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

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

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

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

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

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 Dcese
(b)

\[
H(z,w) = \sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} z^{-J} w^{-K}
\]

\[
\sum_{J=0}^{L} \sum_{K=0}^{L} b_{JK} z^{-J} w^{-K}
\]

Some of the coefficients, \(a_{JK}\) and \(b_{JK}\) may be zero. The convolution 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:

\[
g(m,n) = \sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} f(m-J,n-K) - \sum_{J=0}^{L} \sum_{K=0}^{L} b_{JK} g(m-J,n-K) \quad (3.7)
\]

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 $Z$-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 add 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 arithematic 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 3. Original Landsat File-3 Image
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:

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

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:

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

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

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

Tensor equation (2) can also be interpreted as a matrix equation with \( A_{pq} \) and \( B_{pq} \) taken as \( N \times N \) dimensional coefficient matrices with row index "pq", column index "kl"; and \( g_{kl} \) and \( f_{kl} \) 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 $g_{k1}$ 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:

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

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 $L, R, P, Q, S_k, T_k, S_k F, U_k$ are given by:
(i) For \( p, q \) such that \(-m \leq q - p \leq m:\)
\[
L_{pq} = b_{q-p,0}/b_{00}; \quad R_{pq} = b_{0,q-p}/b_{00}; \quad P_{pq} = a_{q-p,0}/a_{00}; \quad Q_{pq} = a_{0,q-p}/a_{00};
\]
\[
T_{kpq} = b_{k,p-q}/b_{00} - b_{k,0,b_0,p-q}/b_{00}; \quad U_{kpq} = a_{k,p-q}/a_{00} - a_{k,0,a_0,p-q}/a_{00}.
\]
(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:
\[
a_{k,p-q}/a_{00} - a_{k,0,a_0,p-q}/a_{00}^2 = 0
\]
\[
b_{k,p-q}/b_{00} - b_{k,0,b_0,p-q}/b_{00}^2 = 0
\]
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:

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

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:

$$L_uL_1 \cdot G \cdot R_1R_u = c \cdot PFQ$$

where $L_1$ and $R_1$ are lower triangular, and $L_u$ and $R_u$ are upper triangular matrices. 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:

$$G^{(n+1)} = L^{-1} \left( \sum_{k=-m, k \neq 0}^{m} S_k G^{(n)} T_k \right) \cdot R^{-1} + H$$

where $H = L^{-1} \left( cPFQ + c \sum_{k=-m, k \neq 0}^{m} S_k F U_k \right) \cdot R^{-1}$

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

\[
H(Z,W) = \frac{\sum_{J=0}^{2} \sum_{K=0}^{2} a_{JK}Z^{-J}W^{-K}}{\sum_{J=0}^{2} \sum_{K=0}^{2} b_{JK}Z^{-J}W^{-K}} = \frac{N(Z,W)}{D(z,w)} \tag{1}
\]

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

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

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

(5)

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

(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 off's 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: &NASAL  Reloc: %NASAL

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. SHERROD

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
   TDLPF
   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

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.

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

PROGRAMMER: E.E. SHERROD

Source: &LFLTR

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

TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &HFLTR

Reloc: %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
RITEL
PROGRAM NAME: SHOW

PROGRAMMER: DAVE JOHNSON

Source: &SHOW

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

Modules Called:

WFO0000

Subroutines Called:

READL
WLINE
CLSWF
PROGRAM NAME: BLDIM

PROGRAMMER: DAVE JOHNSON

Source: &BLDIM

Reloc: %BLDIN

Loadfile: LBLDIN

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

Modules Called:

WF0000
DIREC

Subroutines Called:

MVW
ROT8
DCODE
DRIVR
PROGRAM NAME: NOISE

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:

LDIREC-------------------------------------------1
LWTAPE-------------------------------------------1
LDPLAM-------------------------------------------1
LPLOTV-------------------------------------------1
LFDIGN-------------------------------------------2
LBDIM-------------------------------------------2
LLFTR-------------------------------------------2
LHFTR-------------------------------------------3
LSHOW-------------------------------------------3
LFIRO-------------------------------------------3
LIMAGE-------------------------------------------4
LRESIZ-------------------------------------------4
LDINTP-------------------------------------------4
LDINTP-------------------------------------------4
LBLDWF-------------------------------------------5
LDSPY-------------------------------------------5
LCURSR-------------------------------------------5
NASA---------------------------------------------6
<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;SPACE</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;FINT</td>
<td>39</td>
</tr>
<tr>
<td>&amp;INTP</td>
<td>42</td>
</tr>
<tr>
<td>&amp;TAPE</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;FLTR</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;BES1O</td>
<td>92</td>
</tr>
<tr>
<td>&amp;BESJ</td>
<td>93</td>
</tr>
<tr>
<td>&amp;BIDW</td>
<td>95</td>
</tr>
<tr>
<td>&amp;SCROL</td>
<td>98</td>
</tr>
<tr>
<td>&amp;LINE</td>
<td>99</td>
</tr>
<tr>
<td>&amp;DRIVE</td>
<td>100</td>
</tr>
<tr>
<td>&amp;DIG1</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,WTAPE
0006 :MR,%ROTS
0007 :MR,%ICMPW
0008 :MR,%TRMGN
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,%DPLAM
0006 :MR,%DPLA1
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,%EES3
0006 :MR,%DRIVR
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, FLDIGN
0003 : PU, FLDIGN
0004 : LG, 3
0005 : MR, ?FDIGN
0006 : MR, ?FDIGN
0007 : RU, LGADR, 99, 1G
0008 : PU, IOC:: 3
0009 : SP, IOC:: 3
0010 : OF, IOC
0011 :
0012 :

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

0001 : LG, 2
0002 : OF, BLDIM
0003 : PU, BLDIM
0004 : MR, ZBLDIM
0005 : MR, MDD.
0006 : MR, ZROTS
0007 : MR, ZRLINE
0008 : MR, DCODE.
0009 : MR, %ROTS.
0010 : RU, LOADR, 99, OG, %, 2
0011 : SP, IOC:: 3
0012 : OF, IOC:: 3
0013 : RP, IOC:: 3
0014 :

LFLTR 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, ZWFIN.
0007 : MR, DCODE.
0008 : MR, %WLINE.
0009 : MR, ZDRIV.
0010 : RU, LOADR, 99, IOC, %, 2
0011 : SP, IOC:: 3
0012 : OF, IOC:: 3
0013 : RP, IOC:: 3
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, 10G: 3
0009 : OF, 10G
0010 : RP, 10G: 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, %WLINE
0006 : OF, SHOW
0007 : RU, LOADR, 99, 1G
0008 : PU, 10G: 3
0009 : SP, 10G: 3
0010 : OF, 10G
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, 10G: 3
0008 : SP, 10G: 3
0009 : OF, 10G
0010 : RP, 10G: 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, LOG: 3
0009 :SP, LOG: 3
0010 :OF, LOG

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, LOG,, 2
0012 :SP, 1OG: 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, LOG: 3
0013 :TR
### LBLDWF T=00004 IS ON CR00022 USING 00002 BLKS R=0009

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>LG, 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>LG, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>OF, BLDWF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>PU, BLDWF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>MR, %BLDWF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>MR, %ICMPW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>RU, LOADR, 99, 0G, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>SP, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>OF, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>RP, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LDSPLY T=00004 IS ON CR00022 USING 00002 BLKS R=0011

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>LG, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>MR, %DSPLY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>MR, %SCROL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>MR, %WLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>MR, %DRIVR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>MR, %RESET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>OF, DSPLY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>RU, LOADR, 99, 0G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>PU, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>SP, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>OF, 10G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LCURSR T=00004 IS ON CR00022 USING 00002 BLKS R=0012

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>LG, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>MR, %CURSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>MR, %WLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>MR, %RLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>MR, %DRIVR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>MR, %MOVEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>OF, CURSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>RU, LOADR, 99, 1G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>PU, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>SP, 10G:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>OF, 10G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LNOISE T=00004 IS ON CR00022 USING 00002 BLKS R=0011

0001 : LG, 1
0002 : MR, %NOISE
0003 : MR, %BLDWF
0004 : MR, %WFTNT
0005 : MR, %ICMPW
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 CR00022 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, PLOTV
0025 : RP, PLOTV
0026 : OF, HFLTR
0027 : RP, HFLTR
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, PLOTV
0049 : OF, HFLTR
0050 : OF, RESIZ
0051 : OF, IMAGE
0052 : OF, BLDIM
0053 : OF, DINTP
0054 : OF, WTAPE
0055
SNASA1 T=00004 IS ON CR00022 USING 00008 BLKS R=0054

0001 FTN4,L
0002 PROGRAM NASA1
0003 C THIS PROGRAM IS THE FATHER PROGRAM FOR THE IMAGE FILTERING
0004 C PROGRAMS
0005 C
0006 DIMENSION IPRAM(5),NAME(3),NSON(3,8),IMESS(30)
0007 DATA NSON/2HDS,2HPL,2HY,2HFD,2HIG,2HN,2HFI,2HLT,
0008 *2HR,2HRE,2HSI,2HZ,2HOW,2H,2HM,2HAG,2HE, 
0009 *2HNO,2HIS,2HE /
0010 C
0011 C SON PROGRAM NAMES (FILES SAME PRECEEDED WITH "&")
0012 C DSPLY - DISPLAY PROGRAM
0013 C FDIGN - FILTER DESIGN MODULE
0014 C FILTR - FILTER IMPLEMENTION MODULE
0015 C RESIZ - IMAGE MODIFICATION MODULE
0016 C SHOW - DISPLAYS WORK FILE
0017 C IMAGE - IMAGE DATA MANAGEMENT MODULE
0018 C NOISE - ADDITIVE GAUSSIAN NOISE
0019 C
0020 CALL RMP1R(IPRAM)
0021 LU=IPRAM(1)
0022 IF(LU.LE.0) LU=1
0023 C
0024 C NPRG IS THE NUMBER OF SONS
0025 C
0026 NPRG=6
0027 ICNT=9
0028 C
0029 C DISPLAY MENU
0030 C
0031 5 WRITE(LU,30)
0032 30 FORMAT(" SELECT PROCESSING OPTION"/,” 1. IMAGE DISPLAY”/,”
0033 *FILTER DESIGN”/,” 3. FILTER IMAGE”/,” 4. MODIFY IMAGE”/,”
0034 *SHOW WORK FILE”/,” 6. IMAGE DATA MANAGEMENT”/,” 7. NOIS
0035 *ON”/,” 8. TERMINATE PROGRAM”)
0036 READ(LU,*) IOPT
0037 IF(IOPT.EQ.0.OR.IOPT.EQ.1) IOPT = 1
0038 IF(IOPT.LT.1.OR.IOPT.GT.8) GO TO 16
0039 IF(IOPT.EQ.8) GO TO 500
0040 C
0041 IPRAM(2)=IOPT
0042 DO 10 I=1,3
0043 10 NAME(I)=NSON(I,IOPT)
0044 WRITE(LU,15) NAME
0045 15 FORMAT(" MODULE TO BE SCHEDULED IS ",3A2)
0046 GO TO 20
0047 16 WRITE(LU,17)
0048 17 FORMAT(" INVALID RESPONSE")
0049 GO TO 5
0050 C
0051 C
0052 20 ICNW=LU+2008
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(5)
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,2H30)
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
0075 SUBROUTINE FILL(IARAY,IA,N)
0076 C THIS SUBROUTINE FILLS ARRAY IARAY WHICH HAS N WORDS WITH THE
0077 C OF IA.
0078 C
0079 C
0080 DIMENSION IARAY(N)
0081 DO 10 I=1,N
0082 10 IARAY(I)=IA
0083 RETURN
0084 $ END
Program DSPLY

This program displays an image on the GMR-27. Image file must be in format described by image display subsystem.

INTEGER SLUI1, STRTL, STRTP, SCROL
INTEGER NAME(6), IDC8(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, 54-1, 260C2B/
DATA SLUI1/34011B/
DATA LLA0, LEO0, LEC0, LBL0, LLBX, LEB0, LEBX/64000B, 44000B, 54000B
1 70001B, 71777B, 50001B, 51777B/
C
C GET INPUT PARAMETERS
C CALL RMPAR(LU)
IF (LU .LE. 0) LU = 1
C OPEN IMAGE DIRECTORY FILE
100 CALL OPEN(IDC8, IERR, 6HIMDIRC)
IF (IERR .LT. 0) GO TO 991
C GET 14AGE 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)
FORMAT(20X, "I M A G E D I S P L A Y S Y S T E M" //)
FORMAT("IMAGE NAME: dB d@")
FORMAT("# LINES: dB d@", 20X,
1 "# PIXELS/LINE: dB d@")
FORMAT("MIN PIXEL: dB d@", 18X,
1 "MAX PIXEL: dB d@")
FORMAT("TEXT: ", /)
FORMAT("dB", 38 " d@")
FORMAT(" ")
FORMAT("")
WRITE(LU, 25)
READ(LU, 2) NAME
IF (NAME .EQ. 2H/E) GO TO 9000
C FIND IMAGE FILE

CALL RNWD,IDCB
CALL READF(IDCB,IERR,IBLK,256,LEN)
IF (IERR .LT. 0) GO TO 991
IF (LEN .EQ. -1) GO TO 800
DO 120 I=1,6
IF (IBLK(I) .NE. NAME(I)) GO TO 110
CONTINUE
C IMAGE FOUND--CHECK IF ON DISC
IF (IBK12 .EQ. 1) GO TO 130
C IMAGE NOT ON DISC
WRITE(LU,12)
FORMAT("
GO TO 105
C IMAGE IS ON DISC
130 CALL CLOSE(IDCB)
RMN = IBLK(9)
Rxnx = IBLK(10)
WRITE(LU,29)(IBLK(I),I=7,10)
FORMAT(")
CALL EXEC(2,LU,TEXT1,37)
CALL EXEC(2,LU,TEXT2,37)
CALL EXEC(2,LU,TEXT3,37)
WRITE(LU,27)
CALL OPEN(IDCB,IERR,JNAME)
IF (IERR .LT. 0) GO TO 991
C EXTRACT DISPLAY INFORMATION
NUML = IBLK7
Nump = IBLK8
STRTL = (256-MINO(256,NUML))/2
STRTP = (512-MINO(512, Nump))/2
ISET5 = IOR(LLAO,IAND(STRTL,1777B))
ISET6 = IOR(LEAO,IAND(STRTP,1777B))
ISET7 = LLB1
ISET8 = LEB1
ISET9 = IOR(LECO,IAND(STRTP,1777B))
CAL DRIVR(2,ISET,10)
0111    IERR = 0
0112    DO 600 I=1,MINO(NUML,256)
0113    IF (IERR .LT. 0) GO TO 991
0114    CALL READF(IDCBL,IERR,IBLK,512,NUM)
0115    IF (NUM .LT. 0) GO TO 600
0116    C
0117    DO 595 J=1,NUM
0118    IBLK(J) = (255./(RMAX-RMIN))*(FLOAT(IBLK(J))-IUIIN)
0119    IF (IBLK(J) .LT. 0) IBLK(J) = 0
0120    IF (IBLK(J) .GT. 377B) IBLK(J) = 377B
0121    595 CONTINUE
0122    C
0123    IBLK(NUM+1) = SLU11
0124    CALL DRVFR(40002B,IBLK,NUM+1)
0125    600 CONTINUE
0126    IFRST = 0
0127    ILAST = 255
0128    C
0129    C OUTPUT SOFT KEY FUNCTIONS
0130    C
0131    WRITE(LU,29)
0132    29 FORMAT(/"FUNCTION KEYS:"/)
0133    WRITE(LU,30)
0134    WRITE(LU,30)
0135    WRITE(LU,30)
0136    30 FORMAT(4("dB	d@	"))/
0137    605 WRITE(LU,31)
0138    31 FORMAT(""
0139    WRITE(LU,32)
0140    32 FORMAT("<< SCROLL SCROLL >> CURSOR ",
0141    124X," NEW IMAGE EXIT ")
0142    610 CALL EXEC(1,LU,INPT,1)
0143    INPT = INPT-7023
0144    IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 610
0145    C
0146    C BRANCH TO APPROPRIATE SECTION
0147    C
0148    C
0149    GO TO (1000,2000,3000,4000,5000,6000,100,9000),INPT
0150    C
0151    C
0152    C
0153    C SCROLL IMAGE BACK
0154    C
0155    1000 IERR = SCROL(IDCBL,-9,NUML,IFRST,ILAST,RMAX,RMIN)
0156    IF (IERR .LT. 0) GO TO 991
0157    GO TO 610
C SCROLL FORWARD

2000   IERR = SCROL(IDCB,17,NUML,IFRST,ILAST,RMAX,RMIN)

IF (IERR .LT. 0) GO TO 991

GO TO 610

3000 CONTINUE

C POSITION CURSOR

4000   CALL EXEC(23,6HCURSR,LU)

GO TO 605

5000 CONTINUE

6000 CONTINUE

GO TO 610

C TERMINATE

9000   CALL CLOSE(IDCB)

CALL RESET(LU)

WRITE(LU,33)

33 FORMAT("END PROGRAM")

CALL EXEC(6)

C FILE NOT FOUND

800   WRITE(LU,3)

3 FORMAT(""

GO TO 105

C

991   CALL RESET(LU)

WRITE(LU,9) IERR

9 FORMAT("FILE ERROR",I6)

CALL CLOSE(IDCB)

END

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

0001   FIN4
0002     PROGRAM CURSR
0003     C
0004     DIMENSION LU(5),IBUF(2352),IZERO(2)
0005     C
0006     INTEGER EA,LA
0007     C
0008     DATA IZERO/44000B,64000B/
0009     C
0010     C
0011     CALL RMPAR(LU)
0012     C
0013     C
0014     C SAVE IMAGE LINES
0015     C
0016     C
0017     DO 50 I=0,20
0018     CALL RLINE(I,0,111,IBUF(112*I+1))
0019     50 CONTINUE
0020     WRITE(LU,1)
0021     1 FORMAT("
0022     WRITE(LU,2)
0023     2 FORMAT("   LEFT    UP    RIGHT
0024     :12X,"    DOWN    ",12X,"    RETURN "")
0025     CALL MOVEC(0,255)
0026     EA = 0
0027     LA = 255
0028     100 CALL EXEC(1,LU,INPT,1)
0029     INPT = INPT-7023
0030     IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 100
0031     C
0032     C
0033     C BRANCH TO APPROPRIATE SECTION
0034     C
0035     C
0036     GO TO (400,200,500,100,100,300,100,600), INPT
0037     C
0038     C MOVE CURSOR UP
0039     C
0040     200 LA = MOD(LA+11,256)
0041     CALL MOVEC(EA,LA)
0042     GO TO 100
0043     C
0044     C MOVE CURSOR DOWN
0045     C
0046     300 LA = MOD(LA+249,256)
0047     CALL MOVEC(EA,LA)
0048     GO TO 100
0049     C
0050     C MOVE CURSOR LEFT
0051     C
0052     400 EA = MOD(EA+499,512)
0053     CALL MOVEC(EA,LA)
0054     GO TO 100
0055     C
C MOVE CURSOR RIGHT
C
EA = MOD(EA+17,512)
CALL MOVEC(EA,LA)
GO TO 100
C RETURN TO PREVIOUS SCREEN
C
500 DO 610 I=0,20
610 CONTINUE
C
RETURN TO PREVIOUS SCREEN
C
END

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

FTN4
FUNCTION ICMPW(IBUF1,IBUF2,ILEN)
DIMENSION IBUF1(1),IBUF2(1)
DO 100 I=1,ILEN
IF (IBUF1(I) NE. IBUF2(I)) GO TO 200
100 CONTINUE
ICMPW = 0
RETURN
ICMPW = I
END

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

FTN4
SUBROUTINE RESET(LU)
WRITE(LU,1)
1 FORMAT(""
C
WAIT 200 MSEC
CALL EXEC(12,0,1,0,-20)
C CLEAR DISPLAY
WRITE(LU,2)
2 FORMAT("""
END
SUBROUTINE RLINE(LINE, IPIX, JPIX, IDATA)

THIS SUBROUTINE READS A LINE FROM GMR-27.

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)

C THIS SUBROUTINE MOVES THE CURSOR ON THE GMR-27. ITS POSITION IS INDICATED IN THE LOWER LEFT HAND CORNER OF THE SCREEN.

