Two Dimensional Recursive Digital Filters
for Near Real Time Image Processing

Final Report

NASA Grant NSG 5406

Period September 15, 1979 to December 30, 1980

By

Dr. Winser E. Alexander - Principal Investigator

Dr. David Olson

Mr. Earnest Sherrod

(NASA-CR-164028) TWO DIMENSIONAL RECURSIVE DIGITAL FILTERS FOR NEAR REAL TIME IMAGE PROCESSING Final Report, 15 Sep. 1979 - 30 Dec. 1980 (North Carolina Agricultural and Technical) 194 p HC A09/MF A01 CSCL 09A G3/33

N81-19398

Unclas
41619

Department of Electrical Engineering

North Carolina Agricultural and Technical

State University

Greensboro, North Carolina 27411

Table of Contents

	Page
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 Demoninators

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 algorithims 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 occured 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

$$F(z,w) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(mX,nY) z^{-m} w^{-n} \tag{3.1}$$

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

$$F(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} f(m,n) z^{-m} w^{-n} \quad (3.2)$$

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

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

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

$$H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n) z^{-m} w^{-n} \quad (3.4)$$

In general, the ZW-Transform for the impulse response is an infinite series. In order to implement the spatial domain filter, we must find a closed form expression for H(z,w) such that

$$H(z,w) = \frac{\sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} z^{-J} w^{-K}}{\sum_{J=0}^{L} \sum_{K=0}^{L} b_{JK} z^{-J} w^{-K}} \tag{3.5}$$

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

$$G(z,w) = H(z,w)F(z,w) \tag{3.6}$$

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

$$g(m,n) = \sum_{J=0}^{L} \sum_{K=0}^{L} a_{JK} f(m-J,n-K) - \sum_{\substack{J=0 \\ J+K>0}}^{L} \sum_{K=0}^{L} b_{JK} g(m-J,n-K) \tag{3.7}$$

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 straight forward. 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 straight forward for the two dimensional problem because a two variable polynomial is not generally factorable into distinct roots. When the polynomial in the denominator of the two dimensional Z-Transform of the impulse response is factorable into distinct roots, the stability analysis procedure is the same as for the one dimensional problem.

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

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

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

5.0 SYNTHESIS

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

Often it is possible to express a desired two dimensional recursive digital filter as the product or sum of two one dimensional digital filters. That is the two dimensional Z-Transform of the digital recursive filter can be expressed as the product or sum of two one dimensional Z-Transforms. In either case, the two dimensional synthesis problem is reduced to the synthesis of two one dimensional filters. However, it is not possible to design sum separable or product separable digital recursive filters for all applications. For these applications, the design of the required two dimensional digital recursive filter is considerably more complicated.

Many imaging systems have a natural circular symmetry. In general, the optical transfer function of a circularly symmetric imaging system is circularly symmetric. Also, it is usually desirable to perform image processing where the processing is uniform with respect to direction. The natural consequence is that filters with circularly symmetric impulse response functions are generally very desirable for image processing. The relationship between circular symmetry of the impulse response and the frequency response dictates that the design requirement is for these filters to have a circularly symmetric frequency response [10].

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

In the design procedure, the squared magnitude characteristic of the desired circularly symmetric filter is chosen in the Laplace Transform domain. The bilinear transformation is then used to map the squared magnitude characteristic into the two dimensional Z-Transform domain. The pseudo-state space representation for the corresponding two dimensional Z-Transform is formed. The eigenvalues of the matrix sum of the two pseudo-state transition matrices are obtained. These eigenvalues occur in reciprocal pairs. The eigenvalues with magnitudes less than one are then used as roots of a denominator polynomial with distinct roots to form the two dimensional Z-Transform of the desired filter.

Note that this design procedure always ensures a stable filter. Stability analysis is simple because the denominator of the ZW - Transform is a product separable. Also note that no restrictions are placed on the numerator polynomial. That is, it is not necessary for the numerator polynomial to either be product separable, sum separable or minimum phase. Examples of stable two dimensional recursive filter designs are given in [12].

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

6.0 IMPLEMENTATION

6.1 Implementation Considerations

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

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

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

This procedure adds two transpose operations to each filtering operation. The result is that in the practical use of filtering large images, recursive digital filters are very significantly more efficient and require far less time to implement than FFT filters.

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

6.2 Transient Response

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

The approach use to minimize this problem is to place the filter in a stable state with an assumed input within the range of the image data. The best assumed input would be the expected value of the input image intensity.

However, this is usually not available. An approximation is obtained by averaging the intensity values of the middle row of the image. The final value theorem [5] is then used to determine the stable state for each of the output stages for the filter. The expected values approximation is then used as the initial condition input for each scan line and the stable state output is used as the initial condition output for each filter stage. Thus, if the initial input is the same as the assumed initial condition, then no transient response occurs.

In practice, the procedure outlined above is simple to implement and 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 straight forward 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 magitude 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.

