NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.
Two Dimensional Recursive Digital Filters
for Near Real Time Image Processing

Final Report
NASA Grant N.G 5406
Period September 15, 1979 to December 30, 1980

By

Dr. Winser E. Alexander - Principal Investigator
Dr. David Olson
Mr. Earnest Sherrod

Department of Electrical Engineering
North Carolina Agricultural and Technical State University
Greensboro, North Carolina 27411
Table of Contents

Abstract 2

1.0 Introduction 2

2.0 Background 3

3.0 Mathematical Theory 5

4.0 Stability Analysis 8

5.0 Synthesis 9

6.0 Implementation 11
   6.1 Implementation Considerations 11
   6.2 Transient Response 13
   6.3 Implementation Algorithms 14

7.0 Applications 15
   7.1 Dynamic Range Compression 15
   7.2 Subjective Image Processing 16
   7.3 Bandwidth Optimization 17
   7.4 Interpolation 17
   7.5 Image Registration, Classification and Evaluation 18

8.0 Image Processing Facilities 19

9.0 Image Processing Results 20

12.0 References 24

Appendixes

A. Investigation of Alternative Realization Techniques

B. Implementation Consideration for Two Dimensional Recursive Digital Filters with Product Separable Denominators

C. Imagery Programs

D. Imagery Program Files

E. Papers
Abstract

This program was specifically oriented toward the demonstration of the feasibility of using two dimensional recursive digital filters for subjective image processing applications that require rapid turn around. The concept of the use of a dedicated minicomputer for the processor for this application was also to be demonstrated. The minicomputer used was the HP1000 series E with a RTE II disc operating system and 32K words of memory. A Grinnel 256 X 512 X 8 bit display system was used to display the images.

Sample images were provided by NASA Goddard on a 800 BPI, 9 track tape. Four 512 X 512 images representing 4 spectral regions of the same scene were provided. These images were filtered with enhancement filters developed during this effort and returned to NASA Goddard for further analysis.

1.0 INTRODUCTION

The goal of this program was to develop algorithms to be used in the laboratory on a near real time basis to enhance the capability of a trained observer to obtain geologically interesting information from Landsat satellite imagery. Each Landsat image is recorded with 4 separate spectral bands: 3 in the visible and 1 in the infrared. Thus each scene to be processed is composed of 4 images. Four such images of a scene of interest was provided by NASA Goddard as test images for the program. Each image was provided with 512 rows of 512 pixels per row and 8 bits per pixel.

The objectives of the program were to develop software to implement previously designed two dimensional recursive digital filters on the Department of Electrical Engineering's HP1000 computer system [3]. These filtering algorithms were to be used in an evaluation of the feasibility of their use to
aid the extraction of geologically interesting data from Landsat images. The sample images were to be processed and provided to NASA Goddard for analysis and evaluation.

It was not an objective of this program to approach near real time performance because there was no opportunity to optimize the system hardware for this purpose. A pipeline or array processor would have to be added to improve the computational capability of the system. However, the performance of the system could be used to assess feasibility of further research and development in this area.

2.0 BACKGROUND

Digital filters can be classified as being of two basic types: transform domain filters and time or spatial domain filters. The filtering process is performed in the frequency or transform domain with transform domain filters. The transforms of the signal to be filtered and the impulse response of the desired filter are multiplied to form the transform of the output signal. The inverse transform of the result provides the filtered output signal. Thus any filtering operation requires two transform operations and a multiplication operation. The Discrete Fourier (DFT) is commonly used for most transform domain filtering operations. The Fast Fourier Transform (FFT) algorithm provides a means of implementing the DFT in a computationally efficient manner. Time or spatial domain digital filters do not require a transform process. The filtering is done by taking a weighted average of input and past output values to compute the current output.

There are basically two types of image enhancement: subjective image enhancement and image correction. In subjective image enhancement, the object is to process the image in such a way as to make an improvement in its
appearance or ability to transfer information in some way. If this type of image enhancement is of interest, the user should have available a multitude of general purpose image processing functions. These would include (but not be limited to) low pass filters, high pass filters, low and high frequency enhancement filters, line enhancement filters and line suppression filters. Most of these filtering operations can effectively be accomplished by two dimensional spatial domain digital filters. There is no inherent need to obtain the DFT in the filtering process.

Spatial domain filtering using digital recursive filters offers savings in computation time and core requirements over the use of transform methods to achieve the same filtering process [1]. This is accomplished for many filtering operations with no sacrifice in the quality of the output. Therefore, it is advantageous to use recursive digital filters for those functions for which appropriate filtering algorithms can be developed.

Spatial domain filtering using digital nonrecursive filters offer advantages over both recursive digital filters and FFT digital filters when the number of filter coefficients are relatively small. However, the filters available that meet this requirement are limited. For this reason, nonrecursive digital filters can only be applied to special cases for use in near real time processing. In general, it requires a greater number of coefficients to realize a particular impulse response for nonrecursive digital filters than for recursive digital filters.

Image correction requires a much more complicated filtering process in general than does subjective image processing. The object is to make corrections for distortion, blurring, smearing, etc., that occurred while the image was being formed. This requires the approximation of a filtering function
which is the inverse of the modulation transfer function (MTF) of the imaging process. It is usually necessary to make modifications for the phase as well as the magnitude of the MTF. The resulting filtering requirements are often very complicated and the design of the required digital filter is not a trivial process.

The application of the two dimensional recursive digital filters to image processing and other two dimensional data has been hampered by two problems: stability and synthesis. The synthesis problem is the problem of expressing the two dimensional Z-Transform of the desired impulse response in closed form and thus determining the filter coefficients. The stability problem is important because the recursive filter requires feedback of past output values and therefore can become unstable. Research results obtained on both of these problems by the authors have demonstrated that two dimensional recursive digital filters are very practical for image processing applications [2,3].

3.0 MATHEMATICAL THEORY

The theoretical basis for the two dimensional Z-Transform [4] involves the theory for sample data systems. Given discrete samples of a two dimensional function, \( f(x,y) \) with sampling increments \( X \) and \( Y \) respectively, the Z-Transform for the function is defined by

\[
F(z,w) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(mX,nY)z^{-m}w^{-n}
\]  

(3.1)

If the function is an image, then the problem can be set up so that \( m \) and \( n \) have no negative values and the range of \( m \) and \( n \) is finite. We further restrict the problem to the case where \( X \) and \( Y \) are constants. Then, if we use the notation \( f(m,n) \) to represent \( f(mX,nY) \), we have
as the ZW-Transform for the image function, \(f(m,n)\), which has \((M + 1)\) columns and \((N + 1)\) rows.

Consider the case where we have an input image with samples \(f(m,n)\) and we wish to filter this image to obtain an output image with corresponding samples, \(g(m,n)\). The samples of the impulse response of the desired filter are given by \(h(m,n)\). The range of \(m\) and \(n\) for the output is the same as for the input. Thus, the ZW-Transform of \(g(m,n)\) is given by

\[
G(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} g(m,n)z^{-m}w^{-n} \tag{3.3}
\]

If we restrict the impulse response such that \(m\) and \(n\) cannot be negative (a causal system), we can write the ZW-Transform for the impulse response as

\[
H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n)z^{-m}w^{-n} \tag{3.4}
\]

In general, the ZW-Transform for the impulse response is an infinite series. In order to implement the spatial domain filter, we must find a closed form expression for \(H(z,w)\) such that
The convolution resulting from the
property of the ZW-Transform gives the relationship resulting from the
convolution of \( f(m,n) \) and \( h(m,n) \) which is the filtering process

\[
G(z,w) = H(z,w)F(z,w)
\]

If we use the closed form of \( H(z,w) \) and restrict \( b_{00} \) to be equal to one and
write the resulting equation for a single output value \( g(m,n) \), we obtain the
difference equation for the causal filter

\[
g(m,n) = \sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} f(m-J,n-K) - \sum_{J=0}^{L} \sum_{K=0}^{L} b_{JK} g(m-J,n-K)
\]

If \( L \) is relatively small (in practice, \( L \) is usually less than 10 for recursive
digital filters), equation (3.7) represents a very efficient algorithm for
filtering images. Equations (3.5) and (3.7) may also represent a nonrecursive
filter if all \( b_{JK} \) except \( b_{00} \) are equal to zero.
4.0 **STABILITY ANALYSIS**

Nonrecursive digital filters are inherently stable. Since there is no feedback of past output values, the impulse response has finite duration. Each output value is a finite sum which is always bounded if the input is bounded.

The stability problem for one dimensional digital recursive filters is straightforward. The roots of the denominator polynomial in the closed form of the one dimensional Z-Transform for the filter impulse response function must have magnitudes less than one. Stability analysis is therefore reduced to finding roots of nth degree polynomials with real, constant coefficients [5]. Stability analysis is not straightforward for the two dimensional problem because a two variable polynomial is not generally factorable into distinct roots. When the polynomial in the denominator of the two dimensional Z-Transform of the impulse response is factorable into distinct roots, the stability analysis procedure is the same as for the one dimensional problem.

The two dimensional stability problem is very complicated if the polynomial in the denominator is not factorable into distinct roots [6]. Efforts by other researchers have been directed toward examining regions of roots for two variable polynomials. The developed procedures are computationally feasible only for very simple filters. An alternate method of assessing stability for one dimensional digital recursive filters is to make a state space representation of the filter [7]. Then the filter is stable if the eigenvalues of the state transition matrix all have magnitudes less than one. Previous research has been directed toward developing the two dimensional equivalent of this procedure [2]. A pseudo-state variable representation is chosen because of difficulties in finding a true state space representation [8]. This difficulty is caused by the bivariance of the transfer function and by its causality. The
resulting matrix equation has two pseudo-state transition matrices.

Previous results have shown that the corresponding filter is unstable if any of the eigenvalues of either of these matrices have magnitudes greater than or equal to one or if any of the eigenvalues of the matrix sum have magnitudes greater than or equal to one. Reprints of papers presenting these results are included as in [2].

In practice, these constraints have been found to be very useful in that all tested filters that were known to be unstable were identified as such by the procedure. Conversely, all filters which were known to be stable met the criteria for stable filters and were not identified as unstable.

5.0 SYNTHESIS

The synthesis of nonrecursive digital filters is not a major effort in the proposed research. Several simple nonrecursive digital filter designs may be found in the literature [9]. It would be appropriate to evaluate these designs with regard to application to near real time processing of Landsat satellite data. However, this was not a part of this program.

Often it is possible to express a desired two dimensional recursive digital filter as the product or sum of two one dimensional digital filters. That is the two dimensional Z-Transform of the digital recursive filter can be expressed as the product or sum of two one dimensional Z-Transforms. In either case, the two dimensional synthesis problem is reduced to the synthesis of two one dimensional filters. However, it is not possible to design sum separable or product separable digital recursive filters for all applications. For these applications, the design of the required two dimensional digital recursive filter is considerably more complicated.
Many imaging systems have a natural circular symmetry. In general, the optical transfer function of a circularly symmetric imaging system is circularly symmetric. Also, it is usually desirable to perform image processing where the processing is uniform with respect to direction. The natural consequence is that filters with circularly symmetric impulse response functions are generally very desirable for image processing. The relationship between circular symmetry of the impulse response and the frequency response dictates that the design requirement is for these filters to have a circularly symmetric frequency response [10].

Previous research efforts have led to a synthesis technique which yields two dimensional recursive lowpass, highpass, low frequency boost and high frequency boost recursive digital filters that are very close to being circularly symmetric when the cutoff frequencies are approximately one half the Nyquist frequency [3,11]. Some degradation is observed as the cutoff frequency approaches either the Nyquist frequency or zero.

In the design procedure, the squared magnitude characteristic of the desired circularly symmetric filter is chosen in the Laplace Transform domain. The bilinear transformation is then used to map the squared magnitude characteristic into the two dimensional Z-Transform domain. The pseudo-state space representation for the corresponding two dimensional Z-Transform is formed. The eigenvalues of the matrix sum of the two pseudo-state transition matrices are obtained. These eigenvalues occur in reciprocal pairs. The eigenvalues with magnitudes less than one are then used as roots of a denominator polynomial with distinct roots to form the two dimensional Z-Transform of the desired filter.
Note that this design procedure always ensures a stable filter. Stability analysis is simple because the denominator of the ZW-Transform is a product separable. Also note that no restrictions are placed on the numerator polynomial. That is, it is not necessary for the numerator polynomial to either be product separable, sum separable or minimum phase. Examples of stable two dimensional recursive filter designs are given in [12].

Another problem of interest in image processing is to filter with a one dimensional filter with the orientation of the filter specified and independent of the sampling direction. This type of filter would be useful for enhancing or suppressing linear features, for system noise suppression or for image correction (i.e., linear smear). However, any one dimensional digital recursive filter which is rotated becomes a two dimensional digital recursive filter with associated problems in stability and synthesis. Constraints with regard to stability of rotated digital filters have been developed [13,14]. However, the problems associated with the actual synthesis of rotated recursive digital filters have not been adequately addressed. This is a problem of interest to this research program. However, it was not pursued during this effort.

6.0 IMPLEMENTATION

6.1 Implementation Considerations

Recursive digital filters have many very desirable features that make them advantageous for real time or near real time image processing applications. In the practical application of recursive digital filters, only a small number of rows of the image to be processed are required to be stored in the computer at one time. Three rows of storage plus three rows of storage for each pair of complex poles in the transfer function to be realized are required. Thus a filter with two poles and two zeros would require the storage of the equivalent
of six rows of the input image. A filter with four poles and four zeros would require the storage of the equivalent of nine rows of the input image.

Most image filtering requirements may be met with a filter having no more than four zeros and four poles. Therefore, an algorithm which allows up to four zeros and four poles is practical. Such a filter would still require only slightly more than 9216 storage locations to filter a 1024 by 1024 image. Some additional storage would be required to store the code for the algorithm including its interface to data handling algorithms. Thus it is quite feasible to use recursive digital filters to filter images up to 1024 by 1024 using a 16 bit minicomputer with only 64k words of storage. If in addition a pipeline or array processor is used to implement the recursive digital filter itself, extremely fast processing can be accomplished. In fact, the processing time may be limited by the time required to transfer the data from and back to the storage medium during the actual filtering process.

Recursive digital filters typically require fewer data transfer operations to filter a given image than FFT filters. This is particularly true for very large images. The FFT filtering algorithm requires that the image be transformed by row and then by column. If the image is too large to fit in the computer at one time, the FFT algorithm becomes inconvenient to use for filtering images. One method commonly used to overcome this difficulty is to filter the image in blocks which are small enough to fit into the computer and then fit these filtered blocks back together to form the output image. Considerable overlap of these blocks is required to avoid artifacts due to periodic convolution. Average levels between blocks also have to be adjusted to avoid a checkerboard effect. Another method commonly used is to transform the image by rows, transpose the image and then transform the image by columns [15].
This procedure adds two transpose operations to each filtering operation. The result is that in the practical use of filtering large images, recursive digital filters are very significantly more efficient and require far less time to implement than FFT filters.

Recursive digital filters inherently have nonlinear phase characteristics. This is true because of feedback of past output values. However, linear phase can be obtained by filtering the image twice [3]. The image is filtered starting from the first row, first pixel and ending with the last row, last pixel. Then the image is filtered backward starting with the last row, last pixel and ending with the first row, first pixel. The result is a filter transfer characteristic which is the magnitude squared of the original characteristic. Thus, the filter with four poles and four zeros effectively has eight poles and eight zeros and linear phase when this procedure is used.

6.2 Transient Response

The use of past values of the output to compute the current output value results in the equivalent of long term storage of information about past inputs for recursive digital filters. Thus, such filters have an infinite impulse response (IIR). In addition, the beginning of each scan line in an image represents a transient which can cause very undesirable results if the implemented filter has a long term transient response. If this situation is not handle properly, then two dimensional recursive digital filters will give very poor results. This is particularly true for high frequency boost or highpass filters.

The approach use to minimize this problem is to place the filter in a stable state with an assumed input within the range of the image data. The best assumed input would be the expected value of the input image intensity.
However, this is usually not available. An approximation is obtained by averaging the intensity values of the middle row of the image. The final value theorem [5] is then used to determine the stable state for each of the output stages for the filter. The expected values approximation is then used as the initial condition input for each scan line and the stable state output is used as the initial condition output for each filter stage. Thus, if the initial input is the same as the assumed initial condition, then no transient response occurs.

In practice, the procedure outlined above is simple to implement and adds very computations to the filtering process. However, additional improvement can be obtained by extending the image by using a reflection of future pixel values. Typically as few as 5 values produces very good results such that no transient response artifacts may be observed with most filter designs.

6.3 Implementation Algorithms

Equation 3.7 provides the fundamental algorithm for the two dimensional recursive digital filter. A straightforward approach is to implement the filter directly as provided. However, consideration must be given to roundoff error (the HP1000 computer uses 32 bit floating point arithmetic) and computational efficiency. In addition, the use of complex numbers should be avoided. Therefore, the fundamental stage for the filters was selected to be a second order stage with $L$ equal to 2 in (3.7). Higher order filters may be implemented using multiple stages. This also allows combinations of filters such as a low pass filter for noise removal and a high frequency boost filter for edge enhancement.
In writing the actual algorithm, care was taken to use one dimensional arrays and to avoid transferring data between arrays when possible. Thus a computationally efficient algorithm was developed.

The fact that the HP1000 series E uses a software floating point arithmetic processor and only has a total of 32 K words (64 K bytes) of memory provided a severe hardware limitation. This system has just recently been upgraded to the series E RTE-IVB with an additional 64 K words of memory and a hardware floating point processor. Thus the performance of the image processing software should be very significantly improved with these hardware changes.

In addition to the implementation considerations described above, research was conducted with regard to devising special algorithms which can be used in parallel or pipeline architectures to approach real time image processing. Appendix A and B provide details on this effort. Appendix C and D gives documentation of the software developed.

7.0 APPLICATIONS

7.1 Dynamic Range Compression

Electro-optical sensors respond to reflected or emitted radiation. A typical electro-optical imaging system uses a single detector or an array of detectors in a scanning mode to form the image. If the signal of interest is the reflected radiation such as is the case for visible imaging systems, the detected signal is made up of two components: the illumination component and the reflection component. Infrared sensors typically detect radiation emitted by objects. It is typical that the available dynamic range of electro-optical imaging systems is several orders of magnitude. On the other hand, display systems are usually limited to at most two orders of magnitude and human observers can only detect approximately 50 different intensity levels [16].
Therefore, it is not possible to directly display all information obtained in many images.

The illumination component of optical images or the overall background radiation for infrared images generally has low spatial frequency content but may have a wide dynamic range [17]. This is the case where shadows exist in optical imagery or hot spots occur in infrared imagery. The reflected component or the emitted component of the signal is usually of priority interest and generally has higher spatial frequency content. This signal is formed by the different emissivity or reflectance of each item in the image.

The detected signal is therefore a product of the illumination or background radiation and the reflectance or emissivity at each point in the image. Homomorphic filtering using spatial domain digital filters provides an effective means of dynamic range compression by providing the capability to suppress the lower frequency component of the signal (illumination or background radiation component) and enhancing the higher frequency component of the signal (reflected or emitted component of the signal) [18]. This procedure is accomplished by taking the logarithm of the input signal, filtering with a high frequency boost filter and exponentiating the resulting output.

7.2 Subjective Image Processing

A simple design procedure can be used to allow an untrained operator to design digital filters for subjective image processing. For example, a low pass or high pass filter may be specified by the cutoff frequency and the number of poles desired [3]. A high frequency enhancement filter or a low frequency boost filter may be specified by a break frequency and the magnitude of the boost. Thus, the user does not have to learn filter theory or be concerned with signal to noise considerations, etc. to design the desired filter. This is a very
valid approach for subjective image processing because decisions about the type of filter desired are usually made based upon experience. Thus the user should be provided with several options which can be implemented with a minimum of effort and without special training. Recursive digital filters are well suited for this application.

7.3 Bandwidth Optimization

If an imaging system is used in an interactive mode, digital filters can be used to effectively change the bandwidth of the imaging system to meet a particular application. Thus under low signal to noise operating conditions, the operator can decrease the bandwidth of the system in an attempt to improve his ability to discern details of an object of interest. This can be accomplished with spatial digital filters simply by changing filter coefficients. No change in hardware is required.

7.4 Interpolation

Often it is desired to change the size of an image in image presentation or display operations. This usually requires a change in the number of rows or columns of the subject image. In changing the size of the image, the sampling theorem must be considered. Artifacts in output images after the use of a simple interpolation scheme are quite often due to aliasing.

An image is usually stored in discrete form. That is, only samples of the image are available in the form of pixel elements. Thus interpolation really involves reconstructing the image to a continuous form and then resampling at the new desired intervals. The ideal interpolation algorithm would involve a reconstruction filter based upon the sampling theorem [5] and a sampling algorithm to resample the image at the desired intervals. However, it is not computationally feasible to use this approach. Therefore, it is common practice
to use a simple algorithm such as nearest neighbor, bilinear or constrained polynomial interpolation for image processing requirements. These algorithms all result in aliasing when either the number of rows or columns is decreased. If the number of rows or columns is increased, these algorithms add undesired noise to the output image which is image dependent [16].

A means of improving the results of these interpolation schemes is to use prefiltering to avoid aliasing and/or post filtering to remove undesirable additive noise. The results using this procedure can be made to be very close to the ideal reconstruction filter interpolator with the proper combination of filtering and a simple interpolation algorithm. The use of recursive digital filters which have been shown to be considerable more efficient computationally than the FFT algorithm for image processing makes this procedure feasible. For example, the bilinear interpolation algorithm can effectively be combined with an antialiasing filter when needed to give results which are very significantly improved over the use of the bilinear interpolation algorithm alone. Computationally, such a scheme would compare very favorably to a constrained polynomial interpolation algorithm and would give superior results for many images.

7.5 Image Registration, Classification and Evaluation

Image registration, classification and evaluation schemes generally do not take advantage of digital filtering. In general, relatively simple schemes are used with human interaction playing a very important role. This is partially true because of the inconvenience of using filtering with current techniques which employ the FFT algorithm and partially because the feasibility of using spatial filtering to improve image registration, classification and evaluation has not been demonstrated.
Two dimensional recursive digital filters have advantages which make them very attractive for use in exploring the feasibility of using spatial filtering to improve these procedures. The filters can be designed with only a small number of parameters specified by the user (usually no more than two parameters must be specified). The actual filtering process requires significantly fewer computations and data transfers than the FFT algorithm and image size is not constrained to power of 2. Thus, very fast turnaround can be achieved.

With very fast turnaround and with the availability of various types of filters, the exploration of the use of filtering for image registration, classification and evaluation becomes far more practical. If spatial filtering proves beneficial, then the implementation can be done with only a small sacrifice in time and without the use of a very large computer system. Thus two dimensional recursive digital filters may be very beneficial to image registration, classification and evaluation. In practice, the use of such filters may prove to be very beneficial in automating these vital procedures.

3.0 IMAGE PROCESSING FACILITIES

The Department of Electrical Engineering at A & T State University has a HP1000 Series E computer system and the University has a DEC10 computer system. Both of these computer systems were used with this program.

The HP1000 is a 16 bit minicomputer system with 32k words of core, a 14.6 megabyte disk drive and a 9 track tape drive. The core will be extended to 192k bytes and the CPU is being upgraded to series E with the RTE- IVB operating system. This upgrade will be completed by the end of February, 1981. The 9 track system can be used to transfer data from and to the DEC10 computer system. A Grinnell Model GMR-27 display image system is also available. This display can display an image with 256 rows and 512 pixels per row with 8 bit accuracy.
Plans also include additional graphics capability and a full color display system.

The DEC10 computer system is an interactive system with a 36 bit word length and double precision arithmetic capability. Thus, it can be used for stability analysis and filter synthesis and evaluation. The current DEC10 system consist of a KL-10 central processor, 512k words of memory, 2 self loading tape drives a communications controller for up to 96 asynchronous dial drives.

The Department of Electrical Engineering also has a HP2648 graphics terminal which is connected to the HP1000 computer. This graphics terminal is used for interactive stability analysis and filter synthesis.

9.0 IMAGE PROCESSING RESULTS

The lack of a hard copy output capability presents considerable difficulty with regard to including actual Landsat images or the processed results in this report. A HP9872 plotter is connected to the HP1000 computer and may be used to plot frequency contour and perspective plots of the actual filter used in the image processing examples. However, a 35-mm camera was used to photograph the Grinnell display screen to obtain the examples that follows.

Figure 1 is the frequency perspective plots of a 5 magnitude High Boost Filter with 0.2 cutoff frequency. Figure 2 is the frequency contour plot of the same filter. Figure 3 is file number three (3) of the Landsat Imagery tape received from NASA. Figure 4 shows the results of processing images with the filter of Figure 1 and then mapping between minimum and maximum logarithmically.
Figure 1. Perspective plot 5x-0.2 High Boost Filter

Figure 2. Contour plot 5x-0.2 High Boost Filter
Figure 4. Enhanced Landsat file-3 Image
12.0 REFERENCES


APPENDIX A

INVESTIGATION OF ALTERNATIVE REALIZATION TECHNIQUES

Another aspect of the research conducted under this contract was that of investigating alternative realization techniques for not only the filter designs chosen, but also for a more general class of filters as well. This investigation although as yet incomplete has resulted in some interesting conceptual reformations of the filter realization problem [1], as well as the suggestion of possibly more computationally efficient algorithms for obtaining the filter solutions.

The typical approach taken in realizing recursive 2 D filters is one of processing the filtered output directly using the forward and backward difference equation formulations of the filter. This approach requires that one either already know the initial condition or boundary condition state of the filtered output (which generally is not the case), or that one uses various statistical estimates of what these boundary states might be in order to begin the recursion. In either case the direct use of the difference equations may not result in a minimum number of arithmetic operations being performed in obtaining a filtered solution [2,3,4].

The approach taken in this aspect of the conducted research was one of formulating the complete set of simultaneous linear algebraic equations to be solved in order to obtain a solution which satisfies the 2 D difference equation description of the filter. This serves to give one a complete description of the constraints which must be satisfied by the filtered solution with or without boundary conditions imposed on the problem.

The class of filters considered were those which possess a rational transfer function. Such a filter may be represented by its bivariate difference equation
written in tensor form as:

\[ b_{ij} g_{p+i, q+j} = a_{ij} f_{p+i, q+j} \]  \hspace{1cm} (1)\]

where \(1 \leq p \leq N, 1 \leq q \leq M, -m \leq i \leq m, -m \leq j \leq m\); and the double appearance of an indice on a given side of the equality implying the usual tensor notation for a summation over the specified range of that indice. The so called finite duration impulse response filter (FIR) is one which satisfies \(b_{00}=1\) with all other \(b_{ij}=0\); whereas the infinite duration impulse response filter (IIR) is one which allows nonzero \(g_{ij}\) for \(i,j \neq 0\). A more formal tensor expression for (1) is given by:

\[ B^{k1}_{pq} g_{k1} = A^{k1}_{pq} f_{k1} \]  \hspace{1cm} (2)\]

where \(1 \leq k \leq N, 1 \leq l \leq M\), and the non-zero components of the coefficient tensors given by \(A^{k1}_{pq} = a_{k-p, l-q}\) and \(B^{k1}_{pq} = b_{k-p, l-q}\); for \(-m \leq k-p \leq m\) and \(-m \leq l-q \leq m\). The 2 D filtering operation requires that one determine all the elements \(g_{pq}\), given all the coefficients \(a_{ij}, b_{ij}\), and the input array \(f_{pq}\).

A solution to equation (2) will exist and be unique if there exists an inverse of the tensor \(B^{k1}_{pq}\), say \(C^{pq}_{uv}\), with \(1 \leq u \leq N, 1 \leq v \leq M\). For such a case, the filtered solution would then be given by:

\[ g_{uv} = C^{pq}_{uv} A^{k1}_{pq} f_{k1} \]  \hspace{1cm} (3)\]

Tensor equation (2) can also be interpreted as a matrix equation with \(A^{k1}_{pq}\) and \(B^{k1}_{pq}\) taken as \(N \times M\) dimensional coefficient matrices with row index "pq", column index "k1"; and \(g_{k1}\) and \(f_{k1}\) interpreted as column vectors. Viewing equation (2) as such a matrix equation reveals the enormity of the computer storage problem encountered in attempting a solution, for if both \(N\)
and M were typically of the order to say 512 (for a 512 by 512 pixel array) then 2^{36} memory locations would be required for the tensor of matrix $B_{pq}^{kl}$ alone.

The matrix equation interpretation of equation (2) also reveals the following characteristics of the coefficient matrix $B_{pq}^{kl}$ for these selected digital filters:

(a) For the "Quarter Plane" digital filter, $B_{pq}^{kl}$ is a triangular matrix. Hence, the solution for the filtered array $g_{kl}$ requires no inversion of the coefficient matrix. By a simple back substitution process, starting at one corner of the array and proceeding by rows or columns, the filtered array may be computed provided that the iteration process is numerically stable.

(b) For the "Symmetric" digital filter, with filter coefficients symmetric with respect to any diagonal passing through the central element $b_{00}$ of the mask $b_{ij}$, the coefficient matrix $B_{pq}^{kl}$ is symmetric.

Among the interesting results developed during the tenure of this research was the fact that for square arrays $N=M$, and filters with $a_{00}, b_{00} \neq 0$; the filtering problem given by equation (2) is also expressible as a matrix equation involving only $N$ by $N$ dimensional sparse coefficient matrices given by:

$$LGR + \sum_{k=-m, k \neq 0}^{m} S_k G T_k = c PFQ + c \sum_{k=-m, k \neq 0}^{m} S_k F U_k$$

(4)

where $c = a_{00}/b_{00}$, the matrix $G = (g_{pq})$ is the filtered array, $F = (f_{pq})$ is the input array; and the nonzero components of the coefficient matrices $L, R, P, Q, S_k, T_k, -T_k$, and $U_k$ are given by:
(i) For $p, q$ such that $-m \leq q-p \leq m$:

$$L_{pq} = b_{q-p,0}/b_{00}; \quad R_{pq} = b_{0,q-p}/b_{00}; \quad P_{pq} = a_{q-p,0}/a_{00}; \quad Q_{pq} = a_{0,q-p}/a_{00};$$

$$T_{kpq} = b_{k,p-q}/b_{00} - b_{k,0}/b_{00}; \quad U_{kpq} = a_{k,p-q}/a_{00} - a_{k,0}/a_{00}.$$  

(ii) And finally, for $p, q$ such that $q-p = k$: $S_{kpq} = 1$.

The reduction in the dimensions of the coefficient matrices shown in equation (4) is one of the practical reasons why one would prefer to solve that expression for the filtered output rather than equation (2). The coefficient matrices in (4) also have other appealing properties in that both $L$ and $R$ are symmetric matrices, all of the matrices have the "bandtype" structure in that they have but one distinct element per respective major or minor diagonal, and all of the matrices are relatively sparse (many zero elements).

Unfortunately expression (4) is not generally solvable by using linear methods due to the fact that one cannot combine those matrices which premultiply the unknown matrix $G$ (i.e., $L$ and the $S_k$), or those matrices which postmultiply $G$ (i.e., $R$ and the $T_k$). It should be noted, however, that for those cases in which equation (4) is not solvable for $G$ using linear methods, this does not imply that there exists no unique solution. It is equation (2) that dominates in that it is always solvable if (4) is solvable, but (2) may still be solvable even if (4) is not linearly solvable. Hence, from the standpoint of linear analysis (2) possesses more potential in solving for $q_{pq}$ than equation (4).

There is an important class of filters for which equation (4) is linearly solvable, and this class is the set of filters which are product separable. The coefficients involved in product separable filters have the properties:

$$a_{k,p-q/a_{00}} = a_{k,0}/a_{00} = 0 \quad b_{k,p-q/b_{00}} = b_{k,0}/b_{00} = 0.$$
Hence, the matrices $T_k$, and $U_k$ are all identically zero and equation (4) reduces to:

$$LGR = c \cdot PFQ$$

and the solution for the filtered output $G$ given by:

$$G = L^{-1}(cPFQ) R^{-1}$$

At first glance it would appear the computation of the filtered output array $G$ is still a formidable task due to the required inversions $L^{-1}$, and $R^{-1}$; however both $L$, and $R$ are Toeplitz matrices and can be inverted efficiently [5], hence we have our first instance of a possibly more efficient algorithm for obtaining filter solutions.

Adding additional restrictions, it has also been determined that if the filter is both product separable as well as symmetric then the coefficient matrices $L$ and $R$ can be further decomposed to give equation (5) the equivalent expression:

$$L G R = c \cdot PFQ$$

Expression (5) is then solvable for $G$ using a minimum number of arithmetic operations without requiring the inversion of $L$ and $R$, provided that the intermediate results are numerically stable.

Finally, for the filter problem described by expression (4), iterative methods of solution such as:

$$G(n+1) = L^{-1} \sum_{k=-m}^{m} S_k G(n) T_k R^{-1} + H$$

where $H = L^{-1} (cPFQ + c \sum_{k=-m}^{m} S_k F U_k) R^{-1}$

as suggested as possible techniques to be applied to obtain filter solutions for those filters which do not satisfy the restrictions required for expressions (5), (6), and (7). The investigation of the convergence of such iterative solution techniques is the subject of current and future research.
REFERENCES


Appendix B

Implementation Consideration for Two Dimensional Recursive Digital Filters with Product Separable Denominators.

Introduction

Consideration is given to the implementation of two dimensional digital recursive filters that have transfer functions with product separable denominators. This structure is of particular importance to this program because the design technique used for the design of approximately circularly symmetric filters results in a transfer function with a product separable denominator. We seek to derive a computationally efficient structure that may also lend itself to implementation with the use of a pipeline or array processor.

Transfer Function

The bivariate Z-transform for the structure of interest is given by

\[ H(Z,W) = \frac{ \sum_{J=0}^{2} \sum_{K=0}^{2} a_{JK} Z^{-J} W^{-K} }{ \sum_{J=0}^{2} \sum_{K=0}^{2} b_{JK} Z^{-J} W^{-K} } = \frac{N(Z,W)}{D(z,w)} \]  

We have assumed that \( L=2 \) for a single second order filter stage. We also assume that the denominator polynomial, \( D(z,w) \) can also be represented as

\[ D(Z,W) = \left[ \sum_{J=0}^{2} c_{J} Z^{-J} \right] \left[ \sum_{K=0}^{2} d_{K} W^{-K} \right] \]  

(2)
We can implement $H(z,w)$ in cascade form

$$H(z,w) = H_1(z,w)H_2(z,w)H_3(z,w)$$

(3)

where

$$H_1(z,w) = \sum_{J=0}^{2} \sum_{K=0}^{2} a_{JK} z^{-J} w^{-K}$$

(4)

$$H_2(z,w) = \frac{1}{2} \sum_{J=0}^{2} c_J z^{-J} ; \quad c_0 = 1$$

(5)

$$H_3(z,w) = \frac{1}{2} \sum_{K=0}^{2} d_K w^{-K} ; \quad d_0 = 1$$

(6)

In direct form, the corresponding difference equations are given by

$$x_1(m,n) = \sum_{J=0}^{2} \sum_{K=0}^{2} a_{JK} f(m-J,n-K)$$

(7)

$$x_2(m,n) = x_1(m,n) - c_1 x_2(m-1,n) - c_2 x_2(m-2,n)$$

(8)

$$g(m,n) = x_2(m,n) - d_1 x_3(m,n-1) - d_2 x_3(m,n-2)$$

(9)

Note that this form only requires 13 multiplies and 13 adds as compared to 17 multiplies and 17 adds for the direct form associated with (1). The block diagram for this implementation is given below.
PROGRAM NAME: NASA

TYPE: Transfer

PROGRAMMER: W.E. ALEXANDER

Source: Reloc:

FUNCTION: This transfer offs and RP's all necessary modules for Image Processing; mounts cartridge 23 and runs NASA 1.

FROM RTE: Run NASA

Modules Called:

SHOW
BLDWF
BLDIM
WTAPE
DSPLY
CURSR
FDIGN
STABI
DPLAM
FILTR
LFLTR
HFLTR
RESIZ
IMAGE
DINTP
NOISE
FIRO

Modules Run: NASA1

Subroutines Called:
PROGRAM NAME: NASA1

PROGRAMMER: W.E. ALEXANDER

SOURCE: &NASA1

FUNCTION: This Program is the father program for the Image Processing from which the major modules are selected.

Modules Called:

DSPLY
FDIGN
FILTR
RESIZ
SHOW
BLDIM
NOISE

Modules Run:

Subroutines Called:

FILL
PROGRAM NAME: DSPLY

PROGRAMMER: DAVE JOHNSON

Source: &DSPLY  Reloc: %DSPLY.

FUNCTION: This program Displays an Image on the Grinnell Image Display System GMR-27.

Modules Called:

   SCROL
   CURSR

Modules Run:

Subroutines Called:

   WLINE
   RLINE
   DRIVR
   RESET
   MOVEC
PROGRAM NAME: FDIGN

PROGRAMMER: E.E. SHERWOOD

Source: &FDIGN, &FDIG1

FUNCTION: This program designs, stability tests and displays a filter on either HP-2648G or on the Grinnell GMR-27.

Modules Called:

STABI
DPLAM
FIRO
PLOTV

Data File Created:

COEFFS
DATA1

Subroutines Called:

LPFLT
BPFLT
BSTFT
TDLTF
ROTAE
FIR
PROGRAM NAME: STABI

PROGRAMMER: E.E. SHERROD

Source: &STABI  
Reloc: %STABI

FUNCTION: This Program evaluates the Recursive Filter Stability Characteristics.

Modules Called:

Subroutines Called:

   STABT
   PRTN
PROGRAM NAME: DPLAM
TYPE: Program
Source: &DPLAM, &DPLA1
Reloc: %DPLAM, %DPLA1

PROGRAMMER: E.E. SHERROD

FUNCTION: This program displays the Filter Characteristics.

Modules Called:
COEFFS
DPLA1

Subroutines Called:
ZWC
CONTR
SET3D
PLT3D
SET2D
PLT2D
PROGRAM NAME: FIRO

PROGRAMMER: E.E. SHERROD

Source: &FIRO

FUNCTION: This program designs Non-Recursive FIR Filters.

Modules Called:

Subroutines Called:

    BESJ
    BESIO
PROGRAM NAME: PLOTV  
TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &EES3  
Reloc: %PLOTV

FUNCTION: This program displays Filter Characteristics on the Grinnell Display GMR-27.

Modules Called:

DATA1

Subroutines Called:

DVECT
PROGRAM NAME: FILTR                      TYPE: Program
PROGRAMMER: E.E. SHERROD
Source: &FILTR                              Reloc: %FILTR
FUNCTION: This program schedules Linear or Homorphic filtering of Images.

Modules Called:
  LFLTR
  HFLTR
  SHOW
  BLDWF

Subroutines Called:
PROGRAM NAME: BLDWF

PROGRAMMER: DAVE JOHNSON

Source: &BLDWF

FUNCTION: This program creates and maintains an Image work file named WF0000 with pixel values stored as 15-bit real numbers.

Modules Called:

DIREC
WF0000

Subroutines Called:

ICMPW
PROGRAM NAME: LFLTR                 TYPE: Program
PROGRAMMER: E.E. SHERROD
Source: &LFLTR                        Reloc: %FILTR
FUNCTION: This program does Linear Filtering using Spatial Domain Recursive Digital Filters.

Modules Called:
    COEFFS

Subroutines Called:
    READL
    RITLN
    FILTR
    WFINT
    CLSWF
PROGRAM NAME: HFLTR

PROGRAMMER: E.E. SHERROD

Source: &HFLTR

FUNCTION: This program performs Homomorphic Filtering using Spatial Domain Recursive Digital Filters.