C

INTEGER WACO

DIMENSION ICR(7),IP00(5),IPXY(3)

EQUIVALENCE (ICR,ICR1),(ICR(2),ICR2),(ICR(3),ICR3),(ICR(5),I

1 (ICR(6),ICR6),(ICR(7),ICR7),(IPXY,IPXY1),(IPXY(2),IPXY2)

DATA IPO0/440008,64000B,24015B,50!`:''13,20002B/

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

DATA IPXY/0,0,24000B/

DATA WACO,LEA0,LLA0/22000B,44000B,64000B/

WRITE POSITION ON SCREEN

CALL DRIVR(2,IPO0,5)

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(LEA0,LAND(IX,777B))

IPXY2 = IOR(LLA0,LAND(IY,377B))

CALL DRIVR(2,IPXY,3)

RETURN

END
PROGRAM IMAGE

THIS PROGRAM IS THE IMAGE FILE MANAGER FOR THE IMAGE DISPLAY SUBSYSTEM.

DIMENSION LU(5), IDCBI(272), IDCB2(528), IDCB3(144), JNAME(3), IFNAM(3), NAME(6), IDATA(512), KNAME(6)

INTEGER ENTRY(256), ISIZE(2), TEXT1(19)

EQUIVALENCE (ENTRY(7), NLINE), (ENTRY(8), NPILX), (ENTRY(12), LOC)
1 (ENTRY(13), JNAME), (ENTRY(16), IFNAM), (ENTRY(19), IFNUM)
2, (ENTRY, KNAME)
2, (ENTRY(129))
EQUIVALENCE (ISIZE(2), ISIZ2)

GET INPUT PARAMETERS

CALL RMPAR(LU)

IF (LU .LE. 0) LU = 1

OUTPUT HEADING

WRITE(LU, 1)

FORMAT(/** IMAGE FILE MANAGER/**)

GET COMMAND INPUT

WRITE(LU, 2)

FORMAT("> ”)
READ(LU, 3) ICMD

EXECUTE COMMAND

IF (ICMD .NE. 2H??) GO TO 1010

COMMAND IS HELP

WRITE(LU, 4)

FORMAT(“/” COMMANDS ARE: “/,
1” BU-BUILD IMAGE FILE”/,
2” DI-DISPLAY IMAGE ON GMR-27”/,
3” SA-SAVE IMAGE TO TAPE”/,
4” RE-RESTORE IMAGE TO DISC”/,
4” DL-DIRECTORY LIST”/,
4” PU-PURGE IMAGE”/,
4” WT-WRITE NASA TAPE”/,
5” ??-HELP”/,
6” EX-EXIT”//)

GO TO 1000
19

0056 C
0057 1010 IF (ICMD .NE. 2HBU) GO TO 1030
0058 C
0059 C BUILD IMAGE COMMAND
0060 C
0061 CALL EXEC(23+10000OB,6HBLDIM ,LU)
0062 GO TO 1020
0063 5 GO TO 900
0064 C
0065 C PROGRAM NOT RP'ED
0066 C
0067 1020 WRITE(LU,6)
0068 6 FORMAT(" BLDIM NOT RP'ED!")
0069 GO TO 1000
0070 C
0071 1030 IF (ICMD .NE. 2HDI) GO TO 1045
0072 C
0073 C DISPLAY IMAGE COMMAND
0074 C
0075 CALL EXEC(23+10000OB,6HDSPLY ,LU)
0076 GO TO 1040
0077 7 GO TO 900
0078 C
0079 C DSPLY NO RP'ED
0080 C
0081 1040 WRITE(LU,8)
0082 8 FORMAT(" DSPLY NOT RP'ED!")
0083 GO TO 1000
0084 C
0085 1045 IF (ICMD .NE. 2HWT) GO TO 1050
0086 C
0087 C WRITE NASA TAPE
0088 C
0089 CALL EXEC(23,6HWTAPE ,LU)
0090 GO TO 1000
0091 C
0092 C
0093 1050 IF (ICMD .NE. 2HSA .AND. ICMD .NE. 2HRE .AND. ICMD .NE. 2HPU) GO TO 1200
0094 1 GO TO 1200
0095 C
0096 C SAVE/RESTORE IMAGE TO/FROM TAPE AND PURGE IMAGE
0097 C
0098 C OPEN DIRECTORY FILE
0099 C
0100 CALL OPEN(IDCBI,IERR,6HIMDIRC,2,2HIM,23,272)
0101 IF (IERR .LT. 0) GO TO 9999
0102 C
0103 C GET IMAGE NAME
0104 C
0105 WRITE(LU,9)
0106 9 FORMAT(" ENTER IMAGE NAME (12 CHARACTERS)? ")
0107 READ(LU,10) NAME
0108 10 FORMAT(6A2)
0109 C
0110 C FIND IMAGE
0111 C
0112 1060 CALL READF(IDCBI,IERR,ENTRY,256,LEN)
0113 IF (LEN .NE. -1) GO TO 1070
0114 C
0115 C EOF REACHED
0116 C
0117 WRITE(LU,11)
0118 11 FORMAT(" IMAGE NOT FOUND!")
0119 CALL CLOSE(IDCBI)
0120 GO TO 1000
0121 C
0122 1070 IF (IERR .LT. 0) GO TO 9999
0123 C
0124 C COMPARE NAME OF IMAGE
0125 C
0126 IF (ICMPW(ENTRY,NAME,6) .NE. 0) GO TO 1060
0127 C
0128 C IMAGE FOUND
0129 C
0130 IF (ICMD .EQ. 2HRE) GO TO 1120
0131 IF (ICMD .EQ. 2HPU) GO TO 1300
0132 C
0133 C TASK IS TO SAVE IMAGE
0134 C
0135 IF (LOC .EQ. 1) GO TO 1090
0136 C
0137 C IMAGE ALREADY ON TAPE
0138 C
0139 WRITE(LU,12)
0140 12 FORMAT(" IMAGE NOT ON DISC!")
0141 GO TO 1000
0142 C
0143 C IMAGE ON DISC
0144 C
0145 1090 CALL OPEN(IDCBI,IERR,JNAME,0,0,0,528)
0146 IF (IERR .LT. 0) GO TO 9999
0147 C
0148 C GET TYPE 0 FILE
0149 C
0150 C WRITE(LU,13)
0151 13 FORMAT(" TYPE MT LU 000# ?_")
0152 C READ(LU,14) IFNAM
0153 14 FORMAT(3A2)
0154 IFNAM=2HLU
0155 IFNAM(2) =2H00
0156 IFNAM(3) =2H08
0157 WRITE(LU,131)
0158 131 FORMAT(" SELECT OPTION"/
0159 READ(LU,*) IPACK
0160 CALL OPEN(IDCBI,IERR,IFNAM)
0161 IF (IERR .LT. 0) GO TO 9999
0162 CALL RNDF(IDCBI,IERR)
0163 IF (IERR .LT. 0) GO TO 9999
0164 WRITE(LU,15)
0165 15 FORMAT(" FILE "?_")
0166 READ(LU,*) IFNUM
0167 CALL SPACE(IDCBI,IERR,IFNUM-1)
0168 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)
1102    IDATA(K)aIOR(IDATA(J),KVAR)
1101    CALL WRITF(IDC33,IERR,IDATA,NPIXL)
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(IDC1,IER, -1)
IF (IER .LT. 0) GO TO 9999
CALL WRITF(IDC1,IER,ENTRY,256)
IF (IER .LT. 0) GO TO 9999

CALL CLOSE(IDC1)
CALL RWNDF(IDC33)
CALL CLOSE(IDC33)
GO TO 1000

RESTORE IMAGE FROM TAPE

IF (LOC .EQ. 2) GO TO 1130
IMAGE ON DISC
WHITE(LU,16)
FORMAT(" IMAGE ALREADY ON DISC")
CALL CLOSE(IDC1)
GO TO 1000
CREATE DISC FILE

ISIZE = (FLOAT(NLINE)*FLOAT(NPIXL)+127.)/128.
ISIZ2 = NPIXL

CALL CREAT(IDC82,IERR,JNAME,ISIZE,2,2HIM,23,328)
IF (IERR .GE. 0) GO TO 1135

CAN'T CREATE DISC FILE

WRITE(LU,19)
FORMAT(" CAN'T CREATE DISC FILE!!")
CALL CLOSE(IDC81)
GO TO 1000

OPEN TYPE 0 FILE

CALL OPEN(IDC83,IERR,IFNAM)

GET LU OF TYPE 0 FILE

CALL LOCF(IDC83,IERR,IREC,IRB,IOFF,JSEC,MTLU)
IF (IERR .LT. 0) GO TO 9999

TELL USER TO MOUNT TAPE
WRITE(LU,17) MTLU
FORMAT(" MOUNT TAPE ON LU ",I2" ENTER RETURN WHEN READY")
CALL EXEC(1,LU,IREC,1)

REWIND TAPE
CALL RWNDF(IDC83,IERR)
IF (IERR .LT. 0) GO TO 9999

SPACE FORWARD TO FILE
CALL SPACE(IDC83,IERR,IFNUM-1)

READ HEADER
CALL READF(IDC83,IERR,IDATA,11)
IF (IERR .LT. 0) GO TO 9999
IF(ICMPW(IDATA,ENTRY,11) .NE. 0) GO TO 1160

HEADF P COMPARES

TRANSFER DATA
DO 1140 I=1,NLINE
CALL READF(IDC83,IERR,IDATA,NPIXL)
IF (IERR .LT. 0) GO TO 9999
CALL WRITF(IDC82,IERR,IDATA,NPIXL)
IF (IERR .LT. 0) GO TO 9999
CONTINUE
0285   CALL RWNDF(IDC3)
0286   CALL CLOSE(IDC3)
0287   CALL CLOSE(IDC2)
0288 C
0289 C UPDATE DIRECTORY ENTRY
0290 C
0291   LOC = 1
0292 C
0293   CALL POSNT(IDC1,IERR,-1)
0294 IF (IERR .LT. 0) GO TO 9999
0295   CALL WRITF(IDC1,IERR,ENTRY,256)
0296 IF (IERR .LT. 0) GO TO 9999
0297   CALL CLOSE(IDC1,IERR)
0298 IF (IERR .LT. 0) GO TO 9999
0299 C
0300 GO TO 1000
0301 C
0302 C LABEL DOES NO MATCH
0303 C
0304 1160 WRITE(LU,18)
0305 18 FORMAT(" WRONG FILE!!")
0306   CALL RWNDF(IDC3)
0307   CALL CLOSE(IDC3)
0308   CALL CLOSE(IDC1)
0309 GO TO 1000
0310 C
0311 C
0312 1200 IF (ICMD .NE. 2HDL) GO TO 1230
0313 C
0314 C DIRECTORY LIST
0315 C
0316 C OPEN DIRECTORY FILE
0317 C
0318   CALL OPEN(IDC1,IERR,6H1MDIRC)
0319 IF (IERR .LT. 0) GO TO 9999
0320 C
0321 C OUTPUT HEADING
0322 C
0323   WRITE(LU,30)
0324 30 FORMAT("IMAGE NAME #LINES #PIXELS LOC TEXT")
0325 C
0326 C OUTPUT INFO
0327 C
0328 1210 CALL READF(IDC1,IERR,ENTRY,256,LEN)
0329 IF (LEN .NE. -1) GO TO 1220
0330 C
0331 C EOF REACHED
0332 C
0333   CALL CLOSE(IDC1)
0334 GO TO 1000
0335 C
0336 1220 IF (IERR .LT. 0) GO TO 9999
0337 C
0338 IF (ENTRY .EQ. -1) GO TO 1210
0339 ICHR = 2HD
0340 IF (LOC .NE. 1) ICHR = 2HT
0341 WRITE(LU,31)NAME,NLINE,NPIXL,ICHR,TEXT
0342 31 FORMAT(6A2,2X,I4,4X,I4,3X,A5,2X,19A2)
0343 GO TO 1210
C 0345 1230 IF (ICMD .EQ. 2'HEX) GO TO 1240
C 0346 C
C 0347 C ILLEGAL COMMAND
C 0348 C
C 0349 WRITE(LU,22)
C 0350 22 FORMAT("ILLEGAL COMMAND!")
C 0351 GO TO 1000
C 0352 C
C 0353 1240 WRITE(LU,23)
C 0354 23 FORMAT("END PROGRAM")
C 0355 CALL EXEC(6)
C 0356 C
C 0357 C PURGE FILE
C 0358 C
C 0359 1300 CALL POSNT(IDCBI,IERR,-1)
C 0360 IF (IERR .LT. 0) GO TO 9999
C 0361 ENTRY = -1
C 0362 CALL WRT6(IDCBI,IERR,ENTRY,256)
C 0363 IF (IERR .LT. 0) GO TO 9999
C 0364 C
C 0365 C PURGE DATA FILE
C 0366 C
C 0367 CALL PURGE(IDCBI2,IERR,JNAME,21M)
C 0368 CALL CLOSE(IDCBI)
C 0369 GO TO 1000
C 0370 C
C 0371 C ERROR
C 0372 C
C 0373 C
C 0374 C
C 0375 9999 WRITE(LU,20) IERR
C 0376 20 FORMAT(" FILE ERROR ",I6)
C 0377 CALL CLOSE(IDCBI)
C 0378 GO TO 1000
C 0379 END
C 0380 $
SUBROUTINE SPACE(IDCB,IERR,NUM)

THIS SUBROUTINE IS USED TO SPACE FORWARD OR BACKWARD THE NUMBER OF FILES SPECIFIED.

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
110 CONTINUE

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 AND
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),IRTM(5)
INTEGER WFINT,READL,RITEIL
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
IMXX1=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       IRTM=ICD
0069 C
0070       CALL BLANK(JMES)
0071    15 CALL SPCHR (IRT, 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 ITYPE =8
0078       INAX=255
0079       GO TO 32
0080    30 ITYPE =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
IF(IGET.LT.0) GO TO 999

480 IGET-RFAUL(-1,0,511,DIRC)
IF(IGET.LT.0) GO TO 999

40 CALL CODE
WRITE(JMES,45) AMAX,AMIN
45 FORMAT(" AMAX = ",E12.5," AMIN = ",E12.5,"FOR IMAGE")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)

SPECIFY IMAGE SCALING OPTION
50 CALL CODE
WRITE(JMES,51)
51 FOR .T-LAT( " SPECIFY IMAGE SCALING OPTION")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,52)
52 FORMAT( " 1. AUTOMATIC SCALING")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,53)
53 FORMAT( " 2. SYSTEM DEFAULT OPTION")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,54)
54 FORMAT( " 3. USER SPECIFIED SCALE FACTOR")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,56)
56 FORMAT( " 4. USER SPECIFIED MAX AND MIN")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,57)
57 FORMAT( " 5. LOG COMPRESSION")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,58)
58 FORMAT( " 6. EXPONENTIATION OPTION")
CALL TRMGN(JMES,LU,1,RTM,1CD,1RTM)
CALL BLANK(JMES)
55 CALL CKFLD(6,ICD,IRS)
GO TO (65,65,75,80,105,160,165,60),IRS
60 IW=2
GO TO 950

AUTOMATIC SCALING SELECTED
65 SCL=AMAX-AMIN
IF (ABS(SCL).LE.1.0E-5) GO TO 70
SCL=FLOAT(IMAX)/SCL
IOPT=1
GO TO '90

SYSTEM DEFAULT SCALING OPTION SELECTED
75 SCL=1.0
IOPT=2
GO TO '100
CALL CODE
WRITE(JMES,151)
FORMAT(" SCALE TOO SMALL")
CALL RMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,152)
FORMAT(" ENTER CR TO GO TO SYSTEM LEVEL MENU")
CALL RMGN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
IRTCD=1HX
GO TO 1000
SCL=(1.0/SCL)*FLOAT(IMAX)
GO TO 190

LOG COMPRESSION OPTION SELECTED

CALL CODE
WRITE(JMES,161)
FORMAT(" LOG COMPRESSION OPTION SELECTED")
CALL RMGN(JMES,LU,0)
CALL BLANK(JMES)
ILOPT=5
SCL=FLOAT(IMAX)/ALOG(AMAX-AMIN+1.0)
GO TO 190

EXPONENTIATION OPTION SELECTED

CALL CODE
WRITE(JMES,166)
FORMAT(" ENTER DESIRED EXPONENT")
CALL RMGN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
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
FORMAT(" EXPONENT= ",1F10.4)
CALL RMGN(JMES,LU,0)
CALL BLANK(JMES)
SCL=FLOAT(IMAX)/((AMAX-AMIN)**POWER)
ILOPT=6

OBTAIN PARAMETERS FOR RESIZING IMAGE

INUM=64
INCNT=0
IMCNT=0
NTST=INUM
CALL CODE
WRITE(JMES,196)
FORMAT(" INDEPENDENT DIRECTIONAL SCALING")
CALL RMGN(JMES,LU,0)
CALL BLANK(JMES)
USER ENTERS SCALE FACTOR

CALL CODE
WRITE(JMES,81)
FORMAT(" ENTER DESIRED SCALE FACTOR")
CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
CALL SPCHR (IRTM,IRT)
GO TO (1000,1000,1000,1000,855),IRT
GAIN=ABS(RTM)
IF (IRT .EQ. 5) GO TO 95
  95 CALL CODE
WRITE(JMES,100) GAIN
FORMAT(" SCALE FACTOR - ",F10.4)
CALL TRMGN(JMES,LU,0)
SCL=GAIN
GO TO 190

USER SPECIFIED MAXIMUM AND MINIMUM

IOPT=4
CALL CODE
WRITE(JMES,106)
FORMAT(" ENTER MAXIMUM FOR IMAGE")
CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
CALL SPCHR (IRTM,IRT)
GO TO (1000,1000,1000,1000,120),IRT
AMXIN=RTM
IF (RTM.NE.OB) GO TO 120
IW=4
GO TO 120
AMXIN=RTM
IF (RTM.NE.OB) GO TO 145
IW=5
GO TO 145
AMXIN=RTM
IF (RTM .EQ. 5) GO TO 155

CALL CODE
WRITE(JMES,131) AMXIN
FORMAT(" MAXIMUM FOR IMAGE = ",1PE15.8)
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL SPCHR (IRTM,IRT)
GO TO (1000,1000,1000,1000,935),IRT
AMNIN=RTM
IF (IRT .EQ. 5) GO TO 155
IW=5
GO TO 155
CALL CODE
WRITE(JMES,150) AMNIN
FORMAT(" MINIMUM FOR IMAGE =",F10.4)
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
SCL=AMXIN-AMNIN
IF (SCL.GE.1.0E-5) GO TO 155
0292 931 CALL CODE
0293 WRITE(JMES,197)
0294 197 FORMAT(" ENTER ROW SCALE FACTOR")
0295 CALL TMGN(JMES,LU,1,RTM,ICD,IRTM)
0296 CALL BLANK(JMES)
0297 200 CALL SPCHR (IRTM,IRT)
0298 YS*RTM
0299 IF (IRT .EQ. 5) GO TO 210
0300 205 IW=7
0301 GO TO 950
0302 210 CALL CODE
0303 WRITE(JMES,211)
0304 211 FORMAT(" ENTER COLUMN SCALE FACTOR ")
0305 CALL TMGN(JMES,LU,1,RTM,ICD,IRTM)
0306 CALL BLANK(JMES)
0307 215 CALL SPCHR (IRTM,IRT)
0308 XS*RTM
0309 IF (IRT .EQ. 5) GO TO 225
0310 220 IW=8
0311 GO TO 950
0312 C
0313 C CHECK TO SEE IF INTERPOLATION IS REQUIRED. IF NOT BRANCH.
0314 C
0315 C
0316 C COMPUTE NEW SIZE OF IMAGE
0317 C
0318 225 NNEW=YS*NROW+.5
0319 NCOLS=XS*ICOLS+.5
0320 C
0321 IF(XS.EQ.1.0.AND.YS.EQ.1.0) GO TO 260
0322 IF(NCOLS.LE.512) GO TO 230
0323 CALL CODE
0324 WRITE(JMES,690) NCOLS,XS
0325 690 FORMAT(" CALCULATED COLUMN SIZE = ",1I5,"(SF = ",1F5.2")")
0326 CALL TMGN(JMES,LU,0)
0327 CALL BLANK(JMES)
0328 CALL CODE
0329 WRITE (JMES,969)
0330 969 FORMAT (" 512 IS MAXIMUM ALLOWABLE OUTPUT")
0331 CALL TMGN (JMES,LU,0)
0332 CALL BLANK(JMES)
0333 CALL CODE
0334 WRITE(JMES,226)
0335 226 FORMAT(" REENTER COLUMN SCALE FACTOR")
0336 CALL TMGN(JMES,LU,1,RTM,ICD,IRTM)
0337 CALL BLANK(JMES)
0338 GO TO 215
0339 230 CALL CODE
0340 WRITE(JMES,695) NNEW,YS,NCOLS,XS
0341 695 FORMAT(" -- OUTPUT IMAGE--",14X,"ROWS = ",1I5,"(SF = ",1F5.2,"*)
0342 14X,"COLUMNS = ",1I5,": (SF = ",1F5.2,"*)")
0343 231 CALL TMGN(JMES,LU,0)
0344 CALL BLANK(JMES)
0345 CALL CODE
0346 WRITE(JMES,1232)
0347 1232 FORMAT(" 1. VALUES OKAY")
0348 CALL TMGN(JMES,LU,0)
0349 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)
232 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
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,O)
CALL BLANK(JMES)
WRITE(JMES,9750)
9750 FORMAT(" TO PREVENT ALIASING")
CALL TRMGN(JMES,LU,O)
CALL BLANK(JMES)
CALL CODE
WRITE(JMES,760)
760 FORMAT(" 1. CONTINUE ")
CALL TRMGN(JMES,LU,O)
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
SCHEDULE FILTER TO PREVENT ALIASING
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,O,NCOLS-1)
INITIZE FOR RESIZING
CALL CODE