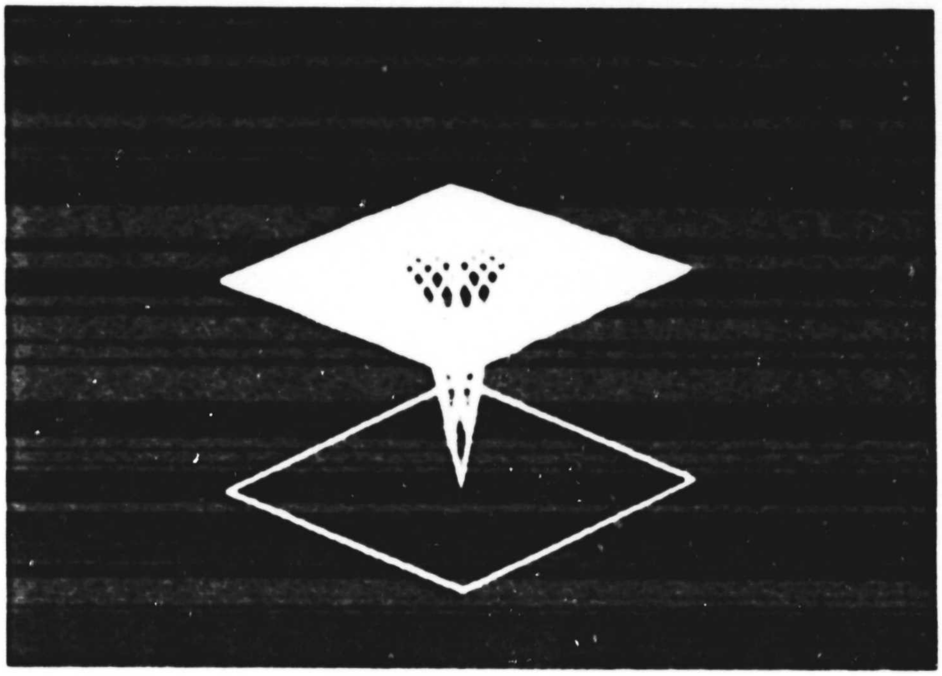

Figure 1. Perspective plot 5x-0.2 High Boost Filter

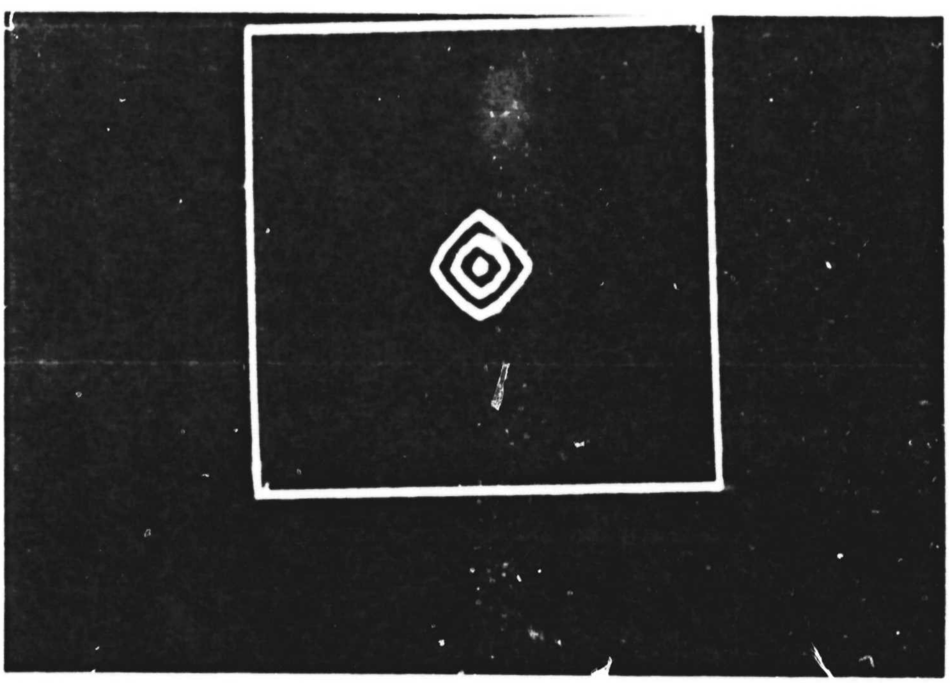

Figure 2. Contour plot 5x-0.2 High Boost Filter

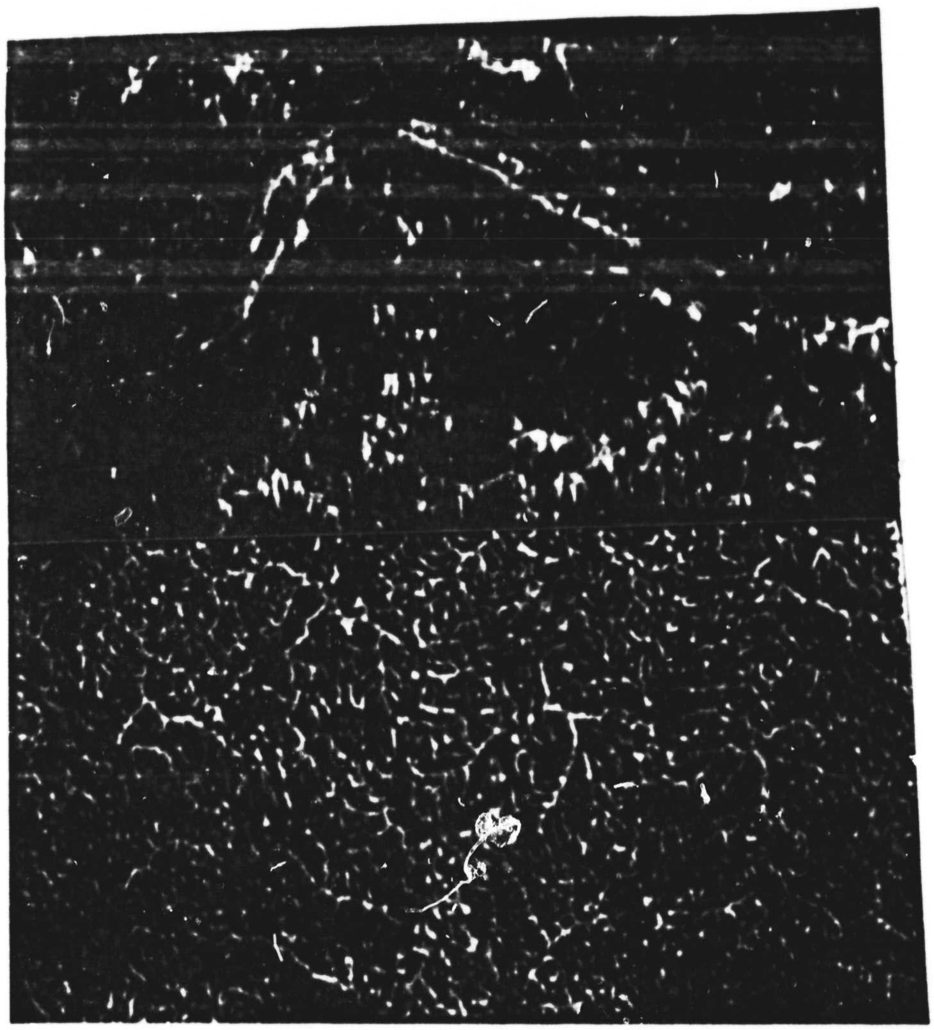

Figure 3. Original Landsat File-3 Image

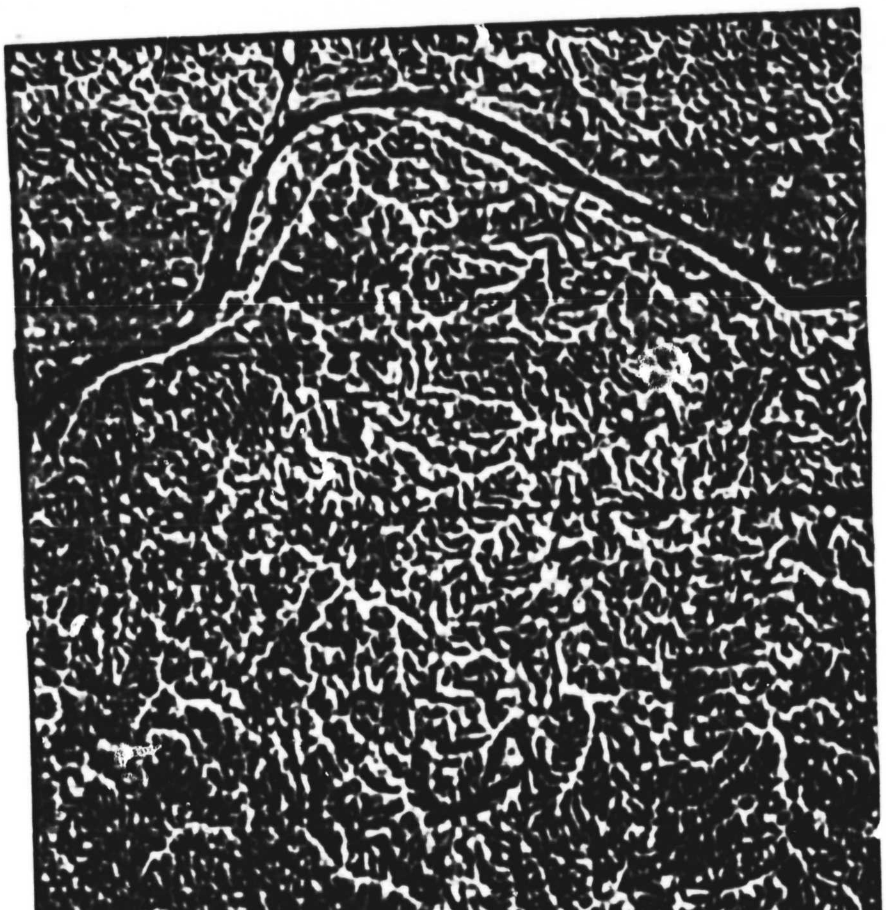

Figure 4. Enhanced Landsat file-3 Image

12.0 REFERENCES

1. Earnest L. Hall, "A Comparison of Computations for Spatial Filtering", Proceedings of the IEEE, Vol. 60, no. 7, 1972, pp 887-891.

2. Winser E. Alexander and Steven A. Pruess, "Stability Analysis of Two Dimensional Digital Recursive Filters", IEEE Transactions on Circuits and Systems, Vol. , No. 1, 1980, pp.

