,N
X=DBLE(FLOAT(I)*DX)
X=X-XCTR-P
EX=DEXP(-DABS(X/W))
P1=(H*EX**2)-(DABS(XR(I))*EX)
IF(X.LT.0.0) P1=-P1
10 P2=P2+P1
DEDP=(2.0*H/W)*P2/DFLOAT(N)
RETURN
END

C *
FUNCTION DEDV2(H,P,W,DX,DF,XR,FR,N)
C * COMPUTES THE PARTIAL DERIVATIVE OF E, THE MEAN SQUARE
C * ERROR WITH RESPECT TO W, THE WIDTH SCALING PARAMETER.
DIMENSION XR(N),FR(N)
DOUBLE PRECISION DEDV2,V1,V2,H,P,XPW,FPW,EX1,EX2
DOUBLE PRECISION XT,FT,W3,W4,W5
P1=3.14159265358979323846
W2=W3=W4=W5=0.0
D0 10 I=1,(N/2-1)
XT=0.5*DBLE(XR(N/2+I)+XR(N/2-I))
FT=0.5*DBLE(FR(N/2+I)+FR(N/2-I))
XPW=DBLE(FLOAT(I)*DX)
XPW = DABS((XPW - P) / W)
FPW2 = DBLE(FL0AT(1) * DF)
FPW2 = (2.0 * PI * W * FPW2) ** 2
EX1 = DEXP(-XPW)
EX2 = 4.0 / (1.0 + FPW2)
W1 = (H * EX1 ** 2) - (XT * EX1)
W1 = (2.0 * H / W) * XPW * W1
W2 = W2 + W1
W3 = W3 - (W * H / 4.0) * FPW2 * EX2 ** 3
W4 = W4 + ((W * H / 2.0) + FT * FPW2 / 2.0) * EX2 ** 2
W5 = W5 - H * FT * EX2
10 CONTINUE
DEDW2 = (W2 + W3 + W4 + W5) / DFL0AT(N/2 - 1)
RETURN
END
**ERRCON**

REAL XR(64), XI(64), XM(64), FR(64), FI(64), FM(64), AM(64)
REAL QX(40, 50), QF(40, 50), RN(64), PH(64)
DIMENSION LV(40, 3), LH(3, 50)
DOUBLE PRECISION FTRH, FTRW, PWRH, PWRW, T
DATA M, N, NH, NW/6, 64, 40, 50/
DATA NID, N0D/105, 108/
DX=DF=1.0/SQRT(FL0AT(N))
XCTR=FLOAT(N/2)*DX

C * MULTIPLICATIVE FACTORS FOR CHANGING PARAMETER VALUES
FTRH=2.0/DFLOAT(NH)
FTRW=2.0/DFLOAT(NW)
NH2=NH/2+1
NW2=NW/2

C * HX, PX AND WX ARE THE TRUE PARAMETERS FOR WHICH
C * THE ERROR SHOULD BE AT A MINIMUM
DATA HX, PX, WX/1.0, 0.0, 0.0, 0.5/

C * NOISE STATISTICS
READ(NID, 1) SNR, NSD
1 F0RMAT(F5.2, */13)
STDV=HX/SNR
XBAR=0.0
NSD1=NSD

C * READ IN DATA FOR CONTOUR PLOTS
READ(NID, 3) NCT, NCH, T
3 F0RMAT(12, IX, 12, IX, F5.3)
D0 5 J=1, 3
5 READ(NID, 9) (LV(I, J), I=1, NH)
D0 7 I=1, 3
7 READ(NID, 9) (LH(I, J), J=1, NW)
9 F0RMAT(80A1)

C * CHARACTERS AND SYMBOLS FOR PLOTS
DATA KH, KP, KW, KX, KF/1HH, 1HP, 1HW, 1HX, 1HF/
DATA KR, KI, KM, KS, K8, K9/1HR, 1HI, 1HM, 1HS, 1H#, 1H+