WRITE(JMES,810)

FORMAT( " RESIZING OF IMAGE IN PROGRESS")

CALL TRMGN(JMES,LU,0)

CALL BLANK(JMES)

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

NN1=NN-1

IGET=READL(NN1,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*(ALOG(F(I)-AMIN+1.0))+0.5

GO TO 360

DO 355 I=1,ICOLS

G(I)=SCL*(F(I)-AMIN)**POWER+0.5
WRITE OUTPUT TO WORK FILE

360 IGET=RITEL(NNMI,0,ICOLS-1,G)
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	IOP(I)=MAXO(IOP(I),0)
365 IF(NN.LT.NTST) GO TO 400
NTST=NTST+INUM
CALL CODE
WRITE(JMES,375) NN,NNEW
375 FORMAT("- RESIZE ROWS DONE/ TO DO ",1I4,"/",1I4,"/")
CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)

400 IF(ITYPE.NE.8) GO TO 405
IGET=WLNE(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,
" - PERCENT CLIPPED AT MAX =",F6.2)
420 CALL TRMGN(JMES,LU,0)
CALL BLANK(JMES)
CALL CODE
CALL WRITE(JMES,421)
421 FORMAT(" 1. CONTINUE[ 2. RESCALE IMAGE"");
CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
CALL BLANK(JMES)
CALL SPCHR (IRTM,IRT)
GO TO (1000,1000,1000,1000,925),IRT
CALL CKFLD(2,ICD,IRS)
GO TO (1000,1000,440,430),IRS
IW=11
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)
END

&ROT8 T=00004 IS ON CR00322 USING 00002 BLKS R=0014

ASMB,R,L,C
NAM ROT8,6
ENT ROT8
EXT .ENTR
* WORD BSS 1
OUT BSS 1
*
ROT8 NOP
JSB .ENTR
DEF WORD
LDA WORD,I
ALF,ALF
STA OUT,I
JMP ROT8,I
END
SUBROUTINE TRMGN(JMES,LU,IP,RTM,ICD,IRTM)

C THIS SUBROUTINE IS USED TO WRITE OUT AND POSSIBLY READ BACK
C FROM THE TERMINAL INFORMATION NECESSARY FOR PROGRAM CONTROL.
C JMES IS THE MESSAGE TO BE OUTPUT TO THE LU. IP (IF =0) MEANS
C WRITE ONLY, (IF =1) MEANS TO WAIT FOR A RESPONSE FROM THE OPERAT
C RTM IS THE RETURN FOR REAL NUMBERS, ICD IS A RETURN FOR INTEGER
C IRTM IS THE RETURN FOR ASCII CHARACTERS. ALL THREE TYPES OF RES
C ARE GENERATED EACH TIME THIS SUBROUTINE IS CALLED. THE MAXIMUM
C OUTPUT MESSAGE IS 80 CHARACTERS LONG. THE MAXIMUM INPUT MESSAGE
C 10 CHARACTERS LONG.

DIMENSION JMES(40),IRTM(5)
ICN4ID=400B+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
END

SUBROUTINE XYFLT(U,V,FCX,FCY,NX,NY,N,A,B)

C WRITTEN BY W.E.ALEXANDER
C SUBROUTINE FOR A PART OF THE SPATIAL
C DOMAIN FILTERING PACKAGE.
C LOW PASS RECURSIVE FILTER DESIGN ROTINE
C WITH FCX NOT EQUAL TO FCY.
C FCX = RCX*S/PI WHERE RCX IS THE CUTOFF
C FCY = RCY*S/PI WHERE RCX IS THE CUTOFF
C THE Y DIRECTION AND S IS THE SAMPLING INTERVAL
C (Y DIRECTION)
C (0.010.LE.FCX.LE.0.950)
C (0.010.LE.FCY.LE.0.950)
C COMPLEX P
C DIMENSION U(3,3,2),V(3,3,2),A(3,2,2),B(3,2,2)
C
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
6 U(I)=0
7 V(I)=0
C
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

NNN=N-1

DO 10 J=1,NNN

DO 10 K = 1,2

IF(K.EQ.2) GO TO 20

CN = CNX

DD = DX

IF (NX.EQ.3) GO TO 22

THT = 135.0*PI/180.0

GO TO 23

IF(J.EQ.1)THT=125.5*PI/180.0

IF(J.EQ.2)THT=157.5*PI/180.0

GO TO 23

IF(J.EQ.1)THT=125.5*PI/180.0

IF(J.EQ.2)THT=157.5*PI/180.0

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 )

C

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

C

B(3,J,K) = S2

C

OBTAIN TWO DIMENSION FILTER

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

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

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

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)

RETURN

END

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 TRMCN(JMES,LU,0)

C

IMAX=512
IMXP=IMAX+1
ICM1=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.ICM1) GO TO 50
GO TO 15
E=(AINT(M-1)-AINT(M))*X+AINT(M)
F=(AINT(M+1)-AINT(M))*X+AINT(M+1)
FOP(I)=(F-E)*Y+E
I = I + 1
IF (I .LE. NCOLS) GO TO 15
IF (X .LT. 1.0 .AND. I .GT. NCOLS) GO TO 51

COMPLETE INTERPOLATION

FOP(NCOLS) = (AINT(ICOLS + IMAX) - AINT(ICOLS)) * Y + AINT(ICOLS)

CALL CODE
WRITE(JMES, 160)
FORMAT('NOW LEAVING INTRP')
CALL TRMGN(JMES, LU, 0)
RETURN
END

&LBRSZ T=00004 IS ON CR00022 USING 00010 BLKS R=0100

SUBROUTINE SPCHR (IATCD, IRT)
THIS ROUTINE CHECKS FOR SPECIAL CHARACTERS IN THE INPUT DATA
IRT = 5
IF (IATCD .EQ. OB) IRT = 0
IF ((IATCD .EQ. 1HX) .OR. (IATCD .EQ. 2HX)) IRT = 1
IF ((IATCD .EQ. 1HB) .OR. (IATCD .EQ. 2HB)) IRT = 2
IF ((IATCD .EQ. 1HD) .OR. (IATCD .EQ. 2HD)) IRT = 3
IF ((IATCD .EQ. 1HR) .OR. (IATCD .EQ. 2HR)) IRT = 4
RETURN
END

SUBROUTINE CKFLD (IA, ICD, IRS)
IRS = ICD + 1
IF (ICD = OB) IRS = 1
RETURN
END

SUBROUTINE INFRM (IA, LU)
DIMENSION IA(3)
ICNWD = 400B + LU
CALL EXEC (2, ICNWD, IA, 3)
RETURN
END

SUBROUTINE XFLTR (AINT, ICOLS, F, A, B, FCX, ITME)
DIMENSION B(3, 2)
DIMENSION F(1), AINT(1), A(3, 2), WF(3), WG(3)
IF (ITME .EQ. 0) CALL BOOST(0.0, 1.0, FCX, 2, A, B)
ITME = ITME + 1
A1 = A(1, 1)
A2 = A(2, 1)
A3 = A(3, 1)
B2 = B(2, 1)
B3 = B(3, 1)
40

0041 C INITIALIZE
0042 C
0043 IMAX=512
0044 INT=ICOLS/2-1
0045 IMXP1=IMAX+1
0046 ASTT=AINT(INT)+AINT(INT+1)+AINT(INT+2)
0047 ASTT=ASTT/3
0048 DO 10 I=1,3
0049 WF(I)=ASTT
0050 10 WG(I)=ASTT
0051 C
0052 C START FORWARD FILTER
0053 C
0054 MM=IMXP1
0055 WF(1)=AINT(IMXP1)
0056 20 WG(1)=AINT(IMAX)+WF(1)+A2*WF(2)+A3*WF(3)+B2*WG(2)+B3*WG(3)
0057 C
0058 C UPDATE
0059 C
0060 AINT(MM)=WG(1)
0061 WG(1)=WG(2)
0062 WG(2)=WG(1)
0063 WF(3)=WF(2)
0064 WF(2)=WF(1)
0065 MM=MM+1
0066 IF (MM.GT. ICOLS) GO TO 30
0067 WF(1)=AINT(MM)
0068 GO TO 20
0069 C
0070 C START REVERSE FILTER
0071 C
0072 30 ASTT=AINT(INT)
0073 DO 40 I=1,3
0074 WF(I)=ASTT
0075 40 WG(I)=ASTT
0076 C
0077 MM=IMAX+ICOLS
0078 WF(1)=AINT(MM)
0079 41 WG(1)=A1*WF(1)+A2*WF(2)+A3*WF(3)+B2*WG(2)+B3*WG(3)
0080 C
0081 C UPDATE
0082 C
0083 AINT(MM)=WG(1)
0084 WG(1)=WG(2)
0085 WG(2)=WG(1)
0086 WF(3)=WF(2)
0087 WF(2)=WF(1)
0088 MM=MM-1
0089 IF (MM .LE. IMAX) GO TO 50
0090 WF(1)=AINT(MM)
0091 GO TO 41
0092 50 RETURN
0093 END
0094 SUBROUTINE BLANK (JMES)
0095 DIMENSION JMES(40)
0096 DO 10 I=1,40
0097 10 JMES(I)=?H
0098 RETURN
0099 END
0100 END
**FUNCTION WFINT**

**INTEGER FUNCTION WFINT(NLINE,NPIXL,PMAX,PMIN,LU)**

**C THIS SUBROUTINE IS USED IN CONJUNCTION WITH IMAGE PROCESSING**

**C IT CREATES AND MAINTAINS AN IMAGE WORK FILE WITH PIXEL VALUES**

**C STORED AS REAL NUMBERS TO PRESERVE PRECISION.**

**C**

**C THIS ONE INITIALIZES THE PROCESS BY CREATING THE WORK FILE**

**C AND RETURNING CERTAIN PERTINENT INFO TO CALLER. IT SHOULD**

**C ONLY BE CALLED ONCE BY EACH CALLER. THE OTHER TWO ARE**

**C READL, WHICH READS A PARTICULAR LINE AND RITEL WHICH WRITES**

**C A PARTICULAR LINE.**

**C**

**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 IDCBI(144),IRTN(5),I3(6)**

**EQUIVALENCE (IB2,1B(2)),(IB(3),RMAX),(IB(5),RMIN)**

**CALL EXEC(23,6HBLDWF ,LU)**

**CALL RMPAR(IRTN)**

**WFINT = IRTN**

**IF (IRTN .LT. 0 ) RETURN**

**GET MAX MIN DATA**

**CALL OPEN(IDCBI,IERR,6HWF0000)**

**IF (IERR .LT. 0) GO TO 100**

**CALL READF(IDCBI,IERR,IB,6)**

**IF (IERR .LT. 0) GO TO 100**

**NLINE = IB**

**NPIXL = IB2**

**PMAX = RMAX**

**PMIN = RMIN**

**CALL CLOSE(IDCBI)**

**WFINT = 0**

**RETURN**

**GET MAX MIN DATA**

**CALL 100 WFINT = IERR**

**CALL CLOSE(IDCBI)**

**END**
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(IDCB,IERR,6HWF0000,2,0,0,528)

IF (IERR .LT. 0) GO TO 999

IFLAG = 1

FILE OPENED--READ APPROPRIATE LINE

CALL READF(IDCB,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
CALL OPEN(IDC0B, IERR, 6HWF0000, 2, 0, 0, 528)
IF (IERR .LT. 0) GO TO 999
IFLAG = 1

CALL READF(IDC0B, IERR, TBUF, 1024, LEN, LINE+2)
IF (IERR .LT. 0) GO TO 999

ISTEP = 1
IF (IPIXL .GT. JPIXL) ISTEP = -1
J = 1
DO 110 I=IPIXL+1,JPIXL+1,ISTEP
   TBUF(I) = RBUF(J)
   J = J+1
110
CALL WRITF(IDC0B, IERR, TBUF, 0, LINE+2)
IF (IERR .LT. 0) GO TO 999

RITEL = 0
RETURN

ERROR RETURN
RITEL = IERR
END

COMMON /CBLK/ IDCB(528), TBUF(512), IFLAG
DATA IFLAG/0/

SUBROUTINE CLSWF(NLINE, NPIXL, PMAX, PMIN)

COMMON /CBLK/ IDCB(528), TBUF(512), IFLAG
DATA IFLAG/0/

DIMENSION IB(6)
EQUIVALENCE (IB2, IB(2)), (RMAX, IB(3)), (RMIN, IB(5))

THIS ROUTINE IS USED TO CLOSE THE WORK FILE
WRITE DATA ON WORK FILE

IB = NLINE
IB2 = NPIXL
RMAX = PMAX
RMIN = PMIN

CALL WRITF(IDC0B, IERR, IB, 6, 1)
CALL CLOSE(IDC0B)
&DINTP T=00004 IS ON CRO0022 USING 00009 ELKS R=0087

0001  FTN4,Q,T,C
0002  PROGRAM DINTP
0003  C
0004  C THIS PROGRAM IS USED CHANGE THE PHYSICAL SIZE OF AN IMAGE
0005  C
0006  C WRITTEN BY WINSER E. ALEXANDER
0007  C
0008  DIMENSION AINT(1024),F(512),IPRAM(5),DIRC(515),INM(2)
0009  EQUIVALENCE (F(1),DIRC(4)),(DIRC(1),INM(1))
0010  EQUIVALENCE (INM(1),NROW),(INM(2),ICOLS),(DIRC(2),AMAX)
0011  EQUIVALENCE (DIRC(3),AMIN)
0012  C
0013  C IN
0014  PUT PARAMETERS (CALL RPMAN)
0015  C IPRAM(1) = LOGICAL UNIT FOR INTERACTIVE DEVICE
0016  C IPRAM(2) = NUMBER OF DESIRED ROWS IN OUTPUT IMAGE
0017  C IPRAM(3) = NUMBER OF DESIRED COLUMNS IN OUTPUT IMAGE
0018  C IMAGE TO BE USED IS ASSUMED TO BE IN IMAGE WORK FILE (WF0000
0019  C
0020  CALL RPMAN(IPRAM)
0021  L=IPRAM(1)
0022  IF(LU.LE.0) LU=1
0023  NEW=IPRAM(2)
0024  NCOLS=IPRAM(3)
0025  NCM1=NCOLS-1
0026  IMX=512
0027  IMXP1=IMX+1
0028  C
0029  C OBTAIN PARAMETERS FROM CURRENT IMAGE
0030  C
0031  IGET=READL(-1,0,511,DIRC)
0032  ICMI=ICOLS-1
0033  IF(IGET.LT.0) GO TO 999
0034  C
0035  C INTERPOLATE IMAGE
0036  C
0037  DY=FLOAT(NROW)/FLOAT(NNEW)
0038  DX=FLOAT(ICOLS)/FLOAT(NCOLS)
0039  IFLT=0
0040  IF(DY.GT.1.0) STOP 111
0041  IF(DX.LT.0.0) IFLT=1
0042  IFR=0
0043  IFE=0
0044  C
0045 C   INITIALIZE ARRAYS
0046 C
0047     IGET=READL(0,IFE,ICOLS-1,AINT(MXP1))
0048     IF(IGET.LT.0) GO TO 999
0049     IGET=READL(1,IFE,ICM1,AINT(1))
0050     IF(IGET.LT.0) GO TO 999
0051 C
0052     MCNT=2
0053     MORG=NROW
0054     DO 100 KK=NNEW,1,-1
0055 C
0056 C   COMPUTE Y
0057 C
0058     20 Y=(NNEW-KK)*DY-(NROW-MORG)
0059     IF(Y.LT.1.0) GO TO 50
0060 C
0061 C   BRING IN NEW ROW
0062 C
0063     CALL MOVE(AINT,ICOLS,IMXP1)
0064 C
0065     MCNT=MCNT+1
0066     IGET=0
0067     IF(MCNT.GT.NROW) IGET=-150
0068     IF(IGET.LT.0) GO TO 999
0069     IGET=READL(MCNT,IFE,ICM1,AINT)
0070     IF(IGET.LT.0) GO TO 999
0071     MORG=MORG-1
0072 C
0073 C   RECOMPUTE Y
0074 C
0075     GO TO 20
0076 C
0077 C   INTERPOLATE FOR NEW ROW
0078 C
0079     50 CALL INTRP(AINT,Y,DX,NCOLS,ICOLS,F)
0080 C
0081 C   OUTPUT CURRENT ROW
0082 C
0083     100 CALL RITEL(KK-1,O,NCKI,F)
0084 C
0085 C   NOTE THAT WORK FILE IS NOT CLOSED BY THIS PROGRAM
0086 C
0087 C   INSERT PARAMETERS IN WORK FILE
0088 C
0089     NROW=NNEW
0090     ICOLS=NCOLS
0091     CALL RITEL(-1,0,ICM1,DIRC)
0092     999 CONTINUE
0093 C
0094 C   ERROR PROCESSING
0095 C
0096     WRITE(LU,1000) IGET
0097 1000 FORMAT(" ERROR CODE = ",I5)
0098     CALL EXEC(6)
0099     END
0100 C
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

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 INVENTOPRY 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), PIMAX)

GET INPUT PARAMETERS

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

WRITE(LU,45)
READ(LU,46) IOPT
IF (IOPT .NE. 2HGO) GO TO 1000
WRITE(LU,51)
READ(LU,*') IFNUM

SPACE TO FILE POSITION

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)
55 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
CALL OPEN(IDCBI,IERR,6HIMCIRC,1,2HIM,23,272)
IF(IERR.LT.0) GO TO 999
FIND IMAGE FILE
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
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(IDCBI2,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 = ",1I8," . MINIMUM = ",1I8)
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 8?_")
C OUTPUT DATA TO TAPE

DO 80 I = 1,512
CALL FILL(IDATA,0,512)
IERR = 0
IF (I .LE. NLINE) CALL READF(IDC2,IERR,IDATA,512)
IF (IERR .LT. 0) GO TO 999
NOUT = 512
IF (ITYPE .EQ. 15) GO TO 70
C PACK DATA
NOUT = 256
C WRITE DATA
CALL EXEC(2,LU2,IDATA,NOUT)
80 CONTINUE
CALL EXEC(3,100B+LU2)
CALL CLOSE(IDC2)
GO TO 5
C FILE ERROR
WRITE(LU,996) IERR
996 FORMT(" FILE ERROR = ",1I4)
CALL CLOSE(IDC1)
CALL CLOSE(IDC2)
CALL EXEC(3,400B+LU2)
1000 CONTINUE
END

C SUBROUTINE FILL(IARAY,ICHAR,NUM)
DIMENSION IARAY(NUM)
DO 10 I = 1,NUM
IARAY(I) = ICHAR
10 RETURN
END$
PROGRAM PLOTV

DIMENSION LU(5)

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

DATA NAME/"HDA","HTA","H1"

CALL RMPA R(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

GO TO 20

CALL READF(IDCB,IERR,BUFF,LU)

IF (IERR .GE. 0) GOTO 40

WRITE(LU,30) IERR

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

GO TO 55

CONTINUE

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

GO TO 20

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 XXY(31), YYYY(31), XYP(31,2), LXY(15,3), AR(60)
DIMENSION XERR(31), CZ(9), IREG(2), U(3,3,2), V(3,3,2)
COMPLEX HA, HB, Z(25)
EQUIVALENCE (A(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))

CALL RMPAR(ILU)
LU-ILU;1)
MN-ILU(2)
AMAX = 0.0
AMIN =1000.0
MDIM = 30
NDIM = 30

CALL EXEC(14,1,IBUF,80)