Modules Called:

COEFFS

Subroutines Called:

WFINT
READL
RITLN
HFILT
CLSWF
PROGRAM NAME: RESIZ

PROGRAMMER: W.E. ALEXANDER and RICHARE MuORE

Source: &RESIZ

FUNCTION: This program allows the user to scale an image and change an image from 8-bits to 15-bits and vice versa. The resizing of an Image is being developed.

Modules Called:
LFLTR
DINTP
TRMGN
LBRSZ
BLDWF

Subroutines Called:
TRMGN
BLANX
SPCHR
CKFLD
WFINT
READL
XYFLT
CLSWF
READL
RITEI
PROGRAM NAME: SHOW          TYPE: Program
PROGRAMMER: DAVE JOHNSON
Source: &SHOW          Reloc: %SHOW

FUNCTION: This program displays an image from the work
file onto the Grinnell System GMR-27.

Modules Called:

WFO000

Subroutines Called:

READL
WLINE
CLSWF
PROGRAM NAME: BLDIM

PROGRAMMER: DAVE JOHNSON

Source: &BLDIM

FUNCTION: This program constructs an 8 or 15-bit image from magnetic tape, disc, GMR-27 display or work file.

Loadfile: LBLDIN

Reloc: %BLDIN

Type: Program

Modules Called:

   WF0000
   DIREC

Subroutines Called:

   MVW
   ROT8
   DCODE
   DRIVR
PROGRAM NAME: NOISE  TYPE: Program
PROGRAMMER: E.E. SHERROD
Source: &NOISE  Reloc: %NOISE

FUNCTION: This program adds Gaussian Noise to an image with user-defined Mean and Standard Deviation from a Gaussian Noise disc file.

Modules Called:
   BLDWF

Subroutines Called:
   READL
   RITEL
   CLSWF
LOAD FILES

Table of Content: Page 1

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDIREC</td>
<td>1</td>
</tr>
<tr>
<td>LWTAPE</td>
<td>1</td>
</tr>
<tr>
<td>LDPLAM</td>
<td>1</td>
</tr>
<tr>
<td>LPLTOV</td>
<td>1</td>
</tr>
<tr>
<td>LFDIGN</td>
<td>2</td>
</tr>
<tr>
<td>LBLDIM</td>
<td>2</td>
</tr>
<tr>
<td>LLFTR</td>
<td>2</td>
</tr>
<tr>
<td>LHFTR</td>
<td>3</td>
</tr>
<tr>
<td>LSHOW</td>
<td>3</td>
</tr>
<tr>
<td>LFIRO</td>
<td>3</td>
</tr>
<tr>
<td>LIMAGE</td>
<td>4</td>
</tr>
<tr>
<td>LRESIZ</td>
<td>4</td>
</tr>
<tr>
<td>LDINTP</td>
<td>4</td>
</tr>
<tr>
<td>LDINTP</td>
<td>5</td>
</tr>
<tr>
<td>LBLDWF</td>
<td>5</td>
</tr>
<tr>
<td>LDSPLY</td>
<td>5</td>
</tr>
<tr>
<td>LCUSR</td>
<td>5</td>
</tr>
<tr>
<td>NASA</td>
<td>6</td>
</tr>
</tbody>
</table>
## Table of Content

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NASA1</td>
<td>7</td>
</tr>
<tr>
<td>&amp;DSPLY</td>
<td>9</td>
</tr>
<tr>
<td>&amp;CURSR</td>
<td>12</td>
</tr>
<tr>
<td>&amp;ICMPW</td>
<td>13</td>
</tr>
<tr>
<td>&amp;RESET</td>
<td>13</td>
</tr>
<tr>
<td>&amp;RLINE</td>
<td>14</td>
</tr>
<tr>
<td>&amp;MOVEC</td>
<td>15</td>
</tr>
<tr>
<td>&amp;IMAGE</td>
<td>16</td>
</tr>
<tr>
<td>&amp;SPAGE</td>
<td>23</td>
</tr>
<tr>
<td>&amp;RESIZ</td>
<td>24</td>
</tr>
<tr>
<td>&amp;ROT8</td>
<td>33</td>
</tr>
<tr>
<td>&amp;TRMGN</td>
<td>34</td>
</tr>
<tr>
<td>&amp;LBRSZ</td>
<td>37</td>
</tr>
<tr>
<td>&amp;WF INT</td>
<td>39</td>
</tr>
<tr>
<td>&amp;DINTP</td>
<td>42</td>
</tr>
<tr>
<td>&amp;WTAPE</td>
<td>44</td>
</tr>
<tr>
<td>&amp;EES3</td>
<td>47</td>
</tr>
<tr>
<td>&amp;DPLAM</td>
<td>48</td>
</tr>
<tr>
<td>&amp;DPLA1</td>
<td>52</td>
</tr>
<tr>
<td>&amp;FDIGN</td>
<td>60</td>
</tr>
<tr>
<td>&amp;BLDIM</td>
<td>70</td>
</tr>
<tr>
<td>&amp;LFLTR</td>
<td>76</td>
</tr>
<tr>
<td>&amp;HFLTR</td>
<td>82</td>
</tr>
<tr>
<td>&amp;SHOW</td>
<td>88</td>
</tr>
<tr>
<td>&amp;FIRO</td>
<td>89</td>
</tr>
<tr>
<td>&amp;WINDO</td>
<td>92</td>
</tr>
<tr>
<td>&amp;BESJO</td>
<td>92</td>
</tr>
<tr>
<td>&amp;BESJ</td>
<td>93</td>
</tr>
<tr>
<td>&amp;BLDF</td>
<td>95</td>
</tr>
<tr>
<td>&amp;SCROL</td>
<td>98</td>
</tr>
<tr>
<td>&amp;WLINE</td>
<td>99</td>
</tr>
<tr>
<td>&amp;DRIVE</td>
<td>100</td>
</tr>
<tr>
<td>&amp;FDIG1</td>
<td>102</td>
</tr>
<tr>
<td>&amp;STABI</td>
<td>105</td>
</tr>
<tr>
<td>&amp;FILTR</td>
<td>117</td>
</tr>
</tbody>
</table>
LDIREC T=00004 IS ON CR00025 USING 00002 BLKS R=0002

0001 : PU, IMDIRC: IM: 23: 4: 100

LWTAPE T=00004 IS ON CR00022 USING 00002 BLKS R=0010

0001 : LG, 3
0002 : SV, 0
0003 : OF, WTAPE
0004 : PU, WTAPE
0005 : MR, ZWTAPE
0006 : MR, ZROTS
0007 : MR, ZICMPW
0008 : MR, ZTMGN
0009 : RU, LOADR, 99, OG, .., 2
0010 : SP, 10G: : 3
0011 : TR

LDPLAM T=00004 IS ON CR00022 USING 00002 BLKS R=0010

0001 : SV, 0
0002 : OF, DPLAM
0003 : PU, DPLAM
0004 : LG, 3
0005 : MR, ZDPLAM
0006 : MR, ZDPLA1
0007 : RU, LOADR, 99, OG
0008 : SP, 10G: : 3
0009 : OF, 10G
0010 : RP, 10G: : 3
0011 : :

LPLOTV T=00004 IS ON CR00022 USING 00002 BLKS R=0011

0001 : SV, 0
0002 : OF, PLOTV
0003 : PU, PLOTV
0004 : LG, 3
0005 : MR, ZEES3
0006 : MR, ZDRIVR
0007 : RU, LOADR, 99, OG
0008 : SP, 10G: : 3
0009 : OF, 10G
0010 : RP, 10G: : 3
0011 : :
LFDIGN T-00004 IS ON CRO0022 USING 00002 BLKS R-0011

0001 : SV, 0
0002 : OF, FDIGN
0003 : PU, FDIGN
0004 : LG, 3
0005 : MR, ZFDIGN
0006 : MR, YFDIGN
0007 : RU, LGADR, 99, 1G
0008 : PU, LG::3
0009 : SP, LG::3
0010 : OF, LG::3
0011 :
0012 :

LBLDIM T-00004 IS ON CRO0022 USING 00002 BLKS R-0009

0001 : LG, 2
0002 : OF, LBLDIM
0003 : PU, LBLDIM
0004 : MR, ZBLDIM
0005 : MR, MVW.
0006 : MR, ZROTS
0007 : MR, ZRLINE
0008 : MR, DCODE.
0009 : MR, ZDRIVR
0010 : RU, LOADR, 99, OG, , , 2
0011 : SP, LG::3
0012 : OF, LG::3
0013 :
0014 :

LLFTR T-00004 IS ON CRO0022 USING 00002 BLKS R-0011

0001 : SV, 0
0002 : OF, LFLTR
0003 : PU, LFLTR::3
0004 : LG, 3
0005 : MR, ZLFLTR
0006 : MR, ZWFINT
0007 : MR, DCODE.
0008 : MR, ZWLINE
0009 : MR, ZDRIVR
0010 : RU, LOADR, 99, 1G
0011 : SP, LG::3
0012 : OF, LG::3
0013 :
0014 :
LHFTR  T=00004 IS ON CR00022 USING 00002 BLKS R=0014

0001: SV, 0
0002: OF, HFLTR
0003: PU, HFLTR: 3
0004: LG, 3
0005: MR, %HFLTR
0006: MR, %WFINT
0007: RU, LOADR, 99, 1G
0008: SP, 1OG: 3
0009: OF, 1OG
0010: RP, 1OG: 3
0011:

LSHOW  T=00004 IS ON CR00022 USING 00002 BLKS R=0011

0001: LG, 1
0002: MR, %SHOW
0003: MR, %WFINT
0004: MR, %DRIVR
0005: MR, %IJLINE
0006: OF, SHOW
0007: RU, LOADR, 99, 1G
0008: PU, 1OG: 3
0009: SP, 1OG: 3
0010: OF, 1OG
0011:

LFIRO  T=00004 IS ON CR00022 USING 00002 BLKS R=0010

0001: LG, 1
0002: MR, %FIRO
0003: MR, %WINDO
0004: MR, %BESIO
0005: OF, FIRO
0006: RU, LOADR, 99, 0G
0007: PU, 1OG: 3
0008: SP, 1OG: 3
0009: OF, 1OG
0010: RP, 1OG: 3
0011:
LIMAGE T=00004 IS ON CR00022 USING 00002 BLKS R=0009
0001 :LG, 1
0002 :OF, IMAGE
0003 :MR, ZIMAGE
0004 :MR, ZSPACE
0005 :MR, ZICMPW
0006 :MR, ZROTS
0007 :RU, LOADR, 99, 1G
0008 :PU, ICG:: 3
0009 :SP, ICG:: 3
0010 :OF, ICG

LRESIZ T=00004 IS ON CR00022 USING 00002 BLKS R=0004
0001 :LG, 3
0002 :SV, 0
0003 :OF, RESIZ
0004 :PU, RESIZ
0005 :MR, ZRESIZ
0006 :MR, ZDRIVR
0007 :MR, ZWLINE
0008 :MR, ZTRMGN
0009 :MR, ZLBRSZ
0010 :MR, ZWFINT
0011 :RU, LOADR, 99, 0G,, 2
0012 :SP, 10G:: 3
0013 :TR

LDINTP T=00004 IS ON CR00022 USING 00002 BLKS R=0011
0001 :LG, 3
0002 :SV, 0
0003 :OF, DINTP
0004 :PU, DINTP
0005 :MR, ZDINTP
0006 :MR, ZDRIVR
0007 :MR, ZWLINE
0008 :MR, ZTRMGN
0009 :MR, ZLBRSZ
0010 :MR, ZWFINT
0011 :RU, LOADR, 99, 0G,, 2
0012 :SP, 10G:: 3
0013 :TR
LBLDWF T=00004 IS ON CRO0022 USING 00002 BLKS R=0009

0001 : LG, 0
0002 : LG, 2
0003 : OF, BLDWF
0004 : PU, BLDWF
0005 : MR, %BLDWF
0006 : MR, %ICMPW
0007 : RU, LOADR, 99, 0G,,,2
0008 : SP, 10G::3
0009 : OF, 10G::3
0010 : RP, 10G::3
0011 : :

LDSPLY T=00004 IS ON CRO0022 USING 00002 BLKS R=0011

0001 : LG, 1
0002 : MR, %DSPLY
0003 : MR, %SCROL
0004 : MR, %WLINE
0005 : MR, %DRIVR
0006 : MR, %RESET
0007 : OF, DSPLY
0008 : RU, LOADR, 99, 0G
0009 : PU, 10G::3
0010 : SP, 10G::3
0011 : OF, 10G

LCURSR T=00004 IS ON CRO0022 USING 00002 BLKS R=0012

0001 : LG, 1
0002 : MR, %CURSR
0003 : MR, %WLINE
0004 : MR, %RLINE
0005 : MR, %DRIVR
0006 : MR, %MOVEC
0007 : OF, CURSR
0008 : RU, LOADR, 99, 1G
0009 : PU, 10G::3
0010 : SP, 10G::3
0011 : OF, 10G
0012 : :
LNOISE T=00004 IS ON CR00022 USING 00002 BLKS R=0011

0001 : LG, 1
0002 : MR, ZNOISE
0003 : MR, ZBLDWF
0004 : MR, ZWFINT
0005 : MR, ZICMPW
0006 : OF, NOISE
0007 : RU, LOADR, 99, 1G
0008 : PU, 10G: : 3
0009 : SP, 10G: : 3
0010 : OF, 10G
0011 : :
NASA T-00004 IS ON C00022 USING 00002 BLKS R=0008

0001 : SV, 4
0002 : OF, SHOW
0003 : RP, SHOW
0004 : OF, BLDWF
0005 : RP, BLDWF
0006 : OF, BLDIM
0007 : RP, BLDIM
0008 : OF, WTAPE
0009 : RP, WTAPE
0010 : OF, DSPLY
0011 : RP, DSPLY
0012 : OF, CURSR
0013 : RP, CURSR
0014 : OF, FDIGN
0015 : RP, FDIGN
0016 : OF, STABI
0017 : RP, STABI
0018 : OF, DPLAM
0019 : RP, DPLAM
0020 : OF, FILTR
0021 : RP, FILTR
0022 : OF, LFLTR
0023 : RP, LFLTR
0024 : OF, PLOT
0025 : RP, PLOT
0026 : OF, HLFLTR
0027 : RP, HLFLTR
0028 : OF, RESIZ
0029 : RP, RESIZ
0030 : OF, IMAGE
0031 : RP, IMAGE
0032 : OF, DINTP
0033 : RP, DINTP
0034 : OF, NOISE
0035 : RP, NOISE
0036 : MC, 23
0037 : SV, 0
0038 : RU, NASA1
0039 : OF, SHOW
0040 : OF, BLDWF
0041 : OF, DSPLY
0042 : OF, CURSR
0043 : OF, FDIGN
0044 : OF, STABI
0045 : OF, DPLAM
0046 : OF, FILTR
0047 : OF, LFLTR
0048 : OF, PLOT
0049 : OF, HLFLTR
0050 : OF, RESIZ
0051 : OF, IMAGE
0052 : OF, BLDIM
0053 : OF, DINTP
0054 : OF, WTAPE
0055
PROGRAM NASA1

THIS PROGRAM IS THE FATHER PROGRAM FOR THE IMAGE FILTERING PROGRAMS

DIMENSION IPRAM(5), NAME(3), NSON(3, 8), IMESS(30)

DATA NSON/2HDS, 2HPL, 2HY, 2HFD, 2HIG, 2HN, 2HFI, 2HLT,
       *2HR, 2HRE, 2HSI, 2HZ, 2HSH, 2HOW, 2H, 2HIM, 2HAG, 2HE,
       *2HNO, 2HIS, 2HE/

SON PROGRAM NAMES (FILES SAME PRECEEDED WITH "&")

DSPLY = DISPLAY PROGRAM
FDIGN = FILTER DESIGN MODULE
FILTR = FILTER IMPLEMENTATION MODULE
RESIZ = IMAGE MODIFICATION MODULE
SHOW = DISPLAYS WORK FILE
IMAGE = IMAGE DATA MANAGEMENT MODULE
NOISE = ADDITIVE GAUSSIAN NOISE

CALL RMPAR(IPRAM)
LU = IPRAM(1)
IF(LU.LE.0) LU = 1

NPRG IS THE NUMBER OF SONS

NPRG = 6
ICNT = 9

DISPLAY MENU

WRITE(LU, 30)
30 FORMAT(" SELECT PROCESSING OPTION", ", 1. IMAGE DISPLAY", ",
       *FILTER DESIGN", ", 3. FILTER IMAGE", ", 4. MODIFY IMAGE", 
       *ON", ", 8. TERMINATE PROGRAM")
READ(LU, *) IOPT
IF(IOPT.EQ.0.OR.IOPT.EQ.1) IOPT = 1
IF(IOPT.LT.1.0R.IOPT.GT.8) GO TO 16
IF(IOPT.EQ.8) GO TO 500

IPRAM(2) = IOPT
DO 10 I = 1, 3
NAME(I) = NSON(I, IOPT)
WRITE(LU, 15) NAME
10 FORMAT(" MODULE TO BE SCHEDULED IS ", 3A2)
GO TO 20

WRITE(LU, 17)
17 FORMAT(" INVALID RESPONSE")
GO TO 5
C
0051 C
0052 20 ICNW=LU+200B
0053 CALL EXEC(13,ICNW,IPRAM(3),IPRAM(4),IPRAM(5))
0054 CALL EXEC(23,NAME,IPRAM(1),IPRAM(2),IPRAM(3),IPRAM(4),IPRAM(6)
0055 WRITE(LU,40) (IPRAM(I),I=1,5)
0056 40 FORMAT("PARAMETERS RETURNED FROM MODULE",5(1H,4E11.3,2X))
0057 GO TO 5
0058 500 CONTINUE
0059 C
0060 C OF ALL SON PROGRAMS
0061 C
0062 DO 510 I=1,NPRG
0063 CALL FILL(IMESS,2H ,30)
0064 CALL CODE
0065 WRITE(IMESS,520) (NSON(J,I),J=1,3)
0066 520 FORMAT(OF",",3A2)
0067 IRTN=MESSS(IMESS,ICNT,LU)
0068 IF(IRTN.LT.0) CALL EXEC(2,LU,IMESS,IRTN)
0069 510 CONTINUE
0070 C
0071 STOP
0072 END
0073 C
0074 C SUBROUTINE FILL(ARRAY,IA,N)
0075 C THIS SUBROUTINE FILLS ARRAY ARRAY WHICH HAS N WORDS WITH THE
0076 C OF IA.
0077 C
0078 DIMENSION IARAY(N)
0079 DO 10 I=1,N
0080 10 IARAY(I)=IA
0081 RETURN
0082 END
PROGRAM DSPLY

THIS PROGRAM DISPLAYS AN IMAGE ON THE GMR-27. IMAGE FILE MUST BE IN FORMAT DESCRIBED BY IMAGE DISPLAY SUBSYSTEM.

INTEGER SLUI, STRTL, STRTP, SCROL

DIMENSION NAME(6), IDCB(144), IBLK(513), ISET(10), LU(5), JNAME(3)
INTEGER TEXT1(38), TEXT2(38), TEXT3(38)

EQUIVALENCE (IBLK(7), IBLK7), (IBLK(8), IBLK8), (IBLK(12), IBLK12)

1 (IBLK(13), JNAME), (TEXT1, IBLK(129)), (IBLK(169), TEXT2),
2 (IBLK(209), TEXT3)

EQUIVALENCE (ISET(5), ISET5), (ISET(6), ISET6), (ISET(7), ISET7),
1 (ISET(8), ISET8), (ISET(9), ISET9)

DATA ISET/100377B, 10377B, 24001B, 30000B, 5*1, 260C2B/
DATA SLUI1/34011B/
DATA LLAO, LEAO, LECO, LLBI, LLBX, LEBI, LEBX/64000B, 44000B, 54000B,
1 70001B, 71777B, 50001B, 51777B/

GET INPUT PARAMETERS

CALL RMPAR(LU)
IF (LU .LE. 0) LU = 1

OPEN IMAGE DIRECTORY FILE

CALL OPEN(IDCBL, IERR, 6HIMDIRC)
IF (IERR .LT. 0) GO TO 991

GET IMAGE FILE NAME

CALL RESET(LU)
WRITE(LU, 20)
WRITE(LU, 21)
WRITE(LU, 22)
WRITE(LU, 23)
WRITE(LU, 24)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
WRITE(LU, 26)
0060 C
0061 C FIND IMAGE FILE
0062 C
0063 CALL RWNDF(IDCB)
0064 110 CALL READF(IDCB,IERR,IBLK,256,LEN)
0065 IF (IERR .LT. 0) GO TO 991
0066 IF (LEN .EQ. -1) GO TO 800
0067 C
0068 DO 120 I=1,6
0069 IF (IBLK(I) .NE. NAME(I)) GO TO 110
0070 120 CONTINUE
0071 C
0072 C IMAGE FOUND--CHECK IF ON DISC
0073 C
0074 IF (IBK12 .EQ. 1) GO TO 130
0075 C
0076 C IMAGE NOT ON DISC
0077 C
0078 WRITE(LU,12)
0079 12 FORMAT(""
0080 GO TO 105
0081 C
0082 C IMAGE IS ON DISC
0083 C
0084 130 CALL CLOSE(IDCB)
0085 RMIN = IBLK(9)
0086 RVAX = IBLK(10)
0087 WRITE(LU,29)(IBLK(I),I=7,10)
0088 29 FORMAT(""
0089 CALL EXEC(2,LU,TEXT1,37)
0090 CALL EXEC(2,LU,TEXT2,37)
0091 CALL EXEC(2,LU,TEXT3,37)
0092 WRITE(LU,27)
0093 CALL OPEN(IDCB,IERR,JNAME)
0094 IF (IERR .LT. 0) GO TO 991
0095 C
0096 C EXTRACT DISPLAY INFORMATION
0097 C
0098 NUML = IBLK7
0099 NUMP = IBLK8
0100 STRTL = (256-MINO(256,NUML))/2
0101 STRTP = (512-MINO(512, NUMP))/2
0102 C
0103 500 ISET5 = IOR(LLAO,IAND(STRTL,1777B))
0104 ISET6 = IOR(LEAO,IAND(STRTP,1777B))
0105 ISET7 = LLB1
0106 ISET8 = LEB1
0107 ISET9 = IOR(LECO,IAND(STRTP,1777B))
0108 C
0109 CALL DRIVR(2,ISET,10)
0110 C
IERR = 0
DO 600 I=1,MINO(NUML,256)
IF (IERR .LT. 0) GO TO 991
CALL READF(IDCBIERR,IBLK,512,NUM)
IF (NUM .LT. 0) GO TO 600
C
DO 595 J=1,NUM
IBLK(J) = (255./(RMAX-RMIN))*(FLOAT(IBLK(J))-IUIIN)
IF (IBLK(J) .LT. 0) IBLK(J) = 0
IF (IBLK(J) .GT. 377B) IBLK(J) = 377B
595 CONTINUE
C
IBLK(NUM+1) = SLU11
CALL DRIVR(40002B,IBLK,NUM+1)
600 CONTINUE
IFIRST = 0
ILAST = 255
C
C OUTPUT SOFT KEY FUNCTIONS
WRITE(LU,29)
29 FORMAT(/"FUNCTION KEYS:"/)
WRITE(LU,30)
WRITE(LU,30)
30 FORMAT(4("dB d@ ")/)
WRITE(LU,31)
31 FORMAT(" ")
WRITE(LU,32)
32 FORMAT(" << SCROLL SCROLL >> ")
124X," NEW IMAGE EXIT ")
610 CALL EXEC(1,LU,INPT,1)
INPT = INPT-7023
IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 610
C
BRANCH TO APPROPRIATE SECTION
GO TO (1000,2000,3000,4000,5000,6000,100,9000),INPT
C
SCROLL IMAGE BACK
1000 IERR = SCROL(IDCBIERR,-9,NUML,IFIRST,ILAST,RMAX,RMIN)
IF (IERR .LT. 0) GO TO 991
GO TO 610
C SCROLL FORWARD
0160 C
0161 2000 IERR = SCROL(IDCB,17,NUML,IFIRST,ILAST,RMAX,RMIN)
0162 IF (IERR .LT. 0) GO TO 991
0163 GO TO 610
0164 3000 CONTINUE
0165 C
0166 C POSITION CURSOR
0167 C
0168 4000 CALL EXEC(23,6HCURSR ,LU)
0169 GO TO 605
0170 5000 CONTINUE
0171 6000 CONTINUE
0172 GO TO 610
0173 C
0174 C TERMINATE
0175 C
0176 9000 CALL CLOSE(IDCB)
0177 CALL RESET(LU)
0178 WRITE(LU,33)
0179 33 FORMAT("END PROGRAM")
0180 CALL EXEC(6)
0181 C
0182 C FILE NOT FOUND
0183 C
0184 800 WRITE(LU,3)
0185 3 FORMAT(""
0186 GO TO 105
0187 C
0188 991 CALL RESET(LU)
0189 WRITE(LU,9) IERR
0190 9 FORMAT("FILE ERROR",I6)
0191 CALL CLOSE(IDCB)
0192 END
0193 $
PROGRAM CURSR

DIMENSION LU(5),IBUF(2352),IZER0(2)
INTEGER EA,LA
DATA IZERO/440008,640008/

CALL RMPAR(LU)

SAVE IMAGE LINES

DO 50 I=0,20
CALL RLINE(I,0,111,IBUF(112*I+1))
50 CONTINUE
WRITE(LU,1)
1 FORMAT(,,12X,,12X,,RETURN
CALL MOVEC(0,255)
EA = 0
LA = 255
100 CALL EXEC(1,LU,INPT,1)
INPT = INPT-7023
IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 100

BRANCH TO APPROPRIATE SECTION

GO TO (400,200,300,100,100,300,100,600), INPT

MOVE CURSOR UP

LA = MOD(LA+11,256)
CALL MOVEC(EA,LA)
GO TO 100

MOVE CURSOR DOWN

LA = MOD(LA+249,256)
CALL MOVEC(EA,LA)
GO TO 100

MOVE CURSOR LEFT

EA = MOD(EA+499,512)
CALL MOVEC(EA,LA)
GO TO 100
15

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0056</td>
<td>C MOVE CURSOR RIGHT</td>
</tr>
<tr>
<td>0057</td>
<td>C</td>
</tr>
<tr>
<td>0058</td>
<td>500 EA = MOD(EA+17,512)</td>
</tr>
<tr>
<td>0059</td>
<td>CALL MOVEC(EA,LA)</td>
</tr>
<tr>
<td>0060</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0061</td>
<td>C</td>
</tr>
<tr>
<td>0062</td>
<td>C RETURN TO PREVIOUS SCREEN</td>
</tr>
<tr>
<td>0063</td>
<td>C</td>
</tr>
<tr>
<td>0064</td>
<td>600 DO 610 I=0,20</td>
</tr>
<tr>
<td>0065</td>
<td>CALL WLINE(I,0,111,IBUF(112*I+1))</td>
</tr>
<tr>
<td>0066</td>
<td>610 CONTINUE</td>
</tr>
<tr>
<td>0067</td>
<td>C</td>
</tr>
<tr>
<td>0068</td>
<td>CALL DRIVR(2,IZERO,2)</td>
</tr>
<tr>
<td>0069</td>
<td>C</td>
</tr>
<tr>
<td>0070</td>
<td>END</td>
</tr>
<tr>
<td>0071</td>
<td>$</td>
</tr>
</tbody>
</table>

&ICMPW T=00004 IS ON CR00022 USING 00002 BLKS R=0011

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>FTN4</td>
</tr>
<tr>
<td>0002</td>
<td>FUNCTION ICMPW(IBUF1,IBUF2,ILEN)</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION IBUF1(1),IBUF2(1)</td>
</tr>
<tr>
<td>0004</td>
<td>DO 100 I=1,ILEN</td>
</tr>
<tr>
<td>0005</td>
<td>IF (IBUF1(I) NE. IBUF2(I)) GO TO 200</td>
</tr>
<tr>
<td>0006</td>
<td>100 CONTINUE</td>
</tr>
<tr>
<td>0007</td>
<td>ICMPW = 0</td>
</tr>
<tr>
<td>0008</td>
<td>RETURN</td>
</tr>
<tr>
<td>0009</td>
<td>200 ICMPW = I</td>
</tr>
<tr>
<td>0010</td>
<td>END</td>
</tr>
<tr>
<td>0011</td>
<td>$</td>
</tr>
</tbody>
</table>

&RESET T=00004 IS ON CR00022 USING 00002 BLKS R=0017

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>FTN4</td>
</tr>
<tr>
<td>0002</td>
<td>SUBROUTINE RESET(LU)</td>
</tr>
<tr>
<td>0003</td>
<td>C</td>
</tr>
<tr>
<td>0004</td>
<td>C</td>
</tr>
<tr>
<td>0005</td>
<td>WRITE(LU,1)</td>
</tr>
<tr>
<td>0006</td>
<td>1 FORMAT(&quot;&quot;&quot;)</td>
</tr>
<tr>
<td>0007</td>
<td>C</td>
</tr>
<tr>
<td>0008</td>
<td>C WAIT 200 MSEC</td>
</tr>
<tr>
<td>0009</td>
<td>C</td>
</tr>
<tr>
<td>0010</td>
<td>CALL EXEC(12,0,1,0,-20)</td>
</tr>
<tr>
<td>0011</td>
<td>C</td>
</tr>
<tr>
<td>0012</td>
<td>C CLEAR DISPLAY</td>
</tr>
<tr>
<td>0013</td>
<td>C</td>
</tr>
<tr>
<td>0014</td>
<td>WRITE(LU,2)</td>
</tr>
<tr>
<td>0015</td>
<td>2 FORMAT(&quot;&quot;&quot;)</td>
</tr>
<tr>
<td>0016</td>
<td>END</td>
</tr>
<tr>
<td>0017</td>
<td>$</td>
</tr>
</tbody>
</table>
SUBROUTINE RLINE(LINE, IPIX, JPIX, IDATA)

WHERE
LINE = LINE # TO READ
IPIX = STARTING PIXEL
JPIX = ENDING PIXEL

IDATA = BUFFER IN WHICH DATA IS RETURNED (1 PIXEL/WORD

DIMENSION IDATA(512), INIT(5)

EQUIVALENCE (LLA, INIT(2)), (LEA, INIT(3)), (LEB, INIT(4))

DATA INIT/100377B, 64000B, 44000B, 50000B, 26002B/

COMPUTE DIRECTION
IDIRC = 1
IF (IPIX .GT. JPIX) IDIRC = -1

SET UP FOR READ BACK

LLA = 64000B + IAND(LINE, 377B)
LEA = 44000B + IAND(IPIX, 777B)
LEB = 50000B + IDIRC + 512

CALL DRIVR(2, INIT, 5)

READ BACK LINE
NUM = IDIRC*(JPIX - IPIX) + 1

CALL DRIVR(1, IDATA, NUM)

RETURN
END
SUBROUTINE MOVEC(IX, IY)

INTEGER WACO

DIMENSION ICR(7), IPOO(5), IPXY(3)

EQUIVALENCE (ICR, ICR1), (ICR(2), ICR2), (ICR(3), ICR3), (ICR(5), ICR6), (ICR(7), ICR7), (IPXY, IPXY1), (IPXY(2), IPXY2)

DATA IPOO/44000B, 64000B, 24015B, 501`.':.13, 2o002B/

DATA ICR/0,0,0,22054B4O,0,0/

DATA IPXY/0,0,24001B/

DATA WACO, LEAO, LLAO/22000B, 44000B, 64000B/

WRITE POSITION ON SCREEN

CALL DRIVR(2, IPOO, 5)

ID1 = IY/100

ICR1 = WACO + ID1 + 60B

ID2 = (IY-ID1*100)/10

ICR2 = WACO + ID2 + 60B

ID3 = (IY-ID1*100-ID2*10)

ICR3 = WACO + ID3 + 60B

ID1 = IX/100

ICR5 = WACO + ID1 + 60B

ID2 = (IX-ID1*100)/10

ICR6 = WACO + ID2 + 60B

ID3 = IX-ID1*100-ID2*10

ICR7 = WACO + ID3 + 60B

CALL DRIVR(2, ICR, 7)

POSITION CURSOR

IPXY1 = IOR(LEAO, LAND(IX, 1777B))

IPXY2 = IOR(LLAO, LAND(IY, 377B))

CALL DRIVR(2, IPXY, 3)

RETURN

END
&IMAGE T=00004 IS ON CR00022 USING 00015 BLKS R=0161

0001  FIN4,Q,C,T
0002  PROGRAM IMAGE
0003  C
0004  C
0005  C THIS PROGRAM IS THE IMAGE FILE MANAGER FOR THE IMAGE DISPLAY
0006  C SUBSYSTEM.
0007  C
0008  C
0009  DIMENSION LU(5),IDCB1(272),IDCB2(528),IDCB3(144),JNAME(3),
0010  1 IFNAM(3),NAME(6),IDATA(512),KNAME(6)
0011  C
0012  INTEGER ENTRY(256),ISIZE(2),TXT1(19)
0013  C
0014  EQUIVALENCE (ENTRY(7),NLINE),(ENTRY(8),NPIXL),(ENTRY(12),LOC
0015  1 (ENTRY(13),JNAME),(ENTRY(16),IFNAM),(ENTRY(19),IFNUM)
0016  2,(ENTRY,KNAME)
0017  2,(TEXT1,ENTRY(129))
0018  EQUIVALENCE (ISIZE(2),ISIZ2)
0019  C
0020  C
0021  C GET INPUT PARAMETERS
0022  C
0023  CALL RMPAR(LU)
0024  IF (LU .LE. 0) LU = 1
0025  C
0026  C OUTPUT HEADING
0027  C
0028  900 WRITE(LU,1)
0029  1 FORMAT(/" IMAGE FILE MANAGER"/)
0030  C
0031  C GET COMMAND INPUT
0032  C
0033  1000 WRITE(LU,2)
0034  2 FORMAT(/" > ")
0035  READ(LU,3) ICMD
0036  3 FORMAT(A2)
0037  C
0038  C EXECUTE COMMAND
0039  C
0040  IF (ICMD .NE. 2H??) GO TO 1010
0041  C
0042  C COMMAND IS HELP
0043  C
0044  WRITE(LU,4)
0045  4 FORMAT(/" COMMANDS ARE:"/,
0046  1" BU-BUILD IMAGE FILE"/,
0047  2" DI-DISPLAY IMAGE ON GMR-27"/,
0048  3" SA-SAVE IMAGE TO TAPE"/,
0049  4" RE-RESTORE IMAGE TO DISC"/,
0050  4" DL-DIRECTORY LIST"/,
0051  4" PU-PURGE IMAGE"/,
0052  4" WT-WRITE NASA TAPE"/,
0053  5" ??-HELP"/,
0054  6" EX-EXIT"/)
0055  GO TO 1000
19

C
1010 IF (ICMD .NE. 2HBU) GO TO 1030
C
BUILD IMAGE COMMAND
C
CALL EXEC(23+100000B,6HBLDIM ,LU)
GO TO 1020
5 GO TO 900
C
PROGRAM NOT RP'ED
C
1020 WRITE(LU,6)
6 FORMAT(" BLDIM NOT RP'ED!")
GO TO 1000
C
1030 IF (ICMD .NE. 2HDI) GO TO 1045
C
DISPLAY IMAGE COMMAND
C
CALL EXEC(23+100000B,6HDSPLY ,LU)
GO TO 1040
7 GO TO 900
C
DSPLY NO RP'ED
C
1040 WRITE(LU,8)
8 FORMAT(" DSPLY NOT RP'ED!")
GO TO 1000
C
1045 IF (ICMD .NE. 2HWT) GO TO 1050
C
WRITE NASA TAPE
C
CALL EXEC(23,6HWTAPE ,LU)
GO TO 1000
C
1050 IF (ICMD .NE. 2HSA .AND. ICMD .NE. 2HRE .AND.
1 ICMD .NE. 2HPU) GO TO 1200
C
SAVE/RESTORE IMAGE TO/FROM TAPE AND PURGE IMAGE
C
OPEN DIRECTORY FILE
C
CALL OPEN(IDCBI,IERR,6HIMDIRC,2,2HIM,23,272)
IF (IERR .LT. 0) GO TO 9999
C
GET IMAGE NAME
C
WRITE(LU,9)
9 FORMAT(" ENTER IMAGE NAME (12 CHARACTERS)? ")
READ(LU,10) NAME
10 FORMAT(6A2)
C
FIND IMAGE
CALL READF(IDCBI,IERR,ENTRY,256,LEN)
IF (LEN .NE. -1) GO TO 1070

C EOF REACHED
WRITE(LU,11)
11 FORMAT(" IMAGE NOT FOUND!")
CALL CLOSE(IDCBI)
GO TO 1000

IF (IERR .LT. 0) GO TO 9999

C COMPARE NAME OF IMAGE
IF (ICMPW(ENTRY,NAME,6) .NE. 0) GO TO 1060
C IMAGE FOUND
IF (ICMD .EQ. 2HRE) GO TO 1120
IF (ICMD .EQ. 2HPU) GO TO 1300

C TASK IS TO SAVE IMAGE
IF (LOC .EQ. 1) GO TO 1090
C I?CAGE ALREADY ON TAPE
WRITE(LU,12)
12 FORMAT(" IMAGE NOT ON DISC!")
GO TO 1000
C I^tkGE ON DISC
1090 CALL OPEN(IDCBI2,IERR,JNAME,0,0,0,528)
IF (IERR .LT. 0) GO TO 9999

C GET TYPE 0 FILE
WRITE(LU,13)
13 FORMAT(3A2)
READ(LU,*) IFNAM
CALL OPEN(IDCBI3,IERR,IFNAM)
IF (IERR .LT. 0) GO TO 9999
CALL RWNDF(IDCBI3,IERR)
IF (IERR .LT. 0) GO TO 9999
WRITE(LU,15)
15 FORMAT(" FILE _")
READ(LU,*) IFNUM
CALL SPACE(IDCBI3,IERR,IFNUM-1)
IF (IERR .LT. 0) GO TO 9999
WRITE HEADER ON TAPE

CALL WRITF(IDC33,IERR,ENTRY,11)
IF (IERR .LT. 0) GO TO 9999

NOW TRANSFER DATA

DO 1100 I = 1,NLINE
CALL READF(IDC22,IERR,ILTA,512)
IF (IERR .LT. 0) GO TO 9999
IF(IPACK .NE. 1) GO TO 1101

PACK DATA

DO 1102 J = 1,NPIXL,2
K = 0.5*(J+1)
IVAR = IDATA(J+1)
CALL ROT8(IVAR,KVAR)
IDATA(K) = IOR(IDATA(J),KVAR)
1101
1100 CONTINUE

WRITE EOF

CALL WRITF(IDC33,IERR,0,-1)
IF (IERR .LT. 0) GO TO 9999

PURGE DISC FILE

CALL PURGE(IDC22,IERR,JNAME,2HIM)
IF (IERR .LT. 0) GO TO 9999

UPDATE ENTRY

LOC = 2
CALL POSNT(IDC11,IERR,-1)
IF (IERR .LT. 0) GO TO 9999
CALL WRITF(IDC11,IERR,ENTRY,256)
IF (IERR .LT. 0) GO TO 9999

CALL CLOSE(IDC11)
CALL RWDIF(IDC33)
CALL CLOSE(IDC33)
GO TO 1000

RESTORE IMAGE FROM TAPE

IF (LOC .EQ. 2) GO TO 1130

IMAGE ON DISC

WRITE(LU,16)
FORMAT(" IMAGE ALREADY ON DISC!")
CALL CLOSE(IDC11)
GO TO 1000
0223 C
0229 C CREATE DISC FILE
0230 C
0231 1130 ISIZE = (FLOAT(NLINE)*FLOAT(NPIXL)+127.)/128.
0232 ISIZ2 = NPIXL
0233 C
0234 1134 CALL CREAF(IDC22,IER1,JNAME,ISIZE,2,2HIM,23,328)
0235 IF (IER1 .GE. 0) GO TO 1135
0236 C
0237 C CAN'T CREATE DISC FILE
0238 C
0239 WRITE(LU,19)
0240 19 FORMAT(" CAN'T CREATE DISC FILE!!")
0241 CALL CLOSE(IDC11)
0242 GO TO 1000
0243 C
0244 C OPEN TYPE 0 FILE
0245 C
0246 1135 CALL OPEN(IDC31,IER1,IFNAM)
0247 C
0248 C GET LU OF TYPE 0 FILE
0249 C
0250 CALL LOCF(IDC31,IER1,IREC,IRB,IOFF,JSEC,MTLU)
0251 IF (IER1 .LT. 0) GO TO 9999
0252 C
0253 C TELL USER TO MOUNT TAPE
0254 C
0255 WRITE(LU,17) MTLU
0256 17 FORMAT(" MOUNT TAPE ON LU ",I2" ENTER RETURN WHEN READY")
0257 CALL EXEC(1,LU,IREC,1)
0258 C
0259 C REWIND TAPE
0260 C
0261 CALL RNDF(IDC31,IER1)
0262 IF (IER1 .LT. 0) GO TO 9999
0263 C
0264 C SPACE FORWARD TO FILE
0265 C
0266 CALL SPACE(IDC31,IER1,IFNUM-1)
0267 C
0268 C READ HEADER
0269 C
0270 CALL READF(IDC31,IER1,DATA,11)
0271 IF (IER1 .LT. 0) GO TO 9999
0272 IF(ICMPW(IDATA,ENTRY,11) .NE. 0) GO TO 1160
0273 C
0274 C HEADF. COMPARES
0275 C
0276 C TRANSFER DATA
0277 C
0278 DO 1140 I=1,NLINE
0279 CALL READF(IDC31,IER1,DATA,NPIXL)
0280 IF (IER1 .LT. 0) GO TO 9999
0281 CALL WRITF(IDC22,IER1,DATA,NPIXL)
0282 IF (IER1 .LT. 0) GO TO 9999
0283 1140 CONTINUE
0284 C
CALL RWNDF(IDC3)
CALL CLOSE(IDC3)
CALL CLOSE(IDC2)

C UPDATE DIRECTORY ENTRY
LOC = 1
CALL POSNT(IDC1, IERR, -1)
IF (IERR .LT. 0) GO TO 9999
CALL WRITF(IDC1, IERR, ENTRY, 256)
IF (IERR .LT. 0) GO TO 9999
CALL CLOSE(IDC1, IERR)
IF (IERR .LT. 0) GO TO 9999
GO TO 1000
C LABEL DOES NO MATCH
WRITE(LU, 18)
FORMAT(" WRONG FILE!!")
CALL RWNDF(IDC3)
CALL CLOSE(IDC3)
CALL CLOSE(IDC1)
GO TO 1000

IF (ICMD .NE. 2HDL) GO TO 1230
C DIRECTORY LIST
C OPEN DIRECTORY FILE
CALL OPEN(IDC1, IERR, 6H1MDIRC)
IF (IERR .LT. 0) GO TO 9999
C OUTPUT HEADING
WRITE(LU, 30)
FORMAT(//"IMAGE NAME #LINES #PIXELS LOC TEXT")
C OUTPUT INFO
CALL READF(IDC1, IERR, ENTRY, 256, LEN)
IF (LEN .NE. -1) GO TO 1220
EOF REACHED
CALL CLOSE(IDC1)
GO TO 1000
IF (IERR .LT. 0) GO TO 9999
IF (ENTRY .EQ. -1) GO TO 1210
ICHR = 2H
IF (LOC .NE. 1) ICHR = 2HT
WRITE(LU, 31)KNANE, NLINE, NPIXL, ICHR, TEXT
FORMAT(6A2, 2X, I4, 4X, I4, 3X, A5, 2`!, 19A2)
GO TO 1210
C IF (ICMD .EQ. 2H16) GO TO 1240
C
C ILLEGAL COMMAND
C
WRITE(LU,22)
22 FORMAT("ILLEGAL COMMAND!")
GO TO 1000
C
WRITE(LU,23)
23 FORMAT("END PROGRAM")
CALL EXEC(6)
C
PURGE FILE
C
CALL POSNT(IDCBI,IERR,-1)
IF (IERR .LT. 0) GO TO 9999
ENTRY = -1
CALL WRITP(IDCBI,IERR,ENTRY,256)
IF (IERR .LT. 0) GO TO 9999
C
PURGE DATA FILE
C
CALL PURGE(IDCBI2,IERR,JNAME,2HIM)
CALL CLOSE(IDCBI)
GO TO 1000
C
ERROR
C
WRITE(LU,20) IERR
20 FORMAT(" FILE ERROR ",I6)
CALL CLOSE(IDCBI)
GO TO 1000
END
$
SUBROUTINE SPACE(IDCB, IERR, NUM)

DIMENSION IDCB(144)

DATA IFRWD, IBACK / 1300B, 1400B/

IERR = 0
IF (NUM .EQ. 0) RETURN

IDIR = IFRWD
IF (NUM .GT. 0) GO TO 100
IDIR = IBACK
NUM = -NUM

DO 110 I = 1, NUM
CALL FCONT(IDCB, IERR, IDIR)
IF (IERR .LT. 0) RETURN
CONTINUE

RETURN
END
PROGRAM RESIZ

WRITTEN BY W. E. ALEXANDER

PROGRAM FORMS A PART OF THE SPATIAL DOMAIN FILTERING PACKAGE

PROGRAM ALLOWS THE USER TO INTERPOLATE AND SCALE AN IMAGE AN
CHANGE ITS DATA TYPE. THUS A FLOATING POINT IMAGE CAN BE MADE
INTO AN EIGHT BIT IMAGE.