C * THIS SET OF STATEMENTS PRODUCES A GAUSSIAN FUNCTION,
C * OF KNOWN PARAMETERS, WITH RANDOM NOISE ADDED.
CALL RANDNO(RN, XBAR, STDV, N, NSD)
WRITE(N0D, 250)
D0 10 I=1, N
X = DX * FL0AT(1)
X = (X - XCTR) / WX
PH(I) = HX * EXP(-X**2)
XR(I) = FR(I) = ABS(PH(I) + RN(I))
XI(I) = FI(I) = 0.0
X
WRITE(N0D,260) I, PH(I), RN(I), XR(I)
10 CONTINUE
WRITE(N0D,280) XBAR, STDV, NSD1, SNR
WRITE(N0D,270)
CALL PL0T2(PH, KS, XR, K9, N, 80, N0D, -1)
C *
TRANSFØRM LSF DATA TO GET OTF DATA.
CALL SØRT(FR, N)
CALL FFT24(FR, FI, M)
CALL UNSØRT(FR, N)
CALL UNSØRT(FI, N)
C *
SCALE FREQUENCY DOMAIN DATA. SCALE FACTØR = DX.
D0 12 I=1*N
FI(I) = FI(I) * DX
12 FR(I) = FR(I) * DX
X
WRITE(N0D,230) DX, N, DF
X
CALL FTLIST3(XR, XI, FR, FI, AM, PH, N, N0D, -1)
C *
BEFØRE PLOTTING TRANSFØRM, CENTER POINT IS REMØVED.
FR(N/2) = AM(N/2) = 0.0
WRITE(N0D,200)
CALL PL0T2(FI, KI, FR, KR, N, 80, N0D, -1)
X
WRITE(N0D,210)
X
CALL PL0T2(AM, KM, FR, KR, N, 80, N0D, -1)
C *
USE EXACT PARAMETER VALUES TO COMPUTE AND PLOT
C *
THE FREQUENCY DOMAIN (OTF) MODEL.
CALL ERR(HX, PX, VX, DX, DF, XR, XM, FR, FM, N, ERX, ERF)
X
WRITE(N0D,240) DX, N, DF
X
CALL FTLIST2(XM, XI, FM, XI, N, N0D, -1)
WRITE(N0D,220)
CALL PL0T2(FM, K8, FM, K8, N, 80, N0D, -1)
C *
COMPUTE ERROR AS A FUNCTION OF H AND W.
D0 20 I=1*NH
PWRH=DFL0AT(NH2-I )*FTRH
H=HX*SNGL(T**PWRH)
D0 20 J=1*NW
PWRW=DFL0AT(J-NW2)*FTRW
W=WX*SNGL(T**PWRW)
CALL ERR(H, PX, VX, DX, DF, XR, XM, FR, FM, N, ERX, ERF)
QX(I, J) = ERX
QF(I, J) = ERF
20 CONTINUE
C *
PLOT ØUT ERROR SURFACE CØNTØURS
WRITE(N0D,290)
WRITE(N0D,320)
CALL C0NT0UR3(QX,NH,NW,LV,LH,XMX,XMN,NCT,NCH,N0D)
WRITE(N0D,330) XMX,XMN
D0 22 I=1,NH
D0 22 J=1,NW
IF(QX(I,J).GT.XMN) G0 T0 22
PWRH=DFLOAT(NH2-I)*FTRH
H=SNGL(T**PWRH)
PWRW=DFLOAT((J-NW2)*FTRW
W=SNGL(T**PWRW)
WRITE(N0D,340) I,J,H,W,NH2,NW2,T,T
22 CONTINUE
C *
WRITE(N0D,300)
WRITE(N0D,320)
CALL C0NT0UR3(QF,NH,NW,LV,LH,XMX,XMN,NCT,NCH,N0D)
WRITE(N0D,330) XMX,XMN
D0 24 I=1,NH
D0 24 J=1,NW
IF(QF(I,J).GT.XMN) G0 T0 24
PWRH=DFLOAT(NH2-I)*FTRH
H=SNGL(T**PWRH)
PWRW=DFLOAT((J-NW2)*FTRW
W=SNGL(T**PWRW)
WRITE(N0D,340) I,J,H,W,NH2,NW2,T,T
24 CONTINUE
C *
D0 26 I=1,NH
D0 26 J=1,NW
26 QX(I,J)=QX(I,J)+QF(I,J)
WRITE(N0D,310)
WRITE(N0D,320)
CALL C0NT0UR3(QX,NH,NW,LV,LH,XMX,XMN,NCT,NCH,N0D)
WRITE(N0D,330) XMX,XMN
D0 28 I=1,NH
D0 28 J=1,NW
IF(QX(I,J).GT.XMN) G0 T0 28
PWRH=DFLOAT(NH2-I)*FTRH
H=SNGL(T**PWRH)
PWRW=DFLOAT((J-NW2)*FTRW
W=SNGL(T**PWRW)
WRITE(N0D,340) I,J,H,W,NH2,NW2,T,T
28 CONTINUE
C *
200 FORMAT(1H1,T8,'TRANSFORM OF DATA:'//,T8,'REAL PART',
'1' ... R R R R'10X,'IMAGINARY PART ... I I I I',//)
X 210 FORMAT(1H1,T8,'TRANSFORM OF DATA:'//,T8,'MODULUS ',
'1' ... M M M'10X,'REAL PART ... R R R',//)
220 FORMAT(1H1,T8,'FREQUENCY DOMAIN MODEL:'//)
X 230 FORMAT(1H1,T20,'SPREAD FUNCTION DATA AND SCALED TRANSFORM',
X 1//'T16,'DX = ',F9.6,T38,'N = ',I4,T53,'DF = ',F9.6,///)
X 240 FORMAT(1H1,T21,'LINE SPREAD FUNCTION MODEL IN SPACE ',
X 1'DOMAIN'/,'T27','0 TF MODEL IN FREQUENCY DOMAIN',
X 27/,'T16','DX = ',F9.6,T38,'N = ',I4,T53,'DF = ',F9.6,/'
X 250 FORMAT(1H1,T11,'ORIGINAL SPREAD FUNCTION TEST DATA',
X 1//'T6,'N','T17,'SIGNAL','T34,'NOISE','T52,'(S+N)/
X 260 FORMAT(2X*'I4*5X;F12.6*5X*F12.6*5X*F12.6)/
X 270 FORMAT(1H1,T8,'SIGNAL +++ SSSSSS+++ ',1OX
X 280 FORMAT(1H1,T18,'RAND0M'*7X,'MEANr*6X* 'STDV'*6X*
X 290 FORMAT(1H1,T20,'WHERE R = ('*F6.4') TIMES THE TRUE HEIGHT',
X 1,'T18','AND W = ('*F6.4') TIMES THE TRUE WIDTH',
X 300 FORMAT(1H1,T16,'MINIMUM SHOULD OCCUR AT Q(',I2,*','I2,')',
X 310 FORMAT(1H1,T20,'TOTAL MEAN SQUARE ERROR SURFACE IN SPACE',
X 320 FORMAT(1H1,T16,'AS A FUNCTION OF HEIGHT AND WIDTH.',
X 330 FORMAT(1H1,T14,'MAXIMUM = ',F9.4,7X,'MINIMUM = ',F9.4)
X 340 FORMAT(1H1,T23,'MINIMUM OCCURS AT Q(',I2,','I2,')',
X 350 FORMAT(1H1,T20,'H AND W BOTH VARIED FROM (1/5*F5.3)',
X 360 FORMAT(1H1,T19,'RETURN',
X 370 FORMAT(1H1,T18,'AFLD U = ',C*F6.4*,' TIMES THE TRUE WIDTH','
X 380 FORMAT(1H1,T20,'MEAN SQUARE ERROR SURFACE IN FREQ',
X 390 FORMAT(1H1,T16,'MEAN SQUARE ERROR SURFACE IN FREQ',
X 400 FORMAT(1H1,T8,'SIGNAL PLUS NOISE ...++++ ','
X 410 FORMAT(1H1,T18,'REAL XR(N)*XM(N)*FR(N)*FM(N)*ERX*ERF
X 420 FORMAT(1H1,T20,'TRANSLATE XR(XM(N/2+1)) FR(N/2+I))
X 430 FORMAT(1H1,T16,'EX1=HYPERCOS-EX1',
X 440 FORMAT(1H1,T16,'FM(N/2+I)=FM(N/2-1)+EX2
X 450 FORMAT(1H1,T16,'ERX=EX1+((EX1-P)/W)**2',
X 460 FORMAT(1H1,T16,'ERF=ERF+((EX2-FT)**2)/AN
X 470 FORMAT(1H1,T20,'RETURN',
X 480 FORMAT(1H1,T18,'HTABLE XR(N/2)=H
X 490 FORMAT(1H1,T16,'FM(N/2)=SQRAT(P)*H*W
X 500 FORMAT(1H1,T16,'XM(N)=FM(N)=000
C * 
C * SUBROUTINE ERR(H,P,W,DX,DF,XR,XM,FR,FM,N,ERX,ERF)
C * 
C * COMPUTE THE MEAN SQUARE ERROR
C * 
C * REAL XR(N),XM(N),FR(N),FM(N)
C * 
C * PI=3.1415926536
C * 
C * ERX=ERF=0.0
C * 
C * DO 5 I=1,(N/2-1)
C * 
C * XT=0.5*DBLE(XR(N/2+1)+XR(N/2-1))
C * 
C * FT=0.5*DBLE(FR(N/2+1)+FR(N/2-1))
C * 
C * EX1=FL0AT(I)*DX
C * 
C * EX1=EX1-PI/W)**2
C * 
C * EX2=FL0AT(I)*DF
C * 
C * EX2=(PI*W*EX2)**2
C * 
C * XM(N/2+1)=XM(N/2-1)=EX1
C * 
C * FM(N/2+1)=FM(N/2-1)=EX2
C * 
C * EXR=EXR+((EX1-XT)**2)/AN
C * 
C * ERF=ERF+((EX2-FT)**2)/AN
C * 
C * CONTINUE
C * 
C * XM(N/2)=H
C * 
C * FM(N/2)=SQRAT(P)*H*W
C * 
C * XM(N)=FM(N)=0.0
C * 
C * RETURN
C * 
C * END
*** GAUSS ***

This program finds the values of the parameters, \( H, P \) and \( W \), in the space domain model:

\[
\text{LSF}(X) = H \times \exp\left(-\frac{(X-P)^2}{W^2}\right)
\]

and simultaneously in the frequency domain model:

\[
\text{OTF}(F) = (\sqrt{\pi} \times H \times W) \times \exp\left(-\pi \times W \times F^2\right)
\]

which result in the least squares fit to the noisy line spread function data. Partial derivatives with respect to each parameter, of an expression for the mean square error, are set equal to zero and solved through successive approximations by a modified Newton's method. Subroutines used: 'FFT24', 'FTLIST2', 'FTLIST3', 'INVFFT', 'PL0T2', 'RANDNO', 'SORT' and "JH CLARK, MAY 1978"

Last modified ... 14:15 13JUL78 JHC

REAL RN(64), G(64), XR(64), XI(64), XM(64)
REAL FR(64), FI(64), FM(64)
EXTERNAL DEDP*, DEDV/1, DEDV/2
DOUBLE PRECISION DEDP*, P*, PS*, PZ*, CP*, DP*, PDP*, RP
DOUBLE PRECISION H*, HI, H1, H2, H3, H4, CH, EX1, EX2, PI
DOUBLE PRECISION DEDV1, DEDV2, W*, V*, CW*, DW*, WD*, RV*, T0L
DOUBLE PRECISION ERR*, ERR1, ERRF*, ERRX*, ERRTOL*, CE
DATA NID, N0D, NHAX/101, 108, 200/

Symbols and characters for plots
DATA KR*, KU*, KM/1HR*, 1H*+, 1H#/
PI=3.14159265358979323846

Note: data is read from file named XXXXX.

In batch... !ASSIGN F:101,(FILE, XXXXX),(IN)

On line... !SET F:101 DC/XXXXX; IN

READ(NID, 1) NRUNS, NW
1 FORMAT(12, 1X, I3)
NRUNS is the number of times the program repeats.
NW is the width of the page for plots by 'PL0T2'.