IF(MN.EQ.3) GO TO 100
MNL=9
DO 10 J=1,3
DO 10 K=1,3
II=J+(K-1)*3
A(II)=U(J,K,1)
GO TO 10
10 B(II)=V(J,K,1)
GO TO 10

DO 103 I=1,5
DO 103 J=1,5
AA(I,J)=0.0
103 BB(I,J)=0.0
DO 102 I=1,3
DO 102 J=1,3
DO 102 K=1,3
DO 102 L=1,3
IK=I+K-1
JL=J+L-1
AA(IK,JL)=AA(IK,JL)+U(I,J,1)*U(K,L,2)
DO 102 I=1,5
DO 102 K=1,5
II=J+(K-1)*5
A(II)=AA(J,K)
10 B(II)=BB(J,K)
MNL=25
51

0058  101 WRITE(LU,1011)
0059  1011 FORMAT(21H COEFFICIENT MATRICES,/,)
0060    WRITE(LU,105) (A(I),I=1,25)
0061    WRITE(LU,105) (B(I),I=1,25)
0062    105 FORMAT(5(1H ,5ElO.2/)/)
0063  C   COMPUTE THE CENTER OF OUTPUT ARRAY
0064  C
0065  0066 WRITE(LU,12)
0067   12 FORMAT(" ENTER MX FOR HORIZONTAL FREQUENCIES"/)
0068    READ(LU,13)MX
0069    13 FORMAT(112)
0070    WRITE(LU,14)
0071   14 FORMAT(" ENTER MY FOR VERTICAL FREQUENCIES"/)
0072    READ(LU,13) MY
0073    203 MXC=MX/2
0074    MXT=2*MXC
0075      NX=0
0076      IF(MXT.NE.MX) NX=1
0077      MYC=MY/2
0078      MYT=2*MYC
0079      NY=0
0080      IF(MYT.NE.MY) NY=1
0081    0081 MXN=MXC+1
0082    MYN=MYC+1
0083   0083 WRITE(LU,301)
0084    300 FORMAT(" COMPUTE SQUARED MAGNITUDE "/)
0085    301 FORMAT(" INITAILIZE ARRAY")
0086  C   COMPUTE SQUARED MAGNITUDE CHARACTERISTIC
0087  C
0088  0089 MX=MXT+NX
0089      MY=MYT+NY
0090      FCX=2.0/FLOAT(MX)
0091      FCY=2.0/FLOAT(MY)
0092      IF(MX.LE.101.AND.MY.LE.61) GO TO 204
0093      IF(MX.LE.101) GO TO 202
0094      MX=101
0095    0096 WRITE(LU,200)
0097    200 FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR HORIZONTAL ")
0098    202 IF(MY.LE.61) GO TO 203
0099      MY=61
0100  0100 WRITE(LU,201)
0101  0101 GO TO 203
0102    201 FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR VERTICAL "/)
0103    204 WRITE(LU,300)
0104  0104 DO 20 I=1,MX+1
0105    20 DO J=1,MY+1
0106    0106 XF=FCX*(I-MXC-1)
0107      XXX(I)=XF
0108    0108 YF=FCY*(J-MYC-1)
0109      YYY(J)=YF
0110  0110 CALL ZWC(Z,XF,YF,MN)
0111    0111 HA=CMPLX(0.0,0.0)
0112    0112 HB=HA
0113    0113 DO 21 K=1,MNL
0114    0114 HA=HA+A(K)*Z(K)
0115    0115 HB=HB+B(K)*Z(K)
CONTINUE
XA=CABS(HA)
XB=CABS(HB)
IF(XB.LE.1.0E-20) XB=1.0E-20
XA=XA/XB
XM(I,J)=XA**2
IF(XM(I,J).LT.AMIN) AMIN=XM(I,J)
IF(XM(I,J).GT.AMAX) AMAX=XM(I,J)
20 CONTINUE
WRITE(LU,302)AMAX,AMIN
302 FORMAT(" AMAX = ",1E10.2,3X,"AMIN = ",1E10.2/) C
SQUARED MAGNITUDE NORMALIZED
C
C OBTAIN W-1 PLOT FROM ARRAY
C
DO 22 I=1,NX
XYP(I,2)=V1(I+MXC,MYN)
XERR(I)=XYP(I,2)
22 XYP(I,1)=FCX*(I-1)
WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,MXN)
306 FORMAT(///1H ,6(1E10.2,3X)/)
XL=0.0
XU=1.0
MC=2
C
C CALCULATE Z=W PLOT
C
X=(MXN)**2+(MYN)**2
FCX=0.7071*FCX
NUM=SQR(T(X)+1)
DO 30 I=1,NX
XF=FCX*(I-1)
CALL ZWC(Z,XF,XF,MN)
HA=CMPLX(0.0,0.0)
HB=HA
DO 31 X=1,MNL
HA=HA+A(K)*Z(K)
31 HB=HB+B(K)*Z(K)
XA=CABS(HA)
XB=CABS(HB)
IF(XB.LE.1.0E-20) XB=1.0E-20
XA=XA/XB
XYP(I,2)=XA**2
XYP(I,1)=XF*1.414
WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,MXN)
COMPUTE ERROR FUNCTION

ERR = 0.0

DO 350 J = 1, MDIM
ERR = ERR + ((XERR(J) - XYP(J, 2)) / AMAX)**2
WRITE(LU, 360) ERR
360 FORMAT( "RELATIVE ERROR = ", 1E15.7/)

COMPUTE CONTOURS

CALL CONTR(XXX, YYY, AMAX, AMIN, MX+1, MY+1, LU)

STOP

END

SUBROUTINE ZWC(Z, XF, YF, MN)

C THIS SUBROUTINE COMPUTES COMPLEX VALUES FOR Z**I*W**j FOR ZW TRANSFORM AND PLACES RESULTS IN ONE DIMENSIONAL ARRAY Z
C XF = HORIZONTAL RELATIVE FREQUENCY
C YF = VERTICAL RELATIVE FREQUENCY

COMPLEX Z(25), R, S
IF(ABS(KF).EQ.1.0) XF = 0.99
IF(ABS(YF).EQ.1.0) YF = 0.99
PI = 3.1415926
RX = COS(PI*XF)
RY = SIN(PI*XF)
SX = COS(PI*YF)
SY = SIN(PI*YF)
R = CMPLX(RX, RY)
S = CMPLX(SX, SY)
IF(MN.GE.3) GO TO 20
DO 10 J = 1, 3
DO 10 K = 1, 3
I = J + (K-1)*3
10 Z(I) = S**((J-1)*R**((K-1))
GO TO 22
20 DO 21 J = 1, 5
21 DO 21 K = 1, 5
22 RETURN
END

BLOCK DATA WORK
COMMON/WORK/WO(130)
COMMON/CNT/XM(900)
COMMON/QDCAZ/IQ(40)
END
SUBROUTINE CONT(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 HAME9/2HDA,2HTA,2H1

WRITE(LU,100)

100 FORMAT(" SELECT TYPE OF FILTER PLOT ")

READ(LU,*) IFLAG

C	 GENERATE CZ

CZ(1)=-1.

DO 3 K=2,9

CZ(K)=CZ(K-1)+.25

3 CONTINUE

C CREATE A PLOT DATA FILE

ITYPE=3

ISIZE(1)=96

CALL PURGE(IDC,1,ERR,NAME9)

IF(ERR.LT.0) WRITE(LU,101) IERR

CALL CREAT(IDC,1,ERR,NAME9,ISIZE,ITYPE)

IF(ERR.GE.0) GO TO 201

WRITE(LU,101) IERR

101 FORMAT("CREATE ERROR",F5.0)

STOP

DO 3-D PLOTS

201 IF (IFLAG.EQ.1) GO TO 10

20 CONTINUE

CALL SET3D(1.,-1.,1.,-1.,AMAX,AMIN,2,0,.5,.5)

CALL PLT3D(XXX,YYY,XM,30,MX,MY,LU)

IF(IFLAG.EQ.2) GOTO 30

DO ISOGAMS

10 CONTINUE

DO 11 I=1,MX

11 CONTINUE

DO 11 J=1,MY

XM(I,J)=XM(I,J)/AMAX

11 CONTINUE

CALL SET2D(1.-1.,1.-1.,AMAX,AMIN,2,0,.5,.5)

CALL PLT2D(XXX,YYY,XM,30,MX,MY,CZ,9,LU)

CONTINUE

CALL CLOSE(IDC)

RETURN

END
SUBROUTINE SET2D(ALPMAX, ALPMIN, BETMAX, BETMIN, IORGN, LALPCL, AL
COMMON/ QDCAZ/ IXXYB(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
ZLALP=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
IF (IALPCL.EQ.1) GO TO 30

AY=0.
BX=0.
AX=ELALP/(ALPMAX-ALPMIN)
BY=ELBET/(BETMAX-BETMIN)
XZ=XZ-.5*ELALP
YZ=YZ-.5*ELALP
GO TO 5

CONTINUE

AY=0.
BY=0.
AX=ELALP/(ALPMAX-ALPMIN)
BX=ELBET/(BETMAX-BETMIN)
XZ=XZ-.5*ELBET
YZ=YZ+.5*ELALP
GO TO 5

CONTINUE

IF (IALPCL.EQ.1) GO TO 40

AY=0.
BY=0.
AX=ELALP/(ALPMAX-ALPMIN)
BY=ELBET/(BETMAX-BETMIN)
XZ=XZ-.5*ELALP
YZ=YZ+.5*ELALP
GO TO 5

CONTINUE

AY=0.
BY=0.
AX=ELALP/(ALPMAX-ALPMIN)
BY=ELBET/(BETMAX-BETMIN)
XZ=XZ-.5*ELALP
YZ=YZ+.5*ELALP
GO TO 5

CONTINUE

IZ=IXZ-AX*ALPMIN-BX*BETMIN
IZ=IZ-AY*ALPMIN-BY*BETMIN
IYYXYB(1,1)=IFIX(XZ+AX*ALPMIN+BX*BETMIN)
IYYXYB(2,1)=IFIX(YZ+AY*ALPMIN+BY*BETMIN)
IYYXYB(1,2)=IFIX(XZ+AX*ALPMIN+BX*BETMAX)
IYYXYB(2,2)=IFIX(YZ+AY*ALPMIN+BY*BETMAX)
IYYXYB(1,3)=IFIX(XZ+AX*ALPMAX+BX*BETMAX)
IYYXYB(2,3)=IFIX(YZ+AY*ALPMAX+BY*BETMAX)
IYYXYB(1,4)=IFIX(XZ+AX*ALPMAX+BX*BETMIN)
IYYXYB(2,4)=IFIX(YZ+AY*ALPMAX+BY*BETMIN)
IYYXYB(3,1)=IYYXYB(1,2)
IYYXYB(4,1)=IYYXYB(2,2)
IYYXYB(3,2)=IYYXYB(1,3)
IYYXYB(4,2)=IYYXYB(2,3)
IYYXYB(3,3)=IYYXYB(1,4)
IYYXYB(4,3)=IYYXYB(2,4)
IYYXYB(3,4)=IYYXYB(1,1)
IYYXYB(4,4)=IYYXYB(2,1)
RETURN
END
SUBROUTINE PLT2D(ALPHA, BETA, GAMMA, IDMN, IALPHA, JBETA, C, NUMC
1
COMMON/QDCAZ/IXYXYB(4, 4), XZ, AX, BX, YZ, AY, BY
DIMENSION ALPHA(1), BETA(1), GAMMA(IDMN, 1), C(1)
INTEGER BUFF(4), NAME9(3)
COMMON IDCB(144)
COMMON/WORK/IXIY(2, 62), JXJY(2, 62)
DATA NAME9/2HDA, 2HTA, 2H1 /
CALL OPEN(IDC, IERR, NAME9)
NOGRID = 0
IF(NUMC.LE.0) GO TO 1
IF (IALPHA) 2, 1, 3
2 CONTINUE
NOGRID = 1
3 CONTINUE
IMAX = IABS(IALPHA);
IMAX2 = IMAX + 2
4 CONTINUE
IF (JBETA) 4, 1, 5
5 CONTINUE
JMAX = IABS(JBETA)
IF (NOGRID.EQ.1) GO TO 6
DO 7 K = 1, 4
CALL F6BUF(IXYXYB(1, K), IXYXYB(2, K),
IXYXYB(3, K), IXYXYB(4, K), BUFF)
CALL WRITF(IDC, IERR, BUFF, 4)
6 CONTINUE
DO 8 N = 1, NUMC
DO 9 I = 1, IMAX + 2
IXIY(I, I) = 0
JXJY(I, I) = 0
9 CONTINUE
DO 10 J = 1, JMAX
10 IF(J.EQ.1) GO TO 11
DO 12 I = 1, IMAX
12 IF(GAMMA(I, J).EQ.GAMMA(I, J-1)) GO TO 13
13 CONTINUE
JXJY(I, I+1) = 0
GO TO 12
14 CONTINUE
BETINT = BETA(J-1) + (BETA(J) - BETA(J-1))*C(N) - GAMMA(I, J-1)) / (GA
13)
ALPINT = ALPHA(I)
IXR = IFIX(XZ + AX * ALPINT + BX * BETINT)
14 CONTINUE
IYR=IFIX(YZ+AY*ALPINT+AY*BETINT)

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

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

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

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)

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(IDCBI,IERR,BUFF,4)
CONTINUE

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

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

RETURN

END
SUBROUTINE SET3D(ALPMAX, ALPMIN, BETMAX, BETMIN, GAMMAX, GAMMIN, 
ORGN, IALPCL, GAMFAC, ALPFAC)
COMMON/QDCAZ, IXXYB(4, 5), XZ, AX, BX, YZ, AY, BY, CY
DATA ELX, ELY, EXLEFT, YBOTOM /1012.1, 856.1, 12., 156./
ALX = ALPFAC*ELX/(ALPMAX - ALPMIN)
AY = ALPFAC*(1. - GAMFAC)*(GAMMAX - ALPMAX)
BX = (1. - ALPFAC)*ELX/(BETMAX - BETMIN)
BY = (1. - ALPFAC)*(1. - GAMFAC)*ELY/(BETMAX - BETMIN)
CY = GAMFAC*ELY/(GAMMAX - GAMMIN)
YZ = - CY*GAMMIN + YBOTOM
XZ = EXLEFT
IF(ORGN.EQ.1) GO TO 1
IF(ORGN.EQ.2) GO TO 2
IF(ORGN.EQ.3) GO TO 3
GO TO 4
1 CONTINUE
XZ = XZ + ELX
AX = - AX
BX = - BX
IF(IALPCL.EQ.1) GO TO 10
YZ = YZ + BY*(BETMAX - BETMIN)
BY = - BY
ALPVRT = ALPMAX
BETVRT = BETMAX
GO TO 5
10 CONTINUE
YZ = YZ + AY*(ALPMAX - ALPMIN)
AY = - AY
ALPVRT = ALPMIN
BETVRT = BETMAX
GO TO 5
2 CONTINUE
ALPVRT = ALPMAX
BETVRT = BETMAX
IF(IALPCL.EQ.1) GO TO 20
XZ = XZ + BX*(BETMAX - BETMIN)
BX = - BX
GO TO 5
20 CONTINUE
XZ = XZ + AX*(ALPMAX - ALPMIN)
AX = - AX
GO TO 5
3 CONTINUE
IF(ALPCL.EQ.1)GO TO 30
0313 YZ=YZ+AY*(ALPMAX-ALPMIN)
0314 AY=-AY
0315 ALPVRT=ALPMIN
0316 BETVRT=BETMAX
0317 GO TO 5
0318 30 CONTINUE
0319 YZ=YZ+AY*(ALPMAX-ALPMIN)
0320 AY=-AY
0321 ALPVRT=ALPMAX
0322 BETVRT=BETMIN
0323 GO TO 5
0324 4 CONTINUE
0325 YZ=YZ+BY*(BETMAX-BETMIN)
0326 ALPVRT=ALPMIN
0327 BETVRT=BETMIN
0328 AY=-AY
0329 BY=-BY
0330 IF(ALPCL.EQ.1)GO TO 40
0331 XZ=XZ+AX*(ALPMAX-ALPMIN)
0332 AX=-AX
0333 GO TO 5
0334 40 CONTINUE
0335 XZ=XZ+BX*(BETMAX-BETMIN)
0336 BX=-BX
0337 5 CONTINUE
0338 XZ=XZ-BX*BETMIN-AX*ALPMIN
0339 YZ=YZ-BY*BETMIN-AY*ALPMIN
0340 IXYXYB(1,1)=XZ+AX*ALPMAX+BX*BETMIN
0341 IXYXYB(2,1)=YZ+AY*ALPMAX+BY*BETMIN+CY*GAMMIN
0342 IXYXYB(3,1)=XZ+AX*ALPMAX+BX*BETMIN
0343 IXYXYB(4,1)=YZ+AY*ALPMAX+BY*BETMIN+CY*GAMMIN
0344 IXYXYB(1,2)=IXYXYB(3,1)
0345 IXYXYB(2,2)=IXYXYB(4,1)
0346 IXYXYB(3,2)=XZ+AX*ALPMAX+BX*BETMAX
0347 IXYXYB(4,2)=YZ+AY*ALPMAX+BY*BETMAX+CY*GAMMIN
0348 IXYXYB(1,3)=IXYXYB(3,2)
0349 IXYXYB(2,3)=IXYXYB(4,2)
0350 IXYXYB(3,3)=XZ+AX*ALPMAX+BX*BETMAX
0351 IXYXYB(4,3)=YZ+AY*ALPMAX+BY*BETMAX+CY*GAMMIN
0352 IXYXYB(1,4)=IXYXYB(3,3)
0353 IXYXYB(2,4)=IXYXYB(4,3)
0354 IXYXYB(3,4)=IXYXYB(1,1)
0355 IXYXYB(4,4)=IXYXYB(2,1)
0356 IXYXYB(1,5)=XZ+AX*ALPVRT+BX*BETVRT
0357 IXYXYB(2,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMIN
0358 IXYXYB(3,5)=IXYXYB(1,5)
0359 IXYXYB(4,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMAX
0360 45 FORMAT (5(7X,17))
0361 RETURN
0362 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/QLCA:/IXYXYB(4, 5), XZ, AX, BX, YZ, AY, BY, CY
INTEGER BUFF(4)
COMMON IDC8(144)
NOGRID=0
IF(I2, CF, L3, 1, 2)
1 CONTINUE
NOGRID=1
2 CONTINUE
L3 JMAX=ABS(I2, )
3 CONTINUE
L4 IF(J2, CF, L5, 2, 5)
4 CONTINUE
L5 JMAX=ABS(I2, )
CONTINUE
L6 QUIP IQ=1, 5
DO K=1, 5
CALL FLBUF(IXYXYB(K), IXYXYB(K), IXYXYB(K), IXYXYB(K), BUFF)
CALL WRITF(IDC8, IERR, BUFF, 4)
CONTINUE
DO I=1, IMAX
IXR=IFIX(XZ+AX*ALPHA(I)+BX*BETA(J))
IYR=IFIX(YZ+AY*ALPHA(I)+BY*BETA(J)+CY*GAMMA(I, J))
IF(I2, EQ, L1)GO TO 9
CALL FLBUF(IXR, IYR, LASTXY(1, I-1), LASTXY(2, I-1), BUFF)
CALL WRITF(IDC8, IERR, BUFF, 4)
CONTINUE
IF(J2, EQ, L1)GO TO 10
CALL FLBUF(IXR, IYR, LASTXY(1, I), LASTXY(2, I), BUFF)
CALL WRITF(IDC8, IERR, BUFF, 4)
CONTINUE
LASTXY(1, I)=IXR
LASTXY(2, I)=IYR
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
&FDIGN T=00004 IS ON CR00022 USING 00056 BLKS R=0498