DIMENSION F(512),G(512),IOP(512),IPRAM(5),NSON(3,2)
DIMENSION A(3,2,2),B(3,2,2),NAME(3),INM(3)
DIMENSION JMES(40),DIRC(515),IKNM(5)
INTEGER WFINT,READL,RITEL
EQUIVALENCE(G(1),IOP(1))
EQUIVALENCE(F(1),G(1))
EQUIVALENCE(DIRC(4),F(1)),(DIRC(1),INM(1)),(INM(1),NROW)
EQUIVALENCE(INM(2),ICOLS),(DIRC(2),AMAX),(DIRC(3),AMIN)
DATA NSON/2HLF,2HLT,2HR ,2HDI,2HNT,2HP /
CALL RMPAR(IPRAM)
LU=IPRAM(1)
IF(LU.EQ.0) LU=1

INITIALIZE PARAMETERS

ITYPE = 8

CALL CODE
WRITE (JMES,6)
FORMAT (" RESIZE")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
NTYPE=32
IRTCD=0
IMXX=512
IMXP1=IMXX+1
IFE=0
ILE=511
IFR=0
ILR=511

CALL CODE
WRITE (JMES,995)
FORMAT (" RESIZE IMAGE ")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)

SPECIFY DATA LENGTH FOR OUTPUT
0053 10 CALL CODE
0054 WRITE(JMES, 11)
0055 11 FORMAT(" SPECIFY OUTPUT DATA TYPE")
0056 CALL TRMGN(JMES, LU, 0)
0057 CALL BLANK(JMES)
0058 CALL CODE
0059 WRITE(JMES, 12)
0060 12 FORMAT(" 1. 8 BIT IMAGE")
0061 CALL TRMGN(JMES, LU, 0)
0062 CALL BLANK(JMES)
0063 CALL CODE
0064 WRITE(JMES, 13)
0065 13 FORMAT(" 2. 15 BIT IMAGE")
0066 CALL TRMGN(JMES, LU, 1, RTM, ICD, IRTM)
0067 C
0068 C IRTM=ICD
0069 C
0070 CALL BLANK(JMES)
0071 15 CALL SPCHIR (IRTM, IRT)
0072 GO TO (500, 10, 5, 20, 17, 17), IRT
0073 17 CALL CKFLD(2, ICD, IRS)
0074 GO TO (25, 25, 30, 30, 20), IRS
0075 20 IW=1
0076 GO TO 950
0077 25 IRTYPE =8
0078 IMAX=255
0079 GO TO 32
0080 30 IRTYPE =15
0081 IMAX=32767
0082 C
0083 C SPECIFY WORK FILE
0084 C
0085 C
0086 32 CALL BLANK(JMES)
0087 CALL CODE
0088 WRITE(JMES, 450)
0089 450 FORMAT(" SELECT OPTION")
0090 CALL TRMGN(JMES, LU, 0)
0091 CALL BLANK(JMES)
0092 CALL CODE
0093 WRITE(JMES, 455)
0094 455 FORMAT(" 1. SPECIFY NEW IMAGE")
0095 CALL TRMGN(JMES, LU, 0)
0096 CALL BLANK(JMES)
0097 CALL CODE
0098 WRITE(JMES, 460)
0099 460 FORMAT(" 2. USE CURRENT WORK FILE")
0100 CALL TRMGN(JMES, LU, 1, RTM, ICD, IRTM)
0101 CALL BLANK(JMES)
0102 465 CALL CKFLD(2, ICD, IRS)
0103 GO TO (475, 475, 480, 485), IRS
0104 485 IW=12
0105 C
0106 C OPEN WORK FILE
0107 C
475 IGET=WFIN (NR0W,ICOLS,AMAX,AMIN,LU)
0110 IF (IGET.LT.0) GO TO 999
0111 480 IGET=READL(-1,0,511,DIRC)
0112 IF (IGET.LT.0) GO TO 999
0113 40 CALL CODE
0114 WRITE(JMES,45) AMAX,AMIN
0115 45 FORMAT(" AMAX= ",E12.5," AMin= ",E12.5," FOR IMAGE")
0116 CALL TRMGN(JMES,LU,0)
0117 CALL BLANK(JMES)
0118 C
0119 C SPECIFY IMAGE SCALING OPTION
0120 C
0121 50 CALL CODE
0122 WRITE(JMES,51)
0123 51 FORMAT(" SPECIFY IMAGE SCALING OPTION")
0124 CALL TRMGN(JMES,LU,0)
0125 CALL BLANK(JMES)
0126 CALL CODE
0127 WRITE(JMES,52)
0128 52 FORMAT(" 1. AUTOMATIC SCALING")
0129 CALL TRMGN(JMES,LU,0)
0130 CALL BLANK(JMES)
0131 CALL CODE
0132 WRITE(JMES,53)
0133 53 FORMAT(" 2. SYSTEM DEFAULT OPTION")
0134 CALL TRMGN(JMES,LU,0)
0135 CALL BLANK(JMES)
0136 CALL CODE
0137 WRITE(JMES,54)
0138 54 FORMAT(" 3. USER SPECIFIED SCALE FACTOR")
0139 CALL TRMGN(JMES,LU,0)
0140 CALL BLANK(JMES)
0141 CALL CODE
0142 WRITE(JMES,56)
0143 56 FORMAT(" 4. USER SPECIFIED MAX AND MIN")
0144 CALL TRMGN(JMES,LU,0)
0145 CALL BLANK(JMES)
0146 CALL CODE
0147 WRITE(JMES,57)
0148 57 FORMAT(" 5. LOG COMPRESSION")
0149 CALL TRMGN(JMES,LU,0)
0150 CALL BLANK(JMES)
0151 CALL CODE
0152 WRITE(JMES,58)
0153 58 FORMAT(" 6. EXPONENTIATION OPTION")
0154 CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0155 CALL BLANK(JMES)
0156 55 CALL CKFLD (6,ICD,IRS)
0157 GO TO (65,65,75,80,105,160,165,60),IRS
0158 60 IW=2
0159 GO TO 950
0160 C
0161 C AUTOMATIC SCALING SELECTED
0162 C
0163 65 SCL=AMAX-AMIN
0164 IF (ABS(SCL).LE.1.0E-5) GO TO 70
0165 SCL=FLOAT(IMAX)/SCL
0166 IOPT=1
0167 GO TO 90
0168 C
0169 C SYSTEM DEFAULT SCALING OPTION SELECTED
0170 C
0171 75 SCL=1.0
0172 IOPT=2
0173 GO TO 100
0174 C
0175 C USER SPECIFIED MAX AND MIN SELECTED
0176 C
0177 70 SCL=AMAX-AMIN
0178 IF (ABS(SCL).LE.1.0E-5) GO TO 70
0179 SCL=FLOAT(IMAX)/SCL
0180 IOPT=1
0181 GO TO 90
0182 C
0183 C USER SPECIFIED SCALE FACTOR SELECTED
0184 C
0185 70 SCL=AMAX-AMIN
0186 IF (ABS(SCL).LE.1.0E-5) GO TO 70
0187 SCL=FLOAT(IMAX)/SCL
0188 IOPT=1
0189 GO TO 90
0190 C
0191 C LOG COMPRESSION SELECTED
0192 C
0193 70 SCL=AMAX-AMIN
0194 IF (ABS(SCL).LE.1.0E-5) GO TO 70
0195 SCL=FLOAT(IMAX)/SCL
0196 IOPT=1
0197 GO TO 90
0198 C
0199 C EXPONENTIATION OPTION SELECTED
0200 C
0201 70 SCL=AMAX-AMIN
0202 IF (ABS(SCL).LE.1.0E-5) GO TO 70
0203 SCL=FLOAT(IMAX)/SCL
0204 IOPT=1
0205 GO TO 90
0206 C
CALL CODE
WRITE(JMES,151)
151 FORMAT(" SCALE TOO SMALL")
CALL TRMN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,152)
152 FORMAT(" ENTER CR TO GO TO SYSTEM LEVEL MENU")
CALL TRMN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
IRTCD=1HX
GO TO 1000
155 SCL=(1.0*SCL)*FLEAT(IMAX)
GO TO 190
LOG COMPRESSION OPTION SELECTED
160 CALL CODE
WRITE(JMES,161)
161 FORMAT(" LOG COMPRESSION OPTION SELECTED")
CALL TRMN(JMES,LU,0)
CALL BLANK(JMES)
IOP=5
SCL=FLEAT(IMAX)/ALOG(AMAX-AMIN+1.0)
GO TO 190
EXPONENTIATION OPTION SELECTED
165 CALL CODE
WRITE(JMES,166)
166 FORMAT(" ENTER DESIRED EXPONENT")
CALL TRMN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
170 CALL SPCHR (IRTM,IRT)
POWER=RTM
IF (IRT .EQ. 5) GO TO 180
IW=6
GO TO 950
POWER=ABS(POWER)
CALL CODE
WRITE(JMES,185) POWER
185 FORMAT(" EXPONENT= ",1F10.4)
CALL TRMN(JMES,LU,0)
CALL BLANK(JMES)
SCL=FLEAT(IMAX)/((AMAX-AMIN)**POWER)
IOP=6
OBTAIN PARAMETERS FOR RESIZING IMAGE
190 INUM=64
INCNT=0
IMCNT=0
NTST=INUM
195 CALL CODE
WRITE(JMES,196)
196 FORMAT(" INDEPENDENT DIRECTIONAL SCALING")
CALL TRMN(JMES,LU,0)
CALL BLANK(JMES)
0174 C USER ENTERS SCALE FACTOR
0175 C
0176 C
0177 80 CALL CODE
0178 WRITE(JMES,81)
0179 81 FORMAT("ENTER DESIRED SCALE FACTOR")
0180 CALL TRMGN(JMES,LU,1,RTM,ICD,IRT)
0181 CALL BLANK(JMES)
0182 85 CALL SPCHR (IRTM,IRT)
0183 GO TO (1000,1000,1000,1000,855),IRT
0184 855 IF (IRT .EQ. 5) GO TO 95
0185 9U IW=3
0186 GO TO 950
0187 95 CALL CODE
0188 WRITE(JMES,100) GAIN
0189 100 FORMAT("SCALE FACTOR = ",F10.4)
0190 CALL TRMGN(JMES,LU,0)
0191 CALL BLANK(JMES)
0192 IOPT=3
0193 CALL SPCHR (IRTM,IRT)
0194 SCL=GAIN
0195 GO TO 190
0196 C USER SPECIFIED MAXIMUM AND MINIMUM
0197 C
0198 C
0199 105 IOPT=4
0200 CALL CODE
0201 WRITE(JMES,106)
0202 106 FORMAT("ENTER MAXIMUM FOR IMAGE")
0203 CALL TRMGN(JMES,LU,1,RTM,ICD,IRT)
0204 CALL BLANK(JMES)
0205 110 CALL SPCHR (IRTM,IRT)
0206 GO TO (1000,1000,1000,1000,910),IRT
0207 910 AMXIN=RTM
0208 IF (RTM .NE. 0B) GO TO 120
0209 115 IW=4
0210 GO TO 950
0211 120 CALL CODE
0212 WRITE(JMES,121) AMXIN
0213 121 FORMAT("MAXIMUM FOR IMAGE = ",1PE15.8)
0214 CALL TRMGN(JMES,LU,0)
0215 CALL BLANK(JMES)
0216 130 CALL CODE
0217 WRITES(JMES,131)
0218 131 FORMAT("ENTER MINIMUM FOR IMAGE")
0219 CALL TRMGN(JMES,LU,1,RTM,ICD,IRT)
0220 CALL BLANK(JMES)
0221 135 CALL SPCHR (IRTM,IRT)
0222 GO TO (1000,1000,1000,1000,935),IRT
0223 935 AMNIN=RTM
0224 IF (IRT .EQ. 5) GO TO 145
0225 9U IW=5
0226 GO TO 950
0227 145 CALL CODE
0228 WRITE(JMES,150) AMNIN
0229 150 FORMAT("MINIMUM FOR IMAGE =",F10.4)
0230 CALL TRMGN(JMES,LU,0)
0231 CALL BLANK(JMES)
0232 SCL=AMXIN-AMNIN
0233 IF (SCL.GE.1.0E-5) GO TO 155
CALL CODE
WRITE(JMES,197)
FORMAT(’ENTER ROW SCALE FACTOR’)
call trmgn(jmes,lu,1,rtm,icd,irtm)
call blank(jmes)
call spchr (irtm,irt)
y = rtm
if (irt.eq.5) go to 210
i = 7
 go to 950
100 call spchr (irtm,irt)
ys = rtm
if (irt.eq.5) go to 220
200 call spchr (irtm,irt)
ys = rtm
if (irt.eq.5) go to 220

check to see if interpolation is required.
if not branch.

compute new size of image

nnew = y*nrow+.5
ncols = x*icols+.5
if (x .eq. 1.0 .and. y .eq. 1.0) go to 260
if (ncols .le. 512) go to 230
call code
write(jmes,690) ncols,xs
format(’CALCULATED COLUMN SIZE = ’,1i5,(’SF = ’,1f5.2)’)
call trmgn(jmes,lu,0)
call blank(jmes)
call code
write(jmes,969)
format(’512 IS MAXIMUM ALLOWABLE OUTPUT’)
call trmgn(jmes,lu,0)
call blank(jmes)
call code
write(jmes,226)
format(’REENTER COLUMN SCALE FACTOR’)
call trmgn(jmes,lu,1,rtm,icd,irtm)
call blank(jmes)
go to 215
230 call code
write(jmes,695) nnew,ys,ncols,xs
format(’-- OUTPUT IMAGE--’,’14x,’rows = ’,’1i5,’(sf = ’,’1f5.2,’)’)
*14x,’columns = ’,’1i5,’(sf = ’,’1f5.2,’)’
call trmgn(jmes,lu,0)
call blank(jmes)
call code
write(jmes,1232)
format(’1. VALUES OKAY’)
call trmgn(jmes,lu,0)
call blank(jmes)
CALL CODE
WRITE(JMES,1233)
1233 FORMAT(" 2. REENTER SCALE FACTOR")
CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
CALL 3BLANK(JMES)
CALL SPCHR (IRTM,IRT)
CALL CKFLD(2,ICD,IRS)
GO TO (234,234,931),IRS
233 IW=9
GO TO 950
C
COMPUTE INCREMENTS FOR INTERPOLATION
C
234 MM=NNEW
IFLT=0
DY=FLOAT(NROW)/FLOAT(NNEW)
DX=FLOAT(ICOLS)/FLOAT(NCOLS)
IF(DY.LE.1.0) GO TO 235
700 CALL CODE
WRITE(JMES,1750)
1750 FORMAT(" IMAGE SHOULD BE FILTERED BEFORE INTERPOLATION")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,9750)
9750 FORMAT(" TO PREVENT ALIASING")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,760)
760 FORMAT(" 1. CONTINUE ")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
WRITE(JMES,761)
761 FORMAT(" 2. PREFILTER IMAGE")
CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
704 CALL SPCHR (IRTM,IRT)
GO TO (1000,1000,1000,1000,904),IRT
904 CALL CKFLD(2,ICD,IRS)
GO TO (702,702,710),IRS
710 IW=10
GO TO 950
C
SCHELULE FILTER TO PREVENT ALIASING
C
702 FCX=0.8*FLOAT(NCOLS)/FLOAT(ICOLS)
FCY=0.8*FLOAT(NNEW)/FLOAT(NROW)
NX=2
NY=2
ITME=0
CALL XYFLT(U,V,FCX,FCY,NX,NY,N,A,B)
DO 705 II=1,3
NAME(II)=NSON(II,1)
CALL EXEC (9,NAME,LU,0,NCOLS-1)
C
INITIALIZE FOR RESIZING
CALL CODE
WRITE(JMES,810)
FORMAT(" RESIZING OF IMAGE IN PROGRESS")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
C SCHEDULE DINTP FOR RESIZING IMAGE
INCW=LU+200B
DO 250 II=1,3
NAME(II)=NSON(II,2)
CLOSE WORK FILE
CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
CALL EXEC(13,ICNW,IPRAM(3),IPRAM(4),IPRAM(5))
CALL EXEC(23,NAME,IPRAM(1),NNEW,NCOLS,IPRAM(4),IPRAM(5))
OPEN WORK FILE
CALL OPEN(IDC8,IGET,6HW:0000,2,0,0,528)
IF(IGET.LT.0) GO TO 999
REMAP INTENSITY VALUES FOR IMAGE
OBTAIN NEW SIZE PARAMETERS
IGET=READL(-1,0,511,DIRC)
IF(IGET.LT.0) GO TO 999
IMCNT=0
IZCNT=0
DO 405 NN=1,NROW
READ IN NEW ROW
NNKI=NN-1
IGET=READL(NNM1,IFE,ICOLS-1,F)
IF(IGET.LT.0) GO TO 999
IF (IOPT.GT.6) GO TO 310
GO TO (306,310,320,330,340,350),IOPT
DO 305 I=1,ICOLS
G(I)=(F(I)-AMIN)*SCL+0.5
GO TO 360
DO 315 I=1,ICOLS
G(I)=F(I)+0.5
GO TO 360
DO 325 I=1,ICOLS
G(I)=(F(I)*SCL+0.5)
GO TO 360
DO 335 I=1,ICOLS
G(I)=(F(I)-AMIN)*SCL+0.5
GO TO 360
DO 345 I=1,ICOLS
G(I)=SCL*(F(I)-AMIN)**POWER+0.5
WRITE OUTPUT TO WORK FILE

IF (IGET.LT.0) GO TO 999

IF OUTPUT IS 8 BIT, WRITE TO DISPLAY

IF (ITYPE.EQ.15) GO TO 365

DO 370 I=1,ICOLS
   IF (G(I).GT.(FLOAT(IMAX)+0.5)) IMCNT=IMCNT+1
   IF (G(I).LT.0.0) IZCNT=IZCNT+1
   IOP(I)=MINO(IFIX(G(I)),IMAX)
   370 CONTINUE

IF (NN.LT.NTST) GO TO 400
   NTST=NTST+INUM
   CALL CODE
   WRITE(JMES,375) NN,NNNEW
   375 FORMAT("- RESIZE ROWS DONE/ TO DO ",1I4,"/",1I4,"/")
   CALL TRMGN(JMES,LU,0)
   CALL BLANK(JMES)

IF (ITYPE.NE.8) GO TO 405
   IGET=WLINE(NNMI,00,ICOLS-1,IOP)
   IF (IGET.LT.0) GO TO 999
   405 CONTINUE

CLOSE WORK FILE

AMAX=FLOAT(IMAX)
AMIN=0.0
CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
IRTCD=0
ITOT=NROW*ICOLS
ATOT=100.0/FLOAT(ITOT)
PZERO=ATOT*FLOAT(IZCNT)
PMAX=ATOT*FLOAT(IMCNT)
IF (PZERO.EQ.0.0 .AND. PMAX.EQ.0.0) GO TO 1000
CALL CODE
WRITE(JMES,410) PZERO,PMAX
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
410 FORMAT("- PERCENT CLIPPED AT ZERO =",F6.2,
1 " - PERCENT CLIPPED AT MAX =",F6.2)
420 CALL TRMGN(JMES,LU,0)
421 FORMAT(" 1. CONTINUE[ 2. RESCALE IMAGE"
425 CALL SPCHR(IRTM,IRT)
430 IW=11
435 GO TO 950
CALL READL(-1,0,511,DIRC)
GO TO 5
CALL CODE
WRITE(JMES,921)
FORMAT (' SCALING SIZE ERROR')
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CONTINUE
GO TO 999
CALL CODE
WRITE (JMES,21)
FORMAT ('/INVALID SELECTION/')
CALL TRMGN (JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK (JMES)
GO TO (15,55,85,110,135,170,200,215,232,704,425),IW
IF(IGET.EQ.-8) CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
CALL EXEC(6)
$  
$  
&ROTB T=00004 IS ON CR00022 USING 00002 BLKS R=0014
ASMB,R,L,C
NAM ROTB,6
ENT ROTB
EXT .ENTR
*  
WORD BSS 1
OUT BSS 1
*  
ROT8 NOP
JSB .ENTR
DEF WORD
LDA WORD,I
ALF,ALF
STA OUT,I
JMP ROTB,I
END
The subroutine `TRMGN` is used to write out and possibly read back terminal information necessary for program control. `JMES` is the message to be output to the LU. `IP` (if 0) means write only, (if 1) means to wait for a response from the operator. `RTM` is the return for real numbers, `ICD` is a return for integer numbers, and `IRTM` is the return for ASCII characters. All three types of return are generated each time this subroutine is called. The maximum output message is 80 characters long. The maximum input message is 10 characters long.

```fortran
DIMENSION JMES(40), IRTM(5)
ICNWD=ICNWD+LU
WRITE THE MESSAGE TO THE LU
CALL EXEC (2,ICNWD,JMES,40)
IF (IP .EQ. 0) RETURN
READ THE MESSAGE BACK FROM THE LU
CALL EXEC (1,ICNWD,IRTM,5)
CALL CODE
READ (IRTM,*)ICD
CALL CODE
READ (IRTM,*)RTM
RETURN
```

The subroutine `XYFLT` is written by W.E. Alexander and is a part of the spatial domain filtering package. It is a low pass recursive filter design routine with `FCX` not equal to `FCY`.

```fortran
COMPLEX P
DIMENSION U(3,3,2), V(3,3,2), A(3,2,2), B(3,2,2)
PI=3.141592654
D = 1.0E-10
N = 3
IF(NX.LE.2.AND.NY.LE.2)N=2
EPS = 1.0
DO 6 I=1,18
IF (I.GT. 12) GO TO 7
A(I)=0
B(I)=0
U(I)=0
V(I)=0
```

The subroutine `TRMGN` is on CR00022 using 00056 BLKS R=0439.
A(1,2,1) = 1.0
A(1,2,2) = 1.0
B(1,2,1) = 1.0
B(1,2,2) = 1.0

NXP = NX-1
TX = SIN(PI*FCX*0.5)/COS(PI*FCX*0.5)
TX = TX**2
IF(TX.LE.D)TX=D
CNX = TX**NXP/EPS
DX = C.25
IF(NX.EQ.3)DX=0.125
DX = CNX**DX

NYP = NY-1
TY = SIN(PI*FCY*0.5)/COS(PI*FCY*0.5)
TY = TY**2
IF(TY.LE.D)TY=D
CNY = TY**NYP/EPS
DY = 0.25
IF(NY.EQ.3)DY=0.125
DY = CNY**DY

CALCULATE COEFFICIENTS

DO 10 J=1,NNN
    DO 10 K = 1,2
    IF(K.EQ.2) GO TO 20
    CNX = CNX
    DD = DX
    IF (NX.EQ.3) GO TO 22
    THT = 135.0*PI/180.0
    GO TO 23
10    CN=CNY
    DD = DY
    IF (NY.EQ.3) GO TO 21
    THT=135.0*PI/180.0
    GO TO 23
20    CN=CNY
22    IF(J.EQ.1)THT=12.5*PI/180.0
21    IF(J.EQ.3)THT=15.5*PI/180.0
23    GO TO 23

ALP=COS(THT)
BET = SIN(THT)
S1 = 1.0+ALP*DD
S2 = 1.0-ALP*DD
S3 = BET*DD
S4=-S3
P=CMPLX(S1,S3)/CMPLX(S2,S4)
S1 = -2*REAL(P)
S2 = (CABS(P))**2
AA = 0.25 * ( 1.0 + S1 + S2 )
A(1,J,K) = AA
A(2,J,K) = 2.0*AA
A(3,J,K) = AA
B(1,J,K) = 1.0
B(2,J,K) = S1
B(3,J,K) = S2

C

C OBTAIN TWO DIMENSION FILTER
C

IF(NX.EQ.3)GO TO 30
A(1,2,1) = 1.0
B(1,2,1) = 1.0

30 IF(NY.EQ.3)GO TO 31
A(1,2,2) = 1.0
B(1,2,2) = 1.0

C

C DO 40 I = 1,3
DO 40 J = 1,3
DO 40 K = 1,2
U(I,J,K) = A(I,K,1)*A(J,K,2)
V(I,J,K) = B(I,K,1)*B(J,K,2)
40 RETURN
C

C ******************SUBROUTINE INTRP*************************
C
SUBROUTINE INTRP(AINT,Y,DX,NCOLS,ICOLS,FOP)
DIMENSION AINT(1), FOP(1), JMES(40)
CALL CODE
WRITE (JMES,150)
150 FORMAT(' NOW IN INTRP')
CALL TRMGN(JMES,LU,0)
C
IMAX=512
IMXP=IMAX+1
ICML=ICOLS-1
I=1
M=1
X=(I-1)*DX-(M-1)
IF(X.LT.1.0) GO TO 25
M=M+1
IF(M.GT.ICML) GO TO 50
GO TO 15
25 E=(AINT(M+1)-AINT(M))*X+AINT(M)
F=(AINT(M+IMXP)-AINT(M+IMAX))*X+AINT(M+IMAX)
FOP(I)=(F-E)*Y+E
&LBRSZ T=00004 IS ON CR00022 USING 00010 BLKS R=0100

0001 FTN4,L
0002 SUBROUTINE SPCHR (IATCD, IRT)
0003 C
0004 C THIS ROUTINE CHECKS FOR SPECIAL CHARACTERS IN THE INPUT DATA
0005 C
0006 IRT=5
0007 IF (IATCD.EQ. OB) IRT=0
0008 IF ((IATCD.EQ.1HX) .OR. (IATCD .EQ. 2HX ))IRT=1
0009 IF ((IATCD.EQ.1HB) .OR. (IATCD .EQ. 2HB ))IRT=2
0010 IF ((IATCD.EQ.1HD) .OR. (IATCD .EQ. 2HD ))IRT=3
0011 IF ((IATCD.EQ.1HR) .OR. (IATCD .EQ. 2HR ))IRT=4
0012 RETURN
0013 END
0014 SUBROUTINE CKFLD(IA,ICD,IRS)
0015 C
0016 C SUBROUTINE TO CHECK FOR CARRIAGE RETURN OR NUMERIC VALUE
0017 C
0018 IRS=ICD+1
0019 IF (ICD .EQ. OB) IRS=1
0020 RETURN
0021 END
0022 SUBROUTINE INFRM (IA,LU)
0023 DIMENSION IA(3)
0024 ICNWD=400B +LU
0025 CALL EXEC (2,ICNWD,IA,3)
0026 RETURN
0027 END
0028 C
0029 C
0030 SUBROUTINE XFLTR(AINT,ICOLS,F,A,B,FCX,ITME)
0031 DIMENSION B(3,2)
0032 DIMENSION F(1),AINT(1),A(3,2),WF(3),WG(3)
0033 C IF (ITME.EQ.0) CALL BOOST(0.0,1.0,FCX,2,A,B)
0034 ITME=ITME+1
0035 A1=A(1,1)
0036 A2=A(2,1)
0037 A3=A(3,1)
0038 B2=B(2,1)
0039 B3=B(3,1)
0040 C

0017 I=I+1
0018 IF(I.LE.NCOLS) GO TO 15
0019 IF(X.LT.1.0.AND.I.GT.NCOLS) GO TO 51
0020 COMPLETE INTERPOLATION
0021 50 FOP(NCOLS)=(AINT(ICOLS+IMAX)-AINT(ICOLS))*Y+AINT(ICOLS)
0022 51 CALL CODE
0023 WRITE(JMES,150)
0024 FORMAT(' NOW LEAVING INTRP')
0025 CALL TRMGN(JMES,LU,O)
0026 END

SUBROUTINE SPCHR (IATCD, IRT)

- This routine checks for special characters in the input data.
- It assigns different values to `IRT` based on various conditions involving `IATCD`.
- The conditions check for specific hexadecimal characters and assign `IATCD` values of OB, 1HX, 2HX, 1HB, 2HB, 1HD, 2HD, or 1HR.
- If `IATCD` is OB, `IRT` is set to 0. Otherwise, it is set to 1, 2, 3, or 4 depending on the condition.

SUBROUTINE CKFLD(IA,ICD,IRS)

- This subroutine checks for carriage return or numeric value.
- It sets `IRS` to `ICD` plus 1 if `ICD` is OB, otherwise it is set to 1.
- It returns after checking for the value.

SUBROUTINE INFRM (IA,LU)

- It declares an integer array `IA(3)`.
- It initializes `ICNWD` to the sum of 400B and `LU`.
- It calls the `EXEC` subroutine with arguments (2, `ICNWD`, IA, 3).
- It returns after calling the subroutine.

SUBROUTINE XFLTR(AINT,ICOLS,F,A,B,FCX,ITME)

- It declares several arrays and variables.
- It checks if `ITME` is 0, and if so, it calls the `BOOST` subroutine with specified arguments.
- It assigns values to `A1`, `A2`, `A3`, `B2`, and `B3` based on their indices.

The code includes logical checks and assignments to different variables, reflecting the context of handling special characters or values in the input data.
INITIALIZE

IMAX=512
INT=ICOLS/2-1
IMXP1=IMAX+1
ASTT=INT+INT+INT+INT
DO 10 I=1,3
WF(I)=ASTT
10 WG(I)=ASTT

START FORWARD FILTER

MM=IMXP1
WF(1)=INT(IMXP1)
WG(1)=2*WF(1)+2*WF(3)+A3*WF(3)+B2*WG(2)+B3*WG(3)

UPDATE

MM=MM+1
IF (MM.GT. ICOLS) GO TO 30
WF(1)=INT(MM)
GO TO 20

START REVERSE FILTER

ASTT=INT+INT
DO 40 I=1,3
WG(I)=ASTT
40 WG(I)=ASTT

MM=IMAX+ICOLS
WF(1)=INT(MM)
WG(1)=2*WF(1)+2*WF(2)+A3*WF(3)+B2*WG(2)+B3*WG(3)

UPDATE

MM=MM-1
IF (MM .LE. IMAX) GO TO 50
WF(1)=INT(MM)
GO TO 41
50 RETURN

END

SUBROUTINE BLANK (JMES)
DIMENSION JMES(40)
DO 10 I=1,40
JMES(I)=?NON
10 RETURN
END
INTEGER FUNCTION WFINT(NLINE,NPIXL,PMAX,PMIN,LU)

THIS SUBROUTINE IS USED IN CONJUNCTION WITH IMAGE PROCESSING
IT CREATES AND MAINTAINS AN IMAGE WORK FILE WITH PIXEL VALUES
STORED AS REAL NUMBERS TO PRESERVE PRECISION.

THIS ONE INITIALIZES THE PROCESS BY CREATING THE WORK FILE
AND RETURNING CERTAIN PERTINENT INFO TO CALLER. IT SHOULD
ONLY BE CALLED ONCE BY EACH CALLER. THE OTHER TWO ARE
READL, WHICH READS A PARTICULAR LINE AND RITEL WHICH WRITES
A PARTICULAR LINE.