L00P=0
99 L00P=L00P+1

Spread function data
READ(NID, 2) M, DX, PEX, HT, SIG
2 FORMAT(11, 2(1X, G10.8), 2(1X, F8.4))
N=2**M
AN=FLOAT(N)  
DF=1.0/(AN*DX)  
NMAX=MAX0(2*N,NMAX)

C  
C * TOLERANCES
READ(NID,3) DP,DV,T0L,ERR0L
3 FORMAT(4(G10.8,1X))

C  
C * NOISE STATISTICS. SNR = (PEAK SIGNAL)/(RMS NOISE)
READ(NID,4) SNR,NSD
4 FORMAT(F5.2,/,13)
XBAR=ERR=0.0
STDV=HT/SNR
NSD1=NSD
ITERH=ITERP=ITERW=0

C  
C * THIS SET OF STATEMENTS PRODUCES A GAUSSIAN FUNCTION,
C * OF KNOWN PARAMETERS, WITH RANDOM NOISE ADDED. THESE
C * DATA CAN BE USED IN LIEU OF ACTUAL SPREAD FUNCTION
C * DATA TO TEST THE PARAMETER ESTIMATION ROUTINE.
CTR=FLOAT(N/2)*DX
CALL RANDN0(RN,XBAR,STDV,N,NSD)
WRITE(N0D,350)
WRITE(N0D,140)
D0 10 I=1,N
XA=DX*FLOAT(I)
XA=(XA-CTR-PEX)/SIG
G(I)=HT*EXP(-XA**2)
XR(I)=ABS(G(I)+RN(I))
XI(I)=FI(I)=0.0
10 WRITE(N0D,150) I,G(I),RN(I),XR(I)

C  
C * THE FIRST ESTIMATE OF THE SHIFT PARAMETER, P, IS
C * THE DISTANCE FROM THE CENTER OF THE SPACE TO THE
C * WEIGHTED MEAN OF THE SPREAD FUNCTION DATA.
WT=SFX=0.0
D0 20 I=1,N
XA=DX*FLOAT(I)
WT=WT+ABS(XR(I))
SFX=SFX+XA*ABS(XR(I))
20 SFX=SFX+XA*ABS(XR(I))
FP=SFX/WT-CTR
P=DBLE(FP)

C  
C * THE FIRST ESTIMATE OF THE WIDTH SCALING PARAMETER, W,
C * IS THE DISTANCE, DX, BETWEEN 2 ADJACENT DATA POINTS.
W=DBLE(DX)

C  
C * THE FIRST ESTIMATE OF THE HEIGHT SCALING PARAMETER, H,
C * IS THE HIGHEST POINT ON THE SPREAD FUNCTION AFTER
C * PARTIAL SMOOTHING THROUGH A RUNNING AVERAGE (3 TERMS).
FH=0.0
D0 15 I=2,N-1
RUN=(XR(I-1)+XR(I)+XR(I+1))/3.0
15 FH=AMAX1(FH,RUN)
H=DBLE(FH)
C *
C * COMPUTE NEXT ESTIMATE OF W FROM LSF DATA
X WRITE(N0D,350)
X WRITE(N0D,190)
22 ITERW=ITERW+1
WD=WF+W+DW
RW=DEDW1(H,PZ,Wd,Wx,DX,XR,N)
RW=R/DEDP(H,Pz,Wd,Wx,DX,XN,N)
W=W+W+(1.0-RW)
C * COMPUTE CHANGE IN W
CW=W1-W
24 CW=CW/2.0
IF(DABS(CW).GT.DBLE(DX)) G0 TO 24
W=W+CW
IF(W.LT.DBLE(DX)) W=DBLE(FL0AT(ITERW)*DX)
X WRITE(N0D,220) ITERW,W,CW
IF(ITERW.GE.NMAX) W=W-CW*0.5; WRITE(N0D,240); G0 TO 40
IF(DABS(CW/W).GT.(T0L*0.01)) G0 TO 22
IF(SNR.LT.7.0) W=W*0.8
IF(SNR.LT.5.0) W=W*0.6
FW=SNGL(W)
X WRITE(N0D,240)
C * COMPUTE NEXT ESTIMATE OF P FROM LSF DATA
40 ITERP=ITERP+1
PDP=PF+DP
RP=DEDP(H,PDP,W,DX,XR,N)
RP=RDP/DEDP(H,P,W,DX,XR,N)
PI=P+DP/(1.0-RP)
C * COMPUTE CHANGE IN P
CP=(Pl-P)/2.0
P=P+CP
X WRITE(N0D,200) ITERP,P,CP
IF(DABS(P).LE.(1.0E-8)) WRITE(N0D,240); P=0.0; G0 TO 39
IF(ITERP.GE.NMAX) WRITE(N0D,320) ITERP, P=DBLE(FP); G0 TO 39
IF(DABS(CP/P).GT.(T0L*0.01)) G0 TO 40
X WRITE(N0D,240)
C * SHIFT ALL DATA POINTS TO CENTER THE FUNCTION AT N/2
39 IF(P.LT.0.0) IP=-INT(0.5-P/DX)
IF(P.GE.0.0) IP=INT(0.5+P/DX)
IF(IABS(IP).GT.(N/2)) IP=0; WRITE(N0D,300)
IF(IP.EQ.0.0) G0 TO 37
IF(IP.LT.0.0) G0 TO 33
D0 31 I=1,(N-IP)
31 XR(I)=XR(I+IP)
   DO 32 I=(N+1-IP),N
32 XR(I)=0.0
   G0 TO 37
33 DO 34 I=(N+IP)
34 XR(N+1-I)=XR(N+1-I+IP)
   DO 36 I=1,-IP
36 XR(I)=0.0
37 PS=P-DBLE(FL0AT(IP)*DX)
   PZ=0.0
   DO 38 I=1,N
38 FR(I)=XR(I)
   WRITE(N0D,310) -IP,PS
   WRITE(N0D,370)
C *
C * PL̃T OUT SPREAD FUNCTION DATA
   WRITE(N0D,350)
   WRITE(N0D,160)
   CALL PL̃T2(XR,KU,XR,KU,N,NW,N0D,-1)
C *
C * FÔURIER TRANSFORM LSF DATA TÔ GET OTF DATA
   CALL SÔRT(FR,N)
   CALL FFT24(FR,FI,M)
   CALL UNSÔRT(FR,N)
   CALL UNSÔRT(FI,N)
C *
C * SCALE FREQUENCY DOMAIN DATA. SCALE FACTÔR = DX.
   DO 12 I=1,N
12 FI(I)=FI(I)*DX
   FR(I)=FR(I)*DX
C *
C * PRINT OUT SPREAD FUNCTION DATA AND TRANSFORM
   WRITE(N0D,350)
   WRITE(N0D,280) DX,N,DF
   CALL FTLIST2(XR,X1,FR,FI,N,N0D,-1)
   FR(N/2)=0.0
C *
C * COMPUTE NEXT ESTIMATE ÔF W ÔFRÔM LSF AND OTF DATA
X   WRITE(N0D,350)
X   WRITE(N0D,190)
35 ITERW=ITERW+1
   VDW=W+DW
   RV=DEDV2(H,PZ,VDW,DX,DF,XR,FR,N)
   RV=RV/DEDV2(H,PZ,W,DX,DF,XR,FR,N)
   W=W+DW/(1.0-RV)
C *
C * COMPUTE CHANGE IN W
   CW=W1-W
44 CW=CW/2.0
   IF(DABS(CW).GT.DBLE(DX)) G0 TO 44
   W=DABS(W+CW)
WRITE(NOD,220) ITERW,CW
IF(ITERW.NGE.NMAX) W=W-CW*0.5; WRITE(NOD,240); G0 TO 45
IF(DABS(CW/W)*0.01) G0 TO 35
WRITE(NOD,240)

C * COMPUTE NEXT ESTIMATE OF H FROM LSF AND OTF DATA

45 ITERH=ITERH+1
H1=H2=H3=H4=0.0
D0 50 I=1, (N/2-1)
XT=0.5*DBLE(XR(N/2+I)+XR(N/2-I))
FT=0.5*DBLE(FR(N/2+I)+FR(N/2-I))
EX1=DBLE(FL0AT(I)*DX)
EX1=((EX1-PZ)/W)**2
EX1=DEXP(-EX1)
EX2=DBLE(FL0AT(I)*DF)
EX2=(PI*W*EX2)**2
EX2=DEXP(-EX2)
H1=H1+EX1*XT
H2=H2+EX2*FT
H3=H3+EX1**2
H4=H4+EX2**2
50 CONTINUE
H2=H1+DSQRT(PI)*W*H2
H4=H3+PI*(W**2)*H4
H1=(H2/H4)