0001  FTN4,L
0002  PROGRAM FDIGN
0003  C
0004  C THIS PROGRAM SCHEDULE FILTER DESIGN,STABILITY,AND DISPLAY
0005  C
0006  COMMON WORK/WO(75)
0007  DIMENSION NAME1(3),NAME2(3),NAME3(3),IRTN(5)
0008  DIMENSION IDCB(144),NAME4(3),NAME5(3)
0009  DIMENSION U(3,3,2),V(3,3,2),ILU(5),IBUF(80),IBUF2(80)
0010  EQUIVALENCE (ILU(1),LU),(IBUF,IBUF2)
0011  EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
0012  DATA NAME1/2HST,2HAB,2HI /
0013  DATA NAME2/2HDP,2HLA,2HM /
0014  DATA NAME3/2HCO,2HEF,2HFS /
0015  DATA NAME4/2HFI,2HRO,2H /
0016  DATA NAME5/2HPL,2HOT,2HV /
0017  DATA V/18*0./
0018  DATA U/18*0./
0019  DATA IBUF/80*0/
0020  C
0021  C GET LU
0022  CALL RMPAR(ILU)
0023  C
0024  C GET FILTER PARAMETERS
0025  C
0026  MN=1
0027  4 WRITE(LU,400)
0028  400 FORMAT(" SELECT FILTER DESIGN"/
0029  1. LOWPASS"/
0030  2. BANDPA
0031  3. HIGHPASS"/
0032  4. BOOST FILTER"/
0033  5. TDLPF (LOWPASS)"
0034  6. ROTATING FILTER "/
0035  7. NON-RECURSIVE FILTERS ")
0036  READ(LU,401) IFIL
0037  IF(IFIL.EQ.4) GO TO 500
0038  IF(IFIL.EQ.3) GO TO 410
0039  IF(IFIL.EQ.6) GO TO 408
0040  IF(IFIL .EQ. 7) GO TO 1102
0041  WRITE(LU,402)
0042  402 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS")/
0043  READ(LU,403) F2
0044  403 FORMAT(F2.2)
0045  IF(IFIL.NE.2) GO TO 407
0046  WRITE(LU,404)
0047  404 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR HIGHPASS")/
0048  READ(LU,403) F1
0049  407 WRITE(LU,405)
0050  405 FORMAT(" ENTER NUMBER OF FILTER STAGES")/
0051  READ(LU,401) MN
0052  IBUF(40) =MN
0053  401 FORMAT(111)
0054  GO TO 411
0055  500 WRITE(LU,510)
0056  510 FORMAT(" SELECT OPTION"/
0057  1. LOW BOOST FILTER"/
0058  2. HI
0059  3. FILTER")/
0060  READ(LU,515) IOPT
0061  515 FORMAT(111)
0062  IF(IOPT.GE.0.AND.IOPT.LE.2) GO TO 530
63

WRITE(LU,520)
0057  520 FORMAT(" INVALID RESPONSE")
0058       GO TO 500
0059 C
0060  530 WRITE(LU,535)
0061  535 FORMAT(" ENTER BOOST MAGNITUDE")
0062      READ(LU,*) BF
0063      WRITE(LU,540)
0064  540 FORMAT(" ENTER RELATIVE BREAK FREQUENCY")
0065      READ(LU,403) F1
0066 IF(IOPT.EQ.0.OR.IOPT.EQ.1) WRITE(LU,545) BF,F1
0067  545 FORMAT(" BOOST MAGNITUDE = ",1E15.5," FREQUENCY = ",1F10.5," *BOOST FILTER."/" IS THIS CORRECT?")
0068 IF(IOPT.EQ.2) WRITE(LU,550) BF,F1
0069  550 FORMAT(" BOOST MAGNITUDE = ",1E15.5," BREAK FREQUENCY = ",1F
0070     * FOR HIGH BOOST FILTER."/" IS THIS CORRECT")
0071      READ(LU,551) IRES
0072  551 FORMAT(1A1)
0073      IF(IRES.EQ.1HY.OR.IRES.EQ.1Hy) GO TO 552
0074      GO TO 530
0075  552 BF=SQRT(ABS(BF))
0076 IF(IOPT.EQ.2) GO TO 560
0077  560 ALP=1.0
0078      BET=BF-1.0
0079      GO TO 412
0080  561 ALP=BF
0081      BET=1.0-BF
0082      GO TO 412
0083  410 WRITE(LU,404)
0084      READ(LU,403) F1
0085  412 WRITE(LU,405)
0086      READ(LU,401) MN
0087  411 MN=MN+1
0088  408 IF(IFIL.EQ.1) CALL LPFLT(U,V,F2,MN,LU)
0089 IF(IFIL.EQ.2) CALL BPFLT(U,V,F1,F2,MN,LU)
0090 IF(IFIL.EQ.3) CALL BSTFT(U,V,F1,F2,MN,1.0,-1.0,LU)
0091 IF(IFIL.EQ.4) CALL BSTFT(U,V,F1,F2,MN,ALP,BET,LU)
0092 IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
0093 IL.E CALL BPFLT(U,V,F1,F2,MN,LU)
0094 IF(IFIL.EQ.3) CALL BSTFT(U,V,F1,MN,1.0,-1.0,LU)
0095 IF(IFIL.EQ.4) CALL BSTFT(U,V,F1,MN,ALP,BET,LU)
0096 IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
0097 IF(IFIL.EQ.6) CALL ROTA(E(U,V,MN,LU)
0098 C    IF(IFIL .EQ. 7) CALL FIR(U,MN,WR,LU)
0099 CONTINUE
0100 ILU(2) = MN
0101 IF(IFIL.EQ.2) ILU(2) =MN + 1
0102 IF(IFIL.EQ.6) ILU(2) =MN-1
0103 C  SCHEDULE STABILITY TEST-STABT
0104 C  CALL EXEC(23,NAME1,LU,ILU(2),0,0,0,IBUF,80)
0105 C  SCHEDULE DISPLAY PROGRAM-DPLAY
0106 C  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(IDC, IERR, NAME3, 2HES)
IF(IERR .LT. 0) WRITE(LU, 1101) IERR
CALL CREATE(IDC, IERR, NAME3, 2, 3, 2HES)
IF(IERR .LT. 0) WRITE(LU, 1101) IERR
CALL WRITF(IDC, IERR, IBUF, 80)
CALL CLOSE(IDC, IERR)
WRITE(LU, 1105)
FORMAT("ENTER DISPLAY DEVICE "// 1. TV"/' 2. HP2648A")
READ(LU, *) IDEV
IF(IDEV .EQ. 2) GO TO 1106
CALL EXEC(23, NAME5, LU, 0, 0, 0, 0)
GO TO 1107
1106 CONTINUE
GILL HP48A(LU)
1107 CONTINUE
GALL EXEC(6)
C
SCHEDULE NON-RECURSIVE FILTERS
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 = AMIN1(F1, F2)
SUBROUTINE BPFLT(U, V, F1, F2, N, LU)
WRITE(LU, *) IDEV
F1----BREAK FREQUENCY FOR LOW FREQUENCY CUTOFF
F2----BREAK FREQUENCY FOR HIGH FREQUENCY CUTOFF
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 = AMIN1(F1, F2)
CALL BSTFT(AA, BB, FC, N, 1.0, -1.0, LU)
SUBROUTINE BSTFT(U,V,FC,N,ALP,BET,LU)

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
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*A**2
IF(DEM.LT.D) GO TO 12
13 P1=(+2.0*A-2.0*SQRT(2.0)*SALP+1.0)/DEM
GO TO 20
12 F=-1.0*DEM
13 IF(F.GT.D) GO TO 13
P1=0.0
20 A=((1.0+P1)**2)/4.0
AS=A**2
P0=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(R,D)
S=((1.0-P1)**4)/R
U(1,1,1) = S*(ALP*POS+BET*AS)
U(1,2,1) = S*(ALP*P1*(1.0+POS)+2.0*BET*AS)
U(2,1,1) = U(1,2,1)
U(2,2,1) = S*(ALP*(1.0+POS)+2.0*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*P1*(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*P1
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*P1*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)
V(I,J,2) = V(I,J,1)
26 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
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=SQRT(ALP)
DEM=1.0-2.0*ALP
IF(DEM.LT.D) GO TO 12
13 P1=(+2.0*ALP-2.0*SQRT(2.0)*SALP+1.0)/DEM
0282 P2=P1
0283 IF(FC.GT.0.3)GO TO 10
0284 P2=(SQRT(T)-EPS)/(SQRT(T)+EPS)
0285 GO TO 20
0286 12 *=1.0*DEM
0287 IF(F.GT.D) GO TO 13
0288 P1=0.0
0289 P2=0.0
0290 20 V(1,1,1)=1.0
0291 V(1,2,1)=P1+P2
0292 V(2,1,1)=V(1,2,1)
0293 V(1,3,1)=P1*P2
0294 V(3,1,1)=V(1,3,1)
0295 V(2,2,1)=V(1,2,1)**2
0296 V(2,3,1)=P1*P2**2+P2*P1**2
0297 V(3,2,1)=V(2,3,1)
0298 V(3,3,1)=V(1,3,1)**2
0299 C
0300 SUM=0.0
0301 DO 25 I=1,3
0302 DO 25 J=1,3
0303 25 SUM=SUM+V(I,J,1)
0304 C
0305 U(1,1,1)=SUM/16.0
0306 U(1,3,1)=U(1,1,1)
0307 U(3,1,1)=U(1,1,1)
0308 U(3,3,1)=U(1,1,1)
0309 U(1,2,1)=SUM/8.0
0310 U(2,1,1)=U(1,2,1)
0311 U(2,3,1)=U(1,2,1)
0312 U(3,2,1)=U(1,2,1)
0313 U(2,2,1)=SUM/4.0
0314 IF(N.EQ.2) GO TO 1
0315 DO 26 I=1,3
0316 DO 26 J=1,3
0317 U(I,J,2)=U(I,J,1)
0318 26 V(I,J,2)=V(I,J,1)
0319 GO TO 2
0320 1 U(1,1,2)=1.0
0321 V(1,1,2)=1.0
0322 2 DO 30 I=1,5
0323 DO 30 J=1,5
0324 A(I,J)=0.0
0325 30 B(I,J)=0.0
0326 DO 31 I=1,5
0327 DO 31 J=1,5
0328 DO 31 K=1,3
0329 DO 31 L=1,3
0330 IK=I-K+1
0331 JL=J-L+1
0332 IF(IK.LE.0.OR.IK.GT.3)GO TO 31
0333 IF(JL.LE.0.OR.JL.GT.3)GO TO 31
0334 A(I,J)=A(I,J)+U(IK,JL,1)*U(K,L,2)
0335 B(I,J)=B(I,J)+V(IK,JL,1)*V(K,L,2)
0336 31 CONTINUE
0337 RETURN
0338 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(PI.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*CSSQRT(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(“,112,”) = “,1E15.5,” + J”,1E15.5/
L=L+1
Z2=(2.0+PK-2.0*CSSQRT(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 IF((L-1).EQ.NN) GO TO 25
C PAIR COMPLEX PAIRS OF ROOTS

C 25 L=1
D 30 K=1,NN
S1=IMAG(P(K))
IF(S1.LT.0.0) GO TO 30
P(L)=P(K)
L=L+1
30 CONTINUE

C OBTAIN FILTER COEFFICIENTS

C 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

C 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**2
B(2,3,K)=C1*C2
B(3,2,K)=C1*C2
40 B(3,3,K)=C2**2
GO TO 600

300 WRITE(1,310)
310 FORMAT(" NUMBER OF STAGES TOO LARGE FOR DIMENSION")
GO TO 600

400 WRITE(1,410)
410 FORMAT(14" FREQUENCY SPECIFICATION OUT OF RANGE")
GO TO 600

500 WRITE(1,510)
510 FORMAT(" NUMBER OF ROOTS LESS THAN EXPECTED")
600 RETURN
END
SUBROUTINE HP48A(J)
DIMENSION IB(14), IA(4)
INTEGER IDCB(144), BUFF(4), NAME(3)
DATA NAME/'2HDA, 2HTA, 2H1/
C
CALL OPEN(IDCB, IERR, NAME)
IF(IERR .GE. 0) GO TO 30
WRITE(LU, 10) IERR
10 JRMAT ('OPEN ERROR', F5.0)
STOP
30 CALL GRAFC(1, LU)
20 CALL READF(IDCB, IERR, BUFF, 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 DVCCT(BUFF, BUFF(2), BUFF(3), BUFF(4), LU)
50 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(IDCB)
RETURN
END
SUBROUTINE GRAFC(IFLAG, LU)
INTEGER IESC
IESC= 33B
C GRAPHICS OFF=0; GRAPHICS ON NOT=0
IF(IFLAG .EQ. 0) GO TO 100
C GRAPHIC ON
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

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

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 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",113,1'!'",113,"Z")
RETURN
END$
&BLDIM  T=00004  IS ON CR00022 USING 00034 BLKS R=0330

0001 FTN4
0002 PROGRAM BLDIM
0003 C
0004 C THIS PROGRAM BUILDS AN IMAGE FILE FOR THE NCA&T IMAGE DISPLAY
0005 C SYSTEM. IMAGE FILES MAY BE GENERATED FROM THE GMR-27 DISPLAY,
0006 C TAPES OR DISC (TYPE 2 FILES).
0007 C
0008 C PROGRAMMER: DLJ
0009 C
0010 DIMENSION LU(5),IDCB1(272),IDCB2(528),NAME(6),ISIZE(2),IDATA
0011 DIMENSION JNAME(3),IBUF(6)
0012 C
0013 INTEGER ENTRY(256),TEXT1(40),TEXT2(40),TEXT3(40),RDREC
0014 C
0015 EQUIVALENCE (ENTRY,NNAME),(ENTRY(7),NLINE),(ENTRY(8),NP1XL),
0016 1 (ENTRY(9),NPIN),(ENTRY(10),IPMAX),(ENTRY(11),ISRC),
0017 2 (ENTRY(13),NAME),(ENTRY(129),TEXT1),(ENTRY(169),TEXT2),
0018 3 (ENTRY(209),TEXT3),(ENTRY(12),ILOC)
0019 EQUIVALENCE (JNAME(2),JNAM2),(JNAME(3),JNAM3),(ISIZE(2),ISIZ
0020 C
0021 C CONSTANTS
0022 C  MPIXL = MAXIMUM PIXELS/LINE (WHEN CHANGING BE SURE TO MODIF
0023 C  ARRAY SIZES)
0024 C
0025 DATA MPIXL/512/
0026 C
0027 C GET INPUT PARAMETERS
0028 C
0029 CALL RMPAR(LU)
0030 IF (LU .LE. 0) LU = 1
0031 C
0032 C OUTPUT HEADING
0033 C
0034 WRITE(LU,1)
0035 1 FORMAT(" B U I L D  I M A G E  S U B  S Y S T E M"
0036 C
0037 C OPEN DIRECTORY FILE
0038 C
0039 CALL OPEN(ID?B1,IERR,6HIMDIRC,0,2HIM,23,272)
0040 IF (IERR .LT. 0) GO TO 9999
0041 C
0042 C GET IMAGE NAME
0043 C
0044 1000 WRITE(LU,2)
0045 2 FORMAT("ENTER 12 CHARACTER IMAGE NAME?(/E TO EXIT)_")
0046 READ(LU," NAME
0047 3 FORMAT(6A2)
0048 IF (NAME .EQ. 2H/E) GO TO 1060
0049 C
0050 C CHECK FOR DUPLICATE NAME
0051 C
IREC = 0
KREC = 0
CALL RNDF(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
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
DUPLICATE NAME FOUND
C
WRITE(LU,4)
4 FORMAT("ERROR-DUPLICATE NAME")
CALL RNDF(IDCBI,IERR)
IF (IERR .LT. 0) GO TO 9999
GO TO 1000
C
EOF REACHED AND NO DUPLICATE FOUND
C
GET IMAGE PARAMETERS
C
1030 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
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
GET SOURCE OF IMAGE
C
1040 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
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 DCODE(KREC,JNAME,JNAME)
0114 CALL PURGE(IDC2,IERR,JNAME,2HIM,23)
0115 CALL CREATE(IDC2,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 IPMIN = 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(IDC2,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(IDC2)
0137 C
0138 C WRITE DIRECTORY ENTRY
0139 C
0140 IF (KREC .EQ. IREC) GO TO 1055
0141 CALL OPEN(IDC1,IERR,6HINDIR,2,2HIM,23,272)
0142 IF (IERR .LT. 0) GO TO 9999
0143 CALL POSNT(IDC1,IERR,KREC)
0144 IF (IERR .LT. 0) GO TO 9999
0145 1055 ILOC = 1
0146 CALL WRITF(IDC1,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(IDC1)
0153 CALL EXEC(6)
0154 C
0155 C ERROR
0156 C
0157 9999 WRITE(LU,9IERR
0158 9 FORMAT(" FILE ERROR-",I6)
0159 CALL CLOSE(IDC1)
0160 END
INTEGER FUNCTION RDREC(ICODE, IBUF, IP1, IP2)

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

DIMENSION IBUF(1), IDATA(1024), NAME(3), RDATA(512), IDCBO(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)
1 FORMAT("ENTER DISC FILE NAME?")
READ(LU,2) NAME
2 FORMAT(3A2)

OPEN FILE

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

IF (IERR .LT. 0) GO TO 999

WRITE(LU,3)
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
IBCBO = 1
RETURN

100 IF (IBUF .NE. 2) GO TO 110

TAPE INPUT

WRITE(LU,4)
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,6HWO000,0,0,0,1040)
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(IDCDB,IERR,RDATA,NUM)
IF (IERR .LT. 0) GO TO 999
IFMT=3
GO TO 160
C TAPE INPUT
C
CALL EXEC(1,MTLU,IDATA,NUM)
GO TO 160
C DISPLAY INPUT
C
IBUF = 0
CALL MVW(IBUF,IBUF(2),NPIXL-1)
IF ((ISTEP .GT. 0) .AND. (ISTRTL .GT. IENDL)) RETURN
IF (ISTEP .LT. 0 .AND. ISTRTL .LT. IENDL) RETURN
CALL RLINF(ISTRTL,ISTRTP,IENDP,IDATA)
ISTRTL = ISTRTL + ISTEP

MOVE DATA TO OUTPUT ARRAY AND UNPACK IF NECESSARY
IF (.NOT. PACKED) GO TO 180
DATA IN PACKED FORMAT
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 = ITEKP
IF (ITEMP .LT. IP2) IP2 = ITEKP
IBUF(2*I-1) = JTEMP
IBUF(2*I) = ITEKP
RETURN

DATA IS UNPACKED
DO 190 I=1,NPIXL
ITEMP = IDATA(I)
IF (IFMT .EQ. 3) ITEKP = RDATA(I)
IF (ITEMP .GT. IP1) IP1 = ITEKP
IF (ITEMP .LT. IP2) IP2 = ITEKP
IBUF(I) = ITEKP
RETURN
C
C RDREC = IERR
END
$
&LFLTR T=00003 IS ON CR00022 USING 00024 BLKS R=0000

0001  FTN4, 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,WFINT
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 = I.5UF(40)
0038  N = NSTAG + 1
0039  CALL CLOSE(IDCB,IERR)
0040  C
0041  C GET CONTROL BLOCK INFORMATION
0042  C
0043  IERR=WFINT(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
INITIALIZE FILTER TO MID-LINE-CLS AVG

NMID=NR0W/2
CNST=0.0
IERR=READL(NMID,0,511,F1)
IF(IERR .LT. 0) GO TO 9999
DO 110 I=1,ICOLS
CNST=CNST+F1(I)
CNST=CNST/FLOAT(ICOLS)
DO 13 I=1,524
F3(I)=CNST
F2(I)=CNST
F1(I)=CNST

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
DO 11 J=1,3
SUM(NSTG,1)=SUM(NSTG,1)+A(I,J,NSTG-1)
SU(I,NSTG,2)=SUM(NSTG,2)+B(I,J,NSTG-1)
DEL=ABS(SUM(NSTG,2))
IF(DEL.LT.1.0E-20)CALL EXEC(2,LU,16FILTER UNSTABLE,8)
SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)

CALCULATE INITIAL CONDITIONS FOR EACH STAGE

SUM(1,2)=CNST
DO 12 NSTG=2,N
SUM(NSTG,2)=SUM(NSTG,2)*SUM(NSTG-1,2)