LU = INTERACTIVE TERMINAL LU
NLINE = # LINES IN IMAGE
NPIXL = # PIXELS/LINE
PMAX = MAXIMUM PIXEL INTENSITY IN IMAGE (REAL)
PMIN = MINIMUM PIXEL INTENSITY IN IMAGE (REAL)

DIMENSION IDCB1(144),IRTN(5),I3(6)
EQUIVALENCE (IB2,IB(2)),(IB(3),RMAX),(IB(5),RMIN)

SCHEDULE BUILD WORK FILE PROGRAM
CALL EXEC(23,6HBLDFW ,LU)
GET RETURNED PARAMETERS
CALL RMPAR(IRTN)
WFINT = IRTN
IF (IRTN .LT. 0 ) RETURN
GET MAX MIN DATA
CALL OPEN(IDCB1,IERR,6HWF0000)
IF (IERR .LT. 0) GO TO 100
CALL READF(IDCB1,IERR,IB,6)
IF (IERR .LT. 0) GO TO 100
NLINE = IB
NPIXL = IB2
PMAX = RMAX
PMIN = RMIN
CALL CLOSE(IDCB1)
WFINT = 0
RETURN

100 WFINT = IERR
CALL CLOSE(IDCB1)
END
READ LINE FROM WORK FILE SUBROUTINE

INTEGER FUNCTION READL(LINE, IPIXL, JPIXL, RBUF)

COMMON /CBLK/ IDCB(528), TBUF(512), IFLAG

DIMENSION RBUF(512)

CHECK IF FILE OPEN

IF (IFLAG .EQ. 1) GO TO 100

MUST OPEN FILE

CALL OPEN(IDCBO, IERR, 6HWF0000, 2, 0, 0, 528)

IF (IERR .LT. 0) GO TO 999

IFLAG = 1

FILE OPENED--READ APPROPRIATE LINE

CALL READF(IDCBO, IERR, TBUF, 1024, LEN, LINE+2)

IF (IERR .LT. 0) GO TO 999

POSITION DATA IN BUFFER

ISTEP = 1

IF (IPIXL .GT. JPIXL) ISTEP = -1

J = 1

DO 110 I = IPIXL+1, JPIXL+1, ISTEP

RBUF(J) = TBUF(I)

110  J = J + 1

READL = 0

RETURN

ERROR

READL = IERR

END

WRITE WORK FILE SUBROUTINE

INTEGER FUNCTION RITEL(LINE, IPIXL, JPIXL, RBUF)

COMMON /CBLK/ IDCB(528), TBUF(512), IFLAG

DIMENSION RBUF(512)

CHECK IF FILE OPENED

IF (IFLAG .EQ. 1) GO TO 100
must open file

0116 C
0117 C CALL OPEN(IDCB, IERR, 6HFO000, 2, 0, 0, 528)
0118 IF (IERR .LT. 0) GO TO 999
0119 IFLAG = 1
0120 C
0121 C FILE OPENED--WRITE APPROPRIATE LINE
0122 C
0123 100 CALL READF(IDCB, IERR, TBUF, 1024, LEN, LINE+2)
0124 IF (IERR .LT. 0) GO TO 999
0125 C
0126 ISTEP = 1
0127 IF (IPIXL .GT. JPIXL) ISTEP = -1
0128 J = 1
0129 DO 110 I = IPIXL+1, JPIXL+I, ISTEP
0130 TBUF(I) = RBUF(J)
110 J = J+1
0132 C
0133 C CALL WRITF(IDCB, IERR, TBUF, 0, LINE+2)
0134 IF (IERR .LT. 0) GO TO 999
0135 C
0136 RITEL = 0
0137 RETURN
0138 C
0139 C ERROR RETURN
0140 C
0141 999 RITEL = IERR
0142 END
0143 C
0144 C
0145 C BLOCK DATA SUBROGRAM
0146 C
0147 C
0148 C BLOCK DATA
0149 C
0150 COMMON /CBLK/ IDCB(528), TBUF(512), IFLAG
0151 C
0152 DATA IFLAG/0/
0153 C
0154 END
0155 C
0156 C
0157 C CLOSE WORK FILE SUBROUTINE
0158 C
0159 C
0160 SUBROUTINE CLSWF(NLINE, NPIXL, PMAX, PMIN)
0161 C
0162 C COMMON /CRLK/ IDCL(528)
0163 C
0164 DIMENSION IB(6)
0165 C
0166 EQUIVALENCE (IB2, IB(2)), (B(3), RMAX), (IB(5), RMIN)
0167 C
0168 C
0169 C THIS ROUTINE IS USED TO CLOSE THE WORK FILE
0170 C
0171 C WRITE DATA ON WORK FILE
0172 C
0173 IB = NLINE
0174 IB2 = NPIXL
0175 RMAX = PMAX
0176 RMIN = PMIN
0177 C
0178 CALL WRITF(IDCB, IERR, IB, 6, 1)
0179 CALL CLOSE(IDCB)
This program is used to change the physical size of an image written by Winsor E. Alexander.

```fortran
001  FTN4,Q,T,C
002  PROGRAM DINTP
003  C
004  C  THIS PROGRAM IS USED CHANGE THE PHYSICAL SIZE OF AN IMAGE
005  C
006  C  WRITTEN BY WINSER E. ALEXANDER
007  C
008  DIMENSION AINT(1024),F(512),IPRAM(5),DIRC(515),INM(2)
009  EQUIVALENCE (F(1),DIRC(4)),(DIRC(1),INM(1))
010  EQUIVALENCE (INM(1),NROW),(INM(2),ICOLS),(DIRC(2),AMAX)
011  EQUIVALENCE (DIRC(3),AMIN)
012  C
013  C INPUT PARAMETERS (CALL RMPAR)
014  C IPRAM(1) = LOGICAL UNIT FOR INTERACTIVE DEVICE
015  C IPRAM(2) = NUMBER OF DESIRED ROWS IN OUTPUT IMAGE
016  C IPRAM(3) = NUMBER OF DESIRED COLUMNS IN OUTPUT IMAGE
017  C
018  C IMAGE TO BE USED IS ASSUMED TO BE IN IMAGE WORK FILE (WF0000
019  C
020  CALL RMPAR(IPRAM)
021  IU=IPRAM(1)
022  IF(LU.LE.0) LU=1
023  NNEW=IPRAM(2)
024  NCOLS=IPRAM(3)
025  NCM1=NCOLS-1
026  LMX=512
027  IMXP1=IMX+1
028  C
029  C OBTAIN PARAMETERS FROM CURRENT IMAGE
030  C
031  IGET=READL(-1,0,511,DIRC)
032  ICM1=ICOLS-1
033  IF(IGET.LT.0) GO TO 999
034  C
035  C INTERPOLATE IMAGE
036  C
037  DY=FLOAT(NRCW)/FLOAT(NNEW)
038  DX=FLOAT(ICOLS)/FLOAT(NCOLS)
039  IFLT=0
040  IF(DY.GT.1.0) STOP 111
041  IF(DX.LT.0.0) IFLT=1
042  IFR=0
043  IFE=0
044  C
```
INITIALIZE ARRAYS

IGET=READL(0,IFE,ICOLS-1,AINT(MXP1))
IF(IGET.LT.0) GO TO 999
IGET=READL(1,IFE,ICM1,AINT(1))
IF(IGET.LT.0) GO TO 999
MCNT=2
MORG=NROW
DO 100 KK=NNEW,1,-1
MCNT=MCNT+1
IGET=0
IF(MCNT.GT.NROW) IGET=-150
IF(IGET.LT.0) GO TO 999
IGET=READL(MCNT,IFE,ICM1,AINT)
IF(IGET.LT.0) GO TO 999
MORG=MORG-1

20 Y=(NNEW-KK)*DY-(NROW-MORG)
IF(Y.LT.1.0) GO TO 50
BRING IN NEW R:
CALL MOVE(AINT,ICOLS,IMXP1)
MCNT=MCNT+1
IGET=0
IF(MCNT.GT.NROW) IGET=-150
IF(IGET.LT.0) GO TO 999
IGET=READL(MCNT,IFE,ICM1,AINT)
IF(IGET.LT.0) GO TO 999
MORG=MORG-1
RECOMPUTE Y
GO TO 20
INTERPOLATE FOR NEW ROW
50 CALL INTRP(AINT,Y,DY,NCOLS,ICOLS,F)
OUTPUT CURRENT ROW
100 CALL RITEL(KK-1,O,NCK1,F)
NOTE THAT WORK FILE IS NOT CLOSED BY THIS PROGRAM
INSERT PARAMETERS IN WORK FILE
NROW=NNEW
ICOLS=NCOLS
CALL RITEL(-1,0,ICM1,DIRC)
CONTINUE
ERROR PROCESSING
WRITE(LU,1000) IGET
1000 FORMAT(" ERROR CODE = ",I5)
CALL EXEC(6)
END
SUBROUTINE MOVE(AINT, ICOLS, IMXP1)

THIS SUBROUTINE MOVES ICOLS ELEMENTS IN ARRAY AINT FROM
A START POINT OF 1 TO A START POINT OF IMXP1

DIMENSION AINT(1)

DO 10 I = 1, ICOLS
  10 AINT(IMXP1+I) = AINT(I)
RETURN
END

&WTape T=00004 IS ON CR00022 USING 00012 BLKS R=0127

FTN4,Q,C,T
PROGRAM WTAPE

THIS PROGRAM FROMS A PART OF THE IMAGE PROCESSING SYSTEM

IT IS USED TO STORE AN IMAGE ON TAPE AND THEN PURGE FROM DIS

THE IMAGE INVENTORY FILE IS UPDATED TO SHOW THAT THE IMAGE

TAPE

WRITTEN BY WINSER E. ALEXANDER

DIMENSION IDCB1(272), IDCB2(528), IMAGE(6), IPRAM(5), JNAME(3)

DIMENSION IDATA(512), ISIZE(2), IRTN(5), IBUF(15)

EQUIVALENCE (IBUF(12), ILOC), (IBUF(13), JNAME), (IBUF(7), NLINE)

EQUIVALENCE (IBUF(8), NPIXL), (IBUF(9), IPMIN), (IBUF(10), IPMAX)

GET INPUT PARAMETERS

CALL RMPAR(IPRAM)
LU = IPRAM(1)
IF (LU .LE. 0) LU = 1
LU2 = 8
POSITION TAPE

WRITE(LU, 45)
READ(LU, 46) IOPT
IF (IOPT .NE. 2) GO TO 1000
WRITE(LU, 51)
READ(LU, *) IFNUM
IFNUM .LE. 0 GO TO 1000
IFNUM .EQ. 1 GO TO 5

CALL EXEC(3, 400B+LU2)
IF (IFNUM .LE. 0) GO TO 1000
IF (IFNUM .EQ. 1) GO TO 5
DO 55 I=1,IFNUM-1
CALL EXEC(3,13008+LU2)
CONTINUE
GET IMAGE NAME FROM USER
WRITE(LU,10)
10 FORMAT(" ENTER IMAGE NAME (12 CHARACTERS /E TO EXIT)?")
READ(LU,20) IMAGE
20 FORMAT(6A2)
IF (IMAGE .EQ. 2H/E) GO TO 1001
OPEN DIRECTORY FILE
30 CALL OPEN(IDCBI,IERR,6HIMCIRC,1,2HIM,23,272)
IF(IERR.LT.0) GO TO 999
FIND IMAGE FILE
40 CALL READF(IDCBI,IERR,IBUF,I5,LEN)
IF(LEN.NE.-1) GO TO 35
WRITE(LU,36)
36 FORMAT("IMAGE NOT FOUND")
GO TO 5
35 IF(IERR.LT.0) GO TO 999
IF(ICMPW(IMAGE,IBUF,6).NE.0) GO TO 40
IMAGE FOUND
CLOSE DIRECTORY FILE AND OPEN IMAGE FILE
CALL CLOSE(IDCBI)
CALL OPEN(IDCBI,IERR,JNAME,I,2HIM,23,525)
IF(IERR.LT.0) GO TO 999
CHECK FOR TAPE ON TRANSPORT
45 FORMAT(" PUT TAPE ON TRANSPORT & PUT TAPE UNIT ON LINE."
*/" ENTER -GO- WHEN READY")
46 FORMAT(1A2)
WRITE(LU,48) IPMAX,IPMIN
48 FORMAT(" MAXIMUM VALUE = ",I18," MINIMUM = ",I18)
IF(IPMAX.LE.255) IMAGE WILL BE PACKED FOR OUTPUT (8 BIT IMAG
ITYPE = 15
IF(IPMAX.LE.255.AND.IPMIN.GE.0) ITYPE = 8
FORMAT("FILE Ø?")
C  OUTPUT DATA TO TAPE
D 0 I = 1, 512
CALL FILL(IDATA, 0, 512)
IERR = 0
IF (I .LE. NLINE) CALL READF(IDCJ2, IERR, IDATA, 512)
IF (IERR .LT. 0) GO TO 999
NOUT = 512
IF (ITYPE .EQ. 15) GO TO 70
C PACK DATA
DO 65 J = 1, 512, 2
CALL ROT(IARAY(J), ITEMP)
NOUT = 256
C WRITE DATA
CALL EXEC(2, LU2, IDATA, NOUT)
CONTINUE
CALL EXEC(3, 100B + LU2)
CALL CLOSE(IDCJ2)
GO TO 5
FILE ERROR
WRITE(LU, 996) IERR
FORMAT(" FILE ERROR = ", I4)
CALL CLOSE(IDCJ1)
CALL CLOSE(IDCJB2)
CALL EXEC(3, 400B + LU2)
CONTINUE
SUBROUTINE FILL(IARAY, ICHAR, NUM)
DIMENSION IARAY(NUM)
DO 10 I = 1, NUM
IARAY(I) = ICHAR
CONTINUE
END$
0001 
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047  
0048  
0049  
0050  
0051  
0052  
0053  
0054  
0055  
0056  
0057  
0058  
0059  
0060  
0061  
0062  

PROGRAM PLOTV
DIMENSION LU(5)
INTEGER IDCB(144),BUFF(4),NAME(3)
DATA NAME/2HDA,2HTA,2H1/

CALL RMPAR(LU)
CALL INITA(0)
CALL OPEN(IDCB,IERR,NAME)
IF (IERR .GE. 0) GO TO 20
WRITE(LU,10) IERR
10 FORMAT("OPEN ERROR",F5.0)
STOP

CONTINUE
CALL READF(IDCB,IERR,BUFF,IERR)
IF (IERR .GE. 0) GOTO 40
WRITE(LU,30) IERR
30 FORMAT("READ ERROR",F5.0)
GO TO 55
40 CONTINUE
CALL DVECT(BUFF,BUFF(2),BUFF(3),BUFF(4),LU)
55 STOP
END

SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)
DIMENSION IBUFF(5)
SCAL =255./1024.
IBUFF1 = (SCAL*IX1+0.5)+128
IBUFF2 = (SCAL*IY1+0.5)
IBUFF3 = (SCAL*IX2+0.5)+128
IBUFF4 = (SCAL*IY2+0.5)
IBUFF1 = IAND(IBUFF1,777B)
IBUFF2 = IAND(IBUFF2,377B)
IBUFF3 = IAND(IBUFF3,777B)
IBUFF4 = IAND(IBUFF4,377B)
IBUFF(1) = IBUFF1 + 44000B
IBUFF(2) = IBUFF2 + 64000B
IBUFF(3) = IBUFF1 + IBUFF3 + 50000B + 512
IBUFF(4) = IBUFF2 + IBUFF4 + 72000B + 256
CALL DRIVR(2,IBUFF,4)
RETURN
END
SUBROUTINE INITA(IBACK)
DIMENSION INIT(6)
DATA INIT/300003,100377B,10377B,24021B,26000B/
IF(IBACK .EQ. 1) INIT(4) = 24221B
CALL DRIVR(2,INIT,5)
RETURN
END
PROGRAM DPLAM

C THIS PROGRAM DISPLAYS THE FILTER CHARACTERISTICS

COMMON/CNT/XM(30,30)
COMMON/WORK/WO(130)
COMMON/QDCAZ/IQ(40)
INTEGER BUFF
COMMON/IDCB(144),BUFF(10)
DIMENSION IBUF(80),ILU(5),A(25),B(25),AA(5,5),BB(5,5)
DIMENSION XXX(31),YYY(31),XYP(31,2),LXY(15,3),AR(60)
COMPLEX HA,HB,Z(25)
EQUIVALENCE (A_4(1,1),XM(1,1)),(BB(1,1),XYP(1,1))
EQUIVALENCE (IREG(1),REG)
EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
C
CALL RMPAR(ILU)
LU=ILU(1)
MN=ILU(2)
AMAX = 0.0
AMIN =1000.0
NDIM = 30
NDIM = 30
C
C GET FILTER COEFF'S
CALL EXEC(14,1,IBUF,80)
C
IF(MN.EQ.3) GO TO 100
MNL=9
DO 10 J=1,3
K=1,3
II=J+(K-1)*3
A(II)=U(J,K,1)
10 B(II)=V(J,K,1)
GO TO 101
DO 103 I=1,5
J=1,5
AA(I,J)=0.0
103 BB(I,J)=0.0
DO 104 I=1,3
J=1,3
DO 105 I=1,3
K=1,3
DO 106 I=1,3
L=1,3
IK=I+K-1
JK=J+L-1
AA(IK,JK)=AA(IK,JK)+U(I,J,1)*U(K,L,2)
104 BB(IK,JK)=BB(IK,JK)+V(I,J,1)*V(K,L,2)
DO 101 J=1,5
K=1,5
II=J+(K-1)*5
A(II)=AA(J,K)
101 B(II)=BB(J,K)
MNL=25
WRITE(LU,1011)
FORMAT(2I101, COEFFICIENT MATRICES,/
WRITE(LU,105) (A(I),I=1,25)
WRITE(LU,105) (B(I),I=1,25)
105 FORMAT(5(1H ,5E10.2/)/)

COMPUTE THE CENTER OF OUTPUT ARRAY

WRITE(LU,12)
12 FORMAT(" ENTER MX FOR HORIZONTAL FREQUENCIES")
READ(LU,13)NX
13 FORMAT(112)
WRITE(LU,14)
14 FORMAT(" ENTER MY FOR VERTICAL FREQUENCIES")
READ(LU,13)MY

MXC=MX/2
MXT=2*MXC
NX=0
IF(MXT.NE.MX) NX=1
MYC=MY/2
MYT=2*MYC
NY=0
IF(MYT.NE.MY) NY=1
MXN=MXC+1
MYN=MYC+1
WRITE(LU,301)
300 FORMAT(" COMPUTE SQUARED MAGNITUDE /
301 FORMAT(" INITIALIZE ARRAY")

COMPUTE SQUARED MAGNITUDE CHARACTERISTIC

MX=MXT+NX
MY=MYT+NY
FCX=2.0/FLOAT(MX)
FCY=2.0/FLOAT(MY)
IF(MX.LE.101.AND.MY.LE.61) GO TO 204
IF(MX.LE.101) GO TO 202
MX=101
WRITE(LU,200)
200 FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR HORIZONTAL ")
202 IF(MY.LE.61) GO TO 203
MY=61
WRITE(LU,201)
GO TO 203
201 FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR VERTICAL ")
204 WRITE(LU,300)
204 DO 20 I=1,MX+1
20 DO 20 J=1,MY+1
20 XF=FCX*(I-MXC-1)
XXX(I)=XF
20 YF=FCY*(J-MY-1)
YYY(J)=YF
CALL ZWC(Z,XF,YF,MN)
HA=CMPLX(0.0,0.0)
HB=HA
DO 21 K=1,MNL
21 HA=HA+A(K)*Z(K)
21 HB=HB+B(K)*Z(K)
0116 21 CONTINUE
0117 XA=CABS(HA)
0118 XB=CABS(HB)
0119 IF(XB.LE.1.0E-20) XB=1.0E-20
0120 XA=XA/XB
0121 XM(I,J)=XA**2
0122 IF(XM(I,J).LT.AMIN) AMIN=XM(I,J)
0123 IF(XM(I,J).GT.AMAX) AMAX=XM(I,J)
0124 20 CONTINUE
0125 WRITE(LU,302)AMAX,AMIN
0126 302 FORMAT(" AMAX = ",1E10.2,3X,"AMIN = ",1E10.2/)
0127 C
0128 C SQUARED MAGNITUDE NORMALIZED
0129 C
0130 C OBTAIN W=1 PLOT FROM ARRAY
0132 C
0133 DO 22 I=1,XXN
0134 XYP(I,2)=XM(I+MXC,MYN)
0135 XERR(I)=XYP(I,2)
0136 22 XYP(I,1)=FCX*(I-1)
0137 WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,XXN)
0138 306 FORMAT(/1H ,6(1E10.2,3X)/)
0139 XL=0.0
0140 XU=1.0
0141 MC=2
0142 C
0143 C CALCULATE Z=W PLOT
0144 C
0145 X=(XXN)**2+(MYN)**2
0146 FCX=0.7071*FCX
0147 NUM=SQR(X)+1
0148 DO 30 I=1,XXN
0149 XF=FCX*(I-1)
0150 CALL ZWC(Z,XF,XF,XXN)
0151 HA=Cmplx(0.0,0.0)
0152 HB=HA
0153 DO 31 X=1,XXN
0154 HA=HA+40(K)*Z(K)
0155 31 HB=HB+B(K)*Z(K)
0156 XA=CABS(HA)
0157 XB=CABS(HB)
0158 IF(XB.LE.1.0E-20) XB=1.0E-20
0159 XA=XA/XB
0160 XYP(I,2)=XA**2
0161 30 XYP(I,1)=XF*1.414
0162 WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,XXN)
0164 C COMPUTE ERROR FUNCTION
0165 C
0166 ERR=0.0
0167 DO 350 J=1,MDIM
0168 350 ERR=ERR+((XERR(J)-XYP(J,2))/AMAX)**2
0169 WRITE(LU,360) ERR
0170 360 FORMAT(" RELATIVE ERROR = ",1E15.7/)
0171 C COMPUTE CONTOURS
0172 C
0174 C PLOT IMAGE OF TRANSFER FUNCTION
0175 C
0176 CALL CONTR(XXX,YYY,AMAX,AMIN,MX+1,MY+1,LU)
0177 C
0178 4443 STOP
0179 END
0180 SUBROUTINE ZWC(Z,XF,YF,MN)
0181 C
0182 C THIS SUBROUTINE COMPUTES COMPLEX
0183 C VALUES FOR Z**i*W**j FOR
0184 C ZW TRANSFORM AND PLACES RESULTS
0185 C IN ONE DIMENSIONAL ARRAY Z
0186 C XF=HORIZONTAL RELATIVE FREQUENCY
0187 C YF=VERTICAL RELATIVE FREQUENCY
0188 C
0189 COMPLEX Z(25),R,S
0190 IF(ABS(XF).EQ.1.0 ) XF = 0.99
0191 IF(ABS(YF).EQ.1.0) YF = 0.99
0192 PI=3.1415926
0193 RX=COS(P1*XF)
0194 RY=SIN(P1*XF)
0195 SX=COS(P1*YF)
0196 SY=SIN(P1*YF)
0197 R=CMPLX(RX,RY)
0198 S=CMPLX(SX,SY)
0199 IF(MN.GE.3) GO TO 20
0200 DO 10 J=1,3
0201 DO 10 K=1,3
0202 I=J+(K-1)*3
0203 10 Z(I)=S**(J-1)*R**(K-1)
0204 GO TO 22
0205 20 DO 21 J=1,5
0206 DO 21 K=1,5
0207 I=J+(K-1)*5
0208 21 Z(I)=S**(J-1)*R**(K-1)
0209 22 RETURN
0210 END
0211 BLOCK DATA WORK
0212 COMMON/WORK/WO(130)
0213 COMMON/CNT/XM(900)
0214 COMMON/QDCAZ/IQ(40)
0215 END
0216 $
54

SUBROUTINE CONTR(XXX, YYY, AMAX, AMIN, MX, MY, LU)

COMMON/CNT/XM(30,30)
DIMENSION XXX(MX), YYY(MY), CZ(9), ISIZE(2)
INTEGER BCFF, NAME9(3)
COMMON /IDCB(144), BUFF(4)
DATA NAME9/2HDA, 2HTA, 2H1 /
WRITE(LU,100)
100 FORMAT(“SELECT TYPE OF FILTER PLOT”)
READ(LU,*) IFLAG
C
C GENERATE CZ
C
CZ(1)=1.
DO 3 K=2,9
CZ(K)=CZ(K-1)+.25
3 CONTINUE
C CREATE A PLOT DATA FILE
C
ITYPE=3
ISIZE(1)=96
CALL PURGE(IDCB, IERR, NAME9)
IF(IERR .LT. 0) WRITE(LU,101) IERR
CALL CREAT(IDCB, IERR, NAME9, ISIZE, ITYPE)
IF(IERR .GE. 0) GO TO 201
WRITE(LU,101) IERR
101 FORMAT(“CREATE ERROR”,F5.0)
STOP
C
DO 3-D PLOTS
C
201 IF (IFLAG.EQ.1) GO TO 10
20 IF (IFLAG.EQ.2) GO TO 20
20 CONTINUE
CALL SET3D(1.,-1.,1.,-1., AMAX, AMIN, 2., 0., 5., 5)
CALL PLT3D(XXX, YYY, XM, 30, MX, MY, LU)
GOTO 30
C
DO ISOGRAMS
C
10 CONTINUE
DO 11 I=1,MX
DO 11 J=1,MY
XM(I,J)=XM(I,J)/AMAX
11 CONTINUE
CALL SET2D(1.,-1.,1.,-1.,3.,0.,1.)
CALL PLT2D(XXX, YYY, XM, 30, MX, MY, CZ, 9, LU)
30 CONTINUE
CALL CLOSE(IDCB)
RETURN
END
SUBROUTINE SET2D(ALPMAX, ALPMIN, BETMAX, BETMIN, IORGN, LALPCL, AL

COMMON/ QDCAZ/ IXYXYB(4, 4), XZ, AX, BX, YZ, AY, BY

DATA XCNTR, YCNTR, EL/512., 512., 1000./

XZ=XCNTR
YZ=YCNTR
IF(ALTOBL-1.)6, 7, 8
6 CONTINUE
ELALP=ALTOBL*EL
ELBET=EL
GO TO 9
7 CONTINUE
ELALP=EL
ELBET=EL
GO TO 9
8 CONTINUE
ELALP=EL
ELBET=EL/ALTOBL
9 CONTINUE
IF (IORGN.EQ.1) GO TO 1
IF (IORGN.EQ.2) GO TO 2
IF (IORGN.EQ.3) GO TO 3
GO TO 4
1 CONTINUE
IF (LALPCL.EQ.1) GO TO 10
BX=0.
AY=0.
AX=-ELALP/(ALPMAX-ALPMIN)
BY=-ELBET/(BETMAX-BETMIN)
XZ=XZ+.5*ELALP
YZ=YZ+.5*ELBET
GO TO 5
10 CONTINUE
AX=0.
BY=0.
BX=-ELBET/(BETMAX-BETMIN)
AY=-ELALP/(ALPMAX-ALPMIN)
XZ=XZ+.5*ELBET
YZ=YZ+.5*ELALP
GO TO 5
2 CONTINUE
IF (LALPCL.EQ.1) GO TO 20
AX=0.
BY=0.
AY=ELALP/(ALPMAX-ALPMIN)
BX=-ELBET/(BETMAX-BETMIN)
XZ=XZ+.5*ELBET
YZ=YZ-.5*ELALP
GO TO 5
20 CONTINUE
AY=0.
BX=0.
AX=-ELALP/(ALPMAX-ALPMIN)
BY=-ELBET/(BETMAX-BETMIN)
XZ=XZ+.5*ELALP
YZ=YZ-.5*ELBET
GO TO 5
3 CONTINUE
0111 IF (IALPCL.EQ.1) GO TO 30
0112 AY=0.
0113 BX=0.
0114 AX=ELALP/(ALPMAX-ALPMIN)
0115 BY=ELBET/(BETMAX-BETMIN)
0116 XZ=XZ-.5*ELALP
0117 YZ=YZ-.5*ELBET
0118 GO TO 5
0119 30 CONTINUE
0120 AX=0.
0121 BY=0.
0122 AX=ELALP/(ALPMAX-ALPMIN)
0123 BX=ELBET/(BETMAX-BETMIN)
0124 XZ=XZ-.5*ELBET
0125 YZ=YZ+.5*ELALP
0126 GO TO 5
0127 40 CONTINUE
0128 IF(IALPCL.EQ.1) GO TO 40
0129 AX=0.
0130 BY=0.
0131 AX=ELALP/(ALPMAX-ALPMIN)
0132 BX=ELBET/(BETMAX-BETMIN)
0133 XZ=XZ-.5*ELALP
0134 YZ=YZ+.5*ELBET
0135 GO TO 5
0136 5 CONTINUE
0137 AX=0.
0138 BX=0.
0139 AX=ELALP/(ALPMAX-ALPMIN)
0140 BY=ELBET/(BETMAX-BETMIN)
0141 XZ=XZ-.5*ELALP
0142 YZ=YZ+.5*ELBET
0143 5 CONTINUE
0144 XZ=XZ-AX*ALPMIN-BX*BETMIN
0145 YZ=YZ-AY*ALPMIN-BY*BETMIN
0146 IXYXYB(1,1)=IFIX(XZ+AX*ALPMIN+BX*BETMIN)
0147 IXYXYB(2,1)=IFIX(YZ+AY*ALPMIN+BY*BETMIN)
0148 IXYXYB(1,2)=IFIX(XZ+AX*ALPMIN+BX*BETMAX)
0149 IXYXYB(2,2)=IFIX(YZ+AY*ALPMIN+BY*BETMAX)
0150 IXYXYB(1,3)=IFIX(XZ+AX*ALPMAX+BX*BETMIN)
0151 IXYXYB(2,3)=IFIX(YZ+AY*ALPMAX+BY*BETMIN)
0152 IXYXYB(1,4)=IFIX(XZ+AX*ALPMAX+BX*BETMAX)
0153 IXYXYB(2,4)=IFIX(YZ+AY*ALPMAX+BY*BETMAX)
0154 IXYXYB(3,1)=IXXYX(1,2)
0155 IXYXYB(4,1)=IXXYX(2,1)
0156 IXYXYB(3,2)=IXXYX(1,3)
0157 IXYXYB(4,2)=IXXYX(2,3)
0158 IXYXYB(3,3)=IXXYX(1,4)
0159 IXYXYB(4,3)=IXXYX(2,4)
0160 IXYXYB(3,4)=IXXYX(1,1)
0161 IXYXYB(4,4)=IXXYX(2,1)
0162 RETURN
0163 END
SUBROUTINE PLT2D(ALPHA,BETA,GAMMA,IDMN,IALPHA,JBETA,C,NUMC
1
COMMON/QDCAZ /IXXYB(4,4),XZ,AX,BX,YZ,AY,BY
0170 DIMENSION ALPHA(1),BETA(1),GAMMA(IDMN,1),C(1)
0171 INTEGER BUFF(4),NAME9(3)
0172 COMMON IDCB(144)
0173 COMMON/WORK/IXXY(2,62),JXJY(2,62)
0174 DATA NAME9/2HDA,2HTA,2H1 /
0175 CALL OPEN(IDCB,IERR,NAME9)
0176 NOGRID=0
0177 IF(NUMC.LE.0) GO TO 1
0178 IF (IALPHA)2,1,3
0179 2 CONTINUE
0180 NOGRID=1
0181 3 CONTINUE
0182 IMAX=LABS(IALPHA);
0183 IMAXP2=IMAX+2
0184 IF (JBETA)4,1,5
0185 4 CONTINUE
0186 NOGRID=1
0187 5 CONTINUE
0188 JMAX=LABS(JBETA)
0189 IF (NOGRID.EQ.1) GO TO 6
0190 DO 7 K=1,4
0191 CALL FLBUF(IXXYB(1,K),IXXYB(2,K),
0192 1 IXYXYB(3,K),IXXYXYB(4,K),BUFF)
0193 CALL WRITF(IDCDB,IERR,BUFF,4)
0194 7 CONTINUE
0195 6 CONTINUE
0196 DO 8 N=1,NUMC
0197 DO 9 I=1,IMAXP2
0198 IXY(1,I)=0
0199 JXJY(1,I)=0
0200 9 CONTINUE
0201 DO 10 J=1,JMAX
0202 IF(J.EQ.1) GO TO 111
0203 DO 12 I=1,IMAX
0204 IF(GAMMA(I,J).EQ.GAMMA(I,J-1)) GO TO 13
0205 IF (GAMMA(I,J).GE.C(N).AND.C(N).GE.GAMMA(I,J-1)) GO TO 14
0206 IF(GAMMA(I,J).LE.C(N).AND.C(N).LE.GAMMA(I,J-1)) GO TO 14
0207 13 CONTINUE
0208 JXJY(1,I+1)=0
0209 GO TO 12
0210 14 CONTINUE
0211 BETINT=BETA(J-1)+(BETA(J)-BETA(J-1))*(C(N)-GAMMA(I,J-1))/(GA
0212 1J)-GAMMA(I,J-1))
0213 ALPINT=ALPHA(I)
0214 IXR=IFIX(XZ+AX*ALPINT+BX*BETINT)
IYR=IFIX(YZ+AY*ALPINT+BY*BETINT)

IF(JXJY(1,I).EQ.0) GO TO 15
CALL FLBUF(IXR,IYR,JXJY(1,I),JXJY(2,I),BUFF)
CALL WRITF(IDC3,IERR,BUFF,4)

CONTINUE

IF(IYR(1,I+1).EQ.0) GO TO 16
CALL FLBUF(IXR,IYR,IYR(1,I+1),IYR(2,I+1),BUFF)
CALL WRITF(IDC3,IERR,BUFF,4)

CONTINUE

IF(IYR(1,I+2).EQ.0) GO TO 17
CALL FLBUF(IXR,IYR,IYR(1,I+2),IYR(2,I+2),BUFF)
CALL WRITF(IDC3,IERR,BUFF,4)

CONTINUE

JXJY(1,I+1)=IXR
JXJY(2,I+1)=IYR

DO 18 I=2,IMAX
IF(GAMMA(I,J).EQ.GAMMA(I-1,J)) GO TO 19
IF(GAMMA(I,J).GE.C(N).AND.C(N).GE.GAMMA(I-1,J)) GO TO 20
IF(GAMMA(I,J).LE.C(N).AND.C(N).LE.GAMMA(I-1,J)) GO TO 20

CONTINUE

IXIY(1,I+1)=0
GO TO 18

CONTINUE

ALPINT=ALPHA(I-1)+(ALPHA(I)-ALPHA(I-1))*(C(N)-GAMMA(I-1,J))/
1(I,J)-GAMMA(I-1,J)
BETINT=BETA(J)
IXR=IFIX(XZ+AX*ALPINT+BX*BETINT)
IYR=IFIX(YZ+AY*ALPINT+BY*BETINT)

IF(JXJY(1,I).EQ.0) GO TO 21
CALL FLBUF(IXR,IYR,JXJY(1,I),JXJY(2,I),BUFF)
CALL WRITF(IDC3,IERR,BUFF,4)

CONTINUE

IF(IYR(1,I+1).EQ.0) GO TO 22
CALL FLBUF(IXR,IYR,IYR(1,I+1),IYR(2,I+1),BUFF)
CALL WRITF(IDC3,IERR,BUFF,4)

CONTINUE

IF(JXJY(1,I+1).EQ.0) GO TO 23
CALL FLBUF(IXR,IYR,JXJY(1,I+1),JXJY(2,I+1),BUFF)
CALL WRITF(IDC3,IERR,BUFF,4)

CONTINUE

IXIY(1,I+1)=IXR
IXIY(2,I+1)=IYR

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE
0265 C ***************************************************************
0266 C
0267 C
0268 C
0269 SUBROUTINE SET3D(ALPMAX, ALPMIN, BETMAX, BETMIN, GAMMAX, GAMMIN,
0270 IORGN, IALPCL, GAMFAC, ALPFAC)
0271 COMMON/QDCAZ,lIXXYB(4,5),XZ,AX,BX,YZ,AY,BY,CY
0272 DATA ELX,ELY,EXLEFT,YBOTOM/1012.,856.,12.,156./
0273 AX=ALPFAC*ELX/(ALPMAX-ALPMIN)
0274 AY=ALPFAC*(1.-GAMFAC)*ELX/(ALPMAX-ALPMIN)
0275 BX=(1.-ALPFAC)*ELX/(BETMAX-BETMIN)
0276 BY=(1.-ALPFAC)*(1.-GAMFAC)*ELY/(BETMAX-BETMIN)
0277 CY=GAMFAC*ELY/(GAMMAX-GAMMIN)
0278 YZ=-CY*GAMMIN+YBOTOM
0279 XZ=EXLEFT
0280 IF(IORGN.EQ.1)GO TO 1
0281 IF(IORGN.EQ.2)GO TO 2
0282 IF(IORGN.EQ.3)GO TO 3
0283 GO TO 4
0284 1 CONTINUE
0285 XZ=XZ+ELX
0286 AX=-AX
0287 BX=-BX
0288 IF(IALPCL.EQ.1)GO TO 10
0289 YZ=YZ+BY*(BETMAX-BETMIN)
0290 BY=-BY
0291 ALPVRT=ALPMAX
0292 BETVRT=BETMIN
0293 GO TO 5
0294 10 CONTINUE
0295 YZ=YZ+AY*(ALPMAX-ALPMIN)
0296 AY=-AY
0297 ALPVRT=ALPMIN
0298 BETVRT=BETMAX
0299 GO TO 5
0300 2 CONTINUE
0301 ALPVRT=ALPMAX
0302 BETVRT=BETMAX
0303 IF(IALPCL.EQ.1)GO TO 20
0304 XZ=XZ+BX*(BETMAX-BETMIN)
0305 BX=-BX
0306 GO TO 5
0307 20 CONTINUE
0308 XZ=XZ+AX*(ALPMAX-ALPMIN)
0309 AX=-AX
0310 GO TO 5
0311 3 CONTINUE
IF(IALPCL.EQ.1) GO TO 30
YZ=YZ+AY*(ALPMAX-ALPMIN)
AY=-AY
ALPVRT=ALPMIN
BETVRT=BETMAX
GO TO 5
30 CONTINUE
YZ=YZ+AY*(ALPMAX-ALPMIN)
BY=-BY
ALPVRT=ALPMAX
BETVRT=BETMIN
GO TO 5
4 CONTINUE
YZ=YZ+BY*(BETMAX-BETMIN)
BY=-BY
ALPVRT=ALPMAX
BETVRT=BETMIN
AY=-AY
BY=-BY
IF(IALPCL.EQ.1) GO TO 40
XZ=XZ+AX*(ALPMAX-ALPMIN)
AX=-AX
GO TO 5
40 CONTINUE
XZ=XZ+AX*(ALPMAX-ALPMIN)
2X=-BX
5 CONTINUE
XZ=XZ-BX*BETMIN-AX*ALPMAX
YZ=YZ-BY*BETMIN-AY*ALPMAX
IXXYYB(1,1)=XZ+AX*ALPMAX+BX*BETMIN
IXXYYB(2,1)=YZ+AY*ALPMAX+BY*BETMIN+CY*GAMMIN
IXXYYB(3,1)=XZ+AX*ALPMAX+BX*BETMIN
IXXYYB(4,1)=YZ+AY*ALPMAX+BY*BETMIN+CY*GAMMIN
IXXYYB(1,2)=IXXYYB(3,1)
IXXYYB(2,2)=IXXYYB(4,1)
IXXYYB(3,2)=XZ+AX*ALPMAX+BX*BETMAX
IXXYYB(4,2)=YZ+AY*ALPMAX+BY*BETMAX+CY*GAMMIN
IXXYYB(1,3)=IXXYYB(3,2)
IXXYYB(2,3)=IXXYYB(4,2)
IXXYYB(3,3)=XZ+AX*ALPMAX+BX*BETMAX
IXXYYB(4,3)=YZ+AY*ALPMAX+BY*BETMAX+CY*GAMMIN
IXXYYB(1,4)=IXXYYB(3,3)
IXXYYB(2,4)=IXXYYB(4,3)
IXXYYB(3,4)=IXXYYB(1,1)
IXXYYB(4,4)=IXXYYB(2,1)
IXXYYB(1,5)=XZ+AX*ALPVRT+BX*BETVRT
IXXYYB(2,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMIN
IXXYYB(3,5)=IXXYYB(1,5)
IXXYYB(4,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMAX
FORMAT(5(7X,17))
RETURN
END
SUBROUTINE PLT3D(ALPHA, BETA, GAMMA, IDMN, IALPHA, JBETA, IFILE, L

DIMENSION ALPHA(1), BETA(1), GAMMA(IDMN,1)
COMMON/WORK/LASTXY(2,200)
COMMON/QDCA:/IXYXYB(4,5),XZ,AX,BX,YZ,AY,BY,CY
INTEGER BUFF(4)
COMMON IDC8(144)
NOGRID=0
IF(ILALPHA)1,2,3
1 CONTINUE
NOGRID=1
3 CONTINUE
IMAX=IABS(ILALPHA)
4 CONTINUE
IF(JBETA)4,2,5
JMAX=IABS(JBETA)
5 CONTINUE
IF(NOGRID.EQ.1)GO TO 6
   FORMAT (5(7X,17))
6 CONTINUE
   FORMAT (10X,"GOOD",I5)
DO 7 K=1,5
7 CONTINUE
CALL FLBUF(IXYXYB(1,K),IXYXYB(2,K),
1 IXYXYB(3,K),IXYXYB(4,K),BUFF)
CALL WRITF(IDC8,IERR,BUFF,4)
9 CONTINUE
IF(J.EQ.1)GO TO 10
   CALL FLBUF(IXR,IYR,LASTXY(1,I-1),LASTXY(2,I-1),BUFF)
   CALL WRITF(IDC8,IERR,BUFF,4)
10 CONTINUE
LASTXY(1,I)=IXR
LASTXY(2,I)=IYR
8 CONTINUE
2 CONTINUE
RETURN
END
SUBROUTINE FLBUF(IX1,IY1,IX2,IY2,BUFF)
INTEGER BUFF(4)
BUFF(1) = IX1
BUFF(2) = IY1
BUFF(3) = IX2
BUFF(4) = IY2
RETURN
END
PROGRAM FDIGN

COMMON/WORK/WO(75)

DIMENSION NAME1(3),NAME2(3),NAME3(3),IRTN(5)
DIMENSION IDC8(144),NAME4(3),NAME5(3)
DIMENSION U(3,3,2),V(3,3,2),ILU(5),IBUF(80),IBUF2(80)
EQUIVALENCE (ILU(1),LU),(IBUF,IBUF2)

DATA NAME1/2HST,2HAB,2HLI/
DATA NAME2/2HDP,2HLA,2HM/
DATA NAME3/2HCO,2HEF,2HFS/
DATA NAME4/2HFI,2HRO,2H/
DATA NAME5/2HPL,2HOT,2HV/
DATA V/18*0./
DATA U/18*0./
DATA IBUF/80*0/