3. Winser E. Alexander and William J. Craft, Documentation for Spatial Domain Filtering Package, Department of Electrical Engineering, North Carolina A & T State University, January, 1979.

4. Lawrence R. Rabiner and Bernard Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1975, pp 442-455.

5. Samuel Stearns, Digital Signal Analysis, Hayden Publishing Co., Inc., Edison, N. J.

6. N. K. Bose, "Problems and Progress in Multidimensional System Theory", Proceedings of the IEEE, Vol. 65, No. 6, 1977, pp. 824-840.

7. Katsuhiko Ogata, State Space Analysis of Control Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., p. 487.

8. E. Fornasini and G. Marchesini, "State Space Realization Theole for Two Dimensional Filters", IEEE Transactions on Automatic Control, Vol. AC01t 1976, pp 484-492.

9. E. L. Hall, Digital Filtering of Images, Ph.D. Dissertation, University of Missouri, Columbia, Mo., 1971.

10. A. Papoulis, Systems and Transforms with Applications in Optics, McGraw-Hill Book Co. New York, N. Y., 1968, p. 140.

11. Winser E. Alexander, A Study of Two Dimensional Recursive Digital Filters, Final Report (Naval Air Systems Commd Contract No. NO-14-77-C-0199), School of Engineering, N. C. A T State University, Greensboro, N. C., November, 1978.

12. Winser E. Alexander and Earnest E. Sherrod, "Two Dimensional Recursive Digital Filters for Subjective Image Processing", 13th Asilomar Conference on Circuits, Systems and Computers, November, 1979.

13. J. M. Costa and A. N. Venetsonopoulos, "Design of Circularly Symmetric Two Dimensional Recursive Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-22, No. 6, 1974, pp 432-442.

14. Dennis Goodman, "A Design Technique for Circularly Symmetric Low Pass Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 4, 1978, pp 290-304.

15. B. R. Hunt, "Data Structures and Computational Organization in Digital Image Enhancement", Proceedings of IEEE, Vol. 60, 1972, pp 884-887.

16. William K. Pratt, Digital Image Processing, John Wiley and Sons, Inc., Somerset, N. J., 1978.

17. Winser E. Alexander, "Electronic Target Enhancement in Infrared Reconnaisance", Proceedings of the 1968 Air Force Science and Engineering Symposium, October, 1968.

18. Thomas G. Stockham, Jr. "Image Processing in the Context of a Visual Model", Proceedings of the IEEE, Vol. 60, 1972, pp 828-842.

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 resursion. 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:

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

where $1 \le p \le N$, $1 \le q \le M$, $-m \le i \le m$, $-m \le j \le m$; 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 $i,j \ne 0$. A more formal tensor expression for (1) is given by:

$$B_{pq}^{kl}\, g_{kl} = A_{pq}^{kl}\, f_{kl} \qquad (2)$$

where $1 \le k \le N$, $1 \le l \le M$, and the non-zero components of the coefficient tensors given by $A_{pq}^{kl} = a_{k-p,l-q}$ and $B_{pq}^{kl} = b_{k-p,l-q}$; for $-m \le k-p \le m$ and $-m \le l-q \le m$. 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}^{kl}, say C_{uv}^{pq}; with $1 \le u \le N$, $1 \le v \le M$. For such a case, the filtered solution would then be given by:

$$g_{uv} = C_{uv}^{pq}\, A_{pq}^{kl}\, f_{kl} \qquad (3)$$

Tensor equation (2) can also be interpreted as a matrix equation with A_{pq}^{kl}, and B_{pq}^{kl} taken as NM by NM 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_{kl} requires no inversion of the coefficient matrix. By a simple back substitution process, starting at one corner of the array and proceeding by rows or columns, the filtered array may be computed provided that the iteration process is numerically stable.

(b) For the "Symmetric" diagital 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 sparce coefficient matrices given by:

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

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

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

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

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

(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 g_{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:

$$LGR = c\ PFQ \qquad (5)$$

and the solution for the filtered output G given by:

$$G = L^{-1}(cPFQ)\ R^{-1} \qquad (6)$$

At first glance it would appear the 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:

$$L_u L_1\ G\ R_u R_1 = c\ PFQ \qquad (7)$$

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:

$$G^{(n+1)} = L^{-1}\ \left(\sum_{k=-m,k\neq 0}^{m}\ S_k G^{(n)} T_k\right)\ R^{-1} + H \qquad (8)$$

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

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

[1]. D. E. Olson, W. E. Alexander and E. E. Sherrod, "Simultaneous Linear Albegraic Equation Formulations of Two Dimensional Digital Filter Realizations," Eighteenth Annual Allerton Conference on Communications, Control, and Computing Proceedings, October 1980.

[2]. R. M. Mersereau, and D. E. Dudgeon, "Two-Dimensional Digital Filtering," IEEE Proc. 63(4): 610-623 (1975).

[3]. S. K. Mitra, A. D. Sagar, and N.A. Pendergrass, "Realizations of Two Dimensional Recursive Digital Filters," IEEE Trans. Circuits Syst. CAS-22(3): 177-184 (1975).