INITIALIZE FILTER

DO 14 I=1,524
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
RM1=1.0E38

FILTER REVERSE

IERR=READL(8,IPXL,JPIXL,F3)
IF(IERR .LT. 0) GO TO 9999
IERR=READL(7,IPXL,JPIXL,F2)
IF(IERR .LT. 0) GO TO 9999
IERR=READL(6,IPXL,JPIXL,F1)
IF(IERR .LT. 0) GO TO 9999
C
C REINITIALIZE FILTER
0139
C
0140
DO 15 II=1,524
0141
F1(II) = CONST
0142
F2(II) = CONST
0143
F3(II) = CONST
0144
15 CONTINUE
0145
C
0146
FILTER FORWARD
0147
C
0148
RMX=-0.1E38
0149
RMI= 0.1E38
0150
LINE =NROW-9
0151
IERR=READL(LINE,IPIXL,PIXL,F3(12))
0152
IF(IERR .LT. 0) GO TO 9999
0153
LINE=LINE+1
0154
IERR=READL(LINE,IPIXL,PIXL,F2(12))
0155
IF(IERR .LT. 0) GO TO 9999
0156
LINE=LINE+1
0157
IERR=READL(LINE,IPIXL,PIXL,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,RMX,RMI)
0164 LNCK = LNCK + 1
0165 LINE = (NROW - 1) - IABS(NRO + 1)
0166 IF(IERR .LT. 0) GO TO 9999
0167 CALL FILTR(1,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0168 IF(LNCK .LT. 0) GO TO 402
0169 CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,1,LU,RMX,RMI)
0170 LNCK = LNCK + 1
0171 LINE = (NROW - 1) - IABS(NRO + 2)
0172 IF(IERR .LT. 0) GO TO 9999
0173 CALL FILTR(1,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0174 IF(LNCK .LT. 0) GO TO 403
0175 CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,1,LU,RMX,RMI)
0176 LNCK = LNCK + 1
0177 LINE = (NROW - 1) - IABS(NRO + 3)
0178 IF(IERR .LT. 0) GO TO 9999
0179 CALL FILTR(1,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0180 IF(LNCK .LT. 0) GO TO 400
0181 CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,1,LU,RMX,RMI)
0182 CONTINUE
0183 C
0184 51 CONTINUE
0185 C
0186 51 CONTINUE
0187 RMX = RMX
0188 RMIN = RMI
0189 CALL CLSWF(NROW,ICOLS,RMAX,RMIN)
0190 CALL PRTN(IRTN)
0191 CALL EXEC(6)
0192 CALL EXEC(2,LU,16HREAD FILE ERROR ,8)
0193 END
0194 C
0195 EQUIVALENCE (IBUF,A),(IBUF(41),B)
0196 IFLAG = 1 FOR FORWARD FILTERING, = 2 FOR REVERSE
0197 C
0198 C
0199 C
0200 IF(IFLAG .EQ. 1) GO TO 200
0201 DO 20 I = 1,11
0202 L = ICOLS + I2 - I
0203 J = ICOLS - 12 + I
0204 F1(L) = F1(J)
0205 F2(L) = F2(J)
0206 F3(L) = F3(J)
0207 C
0208 C
0209 C
0210 C
0211 C
0212 C
0213 DO 10 M = ICOLS+9,1,-1
0214 K = M +2
0215 X1(M) = A(1) * F1(:)
0216 1 + A(2) * F1(J)-B(2)*X1(J)
0217 1 + A(3) * F1(K)-B(3)*X1(K)
0218 1 + A(4) * F2(M)-B(4)*X2(M)
0219 1 + A(5) * F2(J)-B(5)*X2(J)
0220 1 + A(6) * F2(K)-B(6)*X2(K)
0221 1 + A(7) * F3(M)-B(7)*X3(M)
0222 1 + A(8) * F3(J)-B(8)*X3(J)
0223 1 + A(9) * F3(K)-B(9)*X3(K)
0224 \( \text{IF(NSTAG .EQ.1) GO TO 10} \)
0225 \( \text{C G1(M) = A(10) * X1(M)} \)
0226 \( \text{C \quad 1 + A(11) * X1(J)-B(11)*G1(J)} \)
0227 \( \text{C \quad 1 + A(12) * X1(K)-B(12)*G1(K)} \)
0228 \( \text{C \quad 1 + A(13) * X2(M)-B(13)*G2(M)} \)
0229 \( \text{C \quad 1 + A(14) * X2(J)-B(14)*G2(J)} \)
0230 \( \text{C \quad 1 + A(15) * X2(K)-B(15)*G2(K)} \)
0231 \( \text{C \quad 1 + A(16) * X3(M)-B(16)*G3(M)} \)
0232 \( \text{C \quad 1 + A(17) * X3(J)-B(17)*G3(J)} \)
0233 \( \text{C \quad 1 + A(18) * X3(K)-B(18)*G3(K)} \)
0234 \( \text{10 \quad \text{CONTINUE}} \)
0235 \( \text{10 \quad \text{GO TO 400}} \)
0236 \( \text{200 \quad \text{CONTINUE}} \)
0237 \( \text{C \quad G1(M) = A(10) * X1(M)} \)
0238 \( \text{C \quad 1 + A(11) * X1(J)-B(11)*G1(J)} \)
0239 \( \text{C \quad 1 + A(12) * X1(K)-B(12)*G1(K)} \)
0240 \( \text{C \quad 1 + A(13) * X2(M)-B(13)*G2(M)} \)
0241 \( \text{C \quad 1 + A(14) * X2(J)-B(14)*G2(J)} \)
0242 \( \text{C \quad 1 + A(15) * X2(K)-B(15)*G2(K)} \)
0243 \( \text{C \quad 1 + A(16) * X3(M)-B(16)*G3(M)} \)
0244 \( \text{C \quad 1 + A(17) * X3(J)-B(17)*G3(J)} \)
0245 \( \text{C \quad 1 + A(18) * X3(K)-B(18)*G3(K)} \)
0246 \( \text{40 \quad \text{CONTINUE}} \)
0247 \( \text{400 \quad \text{CONTINUE}} \)
0248 \( \text{RETURN} \)
0249 \( \text{END} \)
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, RMX, 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))
12 IF(IERR .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(ITEMP .LT. 0) ITEMP=0
120 IX1(I) = ITEMP
GO TO 200
CONTINUE
IERR= RITEL(LINE,IPIXL,JPIXL,G1(IFL))
IF(IERR .LT. 0) GO TO 9999
DO 121 I=1,524
ITEMP=G1(I) + 0.5
IF(ITEMP .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
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)
EQUIVALENCE(IBUF(1),A(1,1,1)),(IBUF(41),B(1,1,1))
EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
EQUIVALENCE(F1,R1),(F2,IF2),(R1,IF1),(IF1,ILINE),(IF1(2),IC0
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
IPIXL = ILU(3)
IF(IPIXL .EQ. 0) IPIXL = 511
CALL OPEN(IDCB,IERR,NAME)
IF(IERR .LT. 0) GO TO 9999
CALL READF(IDCB,IERR,IBUF,80,IERR)
IF(IERR .LT. 0) GOTO 9999
IPIXL=2
ICOLS=ICOLS-2
JPIXL = IPIXL - 1
DO 110 I=ICOLS
110 CNST=CNST+AMAXO(Fl(I),1)
CNST=(CNST/FLOAT(ICOLS))
CNST = ALOG(CNST)

CALL BIAS(F1,RMIN,ICOLS)
DO 701 I=1,ICOLS
701 CNST=CNST+AMAXO(F1(I),1)
CNST=(CNST/FLOAT(ICOLS))
CNET = ALOG(CNST)

GFT FILTER COEFF'S
CALL OPEN(IDCB,IERR,NAME)
IF(IERR .LT. 0) GO TO 9999
CALL READDF(IDCB,IERR,IBUF,80,IERR)
IF(IERR .LT. 0) GOTO 9999
N=NSTAG+1
CALL CLOSE(IDCB,IERR)

GET CONTROL BLOCK INFORMATION
IERR=1:FINTE(NROW,ICOLS,RMAX,RMIN,LU)
IF(IERR .LT. 0)GOTO 9999
IPIXL = 2
ICOLS=ICOLS-2
JPIXL = ICOLS - 1
INITIALIZE FILTER TO MID LINE-COL AVG
SMID=NROW/2
CNST=0.0
IERR=READL(NMID,IPIXL,JPIXL,F1)
IF(IERR .LT. 0) GO TO 9999
CALL BIAS(F1,RMIN,ICOLS)
DO 701 I=1,ICOLS
701 CNST=CNST+AMAXO(F1(I),1)
CNST=(CNST/FLOAT(ICOLS))
CNET = ALOG(CNST)
DO 9 I=1,523
F1(I) = CNST
F2(I) = CNST
F3(I) = CNST
CONTINUE

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)
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 CALCULATE INITIAL CONDITIONS FOR EACH STAGE
SUM(1,2)=CNST
DO 12 NSTG=2,N
SUM(NSTG,2)=SUM(NSTG, 1)*SUM(NSTG-1, 2)

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


```
0106        LNCK = 1
0107        DO 300 NRO=-6,NROW - 1,3
0108        CALL BIAS(F1,RMIN,ICOLS)
0109        CALL HFLIP(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,IPIXL,JPIXL,X1,G1,NSTAG,2,LU,RMX,RMI,SCL)
0113        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 BIAS(F3,RMIN,ICOLS)
0119        CALL HFLIP(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0120        IF(LNCK .LT. 7) GO TO 302
0121        CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,2,LU,RMX,RMI,SCL)
0122        LNCK = LNCK +1
0123        LINE = IABS(NRO+2)
0124        IF(LINE .GT. NROW-1) GO TO 300
0125        IERR = READL(LINE,IPIXL,JPIXL,F2)
0126        IF(IERR .LT. 0) GO TO 9999
0127        CALL BIAS(F2,RMIN,ICOLS)
0128        CALL HFLIP(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,IPIXL,JPIXL,X2,G1,NSTAG,2,LU,RMX,RMI,SCL)
0132        LNCK = LNCK +1
0133        LINE = IABS(NRO+3)
0134        IF(LINE .GT. NROW-1) GO TO 300
0135        IERR = READL(LINE,IPIXL,JPIXL,F1)
0136        IF(IERR .LT. 0) GO TO 9999
0137        CONTINUE
0138        C
0139        C REINITIALIZE FILTER
0140        CONST = (RMX-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 /(RMX)
0153        RMI = 0.1E38
0154        RMX = -0.1E38
0155        JPIXL = JPIXL-1
0156        LINE = NROW-9
0157        IERR = READL(LINE,IPIXL,JPIXL,F3(12))
0158        IF(IERR .LT. 0) GO TO 9999
0159        LINE = LINE+1
0160        IERR = READL(LINE,IPIXL,JPIXL,F2(12))
0161        IF(IERR .LT. 0) GO TO 9999
0162        LINE = LINE+1
0163        IERR = READL(LINE,IPIXL,JPIXL,F1(12))
0164        IF(IERR .LT. 0) GO TO 9999
```
LNCK = 6
DO 400 NRO= -6,NROW = 1,3
CALL HFILT(1,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
IF(LNCK .LT. 0) GO TO 401
CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,1,LU,RMX,RMI,SCL)
401 LNCK = LNCK + 1
LINE = (NROW - 1) - IABS(NRO + 1)
IF(LINE .LT. 0) GO TO 400
CALL RITLN(LINE,IPIXL,JPIXL,F3(12))
IF(LNCK .LT. 0) GO TO 402
CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,1,LU,RMX,RMI,SCL)
402 LNCK = LNCK + 1
LINE = (NROW - 1) - IABS(NRO + 2)
IF(LINE .LT. 0) GO TO 400
CALL RITLN(LINE,IPIXL,JPIXL,F2(12))
IF(LNCK .LT. 0) GO TO 403
CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,1,LU,RMX,RMI,SCL)
403 LNCK = LNCK + 1
LINE = (NROW - 1) - IABS(NRO + 3)
IF(LINE .LT. 0) GO TO 400
CALL RITLN(LINE,IPIXL,JPIXL,F1(12))
IF(LNCK .LT. 0) GO TO 9999
CONTINUE
CALL EXEC(2,LU,10HCOMPLETED,5)
CALL CLSWF(NROW,ICOLS,RMX,RMI)
RMAX = RMX
RMN = RMI
CALL PRTN(IRTN)
CALL EXEC(6)
CALL EXEC(2,LU,16HREAD FILE ERROR,8)
END
SUBROUTINE HFILT(IFLAG,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
DIMENSION F1(1),F2(1),F3(1),X1(1),X2(1),X3(1),A(1),B(1)
COMMON /IBLK/IBUF(80)
DIMENSION G1(1),G2(1),G3(1)
EQUIVALENCE (IBUF,A),(IBUF(41),B)
IFLAG = 1 FOR FORWARD FILTERING, = 2 FOR REVERSE
REVERSE FILTERING
IF(IFLAG .EQ. 1) GO TO 200
DO 200 I=1,11
L = ICOLS + 12 - I
J = ICOLS - 12 + I
F1(L) = F1(J)
F2(L) = F2(J)
F3(L) = F3(J)
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)
1     + A(3) * ALOG(F1(K)) - B(3) * X1(K)
1     + A(4) * ALOG(F2(M)) - B(4) * X2(M)
1     + A(5) * ALOG(F2(J)) - B(5) * X2(J)
1     + A(6) * ALOG(F2(K)) - B(6) * X2(K)
1     + A(7) * ALOG(F3(M)) - B(7) * X3(M)
1     + A(8) * ALOG(F3(J)) - B(8) * X3(J)
1     + A(9) * ALOG(F3(K)) - B(9) * X3(K)
IF(NSTAG .EQ. 1) GO TO 10
G1(M) = A(10) * X1(M)
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
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 CONTINUE
C
DO 40 M = 3, ICOLS + 11
J = M - 1
K = M - 2
X1(M) = A(1) * F1(M)
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 40
G1(M) = A(10) * X1(M)
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)
40 CONTINUE
400 CONTINUE
RETURN
END
COMMON BLOCK SUBPROGRAM

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

SUBROUTINE RITLN(LINE, IPIXL, JPIXL, XI, G1, NSTAG, IFLAG, LU, RMX, RSLC)

DIMENSION XI(1), G1(1), XX1(523)
INTEGER RTEL

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)
CONTINUE
IERR = RTEL(LINE, IPIXL, JPIXL, XI)
IF(IERR .LT. 0) GO TO 9999
GO TO 12

CONTINUE

11 CONTINUE
C FORWARD
DO 20 M = 2, JPIXL - IPIXL + 1
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)
CONTINUE
IERR = RTEL(LINE, IPIXL, JPIXL, XX1(12))
IF(IERR .LT. 0) GO TO 9999
CONTINUE
RETURN

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

SUBROUTINE BIAS(F1, RMIN, ICOLS)
DIMENSION F1(1)
DO 10 I = 1, ICOLS + 1
F1(I) = F1(I) - RMIN + 1.0
IF(F1(I) .LT. 1.0) F1(I) = 1.0
CONTINUE
RETURN
END
DATA SHOW T-00004 IS ON CRO0022 USING 00005 BLKS R-0037

0001 FIN4
0002 c PROGRAM SHOW
0003 C
0004 D DIMENSION RDATA(512), IDATA(512), LU(5)
0005 C
0006 C INTEGER READL
0007 E EQUIVALENCE (RDATA, LU(2)), (LU(2), ILINE), (LU(3), IPIXL),
0008 1 (RDATA(2), RMAX), (RDATA(3), RMIN)
0009 C
0010 C GET INPUT PARAMETERS
0012 C
0013 C GET SCALE
0016 C 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 = IPIXL
0026 PMAX = RMAX
0027 PMIN = 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) .GE. 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 CLSWF(NLINE, NPIXL, PMAX, PMIN)
0039 CALL EXEC(6)
0040 999 WRITE(LU, 2) IERR
0041 2 FORMAT("FILE ERROR", I7)
0042 END
0043 $
&FIRO  T=00004 IS ON CR00022 USING 00003 BLKS R=0023

0001  FTN4,L
0002  PROGRAM FIRO
0003  DIMENSION ILU(5),IBUF(80),A(3,3,2),H(5,5),NAME(3),IDCB(144)
0004  DIMENSION NAME1(3),NAME2(3)
0005  EQUIVALENCE (IBUF(1),A(1,1,1))
0006  DATA H/25*0./
0007  DATA IBUF/80*0/
0008  DATA NAME/2HCO,2HEF,2HFS/
0009  DATA NAME1/2HDP,2HLA,2ML;
0010  DATA NAME2/2HPL,2HJF,2HV /

0011  C
0012  C GET LU
0013  call rmpar(ILU)
0014  LU=ILU
0015  WRITE(LU,10)
0016  10 FORMAT(" ENTER NUMBER OF STAGES ")
0017  READ(LU,*) NSTG
0018  IBUF(40)=NSTG
0019  C
0020  WRITE(LU,11)
0021  11 FORMAT(" ENTER ALPHA VALUE ")
0022  READ(LU,*) ALPHA
0023  C
0024  H(1,1)=1.0
0025  DO 100 I=1,3
0026  DO 100 J=1,3
0027  CALL WINDO(ALPHA,I,J,WIN)
0028  A(I,J,NSTG)=WIN*H(I,J)
0029  100 CONTINUE
0030  CALL PURGE(IDCB,IERR,NAME,2HES)
0031  IF(IERR .LT. 0) WRITE(LU,999) IERR
0032  CALL CREATE(IDCB,IERR,NAME,2,3,2HES)
0033  IF(IERR .LT. 0) WRITE(LU,999) IERR
0034  CALL WRITF(IDCB,IERR,IBUF,80)
0035  CALL CLOSE(IDCB,IERR)
0036  C
0037  C SCHEDULE DISPLAY
0038  CALL EXEC(23,NAME1,LU,NSTG,0,0,0,IBUF,80)
0039  C
0040  WRITE(LU,40)
0041  40 FORMAT(" ENTER DISPLAY DEVICE ":// 1. TV/" 2. HP2648A")
0042  READ(LU,*) IDEV
0043  IF(IDEV .EQ. 2) GO TO 41
0044  CALL EXEC(23,NAME2)
0045  GO TO 42
0046  41 CONTINUE
0047  CALL HP48A(LU)
0048  42 CONTINUE
0049  999 FORMAT(" FILE ERROR ")
0050  STOP
0051  END
SUBROUTINE HP48A(LU)
DIMENSION IB(14), IA(4)
INTEGER IDCB(14), BUFF(4), NAME(3)
DATA NAME/2HDA, 2HTA, 2H1 /
CALL OPEN(IDCB, IERR, NAME)
IF (IERR .GE. 0) GO TO 30
WRITE(LU, 10) IERR
10 FORMAT ("OPEN ERROR", F5.0)
STOP
IF (IERR .GE. 0) GO TO 40
WRITE(LU, 31) IERR
31 FORMAT ("READ ERROR", F5.0)
GO TO 55
40 CONTINUE
CALL DVECT(BUFF, BUFF(2), BUFF(3), BUFF(4), LU)
GO TO 20
CALL EXEC(13, LU, ISTAT)
ISTAT = IAND(ISTAT, 140000B)
IF (ISTAT .NE. 0) GO TO 55
CALL GRAFC(0, LU)
CALL CLOSE(IDCB)
RETURN
SUBROUTINE GRAFC(IFLAG, LU)
INTEGER IESC
IESC = 33B
C GRAPHICS OFF = 0; GRAPHICS ON NOT = 0
IF (IFLAG .EQ. 0) GO TO 100
GO TO 200
100 WRITE(LU, 30) IESC
30 FORMAT ("GRAPHIC")
200 RETURN
C GRAPHIC ON
WRITE(LU, 10) IESC
10 FORMAT ("*dC")
WRITE(LU, 12) IESC
12 FORMAT ("*dF")
WRITE(LU, 14) IESC
14 FORMAT ("*dA")
GO TO 200
WRITE(LU, 30) IESC
30 FORMAT ("GRAPHIC")
SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)

C
C SUBROUTINE DRAWS A LINE BETWEEN THE TWO POINTS (IX1,IY1)
C AND (IX2,IY2). THE POINT (IXO,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
END$
SUBROUTINE WINDO(ALPHA,N,M,WIN)

XN=SQR(M**2 + N**2)

BETA=ALPHA*SQR(1.-XN)

CALL BESIO(ALPHA,BIAA)

CALL BESIO(BETA,BIBB)

BETAI=ALPHA*SQR(2)

CALL BESIO(BETAI,BIB)

ZMIN=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

Z=X*X*7.11111E-2

RIO=(((4.5813E-3*Z+3.60768E-2)*Z+2.659732E-1)*Z+1.206749E0

1089942E0)*Z+3.515623E0)*Z+1.