C GET LU
CALL RMPAR(ILU)

C GET FILTER PARAMETERS

MN=1
WRITE(LU,400)

400 FORMAT(" SELECT FILTER DESIGN"/
1. LOWPASS"/
2. BANPPA
1 3. HIGHPASS"/
4. BOOST FILTER"/
5. TDLPF (LOWPASS)"
2" 6. ROTATING FILTER "/
7. NON-RECURSIVE FILTERS " )
READ(LU,401) IFIL
IF(IFIL.EQ.4) GO TO 500
IF(IFIL.EQ.3)GO TO 410
IF(IFIL.EQ.6) GO TO 408
IF(IFIL .EQ. 7) GO TO 1102
WRITE(LU,402)

402 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS"/
READ(LU,403)F2

403 FORMAT(F2.2)

404 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR HIGHPASS"/
READ(LU,403) F1

407 WRITE(LU,405)

405 FORMAT(" ENTER NUMBER OF FILTER STAGES"/
READ(LU,401) MN
IBUF(40) =MN

401 FORMAT(111)
GO TO 411

WRITE(LU,510)

510 FORMAT(" SELECT OPTION"/
1. LOW BOOST FILTER"/
2. HI
ST FILTER"/
READ(LU,515) IOPT

515 FORMAT(111)
IF(IOPT.GE.0.AND.IOPT.LE.2) GO TO 530
WRITE(LU,520)
520 FORMAT(" INVALID RESPONSE")
GO TO 500
C
WRITE(LU,535)
535 FORMAT(" ENTER BOOST MAGNITUDE")
READ(LU,*1) BF
WRITE(LU,540)
540 FORMAT(" ENTER RELATIVE BREAK FREQUENCY")
READ(LU,545) F1
IF(IOPT.EQ.0.OR.IOPT.EQ.1) WRITE(LU,545) BF,F1
545 FORMAT(" BOOST MAGNITUDE = ",1E15.5," FREQUENCY = ",1F10.5," *OHBOOST FILTER."/,% IS THIS CORRECT")
IF(IOPT.EQ.2) WRITE(LU,550) BF,F1
550 FORMAT(" BOOST MAGNITUDE = ",1E15.5," BREAK FREQUENCY = ",1F
* FOR HIGH BOOST FILTER."/,% IS THIS CORRECT")
READ(LU,551) IRES
551 FORMAT(1A1)
IF(IRES.EQ.1HY.OR.IRES.EQ.1Hy) GO TO 552
552 BF=SQRT(ABS(BF))
IF(IOPT.EQ.2) GO TO 560
557 ALP=BF
560 ALP=BF
IF(IFIL.EQ.1) CALL LPFLT(U,V,F2,MN,LU)
IF(IFIL.EQ.2) CALL BPFLT(U,V,F1,F2,MN,LU)
IF(IFIL.EQ.3) CALL BSTFT(U,V,F1,MN,1.0,-1.0,LU)
IF(IFIL.EQ.4) CALL BSTFT(U,V,F1,MN,ALP,BET,LU)
IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
IF(IFIL.EQ.6) CALL ROTAE(U,V,MN,LU)
IL.E CALL BPFLT(U,V,F1,F2,MN,LU)
IF(IFIL.EQ.3) CALL BSTFT(U,V,F1,MN,1.0,-1.0,LU)
IF(IFIL.EQ.4) CALL BSTFT(U,V,F1,MN,ALP,BET,LU)
IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
IF(IFIL.EQ.6) CALL ROTAE(U,V,MN,LU)
C IF(IFIL.EQ.7) CALL FIR(U,MN,WR,LU)
CONTINUE
IF(IFIL.EQ.1) ILU(2) = MN
IF(IFIL.EQ.2) ILU(2) =MN + 1
IF(IFIL.EQ.6) ILU(2) =MN-1
SCHEDULE STABILITY TEST-STABT
CALL EXEC(23,NAME1,LU,ILU(2),0,0,0,IBUF,80)
SCHEDULE DISPLAY PROGRAM-DPLAY
CALL EXEC(23,NAME2,LU,ILU(2),0,0,0,IBUF2,80)
IF(IFIL .EQ. 3) IBUF(40) =MN
IF(IFIL .EQ. 4) IBUF(40) =MN
IF(IFIL .EQ. 6) IBUF(40) =MN-1
CALL PURGE(IDCB,IERR,NAME3,2HES)
IF(IERR .LT. 0) WRITE(LU,1101) IERR
CALL CREAT(IDCB,IERR,NAME3,2,3,2HES)
IF(IERR .LT. 0) WRITE(LU,1101) IERR
1101 FORMAT("CREATE ERROR",F5.0)
CALL WRITF(IDCB,IERR,IBUF,80)
IF(IERR .LT. 0) WRITE(LU,1101) IERR
CALL CLOSE(IDCB,IERR)
WRITE(LU,1105)
1105 FORMAT(" CREATE ERROR",F5.0)
CALL WRITF(IDCB,IERR,IBUF,80)
IF(IERR .LT. 0) WRITE(LU,1101) IERR
CALL EXEC(23,NAME5,LU,0,0,0,0)
GO TO 1107
1106 CONTINUE
CALL HP48A(LU)
1107 CONTINUE
CALL EXEC(6)
C
C SCHEDULE NON-RECURSIVE FILTERS
1102 CONTINUE
CALL EXEC(23,NAME4,LU,0,0,0,0)
STOP
END
SUBROUTINE BPFLT(U,V,F1,F2,N,LU)
C	 WRITTEN BY W. E. ALEXANDER
C	 F1----BREAK FREQUENCY FOR LOW FREQUENCY CUTOFF
C	 F2----BREAK FREQUENCY FOR HIGH FREQUENCY CUTOFF
C	 SUBROUTINE DESIGNS BANDPASS FILTER FROM LPFLT AND HPFLT
DIMENSION U(3,3,2),V(3,3,2),AA(3,3,2),BB(3,3,2)
IF(F1.LT.0.001.OR.F1.GT.0.999) RETURN
IF(F2.LT.0.001.OR.F2.GT.0.999) RETURN
FC=AMAX1(F1,F2)
CALL LPFLT(AA,BB,FC,N,LU)
DO 20 I=1,3
DO 20 J=1,3
U(I,J,1)=AA(I,J,1)
20 V(I,J,1)=BB(I,J,1)
FC=AMAX1(F1,F2)
CALL LPFLT(AA,BB,FC,N,LU)
DO 20 I=1,3
DO 20 J=1,3
U(I,J,1)=AA(I,J,1)
20 V(I,J,1)=BB(I,J,1)
FC=AMAX1(F1,F2)
CALL LPFLT(AA,BB,FC,N,LU)
SUBROUTINE BSTFT(U, V, FC, N, ALP, BET, LU)

FREQUENCY BOOST DESIGN ROUTINE

FC = RC*S/PI WHERE RC IS THE CUTOFF FREQUENCY IN RADIANS AND 
S IS THE SAMPLING INTERVAL. THUS FC = 0.5 GIVES A CUTO 
FREQUENCY AT ONE FOURTH SAMPLING FREQUENCY.

FOR HIGH PASS FILTER, LET ALP = 1.0 AND BET = -1.0
FOR HIGH FREQUENCY BOOST FILTER, ALP = BF AND BET = -1.0*(BF-1.0 
WHERE BF = SQRT(DESIRED FILTER GAIN AT ONE, HALF SAMPLING)
FOR LOW FREQUENCY BOOST FILTER, ALP = 1.0 AND BET = (BF-1.0)

DIMENSION U(3,3,2), V(3,3,2)

DO 21 K=1,2
DO 21 I=1,3
DO 21 J=1,3
U(I,J,K)=0.0
21 V(I,J,K)=0.0

WRITE(LU,14) FC

14 FORMAT(1HO, " FC = \", 1E22.5, \", FOR BOOST FILTER")

PI = 3.141592654
D = 1.0E-10
PWR = 0.25
IF (N.EQ.3) PWR = 0.125
EPS = 2.0**PWR - 1.0
IF(N.EQ.2.AND.BET.LT.0.0) EPS = 1.50702
IF(N.EQ.3.AND.BET.LT.0.0) EPS = 2.4711
XP = PI*FC*0.5
T = (SIN(XP)/COS(XP))**2.0
IF(T.GT.D) GO TO 10
AAA = EPS/D
GO TO 11
10 AAA = EPS/T
11 SALP = SQRT(AAA)
DEM = 1.0 - 2.0*AAA
IF(DEM.LT.D) GO TO 12
Pi = (+2.0*AAA - 2.0*SQRT(2.0)*SALP + 1.0)/DEM
GO TO 20
12 F = -1.0*DEM
IF(F.GT.D) GO TO 13
P1 = 0.0
13 PL = (1.0 + P1)**2
AS = A**2
POS = P1**2
R = 4.0*(POS**2) + ((1.0 + POS)**2 - 4.0*AS) - 4.0*(P1*(1.0 + POS) - 2.0*A
IF(ABS(R).LT.D) R = SIGN(D, R)
S = (1.0 - P1)**4/R
U(1,1,1)=S*(ALP*POS+BET*AS)
U(1,2,1)=S*(ALP*Pl*(1.0+POS)+2.0*BET*AS)
U(2,1,1)=U(1,2,1)
U(2,2,1)=S*(ALP*(1.0+POS)**2+4.0*BET*AS)
U(1,3,1)=S*(ALP*POS+BET*AS)
U(3,1,1)=U(1,3,1)
U(2,3,1)=S*(ALP*Pl*(1.0+POS)+2.0*BET*AS)
U(3,2,1)=U(2,3,1)
U(3,3,1)=S*(ALP*POS+BET*AS)

V(1,1,1)=1.0
V(1,2,1)=2.0*Pl
V(2,1,1)=V(1,2,1)
V(2,2,1)=4.0*POS
V(1,3,1)=POS
V(3,1,1)=V(1,3,1)
V(2,3,1)=2.0*Pl*POS
V(3,2,1)=V(2,3,1)
V(3,3,1)=POS**2

IF(N.EQ.2) GO TO 27
DO 26 I=1,3
DO 26 J=1,3
U(I,J,2)=U(I,J,1)
26 V(I,J,2)=V(I,J,1)
N = N-1
RETURN
END

SUBROUTINE LPFLT(U,V,FC,N,LU)
C LOW PASS RECURSIVE FILTER DESIGN ROUTINE
FC=RC*S/PI
WHERE RC IS THE CUTOFF FREQUENCY IN RADIANS AND
S IS THE SAMPLING INTERVAL. Thus FC=0.5 GIVES A CUTO
FREQUENCY AT ONE FOURTH SAMPLING FREQUENCY.

DIMENSION U(3,3,2),V(3,3,2)
COMMON/WORK/A(5,5),B(5,5)

DO 21 K=1,2
DO 21 I=1,3
DO 21 J=1,3
U(I,J,K)=0.0
21 V(I,J,K)=0.0
IF(FC.GE.0.99) FC=0.99
WRITE(LU,14)FC
14 FORMAT(1HO," FC = ",1E10.4," FOR LOWPASS FILTER ",/)
PI=3.141592654
D=1.0E-10
PWR=0.25
IF (N.EQ.3) PWR=0.125
EPS=2.0**PWR-1.0
XP=PI*FC*0.5
T=(SIN(XP)/COS(XP))**2.0
IF(T.GT.D) GO TO 10
ALP=EPS/D
GO TO 11
10 ALP=EPS/T
11 SALP=SQR(ALP)
DEM=1.0-2.0*ALP
IF(DEM.LT.D) GO TO 12
13 Pl=(+2.0*ALP-2.0*SQR(2.0)*SALP+1.0)/DEM
P2=P1
IF(FC.GT.0.3)GO TO 10
P2=(SQRT(T)-EPS)/(SQRT(T)+EPS)
GO TO 20
IF(F.GT.D) GO TO 13
P1=0.0
P2=0.0
V(1,1,1)=1.0
V(1,2,1)=P1+P2
V(2,1,1)=V(1,2,1)
V(1,3,1)=P1*P2
V(3,1,1)=V(1,3,1)
V(2,2,1)=V(1,2,1)**2
V(2,3,1)=P1*P2**2+P2*P1**2
V(3,2,1)=V(2,3,1)
V(3,3,1)=V(1,3,1)**2
SUM=0.0
DO 25 I=1,3
DO 25 J=1,3
25 SUM=SUM+V(I,J,1)
U(1,1,1)=SUM/16.0
U(1,3,1)=U(1,1,1)
U(3,1,1)=U(1,1,1)
U(3,3,1)=U(1,1,1)
U(1,2,1)=SUM/8.0
U(2,1,1)=U(1,2,1)
U(2,3,1)=U(1,2,1)
U(3,2,1)=U(1,2,1)
U(2,2,1)=SUM/4.0
IF(N.EQ.2) GO TO 1
DO 26 I=1,3
DO 26 J=1,3
26 U(I,J,2)=U(I,J,1)
V(1,1,2)=V(1,1,1)
DO 30 I=1,5
DO 30 J=1,5
A(I,J)=0.0
B(I,J)=0.0
DO 31 I=1,5
DO 31 J=1,5
DO 31 K=1,3
DO 31 L=1,3
IK=I-K+1
JL=J-L+1
IF(IK.LE.0.OR.IK.GT.3)GO TO 31
IF(JL.LE.0.OR.JL.GT.3)GO TO 31
A(I,J)=A(I,J)+U(IK,JL,1)*U(K,L,2)
B(I,J)=B(I,J)+V(IK,JL,1)*V(K,L,2)
CONTINUE
RETURN
END
SUBROUTINE TDLPF(A,B,MN,RC,NDIM,LU)

INPUTS
N — NUMBER OF FILTER STAGES
RC — RELATIVE CUTOFF FREQUENCY FOR FILTER
NDIM — ARRAY DIMENSION IN CALLING PROGRAM

OUTPUTS
A — COEFFICIENT ARRAY (NUMERATOR)
B — COEFFICIENT ARRAY (DENOMINATOR)

DIMENSION A(3,3,NDIM),B(3,3,NDIM)

COMPLEX P(10),PK,Q,Z1,Z2

INITIALIZE
N=MN-1
PI=3.141592654
D=1.0E-10

IF(N.GT.NDIM) GO TO 300
IF(0.01.GT.RC.OR.0.99.LT.RC) GO TO 400

AA=0.5*PI*RC
AA=SIN(AA)/COS(AA)
PWR=1.0/FLOAT(N)
EPS=SQR(2.0)-1.0
EPS=1.0
EPS=EPS**PWR
C=AA**2/EPS

FIND ROOTS
L=1
NN=2.0*N
CONST=FLOAT(NN+1)/FLOAT(NN)

DO 10 K=1,NN
THETA=PI*(1.0+2.0*(K-1))*CONST
PK=C*CMPLX(COS(THETA),SIN(THETA))
WRITE(LU,14) PK
14 FORMAT(" PK = ",1E15.5," + J",1E15.5/)
Q=2.0-PK

IF(CABS(Q).LE.D) Q=D
IF(CABS(PK).LE.D) PK=SIGN(D,REAL(PK))
Z1=(2.0+PK+2.0*CSQRT(2.0*PK))/Q
WRITE(LU,12) Z1
12 FORMAT(" Z1 = ",1E15.5," + J",1E15.5/)

IF(CABS(Z1).GE.1.0) GO TO 15
P(L)=Z1
WRITE(LU,11) L,P(L)
11 FORMAT(" P("",1I2,"" ) = "",1E15.5," + J",1E15.5/)

L=L+1
Z2=(2.0+PK-2.0*CSQRT(2.0*PK))/Q
WRITE(LU,13) Z2
13 FORMAT(" Z2 = "",1E15.5," + J",1E15.5/)

IF(CABS(Z2).GE.1.0) GO TO 20
F(L)=Z2
WRITE(LU,11) L,P(L)
L=L+1

20 CONTINUE
PAIR COMPLEX PAIRS OF ROOTS

25 L=1
DO 30 K=1,N
SL=IMAG(P(K))
IF(SL.LT.0.0) GO TO 30
P(L)=P(K)
L=L+1
30 CONTINUE

OBTAIN FILTER COEFFICIENTS
IF((L-1).LT.N) GO TO 500
DO 40 K=1,N
C1=-2.0*REAL(P(K))
C2=ABS(P(K))**2
AM=(1.0+C1+C2)**2/16.0
A(1,1,K)=AM
A(1,2,K)=2.0*AM
A(2,1,K)=2.0*AM
A(1,3,K)=AM
A(3,1,K)=AM
A(2,2,K)=4.0*AM
A(2,3,K)=2.0*AM
A(3,2,K)=2.0*AM
A(3,3,K)=AM
B(1,1,K)=1.0
B(2,1,K)=C1
B(1,2,K)=C1
B(1,3,K)=C2
B(3,1,K)=C2
B(2,2,K)=C1*C2
B(2,3,K)=C1*C2
B(3,2,K)=C1*C2
B(3,3,K)=C2**2
40 B(3,3,K)=C2**2
GO TO 600
WRITE(LU,310)
310 FORMAT(" NUMBER OF STAGES TOO LARGE FOR DIMENSION")
GO TO 600
WRITE(LU,410)
410 FORMAT(" FREQUENCY SPECIFICATION OUT OF RANGE")
GO TO 600
WRITE(LU,510)
510 FORMAT(" NUMBER OF ROOTS LESS THAN EXPECTED")
600 RETURN
END
SUBROUTINE HP48A(J)

INTEGER IDCB(144),BUFF(4),NAME(3)

DATA NAME/2HDA,2HTA,2H1/

CALL OPEN(IDCB,1,ERR,NAM)

IF (ERR .GE. 0) GO TO 30

WRITE(LU,10) IERR

10 JRMAT = "OPEN ERROR",F5.0)

STOP

30 CALL GRAFC(1,LU)

CALL READF(IDCB,1,ERR,BUFF,4,ILOG)

IF (ILOG .EQ. -1) GO TO 55

IF (ERR .GE. 0) GOTO 40

WRITE(LU,31) IERR

31 FORMAT("READ ERROR",F5.0)

GO TO 55

40 CONTINUE

CALL DVCCT(BUFF,BUFF(2),BUFF(3),BUFF(4),LU)

50 GO TO 20

CALL EXEC(13,LU,ISTAT)

ISTAT = IAND(ISTAT,14000000)

IF (ISTAT .NE. 0) GO TO 55

CALL GRAFC(0,LU)

CALL CLOSE(IDCB)

RETURN

SUBROUTINE GRAFC(IFLAG,LU)

INTEGER IESC

IFLAG = 33B

GRAPHIC OFF=0; GRAPHICS ON NOT=0

IF(IFLAG.EQ.0) GO TO 100

GO TO 20

CALL EXEC(13,LU,ISTAT)

ISTAT = IAND(ISTAT,14000000)

IF (ISTAT .NE. 0) GO TO 55

CALL GRAFC(0,LU)

CALL CLOSE(IDCB)

RETURN

IFLAG = 33B

WRITE(LU,10) IESC

10 FORMAT(1R2,"*3C")

WRITE(LU,12) IESC

12 FORMAT(1R2,"*dF")

WRITE(LU,14) IESC

14 FORMAT(1R2,"*dA")
GO TO 200

C GRAPHICS OFF

C

WRITE(LU,30) IESC
FORMAT(1R2,"*d")
WRITE(LU,40) IESC
FORMAT(1R2,"*e")
RETURN

END

SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)

SUBROUTINE DRAWS A LINE BETWEEN THE TWO POINTS (IX1,IY1)
AND (IX2,IY2). THE POINT (IXO,IYO) DEFINES THE
THE ORIGIN.

IXO=0
IYO=0
XSCAL =356.0/1024.0
YSCAL =XSCAL
X1 = IX1*XSCAL + 0.5
X2 = IX2*XSCAL + 0.5
Y1 = IY1*YSCAL + 0.5
Y2 = IY2*YSCAL + 0.5
JX1 = X1 + IX0
JX2 = X2 + IX0
JY1 = Y1 + IYO
JY2 = Y2 + IYO
WRITE(LU,10) JX1,JY1,JX2,JY2
FORMAT("pa",113,1H,,113,1H,,113,1H,,113,"Z")
RETURN
END
END$
PROGRAM BLDIM

THIS PROGRAM BUILDS AN IMAGE FILE FOR THE NCA&T IMAGE DISPLAY SYSTEM. IMAGE FILES MAY BE GENERATED FROM THE GMR-27 DISPLAY, TAPES OR DISC (TYPE 2 FILES).

PROGRAMMER: DLJ

DIMENSION LU(5), IDCBI(272), IDCBO(528), NAME(6), ISIZE(2), IDATA
DIMENSION JNAME(3), IBUF(6)

INTEGER ENTRY(256), TEXT1(40), TEXT2(40), TEXT3(40), RDREC

EQUIVALENCE (ENTRY, NAME), (ENTRY(7), NLINE), (ENTRY(8), MPIXL),
1 (ENTRY(9), IPMIN), (ENTRY(10), IPMAX), (ENTRY(11), ISRC),
2 (ENTRY(13), JNAME), (ENTRY(129), TEXT1), (ENTRY(169), TEXT2),
3 (ENTRY(209), TEXT3), (ENTRY(12), ILOC)

EQUIVALENCE (JNAME(2), JNAME2), (JNAME(3), JNAME3), (ISIZE(2), ISIZ

CONSTANTS

MPIXL = MAXIMUM PIXELS/LINE (WHEN CHANGING BE SURE TO MODIF ARRAY SIZES)

DATA MPIXL/512/

GET INPUT PARAMETERS

CALL RPMAR(LU)
IF (LU .LE. 0) LU = 1

OUTPUT HEADING

WRITE(LU,1)
FORMAT(//"BUILD IMAGE SUBSYSTEM"

OPEN DIRECTORY FILE

CALL OPEN(ID?B1, IERR, 6HIMDIRC, 0, 2HIM, 23, 272)
IF (IERR .LT. 0) GO TO 9999

GET IMAGE NAME

1000 WRITE(LU,2)
FORMAT("ENTER 12 CHARACTER IMAGE NAME?(/E TO EXIT)"")
READ(LU,`` NAME
FORMAT(6A2)
IF (NAME .EQ. 2H/E) GO TO 1060

CHECK FOR DUPLICATE NAME

C
IREC - 0
KREC - 0
CALL RWNDF(IDCBI,IERR)
IF (IERR .LT. 0) GO TO 9999
IREC = IREC + 1
CALL READF(IDCBI,IERR,IBUF,6,LEN)
IF (IERR .LT. 0) GO TO 9999
IF (LEN .EQ. -1) GO TO 1030
C
C COMPARE NAME
C
IF (IBUF .EQ. -1) KREC = IREC
C
DO 1020 I=1,6
IF (NAME(I) .NE. IBUF(I)) GO TO 1010
1020 CONTINUE
C
C DUPLICATE NAME FOUND
C
WRITE(LU,4)
4          FORMAT("ERROR-DUPLICATE NAME")
CALL RWNDF(IDCBI,IERR)
IF (IERR .LT. 0) GO TO 9999
GO TO 1000
C
C EOF REACHED AND NO DUPLICATE FOUND
C
GET IMAGE PARAMETERS
C
WRITE(LU,5)
5          FORMAT("# LINES IN IMAGE?")
READ(LU,*) NLINE
WRITE(LU,6)
6          FORMAT("# PIXELS/LINE?")
READ(LU,*) NPIXL
IF (NPIXL .GT. MPIXL) NPIXL = MPIXL
C
C GET 3-LINES OF DESCRIPTIVE TEXT
C
WRITE(LU,7)
7          FORMAT("ENTER UP TO 3 LINES OF DESCRIPTIVE TEXT")
TEXT1 = 2H
CALL MVW(TEXT1,TEXT1(2),119)
CALL EXEC(1,400B+LU,TEXT1,40)
CALL EXEC(1,400B+LU,TEXT2,40)
CALL EXEC(1,400B+LU,TEXT3,40)
C
C GET SOURCE OF IMAGE
C
WRITE(LU,8)
8          FORMAT("IMAGE SOURCE?(1=DISC FILE;2=TAPE;3=DISPLAY;4=WORK FI
READ(LU,*) ISRC
IF (ISRC .LT. 0) GO TO 1060
IF (ISRC .LT. 1 .OR. ISRC .GT. 4) GO TO 1040
C
0107 C CREATE DATA FILE
0108 C
0109  ISIZE = (FLOAT(NPIXL)*FLOAT(NLINE) + 127.)/ 128.
0110  ISIZ2 = NPIXL
0111  IF (KREC .EQ. 0) KREC = IREC
0112  JNAME = 2HIM
0113  CALL DCDON(KREC,JNAM2,JNAM3)
0114  CALL PURGE(IDCBI,IERR,JNAME,2HIM,23)
0115  CALL CREAT(IDCBI,IERR,JNAME,ISIZE,2HIM,23,528)
0116  IF (IERR .LT. 0) GO TO 9999
0117 C
0118 C INITIALIZE INPUT ROUTINE
0119 C
0120  IERR = RDREC(-LU,ISRC,NLINE,NPIXL)
0121  IF (IERR .LT. 0) GO TO 9999
0122 C
0123 C GET EACH LINE AND WRITE TO FILE
0124 C
0125  IPMAX = 0
0126  IPHIN = 377B
0127  DO 1050 I=1,NLINE
0128  IERR = RDREC(1,IDATA,IPMAX,IPMIN)
0129  IF (IERR .LT. 0) GO TO 9999
0130  CALL WRITF(IDCBI,IERR,IDATA,NPIXL)
0131  IF (IERR .LT. 0) GO TO 9999
0132 C WRITE(LU,1051) IPMAX,IPMIN
0133 1051 FORMAT(2I12)
0134  1050 CONTINUE
0135 C
0136 CALL CLOSE(IDCBI)
0137 C
0138 C WRITE DIRECTORY ENTRY
0139 C
0140  IF (KREC .EQ. IREC) GO TO 1055
0141  CALL OPEN(IDCBI,IERR,6HINDIRC,2,2HIM,23,272)
0142  IF (IERR .LT. 0) GO TO 9999
0143  CALL POSNT(IDCBI,IERR,KREC)
0144  IF (IERR .LT. 0) GO TO 9999
0145  1055 ILOC = 1
0146  CALL WRITF(IDCBI,IERR,ENTRY,256)
0147  IF (IERR .LT. 0) GO TO 9999
0148  GO TO 1000
0149 C
0150 C TERMINATE
0151 C
0152 1060 CALL CLOSE(IDCBI)
0153  CALL EXEC(6)
0154 C
0155 C ERROR
0156 C
0157 9999 WRITE(LU,9)IERR
0158 9 FORMAT(" FILE ERROR-",I6)
0159  CALL CLOSE(IDCBI)
0160 END
INTEGER FUNCTION RDREC(ICODE, IBUF, IP1, IP2)

C THIS SUBROUTINE IS USED TO INPUT IMAGE FROM DISC, TAPE OR DISPLA

DIMENSION IBUF(1), IDATA(1024), NAME(3), RDATA(512), IDCB(1040)

LOGICAL PACKED

EQUIVALENCE (IDATA, RDATA)

IF (ICODE .GT. 0) GO TO 120

INITIALIZATION

NLINE = IP1
NPIXL = IP2
LU = -ICODE

IF (LU .GT. 0) GO TO 90

SPACE FOR CALL WITH NO INTERACTION

INTERACTIVE CALL

90 IF (IBUF .NE. 1) GO TO 100

GET DISC FILE NAME

WRITE(LU,1)
FORMAT("ENTER DISC FILE NAME? ")
READ(LU,2) NAME

OPEN FILE

CALL OPEN(IDCBO, IERR, NAME, 0, 0, 0, 1040)

IF (IERR .LT. 0) GO TO 999

WRITE(LU,3)
FORMAT(" DATA FORMAT (1=UNPACKED; 2=PACKED; 3=REAL)? ")
READ(LU,*) IFMT
PACKED = .TRUE.
IF (IFMT .NE. 2) PACKED = .FALSE.
NUM = NPIXL
IF (PACKED) NUM = (NPIXL+1)/2
IF (IFMT .EQ. 3) NUM = 2*NPIXL
IBCOD = 1
RETURN

TAPE INPUT

WRITE(LU,4)
FORMAT("TAPE LU? ")
READ(LU,*) MTLU
0219  C REWIND TAPE
0220  C
0221  CALL EXEC(3,MTLU+400B)
0222  WRITE(LU,9)
0223  9  FORMAT(" FILE #? ")
0224  READ(LU,*) IFILE
0225  IF (IFILE .LE. 0) CALL EXEC(6)
0226  IF (IFILE .EQ. 1) GO TO 107
0227  DO 105 I=1,IFILE-1
0228  CALL EXEC(3,MTLU+1300B)
0229  105 CONTINUE
0230  C
0231  107 WRITE(LU,3)
0232  READ(LU,*) IFMT
0233  PACKED = .TRUE.
0234  IF (IFMT .NE. 2) PACKED = .FALSE.
0235  NUM = NPIXL
0236  IF (PACKED) NUM = (NPIXL+1)/2
0237  IF (IFMT .EQ. 3) NUM = 2*NPIXL
0238  IBCOD = 2
0239  RETURN
0240  C
0241  110 IF (IBUF .NE. 3) GO TO 115
0242  C
0243  C DISPLAY INPUT
0244  C
0245  WRITE(LU,5)
0246  5  FORMAT("ENTER START LINE,END LINE,START PIXEL,END PIXEL? ")
0247  READ(LU,*) ISTRTL, IENDL, ISTRTP, IENDP
0248  ISTEP = 1
0249  IF (ISTRTL .GT. IENDL) ISTEP = -1
0250  PACKED = .FALSE.
0251  NUM = NPIXL
0252  IBCOD = 3
0253  RETURN
0254  C
0255  C INPUT IS WORK FILE
0256  C
0257  115 CALL OPEN(IDCB,IERR,6HWF0000,0,0,0,1040)
0258  IF (IERR .LT. 0) GO TO 999
0259  PACKED = .FALSE.
0260  NUM = 2*NPIXL
0261  IBCOD = 1
0262  C
0263  C POSITION FILE
0264  C
0265  CALL READF(IDCB,IERR,IDATA,0)
0266  IF (IERR .LT. 0) GO TO 999
0267  C
0268  RETURN
0269  C
0270  C
0271  C DATA INPUT SECTION
0272  C
0273  C BRANCH TO APPROPRIATE SUB SECTION
0274  C
C GO TO (130,140,150),IBCOD
C FILE INPUT
C CALL READF(IDC,B,IERR,RDATA,NUM)
IF (IERR .LT. 0) GO TO 999
IFMT=3
GO TO 160
C TAPE INPUT
C CALL EXEC(1,MFLU,IDATA,NUM)
GO TO 160
C DISPLAY INPUT
C IBUF = 0
CALL MVW(IBUF,IBUF(2),NPIXL-1)
IF ((ISTEP .GT. 0) .AND. (ISTRNL .GT. IENDL)) RETURN
IF (ISTEP .LT. 0 .AND. ISTRNL .LT. IENDL) RETURN
CALL RLINF(ISTRNL,ISTRTP,IENDP,IDATA)
ISTRNL = ISTRNL + ISTEP
C MOVE DATA TO OUTPUT ARRAY AND UNPACK IF NECESSARY
C IF (.NOT. PACKED) GO TO 180
C DATA IN PACKED FORMAT
C DO 170 I=1,NUM
ITEMP = IDATA(I)
CALL ROTB(ITEMP,JTEMP)
JTEMP = IAND(JTEMP,377B)
IF (JTEMP .GT. IP1) IP1 = JTEMP
IF (JTEMP .LT. IP2) IP2 = JTEMP
ITEMP = IAND(ITEMP,377B)
IF (ITEMP .GT. IP1) IP1 = ITEMP
IF (ITEMP .LT. IP2) IP2 = ITEMP
IBUF(2*I-1) = JTEMP
IBUF(2*I) = ITEMP
RETURN
C DATA IS UNPACKED
C DO 190 I=1,NPIXL
ITEMP = IDATA(I)
IF (IFMT .EQ. 3) ITEMP = RDATA(I)
IBUF(I) = ITEMP
RETURN
C RDREC = IERR
END

&LFLTR T=00003 IS ON CR00022 USING 00024 BLKS R=0000

0001  FIN4,L
0002  PROGRAM LFLTR
0003  C
0004  C  WRITTEN BY E. E. SHERROD
0005  C
0006  C  PROGRAM DOES LINEAR FILTERING USING SPATIAL DOMAIN
0007  C  RECURSIVE DIGITAL FILTERS
0008  C
0009  C
0010  C
0011  C
0012  C
0013  DIMENSION A(3,3,2),B(3,3,2),ILU(5),SUM(3,2)
0014  DIMENSION F1(524),F2(524),F3(524)
0015  DIMENSION G1(1),G2(1),G3(1),IX1(3)
0016  DIMENSION X1(524),X2(524),X3(524)
0017  DIMENSION IDCB(144),NAME(3),IRTN(5)
0018  COMMON /IBLK/IBUF(80)
0019  INTEGER READL,RITEL,WFIN
0020  EQUIVALENCE(IBUF(1,A(1,1,1)),(IBUF(41),B(1,1,1))
0021  EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
0022  DATA NAME/2HCO,2HEF,2HFS/
0023  C
0024  C  NROW X 512 IMAGE
0025  C
0026  CALL RMPAR(ILU)
0027  C
0028  LU=ILU(1)
0029  IPIXL=ILU(2)
0030  JPIXL=ILU(3)
0031  C
0032  C  GET FILTER COEFF'S
0033  CALL OPEN(IDCB,IERR,NAME)
0034  IF(IERR .LT. 0) GO TO 9999
0035  CALL READF(IDCB,IERR,IBUF,80,IERR)
0036  IF(IERR .LT. 0) GO TO 9999
0037  NSTAG = ILU(40)
0038  N = NSTAG + 1
0039  CALL CLOSE(IDCB,IERR)
0040  C
0041  C  GET CONTROL BLOCK INFORMATION
0042  C
0043  IERR=WFIN(NROW,ICOLS,RMAX,RMIN,LU)
0044  IF(IERR .LT. 0)GOTO 9999
0045  IPIXL = 2
0046  ICOLS=ICOLS-2
0047  JPIXL =ICOLS - 1
0048  C
0049 C
0050 C INITIALIZE FILTER TO MID LINE-COL AVG
0051 C
0052 C NMID-NROW/2
0053 CNST=0.0
0054 IERR=READL(NMID,0,511,F1)
0055 IF(IERR .LT. 0) GO TO 9999
0056 DO 701 I=1,ICOLS
0057 CNST=CNST+F1(I)
0058 CNST=CNST/FLOAT(ICOLS)
0059 DO 701 I=1,ICOLS
0060 DO 13 I=1,524
0061 F3(I)=CNST
0062 F2(I)=CNST
0063 F1(I)=CNST
0064 CALCULATE FINAL VALUE FOR EACH STAGE
0065 DO 10 NSTG=2,N
0066 SUM(NSTG,1)=0.0
0067 SUM(NSTG,2)=0.0
0068 DO 11 I=1,3
0069 SUM(NSTG,1)=SUM(NSTG,1)+SUII(NSTG,1)+A(I,J,NSTG-1)
0070 SUM(NSTG,2)=SUM(NSTG,2)+B(I,J,NSTG-1)
0071 SUM(NSTG,2)=ABS(SUM(NSTG,2))
0072 IF(SUM(NSTG,2).LT.1.0E-20)CALL EXEC(2,LU,16HFILTER UNSTABLE ,8)
0073 SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)
0074 CALCULATE INITIAL CONDITIONS FOR EACH STAGE
0075 DO 12 NSTG=2,N
0076 SUM(NSTG,2)=SUM(NSTG,1)*SUII(NSTG-1,2)
0077 C INITIALIZE FILTER
0078 DO 14 I=1,524
0079 X3(I)=SUM(2,2)
0080 X2(I)=SUM(2,2)
0081 X1(I)=SUM(2,2)
0082 IF NSTAG .EQ. 1 GO TO 14
0083 G3(I)=SUM(3,2)
0084 G2(I)=SUM(3,2)
0085 G1(I)=SUM(3,2)
0086 CONTINUE
0087 CONTINUE
0088 CONTINUE
0089 CONTINUE
0090 CONTINUE
0091 CONTINUE
0092 CONTINUE
0093 CONTINUE
0094 FILTER REVERSE
0095 DO 14 I=1,524
0096 IERR=READL(8,IPIXL,IPIXL,F3)
0097 IF(IERR .LT. 0) GO TO 9999
0098 IF(IERR .LT. 0) GO TO 9999
0099 IF(IERR .LT. 0) GO TO 9999
0100 IF(IERR .LT. 0) GO TO 9999
0101 IF(IERR .LT. 0) GO TO 9999
0102 IF(IERR .LT. 0) GO TO 9999
0103 IF(IERR .LT. 0) GO TO 9999
0104 IF(IERR .LT. 0) GO TO 9999
0105 IF(IERR .LT. 0) GO TO 9999

79
0106 C
0107 C LNCK = 1
0108 DO 300 NRO=-6,NROW - 1,3
0109 CALL FILTR(2,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0110 IF(LNCK .LT. 7) GO TO 301
0111 LINE = IABS(NRO)
0112 CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,2,LU,RMX,RMI)
0113 301 LNCK =LNCK +1
0114 LINE=IABS(NRO+1)
0115 IF(LINE .GT. NROW-1) GO TO 300
0116 IERR=READL(LINE,IPIXL,JPIXL,F3)
0117 IF(IERR .LT. 0) GO TO 9999
0118 CALL FILTR(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0119 IF(LNCK .LT. 7) GO TO 302
0120 CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,2,LU,RMX,RMI)
0121 302 LNCK =LNCK +1
0122 LINE=IABS(NRO+2)
0123 IF(LINE .GT. NROW-1) GO TO 300
0124 IERR=READL(LINE,IPIXL,JPIXL,F2)
0125 IF(IERR .LT. 0) GO TO 9999
0126 CALL FILTR(2,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0127 IF(LNCK .LT. 7) GO TO 303
0128 IF(LINE .GT. NROW-1) GO TO 300
0129 CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,2,LU,RMX,RMI)
0130 303 LNCK =LNCK +1
0131 LINE=IABS(NRO+3)
0132 IF(LINE .GT. NROW-1) GO TO 300
0133 IERR=READL(LINE,IPIXL,JPIXL,F1)
0134 IF(IERR .LT. 0) GO TO 9999
0135 300 CONTINUE
0136 C
0137 C REINITIALIZE FILTER
0138 C
0139 CONST=(RMX-RMI)/2.
0140 DO 15 II=1,524
0141 F1(II) = CONST
0142 F2(II) = CONST
0143 F3(II) = CONST
0144 15 CONTINUE
0145 C
0146 C FILTER FORWARD
0147 C
0148 RMX=-0.1E38
0149 RMI= 0.1E38
0150 LINE =NROW-9
0151 IERR=READL(LINE,IPIXL,JPIXL,F3(12))
0152 IF(IERR .LT. 0) GO TO 9999
0153 LINE=LINE+1
0154 IERR=READL(LINE,IPIXL,JPIXL,F2(12))
0155 IF(IERR .LT. 0) GO TO 9999
0156 LINE=LINE+1
0157 IERR=READL(LINE,IPIXL,JPIXL,F1(12))
0158 IF(IERR .LT. 0) GO TO 9999
C  

```
0159  LNCK =-6
0160  DO 400 NRO= -6,NROW - 1,3
0161  CALL FILTR(1,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0162  IF(LNCK .LT. 0) GO TO 401
0163  CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,1,LU,LMX,RLM)
0164  400 CONTINUE
0165  LNCK=LNCK+1
0166  LNCK=(NROW-1)-IABS(NRO+1)
0167  IERR=READL(LINE,IPIXL,JPIXL,F3(12))
0168  IF(IERR .LT. 0) GO TO 9999
0169  CALL FILTR(1,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0170  IF(LNCK .LT. 0) GO TO 402
0171  CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,1,LU,LMX,RLM)
0172  LNCK =LNCK +1
0173  LINE=(NROW-1)-IABS(NRO+2)
0174  IF(LINE .LT. 0) GO TO 400
0175  IERR=READL(LINE,IPIXL,JPIXL,F2(12))
0176  IF(IERR .LT. 0) GO TO 9999
0177  CALL FILTR(1,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0178  IF(LNCK .LT. 0) GO TO 403
0179  CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,1,LU,LMX,RLM)
0180  LNCK =LNCK +1
0181  LINE=(NROW-1)-IABS(NRO+3)
0182  IF(LINE .LT. 0) GO TO 400
0183  IERR=READL(LINE,IPIXL,JPIXL,F1(12))
0184  IF(IERR .LT. 0) GO TO 9999
0185  400 CONTINUE
0186  C
0187  51 CONTINUE
0188  RMAX=LMX
0189  RMIN=RLM
0190  CALL CLSWF(NROW,ICOLS,RMAX,RMIN)
0191  CALL PRIN(RTRN)
0192  CALL EXEC(6)
0193  9999 CALL EXEC(2,LU,16HREAD FILE ERROR ,8)
0194  END
0195  SUBROUTINE FILTR(IFLAG,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0196  DIMENSION F1(1),F2(1),F3(1),X1(1),X2(1),X3(1),G1(1),A(1),B(1)
0197  COMMON /IBLK/IBUF(80)
0198  DIMENSION G1(1),G2(1),G3(1)
0199  C
0200  EQUIVALENCE (IBUF,A),(IBUF(41),B)
0201  C  IFLAG = 1 FOR FORWARD FILTERING, = 2 FOR REVERSE
0202  C
0203  C REVERSE FILTERING
0204  C
0205  IF(IFLAG .EQ. 1) GO TO 200
0206  DO 20 I=1,11
0207  L =ICOLS+0.2 - I
0208  J = ICOLS-12 + I
0209  F1(L) = F1(J)
0210  F2(L) = F2(J)
0211  20  F3(L) = F3(J)
0212  C
```