C * COMPUTE CHANGE IN H
CH=(H1-H)*0.5
H=H+CH

WRITE(NOD,210) ITERH,H,CH

C * COMPUTE THE MEAN SQUARE ERROR
ERRX=ERRF=0.0
D0 55 I=1, (N/2-1)
XT=0.5*DBLE(XR(N/2+I)+XR(N/2-I))
FT=0.5*DBLE(FR(N/2+I)+FR(N/2-I))
EX2=DBLE(FL0AT(I)*DF)
EX1=DBLE(FL0AT(I)*DX)
EX1=((EX1-PZ)/W)**2
EX1=DEXP(-EX1)
XM(N/2+I)=XM(N/2-I)=SNGL(EX1)
EX2=(PI*W*EX2)**2
EX2=DEXP(-EX2)
EX2=(DSQRT(PI)*H*W)*EX2
FM(N/2+I)=FM(N/2-I)=SNGL(EX2)
ERRX=ERRX+(EX1-XT)**2
ERRF=ERRF+(EX2-FT)**2
55 CONTINUE
ERR1=(ERRX+ERRF)/DFL0AT(N/2-1)
XM(N/2)=SNGL(H)
FM(N/2)=SNGL(DSQRT(PI)*H*W)
XM(N) = FM(N) = 0.0

C * COMPUTE CHANGE IN ERROR
CE = ERR1 - ERR
ERR = ERR1

X WRITE(NOD, 230) ERR, CE
IF(ITER, GT, NMAX) GO TO 60
IF(ITERP, GT, NMAX) GO TO 60
IF(DABS(CE).GT.(ERR*ERRTOL*0.01)) GO TO 35

60 IF(ITER, GT, NMAX) WRITE(NOD, 330)
WRITE(NOD, 350)
WRITE(NOD, 250) N, DX, HT, SIG
WRITE(NOD, 260) XBAR, STDV, NSD1
WRITE(NOD, 270) TOL, ERRTOL, DP, DW

C *

C * PRINT OUT PARAMETERS
WRITE(NOD, 350)
WRITE(NOD, 100)
WRITE(NOD, 110) PEX, FP, P
WRITE(NOD, 120) SIG, FV, W
WRITE(NOD, 130) HT, FH, H

C *

C * PRINT OUT MODELS FOR LSF AND OTF
WRITE(NOD, 350)
WRITE(NOD, 340) DX, N, DF
CALL FTLIST2(XM, XI, FM, XI, N, NOD, -1)

C *

C * PLOT OUT DATA AND MODELS
WRITE(NOD, 350)
WRITE(NOD, 180)
CALL PLOT2(XR, KU, XM, KM, N, NW, NOD, -1)
WRITE(NOD, 350)
WRITE(NOD, 170)
CALL PLOT2(FR, KU, FM, KM, NW, NOD, -1)
IF(LOOP.LT.NRUNS) GO TO 99

C *

100 FORMAT(H10,T23,'RESULTS OF PARAMETER ESTIMATION R0'
1,'UTINE'///,T31,'EXACT',T44,'INITIAL',T59,'FINAL',
2/T28,'PARAMETERS ESTIMATES ESTIMATES')
110 FORMAT(H7,T19,'SHIFT:',3X,3(F8.4,6X))
120 FORMAT(H7,T19,'WIDTH:',3X,3(F8.4,6X))
130 FORMAT(H7,T18,'HEIGHT:',3X,3(F8.4,6X))
140 FORMAT(H7,T12,'ORIGINAL SPREAD FUNCTION',
1'DATA BEFORE CENTERING'///,T6,'N',T17,
2'SIGNAL',T34,'N0ISE',T50,'ABS(S+N)'///)
150 FORMAT(2X,14,5X,F12.6,5X,F12.6,5X,F12.6)
160 FORMAT(1H1,T8,'ORIGINAL SPREAD FUNCTION DATA',
1'AFTER CENTERING'///)
170 FORMAT(1H1,T8,'MEASURED MTF ... + + + +',///,T8,
**FUNCTION DEDW1(H,P,W,DX,XR,N)**

C * COMPUTES THE PARTIAL DERIVATIVE OF T0 E, THE MEAN SQUARE

**DIMENSION XR(N)**

**DOUBLE PRECISION DEDW1,H,P,W1,W2,X,XCTR,EX, SUM SUN=0.0**
XCTR=DBLE(FLOAT(N/2)*DX)
D0 10 I=1,N
X=DBLE(DX*FLOAT(I))
X=(X-XCTR-P)/W
EX=DEXP(-X**2)
W1=H*EX**2
W2=DBLE(XR(I))*EX
10 SUM=SUM+(W1-W2)*X**2
DEDW1=(SUM+4.0*H/W)/FLOAT(N)
RETURN
END

FUNCTION DEDP(H,P,W,DX,XR,N)
C* COMPUTES THE PARTIAL DERIVATIVE OF THE MEAN SQUARE
C* ERROR WITH RESPECT TO P, THE SHIFT PARAMETER.
DIMENSION XR(N)
DOUBLE PRECISION DEDP,P1,P2,H,W,X,EX,XCTR
P1=P2=0.0
XCTR=DBLE(FLOAT(N/2)*DX)
D0 10 I=1,N
X=DBLE(FLOAT(I)*DX)
X=(X-XCTR-P)/W
EX=DEXP(-X**2)
P1=P1+H*X*EX**2
10 P2=P2+X*DABS(XR(I))*EX
DEDP=(4.0*H/W)*(P1-P2)/FLOAT(N)
RETURN
END

FUNCTION DEDW2(H,P,W,DX,DF,XR,FR,N)
C* COMPUTES THE PARTIAL DERIVATIVE OF E, THE MEAN SQUARE
C* ERROR WITH RESPECT TO W, THE WIDTH SCALING PARAMETER.
DIMENSION XR(N),FR(N)
DOUBLE PRECISION DEDW2,W1,W2,H,P,XPW,FPW,EX1,EX2
DOUBLE PRECISION RPI,XT,FT,W3,W4
PI=3.14159265358979323846
RPI=DSQRT(PI)
W1=W2=W3=W4=0.0
D0 10 I=1,(N/2-1)
XT=0.5*DBLE(XR(N/2+I)+XR(N/2-I))
FT=0.5*DBLE(FR(N/2+I)+FR(N/2-I))
XPW=DBLE(FLOAT(I)*DX)
XPW=((XPW-P)/W)**2
FPW=DBLE(FLOAT(I)*DF)
FPW=(FPW*W*PI)**2
EX1=DEXP(-XPW)
EX2=DEXP(-FPW)
W1=W1+H*XPW*EX1**2
W2=W2+XPW*EX1*XT
W3=W3+(2.0*FPW-1.0)*EX2*FT
\[ W_4 = W_4 + (2.0 \times FPW - 1.0) \times H \times \text{RPI} \times (EX^2) \]

10 CONTINUE

\[ W_1 = (4.0 \times H/W) \times (W_1 - W_2) \]
\[ W_3 = 2.0 \times \text{RPI} \times H \times (U_3 - W_4) \]
\[ \text{DEDW2} = (W_1 + W_3) / \text{DFLAT(N/2-1)} \]
RETURN
END
APPENDIX B: SUBROUTINES