RETURN

Z=3.75/RIO

RIO=EXP(RIO)/SQR(RIO)*(((((((3.92377E-3*Z-1.647633E-2)*Z+2.507706E-2)*Z+9.16281E-3)*Z-1.57565E-3)*Z+2.25319E-2+1.328592E-2)*Z+3.989423E-1)

RETURN

END
SUBROUTINE BESJ

PURPOSE

COMPUTE THE J BESSSEL FUNCTION FOR A GIVEN ARGUMENT AND

USAGE

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

DESCRIPTION OF PARAMETERS

X   -THE ARGUMENT OF THE J BESSSEL FUNCTION DESIRED
N   -THE ORDER OF THE J BESSSEL FUNCTION DESIRED
BJ  -THE RESULTANT J BESSSEL 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 B
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. GOLDSTEI
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 BESSLER 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
31 IF(X-15.)32,32,34
32 NTST=20.+10.*X-(X**2/3))
GO TO 36
34 NTST=90.+X/2.
36 IF(N-NTST)40,38,38
38 IER=4
RETURN
0058 40 IER=0
0059 50 M1=N+1
0060 BPREV=0.
0061 C
0062 C COMPUTE STARTING VALUE OF M
0063 C
0064 IF(X-5.)50,60,60
0065 50 MA=X+6.
0066 GO TO 70
0067 60 MA=1.4*X+60./X
0068 70 MB=N+IFIX(X)/4+2
0069 NZERO=MAXO(MA,MB)
0070 C
0071 C SET UPPER LIMIT OF M
0072 C
0073 MMAX=TEST
0074 100 DO 190 M=NZERO,MMAX,3
0075 C
0076 C SET F(M),F(M-1)
0077 C
0078 FM1=1.0E-28
0079 FM=.0
0080 ALPHA=.0
0081 IF(M-(M/2)*2)120,110,120
0082 110 JT=-1.
0083 GO TO 130
0084 120 JT=1.
0085 130 M2=N-2
0086 DO 160 K=1,M2
0087 HK=M-K
0088 BMK=2.*FLOAT(MK)*FM1/X-FM
0089 FM=FM1
0090 FM1=BMK
0091 IF(MK-N-1.)150,140,150
0092 140 BJ=BMK
0093 150 JT=-JT
0094 S=1+JT
0095 160 ALPHA=ALPHA+BMK*S
0096 BMK=2.*FM1/X-FM
0097 IF(N)180,170,180
0098 170 BJ=BMK
0099 180 ALPHA=ALPHA+BMK
0100 BJ=BJ/ALPHA
0101 IF(ABS(BJ-BPREV)-ABS(D*BJ))200,200,190
0102 190 BPREV=BJ
0103 IER=3
0104 200 RETURN
0105 END
0106 $
&BLDWF T=00004 IS ON CR00022 USING 00022 BLKS R=4113

0001 FTN4
0002 PROGRAM BLDWF
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 IDCB1(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)),(IPIXL,IRTN(5)),(ILINE,RDATA),
0017 1(RDATA(2),RMAX),(RDATA(3),RMIN),(IBUF(12),ILOC),(IBUF(13),JN
0018 EQUIVALENCE (IBUF(7),NLINE),(IBUF(8),NPIXL),(IBUF(9),TPMIN),
0019 1 (IBUF(10),IPMAX),(ISIZE(2),ISIZE2)
0020 C
0021 C
0022 C
0023 C
0024 C GET INPUT PARAMETERS
0025 C
0026 CALL RMPAR(LU)
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 (ICMFW(IMAGE,12H ,6) .EQ. 0) GO TO 140
0045 C
0046 C OPEN DIRECTORY FILE...
0047 C
0048 90 CALL OPEN(IDCB1,IERR,6HIMDIRC,1,2HIM,23,272)
0049 C
0050 C IF (IERR .LT. 0) GO TO 9999
C FIND IMAGE
0053 100 CALL READF(IDCBI,IERR,IBUF,15,LEN)
0054 IF (IERR .LT. 0) GO TO 9999
0055 IF (LEN .EQ. -1) GO TO 9990
0056 C IMAGE FOUND
0057 IF (ICMPW(IMAGE,IBUF,6) .NE. 0) GO TO 100
0058 C IMAGE IS ON DISC
0059 C CREATE WORK FILE
0060 C
0061 C ASK IF USER WANTS TO SAVE WORK FILE
0062 WRITE(LU,6)
0063 FORMAT(" DO YOU WANT TO SAVE IMAGE IN CURRENT WORK FILE? ")
0064 READ(LU,2) IANS
0065 IF (IANS .EQ. 2HNO) GO TO 110
0066 C SCHEDULE BUILD IMAGE PROGRAM
0067 C
0068 CALL OPEN(IDCBI,IERR,6HWFOO00)
0069 IF (IERR .EQ. -6) GO TO 110
0070 IF (IERR .LT. 0) GO TO 9999
0071 C OPEN IMAGE DATA FILE
0072 CALL OPEN(IDCBI,IERR,JNAME,1,2HIM,23,528)
0073 IF (IERR .LT. 0) GO TO 9999
0074 CALL EXEC(23,6HBLDIM ,LU)
0075 CALL CLOSE(IDCBI)
0076 CALL EXEC(23,6HBLDIM ,LU)
0077 CALL CREAT(IDCBI,IERR,6HWFOO00,ISIZE,2,0,0,1040)
0078 IF (IERR .LT. 0) GO TO 9999
0079 C COPY DATA AND CONVERT TO REAL
0080 C
0081 CALL WRITF(IDCBI,IERR,RDATA,1)
0082 DO 120 I-1,NLINE
0083 DO 115 J=1,NPIXL
0084 RDATA(J) = IDATA(J)
0085 CALL WRITF(IDCBI,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, RMI

THIS SUBROUTINE IS USED TO SCROLL AN IMAGE ON THE GMR-27

IDCB = OPENED DATA CONTROL BLOCK FOR THE IMAGE
IDIRC = DIRECTION TO SCROLL (N = BACK N LINES N= FORWARD N LINE
NLINE = # LINES IN IMAGE
IFIRST = LOWEST IMAGE LINE DISPLAYED
ILAST = HIGHEST IMAGE LINE DISPLAYED

DIMENSION IDCB(144), DATA(512)
INTEGER IDCB
DATA IUP, IDOWN/34060B, 340408/
CHECK IF NO WORK NECESSARY
IF (IDIRC .EQ. 0) RETURN
IF (IDIRC .GT. 0) GO TO 200
DO 100 I = -1, IDIRC, 1
IF (IFIRST .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)
IFIRST = IFRST - 1
ILAST = ILAST + 1
100 CONTINUE
IF (SCROL .LT. 0) RETURN
CALL DRIVR(2, IDOWN, 1)
CALL WLINE(255, 0, LEN-1, IDATA)
ILAST = ILAST + 1
210 IFRST = IFRST + 1
RETURN
END
SUBROUTINE WLINE(LINE, IPIX, JPIX, IDATA)

C THIS SUBROUTINE WRITES A DESIGNATED LINE TO THE GMR-27

C  LINE - LINE NUMBER
C  IPIX - STARTING PIXEL
C  JPIX - ENDING PIXEL
C  IDATA - BUFFER CONTAINING IMAGE DATA FOR LINE

C

DIMENSION IDATA(512), INIT(6)

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

DATA INIT/100377B, 64000B, 44000B, 50000B, 24041B, 26002B/

COMPUTE DIRECTION

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

SET UP TO WRITE LINE

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

WRITE LINE

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

RETURN

END
&DRIVR T-00004 IS ON CRO0022 USING 00012 BLKS R-0241

0001 ASMB,R,L,C
0002 NOM 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 DFS 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 *
0061 * READ ENTRY
0062 *
0063 D.2 SSA SKIP 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,IM,LU)
COMPLEX P(10),Q(10),PP
DIMENSION U(3,3,2),V(3,3,2)
COMMON/WORK/AMAG(10),A(3,3),B(3,3)
WRITE(LU,100)
FORMAT(9 SELECT FILTER "/, 1. BUTTERWORTH "/, 1 2. CHEBYSHEV "/, 3. LINEAR PHASE "/)
READ(LU,*) ITYPE
WRITE(LU,110)
FORMAT(1 ENTER THE NUMBER OF FILTER STAGES "/)
READ(LU,*) NSTG
WRITE(LU,120)
FORMAT(1 ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS "/)
READ(LU,*) WR
WRITE(LU,140)
FORMAT(1 ARE ALL ZEROS LOCATED AT INFINITY "/, 1 1 = YES "/, 2 = NO "/)
READ(LU,*) IFLAG
WRITE(LU,151)
FORMAT(1 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
DO 1 J-1,NSTG
WRITE(LU,130) J
FORMAT(1 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 11 I=1,3
DO 111 K=1,3
U(I,K,J) = A(I,K)
V(I,K,J) = B(I,K)
WRITE(LU,40) P(J),AMAG(J)
FORMAT(1X,1(" P="1E15.5," +J",1E15.5,/"," PMAG= ",,E15.5)
CONTINUE
MN = NSTG + 1
WRITE(1,1112) U
WRITE(1,1112) V
FORMAT(3E15.4)
RETURN
END
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=-0.0*COS(X)*SIN(X)

C3=SIN(X)**2

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

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

C9=|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*(C1-C3-C7+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) G0 TO 10

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

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

C6=|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+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

GO TO 20

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

$
PROGRAM START

THIS PROGRAM EVALUATES THE FILTER STABILITY CHARACTERISTICS

COMMON/WORK/WO(130)

INTEGER BUFF

DIMENSION IBUF(80),ILU(5),IRTN(5)

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

EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))

CALL RMPAR(ILU)

LU=ILU(1)

MN=ILU(2) + 1

GET FILTER COEFF'S

CALL EXEC(14,1,IBUF,80)

CALL STABT(V,MN,IRTN,LU)

IRTN = IRTN

CALL PRIN(IRTN)

END

SUBROUTINE STABT(V,MN,IRTN,LU)

SUBROUTINE CHECKS STABILITY OF SYSTEM EQUATION-

Y(M,N)=A*Y(M-1,N)+B*Y(M,N-1)

C---COEFFICIENT MATRIX OF DENOMINATOR OF ZW-TRANSFORM OF SYS

LOGICAL ISTAB

DIMENSION V(3,3,2)

DIMENSION C(5,5),A(25,25),B(25,25),S(25,25),EVR(25),EVI(25)

COMMON/WORK/IERR(25)

IM=25

M=2*(MN-1)+1

M=N**2

IF(MN.EQ.3) GO TO 5

PUT COEFFICIENTS IN STABILITY ARRAY

DO 6 I=1,3

DO 6 J=1,3

C(I,J)=V(I,J,1)

GO TO 13

DO 5 I=1,5

DO 5 J=1,5

DO 5 K=1,3

DO 5 L=1,3

IF((IK .LE. 0) .OR. (IK .GT. 3)) GO TO 10

IF((JL .LE. 0) .OR. (JL .CT. 3)) GO TO 10

C(I,J)=C(I,J)+V(IK,JL,1)*V(K,L,2)

CONTINUE
CONTINUE
11 FORMAT(200H COEFFICIENT MATRIX,/)  
12 FORMAT(IH,5F15.6)  
C FORM A AND B MATRICES
DO 22 I=1,N
DO 22 J=1,M
A(I,J)=0.0
B(I,J)=0.0
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.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.EQ.1) B(K,N,K)=1.0
25 CONTINUE
FIND EIGENVALUES OF A AND B
DO 27 I=1,M
DO 27 J=1,M
S(I,J)=A(I,J)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,71)
71 FORMAT(/,10X,19HEIGEN VALUES OF (A))
TEST=1.0
IONE=0
CALL PNTEV(EVR,EVI,MDM:,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) GOTO 405
FOR!l-kT(" 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 28 I=1,M
DO 28 J=1,M
28 S(I,J)=B(I,J)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,72)
72 FORMAT(/,10X,10HEIGEN VALUES OF (B))
CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) GOTO 405
C
C FIND EIGENVALUES OF A+B
DO 29 I=1,M
DO 29 J=1,M
S(I,J)=A(I,J)+B(I,J)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,73)
73 FORMAT(/,10X,10HEIGEN VALUES OF (A+B))
CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) WRITE(LU,400)
IF(ISTAB) GOTO 404
WRITE(LU,401)
IRTCD = 0
GO TO 500
C
C FIND EIGENVALUES OF A*S
DO 30 I=1,M
DO 30 J=1,M
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
S(K,L)=1.0
CALL MLTMX(A,S,M,MDIM)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,74)
74 FORMAT(/,10X,10HEIGEN VALUES OF (A*S))
C
IONE=1
TEST=0.5
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
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
S(K,L)=1.0
CALL MLTMX(B,S,M,MDIM)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,75)
75 FORMAT(/,10X,10HEIGEN 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)
C FIND EIGENVALUES OF ABS(A)+ABS(B)

DO 33 I=1,M
DO 33 J=1,M
S(I,J)=ABS(A(I,J))+ABS(B(I,J))
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,76)
76 FORMAT(/,10X,29HEIGEN VALUES OF ABS(A)+ABS(B))

TEST=1.0

CALL PNT(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) WRITE(LU,401)

C FIND EIGENVALUES OF A*B

CALL MLTMX(A,B,M,MDIM)
CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
WRITE(LU,77)
77 FORMAT(/,10X,21HEIGEN VALUES OF (A*B))
CALL PNT(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
IF(ISTAB) WRITE(LU,401)
GO TO 501
500 IF(ISTAB) IRTCD = 1000
RETURN

END

SUBROUTINE PNT(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
LOGICAL ISTAB
DIMENSION EVR(MDIM),EVI(MDIM),IERR(MDIM)

ISTAB=.FALSE.
D=1.0E-20
RMX=0.0
DO 20 I=1,M
R=EVR(I)**2+EVI(I)**2
R=SQRTR(R)
RMX=MAX1(RMX,R)
IF(R.LT.D) GO TO 20
IF(IERR(I).LT.0) 14RITF.(LU,93)
10 FORMUT(1H114.7,4X,2H4+J,E14.7)
11 FORMUT(13HABS(LMDA) = ,E14.7)
30 FORMUT(19H SPECTRAL RADIUS = ,E14.7/
93 FORMUT(/,10X,"IERR(",I2," ) = ",I2/
RETURN
END

SUBROUTINE RNAN(N,M,S,EVR,EVI,IERR)
SUBROUTINE WAS WRITTEN TO CALL HSBG AND ATEIG IBM SCIENTIFIC
SUBROUTINES TO CALCULATE THE EIGENVALUES OF A REAL MATR
M-----ORDER OF THE MATRIX S
N-----SIZE OF FIRST DIMENSION ASSIGNED TO THE ARRAY S IN THE
CALLING PROGRAM
SUBROUTINE ATEIG
PURPOSE
COMPUTE THE EIGENVALUES OF A REAL ALMOST TRIANGULAR MATRIX

USAGE
CALL ATEIG(M,A,RR,RI,IANA,IA)

DESCRIPTION OF THE PARAMETERS

M	ORDER OF THE MATRIX
A	THE INPUT MATRIX, M = M
RR	VECTOR CONTAINING THE REAL PARTS OF THE EIGENVALUES ON RETURN
RI	VECTOR CONTAINING THE IMAGINARY PARTS OF THE EIGENVALUES ON RETURN
IANA	VECTOR WHOSE DIMENSION MUST BE GREATER THAN OR EQUAL TO M, CONTAINING ON RETURN INDICATIONS ABOUT THE EIGENVALUES APPEARED (SEE NATH. 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 THE DIMENSION OF M.

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

METHOD
QR DOUBLE ITERATION

REFERENCES

SUBROUTINE ATEIG(M,A,RR,RI,IANA,IA)
DIMENSION A(1),RR(1),RI(1),Prr(2),PRI(2),IANA(1)
INTEGER P,P1,Q
E7=1.0E-8
E6=1.0E-6
E10=1.0E-10
DELTA=0.5
MAXIT=30
INITIALIZE

N=N

20 N1=N-1
IN=N1*IA
NN=IN+N
IF(N1) 30,1300,30
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
Y1=IN-IA
NN1=IN1+N
NIN=IN+N1
NIN1=IN1+N1
60 T=A(N1N1)-A(NN)
U=T*T
V=4.0*A(NIN)*A(NN1)
IF(ABS(V)-U*E7) 100,100,65
65 T=U+V
67 T=0.0
100 IF(T)120,110,110
110 RR(N1)=A(N1N1)
GO TO 130
113

0349 120 RR(N1)=A(NN)
0350 RR(N1)=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)=U
0357 RI(N)=V
0358 160 IF(N2)1280,1280,180
0359 C
0360 C TESTS OF CONVERGENCE
0361 C
0362 180 N1N2=N1N1-L1
0363 RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
0364 EPS=1.0*SQRT(RMOD)
0365 IF(ABS(A(N1N2))-EPS)1280,1280,240
0366 240 IF(ABS(A(NNN1))-1.0*ABS(A(NN))) 1300,1300,250
0367 250 IF(ABS(PAN1-A(NNN2))-ABS(A(N1N2))*E6) 1240,1240,260
0368 260 IF(ABS(PAN-A(NNN1))-ABS(A(NNN1))*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 1 340,360,360
0378 340 J=J+1
0379 360 CONTINUE
0380 GO TO (440,460,460,480),J
0381 440 R=0.0
0382 S=0.0
0383 GO TO 500
0384 460 J=N+2-J
0385 R=RR(J)*RR(J)
0386 S=RR(J)+RR(J)
0387 C TO 500
0388 480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
0389 S=RR(N)+RR(N1)
0390 C
0391 C SAVE THE LAST TWO SUBDIAGONAL TERMS AND THE ROOTS OF THE
0392 C SUBMATRIX BEFORE ITERATION
0393 C
0394 500 PAN=A(NNN1)
0395 PAN1=A(N1N2)
0396 DO 520 I=1,2
0397 K=NP-I
0398 PRR(I)=RR(K)
0399 520 PRI(I)=RI(K)
0400 C
0401 C SEARCH FOR A PARTITION OF THE MATRIX, DEFINED BY P AND Q
0402 C
0403 P=N2
0404 IF (N-N3)600,600,525
0405 525 IPI=N1N2
0406 DO 580 J=2,N2
0407 IPI=IPI-IA-1
0408 IF(ABS(A(IPI))-EPS) 600,600,530
0409 530 IPIP=IPI+IA
IPIP2 = IPIP + IA
D = A(IPIP) * (A(IPIP) - S) + A(IPIP2) * A(IPIP + 1) + R
IF(D) = 540, 560, 540
540 IF(ABS(A(IPI)) * A(IPIP + 1)) * (ABS(A(IPIP) + A(IPIP2 + 1) - S) + ABS(A(IPI)) - ABS(D) * EPS) = 620, 620, 560
560 P = N1 - J
580 CONTINUE
600 Q = P
620 GO TO 680
660 P1 = P - 1
680 IF (PI - 1) = 680, 680, 650
690 DO 660 I = 2, PI
700 IPI = IPI - IA - 1
710 IF(ABS(A(IPI)) - EPS) = 680, 680, 660
720 Q = Q - 1
780 CAP = SQRT(G1 * G1 + G2 * G2 + G3 * G3)
800 IF(CAP) = 800, 840, 840
820 ALPM = 2.0
840 PSI1 = 0.0
860 PSI2 = 0.0
880 A(IPI + 1) = -A(IPI)
900 IF(I - Q) = 900, 960, 900
920 A(IPI) = -CAP
940 A(IPI) = -A(IPI)
960 CONTINUE
DO 1040 J=1,N
T=PS11*A(IJ+1)

IF(I-N1)980,1000,1000

980 IP2J=I+2
T=T+PS12*A(IP2J)

1000 ETA=ALPHA*(T+A(IJ))
A(IJ)=A(IJ)-ETA
A(IJ)=A(IJ+1)-PS11*ETA

IF(I-N1)1020,1040,1040

1020 A(IP2J)=A(IP2J)-PS12*ETA

1040 IJ=IJ+IA

C C COLUMN OPERATION

C

IF(I-N1)1080,1060,1060

1060 K=N
GO TO 1100

1080 K=I+2
1100 IP=IIP-I

DO 1180 J=Q,K
JI=JI+J

JI=JIP+IA
T=PS11*A(JIP)
IF(I-N1)1120,1140,1140

1120 JIP2=JIP+IA
T=T+PS11*A(JIP2)

1140 ETA=ALPHA*(T+A(JI))
A(JI)=A(JI)-ETA
A(JI)=A(JI)-PSI1

IF(I-N1)1160,1180,1180

1160 A(JIP2)=A(JIP2)-ETA*PS12

1180 CONTINUE

1180 K=I+2
JIP=JIP+IA

IF(I-N2)1200,1220,1220

1200 JI=IJ+3
JIP=JIP+IA

1220 II=IIP+1
IT=IT+I
GO TO 60

C C END OF ITERATION

C

IF(ABS(A(N11))-ABS(A(N1N2))) 1300,1280,1280

1300 C TWO EIGENVALUES HAVE BEEN FOUND

C

C

C

C

ONE EIGENVALUE HAS BEEN FOUND
SUBROUTINE HSBG

PURPOSE
TO REDUCE A REAL MATRIX INTO UPPER ALMOST TRIANGULAR FORM

USAGE
CALL HSBG(N,A,IA)

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 ARRAY A IN THE CALLING PROGRAM WHEN THE MATRIX IS IN DOUBLE SUBSCRIPTED DATA STORAGE MODE. IA=N WHEN THE MATRIX IS IN SSP VECTOR STORAGE MODE.