DO 10 M = ICOLS + 9, 1, -1
J = M + 1
K = M + 2
X1(M) = A(1) * F1(:)
1 + A(2) * F1(J) - B(2) * X1(J)
1 + A(3) * F1(K) - B(3) * X1(K)
1 + A(4) * F2(M) - B(4) * X2(M)
1 + A(5) * F2(J) - B(5) * X2(J)
1 + A(6) * F2(K) - B(6) * X2(K)
1 + A(7) * F3(M) - B(7) * X3(M)
1 + A(8) * F3(J) - B(8) * X3(J)
1 + A(9) * F3(K) - B(9) * X3(K)
IF(NSTAG .EQ. 1) GO TO 10
C
G1(M) = A(10) * X1(M)
C
1 + A(11) * X1(J) - B(11) * G1(J)
1 + A(12) * X1(K) - B(12) * G1(K)
1 + A(13) * X2(M) - B(13) * G2(M)
1 + A(14) * X2(J) - B(14) * G2(J)
1 + A(15) * X2(K) - B(15) * G2(K)
1 + A(16) * X3(M) - B(16) * G3(M)
1 + A(17) * X3(J) - B(17) * G3(J)
1 + A(18) * X3(K) - B(18) * G3(K)
10 CONTINUE
GO TO 400
200 CONTINUE
RETURN
END
COMMON BLOCK SUBPROGRAM

BLOCK DATA IBLK COMMON /IBLK/IBUF(80) DATA IBUF/80*0/ END

SUBROUTINE RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,IFLAG,LU,LMX,R

DIMENSION X1(1),G1(1),IX1(524)

INTEGER RITEL

IFL=1 IF(IFLAG .EQ. 1) IFL = 12 IF(NSTAG .EQ. 2) GO TO 100

IERR = RITEL(LINE,IPIXL,JPIXL,X1(IFL)) IF(IFLAG .LT. 0) GO TO 9999

DO 120 I=IFL,JPIXL-IPIXL +IFL IF(X1(I) .GT. RMX) RMX=X1(I)

IF(X1(I) .LT. RMI) RMI=X1(I)

ITEMP = X1(I) + 0.5 IF(IFLAG .LT. 0) ITEMP=0

IF(IFLAG .GT. 377B) ITEMP=377B

IX1(I) = ITEMP

GO TO 200

CONTINUE

IERR = RITEL(LINE,IPIXL,JPIXL,G1(IFL)) IF(IFLAG .LT. 0) GO TO 9999

DO 121 I=1,524

ITEMP=G1(I) + 0.5 IF(IFLAG .LT. 0) ITEMP=0

121 IX1(I) =IAND(ITEMP,777B)

ISTRT=(511-JPIXL)/2

ISTOP=ISTRT+JPIXL

CALL WLINE(LINE,ISTRT,ISTOP,IX1(IFL)) RETURN

9999 CALL EXEC(2,LU,16HWRITE FILE ERROR,8)

END
PROGRAM HFLTR

WRITTEN BY E. E. SHERROD

PROGRAM DOES HOMOMORPHIC FILTERING USING SPATIAL DOMAIN
RECURSIVE DIGITAL FILTERS

COMMON /IBLK/IBUF(80)
DIMENSION IF1(2),IF2(523),R1(523)
DIMENSION A(3,3,2),B(3,3,2),ILU(5),SUM(3,2)
DIMENSION F1(523),F2(523),F3(523)
DIMENSION G1(1),G2(1),G3(1),IX1(3)
DIMENSION X1(523),X2(523),X3(523)
DIMENSION IDCB(144),NAME(3),IRTN(5)
INTEGER :F. A. D L, RITEL, WFINT
EQUIVALENCE (IBUF(1),A(1,1,1)), (IBUF(41),B(1,1,1))
EQUIVALENCE (IRTN(4), RMAX), (IRTN(4), RMIN)
EQUIVALENCE (F1,R1), (F2,IF2), (R1,IF1), (IF1,ILINE), (IF1(2), ICO
DATA NAME/2HCO,2HEF,2HFS/
CALL RMPAR(ILU)
LU=ILU(1)
IF(LU .EQ. 0) LU=1
IPIXL = ILU(2)
IF(IPIXL .EQ. 0) IPIXL = 0
JPIXL = ILU(3)
IF(IPIXL .EQ. 0) JPIXL = 511
C GET FILTER COEFF'S
CALL OPEN(IDCB,IERR,NAME)
CALL READF(IDCB,IERR,IBUF,80,IERR)
CALL CLOSE(IDCB,IERR)
CALL CONTROL BLOCK INFORMATION
IERR=FINF(NROW,ICOLS,RMAX,RMIN,LU)
IF(IERR .LT. 0) GOTO 9999
IPXIL=2
ICOLS=ICOLS-2
JPIXL = ICOLS - 1
CALL INITIALIZE FILTER TO MID LINE-COL AVG
N = NROW/2
CNST=0.0
IERR=READL(NMID,IPXIL,JPIXL,F1)
IF(IERR .LT. 0) GOTO 9999
CALL BIAS(F1, RMIN, ICOLS)
DO 110 I=1,ICOLS
CNST+AMAXO(F1(I),1)
110 CNST=(CNST/FLOAT(ICOLS))
CNST = ALOG(CNST)
DO 9 I=1,523
F1(I) = CNST
F2(I) = CNST
F3(I) = CNST
CONTINUE
C
C CALCULATE FINAL VALUE FOR EACH STAGE
DO 10 NSTG=2,N
SUM(NSTG,1)=0.0
SUM(NSTG,2)=0.0
DO 11 I=1,3
SUM(NSTG,1)=SUM(NSTG,1)+A(I,J,NSTG-1)
11 SUM(NSTG,2)=SUM(NSTG,2)+B(I,J,NSTG-1)
DEL=ABS(SUM(NSTG,2))
IF(DEL.LT.1.0E-10)CALL EXEC(2,LU,16HFILTER UNSTABLE ,8)
SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)
C
C CALCULATE INITIAL CONDITIONS FOR EACH STAGE
SUM(1,2)=CNST
DO 12 NSTG=2,N
12 SUM(NSTG,2)=SUM(NSTG,1)*SUM(NSTG-1,2)
C
C INITIALIZE FILTER
DO 14 I=1,523
X3(I)=(SUM(2,2))
X2(I)=(SUM(2,2))
X1(I)=(SUM(2,2))
IF (NSTAG .EQ. 1) GO TO 14
G3(I)=(SUM(3,2))
G2(I)=(SUM(3,2))
G1(I)=(SUM(3,2))
CONTINUE
RMX=-1.0E38
RMI= 1.0E38
C
FILTER REVERSE
CALL EXEC(2,LU,16HREVERSE FILTERIN,8)
SCL = 1.0
IERR=READL(8,IPIXL,JPIXL,F3)
IF(IERR .LT. 0) GO TO 9999
CALL BIAS(F3,RMIN,ICOLS)
IERR=READL(7,IPIXL,JPIXL,F2)
IF(IERR .LT. 0) GO TO 9999
CALL BIAS(F2,RMIN,ICOLS)
IERR=READL(6,IPIXL,JPIXL,F1)
IF(IERR .LT. 0) GO TO 9999
```fortran
0106  LNCK = 1
0107  DO 300 NRO=-6,NROW - 1,3
0108  CALL BIAS(F1,RMIN,ICOLS)
0109  CALL HFILT(7,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0110  IF(LNCK .LT. 7) GO TO 301
0111  LINE = IABS(NRO)
0112  CALL RITLN(LINE,1PIXL,1PIXL,X1,G1,NSTAG,2,LU,LMX,RMI,SCL)
0113  301  LNCK = LNCK + 1
0114  LINE = IABS(NRO+1)
0115  IF(LINE .GT. NROW-1) GO TO 300
0116  IERR = READL(LINE,1PIXL,1PIXL,F3)
0117  IF(IERR .LT. 0) GO TO 9999
0118  CALL BIAS(F3,RMIN,ICOLS)
0119  CALL HFILT(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0120  IF(LNCK .LT. 7) GO TO 302
0121  CALL RITLN(LINE,1PIXL,1PIXL,X3,G1,NSTAG,2,LU,LMX,RMI,SCL)
0122  302  LNCK = LNCK + 1
0123  LINE = IABS(NRO+2)
0124  IF(LINE .GT. NROW-1) GO TO 300
0125  IERR = READL(LINE,1PIXL,1PIXL,F2)
0126  IF(IERR .LT. 0) GO TO 9999
0127  CALL BIAS(F2,RMIN,ICOLS)
0128  CALL HFILT(2,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0129  IF(LNCK .LT. 7) GO TO 303
0130  IF(LINE .GT. NROW-1) GO TO 300
0131  CALL RITLN(LINE,1PIXL,1PIXL,X2,G1,NSTAG,2,LU,LMX,RMI,SCL)
0132  303  LNCK = LNCK + 1
0133  LINE = IABS(NRO+3)
0134  IF(LINE .GT. NROW-1) GO TO 300
0135  IERR = READL(LINE,1PIXL,1PIXL,F1)
0136  IF(IERR .LT. 0) GO TO 9999
0137  300  CONTINUE
0138  C
0139  C REINITIALIZE FILTER
0140  CONST = (LMX-RMI)/2.
0141  DO 15 J = 1,523
0142     F1(J) = CONST
0143     F2(J) = CONST
0144     F3(J) = CONST
0145  15  CONTINUE
0146  C
0147  C FILTER FORWARD
0148  C
0149  CALL EXEC(2,LU,16HFORWARD FILTERIN,8)
0150  C
0151  C SCALE FOR LN(32766)
0152  SCL = 10.397147 /(LMX)
0153  RMI = 0.1E38
0154  RMX = -0.1E38
0155  JPIXL = JPIXL - 1
0156  LINE = NROW-9
0157  IERR = READL(LINE,1PIXL,1PIXL,F3(12))
0158  IF(IERR .LT. 0) GO TO 9999
0159  LINE = LINE + 1
0160  IERR = READL(LINE,1PIXL,1PIXL,F2(12))
0161  IF(IERR .LT. 0) GO TO 9999
0162  LINE = LINE + 1
0163  IERR = READL(LINE,1PIXL,1PIXL,F1(12))
0164  IF(IERR .LT. 0) GO TO 9999
```
C 0165 LNCK =-6
0166 DO 400 NRO = -6, NROW - 1, 3
0168 CALL HFILT(1, F1, F2, F3, X1, X2, X3, G1, NSTAG, ICOLS)
0169 IF(LNCK .LT. 0) GO TO 401
0170 CALL RITLN(LINE, IPIXL, JPIXL, X1, G1, NSTAG, 1, LU, RMX, RMI, SCL)
0171 401 LNCK = LNCK +1
0172 LINE = (NROW - 1) - IABS(NRO +1)
0173 IERR = READL(LINE, IPIXL, JPIXL, F3(12))
0174 IF(IERR .LT. 0) GO TO 9999
0175 CALL HFILT(1, F3, F1, F2, X3, X1, X2, G1, NSTAG, ICOLS)
0176 IF(LNCK .LT. 0) GO TO 402
0177 CALL RITLN(LINE, IPIXL, JPIXL, X3, G1, NSTAG, 1, LU, RMX, RMI, SCL)
0178 402 LNCK = LNCK +1
0179 LINE = (NROW - 1) - IABS(NRO +2)
0180 IF(LINE .LT. 0) GO TO 400
0181 IERR = READL(LINE, IPIXL, JPIXL, F2(12))
0182 IF(IERR .LT. 0) GO TO 9999
0183 CALL HFILT(1, F2, F3, F1, X2, X3, X1, G1, NSTAG, ICOLS)
0184 IF(LNCK .LT. 0) GO TO 403
0185 CALL RITLN(LINE, IPIXL, JPIXL, X2, G1, NSTAG, 1, LU, RMX, RMI, SCL)
0186 403 LNCK = LNCK +1
0187 LINE = (NROW - 1) - IABS(NRO +3)
0188 IF(LINE .LT. 0) GO TO 400
0189 IERR = READL(LINE, IPIXL, JPIXL, F1(12))
0190 IF(IERR .LT. 0) GO TO 9999
0191 400 CONTINUE
0192 C
0193 51 CONTINUE
0194 CALL EXEC(2, LU, 10HCOMPLETED , 5)
0195 C
0196 CALL CLSWF(NROW, ICOLS, RMX, RMI)
0197 C
0198 RMAX = RMX
0199 RMIN = RMI
0200 CALL PRTN(IRTN)
0201 CALL EXEC(6)
0202 9999 CALL EXEC(2, LU, 16HREAD FILE ERROR , 8)
0203 END
0204 SUBROUTINE HFILT(IFLAG, F1, F2, F3, X1, X2, X3, G1, NSTAG, ICOLS)
0205 DIMENSION F1(1), F2(1), F3(1), X1(1), X2(1), X3(1), A(1), B(1)
0206 COMMON /IBLK/ IBUF(80)
0207 DIMENSION G1(1),G2(1),G3(1)
0208 C
0209 EQUIVALENCE (IBUF,A), (IBUF(41), B)
0210 C IFLAG = 1 FOR FORWARD FILTERING, = 2 FOR REVERSE
0211 C
0212 C REVERSE FILTERING
0213 C
0214 IF(IFLAG .EQ. 1) GO TO 200
0215 DO 200 I=1, 11
0216 L = ICOLS+12 - I
0217 J = ICOLS-12 + I
0218 F1(L) = F1(J)
0219 F2(L) = F2(J)
0220 20 F3(L) = F3(J)
0221 C
DO 10 M = ICOLS+9,1,-1
   J = M + 1
   K = M +2
   X1(M) = A(1) * ALOG(F1(M))
1   + A(2) * ALOG(F1(J)) - B(2) * X1(J)
   + A(3) * ALOG(F1(K)) - B(3) * X1(K)
   + A(4) * ALOG(F2(M)) - B(4) * X2(M)
   + A(5) * ALOG(F2(J)) - B(5) * X2(J)
   + A(6) * ALOG(F2(K)) - B(6) * X2(K)
   + A(7) * ALOG(F3(M)) - B(7) * X3(M)
   + A(8) * ALOG(F3(J)) - B(8) * X3(J)
   + A(9) * ALOG(F3(K)) - B(9) * X3(K)
   IF(NSTAG .EQ. 1) GO TO 10
   G1(M) = A(10) * X1(M)
   + A(11) * X1(J) - B(11) * G1(J)
   + A(12) * X1(K) - B(12) * G1(K)
   + A(13) * X2(M) - B(13) * G2(M)
   + A(14) * X2(J) - B(14) * G2(J)
   + A(15) * X2(K) - B(15) * G2(K)
   + A(16) * X3(M) - B(16) * G3(M)
   + A(17) * X3(J) - B(17) * G3(J)
   + A(18) * X3(K) - B(18) * G3(K)
10   CONTINUE
   GO TO 400
200   CONTINUE
C
C FORWARD FILTERING
C
DO 30 I = 1,11
   L = 12 - I
   J = 12 + I
   F1(L) = F1(J)
   F2(L) = F2(J)
   F3(L) = F3(J)
30
C
DO 40 M = 3,ICOLS+11
   J = M - 1
   K = M + 2
   X1(M) = A(1) * F1(M)
   + A(2) * F1(J) - B(2) * X1(J)
   + A(3) * F1(K) - B(3) * X1(K)
   + A(4) * F2(M) - B(4) * X2(M)
   + A(5) * F2(J) - B(5) * X2(J)
   + A(6) * F2(K) - B(6) * X2(K)
   + A(7) * F3(M) - B(7) * X3(M)
   + A(8) * F3(J) - B(8) * X3(J)
   + A(9) * F3(K) - B(9) * X3(K)
   IF(NSTAG .EQ. 1) GO TO 40
   G1(M) = A(10) * X1(M)
   + A(11) * X1(J) - B(11) * G1(J)
   + A(12) * X1(K) - B(12) * G1(K)
   + A(13) * X2(M) - B(13) * G2(M)
   + A(14) * X2(J) - B(14) * G2(J)
   + A(15) * X2(K) - B(15) * G2(K)
   + A(16) * X3(M) - B(16) * G3(M)
   + A(17) * X3(J) - B(17) * G3(J)
   + A(18) * X3(K) - B(18) * G3(K)
40   CONTINUE
400   CONTINUE
RETURN
END
SUBROUTINE RITLN(LINE, IPIXL, JPIXL, X1, G1, NSTAG, IFLAG, LU, RMX, R1SCL)

DIMENSION X1(1), G1(1), XX1(523)

INTEGER RITEL

C IFLAG = 1 FOR FORWARD = 2 FOR REVERSE

C REV

IF(NSTAG .EQ. 2) GO TO 12
IF(IFLAG .EQ. 1) GO TO 11
DO 10 M = 1, JPIXL - IPIXL + 1
IF(X1(M) .GT. RMX) RMX = X1(M)
IF(X1(M) .LT. RMI) RMI = X1(M)
10 CONTINUE
IERR = RITEL(LINE, IPIXL, JPIXL, X1)
IF(IERR .LT. 0) GO TO 9999
GO TO 12
C
11 CONTINUE
C FORWARD
DO 20 M = 12, JPIXL - IPIXL + 12
X = SCL * (X1(M))
IF(X .GT. 10.397147) X = 10.397177
XX1(M) = EXP(X)
IF(XX1(M) .GT. RMX) RMX = XX1(M)
IF(XX1(M) .LT. RMI) RMI = XX1(M)
20 CONTINUE
IERR = RITEL(LINE, IPIXL, JPIXL, XX1(12))
IF(IERR .LT. 0) GO TO 9999
12 CONTINUE
RETURN
9999 CALL EXEC(2, LU, 16IVING FILE ERROR, 8)
END

SUBROUTINE BIAS(F1, RMIN, ICOLS)

DO 10 I = 1, ICOLS + 11
F1(I) = F1(I) - RMIN + 1.0
IF(F1(I) .LT. 1.) F1(I) = 1.0
10 CONTINUE
RETURN
END

$
&SHOW T=00004 IS ON CR00022 USING 00005 BLKS R=0037

0001 FIN4
0002 PROGRAM SHOW
0003 C
0004 DIMENSION RDATA(512), IDATA(512), LU(5)
0005 C
0006 INTEGER READL
0007 EQUIVALENCE (RDATA, LU(2)), (LU(2), ILINE), (LU(3), PPIXL),
0008 1 (RDATA(2), RMAX), (RDATA(3), RMIN)
0009 C
0010 C GET INPUT PARAMETERS
0011 C
0012 CALL RMPAR(LU)
0013 C
0014 C GET SCALE
0015 C
0016 WRITE(LU, 1)
0017 1 FORMAT("INPUT RANGE?")
0018 READ(LU, *) RL, RH
0019 C
0020 C READ WORK FILE HEADER
0021 C
0022 IERR = READL(-1, 0, 511, RDATA)
0023 IF (IERR .LT. 0) GO TO 999
0024 NLINE = ILINE
0025 NPIXL = PPIXL
0026 PMAX = RMAX
0027 RMIN = RMIN
0028 DO 100 I = 0, NLINE - 1
0029 IF (READL(I, 0, NPIXL - 1, RDATA) .LT. 0) GO TO 999
0030 DO 90 J = 1, NPIXL
0031 IDATA(J) = RL + ((RH - RL) / (PMAX - PMIN)) * (RDATA(J) - PMIN)
0032 IF (IDATA(J) .GT. 255) IDATA(J) = 255
0033 IF (IDATA(J) .LT. 0) IDATA(J) = 0
0034 90 CONTINUE
0035 C
0036 CALL WLINE(I, 0, 511, IDATA)
0037 100 CONTINUE
0038 CALL CLSFW(NLINE, NPIXL, PMAX, PMIN)
0039 CALL EXEC(6)
0040 WRITE(LU, 2) IERR
0041 2 FORMAT("FILE ERROR", I7)
0042 END
0043 $
PROGRAM FIRO

DIMENSION ILU(5), IBUF(80), A(3, 3, 2), H(5, 5), NAME(3), IDC8(144)
DIMENSION NAME1(3), NAME2(3)

EQUIVALENCE (IBUF(1), A(1, 1, 1))

DATA H/25*0. /DATA IBUF/80*0/
DATA NAME/2HCO, 2HEF, 2HFS/
DATA NAME1/2HDP, 2HLA, 2H?
DATA NAME2/2HPL, 2H?

C GET LU
CALL RMPAR(ILU)
LU=ILU
WRITE(LU, 10)
10 FORMAT(" ENTER NUMBER OF STAGES ")
READ(LU, *) NSTG
IBUF(40)=NSTG

WRITE(LU, 11)
11 FORMAT(" ENTER ALPHA VALUE ")
READ(LU, *) ALPHA

H(1, 1)=1.0
DO 100 I=1, 3
DO 100 J=1, 3
CALL WINDO(ALPHA, I, J, WIN)
A(I, J, NSTG)=WIN*H(I, J)
100 CONTINUE

CALL PURGE(IDC8, IERR, NAME, 2HES)
IF(IERR .LT. 0) WRITE(LU, 999) IERR
CALL CREAT(IDC8, IERR, NAME, 2, 3, 2HES)
IF(IERR .LT. 0) WRITE(LU, 999) IERR
CALL WRITF(IDC8, IERR, IBUF, 80)
CALL CLOSE(IDC8, IERR)

C SCHEDULE DISPLAY
CALL EXEC(23, NAME1, LU, NSTG, 0, 0, 0, IBUF, 80)

WRITE(LU, 40)
40 FORMAT(" ENTER DISPLAY DEVICE "/ 1. TV"/ 2. HP2648A")
READ(LU, *) IDEV
IF(IDEV .EQ. 2) GO TO 41
CALL EXEC(23, NAME2)
GO TO 42
41 CONTINUE
CALL HP48A(LU)
42 CONTINUE
999 FORMAT(" FILE ERROR ")
STOP
END
SUBROUTINE HP48A(LU)
DIMENSION IB(14), IA(4)
INTEGER IDC(144), BUF(4), NAME(3)
DATA NAME/2HDA,2HTA,2HI/ 
CALL OPEN(IDC, IERR, NAME)
IF (IERR .GE. 0) GO TO 30
WRITE(LU,10) IERR
10 FORMAT("OPEN ERROR",F5.0)
STOP
30 CALL GRAFC(1,LU)
CALL READ1(IDC, IERR, BUF, 4, ILOG)
IF(ILOG .EQ. -1) GO TO 55
IF (IERR .GE. 0) GOTO 40
WRITE(LU,31) IERR
31 FORMAT("READ ERROR",F5.0)
GO TO 55
40 CONTINUE
CALL DVECT(BUF, BUF(2), BUF(3), BUF(4), LU)
GO TO 20
55 CALL EXEC(13, LU, ISTAT)
ISTAT = IAND(ISTAT, 140000B)
IF(ISTAT .NE. 0) GO TO 55
CALL GRAFC(0, LU)
CALL CLOSE(IDC)
RETURN
END
SUBROUTINE GRAFC(IFLAG, LU)
INTEGER IESC
IESC = 33B
CALL GRAFC(IFLAG, LU)
IF(IFLAG.EQ.0) GO TO 100
WRITE(LU,30) IESC
30 FORMAT("GRAPHIC OFF",F5.0)
GO TO 200
100 WRITE(LU,10) IESC
10 FORMAT("GRAPHIC ON",F5.0)
GO TO 200
WRITE(LU,30) IESC
30 FORMAT("GRAPHIC OFF",F5.0)
GO TO 200
SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)

C SUBROUTINE DRAWS A LINE BETWEEN THE TWO POINTS (IX1,IY1)
C AND (IX2,IY2). THE POINT (IX0,IYO) DEFINES THE
C THE ORIGIN.

IXO = 0
IYO = 0
XSCAL = 356.0/1024.0
YSCAL = XSCAL
X1 = IX1*XSCAL + 0.5
X2 = IX2*XSCAL + 0.5
Y1 = IY1*YSCAL + 0.5
Y2 = IY2*YSCAL + 0.5
JX1 = X1 + IXO
JX2 = X2 + IXO
JY1 = Y1 + IYO
JY2 = Y2 + IYO
WRITE(LU,10) JX1,JY1,JX2,JY2
10 FORMAT("pa",1I3,1H,,1I3,1H,,1I3,1H,,1I3,"Z")
RETURN
END

93
SUBROUTINE WINDO(ALPHA,N,M,WIN)
    XN=SQRT(N**2 + M**2)
    BETA=ALPHA*SQRT(1.-XN)
    CALL BESIO(ALPHA,BIAA)
    CALL BESIO(BETA,BIBB)
    BETAI=ALPHA*SQRT(2)
    CALL BESIO(BETAI,BIB)
    ZHIN=BIB/BIAA
    WIN=(BIB/BIAA-ZMIN)/(1.0-ZMIN)
    RETURN
END

SUBROUTINE BESIO(X,RIO)
    RIO=ABS(X)
    IF(RIO-3.75) 1,1,2
    1 Z=X*X*7.111111E-2
    RIO=((((4.5813E-3*Z+3.60768E-2)*Z+2.659732E-1)*Z+1.206749E0
    1089942E0)*Z+3.515623E0)*Z+1.
    RETURN
    2 Z=3.75/RIO
    RIO=EXP(RIO)/SQRT(RIO)*((((((3.92377E-3*Z-1.647633E-2)*Z+2
    17E-2)*Z-2.057706E-2)*Z+9.16281E-3)*Z-1.57565E-3)*Z+2.25319E-
    012 2+1.328592E-2)*Z+3.989423E-1)
    RETURN
END
SUBROUTINE BESJ

PURPOSE

   COMPUTE THE J BESSEL FUNCTION FOR A GIVEN ARGUMENT AND

USAGE

   CALL BESJ(X,N,BJ,D,IER)

DESCRIPTION OF PARAMETERS

   X    -THE ARGUMENT OF THE J BESSEL FUNCTION DESIRED
   N    -THE ORDER OF THE J BESSEL FUNCTION DESIRED
   BJ   -THE RESULTANT J BESSEL FUNCTION
   D    -REQUIRED ACCURACY
   IER  -RESULTANT ERROR CODE WHERE,
         IER=0  NO ERROR
         IER=1  N IS NEGATIVE
         IER=2  X IS NEGATIVE OR ZERO
         IER=3  REQUIRED ACCURACY NOT OBTAINED
         IER=4  RANGE OF N COMPARED TO X NOT CORRECT (SEE R

REMARKS

   N MUST BE GREATER THAN OR EQUAL TO ZERO, BUT IT MUST BE
   LESS THAN
   20+10*X-X** 2/3  FOR X LESS THAN OR EQUAL TO 15
   90+X/2           FOR X GREATER THAN 15

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED

NONE

METHOD

   RECURRENCE RELATION TECHNIQUE DESCRIBED BY H. GOLDSTEIN
   R.M. THALER,'RECURRENCE TECHNIQUES FOR THE CALCULATION
   BESSEL FUNCTIONS', M.T.A.C., V.13, PP.102-108 AND I.A. ST
   AND M. ABRAMOWITZ,'GENERATION OF BESSEL FUNCTIONS ON H
   SPEED COMPUTERS', M.T.A.C., V.11, 1957, PP.255-257

SUBROUTINE BESJ(X,N,BJ,D,IER)

BJ=.0
IF(N)10,20,20
   10 IER=1
   RETURN
20 IF(X)30,31,32
   30 IER=2
   RETURN
   32 NTEST=20.+10.*X-(X***(2/3))
   GO TO 36
34 NTEST=90.+X/2.
36 IF(N-NTEST)40,38,38
   38 IER=4
   RETURN
IER=0
M1=N+1
BPREV=0.

COMPUTE STARTING VALUE OF M

IF(X-5.)50,60,60
MA=X+6.
GO TO 70
MA=1.4*X+60./X
MB=N+IFIX(X)/4+2
NZERO=MAXO(MA,MB)

SET UPPER LIMIT OF M

MMAX=NTEST
DO 190 M=NZERO,MMAX,3
SET F(M),F(M-1)

FA=1.0E-28
FM=.0
ALPHA=0.
IF(M-(M/2)*2)120,110,120
JT=-1.
GO TO 130
JT=1.
M2=M-2
DO 160 K=1,M2
IK=M-K
BMK=2.*FLOAT(MK)*FM1/X-FM
FM=FM1
FM1=BMK
IF(MK-N-1.)150,140,150
BJ=BMK
JT=-JT
S=1+JT
ALPHA=ALPHA+BMK*S
BMK=2.*FM1/X-FM
IF(N)180,170,180
BJ=BMK
ALPHA=ALPHA+BMK
BJ=BJ/ALPHA
IF(ABS(BJ-BPREV)-ABS(D*BJ))200,200,190
BPREV=BJ
IER=3
RETURN
END
$
&BLDF W T=00004 IS ON CR00022 USING 00022 BLKS R=4113

0001 FTN4
0002 PROGRAM BLDF
0003 C
0004 C THIS PROGRAM IS USED IN CONJUNCTION WITH IMAGE PROCESSING
0005 C IT CREATES AND MAINTAINS AN IMAGE WORK FILE WITH PIXEL VALUES
0006 C STORED AS REAL NUMBERS TO PRESERVE PRECISION.
0007 C
0008 C
0009 C
0010 C
0011 C
0012 DIMENSION IDC1(272),IDCB2(1040),IDCB3(528),IMAGE(6),LU(5)
0013 DIMENSION IRTN(5),JNAME(3),IBUF(15),RDATA(512),IDATA(512),
0014 1 ISIZE(2)
0015 C
0016 EQUIVALENCE (ILINE,IRTN(4)),(IPXL,IRTN(5)),(ILINE,RDATA),
0017 1(RDATA(2),AMAX),(RDATA(3),RMIN),(IBUF(12),ILOC),(IBUF(13),JN
0018 EQUIVALENCE (IBUF(7),NILINE),(IBUF(8),NPXL),(IBUF(9),IPMIN),
0019 1 (IBUF(10),IPMAX),(ISIZE(2),ISIZ2)
0020 C
0021 C
0022 C
0023 C GET INPUT PARAMETERS
0024 C
0025 CALL RMPAR(LU)
0026 C
0027 IF (LU .LE. 0) LU = 1
0028 C
0029 C REUSE WORK FILE
0030 C
0031 WRITE(LU,7)
0032 7 FORMAT(//'DO YOU WANT TO REUSE THE CURRENT WORK FILE? Y OR
0033 READ(LU,2) IANS
0034 IF(IANS .EQ. 1HY )GO TO 200
0035 C GET IMAGE NAME FROM USER
0036 C
0037 WRITE(LU,1)
0038 1 FORMAT("ENTER IMAGE NAME (12 CHARACTER)? ")
0039 READ(LU,2) IMAGE
0040 2 FORMAT(6A2)
0041 C
0042 C CHECK IF WORK FILE WANTED
0043 C
0044 IF (ICMPW(IMAGE,12H ,6) .EQ. 0) GO TO 140
0045 C
0046 C OPEN DIRECTORY FILE:
0047 C
0048 CALL OPEN(IDC1,L,1,2HIMDIRC,6HIM,23,272)
0049 IF (LERR .LT. 0) GO TO 9999
0050 C
C FIND IMAGE

C

CALL READF(IDCBI,IERR,IBUF,15,LEN)

IF (IERR .LT. 0) GO TO 9999

IF (LEN .EQ. -1) GO TO 9990

IF (ICMPW(IMAGE,IBUF,6) .NE. 0) GO TO 100

C IMAGE FOUND

IF (ILOC .EQ. 1) GO TO 9980

C IMAGE IS ON DISC

C CREATE WORK FILE

CALL OPEN(IDCBI2,IERR,6HWF0000)

IF (IERR .EQ. -6) GO TO 110

IF (IERR .LT. 0) GO TO 9999

C ASK IF USER WANTS TO SAVE WORK FILE

WRITE(LU,6)

FORMAT("DO YOU WANT TO SAVE IMAGE IN CURRENT WORK FILE?")

READ(LU,2) IANS

IF (IANS .EQ. 2HNO) GO TO 110

C SCHEDULE BUILD IMAGE PROGRAM

CALL CLOSE(IDCBI2)

CALL EXEC(23,6HBLDIM ,LU)

CALL PURGE(IDCBI2,IERR,6HWFO0000)

IF (NPIXL .LT. 3) NPIXL = 3

ISIZE = (2.0*FLOAT(NLINE+1)*FLOAT(NPIXL)+127.)/128.