CONTOUR3
EDGES
FFT24
FTLIST2
FTLIST3
INVFFT
PLOT2
RANDNO
SORT
UNSORT
SUBROUTINE C0NT0UR3(A,N,M,LV,LH,XMX,XMN,NCT,NCH,N0D)
C * STØRED IN A RECTANGULAR ARRAY.
C * THIS SUBROUTINE PLOTS SURFACE CONTOURS FOR VALUES
C * STORED IN A N BY M ARRAY CONTAINING DATA FOR PLOT.
C * A = NAME OF N BY M ARRAY CONTAINING DATA FOR PLOT.
C * N = NUMBER OF ROWS IN A. (LENGTH OF PLOT)
C * M = NUMBER OF COLUMNS IN A. (WIDTH OF PLOT) 126 MAX.
C * LV = AN N BY 3 ARRAY CONTAINING ALPHANUMERIC
C * INFORMATION PRINTED VERTICALLY TO LEFT OF PLOT.
C * LH = A 3 BY M ARRAY CONTAINING ALPHANUMERIC
C * INFORMATION PRINTED HORIZONTALLY BELOW PLOT.
C * XMX = OUTPUT VARIABLE EQUAL TO MAXIMUM ARRAY VALUE.
C * XMN = OUTPUT VARIABLE EQUAL TO MINIMUM ARRAY VALUE.
C * NCT = NUMBER OF CONTOURS DESIRED. NCT = 2,...,71.
C * NCH = OPTION SPECIFYING A SPECIAL CHARACTER TO BE
C * PRINTED ON ALTERNATE CONTOURS. IF NCH = 1, NO
C * SPECIAL CHARACTER IS PRINTED AND CONTOURS ARE
C * NUMBERED CONSECUTIVELY: 0,1,...,9,A,...,Z.
C * NOTE: IF NCH = 1, THEN NCT = 2,...,36.
C * IF NCH = 0, THEN ALTERNATE CONTOURS ARE BLANK.
C * OTHERWISE, ALTERNATE CONTOURS ARE AS FOLLOWS:
C * VALUE OF NCH ...... 38 39 40 41 42 43 44 45 46
C * SYMBOIL PLOTTED .... + - /* \: = $ %
C * N0D = NUMBER ASSIGNED TO THE OUTPUT DEVICE.
C * NOTE: MOST PRINTERS USE DIFFERENT VERTICAL (LINES PER
C * INCH) AND HORIZONTAL (CHARACTERS PER INCH) SPACINGS.
C * FOR A SQUARE PLOT, SELECT DESIRED WIDTH (M) AND THEN
C * COMPUTE LENGTH (N) AS FOLLOWS:
C * N = (M)*(LINES PER INCH)/(CHARACTERS PER INCH)
C * AUTHOR: J H CLARK, JULY 1978
C *
DIMENSION A(N,M),LV(N,3),LH(3,M),KØDE(46),LINE(126)
DATA (KØDE(I),I=1,46)/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,
C 1H8,1H9,1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,
C 1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,
C 1HY,1HZ,1I,1H*,1H+,1H-,1H%,1H*,1H/,1H*,1H**,1H#,1H$,
XMX=XMN=A(1,1)
D0 10 I=1,N
D0 10 J=1,M
XMX=AMAX1(A(I,J),XMX)
10 XMN=AMIN1(A(I,J),XMN)
RANGE=XMX-XMN
IF(NCH.LE.0) NCH=37
IF(NCH.GT.46) NCH=1
D0 40 I=1,N
D0 30 J=1,M
X = A(I,J) - XMN
IX = INT(FLOAT(NCT - 1) * X / RANGE + 0.5)
IF(NCH .EQ. 1) G0 T0 20
IF((IX/2)*2.EQ.IX) KX=IX/2+1
IF((IX/2)*2.NE.IX) KX=NCH
G0 T0 30
20 KX=IX+1
30 LINE(J)=KODE(KX)
40 WRITE(N0D,60) (LV(I,J),J=1,3),(LINE(L),L=1,M)
   WRITE(N0D,80)
   D0 50 I=1,3
50 WRITE(N0D,70) (LH(I,J),J=1,M)
60 F0RMAT(3X,3A1,4X,126A1)
70 F0RMAT(10X,126A1)
80 F0RMAT(///)
   RETURN
   END
C * PROGRAM EDGES ... TO CREATE IDEAL AND NOISY EDGES
C * FOR AN EXPONENTIAL LINE SPREAD FUNCTION MODEL AND
C * WRITE THEM INTO DATA FILES. TO SET UP FILES, ...
C * ON LINE: !SET F:101 DC/FID1;OUT;SAVE
C * !SET F:102 DC/FID2;OUT;SAVE
C * IN BATCH: !ASSIGN F:101,(FILE,FID1), (OUT),(SAVE)
C * !ASSIGN F:102,(FILE,FID2), (OUT),(SAVE)
C * WHERE FID1 AND FID2 ARE THE FILE NAMES FOR THE IDEAL
C * EDGE AND NOISY EDGE RESPECTIVELY. J H CLARK, AUG 1978
C *
DIMENSION E(64),R(64)
READ(100,10) N,DX
READ(100,20) H,P,W,F0G
READ(100,10) NSD,SNR
10 FORMAT(I3,1X,G9.6)
20 FORMAT(4(G9.6,1X))
AVG=0.0
STDV=(H*W+F0G)/SNR
CALL RANDN0(R,AVG,STDV,N,NSD)
D0 30 I=1,N
X=DX*FL0AT(I-N/2)-P
IF(X.LT.0.0) E(I)=H*W*EXP(X/W)/2.0+F0G
IF(X.GE.0.0) E(I)=H*W*(2.0-EXP(-X/W))/2.0+F0G
30 R(I)=ABS(R(I))+E(I))
D0 40 I=1,N/8
J=8*I-7
WRITE(101,50) (E(K),K=J,J+7)
40 WRITE(102,50) (R(K),K=J,J+7)
50 FORMAT(8(F4.2,1X))
X WRITE(108,90)
X D0 60 I=1,N/8
X J=8*I-7
X 60 WRITE(108,80) (E(K),K=J,J+7)
X WRITE(108,90)
X D0 70 I=1,N/8
X J=8*I-7
X 70 WRITE(108,80) (R(K),K=J,J+7)
X 80 FORMAT(8(2X,F4.2),/
X 90 FORMAT(5/)
X WRITE(108,100)
X 100 FORMAT(1H1,T8,'NOISY EDGE ... + + + +','
X 110X,'IDEAL EDGE ... # # # #',/)
X DATA 'KR,K,E,'KE,N,72,108,-1')
STOP
END
SUBROUTINE FFT24(XR, XI, M)