REMARKS
THE HESSENBERG FORM REPLACES THE ORIGINAL MATRIX IN THE ARRAY A.

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

METHOD
SIMILARITY TRANSFORMATIONS USING ELEMENTARY ELIMINATION MATRICES, WITH PARTIAL PIVOTING.

REFERENCES

SUBROUTINE HSBG(N,A,IA)
DIMENSION A(1)
DOUBLE PRECISION S
L=N
NI=A*L
LIA=NIA-IA

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
ISUB=LIA+L
IPIV=ISUB-IA
PIV=ABS(A(IPIV))
IF(L-3) 90,90,50
50 M=IPIV-IA
DO 80 I=L,M,IA
T=ABS(A(I))
IF(T-PIV) 80,80,60
60 IPIV=I
PIV=T
80 CONTINUE
90 IF(PIV) 100,320,100
100 IF(PIV-ABS(A(ISUB))) 180,180,120
C
INTERCHANGE THE COLUMNS
C
120 M=IPIV-L
DO 140 I=1,L
J=M+I
T=A(J)
K=LIA+I
A(J)=A(K)
140 A(K)=T
C
INTERCHANGE THE ROWS
C
M=L2-M/LA
DO 160 I=L1,NIA,IA
T=A(I)
J=I-M
A(I)=A(J)
160 A(J)=T
C
tERMS OF THE ELEMENTARY TRANSFORMATION
C
180 DO 200 I=L,LIA,IA
200 A(I)=A(I)/A(ISUB)
C
RIGHT TRANSFORMATION
C
J=-IA
DO 240 I=1,L2
J=J+IA
LJ=L+J
DO 220 K=1,L1
KJ=K+J
KL=K+LIA
220 A(KJ)=A(KJ)-A(LJ)*A(KL)
240 CONTINUE
C
LEFT TRANSFORMATION
0637     K=IA
0638     DO 300 I=1,M
0639     K=K+IA
0640     LK=K+L1
0641     S=A(LK)
0642     LJ=L-IA
0643     DO 280 J=1,L2
0644     JK=K+J
0645     LJ=LJ+IA
0646     280 S=S+A(LJ)*A(JK)*1.0D0
0647     300 A(LK)=S
0648     C
0649     SET THE LOWER PART OF THE MATRIX TO ZERO
0650     C
0651     DO 310 I=L,LIA,IA
0652     310 T(I)=0.0
0653     320 L=L1
0654     GO TO 20
0655     360 RETURN
0656     END
0657     SUBROUTINE MLTX(A,S,M,MDIM)
0658     C
0659     SUBROUTINE OBTAINS THE MATRIX MULTIPLICATION OF A AND S AND
0660     THE RESULTS IN S.
0661     C
0662     DIMENSION S(MDIM,MDIM),A(MDIM,MDIM)
0663     COMMON /WORK/T(25,25)
0664     DO 10 I=1,M
0665     DO 10 J=1,M
0666     C=0.0
0667     DO 20 K=1,M
0668     20 C=C+A(I,K)*S(K,J)
0669     10 T(I,J)=C
0670     DO 50 I=1,M
0671     DO 50 J=1,M
0672     50 S(I,J)=T(I,J)
0673     RETURN
0674     END
0675     BLOCK DATA WORK
0676     COMMON /WORK/ WO(625)
0677     END
0678     "$
PROGRAM FILTR

C WRITTEN BY E. E. SHERROD

C THIS PROGRAM SELECTS THE FILTERING TYPE

DIMENSION ILU(5),NAME1(3),NAME2(3),IRTN(5),NAME3(3)
EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
DATA NAME1/2HLF,2HLT,2HR/
DATA NAME2/2HHF,2HLT,2HR/
DATA NAME3/2HSH,2HOW,2H/

C GET LU

CALL RMPAR(ILU)
IPIXL =0
JPIXL =511
LU=ILU(1)
WRITE(LU,10)
10 FORMAT(" SELECT FILTERING TYPE "/ 1. LINEAR "/ 2. HOMOMORP"
READ(LU,* ) IFITR
IF(IFITR .EQ. 1) CALL EXEC(23,NAME1,LU,IPIXL,JPIXL,0,0)
CALL RPAR(IRTN)
IF(IFITR .EQ. 1) GO TO 30
IF(IFITR .EQ. 2) CALL EXEC(23,NAME2,LU,IPIXL,JPIXL,0,0)
CALL RPAR(IRTN)
WRITE(LU,40) RMAX,RMIN
40 FORMAT(" MAX PIXEL = ",F12.2,10X," MIN PIXEL = ",F12.2)
IX=RMAX-RMIN +0.5
WRITE(LU,50) IX
50 FORMAT(" NUMBER OF GRAY LEVELS = ",I5)
IF(IFITR .EQ. 2) CALL EXEC(23,NAME3,LU,0,511,0,0)
STOP
END
1001  FTN4,L
1002  PROGRAM NOISE
1003  C
1004  DIMENSION RDATA(512),GNOISE(512),LU(5),IU(5),IBUF(40)
1005  C
1006  INTEGER READL
1007  EQUIVALENCE (RDATA,LU(2)),(LU(2),ILINE),(LU(3),IPIXL),
1008  (RDATA(2),RMAX),(RDATA(3),RMIN)
1009  DATA RDATA/512*0.0/
1010  C
1011  C GET INPUT PARAMETERS
1012  C
1013  CALL RMPAR(LU)
1014  C
1015  C SCHEDULE BUILD WORK FILE PROGRAM
1016  CALL EXEC(23,6HBLDF,LU)
1017  C
1018  C READ WORK FILE HEADER
1019  C
1020  IERR = READL(-1,0,511,RDATA)
1021  IF (IERR .LT. 0) GO TO 999
1022  NLINE=ILINE
1023  NPIXL=IPIXL
1024  PMAX=RMAX
1025  PMIN=RMIN
1026  C
1027  C GET NOISE INFO
1028  WRITE(LU,13)
1029  FORMAT(" ENTER NOISE MEAN VALUE _")
1030  READ(LU,*) AM
1031  WRITE(LU,14)
1032  FORMAT(" ENTER STANDARD DEVIATION VALUE _")
1033  READ(LU,*) S
1034  IF(S .LE. 0) GO TO 1000
1035  C
1036  DO 100 I=0,NLINE-1
1037  IF (READL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
1038  C
1039  C GET NOISE
1040  C
1041  DO 101 JA=0,51
1042  CALL EXEC(1,8,IBUF,40)
1043  JJ=10*JA
1044  CALL CODE(80)
1045  READ(IBUF,12)(GNOISE(K+JJ),K=1,10 )
1046  12 FORMAT(10F8.5)
1047  101 CONTINUE
DO 90 J =1,NPIXL
RDATA(J) = RDATA(J) + NOISE(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
C
CALL CLSWF(NLINE,NPIXL,PMAX,PMIN)
CALL CLOSE(IDCB1)
999 WRITE(LU,2) IERR
2 FORMAT("FILE ERROR",I7)
END
$
Stability Analysis of Two-Dimensional Digital Recursive Filters

WINSER 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 radius 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 \( \bar{A} \) [7] 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

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

For this 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 \( C(\bar{B}_1, \bar{B}_2) \) 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

\[ C(\bar{B}_1^J, \bar{B}_2^K) = \left( \sum_{\tau=1}^{J+K} + K \right)! \bar{B}_1^\tau \bar{B}_2^\tau / (J!K!) \]

In general, the matrices do not commute. Therefore, we give as an example \( C(\bar{B}_1^J, \bar{B}_2^K) \) as that input vector, \( H_{JK} \), which corresponds to a unit impulse at the \((J,K)\) position where \( J < m \) and \( K < n \), is given by

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

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 \( G_{j+1,k} \), \( G_{j+1,k+1} \), and \( G_{j+1,k+1} \) for input vector \( H_{j,k} \), and if all initial condition vectors are null vectors, we obtain

\[
\begin{align*}
G_{j+1,k} &= \overline{B}_j G_{j,k} = \overline{B}_j \overline{A} H_{j,k} \\
G_{j+1,k+1} &= \overline{B}_j G_{j,k+1} + \overline{B}_j G_{j+1,k} = \left( \overline{B}_j^2 B_2 + \overline{B}_j B_1 \right) \overline{A} H_{j,k}
\end{align*}
\]

If we use Lemma 1, we obtain

\[
\begin{align*}
G_{m+1,n} &= \overline{B}_1 C(\overline{B}_1^{-m}, \overline{B}_2^{-n}) \\
G_{m+1,n-1} &= + \overline{B}_1 C(\overline{B}_1^{-m-1}, \overline{B}_2^{-n-1}) \overline{A} H_{j,k}.
\end{align*}
\]

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

\[
G_{m+1,n} = \overline{B}_1 G_{m,n} + \overline{B}_1 G_{m+1,n-1}.
\]

IV. Conclusions

In this paper, a new approach to stability analysis of two-dimensional digital recursive filters has been presented. Theorems have been presented which can be used in the practical application of this approach. The authors feel that it is important to note that no known unstable filter has been found in this research effort which did not have either \( \rho(\overline{B}_1), \rho(\overline{B}_2) \), or \( \rho(\overline{B}_1 + \overline{B}_2) \) greater than or equal to one. One is lead to conjecture that for a large class of filters, any filter in the class is stable if the subject spectral radii are all less than one. However, the proof of this is not trivial.
equal to one. It follows directly that
\[ G_{m+1,n} = C(\tilde{B}_1^{m+1-j}, \tilde{B}_2^{m+1-k}) \bar{A} H_{J,K}. \]  
(A6)

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

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

It follows that
\[ G_{m,n+1} = C(\tilde{B}_1^{m-1-j}, \tilde{B}_2^{m-1-k}) \bar{A} H_{J,K}. \]  
(A9)

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

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

It follows from (A5) and (A11) that
\[ G_{m+1,n+1} = C(\tilde{B}_1^{m-1-j}, \tilde{B}_2^{m-1-k}) \bar{A} H_{J,K}. \]  
(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 \( n \). (A13)

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
\[ S = \lim_{m \to \infty} \| G_{m,n} \| = \lim_{m \to \infty} \| \epsilon \bar{Q}^m P_Q \| = \lim_{m \to \infty} | \epsilon \lambda_Q^m P_Q |. \]  
(A15)

Note that \( S \) is infinite 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
\[ G_{m,n} = f(J,K)C(\tilde{B}_1^{m-j}, \tilde{B}_2^{m-k}) \bar{A} H_{J,K} \]
\[ g(M,N) = \delta_{G,M,N} \]  
(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
\[ G_{m,n} = f(0,N)C(\tilde{B}_1^m, \tilde{B}_2^m) \bar{A} H_{0,N} = f(0,N) \bar{B}_1^m \bar{A} H_{0,N}. \]  
(A17)

If we apply Lemma 2, we see that the system is unstable if \( \rho(B_1) > 1 \). If we let \( M = N \) and \( K = 0 \) in (A16), we have
\[ G_{m,m} = f(M,0)C(\tilde{B}_1^m, \tilde{B}_2^m) \bar{A} H_{m,0} = f(M,0) \bar{B}_1^m \bar{A} H_{m,0}. \]  
(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
\[ G_{m,m} = \sum_{J=0}^{M} f(M-J)C(\tilde{B}_1^M, \tilde{B}_2^M) \bar{A} H_{m,m-J}. \]  
(A19)

Since all inputs are equal, we can write
\[ G_{m,m} = f(0,M) \left[ \sum_{J=0}^{M} C(\tilde{B}_1^M, \tilde{B}_2^M) \right] \bar{A} H_{0,M} \]  
(A20)

\[ G_{m,m} = f(0,M)(\tilde{B}_1 + \tilde{B}_2)^M \bar{A} H_{0,M} \]  
(A21)

since
\[ (\tilde{B}_1 + \tilde{B}_2)^M = \sum_{J=0}^{M} C(\tilde{B}_1^{M-J}, \tilde{B}_2^J) \]  
(A22)

whether or not \( \tilde{B}_1 \) and \( \tilde{B}_2 \) commute. If we apply Lemma 2, we see that the system is unstable if \( \rho(\tilde{B}_1 + \tilde{B}_2) > 1 \) and Theorem 2 holds.

Acknowledgment

The authors wish to thank E. J. 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

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.

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.
STABILITY ANALYSIS OF TWO DIMENSIONAL DIGITAL RECURSIVE FILTERS

Winzer E. Alexander

Department of Electrical Engineering
North Carolina A&T State University

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 \(g(m,n)\) is then given by \(g(m,n) = DG\) 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 \(b_{jk}\) or one as appropriate. It is convenient to consistently divide each 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 [1,2,3]. 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 [4] 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 radii 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 (\( \rho(\text{abs}(B_1) + \text{abs}(B_2)) < 1 \)).

Theorem 3

There is a particular permutation matrix \( \mathbf{S} \) [5] such that if \( \rho(B_1), \rho(B_2) \) and \( \rho(B_1 + B_2) \) 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 \( b_{jk} = b_{kj} \) 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 \( |b_{01}| \geq 1 \), or if

\[
\max \left( \frac{-b_{01}^2}{2}, \frac{1}{2} \sqrt{b_{01}^2 + 4b_{01}} \right) \geq 1
\]

Example 3

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

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

If we apply Theorem 1, we obtain \( \rho(B_1) = 0.95 \), \( \rho(B_2) = 0.95 \) 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

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

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

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

\[
g(m,n) = f(m,n) - 0.5 g(m-1,n) - 0.5 g(m,n-1) - 0.9 g(m-2,n) - 0.5 g(m-2,n-1)
\]

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

225
If we apply Theorem 3, we obtain \( p(B_1) = 0.5 \),
\( p(B_2) = 0.5 \), \( \rho(B_1 + B_2) = 0.866 \), \( \rho(B_1) = \rho(B_2) = 0.35355 \). Therefore, we conclude that this filter is stable. This filter was verified to be stable by Mariad Fahmy [8].

**Example 6**

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

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

(11)

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

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

(12)

In considering more complex examples, it is convenient to present the coefficients \( b_{jk} \) in matrix form. Let the matrix \( V \) 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 \( V \) matrix corresponding to example (1) is given by

\[
V = \begin{bmatrix}
1.0 & b_{01} \\
-1.0 & 1.0
\end{bmatrix}
\]

(13)

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

**Example 7**

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

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

(14)

For this example, \( p(B_1) = 2.169 \), \( p(B_2) = 2.104 \) and \( \mu(B_1 + B_2) = 2.599 \). 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 \( B_1 \) and \( B_2 \) 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, highpass, 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 ZW-Transform domain. A pseudostate space representation for the corresponding two dimensional ZW-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 ZW-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 or modifying 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 ZW-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 ZW-Transform for the function is defined by

\[
F(z,w) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(mX, nY) z^{-m-w} \quad (2.1)
\]

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

\[
F(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} f(m,n) z^{-m-w} \quad (2.2)
\]

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

\[
G(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} g(m,n) z^{-m-w} \quad (2.3)
\]

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

\[
H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n) z^{-m-w} \quad (2.4)
\]

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 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

\[
C(z,w) = d(z,w)F(z,w) \quad (2.6)
\]

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

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

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

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.

If the filter is unstable, then either \( \hat{B} \), \( \hat{C} \) or \( (\hat{B} + \hat{C}) \) has at least one eigenvalue with a magnitude greater than one, equal to one. Thus, stability analysis involves setting up the matrices \( \hat{B} \) and \( \hat{C} \) 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 ZW-Transform of the two dimensional filter may be expressed as the product

\[
H(z,w) = H_1(z)H_2(w) \quad (4.1)
\]

or as the sum

\[
H(z,w) = H_1(z) + H_2(w) \quad (4.2)
\]

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 Transform (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.

$$H(r,\theta) = \frac{1}{1 + \left(\frac{r}{R}\right)^2}$$  \hspace{1cm} (4.3)

where $r$ and $\theta$ 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 $H(r,\theta)$ to obtain

$$H1(r,\theta) = \frac{1}{1 + \left(\frac{a}{R}\right)^2}$$  \hspace{1cm} (4.4)

where

$$a = \left(\frac{2RT}{2}\right)^2$$  \hspace{1cm} (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 for the squared magnitude characteristic of the desired filter.

$$H(z,w) = \frac{1}{1 + a^2(z^2 + w^2)^n}$$  \hspace{1cm} (4.6)

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

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

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

1. Construct the coefficient matrices $\hat{b}$ and $\hat{c}$ of (3.1) which corresponds to (4.6).
2. Calculate the eigenvalues of the matrix sum $(\hat{b} + \hat{c})$; 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 $w$ for the denominator polynomial and by using the minimum phase version of the numerator polynomial.

The resulting filter ZW-Transform is given by

$$H(z,w) = \frac{1 + p^2(z+1)(w+1)}{4(z+p)(w+p)}$$  \hspace{1cm} (4.7)

where

$$p = \frac{2a - (2RT)^2a + 1}{1 - 2a}$$  \hspace{1cm} (4.8)

5.0 FILTER DESIGN
5.1 Low Pass filter

Equation (4.7) gives the ZW-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 $H(z,w)$ 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 zeros 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:

$$H(z,w) = \prod_{i=1}^n \frac{1}{\left[z^2 + q(i)z + q(2i)\right]}$$  \hspace{1cm} (5.1)

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

$$LP(z,w) = \frac{(1+p)^2(z^2+2z+1)(w^2+2w+1)}{16(z^2+2zp+p^2)(w^2+2wp+p^2)}$$  \hspace{1cm} (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:

\[ H(z, w) = c + d|LP(z, w)|^2 \]  
\[ \text{(5.3)} \]

Note that (5.3) has roots of \( z \) and of \( w \) 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:

\[ V(z, w) = \frac{H(z, w)}{N(z, w)} \]  
\[ \text{(5.4)} \]

where \( H(z, w) \) and \( N(z, w) \) have minimum phase.

A 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. If 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:

\[ c = 1.0 \]
\[ d = BF - 1.0 \]  
\[ \text{(5.5)} \]

If a high frequency boost filter 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:

\[ c = BF \]
\[ d = 1.0 - BF \]  
\[ \text{(5.6)} \]

The shape of the resulting filter is affected by the value of the root \( p \) which is derived from the design of the low pass filter. From (4.7) and (4.9), 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 cutoff 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 frequency boost approaches the Nyquist frequency. This is caused by 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

ORIGINAL PAGE IS OF POOR QUALITY
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:

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

where $1 \leq p \leq N$, $1 \leq q \leq M$, 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:

$$B_{pq} g_k = A_{pq} f_k$$  \hspace{1cm} (2)

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

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 $g_{pq}$, 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:

$$g_{uv} = C_{uv} A_{pq} f_k$$  \hspace{1cm} (3)

Tensor equation (2) can also be interpreted as a matrix equation with the $A_{pq}$, $B_{pq}$ taken as $NM \times NM$ 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 $a_{0,0} \neq 0$; then equation (2) is also expressible as a matrix equation involving only $NxN$ matrices given by:

$$LGR + \sum_{k=-m, k \neq 0}^{m} S_k G T_k = c \, PFQ + c \sum_{k=-m, k \neq 0}^{m} S_k F \, U_k$$  \hspace{1cm} (4)

where $c = a_{0,0}/b_{0,0}$, the matrices $G = [g_{pq}]$, $F = [f_{pq}]$; and the non-zero components of the coefficient matrices $L$, $R$, $P$, $Q$, $S_k$, $T_k$ and $U_k$ given by:

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

$$L_{pq} = b_{q-p,0}/b_{0,0}; \hspace{1cm} R_{pq} = b_{0,q-p}/b_{0,0}; \hspace{1cm} P_{pq} = a_{q-p,0}/a_{0,0}; \hspace{1cm} Q_{pq} = a_{0,0}, q-p/a_{0,0}.$$  

$$T_{kpq} = b_{k,p-q}/b_{0,0} - b_{k,0} b_{p-q}/b_{0,0}^2; \hspace{1cm} U_{kpq} = a_{k,p-q}/a_{0,0} - a_{k,0} a_{p-q}/a_{0,0}^2.$$  

(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.