ISIZ2 = 2*NPIXL

CALL CREAT(IDCBI2,IERR,6HWFO000,ISIZE,2,0,0,1040)

IF (IERR .LT. 0) GO TO 9999

C OPEN IMAGE DATA FILE

CALL OPEN(IDCBI3,IERR,JNAME,1,2HIM,23,528)

IF (IERR .LT. 0) GO TO 9999

C COPY DATA AND CONVERT TO REAL

C POSITION TO RECORD # 2

CALL WRITF(IDCBI2,IERR,RDATA,1)

DO 120 I=1,NLINE

CALL READF(IDCBI3,IERR,IDATA,512,LEN)

IF (IERR .LT. 0) GO TO 9999

DO 115 J=1,NPIXL

RDATA(J) = IDATA(J)

115

120

CALL WRITF(IDCBI2,IERR,RDATA)
IF (IERR .LT. 7) GO TO 9999

CONTINUE

RPMAX = IPMAX
RPMIN = IPMIN

CLOSE ALL IMAGE FILES

CALL CLOSE(IDCB1)
CALL CLOSE(IDCB3)

WRITE INFO IN WORK FILE RECORD 1

ILINE = NLINE
IPIXL = NPIXL
RMAX = RPMAX
RMIN = RPMIN
CALL WRITF(IDCB2,IERR,RDATA,6,1)
IF (IERR .LT. 0) GO TO 9999

CALL CLOSE(IDCB2)

IRTN = 0
CALL PRTN(IRTN)
CALL EXEC(6)

ERRORS

IMAGE NOT ON DISC
WRITE(LU,4)
FORMAT(" IMAGE NOT ON DISC!")
IRTN = -100
GO TO 200

IMAGE NOT FOUND
WRITE(LU,3)
FORMAT(" IMAGE NOT FOUND!")
IRTN = -101
GO TO 200

FILE ERROR
WRITE(LU,5) IERR
FORMAT("FILE ERROR =",16)

IF(IERR.EQ.-8) CALL CLOSE(IDCB1,IERR)
IRTN = -103
GO TO 200
END

$
INTEGER FUNCTION SCROL(IDCB, IDIRC, NLINE, IFRST, ILAST, RMAX, RMIN)

DIMENSION IDCB(144), IDATA(512)

DATA IUP, IDOWN / 34060B, 34040B/

C CHECK IF NO WORK NECESSARY
IF (IDIRC .EQ. 0) RETURN

IF (IDIRC .GT. 0) GO TO 200

DO 100 I = -1, IDIRC - 1
IF (IFRST .LE. 0) RETURN
CALL READF(IDCB, SCROL, IDATA, 512, LEN, IFRST)

DO 110 J = 1, LEN
IDATA(J) = (255. / (RMAX - RMIN)) * (IDATA(J) - RMIN)
IF (IDATA(J) .LT. 0) IDATA(J) = 0
IF (IDATA(J) .GT. 255) IDATA(J) = 255
110 CONTINUE

IF (SCROL .LT. 0) RETURN
CALL DRIVR(2, IUP, 1)
CALL WLINE(0, 0, LEN - 1, IDATA)
IFRST = IFRST + 1

100 ILAST = ILAST - 1
RETURN

C SCROLL IMAGE DOWN
DO 210 I = 1, IDIRC
IF (ILAST .GE. NLINE - 1) RETURN
CALL READF(IDCB, SCROL, IDATA, 512, LEN, ILAST + 1)

DO 220 J = 1, LEN
IDATA(J) = (255. / (RMAX - RMIN)) * (IDATA(J) - RMIN)
IF (IDATA(J) .LT. 0) IDATA(J) = 0
IF (IDATA(J) .GT. 255) IDATA(J) = 255
220 CONTINUE

IF (SCROL .LT. 0) RETURN
CALL DRIVR(2, IDOWN, 1)
CALL WLINE(255, 0, LEN - 1, IDATA)
ILAST = ILAST + 1
IFRST = IFRST - 1

210 RETURN

END
SUBROUTINE WLINE(LINE,IPIX,JPIX,IDATA)
C
C THIS SUBROUTINE WRITES A DESIGNATED LINE TO THE GMR-27
C
LINE = LINE NUMBER
IPIX = STARTING PIXEL
JPIX = ENDING PIXEL
IDATA = BUFFER CONTAINING IMAGE DATA FOR LINE

DIMENSION IDATA(512),INIT(6)
EQUIVALENCE (LLA,INIT(2)),(LEA,INIT(3)),(LEB,INIT(4))
DATA INIT/100377B,64000B,44000B,50000B,24041B,26002B/

IDIRC = 1
IF (IPIX .GT. JPIX) IDIRC = -1

LLA = 64000B + IAND(LINE,377B)
LEA = 44000B + IAND(IPIX,777B)
LEB = 50000B + IDIRC + 512
CALL DRIVR(2,INIT,6)

NUM = IDIRC*(JPIX-IPIX)+1
CALL DRIVR(2,IDATA,NUM)

RETURN
END
&DRVR T-00004 IS ON CR00022 USING 00012 BLKS R-0241

0001  ASMB,R,L,C
0002  NAN DRIVR,6
0003  ENT DRIVR
0004  EXT .ENTR,$LIBR,$LIBX
0005  *
0006  *
0007  OPCODE BSS 1
0008  BUFR BSS 1
0009  LEN  BSS 1
0010  *
0011  DRIVR NOP           ENTRY
0012  JSB  .ENTR          GET
0013  DEF OPCODE          PARAMETERS.
0014  LDA LEN,I           GET @ WORDS
0015  CMA,INA            NEGATE
0016  STA CNT             & SAVE.
0017  SSA,RSS            IF NOT NEGATIVE
0018  JMP EXIT           EXIT
0019  *
0020  JSB  $LIBR          TURN OFF
0021  NOP                INTERRUPTS.
0022  LDA OPCODE,I        CHECK REQUEST
0023  SLA,ELA            IF READ
0024  JMP  D.2           GO PROCESS
0025  *
0026  * WRITE REQUEST
0027  *
0028  SSA,RSS            IF DMA NOT REQUIRED
0029  JMP  D.1           GO DO PROGRAMMED I O
0030  *
0031  * DMA OUTPUT
0032  *
0033  LDA CW1            GET CONTROL WORD 1
0034  OTA DMA2        USE CHANNEL 2
0035  CLC 3B            PREPARE TO SEND ADDRESS
0036  LDA BUFR
0037  OTA 3B
0038  STC 3B            PREPARE TO SEND COUNT
0039  LDA CNT
0040  OTA 3B
0041  LDA BUFR,I
0042  OTA SC
0043  STC SC,C        START DEVICE
0044  STC DMA2,C      START DMA
0045  SFS DMA2
0046  JMP  *-1
0047  CLF DMA2
0048  JMP  EXIT+1
0049  *
0050  *
0051  D.1  LDA  BUFR,I  GET DATA WORD
0052       OTA SC OUTPUT IT.
0053       STC SC,C TURN ON DEVICE
0054       SFS SC WAIT 'TIL
0055       JMP *-1 DONE
0056       ISZ BUFR BUMP BUFFER ADDRESS
0057       ISZ CNT LAST WORD?
0058       JMP D.1 NO GO BACK.
0059       JMP EXIT GO EXIT
0060 * READ ENTRY
0061 * READ ENTRY
0062 *
0063  D.2  SSA SKI IF SPECIAL
0064       JMP D.3 MODE
0065       LDA SPD8 SET UP
0066       OTA SC
0067       STC SC,C FOR
0068       SFS SC
0069       JMP *-1 READ.
0070  D.3  LDA RDPD GET READ DATA CODE
0071       OTA SC
0072       STC SC,C START DEVICE
0073       SFS SC WAIT 'TIL
0074       JMP *-1
0075  D.4  LDA RDPD
0076       OTA SC
0077       STC SC,C
0078       SFS SC
0079       JMP *-1
0080       LIA SC DONE. GET WORD.
0081       STA BUFR,I STUFF IN BUFFER
0082       ISZ BUFR BUMP BUFFER
0083       ISZ CNT DONE?
0084       JMP D.4 NO GO BACK.
0085 *
0086  EXIT CLC SC TURN OFF DEVICE
0087       JSB $LIBX RESTORE RTE AND
0088       DEF DRIVR RETURN
0089 *
0090 *
0091 *
0092  A EQU 0
0093 *
0094  SC EQU 22B
0095  RDPD OCT 160000
0096  SPD8 OCT 120400
0097  CNT BSS 1
0098  CW1 OCT 120022 * HAVE TO CHANGE WITH SELECT CODE
0099  DMA2 EQU 7
0100  END
SUBROUTINE ROTAE(U,V,WN,LU)
COMPLEX P(10),Q(10),QQ,PP
DIMENSION U(3,3,2),V(3,3,2)
COMMON/WORK/AMAG(10),A(3,3),B(3,3)
WRITE(LU,100)
100 FORMAT(" SELECT FILTER ", 1. BUTTERWORTH ", 2. CHEBYSHEV ", 3. LINEAR PHASE ")
READ(LU,*) ITYPE
WRITE(LU,110)
110 FORMAT(" ENTER THE NUMBER OF FILTER STAGES ")
READ(LU,*) NSTG
WRITE(LU,120)
120 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS ")
READ(LU,*) WR
WRITE(LU,140)
140 FORMAT(" ARE ALL ZEROS LOCATED AT INFINITY ", 1 = YES ", 2 = NO ")
READ(LU,*) IFLAG
WRITE(LU,151)
151 FORMAT(" ENTER RIPPLE FACTOR ")
READ(LU,*) ELP
IF(ITYPE.EQ.1) CALL BUTTER
IF(ITYPE.EQ.2) CALL CHEB1(NSTG,WR,P,AMAG,ELP)
IF(ITYPE.EQ.3) CALL LINEAR PHASE
20 DO 10 J=1,NSTG
30 WRITE(LU,130) J
130 FORMAT(" ENTER ROTATION ANGLE IN NEG. DEGREES FOR STAGE ")
READ(LU,*) THETA
PMAG = AMAG(J)
Q(J) = CMPLX(-1.,0.)
QQ = Q(J)
PP = P(J)
CALL SROTT(A,B,PMAG,PP,QQ,IFLAG,THETA)
DO 1111 I=1,3
1111 U(I,K,J) = A(I,K)
DO 1111 K=1,3
V(I,K,J) = B(I,K)
WRITE(LU,40) P(J),AMAG(J)
40 FORMAT(1X,1(" P="1E15.5," +J",1E15.5,/)
10 CONTINUE
MN = NSTG + 1
WRITE(1,1112) U
WRITE(1,1112) V
1112 FORMAT(3E15.4)
RETURN
SUBROUTINE CHEB1(N, WR, P, AMAG, ELP)
DIMENSION AMAG(N)
COMPLEX P(N), PN
PI=3.1415927
E=1.0/ELP
SINHIV=ALOG(E+SQRT(E**2+1.0))
ALP=(-1.0*SINHIV)/FLOAT(N)
IF(WR.EQ.1.0) GO TO 30
X=0.5*WR*PI
IF(COS(X).EQ.0.0) GOTO 30
XTAN=SIN(X)/COS(X)
KK=1
NTWO=4*N
XX=1.0/FLOAT(NTWO)
DO 20 I=1,NTWO
GAMMA=(2*I-1)*PI*XX
C1=(EXP(ALP)-EXP(-ALP))/2.
C2=SIN(GAMMA)
C3=(EXP(ALP)+EXP(-ALP))/2.
C4=COS(GAMMA)
XR=C1*C2
XI=C3*C4
PN=XTAN*CMPLX(XR, XI)
IF(REAL(PN).GT.0.0) GO TO 20
IF(AIMAG(PN).LT.0.0) GO TO 20
P(KK)=PN
AMAG(KK)=CABS(PN)**2
20 KK=KK+1
GO TO 34
30 WRITE(LU,33)
33 FORMAT(" CUTOFF FREQ. CAN NOT = 1.0 ")
RETURN
END
SUBROUTINE SROTT(A,B,PMAG,PP,QQ,IFLAG,THETA)

DIMENSION A(3,3),B(3,3)

COMPLEX PP,QQ

ADJ=0.999

X=THETA*0.0174533

C1=COS(X)**2

C2=-7.0*COS(X)*SIN(X)

C3=SIN(X)**2

C7=-2.0*REAL(PP)*COS(X)

C8=2.0*REAL(PP)*SIN(X)

C9=CABS(PP)**2

B(1,1)=C1+C2+C3+C7+C8+C9

B(1,2)=2.0*(C1-C3+C7+C9)*ADJ

B(1,3)=(C1-C2+C3+C7-C8+C9)*ADJ**2

B(2,1)=2.0*(C3-C1+C8+C9)*ADJ

B(2,2)=4.0*(C9-C1-C3)*ADJ**2

B(2,3)=2.0*(C3-C1-C8+C9)*ADJ**3

B(3,1)=(C1-C2+C3-C7+C8+C9)*ADJ**2

B(3,2)=2.0*(C1-C3-C7+C9)*ADJ**3

B(3,3)=(C1+C2+C3-C7-C8+C9)*ADJ**4

IF(IFLAG.EQ.1) GO TO 10

L4=-2.0*REAL(QQ)*COS(X)

C5=2.0*REAL(QQ)*SIN(X)

C6=CABS(QQ)**2

A(1,1)=C1+C2+C3+C4+C5+C6

A(1,2)=2.0*(C1-C3+C4+C6)

A(1,3)=(C1-C2+C3+C4-C5+C6)

A(2,1)=2.0*(C3-C1+C5+C6)

A(2,2)=4.0*(C6-C1-C3)

A(2,3)=2.0*(C3-C1-C5+C6)

A(3,1)=(C1-C2+C3-C4+C5+C6)

A(3,2)=2.0*(C1-C3-C4+C5+C6)

A(3,3)=(C1+C2+C3-C4-C5+C6)

GO TO 20

A(1,1)=1.0

A(1,2)=2.0

A(1,3)=1.0

A(2,1)=2.0

A(2,2)=4.0

A(2,3)=2.0

A(3,1)=1.0

A(3,2)=2.0

A(3,3)=1.0

CONTINUE

SCAL = 1./B(1,1)

DO 30 I=1,3

DO 30 K=1,3

B(I,K)=(B(I,K)*SCAL)

A(I,K)=(A(I,K)*SCAL*PMAG)

CONTINUE

RETURN

END
&STABI T=00004 IS ON CR00022 USING 00070 BLKS R=0668

0001 FtN4,L
0002 PROGRAM START
0003 C
0004 C THIS PROGRAM EVALUATES THE FILTER STABILITY CHARACTERISTICS
0005 C
0006 C
0007 COMMON/WORK/WO(130)
0008 C INTEGER BUFF
0009 DIMENSION IBUF(80),ILU(5),IRTN(5)
0010 DIMENSION V(3,3,2),U(3,3,2)
0011 EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
0012 C
0013 CALL RMPAR(ILU)
0014 LU=ILU(1)
0015 MN=ILU(2) + 1
0016 C
0017 C GET FILTER COEFF’S
0018 CALL EXEC(14,1,IBUF,80)
0019 C
0020 C
0021 C CALL STABT(V,MN,IRTCD,LU)
0022 C IRTN = IRTCD
0023 C
0024 C CALL PRIN(IRTN)
0025 END
0026 C SUBROUTINE STABT(V,MN,IRTCD,LU)
0027 C SUBROUTINE CHECKS STABILITY OF SYSTEM EQUATION- Y(M,N)=A*Y(M-1,N)+B*Y(M,N-1)
0028 C ---COEFFICIENT MATRIX OF DENOMINATOR OF ZW-TRANSFORM OF SYS
0029 C IMPULSE FUNCTION
0030 C
0031 C LOGICAL ISTAB
0032 C DIMENSION V(3,3,2)
0033 C DIMENSION C(5,5),A(25,25),B(25,25),S(25,25),EVR(25),EVI(25)
0034 C COMMON/WORK/IERR(25)
0035 C HDIM=25
0036 C N=2*(MN-1)+1
0037 C M=N**2
0038 C IF(MN.EQ.3) GO TO 5
0039 C PUT COEFFICIENTS IN STABILITY ARRAY
0040 C
0041 DO 6 I=1,3
0042 DO 6 J=1,3
0043 C(I,J)=V(I,J,1)
0044 GO TO 13
0045 5 DO 10 I=1,5
0046 DO 10 J=1,5
0047 6 C(I,J)=V(I,J,1)
0048 GO TO 13
0049 5 DO 10 I=1,5
0050 DO 10 J=1,5
0051 DO 10 K=1,3
0052 DO 10 L=1,3
0053 IK=I-K+1
0054 JL=J-L+1
0055 IF((IK .LE. 0) .OR. (IK .GT. 3)) GO TO 10
0056 IF((JL .LE. 0) .OR. (JL .GT. 3)) GO TO 10
0057 C(I,J)=C(I,J)+V(IK,JL,1)*V(K,L,2)
0058 10 CONTINUE
DO 13 I=1,N
13 CONTINUE

WRITE(LU,11)
11 FORMAT(20HO COEFFICIENT MATRIX,/)  

DO 21 I=1,N
21 WRITE(LU,12) (C(I,J),J=1,N),N
12 FORMAT(1H ,5F15.6)

C FORM A AND B MATRICES

DO 22 I=1,M
    DO 22 J=1,M
        A(I,J)=0.0
        B(I,J)=0.0

22 S(I,J)=0.0

NOW=N-1
    DO 23 J=1,N
        DO 23 I=1,NOW
            K=I+(J-1)*N
            IF(J.EQ.1) GO TO 24
            A(1,K)=-C(I+1,J)
            IF(J.GT.1) A(1,K)--0.5*C(I+1,J)
            IF(J.LT.1) A(K+1,K)=0.5
            IF(J.EQ.1) A(K+1,K)=1.0
        23 CONTINUE
        DO 25 J=1,NOW
            DO 25 I=1,N
                K=I+(J-1)*N
                IF(I.EQ.1) GO TO 26
                B(1,K)=-C(I,J+1)
                IF(I.GT.1) B(1,K)--0.5*C(I,J+1)
                IF(I.LT.1) B(KN,K)=0.5
                IF(I.EQ.1) B(KN,K)=1.0
        25 CONTINUE
        23 CONTINUE

DO 25 J=1,NOW
     DO 25 I=1,N
     25 S(I,J)=0.0

DO 28 I=1,M
    DO 23 J=1,M
        S(I,J)=B(I,J)

FIND EIGENVALUES OF A AND B

DO 27 I=1,M
    DO 27 J=1,M
        27 S(I,J)=A(I,J)
    CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
    WRITE(LU,71)
    71 FORMAT(//,1O4,19H EIGEN VALUES OF (A))

TEST=1.0
IONE=0

CALL PNTEV(EVR,EVI,MDIM,S,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) GO TO 405
400 FORMAT(" FILTER IS UNSTABLE!")
401 FORMAT(" FILTER IS STABLE")

DO 94 I=1,M
    DO 94 J=1,M
        94 S(I,J)=0.0
    DO 94 28 I=1,M
    DO 94 J=1,M
        28 S(I,J)=B(I,J)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,72)
72 FORMAT(/,10X,19HEIGEN VALUES OF (B))
CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) GOTO 405
C
FIND EIGENVALUES OF A+B
C
DO 29 I=1,M
DO 29 J=1,M
29 S(I,J)=A(I,J)+B(I,J)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,73)
73 FORMAT(/,10X,21HEIGEN VALUES OF (A+B))
CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) WRITE(LU,400)
IF(ISTAB) GO TO 404
WRITE(LU,401)
404 ICNTD = 0
GO TO 500
C
FIND EIGENVALUES OF A*S
C
DO 30 I=1,M
DO 30 J=1,M
30 S(I,J)=0.0
DO 31 I=1,N
DO 31 J=1,N
K=J+(I-1)*N
L=I+(J-1)*N
31 S(K,L)=1.0
CALL MLTXM(A,S,M,MDIM)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,74)
74 FORMAT(/,10X,21HEIGEN VALUES OF (A*S))
C
IONE=1
TEST=0.5
C
ICNT=0
CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) ICNT=1
DO 230 I=1,M
DO 230 J=1,M
230 S(I,J)=0.0
DO 231 I=1,N
DO 231 J=1,N
K=J+(I-1)*N
L=I+(J-1)*N
231 S(K,L)=1.0
CALL MLTXM(B,S,M,MDIM)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,75)
75 FORMAT(/,10X,21HEIGEN VALUES OF (B*S))
CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) ICNT=ICNT+1
IF(ICNT.EQ.2) WRITE(LU,401)
C177 C FIND EIGENVALUES OF ABS(A)+ABS(B)
0178 C
0179 C DO 33 I=1,M
0180 DO 33 J=1,M
0182 33 S(I,J)=ABS(A(I,J))+ABS(B(I,J))
0183 CALL RNAN(MDIM,M,S,EvRI,EVI,IERR)
0184 WRITE(LU,76)
0185 76 FORMAT(/,10X,29H EIGEN VALUES OF ABS(A)+ABS(B))
0186 C
0187 TEST=1.0
0188 C CALL PNT(EVRI,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0189 CALL PNT(EVRI,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0190 IF(ISTAB) WRITE(LU,401)
0191 C
0192 C FIND EIGENVALUES OF A*B
0193 C
0194 CALL MLTMX(A,B,M,MDIM)
0195 CALL RNAN(MDIM,M,S,EvRI,EVI,IERR)
0196 WRITE(LU,77)
0197 77 FORMAT(/,10X,29H EIGEN VALUES OF (A*B))
0198 CALL PNT(EVRI,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0199 IF(ISTAB) WRITE(LU,401)
0200 GO TO 501
0201 500 IF(ISTAB) IRTCD = 1000
0202 501 RETURN
0203 END
0204 SUBROUTINE PNT(EVRI,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0205 LOGICAL ISTAB
0206 DIMENSION EVR(MDIM),EVI(MDIM),IERR(MDIM)
0207 C
0208 ISTAB=.FALSE.
0209 D=1.0E-20
0210 RX=0.0
0211 DO 20 I=1,M
0212 R=EVRI(I)**2+EVI(I)**2
0213 R=SQRTR(R)
0214 RX=MAX1(RX,R)
0215 IF(R.RT.D) GO TO 20
0216 IF(IERR(I).LT.0) WRITE(LU,93) I,IERR(I)
0217 20 CONTINUE
0218 WRITE(LU,30) RX
0219 C
0220 IF(IONE.EQ.0.AND.RX.EQ.TEST) ISTAB=.TRUE.
0221 IF(IONE.EQ.1.AND.RX.LE.TEST) ISTAB=.TRUE.
0222 10 FORMAT(1H ,E14.7,4X,2H+J,E14.7)
0223 11 FORMAT(13H ABS(LMDA) = ,E14.7)
0224 30 FORMAT(19H SPECTRAL RADIUS = ,E14.7/)
0225 93 FORMAT(/,10X,"IERR(",I2," ) = ",I2/
0226 RETURN
0227 END
0228 SUBROUTINE RNAN(N,M,S,EvRI,EVI,IERR)
0229 C SUBROUTINE WAS WRITTEN TO CALL HSBG AND ATEIG IBM SCIENTIFIC
0230 C SUBROUTINES TO CALCULATE THE EIGENVALUES OF A REAL MATR
0231 C M----ORDER OF THE MATRIX S
0232 C N----SIZE OF FIRST DIMENSION ASSIGNED TO THE ARRAY S IN THE
0233 C CALLING PROGRAM
DIMENSION S(25,25),EVR(25),EVI(25)
COMMON/WORK/LANA(25)
CALL HSFG(M,S,N)
CALL ATEIG(M,S,EVR,EVI,LANA,N)
RETURN
END
SUBROUTINE ATEIG
PURPOSE
COMPUTE THE EIGENVALUES OF A REAL ALMOST TRIANGULAR MATRIX
USAGE
CALL ATEIG(M,A,RR,RI,LANA,IA)
DESCRIPTION OF THE PARAMETERS
M ORDER OF THE MATRIX
A THE INPUT MATRIX, M BY M
RR VECTOR CONTAINING THE REAL PARTS OF THE EIGENVALUES ON RETURN
RI VECTOR CONTAINING THE IMAGINARY PARTS OF THE EIGENVALUES ON RETURN
LANA VECTOR WHOSE DIMENSION MUST BE GREATER THAN OR EQUAL TO M, CONTAINING ON RETURN INDICATIONS ABOUT THE EIGENVALUES APPEARED (SEE MATH. DESCRIPTION
IA SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARRAYS IN THE CALLING PROGRAM WHEN THE MATRIX IS IN DO SUBSCRIPTED DATA STORAGE MODE.
IA-M WHEN THE MATRIX IS IN SSP VECTOR STORAGE MODE.
REMARKS
THE ORIGINAL MATRIX IS DESTROYED
THE DIMENSION OF RR AND RI MUST BE GREATER OR EQUAL TO M
SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE
METHOD
QR DOUBLE ITERATION
REFERENCES
J. H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROBLEM
CLARENDON PRESS, OXFORD, 1965.

SUBROUTINE ATEIG(M,A,RR,RI,LANA,IA)
DIMENSION A(1),RR(1),RI(1),P(RR(1)),PRI(1),LANA(1)
INTEGER P,P1,Q
E7=1.0E-8
E6=1.0E-6
E10=1.0E-10
DELTA=0.5
MAXIT=30
INITIALIZATION

N=N
N1=N-1
IN=N1*IA
NN=IN+N
IF(N1) 30,1300,30
NP=N+1

ITERATION COUNTER

IT=0

ROOTS OF THE 2ND ORDER MAIN SUBMATRIX AT THE PREVIOUS ITERATION

DO 40 I=1,2
PRR(I)=0.0
40
PRI(I)=0.0

LAST TWO SUBDIAGONAL ELEMENTS AT THE PREVIOUS ITERATION

PAN=0.0
PAN1=0.0

ORIGIN SHIFT

R=0.0
S=0.0

ROOTS OF THE LOWER MAIN 2 BY 2 SUBMATRIX

N2=N1-1
NI=IN-IA
NN1=IN1+N
IN1=IN+N1
IN11=IN1+N1

60 T=A(N1N1)-A(NN)
U=T*T
V=4.0*A(N1N)*A(NN1)
IF(ABS(V)-U*E7) 100,100,65
65 T=U+V
67 IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67,67,68
68 U=(A(N1N1)+A(NN))/2.0
V=SQR(TABS(T))/2.0
IF(T)>40,70,70
70 IF(U) 80,75,75
75 RR(NL)=U+V
RR(NL)=U-V
GO TO 130
80 RR(NL)=U-V
RR(NL)=U+V
GO TO 130
100 IF(T)120,110,110
110 RR(N1)=A(N1N1)
RR(N)=A(NN)
GO TO 130
0349  120  RR(N1)=A(NN)
0350       RR(N)=A(N1N1)
0351  130  RI(N)=0.0
0352       RI(N1)=0.0
0353       GO TO 160
0354  140  RR(N1) = U
0355       RR(N)=U
0356       RI(N1)=V
0357       RI(N)=V
0358  160  IF(N2)1280,1280,180
0359       IF(N2)1280,1280,180
0360       TESTS OF CONVERGENCE
0361       C
0362  180  N1N2=N1N1-IA
0363       RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
0364       EPS=E10*SQRTR(RMOD)
0365       IF(ABS(A(N1N2))<EPS)1280,1280,240
0366  240  IF(ABS(A(N1N1))<E10*ABS(A(NN))) 1300,1300,250
0367  250  IF(ABS(PAN-A(N1N2))<ABS(A(N1N2))*E6) 1240,1240,260
0368  260  IF(ABS(PAN-A(NNN))<ABS(A(N1N1))*E6)1240,1240,300
0369  300  IF(IT-MAXIT) 320,1240,1240
0370       C
0371       C      COMPUTE THE SHIFT
0372       C
0373  320  J=1
0374       DO 360 I = 1,2
0375       K=NP-I
0376       IF(ABS(RR(K)-PRR(I))+ABS(RI(K)-PRI(I))=DELTA*ABS(RR(K))
0377       1
0378       S=RI(K)) 340,360,360
0379  340  J=J+1
0380       360  CONTINUE
0381       GO TO (440,460,460,480),J
0382  440  R=0.0
0383       S=0.0
0384       GO TO 500
0385  460  J=NP+2-J
0386       R=RR(J)*RR(J)
0387       S=RR(J)+RR(J)
0388       GO TO 500
0389  480  R=RR(N)*RR(N1)-RI(N)*RI(N1)
0390       S=RR(N)+RR(N1)
0391       C
0392       C      SAVE THE LAST TWO SUBDIAGONAL TERMS AND THE ROOTS OF THE
0393       C      SUBMATRIX BEFORE ITERATION
0394       C
0395  500  PAN=A(NN1)
0396       PAN1=A(N1N2)
0397       DO 520 I=1,2
0398       K=NP-I
0399       PRR(I)=RR(K)
0400       PRI(I)=RI(K)
0401       C
0402       C      SEARCH FOR A PARTITION OF THE MATRIX, DEFINED BY P AND Q
0403       C
0404       P=N2
0405       IF (N-3)600,600,525
0406       525  IPI=N1N2
0407       DO 580 J=2,N2
0408       IF(ABS(A(IPI))<EPS) 600,600,530
0409       530  IP1P=IPI+1A
QR Double Iteration

Initialization of the Transformation

Row Operation
COLUMN OPERATION

IF(I-N1)1080,1060,1060

K=N

GO TO 1100

DO 1180 J=Q,K

JI=JI-LK

T=PSI1*A(JIP)

IF(I-N1)1120,1140,1140

JIP2=JIP+L1

T=T+PSI2*A(JIP2)

ETA=ALPHA*(T+A(JI))

A(JI)=A(JI)-ETA

A(JIP)=A(JIP)-ETA*PSI1

IF(I-N1)1160,1180,1180

A(JIP2)=A(JIP2)-ETA*PSI2

CONTINUE

IF(I-N2)1200,1220,1220

JI=II+3

JIP=JIP+IA

JIP2=JIP+IA

ETA=ALPHA*PSI2*A(JIP2)

A(JI)=ETA

A(JIP)=ETA*PSI1

A(JIP2)=ETA*PSI2

II=IIP+1

IT=IT+1

GO TO 60

END OF ITERATION

IF(ABS(A(N1))<ABS(A(N1N2))) 1300,1280,1280

TWO EIGENVALUES HAVE BEEN FOUND

ONE EIGENVALUE HAS BEEN FOUND
C SUBROUTINE HSBG

C PURPOSE
TO REDUCE A REAL MATRIX INTO UPPER ALMOST TRIANGULAR F

C USAGE
CALL HSBG(N,A,IA)

C DESCRIPTION OF THE PARAMETERS
N ORDER OF THE MATRIX
A THE INPUT MATRIX, N BY N
IA SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARR
A IN THE CALLING PROGRAM WHEN THE MATRIX IS IN
DOUBLE SUBSCRIPTED DATA STORAGE MODE. IA=N WHE
THE MATRIX IS IN SSP VECTOR STORAGE MODE.

C REMARKS
THE HESSENBORG FORM REPLACES THE ORIGINAL MATRIX IN TH
ARRAY A.

C SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

C METHOD
SIMILARITY TRANSFORMATIONS USING ELEMENTARY ELIMINATI
MATRICES, WITH PARTIAL PIVOTING.

C REFERENCES
I.H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROBLEM -
CLARENDON PRESS, OXFORD, 1965.

SUBROUTINE HSBG(N,A,IA)
DIMENSION A(1)

L=N
NL=NL-IA

L1=L-1
L2=L1-1

L IS THE ROW INDEX OF THE ELIMINATION

20 IF(L-3) 360,40,40
40 LIA=LIA-IA
L1=L-1
L2=L1-1

SEARCH FOR THE PIVOTAL ELEMENT IN THE LTH ROW
0585    ISUB=LIA+L
0586    IPIV=ISUB-IA
0587    PIV=ABS(A(IPIV))
0588    IF(L-3) 90,90,50
0589    50 M=IPIV-IA
0590    DO 80 I=L,M,IA
0591    T=ABS(A(I))
0592    IF(T-PIV) 80,80,60
0593    60 IPIV=I
0594    PIV=T
0595    80 CONTINUE
0596    90 IF(PIV) 100,320,100
0597    100 IF(PIV-ABS(A(ISUB))) 180,180,120
0598 C
0599 C     INTERCHANGE THE COLUMNS
0600 C
0601    120 M=IPIV-L
0602    DO 140 I=1,L
0603    J=M+I
0604    T=A(J)
0605    K=LIA+I
0606    A(J)=A(K)
0607    140 A(K)=T
0608 C
0609 C     INTERCHANGE THE ROWS
0610 C
0611    M=L2-M/IA
0612    DO 160 I=L1,NIA,IA
0613    T=A(I)
0614    J=I-M
0615    A(I)=A(J)
0616    160 A(J)=T
0617 C
0618 C     TERMS OF THE ELEMENTARY TRANSFORMATION
0619 C
0620    180 DO 200 I=L,LIA,IA
0621    200 A(I)=A(I)/A(ISUB)
0622 C
0623 C     RIGHT TRANSFORMATION
0624 C
0625    J=-IA
0626    DO 240 I=1,L2
0627    J=J+IA
0628    LJ=L+J
0629    DO 220 K=1,L1
0630    KJ=K+J
0631    KL=K+LIA
0632    220 A(KJ)=A(KJ)-A(LJ)*A(KL)
0633    240 CONTINUE
0634 C
0635 C     LEFT TRANSFORMATION
0636 C
DO 300 I=1,M
K=K+IA
LK=L+LK
S=A(LK)
LJ=L-LA
DO 280 J=1,L2
JK=K+J
LJ=LJ+IA
280 S=S+A(LJ)*A(JK)*1.000
300 A(LK)=S
SET THE LOWER PART OF THE MATRIX TO ZERO
DO 310 I=L,LIA,IA
310 T(I)=0.0
320 L=L1
GO TO 20
360 RETURN
END
SUBROUTINE MLTIX(A,S,M,MDIM)
SUBROUTINE OBTAINS THE MATRIX MULTIPLICATION OF A AND S AND THE RESULTS IN S.
DIMENSION S(MDIM,MDIM),A(MDIM,MDIM)
COMMON/WORK/T(25,25)
DO 10 I=1,M
DO 10 J=1,M
C=0.0
DO 20 K=1,M
20 C=C+A(I,K)*S(K,J)
10 T(I,J)=C
DO 50 I=1,M
DO 50 J=1,M
50 S(I,J)=T(I,J)
RETURN
END
BLOCK DATA WORK
COMMON /WORK/ WO(625)
END
&FILT T-00004 IS ON CRO0022 USING 00004 BLKS R-0022

0001  FIN4,L
0002  PROGRAM FILTR
0003  C WRITTEN BY E. E. SHERROD
0004  C
0005  C THIS PROGRAM SELECTS THE FILTERING TYPE
0006  C
0007  DIMENSION ILU(5),NAME1(3),NAME2(3),IRTN(5),NAME3(3)
0008  EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
0009  DATA NAME1/2HLF,2HLT,2HR /
0010  DATA NAME2/2HHF,2HLT,2HR /
0011  DATA NAME3/2HSH,2HOW,2H /
0012  C
0013  C GET LU
0014  C
0015  CALL RMPAR(ILU)
0016  IPIXL =0
0017  JPIXL =511
0018  LU=ILU(1)
0019  WRITE(LU,10)
0020  10  FORMAT(" SELECT FILTERING TYPE "/ 1. LINEAR "/ 2. HOMOMORP
0021  READ(LU,* ) IFITR
0022  IF(IFITR .EQ. 1) CALL EXEC(23,NAME1,LU,IPIXL,JPIXL,0,0)
0023  CALL RMPAR(IRTN)
0024  IF(IFITR .EQ. 1) GO TO 3)
0025  IF(IFITR .EQ. 2) CALL EXEC(23,NAME2,LU,IPIXL,JPIXL,0,0)
0026  CALL RMPAR(IRTN)
0027  30  WRITE(LU,40) RMAX,RMIN
0028  40  FORMAT(" MAX PIXEL = ",F12.2,10X," MIN PIXEL = ",1F12.2)
0029  IX=RMAX-RMIN +0.5
0030  WRITE(LU,50) IX
0031  50  FORMAT(" NUMBER OF GRAY LEVELS = ",15)
0032  IF(IFITR .EQ. 2)CALL EXEC(23,NAME3,LU,0,511,0,0)
0033  STOP
0034  END
0035  END$
PROGRAM NOISE

DIMENSION RDATA(512),GNOISE(512),LU(5),IU(5),IBUF(40)

INTEGER READL
EQUIVALENCE (RDATA,LU(2)),(LU(2),ILINE),(LU(3),IPIXL),
(RDATA(2),RMAX),(RDATA(3),RMIN)
DATA RDATA/512*0.0/

C GET INPUT PARAMETERS
CALL RMPAR(LU)

C SCHEDULE BUILD WORK FILE PROGRAM
CALL EXEC(23,6HBLDF, IU)

C READ WORK FILE HEADER
IERR = READL(-1,0,511,RDATA)
IF (IERR .LT. 0) GO TO 999
NLINE=ILINE
NPIXL=IPIXL
P.MAX=RMAX
P.MIN=RMIN

C GET NOISE INFO
WRITE(LU,13)
FORMAT(" ENTER NOISE MEAN VALUE ")
READ(LU,*) AM
WRITE(LU,14)
FORMAT(" ENTER STANDARD DEVIATION VALUE ")
READ(LU,*) S
IF(S .LE. 0) GO TO 1000

DO 100 I=0,NLINE-1
IF (READL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999

C GET NOISE
DO 101 JA=0,51
CALL EXEC(1,8,IBUF,40)
JJ=10*JA
CALL CODE (80)
READ( IBUF,12) (GNOISE(K+JJ),K=1,10 )
FORMAT(10F8.5)
CONTINUE
DO 90 J =1,NPIXL
RDATA(J) = RDATA(J) + GNOISE(J) * S + AM
600 FORMAT( F20.3)
90 CONTINUE
C
WRITE SIGNAL + NOISE TO WORK FILE
C
IF(RITEL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
IF(MOD(I,64) .EQ. 0) WRITE(LU,4)
4 FORMAT(" **** ADDING NOISE ****")
100 CONTINUE
1000 CALL CLSWF(NLINE,NPIXL,PMAX,PMIN)
CALL CLOSE(IDC1)
999 WRITE(LU,2) IERR
2 FORMAT("FILE ERROR",I7)
END
$
APPENDIX E
Stability Analysis of Two-Dimensional Digital Recursive Filters

WINSEER E. ALEXANDER, MEMBER, IEEE, AND STEVEN A. PRUESS

Abstract—A new approach to the stability problem for the two-dimensional digital recursive filter is presented. The bivariate difference equation representation of the two-dimensional recursive digital filter is converted to a multi-input—multi-output (MIMO) system similar to the state-space representation of the one-dimensional digital recursive filter. In this paper, a pseudo-state representation is used and three coefficient matrices are obtained. A general theorem for stability of two-dimensional digital recursive filters is derived and a very useful theorem is presented which expresses sufficient requirements for instability in terms of the spectral radii of these matrices.

I. INTRODUCTION

A two-dimensional digital recursive filter can be characterized by the bivariate difference equation

\[ g(m,n) = \sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} f(m-J, n-K) \]
\[ - \sum_{J=0}^{L} \sum_{K=0}^{L} b_{JK} g(m-J, n-K) \]  

(1)

where the coefficients \( a_{JK} \) and \( b_{JK} \) are constants [1] and some of these constants may be zero. In general, this form does not require that the corresponding numerator and denominator polynomials for the two-dimensional Z transform of the transfer function both be of degree \( L \). Zeros may be added to form the structure as given in (1).

There are two major problems to consider in the design of recursive filters for two-dimensional signal processing: synthesis and stability. The synthesis problem consists of determining the filter coefficients so that the required frequency response is realized. If the resulting filter is to be useful, it must be bounded-input—bounded-output (BIBO) stable. In this paper the stability problem is considered and a new approach to stability analysis for the two-dimensional digital recursive filter is presented.

For the one-dimensional case, there are essentially two methods of determining necessary and sufficient conditions for stability of digital filters: examining regions of analyticity for the characteristic polynomial and by direct evaluation of the characteristics of the impulse response [2]—[4]. In particular, if the system corresponding to the digital filter is represented by a state-space equation, then one can determine stability from the coefficient matrices in the state-space equation [4]. For the two-dimensional case, generalizations of the first method involves examining regions of analyticity for bivariate polynomials [5].
This paper attempts to generalize the second method for the two-dimensional case, i.e., to establish stability by computing the spectral radii of coefficient matrices with real coefficients. The spectral radius of a matrix is the magnitude of the largest magnitude eigenvalue of that matrix.

II. **PSEUDO-STATE-SPACE REPRESENTATION**

A pseudo-state-space representation of (1) is used in the development of the stability analysis theorems in this paper. This representation is very similar to a state-space representation of the two-dimensional digital recursive filter as defined by Fornasini and Marchesini [6]. The two can be made to be equivalent by letting one of the coefficient matrices in the Fornasini and Marchesini model be the null matrix. The pseudo-state-space representation of the two-dimensional recursive filter is given by

\[
G_{m,n} = \bar{B}_1 G_{m,n-1} + \bar{B}_2 G_{m-1,n} + \bar{A} F_{m,n}
\]

\[
g(m,n) = DG_{m,n}
\]

\(G_{m,n}\) is a column vector such that its elements are the outputs, \(g(m-J,n-K)\) where \(0 < J < L\) and \(0 < K < L\). Note that \(G_{m,n}\) contains all of the outputs that are represented in (1) including \(g(m,n)\). Similarly, \(F_{m,n}\) is a column vector such that its elements are the inputs, \(f(m-J,n-K)\) where \(0 < J < L\) and \(0 < K < L\).

We can then define matrices \(\bar{B}_1\), \(\bar{B}_2\), and \(\bar{A}\) such that (1) and (2) are equivalent. The matrices \(\bar{B}_1\), \(\bar{B}_2\), and \(\bar{A}\) are all of order \((L + 1)^2\) by \((L + 1)^2\). The vector \(D\) is a row vector with \(L + 1\) elements.

The ordering of the outputs in \(G_{m,n}\) and of the inputs in \(F_{m,n}\) is not unique. However, the ordering does affect the relative position of the elements of the corresponding coefficient matrices. Also note that \(G_{m-1,n}\) and \(G_{m,n-1}\) have elements in common. Where this occurs, the corresponding elements of \(\bar{B}_1\) and \(\bar{B}_2\) can be divided such that the magnitude of each is no larger than that of the corresponding \(b_{jk}\) or one as appropriate. It is convenient to consistently divide equally and choose a particular ordering scheme for \(G_{m,n}\).

**Example**

Consider the two-dimensional digital recursive filter with bivariate difference equation given by

\[
g(m,n) = a_{00} f(m,n) + a_{10} f(m-1,n) + a_{01} f(m,n-1) + a_{11} f(m-1,n-1) - b_{10} g(m-1,n) \]

\[-b_{11} g(m-1,n-1) - b_{20} g(m,n-1) - b_{21} g(m-1,n-1).
\]

This for example, with \(G_{m,n}\) and \(F_{m,n}\) given in transpose form, we have

\[
G_{m,n} = \begin{bmatrix} g(m,n) & g(m-1,n) & g(m,n-1) & g(m-1,n-1) \end{bmatrix}^T
\]

\[
F_{m,n} = \begin{bmatrix} f(m,n) & f(m-1,n) & f(m,n-1) & f(m-1,n-1) \end{bmatrix}^T
\]

Iii. **STABILITY ANALYSIS**

The stability analysis herein will be confined to the linear shift invariant (LSI) two-dimensional discrete system. Such a system is BIBO stable if and only if the discrete impulse response of the system, \(h(m,n)\), is absolutely summable, i.e., \(\sum_{m,n} |h(m,n)| < \infty\) [1].

Let us define the particular vector \(H_{JK}\) as that input vector which represents a single unit sample at the \((J,K)\) position of the two-dimensional data array with all other inputs samples zero. Let us further define the initial condition vectors, \(G_{J-1,K}\) and \(G_{J,K-1}\), as null vectors. Then for \(m = J\) and \(n = K\), (2) reduces to

\[
G_{J,K} = \bar{A} H_{JK}
\]

\[
h(J,K) = DG_{J,K}
\]

Define the term \(C(\bar{B}_1^J, \bar{B}_2^K)\) as the sum of all product terms involving all permutations of \(\bar{B}_1\) as a factor \(J\) times and \(\bar{B}_2\) as a factor \(K\) times. It is helpful to note that if \(\bar{B}_1\) and \(\bar{B}_2\) commute, then

\[
C(\bar{B}_1^J, \bar{B}_2^K) = \left(\frac{J + K}{K}\right) \bar{B}_1^J \bar{B}_2^K = \left(\frac{J + K}{K}\right) \bar{B}_1^J \bar{B}_2^K / (J!K!)
\]

In general, the matrices do not commute. Therefore, we give as an example \(C(\bar{B}_1^J, \bar{B}_2^K) = \bar{I} + \bar{B}_1 \bar{B}_2 + \bar{B}_1 \bar{B}_2 \bar{B}_1 + \bar{B}_2 \bar{B}_1 \bar{B}_2\).