C * COMPUTES THE RADIX 4+2 FAST FOURIER TRANSFORM OF AN
C * EQUISPACEO, WELL ORDERED COMPLEX SEQUENCE WHOSE REAL
C * PART IS STORED IN THE VECTOR XR, AND WHOSE IMAGINARY
C * PART IS STORED IN THE VECTOR XI. AT EXIT THESE TWO
C * COMPONENT SEQUENCES HAVE BEEN REPLACED RESPECTIVELY
C * BY THE REAL AND IMAGINARY PARTS OF THE FOURIER TRANS-
C * FORM OF THE COMPLEX INPUT SEQUENCE FOR 'FREQUENCY'
C * VALUES 0, 1, ..., (2**M)-1
C *
C * USAGE: CALL FFT24(XR, XI, M)
C *
C * XR = VECTOR CONTAINING REAL PART OF INPUT SEQUENCE.
C * XI = VECTOR CONTAINING IMAGINARY PART OF INPUT SEQUENCE.
C * M = INTEGER SUCH THAT THE TRANSFORM IS PERFORMED ON
C * THE FIRST 2**M POINTS OF X(I) = (XR(I), XI(I)).
C *
C * RESTRICTIONS:
C * 1. M MUST BE IN THE RANGE 1, ..., 14. IF M EXCEEDS 14,
C * THE TRANSFORM WILL BE PERFORMED USING THE FIRST
C * 16384 POINTS OF THE INPUT SEQUENCE. IF M IS LESS
C * THAN 1, FFT24 RETURNS CONTROL TO CALLING PROGRAM.
C * 2. XR AND XI MUST BE DIMENSIONED BY CALLING PROGRAM
C * AT LEAST AS LARGE AS 2**M.
C *
C *REFERENCE: GENTLEMEN AND SANDE, 'FAST FOURIER TRANS-
C * FORMS FOR FUN AND PROFIT', PROC. 1966 FALL JOINT
C * COMPUTER CONFERENCE, PP 563-578.
C *
C * AUTHOR: R P BRUMBACK, SEA OPERATIONS DEPARTMENT,
C * AC ELECTRONICS DEFENCE RESEARCH LABS
C * SANTA BARBARA, CALIFORNIA
C *
C * SPECIFICATION STATEMENTS
C INTEGER P, G, R, S
DIMENSION XR(1), XI(1), IR(4, 6), IMX(7)
C * COVERTLY SET THE DO INDICES FOR THE UNSCRAMBLING LOOPS.
EQUIVALENCE(I2MAX, IMX(1)), (I3MAX, IMX(2)), (I4MAX, IMX(3))
EQUIVALENCE(I5MAX, IMX(4)), (I6MAX, IMX(5)), (I7MAX, IMX(6))
EQUIVALENCE(I8MAX, IMX(7))
C * EXIT FOR 1-POINT TRANSFORMS.
IF (M) 32, 32, 31
C * FACTOR NPTS INTO FORM (2**P) *(4**Q).
32 RETURN
31 AM=M
KM=M/2
AKM=AM/2
AAKM=KM
IF (AKM-AAKM) 48, 48, 50
48 P=1
GO TO 49
50 P=2
49 IF (M/2-7)45,45,46
45 Q=M/2
GO TO 47
46 Q=7
47 IF (Q-7)44,43,44
43 P=1
44 N=P*(4**Q)
C * LOAD DATA.
  PIX2=6.28318530
DO 800 I=1,7
800 IMX(I)=1
C * BRANCH FOR 2-POINT TRANSFORMS.
  IF (Q)30,31,30
C * INITIALIZE SUMMING DECIMATOR
30 S=N
C * EXECUTE SCRAMBLED TRANSFORM FOR THE Q FACTORS OF 4.
  DO 2 I=1,Q
C * R=REPLICATION DECIMATOR
  R=S
C * S=SUMMING DECIMATOR
  S=S/4
C * KMAX=REPLICATION LIMIT.
  KMAX=N-R
C * DJ=INVERSED DENOMINATOR FOR TWIDDLE FACTOR.
  DJ=PIX2/FLOAT(R)
C * INITIALIZE DISPLACEMENT LOOP.
  DO 2 J=1,S
C * BRANCH FOR J=1. (TWIDDLE FACTOR IS 1)
  IF (J-1)33,33
C * CALCULATE TWIDDLE FACTORS.
  T1I=SIN(DJ*FLOAT(J-1))
  FC=1.-T1I*T1I
  T1R=SQRT(FC)
  FC=FC+FC-1.
  T2R=FC
  FC=FC+FC
  T3R=T1R*(FC-1.)
  FC=T1I*(FC+1.)
  T2I=T1I*(T1I+T1I)
  T1=K1M=J+KMAX
C * INITIALIZE REPLICATION LOOP
  DO 2 K=J,K1M+R
C * COMPUTE INDICES.
  INDX2=K+S
  INDX3=INDX2+S
  INDX4=INDX3+S
C * LOAD SCRATCH STORAGE.
  AR=XR(K)+XR(INDX3)
  AI=XI(K)+XI(INDX3)
  BR=XR(INDX2)+XR(INDX4)
\[
\begin{align*}
&\text{Bi}=\text{XI}(\text{INDX2})+\text{XI}(\text{INDX4}) \\
&\text{CR}=\text{XR}(\text{K})-\text{XR}(\text{INDX3}) \\
&\text{CI}=\text{XI}(\text{K})-\text{XI}(\text{INDX3}) \\
&\text{DR}=\text{XI}(\text{INDX4})-\text{XI}(\text{INDX2}) \\
&\text{DI}=\text{XR}(\text{INDX2})-\text{XR}(\text{INDX4}) \\
\end{align*}
\]

C * COMPUTE SCRAMBLED TRANSFORM FOR FREQUENCY I, & TWIDDLE.
\[
\begin{align*}
&\text{XR}(\text{INDX2})=\text{CR}+\text{DR} \\
&\text{XI}(\text{INDX2})=\text{CI}+\text{DI} \\
&\text{XR}(\text{INDX3})=\text{AR}-\text{BR} \\
&\text{XI}(\text{INDX3})=\text{AI}-\text{BI} \\
&\text{XR}(\text{INDX4})=\text{CR}-\text{DR} \\
&\text{XI}(\text{INDX4})=\text{CI}-\text{DI} \\
\end{align*}
\]

C * BRANCH IF TWIDDLE FACTOR IS 1.
\[
\begin{align*}
&\text{IF}((J-1)\mod 16)=0, 16 \\
&\text{C }= \text{T1R*XR(INDX2)}-\text{T1I*XI(INDX2)} \\
&\text{XI}(\text{INDX2})=\text{T1R*XI(INDX2)}+\text{T1I*XR(INDX2)} \\
&\text{XR}(\text{INDX2})=\text{CR} \\
&\text{C }= \text{T2R*XR(INDX3)}-\text{T2I*XI(INDX3)} \\
&\text{XI}(\text{INDX3})=\text{T2R*XI(INDX3)}+\text{T2I*XR(INDX3)} \\
&\text{XR}(\text{INDX3})=\text{CR} \\
&\text{C }= \text{T3R*XR(INDX4)}-\text{T3I*XI(INDX4)} \\
&\text{XI}(\text{INDX4})=\text{T3R*XI(INDX4)}+\text{T3I*XR(INDX4)} \\
&\text{XR}(\text{INDX4})=\text{CR} \\
&\text{XR}(\text{K})=\text{AR}-\text{BR} \\
&\text{XI}(\text{K})=\text{AI}+\text{BI} \\
\end{align*}
\]

C * SKIP NEXT LOOP IF THERE IS NO FACTOR OF 2.
\[
\begin{align*}
&\text{IF}(P-1)\mod 2=0, 2 \\
\end{align*}
\]

C * MOVE FACTOR OF 2 UP FOR COMPLETE, SCRAMBLED TRANSFORM.
\[
\begin{align*}
&\text{Klim}=\text{N}-1 \\
&\text{DO } 4 \text{ K}=1, \text{Klim}, 2 \\
&\text{XR}(\text{K})=\text{XR}(\text{K})+\text{XR}(\text{K}+1) \\
&\text{XI}(\text{K})=\text{XI}(\text{K})+\text{XI}(\text{K}+1) \\
&\text{XR}(\text{K}+1)=\text{XR}(\text{K})-\text{XR}(\text{K}+1)-\text{XR}(\text{K}+1) \\
&\text{XI}(\text{K}+1)=\text{XI}(\text{K})-\text{XI}(\text{K}+1)-\text{XI}(\text{K}+1) \\
&\text{IF}((\text{N}-4)\mod 6)=0, 6 \\
&\text{RETURN} \\
\end{align*}
\]

C * NOW TRANSFORM IS IN DIGIT REVERSED ORDER. IF ALL

C * FACTORS ARE 4, USE RADIX 4 UNSCRAMBLING ALGORITHM

C * AT ENTRY 13.
\[
\begin{align*}
&\text{IF}(P-1)\mod 4=0, 4 \\
&\text{R}=7-Q \\
&\text{DO } 6 \text{ I}=1, 6 \\
&\text{IMX}(\text{I})=4 \\
&\text{IR}(1, 6)=0 \\
&\text{IR}(2, 6)=4 \\
&\text{IR}(3, 6)=2 \\
&\text{IR}(4, 6)=6 \\
&\text{DO } 7 \text{ I}=1, 5 \\
&\text{S}=6-I \\
&\text{DO } 7 \text{ K}=1, 4 \\
&\text{IR}(\text{K}, \text{S})=2*\text{IR}(\text{K}, \text{S}+1) \\
\end{align*}
\]
S=1
J=2
8 I=0
   DO 9 I2=1,I2MAX
   DO 9 I3=1,I3MAX
   I3S=IR(I3,2)+IR(I2,1)
   DO 9 I4=1,I4MAX
   I4S=IR(I4,3)+I3S
   DO 9 I5=1,I5MAX
   I5S=IR(I5,4)+I4S
   DO 9 I6=1,I6MAX
   I6S=IR(I6,5)+I5S
   DO 9 I7=1,I7MAX
   I7S=IR(I7,6)+I6S
   DO 9 P=1,J,S
   R=I7S+P
   I=I41
   IF (I-R)>89,9
   C * INTERCHANGE ITH AND RTH ELEMENTS.
   38 AR=XR(I)
      AI=XI(I)
      XR(I)=XR(R)
      XI(I)=XI(R)
      XR(R)=AR
      XI(R)=AI
      CONTINUE
      IF (S-1),41,40,41
   41 RETURN
   C * NOW TRANSFORM IS SCRAMBLED IN BIT REVERSED ORDER.
   C * NEXT, EXECUTE PAIRWISE UNSCRAMBLING.
   40 DO 10 I=1,Q
   10 IMX(I)=4
      IF (Q-Q-6)>8,111,111
   39 Q=Q+1
   DO 11 I=Q,6
   11 IMX(I)=1
   111 IR(2+I)=2
      IR(3+I)=1
      IR(4+I)=3
   DO 12 I=1,5
   DO 12 K=2,4
   12 IR(K+I1)=4*IR(K+I)
   S=N/2
   J=S+1
   C * RETURN TO 8 AND PAIRWISE UNSCRAMBLE.
   GO TO 8
   C * PAIRWISE UNSCRAMBLING ALGORITHM FOR ALL FACTORS = 4.
   13 R=1
   DO 14 I=1,Q
      IMX(I)=3*R+1
   14 R=4*R
   I=0
DO 15 I2=1,4
DO 15 I3=1,13MAX,4
I3S=I3+I2-2
DO 15 I4=1,14MAX,16
I4S=I4+I3S-1
DO 15 I5=1,15MAX,64
I5S=I5+I4S-1
DO 15 I6=1,16MAX,256
I6S=I6+I5S-1
DO 15 I7=1,17MAX,1024
I7S=I7+I6S-1
DO 15 I8=1,18MAX,4096
C * COMPUTE DIGIT REVERSED CONJUGATE OF I.
R=I8+I7S
1=I+1
IF (I-R)42,15,15
C * UNSCRAMBLE ITH ELEMENT
42 AR=XR(I)
AI=XI(I)
XR(I)=XR(R)
XI(I)=XI(R)
XR(R)=AR
XI(R)=AI
15 CONTINUE
RETURN
END
SUBROUTINE FTLIST2(XR, XI, FR, FI, N, NOD, IOP)