[4]. E. L. Hall, "A Comparison of Computations for Spatial Filtering," IEEE Proc. 65(6), June, 1977.

[5]. S. Zohar, "Toeplitz Matrix Inversion: The Algorithm of W. F. Trench," Journal of the Assoc. for Computing Machinery, Vol. 16, No. 4, October 1969: 592-601.

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

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

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

$$D(Z,W) = \left[\sum_{J=0}^{2} c_J Z^{-J}\right] \left[\sum_{K=0}^{2} d_K W^{-K}\right] \qquad (2)$$

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

$$H(Z,W) = H_1(Z,W)H_2(Z,W)H_3(Z,W) \tag{3}$$

where

$$H_1(Z,W) = \sum_{J=0}^{2} \sum_{K=0}^{2} a_{JK}Z^{-J}W^{-KU} \tag{4}$$

$$H_2(Z,W) = 1/\sum_{J=0}^{2} c_J Z^{-J} \quad ; \quad c_0 = 1 \tag{5}$$

$$H_3(Z,W) = 1/\sum_{K=0}^{2} d_J W^{-K} \quad ; \quad d_0 = 1 \tag{6}$$

In direct form, the corresponding difference equations are given by

$$X_1(m,n) = \sum_{J=0}^{2} \sum_{K=0}^{2} a_{JK}\, f(m-J,n-K) \tag{7}$$

$$X_2(m,n) = X_1(m,n) - c_1X_2(m-1,n) - c_2X_2(m-2,n) \tag{8}$$

$$g(m,n) = X_2(m,n) - d_1X_3(m,n-1) - d_2X_3(m,n-2) \tag{9}$$

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.

$X_1(m,n)$ $X_2(m,n)$ $X_3(m,n)$

f(m,n) ====> [$H_1(Z,W)$] ====> [$H_2(Z,W)$] ====> [$H_3(Z,W)$] ====> g(m,n)

APPENDIX C

PROGRAM NAME: NASA TYPE: Transfer

PROGRAMMER: W.E. ALEXANDER

Source: Reloc:

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

FROM RTE: Run NASA

Modules Called:

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

Modules Run: NASA1

Subroutines Called:

PROGRAM NAME: NASA1 TYPE: Program

PROGRAMMER: W.E. ALEXANDER

SOURCE: &NASA1 Reloc: %NASA1

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

Modules Called:

DSPLY
FDIGN
FILTR
RESIZ
SHOW
BLDIM
NOISE

Modules Run:

Subroutines Called:

FILL

PROGRAM NAME: DSPLY TYPE: Program

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 TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &FDIGN, &FDIG1 Reloc: %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 TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &STABI Reloc: %STABI

FUNCTION: This Program evaluates the Recursive Filter Stability Characteristics.

Modules Called:

Subroutines Called:

STABT
PRTN

PROGRAM NAME: DPLAM TYPE: Program

Source: &DPLAM, &DPLA1 Reloc: %DPLAM, %DPLA1

PROGRAMMER: E.E. SHERROD

FUNCTION: This program displays the Filter Characteristics.

Modules Called:

COEFFS
DPLA1

Subroutines Called:

ZWC
CONTR
SET3D
PLT3D
SET2D
PLT2D

PROGRAM NAME: FIRO

TYPE: PROGRAM

PROGRAMMER: E.E. SHERROD

Reloc: %FIRO

Source: &FIRO

FUNCTION: This program designs Non-Recursive FIR Filters.

Modules Called:

Subroutines Called:

BESJ
BESIO

PROGRAM NAME: PLOTV TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &EES3 Reloc: %PLOTV

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

Modules Called:

DATA1

Subroutines Called:

DVECT

PROGRAM NAME: FILTR TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &FILTR Reloc: %FILTR

FUNCTION: This program schedules Linear or Homorphic filtering of Images.

Modules Called:

LFLTR
HFLTR
SHOW
BLDWF

Subroutines Called:

PROGRAM NAME: BLDWF TYPE: Program

PROGRAMMER: DAVE JOHNSON

Source: &BLDWF Reloc: %BLDWF

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

Modules Called:

DIREC
WF0000

Subroutines Called:

ICMPW

PROGRAM NAME: LFLTR TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &LFLTR Reloc: %FILTR

FUNCTION: This program does Linear Filtering using Spatial Domain Recursive Digital Filters.

Modules Called:

COEFFS

Subroutines Called:

READL
RITLN
FILTR
WFINT
CLSWF

PROGRAM NAME: HFLTR 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 TYPE: Program

PROGRAMMER: W.E. ALEXANDER and RICHARE MUORE

Source: &RESIZ R ELOC: %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 TYPE: Program

PROGRAMMER: DAVE JOHNSON

Source: &SHOW Reloc: %SHOW

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

Modules Called:

WF0000

Subroutines Called:

READL
WLINE
CLSWF

PROGRAM NAME: BLDIM TYPE: Program

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 TYPE: Program

PROGRAMMER: E.E SHERROD

Source: &NOISE Reloc: %NOISE

FUNCTION: This program add 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

APPENDIX D

LOAD FILES

Table of Content:

LDIREC---1
LWTAPE---1
LDPLAM---1
LPLOTV---1
LFDIGN---2
LBLDIM---2
LLFTR--2
LHFTR--3
LSHOW--3
LFIRO--3
LIMAGE---4
LRESIZ---4
LDINTP---4
LBLDWF---5
LDSPLY---5
LCURSR---5
NASA---6

PROGRAM FILES

Table of Content:

&NASA1--7
&DSPLY--9
&CURSR--12
&ICMPW--13
&RESET--13
&RLINE--14
&MOVEC--15
&IMAGE--16
&SPAGE--23
&RESIZ--24
&ROT8--33
&TRMGN--34
&LBRSZ--37
&WFINT--39
&DINTP--42
&WTAPE--44
&EES3--47
&DPLAM--48
&DPLA1--52
&FDIGN--60
&BLDIM--70
&LFLTR--76
&HFLTR--82
&SHOW--88
&FIRO--89
&WINDO--92
&BESIO--92
&BESJ--93
&BLDWF--95
&SCROL--98
&WLINE--99
&DRIVE--100
&FDIG1--102
&STABI--105
&FILTR--117

LDIREC T=00004 IS ON CR00025 USING 00002 BLKS R=0002

```
:PU,IMDIRC:IM:23:4:100
:CR,IMDIRC:IM:23:4:100
```

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

```
:LG,3
:SV,0
:OF,WTAPE
:PU,WTAPE
:MR,%WTAPE
:MR,%ROT8
:MR,%ICMPW
:MR,%TRMGN
:RU,LOADR,99,0G,,,2
:SP,10G::3
:TR
```

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

```
:SV,0
:OF,DPLAM
:PU,DPLAM
:LG,3
:MR,%DPLAM
:MR,%DPLA1
:RU,LOADR,99,0G
:SP,10G::3
:OF,10G
:RP,10G::3
::
```

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

```
:SV,0
:OF,PLOTV
:PU,PLOTV
:LG,3
:MR,%EES3
:MR,%DRIVR
:RU,LOADR,99,0G
:SP,10G::3
:OF,10G
:RP,10G::3
::
```

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

```
:SV,0
:OF,FDIGN
:PU,FDIGN
:LG,3
:MR,%FDIGN
:MR,%FDIG1
:RU,LOADR,99,1G
:PU,10G::3
:SP,10G::3
:OF,10G
::
```

LBLDIM T=00004 IS ON CR00022 USING 00002 BLKS R=0009

```
:LG,2
:OF,BLDIM
:PU,BLDIM
:MR,%BLDIM
:MR,MVW.
:MR,%ROTS
:MR,%RLINE
:MR,DCODE.
:MR,%DRIVR
:RU,LOADR,99,OG,,,2
:SP,10G::3
:OF,10G::3
:RP,10G::3
::
```

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

```
:SV,0
:OF,LFLTR
:PU,LFLTR::3
:LG,3
:MR,%LFLTR
:MR,%WFINT
:MR,DCODE.
:MR,%WLINE
:MR,%DRIVR
:RU,LOADR,99,1G
:SP,10G::3
:OF,10G
:RP,10G::3
::
```

LHFTR T=00004 IS ON CR00022 USING 00002 BLKS R=0014

```
:SV,0
:OF,HFLTR
:PU,HFLTR::3
:LG,3
:MR,%HFLTR
:MR,%WFINT
:RU,LOADR,99,1G
:SP,1OG::3
:OF,1OG
:RP,1OG::3
::
```

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

```
:LG,1
:MR,%SHOW
:MR,%WFINT
:MR,%DRIVR
:MR,%WLINE
:OF,SHOW
:RU,LOADR,99,1G
:PU,1OG::3
:SP,1OG::3
:OF,1OG
::
```

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

```
:LG,1
:MR,%FIRO
:MR,%WINDO
:MR,%BESIO
:OF,FIRO
:RU,LOADR,99,OG
:PU,1OG::3
:SP,1OG::3
:OF,1OG
:RP,1OG::3
::
```

LIMAGE T=00004 IS ON CR00022 USING 00002 BLKS R=0009

```
:LG,1
:OF,IMAGE
:MR,%IMAGE
:MR,%SPACE
:MR,%ICMPW
:MR,%ROT8
:RU,LOADR,99,1G
:PU,1OG::3
:SP,1OG::3
:OF,1OG
```

LRESIZ T=00004 IS ON CR00022 USING 00002 BLKS R=0004

```
:LG,3
:SV,0
:OF,RESIZ
:PU,RESIZ
:MR,%RESIZ
:MR,%DRIVR
:MR,%WLINE
:MR,%TRMGN
:MR,%LBRSZ
:MR,%WFINT
:RU,LOADR,99,OG,,,2
:SP,1OG::3
:TR
```

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

```
:LG,3
:SV,0
:OF,DINTP
:PU,DINTP
:MR,%DINTP
:MR,%DRIVR
:MR,%WLINE
:MR,%TRMGN
:MR,%LBRSZ
:MR,%WFINT
:RU,LOADR,99,OG,,,2
:SP,1OG::3
:TR
```