**Lemma 1**

The contribution to the output vector, \(G_{m,n}\), for a single input vector, \(H_{J,K}\), which corresponds to a unit impulse at the \((J,K)\) position where \(J < m\) and \(K < n\), is given by

\[
G_{m,n} = C(\bar{B}_1^{m-J}, \bar{B}_2^{n-K}) \bar{A} H_{J,K}
\]

for the LSI system represented by (2).

The proof of Lemma 1 is given in the Appendix. Lemma 1 provides a convenient means of finding the output of the two-dimensional digital recursive filter for all values of \(m\) and \(n\) when the filter is excited by a single input at any point in the array. Since the filter is linear and shift invariant, we can use the principle of superposition to find the output for any particular sequence of inputs. Thus the unit impulse response of the filter is given.
Several other theorems relating to sufficient conditions for stability have been found [7]. However, it has been shown that these constraints are too restrictive for general use. That is, useful stable filters can be found which do not satisfy the corresponding sufficient conditions for stability.

Computer algorithms are readily available to find the spectral radius of a matrix with real coefficients. Thus Theorem 2 presents a convenient and easily implemented technique to assess the stability of two-dimensional digital recursive filters.

**Appendix**

In this Appendix, the proofs for Lemmas 1 and 2 and Theorem 2 are given. When a specific norm is not given, any convenient norm is appropriate.

**A1. Proof of Lemma 1**

We proceed with a proof by induction. If we use (2) and (8) to obtain \( G_{j+1,k} \) and \( G_{j+1,k+1} \) for input vector \( H_{j,k} \) and if all initial condition vectors are null vectors, we obtain

\[
G_{j+1,k} = \tilde{B}_2 G_{j,k} = \tilde{B}_2 \tilde{A} H_{j,k}
\]

\[
G_{j+1,k+1} = \tilde{B}_1 G_{j,k+1} + \tilde{B}_2 G_{j+1,k} = \left( \tilde{B}_1 \tilde{B}_2 + \tilde{B}_2 \tilde{B}_1 \right) \tilde{A} H_{j,k}
\]

If we use Lemma 1, we obtain

\[
G_{j+1,k} = C(\tilde{B}_1, \tilde{B}_2) \tilde{A} H_{j,k} = \tilde{B}_1 \tilde{A} H_{j,k}
\]

\[
G_{j+1,k+1} = C(\tilde{B}_1, \tilde{B}_2) \tilde{A} H_{j,k} = \left( \tilde{B}_1 \tilde{B}_2 + \tilde{B}_2 \tilde{B}_1 \right) \tilde{A} H_{j,k}
\]

Thus for any arbitrary \( m \) and \( n \) such that \( m > J \) and \( n > K \), we can use (2) to write

\[
G_{m+1,n} = \tilde{B}_2 G_{m,n} + \tilde{B}_1 G_{m+1,n-1}
\]

\[
= \tilde{B}_2 C(\tilde{B}_1, \tilde{B}_2) \tilde{A} H_{j,k}
\]

Consider the term, \( C(\tilde{B}_1, \tilde{B}_2) \). All of the products in the term either have \( \tilde{B}_1 \) as the first factor or \( \tilde{B}_2 \) as the first factor. If \( \tilde{B}_1 \) is the first factor, we must post-multiply by the sum of all possible products such that the power of \( \tilde{B}_1 \) is decreased by one. If \( \tilde{B}_2 \) occurs as the first factor, we must post-multiply by the sum all possible products such that the power of \( \tilde{B}_2 \) is decreased by one. We conclude that

\[
C(\tilde{B}_1, \tilde{B}_2) = \tilde{B}_1 C(\tilde{B}_1^{-1}, \tilde{B}_2^{-1}) + \tilde{B}_2 C(\tilde{B}_1, \tilde{B}_2^{-1})
\]

for all \( J \) and \( K \), such that both \( J \) and \( K \) are greater than or
equal to one. It follows directly that
\[ G_{m+1,n} = C(B_1^{m+1-j}, B_2^{n+1-k}) \bar{A} H_{j,k}. \] (A6)

Similarly from (2) we can write
\[ G_{m+1,n} = \bar{B}_1 G_{m-1,n+1} + \bar{B}_2 G_{m,n}. \] (A7)

Using (9) to find expressions for \( G_{m-1,n+1} \) and \( G_{m,n} \), we have
\[ G_{m+1,n} = \left[ \bar{B}_1 C(B_1^m, B_2^{n+1-k}) + \bar{B}_2 C(B_1^{m+1-j}, B_2^n) \right] \bar{A} H_{j,k}. \] (A8)

It follows that
\[ G_{m+1,n} = C(B_1^m, B_2^{n+1-k}) \bar{A} H_{j,k}. \] (A9)

Finally, from (2) we obtain
\[ G_{m+1,n+1} = \bar{E}_1 G_{m+1,n} + \bar{E}_2 G_{m,n}. \] (A10)

Using Lemma 1 to express \( G_{m+1,n} \) and \( G_{m,n} \), we obtain
\[ G_{m+1,n+1} = \left[ \bar{B}_1 C(B_1^m, B_2^{n+1-k}) + \bar{B}_2 C(B_1^{m+1-j}, B_2^n) \right] \bar{A} H_{j,k}. \] (A11)

It follows from (A5) and (A11) that
\[ G_{m+1,n+1} = C(B_1^m, B_2^{n+1-k}) \bar{A} H_{j,k}. \] (A12)

and Lemma 1 holds.

A2. Proof of Lemma 2

In the proof of Lemma 2, we shall show that if the response to a particular sequence of input vectors can be represented as given in Lemma 2, then the system is unstable if \( \rho(Q) > 1 \) \[9\].

Define the eigenvalue corresponding to the spectral radius of \( Q \) as \( \lambda_Q \) and the corresponding eigenvector as \( P_Q \). Then if the system transfer function has mutually prime numerator and denominator polynomials we can select a sequence of input vectors such that \( \bar{A} F_{j,n} = \epsilon P_Q + R_{j,n} \) for all \( j \) and \( n \). \[13\]

where \( \epsilon \) is an arbitrary nonzero finite constant and \( R_{j,n} \) is not in the direction of \( P_Q \). We then have
\[ G_{m,n} = \bar{Q}^m \bar{A} F_{j,n} = \epsilon \bar{Q}^m P_Q + \bar{Q}^m R_{j,n}. \] (A14)

Then since \( \lambda_Q \) is the eigenvalue corresponding to the spectral radius, the norm of \( G_{m,n} \) is dominated by the term \( \epsilon \bar{Q}^m P_Q \) in the limit as \( m \) approaches infinity. Thus
\[ S = \lim_{m \to \infty} ||G_{m,n}|| = \lim_{m \to \infty} ||\epsilon \bar{Q}^m P_Q|| = \lim_{m \to \infty} ||\lambda_Q^m P_Q||. \] (A15)

Note that \( S \) is finite if \( \lambda_Q \) is greater than one and Lemma 2 holds.

A3. Proof of Theorem 2

For this proof, we show that we can find a particular sequence of inputs that give unbounded output if any one of the spectral radii specified in Theorem 2 is greater than one.

From Lemma 1 the output from a single arbitrary bounded input at the \((j,k)\) position can be given by
\[ G_{m,n} = f(j,k) C(B_1^{m-j}, B_2^{n-k}) \bar{A} H_{j,k} \]
\[ g(M,N) = DG_{M,N} \] (A16)

where \( f(j,k) \) is the scalar input at the \((j,k)\) position. If we let \( K = N \) and \( J = 0 \) in (A16), we have
\[ G_{M,N} = f(0,N) C(B_1^M, B_2^N) \bar{A} H_{0,N} = f(0,N) B_1^M \bar{A} H_{0,N}. \] (A17)

If we apply Lemma 2, we see that the system is unstable if \( \rho(B_1) > 1 \). If we let \( J = M \) and \( K = 0 \) in (A16), we have
\[ G_{M,N} = f(M,0) C(B_1^M, B_2^N) \bar{A} H_{M,0} = f(M,0) B_2^M \bar{A} H_{M,0}. \] (A18)

If we apply Lemma 2, we see that the system is unstable if \( \rho(B_2) > 1 \).

If we use a particular sequence of inputs \( f(j,M-J) \) for \( 0 < j < M \) where all \( f(j,M-J) \) are bounded and equal. Using the principle of superposition and (A16) we have
\[ G_{M,M} = \sum_{J=0}^{M} f(M-J,J) C(B_1^{M-J}, B_2^J) \bar{A} H_{M-J,J}. \] (A19)

Since all inputs are equal, we can write
\[ G_{M,M} = f(0,M) \left[ \sum_{J=0}^{M} C(B_1^{M-J}, B_2^J) \right] \bar{A} H_{0,M} \] (A20)

\[ G_{M,M} = f(0,M) (B_1 + B_2)^M \bar{A} H_{0,M} \] (A21)

since
\[ (B_1 + B_2)^M = \sum_{J=0}^{M} C(B_1^{M-J}, B_2^J) \] (A22)

whether or not \( B_1 \) and \( B_2 \) commute. If we apply Lemma 2, we see that the system is unstable if \( \rho(B_1 + B_2) > 1 \) and Theorem 2 holds.

ACKNOWLEDGMENT

The author wishes to thank E. I. Jury of the Department of Electrical Engineering, University of California, Berkeley, and the reviewers of this paper for supplying helpful comments, suggestions and references.

REFERENCES


Steven A. Pruess was born in Cedar Rapids, IA, on June 8, 1944. He received the B.S. degree in mathematics from Iowa State University, Ames, in 1966 and the M.S. and Ph.D. degrees in computer science from Purdue University, Lafayette, IN, in 1968 and 1970, respectively. Since 1970 he has been a member of the faculty of the Department of Mathematics and Statistics at the University of New Mexico, Albuquerque, where he is currently Associate Professor. His research interests are in numerical analysis and approximation theory. Dr. Pruess is a member of the Society for Industrial and Applied Mathematics and the American Mathematical Society.

Winmer E. Alexander (M'70) was born in Columbia, NC, on August 19, 1942. He received the B.S. degree in electrical engineering from North Carolina Agricultural and Technical State University, in 1964. He received the M.S. degree in engineering and the Ph.D degree in electrical engineering from the University of New Mexico, Albuquerque, in 1966 and 1974, respectively. He is currently Professor and Chairman of the Department of Electrical Engineering at North Carolina A and T State University. Before joining A and T he was a member of the technical staff at Sandia Laboratories from January 1970 to January 1976 and from June 1964 to July 1966. He served as an officer in the U.S. Air Force from July 1966 to December 1969 and was honorably discharged as a Captain in December 1969. His research interests are digital signal processing and image processing.
STABILITY ANALYSIS OF TWO DIMENSIONAL DIGITAL RECURSIVE FILTERS

Winser E. Alexander

Department of Electrical Engineering
North Carolina A&T State University

ABSTRACT

This paper presents a new procedure for stability analysis of two dimensional recursive digital filters. A matrix recursive equation which is similar to the state space representation of the one dimensional digital recursive filter is formulated. This matrix recursive equation is used to assess stability of the two dimensional digital recursive filter in terms of the spectral radii of the coefficient matrices.

Examples of the use of this technique to assess stability of two dimensional digital recursive filters are given. It is demonstrated that this technique reduces the stability analysis problem to examining the spectral radii of matrices with constant coefficients.

INTRODUCTION

A causal two dimensional digital recursive filter may be represented by the bivariate difference equation

\[
g(m,n) = \sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} f(m-J,n-K) + \sum_{J=0}^{L} \sum_{K=0}^{L} b_{JK} g(m-J,n-K)
\]

where some of the coefficients \(a_{JK}\) and \(b_{JK}\) may be zero. Such a filter uses feedback of past output values to calculate the current output. Therefore, it may be bounded input-bounded output (BIBO) unstable. That is, the output may not be bounded for a given bounded input. This paper considers this stability problem and presents a simple technique to assess stability of two dimensional recursive digital filters.

The Matrix Recursive Form

The bivariate difference equation represented by (1) can be described by the matrix recursive equation

\[
G_{m,n} = B_1 G_{m-1,n} + B_2 G_{m,n-1} + Af_{m,n}
\]

where \(G_{m,n}\) is a column vector made up of all outputs in (1), \(F_{m,n}\) is a column vector made up of all inputs in (1) and the matrices \(B_1\), \(B_2\) and \(A\) are appropriate matrices to make (1) and (2) equivalent. The matrices \(B_1\), \(B_2\) and \(A\) are all of order \((L+1)^2\). The current output is then given by

\[
g(m,n) = DG_{m,n}
\]

where \(D\) is a row vector with \((L+1)\) elements.

The ordering of the outputs in \(G_{m,n}\) and of the inputs in \(F_{m,n}\) is not unique. However, the ordering does affect the relative position of the elements of the corresponding \(B_1\) and \(B_2\) matrices. Also note that there are identical elements in \(B_1\) and \(B_2\). Where this occurs, the corresponding elements of \(B_1\) and \(B_2\) can be divided such that the magnitude of each is no longer than that of the corresponding \(b_{IK}\) or one as appropriate. It is convenient to consistently divide evenly and choose a particular ordering scheme.

Example 1

Consider the recursive digital filter with bivariate difference equation given by

\[
g(m,n) = f(m,n) - b_{10} g(m-1,n) - b_{01} g(m,n-1) - b_{11} g(m-1,n-1)
\]

For this example, we have

\[
G_{m,n} = \begin{bmatrix} g(m,n) \\ g(m-1,n) \\ g(m,n-1) \\ g(m-1,n-1) \end{bmatrix}, \quad F_{m,n} = \begin{bmatrix} f(m,n) \\ f(m-1,n) \\ f(m,n-1) \\ f(m-1,n-1) \end{bmatrix}
\]
Stability Analysis

For the one dimensional case, there are essentially two methods of determining necessary and sufficient conditions for stability; examining regions of analyticity for the characteristic polynomial and by direct evaluation of the characteristics of the impulse response \(i, j, k\). In particular, if the filter is represented as a state space equation, then one can determine stability from the coefficient matrices in the state space equation [1]. The usual approach for stability analysis of two dimensional digital recursive filters involves examining regions of analyticity for bivariate polynomials [2] which is computationally feasible only for very simple filters. This paper represents an attempt to generalize the second method for the two dimensional case, i.e., to establish stability by computing the spectral radius of coefficient matrices with real coefficients.

The following theorems relating to stability analysis of two dimensional digital recursive filters have been developed [5]. Space will not allow proof of the theorems in this paper. The reader is referred to reference [5] for further details.

**Theorem 1**

The linear space invariant (LSI) two dimensional digital recursive filter represented by (2) and for which the numerator and denominator polynomials of the corresponding transfer function are mutually prime is unstable if any one of the spectral radii \(\rho(B_1), \rho(B_2), \rho(B_1 + B_2)\) is greater than or equal to one. The spectral radius of a given matrix is the magnitude of the largest magnitude eigenvalue of that matrix).

**Theorem 2**

The LSI two dimensional digital recursive filter represented by (2) is stable if the spectral radius of the matrix made up of the sum of the magnitude of the coefficients of \(B_1\) and \(B_2\) is less than one (\(\rho(\|B_1\| + \|B_2\|) < 1\)).

**Theorem 3**

There is a particular permutation matrix \(\mathbf{S}\) [5] such that if \(\rho(B_1), \rho(B_2), \rho(B_1 + B_2)\) are all less than one, then the LSI digital recursive filter is stable if both \(\rho(B_1 S)\) and \(\rho(B_2 S)\) are less than one-half.

**Conjecture**

If the coefficients of (1) are symmetric such that \(b_{j+k} = b_{k+j}\) for all \(j\) and \(k\), then the LSI recursive digital filter described by (2) and for which the numerator and denominator polynomials of the corresponding transfer function are mutually prime is stable if and only if \(\rho(B_1), \rho(B_2), \rho(B_1 + B_2)\) are all less than one.

**Example 2**

From Theorem 1, we obtain the results that the filter represented by (3) is unstable if \(b_{01} \geq 1\), \(\rho(B_1) \geq 1\), or if

\[
\max \left\{ \frac{-b_{00} - b_{11}}{2}, \frac{\sqrt{b_{00}^2 - 4b_{01}b_{11}}}{} \right\} \geq 1
\]

(7)

**Example 3**

Consider the example (also used by Shanks [6]) where the bivariate difference equation is given by

\[
g(m,n) = f(m,n) + 0.95 g(m-1,n) + 0.95 g(m,n-1) - 0.5 g(m-1,n-1)
\]

(8)

If we apply Theorem 1, we obtain \(\rho(B_1) = 0.95\), \(\rho(B_2) = 0.949\) and \(\rho(B_1 + B_2) = 1.584\). Thus it follows that this filter is unstable.

**Example 4**

Consider the example used by Read and Treitel [7] with bivariate difference equation given by

\[
g(m,n) = f(m,n) + 0.75 g(m-1,n) - 1.5 g(m-1,n-1)
\]

\[-0.9 g(m-2,n) - 1.2 g(m,n-2) - 1.3 g(m-2,n-1)
\]

(9)

If we apply Theorem 1, we obtain \(\rho(B_1) = 1.095\), \(\rho(B_2) = 0.949\) and \(\rho(B_1 + B_2) = 1.284\). We conclude as did Read and Treitel that this filter is unstable.

**Example 5**

Consider the example by Huang [8] with difference equation given by

\[
g(m,n) = f(m,n) - 0.5 g(m-1,n) - 0.5 g(m,n-1)
\]

\[-0.25 g(m-1,n-1) - 0.25 g(m-2,n) - 0.25 g(m,n-1)
\]

(10)
If we apply Theorem 3, we obtain \( p(\mathbf{B}_1) = 0.5, \)
\( p(\mathbf{B}_2) = 0.5, \) \( \omega(\mathbf{B}_1 + \mathbf{B}_2) = 0.866; \) \( p(\mathbf{B}_3) = p(\mathbf{B}_5) = \)
0.333. Therefore, we conclude that this filter is stable. This filter was verified to be stable by Maria and Fahmy [8].

**Example 6**

Consider the example used by Huang [8] with difference equation given by

\[
g(m,n) = f(m,n) - b_{10} g(m-1,n) - b_{01} g(m,n-1) \tag{11}
\]

If we apply Theorem 2, it is interesting to note that we get the same sufficient condition for stability as obtained by Huang:

\[
|b_{10}| + |b_{01}| < 1
\tag{12}
\]

In considering a more complex example, it is convenient to present the coefficients in matrix form. Let the matrix \( \mathbf{V} \) be made up of the elements \( V_{jk} \) for row \( j \) and column \( k \) where

\[
V_{jk} = b_{j-1,k-1}.
\]

For example, the \( \mathbf{V} \) matrix corresponding to example (1) is given by

\[
\mathbf{V} = \begin{bmatrix}
1.0 & b_{01} \\
0 & 1.0
\end{bmatrix}
\tag{13}
\]

Note that \( \mathbf{V} \) is of order \((L+1)\) by \((L+1)\).

**Example 7**

Consider the example used by Read and Treitel where \( \mathbf{V} \) is given by

\[
\mathbf{V} = \begin{bmatrix}
1.0 & 1.5 & -1.9 & -0.8 & 1.1 \\
1.4 & 2.1 & -2.6 & -1.1 & 1.5 \\
-1.8 & -2.4 & 3.3 & 1.3 & -1.6 \\
-0.7 & -0.9 & 1.1 & 0.5 & -0.8 \\
-0.9 & 1.3 & -1.6 & -0.6 & 1.0
\end{bmatrix}
\tag{14}
\]

For this example, \( p(\mathbf{B}_1) = 2.169; \) \( p(\mathbf{B}_2) = 2.104 \) and \( \omega(\mathbf{B}_1 + \mathbf{B}_2) = 2.599. \) Thus Read and Treitel's conclusion that this filter is unstable is verified.

**CONCLUSION**

A new procedure for assessing stability of two dimensional recursive digital filters has been presented. The formulation of the \( \mathbf{B}_1 \) and \( \mathbf{B}_2 \) matrices is very simple and straightforward and the matrices are sparse (mostly zeros). Computer algorithms are readily available to obtain the spectral radius of a matrix with real coefficients. Thus stability analysis is greatly simplified with respect to methods which have previously been presented.

It is also important to note that in this research effort all known unstable filters have been detected as being unstable when Theorem 1 was applied. We surmise that for a very large class of filters, any filter within the class not detected as being unstable after applying Theorem 1 is stable. Research continues to prove or disprove this conjecture.

**REFERENCES**


Two Dimensional Digital Filters for Subjective Image Processing

Winser E. Alexander and Earnest E. Sherrod
Department of Electrical Engineering
North Carolina Agricultural and Technical State University

Abstract

This paper presents a design technique for designing approximately circularly symmetric lowpass, bandpass, high frequency boost and low frequency boost digital filters for subjective image processing applications. An approach is used which parallels the use of the Butterworth, Chebychev or other type of polynomial approximations to obtain one dimensional lowpass digital recursive filters. The other filter designs are then derived from the lowpass filter design. The designed filters are very close to being circularly symmetric for a wide range of critical frequencies. In the design procedure, the squared magnitude characteristic of the desired circularly symmetric filter is chosen in the Laplace Transform domain. The bilinear transformation is then used to map the squared magnitude characteristic into the two dimensional Z-Transform domain. A pseudo-state space representation for the corresponding two dimensional Z-Transform is obtained. The eigenvalues with magnitudes less than one are then used as roots of a denominator polynomial with distinct roots to form the Z-Transform of the stable two dimensional digital filter.

1.0 INTRODUCTION

There are basically two types of image processing: subjective image processing and image correction. Subjective image processing involves the modification of an image in some way to improve the ability of the observer to obtain information or to improve the appearance of the image. Image correction involves the removal of noise or other errors in the image caused by the system producing the image. This paper primarily addresses the design of digital filters for use in subjective image processing.

The user interested in subjective image processing typically desires a variety of filters that can be applied based upon experience or a preliminary evaluation of the subject image. He then wants to observe the results of this filtering operation and make adjustments in the filter parameters before filtering again. Therefore, a computationally efficient algorithm is desirable and fast turn around is vital.

The two dimensional recursive digital filter is a good choice to meet these requirements [1]. The size of the image is not constrained to powers of integers and the number of computations per pixel does not increase as the size of the image is increased. In addition, the image is processed by row which is the normal mode for storage of images on tape or disk.

The common techniques of edge enhancement, contrast enhancement, dynamic range compression, etc. may be accomplished with recursive digital filters. These applications involve lowpass, highpass, bandpass, high boost and low boost digital filters. This paper presents a design technique which can be used to design approximately circularly symmetric recursive digital filters.

2.0 MATHEMATICAL THEORY

The theoretical basis for the two dimensional Z-Transform [2] involves the theory for sample data systems. Given discrete samples of a two dimensional function, \( f(x,y) \) with sampling increments \( X \) and \( Y \) respectively, the Z-Transform for the function is defined by

\[
F(z,w) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(mX, nY) z^{-m} w^{-n}
\]  

(2.1)

If the function is an image, then (2.1) reduces to the case where \( m \) and \( n \) have no negative values and the range of \( m \) and \( n \) is finite. We further restrict the problem to the case where \( X \) and \( Y \) are constants. Then, if we use the notation \( f(m,n) \) to represent \( f(mX,nY) \), we have

\[
F(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} f(m,n) z^{-m} w^{-n}
\]  

(2.2)

as the Z-Transform for the image function, \( f(m,n) \), which has \( (M+1) \) columns and \( (N+1) \) rows.

Consider the case where we have an input image with samples \( f(m,n) \) and we wish to filter this image to obtain an output image with corresponding samples, \( g(m,n) \). The samples of the impulse response of the desired filter are given by \( h(m,n) \). The range of \( m \) and \( n \) for the output is the same as for the input. Thus, the Z-Transform of \( g(m,n) \) is given by

\[
G(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} g(m,n) z^{-m} w^{-n}
\]  

(2.3)

If we restrict the impulse response such that \( m \) and \( n \) cannot be negative (a causal system), we can write the Z-Transform for the impulse response as

\[
H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n) z^{-m} w^{-n}
\]  

(2.4)

In general, the Z-Transform for the impulse response is an infinite series. In order to implement the spatial domain filter, we must find a closed form expression for \( H(z,w) \) such that
The convolution property of the ZW-Transform gives the relationship resulting from the convolution of \( t(m,n) \) and \( h(m,n) \) which is the filtering process:

\[
C(z,w) = H(z,w)F(z,w) \tag{2.6}
\]

If we use the closed form of \( H(z,w) \) and restrict \( b(0,0) \) to be equal to one and write the resulting equation for a single output value \( g(m,n) \), we obtain the difference equation:

\[
L \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} a(j,k)z^{-m-j-n} g(m,n) - \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} b(j,k)g(m-j,n-k) \tag{2.7}
\]

If \( L \) is relatively small (in practice, \( L \) is usually less than \( 10 \) for recursive digital filters), equation (2.7) represents a very efficient algorithm for filtering images. Equations (2.5) and (2.7) may also represent a nonrecursive filter if all \( b(j,k) \) except \( b(0,0) \) are equal to zero.

### 3.0 STABILITY ANALYSIS

Nonrecursive digital filters are inherently stable. Since there is no feedback of past output values, the impulse response has finite duration. Each output value is a finite sum which is always bounded if the input is bounded.

The stability problem for one dimensional digital recursive filters is straightforward. The roots of the denominator polynomial in the closed form of the one dimensional \( Z \)-Transform for the filter impulse response function must have magnitudes less than one. Stability analysis is therefore reduced to finding roots of \( n \)th degree polynomials with real, constant coefficients [3]. Stability analysis is not straightforward for the two dimensional problem because a two variable polynomial is not generally factorable into distinct roots. When the polynomial in the denominator of the two dimensional \( Z \)-Transform of the impulse response is factorable into distinct roots, the stability analysis procedure is the same as for the one dimensional problem.

The two dimensional stability problem is very complicated if the polynomial in the denominator is not factorable into distinct roots [4]. Efforts by other researchers have been directed toward examining regions of roots for two variable polynomials. An alternate method of assessing stability for one dimensional digital recursive filters is to make a state space representation of the filter [5]. Then the filter is stable if the eigenvalues of the state transition matrix all have magnitudes less than one. Previous research has been directed toward developing the two dimensional equivalent of this procedure [6,7]. A pseudo-state variable representation is chosen because of difficulties in finding a true state space representation [8]. This difficulty is caused by the bivariance of the transfer function and by its causality. The resulting matrix equation has two pseudo-state transition matrices.

Alexander [6] has shown that the recursive algorithm of (2.7) can be represented by the matrix recursive equation:

\[
G_{m,n} = G_{m-1,n} + C_{m,n} + AF_{m,n} \tag{3.1}
\]

Where \( G_{m,n} \) is a vector such that the elements of \( G_{m,n} \) are the outputs \( g(m-j,n-k) \) in (2.7) where \( 0 \leq j \leq L \) and \( 0 \leq k \leq L \). \( F_{m,n} \) is a vector such that the elements of \( F_{m,n} \) are the inputs \( f(m-j,n-k) \) in (2.7) where \( 0 \leq j \leq L \) and \( 0 \leq k \leq L \). \( B, C \) and \( A \) are appropriate coefficient matrices such that (2.7) and (3.1) are equivalent.

If the filter is unstable, then either \( B + C \) or \( 1 + C \) has at least one eigenvalue with a magnitude greater than one, equal to one. Thus, stability analysis involves setting up the matrices \( B \) and \( C \) and finding the spectral radius of each matrix individually and of their sum.

### 4.0 SYNTHESIS

Often it is possible to express a desired two dimensional recursive digital filter as the product or sum of two one dimensional digital filters. That is, the \( Z \)-Transform of the two dimensional filter may be expressed as the product:

\[
H(z,w) = H_1(z)H_2(w) \tag{4.1}
\]

or as the sum:

\[
H(z,w) = H_1(z) + H_2(w) \tag{4.2}
\]

In either case, the two dimensional synthesis problem is reduced to the synthesis of two one dimensional filters [9,10]. However, it is not possible to design sum separable or product separable digital recursive filters for all applications. For these applications where sum separable or product separable designs are not possible, the design of the required two dimensional digital recursive filter is considerably more complicated.

Many imaging systems have a natural circular symmetry. In general, the optical transfer function (OTF) of a circularly symmetric imaging system is circularly symmetric. Also, it is usually desirable to perform image processing where the processing is uniform with respect to direction. The natural
consequence is that filters with circularly symmetric impulse response functions are generally very desirable for image processing. A filter with a circularly symmetric impulse response is assured by restricting the Discrete Fourier Transfer (DFT) for the filter to be circularly symmetric [11].

One popular method of designing digital recursive filters is to start with the Laplace Transform of the desired filtering function, make a suitable transformation to the Z-Transform domain and thus obtain the coefficients for the digital recursive filter. One such technique involves designing digital recursive filters from the squared magnitude characteristics of the desired filter which is really the squared magnitude of the Fourier Transform. This procedure is difficult to extend to two dimensions because of the difficulties encountered in factoring bivariate polynomials.

To illustrate this difficulty, consider the circularly symmetric Butterworth low pass filter squared magnitude characteristic.

\[ H(r, \omega) \]

\[ \frac{1}{1 + (-1)^n (r^2 + \omega^2)^n / R^n} \]  

where \( r \) and \( \omega \) are the Laplace Transform variables for the \( x \) and \( y \) direction respectively and \( R \) is the desired radial cutoff frequency.

The bilinear transformation [9] can be used to obtain the corresponding recursive digital filter. First, we prewarp \( H(r, \omega) \) to obtain

\[ H_1(r, \omega) = \frac{1}{1 + a^n (r^2 + \omega^2)^n} \]  

where

\[ a^2 = (-1)^n / (RT/2)^n \]  

(4.5)

(The assumption is made in this example that the sampling increment is the same in each direction and is equal to \( T \).) Applying the bilinear transformation, we have an approximation for the Z-Transform of the squared magnitude characteristic of the desired filter.

\[ H(z, \omega) = \frac{1}{1 + a^n ((z-1)/(z+1))^2 + ((\omega-1)/(\omega+1))^2} \]  

(4.6)

If the polynomial in the denominator of (4.6) were factorable into distinct roots of \( z \) and \( \omega \), then those roots would occur in reciprocal pairs. The design procedure would then be completed by forming \( H(z, \omega) \) from those roots for which the magnitude of \( z \) is less than one and those for which the magnitude of \( \omega \) is less than one. The numerator polynomial of \( H(z, \omega) \) is allowed to have roots with a magnitude of one.

\[ H(z, \omega) \]  

which is formed with the smaller in magnitude of each of the reciprocal pairs of roots in the numerator and denominator is said to have minimum phase. The minimum phase version of any filter is stable for any input sequence unless the denominator of its Z-Transform has roots where either the magnitude of \( z \) or \( \omega \) is equal to one. In that case, it is conditionally stable.

However, the polynomial in the denominator of (4.6) is not factorable into distinct roots. Therefore, forming of the minimum phase version of \( H(z, \omega) \) is not straightforward and the design procedure is not successful.

A minimum phase approximation to \( H(z, \omega) \) can be obtained with the following procedure:

1. Construct the coefficient matrices \( \tilde{B} \) and \( \tilde{C} \) of (3.1) which corresponds to (4.6).
2. Calculate the eigenvalues of the matrix sum \( (\tilde{B} + \tilde{C}) \). They occur in reciprocal pairs.
3. Form the minimum phase approximation of the filter by using the smaller magnitude eigenvalue of each of the reciprocal pairs as a root of \( z \) and of \( \omega \) for the denominator polynomial and by using the minimum phase version of the numerator polynomial.

The resulting filter Z-Transform is given by

\[ H(z, \omega) = \frac{(1+p)^2(z+1)(w+1)}{4(z+p)(w+p)} \]  

where

\[ \rho = \frac{(2a - (2T^2)a + 1)}{1 - 2a} \]  

(4.8)

5.0 FILTER DESIGN

5.1 Low Pass filter

Equation (4.7) gives the Z-Transform for the low pass filter approximation which was derived from the circularly symmetric low pass filter squared magnitude characteristic of (4.3). For this particular design, the roots of \( H(z, \omega) \) are real. In general, the roots may be real or they may occur in complex conjugate pairs. If the resulting filter is applied in a straightforward manner, the algorithm must handle complex numbers. This can be avoided by using a basic filter structure which uses only binomial functions resulting from the multiplication of two roots. When complex roots are involved, the pair of complex conjugate roots would form a basic filter stage. The penalty paid for this basic filter structure is that filters with odd numbers of zero or poles can only be implemented by adding at least one null root. The addition of this null root results in unnecessary calculations in the algorithm which implements the filter. Thus, all filters designed will have the basic structure:

\[ H(z, \omega) = \prod \frac{[z^2 + q(11)z + q(21)][w^2 + q(11)w + q(21)]}{1 \cdot [z^2 + p(11)z + p(21)][w^2 + p(11)w + p(21)]} \]  

(5.1)

The basic low pass filter using this form is then given by

\[ LP(z, \omega) = \frac{(1+p)^2(z^2+2z+1)(w^2+2w+1)}{16(z^2+2p^2)(w^2+2pw+p^2)} \]  

(5.2)
5.2 The Frequency Boost Filter

A frequency boost filter can be designed from the magnitude response characteristics of the low pass filter. Consider the filter which has a ZW-transform given by:

$$H(z, w) = c + d|LP(z, w)|^2$$  \hspace{1cm} (5.3)

Note that (5.3) has roots of $z$ and $w$ with magnitude greater than one since the roots occur in reciprocal pairs. This problem is overcome by using the minimum phase version of (5.3). Thus the ZW-transform of the desired filter is given by:

$$H(z, w) = \frac{V(z, w)}{D(z, w)}$$  \hspace{1cm} (5.4)

where $V(z, w)$ and $D(z, w)$ have minimum phase.

A high frequency boost filter can be designed by changing the values of $c$ and $d$ in (5.3). For the low pass filter, $c$ has a value of one and $d$ has a value of minus one. A low frequency boost filter is desired with a magnitude gain of $BF$ at $DC$ and a gain of one at the Nyquist frequency; this can be achieved by setting:

$$c = 1.0$$  \hspace{1cm} (5.5)
$$d = BF - 1.0$$

If a high frequency boost is desired with a magnitude gain of $BF$ at the Nyquist frequency and a gain of one at $DC$, this can be achieved by setting:

$$c = BF$$  \hspace{1cm} (5.6)
$$d = 1.0 - BF$$

The shape of the resulting filter is also affected by the value of the root $p$ which is derived from the design of the low pass filter. From (4.7) and (4.8), observe that $p$ is a function of the desired radial cutoff frequency $R$, for the low pass filter. Note that three parameters, $c$, $d$ and $R$, are required to design the filter specified by (5.3). However, if a high frequency boost or a low frequency boost filter is desired, then only two parameters, $R$ and $BF$ are required because $c$ and $d$ can be derived from $BF$.

6.0 Filter Design Examples

Figure 1 gives the perspective plot of a lowpass filter designed with the described technique with a cut off frequency which is 0.3 times the Nyquist frequency. Figure 2 gives the contour plot for this filter design. Figure 3 gives the perspective plot for a high frequency boost filter with a break frequency of 0.5 times the Nyquist frequency and a boost magnitude of 25.6. Figure 4 gives the contour plot for this filter design. Note that these examples are essentially circularly symmetric. Some degradation is observed as the break frequency approaches the Nyquist frequency. This causes the mapping characteristics of the bilinear transformation. Some degradation also occurs as the break frequency approaches DC, however, this can be corrected by using rotated filter combinations [12].

7.0 Conclusion

A design technique has been presented which can be used to design approximately circularly symmetric digital recursive filters for subjective image processing applications. These filters include lowpass, highpass, low and high frequency boost and bandpass filters. The filters are inherently stable because the denominator polynomial of the ZW-Transform is minimum phase.

REFERENCES


FIGURE 1. LOW PASS FILTER  
STAGE = 1  
RC = 0.3

FIGURE 2. LOW PASS FILTER  
STAGE = 1  
RC = 0.3

FIGURE 3. HIGH BOOST FILTER  
BOOST FACTOR = 25.6  
RC = 0.5

FIGURE 4. HIGH BOOST FILTER  
BOOST FACTOR = 25.6  
RC = 0.5
ABSTRACT

It is shown that 2D digital filter realizations are equivalent to the solution of tensor equations, and they are also equivalent to the solution of matrix equations. Both recursible and non-recursible filters are included in these formulations.

SUMMARY

A 2D digital filter, which possesses a rational transfer function, may be represented by its bivariate difference equation written in tensor form as:

\[ b_{ij} g_{p+i,q+j} = a_{ij} f_{p+i,q+j} \]  

(1)

where \( 1 \leq p \leq N, 1 \leq q \leq M, -m \leq i \leq m, -m \leq j \leq m \); and the double appearance of an indice on a given side of the equality implying the usual summation over the appropriate range of that indice. A more formal expression of (1) is:

\[ B_{pq} g_{kl} = A_{pq} f_{kl} \]  

(2)

where \( 1 \leq k \leq N, 1 \leq l \leq M \), and the non-zero components of the coefficient tensors given by \( A_{pq} = a_{k-p,1-q} \); and \( B_{kl} = b_{k-p,1-q} \); for \( -m \leq k-p \leq m \), \( -m \leq l-q \leq m \).

The 2D filtering operation requires that one determine all the \( g_{pq} \), given all \( a_{ij} \), \( b_{ij} \), and \( f_{pq} \). A solution will exist and be unique if there exists an inverse of the tensor \( B_{kl} \), say \( C_{uv} \), with \( 1 \leq u \leq N, 1 \leq v \leq M \). For such a case, the filtered solution would then be given by:

\[ g_{uv} = C_{uv} A_{pq} f_{kl} \]  

(3)

Tensor equation (2) can also be interpreted as a matrix equation with the \( A_{pq} \) and \( B_{pq} \) taken as \( N \times N \) dimensional coefficient matrices with row index "pq", column index "kl"; and \( g_{kl}, f_{kl} \) taken as column matrices.

For the case when \( N = M \), and \( a_{00}, b_{00} \neq 0 \); then equation (2) is also expressible as a matrix equation involving only \( N \times N \) matrices given by:

\[ LGR + \sum_{k=-m, k \neq 0}^{m} S_{k} G T_{k} = c PFQ + c \sum_{k=-m, k \neq 0}^{m} S_{k} F U_{k} \]  

(4)

where \( c = a_{00}/b_{00} \), the matrices \( G = [g_{pq}] \), \( F = [f_{pq}] \); and the non-zero components of the coefficient matrices \( L, R, P, Q, S_{k}, T_{k}, U_{k} \) given by:

(i) For \( p, q \) such that \( -m \leq q-p \leq m \):

- \( L_{pq} = b_{q-p,0}/b_{00} \); \( R_{pq} = b_{0,q-p}/b_{00} \); \( P_{pq} = a_{q-p,0}/a_{00} \); \( Q_{pq} = a_{0,q-p}/a_{00} \).

- \( T_{kpq} = b_{k,p-q}/b_{00} - b_{k,0} b_{0,p-q}/b_{00}^{2} \); \( U_{kpq} = a_{k,p-q}/a_{00} - a_{k,0} a_{0,q-p}/a_{00}^{2} \).

(ii) And finally, for \( p, q \) such that \( q-p=k \): \( S_{kpq} = 1 \).

Non-recursible filters generally require solutions of the form given by (3). For recursible filters (4) simplifies allowing solution by compact schemes.