C ** THIS SUBROUTINE PRINTS OUT THE VALUES OF A TABULATED
C ** COMPLEX FUNCTION AND ITS DISCRETE FOURIER TRANSFORM
C ** IN SIX LABELED COLUMNS. EACH ROW IS INDEXED.
C ** WIDTH OF PRINTED OUTPUT IS 80 CHARACTER POSITIONS.
C ** XR = VECTOR CONTAINING THE REAL PART OF THE
C ** SPACE DOMAIN FUNCTION.
C ** XI = VECTOR CONTAINING THE IMAGINARY PART OF THE
C ** SPACE DOMAIN FUNCTION.
C ** FR = VECTOR CONTAINING THE REAL PART OF THE
C ** FREQUENCY DOMAIN FUNCTION.
C ** FI = VECTOR CONTAINING THE IMAGINARY PART OF THE
C ** FREQUENCY DOMAIN FUNCTION.
C ** N = THE NUMBER OF POINTS FROM EACH OF THE ABOVE
C ** VECTORS TO BE PRINTED OUT. N SHOULD BE EVEN.
C ** NOD = THE NUMBER ASSIGNED TO THE OUTPUT DEVICE.
C ** IOP = ROW INDEX NUMBERING OPTION
C ** VALUE OF IOP: RESULTANT NUMBERING SCHEME
C ** -1: (1-N/2), (2-N/2), ..., -1, 0, 1, ..., (N/2)
C ** 0: 0, 1, ..., (N/2), (1-N/2), (2-N/2), ..., -2, -1
C ** +1: 1, 2, ..., (N-1), N
C ** J H CLARK, JUNE 1978

DIMA[R(1), XI(N), FR(N), FI(N)
WRITE(NOD, 50)
T=1.0E-6
DO 40 I=1, N
IF(IOP.GE.1) K=1; GO TO 10
IF(IOP.LE.0) K=1-N/2; GO TO 10
K=-1
IF(I.GT.(N/2+1)) K=I-(N+1)
10 IF(ABS(FR(I)).GT.T) GO TO 20
IF(FI(I).GT.T) PH=-1.570796327
IF(FI(I).LT.T) PH=1.570796327
GO TO 30
20 PH=ATAN2(-FI(I), FR(I))
30 IF(ABS(FI(I)).LE.T) PH=0.0
AM=SQRT(FR(I)**2+FI(I)**2)
40 WRITE(NOD, 60) K, XR(I), XI(I), FR(I), FI(I), AM, PH
50 FORMAT(T16, 'SPACE DOMAIN', T31, 'FREQUENCY DOMAIN',
       1X, T6, 'I', T14, 'REAL', T22, 'IMAGINARY', T39, 'REAL',
       2T48, 'IMAGINARY', T62, 'MODULUS', T75, 'PHASE', T76, '////')
60 FORMAT(1X, I5, 2(1X, F11.5), 2X, 4(1X, F11.5))
RETURN
END
SUBROUTINE FTLIST3(XR, XI, FR, FI, AM, PH, N, NOD, IOP)
C * THIS SUBROUTINE PRINTS OUT THE VALUES OF A TABULATED
C * COMPLEX FUNCTION AND ITS DISCRETE FOURIER TRANSFORM
C * IN SIX Labeled COLUMNS. EACH ROW IS Indexed.
C * WIDTH OF PRINTED OUTPUT IS 80 CHARACTER POSITIONS.
C * THIS VERSION DIFFERS FROM 'FTLIST2' ONLY IN THAT THE
C * MODULUS AND PHASE OF THE FREQUENCY DOMAIN FUNCTION ARE
C * RETURNED TO THE USER IN VECTORS AM AND PH RESPECTIVELY.
C * XR = INPUT VECTOR CONTAINING THE REAL PART
C * OF THE SPACE DOMAIN FUNCTION.
C * XI = INPUT VECTOR CONTAINING THE IMAGINARY PART
C * OF THE SPACE DOMAIN FUNCTION.
C * FR = INPUT VECTOR CONTAINING THE REAL PART
C * OF THE FREQUENCY DOMAIN FUNCTION.
C * FI = INPUT VECTOR CONTAINING THE IMAGINARY PART
C * OF THE FREQUENCY DOMAIN FUNCTION.
C * AM = OUTPUT VECTOR CONTAINING THE MODULUS OF THE
C * FREQUENCY DOMAIN FUNCTION.
C * PH = OUTPUT VECTOR CONTAINING THE PHASE OF THE
C * FREQUENCY DOMAIN FUNCTION.
C * N = THE NUMBER OF POINTS FROM EACH OF THE ABOVE
C * VECTORS TO BE PRINTED OUT. N SHOULD BE EVEN.
C * NOD = THE NUMBER ASSIGNED TO THE OUTPUT DEVICE.
C * IOP = ROW INDEX NUMBERING OPTION
C * VALUE OF IOP: RESULTANT NUMBERING SCHEME
C * -1: (1-N/2),(2-N/2),...,-1,0,1,...,(N/2)
C * 0: 0,1,...,(N/2),(1-N/2),(2-N/2),...,-2,-1
C * 1: 1,2,...,(N-1),N
C *
C * J H CLARK, JUNE 1978
C *
DIMENSION XR(N), XI(N), FR(N), FI(N), AM(N), PH(N)
WRITE(NOD,50)
T=1.0C-6
DO 40 I=1,N
IF(IOP.GE.1) K=I; GO TO 10
IF(IOP.LE.-1) K=I-N/2; GO TO 10
K=I-1
IF(I.GT.(N/2+1)) K=I-(N+1)
10 IF(ABS(FR(I)).GT.T) GO TO 20
IF(FI(I).GT.T) PH(I)=-1.570796327
IF(FI(I).LT.-T) PH(I)=1.570796327
GO TO 30
20 PH(I)=ATAN2(-FI(I),FR(I))
30 IF(ABS(FI(I)).LE.T) PH(I)=0.0
AM(I)=SQRT(FR(I)**2+FI(I)**2)
40 WRITE(NOD,60) K,XR(I),XI(I),FR(I),FI(I),AM(I),PH(I)
50 FORMAT(T16,'SPACE DOMAIN',T51,'FREQUENCY DOMAIN',
1/,T62,'I',T14,'REAL',T22,'IMAGINARY',T39,'REAL',
2T48,'IMAGINARY',T62,'MODULUS',T75,'PHASE'/)
60 FORMAT(1X,I5,2(1X,F11.5),2X,4(1X,F11.5))
RETURN
END
SUBROUTINE INVFFT(XR, XI, N)
C THIS SUBROUTINE PERMITS A FORWARD WORKING FAST FOURIER
C TRANSFORM PROGRAM TO BE USED FOR AN INVERSE TRANSFORM.
C XR = VECTOR CONTAINING REAL PART OF OUTPUT FROM
C PREVIOUS FORWARD WORKING FFT ROUTINE.
C XI = VECTOR CONTAINING IMAGINARY PART OF OUTPUT
C FROM PREVIOUS FORWARD WORKING FFT ROUTINE.
C N = NUMBER OF POINTS IN THE SPACE.
C DIMENSION XR, XI TO (N) OR LARGER IN CALLING PROGRAM.
C AT EXIT, XR AND XI CONTAIN THE REAL AND IMAGINARY
C PARTS OF THE SPACE OR TIME DOMAIN FUNCTION CORRESPOND-
C ING TO THE FREQUENCY DOMAIN FUNCTION WHICH WAS INPUT
C TO THE FORWARD FFT ROUTINE.      J H CLARK, MAY 1978
DIMENSION XR(1), XI(1)
AN=FLOAT(N)
DO 10 I=1,N
  XR(I)=XR(I)/AN
10  XI(I)=XI(I)/(-AN)
RETURN
END
SUBROUTINE PLOT2(A1,K1,A2,K2,N,NW,NOD,IOP)