LBLDWF T=00004 IS ON CR00022 USING 00002 BLKS R=0009

```
:LG,0
:LG,2
:OF,BLDWF
:PU,BLDWF
:MR,%BLDWF
:MR,%ICMPW
:RU,LOADR,99,0G,,,2
:SP,10G::3
:OF,10G::3
:RP,10G::3
::
```

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

```
:LG,1
:MR,%DSPLY
:MR,%SCROL
:MR,%WLINE
:MR,%DRIVR
:MR,%RESET
:OF,DSPLY
:RU,LOADR,99,0G
:PU,10G::3
:SP,10G::3
:OF,10G
```

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

```
:LG,1
:MR,%CURSR
:MR,%WLINE
:MR,%RLINE
:MR,%DRIVR
:MR,%MOVEC
:OF,CURSR
:RU,LOADR,99,1G
:PU,10G::3
:SP,10G::3
:OF,10G
::
```

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

```
:LG,1
:MR,%NOISE
:MR,%BLDWF
:MR,%WFINT
:MR,%ICMPW
:OF,NOISE
:RU,LOADR,99,1G
:PU,1OG::3
:SP,1OG::3
:OF,1OG
::
```

NASA T=00004 IS ON CR00022 USING 00002 BLKS R=0008

```
:SV,4
:OF,SHOW
:RP,SHOW
:OF,BLDWF
:RP,BLDWF
:OF,BLDIM
:RP,BLDIM
:OF,WTAPE
:RP,WTAPE
:OF,DSPLY
:RP,DSPLY
:OF,CURSR
:RP,CURSR
:OF,FDIGN
:RP,FDIGN
:OF,STABI
:RP,STABI
:OF,DPLAM
:RP,DPLAM
:OF,FILTR
:RP,FILTR
:OF,LFLTR
:RP,LFLTR
:OF,PLOTV
:RP,PLOTV
:OF,HFLTR
:RP,HFLTR
:OF,RESIZ
:RP,RESIZ
:OF,IMAGE
:RP,IMAGE
:OF,DINTP
:RP,DINTP
:OF,NOISE
:RP,NOISE
:MC,23
:SV,0
:RU,NASA1
:OF,SHOW
:OF,BLDWF
:OF,DSPLY
:OF.CURSR
:OF,FDIGN
:OF,STABI
:OF,DPLAM
:OF,FILTR
:OF,LFLTR
:OF,PLOTV
:OF,HFLTR
:OF,RESIZ
:OF,IMAGE
:OF,BLDIM
:OF,DINTP
:OF,WTAPWTAPE
```

```
&NASA1 T=00004 IS ON CR00022 USING 00008 BLKS R=0054

FTN4,L
      PROGRAM NASA1
C     THIS PROGRAM IS THE FAT'HER PROGRAM FOR THE IMAGE FILTERING
C          PROGRAMS
C
      DIMENSION IPRAM(5),NAME(3),NSON(3,8),IMESS(30)
      DATA NSON/2HDS,2HPL,2HY ,2HFD,2HIG,2HN ,2HFI,2HLT,
     *2HR ,2HRE,2HSI,2HZ ,2HSH,2HOW,2H  ,2HIM,2HAG,2HE ,
     *2HNO,2HIS,2HE /
C
C     SON PROGRAM NAMES (FILES SAME PRESEDED WITH "&")
C         DSPLY - DISPLAY PROGRAM
C         FDIGN - FILTER DESIGN MODULE
C         FILTR - FILTER IMPLEMENTION MODULE
C         RESIZ - IMAGE MODIFICATION MODULE
C         SHOW -  DISPLAYS WORK FILE
C         IMAGE - IMAGE DATA MANAGEMENT MODULE
C         NOISE - ADDITIVE GAUSSIAN NOISE
C
      CALL RMPAR(IPRAM)
      LU=IPRAM(1)
      IF(LU.LE.0) LU=1
C
C     NPRG IS THE NUMBER OF SONS
C
      NPRG=6
      ICNT=9
C
C     DISPLAY MENU
C
    5 WRITE(LU,30)
   30 FORMAT(" SELECT PROCESSING OPTION"/,"  1.  IMAGE DISPLAY"/,"
     *FILTER DESIGN"/,"  3.  FILTER IMAGE"/,"  4.  MODIFY IMAGE"/,
     * SHOW WORK FILE"/,"  6.  IMAGE DATA MANAGEMENT"/,"  7.  NOIS
     *ON"/,"  8.  TERMINATE PROGRAM")
      READ(LU,*) IOPT
      IF(IOPT.EQ.0.OR.IOPT.EQ.1) IOPT = 1
      IF(IOPT.LT.1.OR.IOPT.GT.8) GO TO 16
      IF(IOPT.EQ.8) GO TO 500
C
      IPRAM(2)=IOPT
      DO 10 I=1,3
   10 NAME(I)=NSON(I,IOPT)
      WRITE(LU,15) NAME
   15 FORMAT(" MODULE TO BE SCHEDULED IS ",3A2)
      GO TO 20
   16 WRITE(LU,17)
   17 FORMAT(" INVALID RESPONSE")
      GO TO 5
C
```

```
C
   20 ICNW=LU+200B
      CALL EXEC(13,ICNW,IPRAM(3),IPRAM(4),IPRAM(5))
      CALL EXEC(23,NAME,IPRAM(1),IPRAM(2),IPRAM(3),IPRAM(4),IPRAM(
      WRITE(LU,40) (IPRAM(I),I=1,5)
   40 FORMAT("PARAMETERS RETURNED FROM MODULE"/,5(1H,4E11.3,2X))
      GO TO 5