C * THIS SUBROUTINE PLOTS TWO TABULATED FUNCTIONS
C * A1 = VECTOR CONTAINING FIRST FUNCTION
C * K1 = CHARACTER OR SYMBOL PLOTTED FOR A1
C * A2 = VECTOR CONTAINING SECOND FUNCTION
C * K2 = CHARACTER OR SYMBOL PLOTTED FOR A2
C * N = NUMBER OF DATA POINTS IN EACH FUNCTION
C * NW = NUMBER OF SPACES ON WIDTH OF PAGE.  136, MAX.
C * NOD = NUMBER ASSIGNED TO OUTPUT DEVICE
C * IOP = ABSISSA INDEX NUMBERING OPTION
C * VALUE OF IOP:  RESULTANT NUMBERING SCHEME
C * -1:  (1-N/2),(2-N/2),...,1,0,1,...,(N/2)
C *  0:  0,1,...,(N/2),(1-N/2),(2-N/2),...,2-N-1
C *  1:  1,2,...,(N-1),N
C * CHOOSE CHARACTERS FOR K1 AND K2 FROM THIS LIST:
C * A-Z, 0-9, *, /, \, % ( = ' $
C * FOR EXAMPLE, IF * AND $ ARE CHOSEN FOR K1 AND K2,
C * MAIN PROGRAM SHOULD INCLUDE THIS DATA DECLARATION:
C * DATA K1,K2/1H*1HS/
C * ALTERNATELY, ONE CAN READ IN ALPHANUMERIC CHARACTERS
C * VIA THE 'A' FORMAT.

C *

DIMENSION A1(N),A2(N),LINE(130)
DATA KBL,KZ/1H ,1H0/
ISCALE(X)=INT(X*W/HT+0.5)+1
IF(NW.6GT.136) NW=136
JW=NW-7
W=FLOAT(JW-1)
S1=S2=A1(1)
DO 10 I=1,N
  S1=AHINI(S1,A1(I),A2(I))
  S2=AHINI(S2,A1(I),A2(I))
10 CONTINUE

C *

TEST SIGN AND RANGE OF VALUES TO BE PLOTTED
IF(S1.6E.0.) GO TO 25
IF(S2.6GT.0.) GO TO 20

C *

IF ALL VALUES ARE NEGATIVE, ...
HT=-S1
IZ=JW
GO TO 30

C *

IF VALUES ARE BOTH NEGATIVE AND POSITIVE, ...
20 HT=S2-S1
Z=-S1
IZ=ISCALE(Z)
GO TO 30

C *

IF ALL VALUES ARE POSITIVE, ...
25 HT=S2
IZ=1
C * PLOT OUT SCALED FUNCTIONS
30 DO 45 I=1,N
C * SET ABSCISSA INDEX
   K=I
   IF(IOP.GE.1) GO TO 35
   K=I-N/2
   IF(IOP.LE.-1) GO TO 35
   K=I-1
   IF(I.GT.(N/2+1)) K=I-(N+1)
C * STORE BLANKS IN ALL CHARACTER POSITIONS
35 DO 40 L=1,JW
   LINE(L)=KBL
C * SCALE FUNCTIONS AND PRINT EACH LINE
   X=A1(I)
   IF(S1.LT.0.) X=X-S1
   IA1=ISCALE(X)
   LINE(IA1)=K1
   X=A2(I)
   IF(S1.LT.0.) X=X-S1
   IA2=ISCALE(X)
   LINE(IA2)=K2
C * SET POSITION OF ZERO LINE
   LINE(IZ)=KZ
45 WRITE(NUD,50) K,(LINE(L),L=1,JW)
50 FORMAT(1X,I4,2X,130A1)
RETURN
END
SUBROUTINE RANDNO(RN,XBAR,S,N,NSD)
C THIS SUBROUTINE COMPUTES N PSEUDO RANDOM NUMBERS,
C WHICH ARE NORMALLY DISTRIBUTED WITH MEAN XBAR AND
C STANDARD DEVIATION, S. ON FIRST CALL, NSD MUST
C CONTAIN AN ODD INTEGER OF 9 OR FEWER DIGITS.
C THEREAFTER, NSD CONTAINS A UNIFORMLY DISTRIBUTED
C RANDOM INTEGER. THE RANDOM NUMBERS ARE STORED IN
C THE FIRST N POSITIONS OF VECTOR RN, WHICH MUST BE
C DIMENSIONED BY THE CALLING PROGRAM AT LEAST AS
C LARGE AS N. BASED ON IBM SUBROUTINES 'GAUSS' AND
C 'RANIDU'. J H CLARK, APRIL 1978
DIMENSION RN(1)
DO 8 J=1,N
A=0.0
DO 7 I=1,12
IY=NSD*65539
IF(IY)5,6,6
5 IY=IY+2147433647+1
6 Y=FLOAT(IY)
Y=Y*0.4656613E-9
NSD=IY
7 A=A+Y
8 RN(J)=(A-6.0)*S+XBAR
RETURN
END

SUBROUTINE SORT(G,N)
C * THIS SUBROUTINE TAKES THE VALUES OF A FUNCTION, G(X),
C * AND ITS DISCRETE COUNTERPART, G(I), WHICH ARE CENTERED
C * IN ONE DOMAIN (X=0 IS AT I=N/2), AND SORTS THEM FOR
C * SUBSEQUENT FOURIER TRANSFORMATION TO THE OTHER DOMAIN
C * BY FFT. (X=0 GOES TO I=1).
C * G = NAME OF VECTOR CONTAINING VALUES TO BE SORTED,
C * N = NUMBER OF POINTS IN THE SPACE. (LENGTH OF G)
C * NOTE: N MUST BE EVEN.
C * BEFORE SORTING ... -3 -2 -1 0 +1 +2 +3 +4
C * AFTER SORTING .... 0 +1 +2 +3 +4 -3 -2 -1
C * J H CLARK, JUNE 1978
DIMENSION G(N)
B=G(N-1)
C=G(N)
DO 5 I=1,(N/2-1)
  A=B
  B=C
  C=G(I)
  G(I)=G(N/2-1+I)
5  G(N/2-1+I)=A
  G(N-1)=B
  G(N)=C
RETURN
END
SUBROUTINE UNSORT(G,N)
C  * THIS SUBROUTINE TAKES THE VALUES OF A TRANSFORMED
C  * FUNCTION, G(X), REPRESENTED BY ITS DISCRETE
C  * COUNTERPART, G(I), AND RE-SORTS THEM SO THAT THE
C  * FUNCTION IS RELOCATED IN THE CENTER OF THE SPACE.
C  * (X=0 IS SHIFTED FROM I=1 TO I=N/2).
C  * G = NAME OF VECTOR CONTAINING VALUES TO BE SORTED.
C  * N = NUMBER OF POINTS IN THE SPACE. (LENGTH OF G).
C  * NOTE: N MUST BE EVEN.
C  * BEFORE SORTING ... 0 +1 +2 +3 +4 -3 -2 -1
C  * AFTER SORTING ... -3 -2 -1 0 +1 +2 +3 +4
C  * J H CLARK, JUNE 1978
DIMENSION G(N)
B=G(2)
C=G(1)
DO 5 I=1,(N/2-1)
  A=B
  B=C
  C=G(N+1-I)
  G(N+1-I)=G(N/2+2-I)
5  G(N/2+2-I)=A
G(2)=B
G(1)=C
RETURN
END