  500 CONTINUE
C
C     OF ALL SON PROGRAMS
C
      DO 510 I=1,NPRG
      CALL FILL(IMESS,2H  ,30)
      CALL CODE
      WRITE(IMESS,520) (NSON(J,I),J=1,3)
  520 FORMAT("OF,",3A2)
      IRTN=MESSS(IMESS,ICNT,LU)
      IF(IRTN.LT.0) CALL EXEC(2,LU,IMESS,IRTN)
  510 CONTINUE
C
      STOP
      END
C
C
      SUBROUTINE FILL(IARAY,IA,N)
C
C     THIS SUBROUTINE FILLS ARRAY IARAY WHICH HAS N WORDS WITH THE
C          OF IA.
C
      DIMENSION IARAY(N)
      DO 10 I=1,N
   10 IARAY(I)=IA
      RETURN

$     END
```

```
FTN4
      PROGRAM DSPLY
C
C  THIS PROGRAM DISPLAYS AN IMAGE ON THE GMR-27.  IMAGE FILE MUST
C  BE IN FORMAT DESCRIBED BY IMAGE DISPLAY SUBSYSTEM.
C
      INTEGER SLU11,STRTL,STRTP,SCROL
C
      DIMENSION NAME(6),IDCB(144),IBLK(513),ISET(10),LU(5),JNAME(3
      INTEGER TEXT1(38),TEXT2(38),TEXT3(38)
C
      EQUIVALENCE (IBLK(7),IBLK7),(IBLK(8),IBLK8),(IBLK(12),IBK12)
     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)
C
      DATA ISET/100377B,10377B,24001B,30000B,5*-1,26002B/
      DATA SLU11/34011B/
      DATA LLA0,LEA0,LEC0,LLB1,LLBX,LEB1,LEBX/64000B,44000B,54000B
     1 70001B,71777B,50001B,51777B/
C
C
C GET INPUT PARAMETERS
C
      CALL RMPAR(LU)
      IF (LU .LE. 0) LU = 1
C
C OPEN IMAGE DIRECTORY FILE
C
100   CALL OPEN(IDCB,IERR,6HIMDIRC)
      IF (IERR .LT. 0) GO TO 991
C
C GET IMAGE FILE NAME
C
      CALL RESET(LU)
      WRITE(LU,20)
C
      WRITE(LU,21)
      WRITE(LU,22)
      WRITE(LU,23)
      WRITE(LU,24)
      WRITE(LU,26)
      WRITE(LU,26)
      WRITE(LU,26)
 20   FORMAT(20X,"I M A G E   D I S P L A Y   S Y S T E M"//)
 21   FORMAT("IMAGE NAME: dB          d@"/)
 22   FORMAT("# LINES:    dB    d@",20X,
     1"# PIXELS/LINE: dB    d@"/)
 23   FORMAT("MIN PIXEL:  dB       d@",18X,
     1"MAX PIXEL:     dB      d@"/)
 24   FORMAT("TEXT:",/)
 26   FORMAT("dB",38"  ","d@")
 25   FORMAT("_")
 27   FORMAT("‾")
105   WRITE(LU,25)
      READ(LU,2) NAME
  2   FORMAT(6A2)
      IF (NAME .EQ. 2H/E) GO TO 9000
```

```
C
C  FIND IMAGE FILE
C
        CALL RWNDF(IDCB)
  110   CALL READF(IDCB,IERR,IBLK,256,LEN)
        IF (IERR .LT. 0) GO TO 991
        IF (LEN .EQ. -1) GO TO 800
C
        DO 120 I=1,6
        IF (IBLK(I) .NE. NAME(I)) GO TO 110
  120   CONTINUE
C
C  IMAGE FOUND--CHECK IF ON DISC
C
        IF (IBK12 .EQ. 1) GO TO 130
C
C  IMAGE NOT ON DISC
C
        WRITE(LU,12)
  12    FORMAT("
        GO TO 105
C
C IMAGE IS ON DISC
C
  130   CALL CLOSE(IDCB)
        RMIN = IBLK(9)
        RMAX = IBLK(10)
        WRITE(LU,28)(IBLK(I),I=7,10)
  28    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
C  EXTRACT DISPLAY INFORMATION
C
        NUML = IBLK7
        NUMP = IBLK8
        STRTL = (256-MIN0(256,NUML))/2
        STRTP = (512-MIN0(512, NUMP))/2
C
  500   ISET5 = IOR(LLA0,IAND(STRTL,1777B))
        ISET6 = IOR(LEA0,IAND(STRTP,1777B))
        ISET7 = LLB1
        ISET8 = LEB1
        ISET9 = IOR(LEC0,IAND(STRTP,1777B))
C
        CALL DRIVR(2,ISET,10)
C
```

```
      IERR = 0
      DO 600 I=1,MINO(NUML,256)
      IF (IERR .LT. 0) GO TO 991
      CALL READF(IDCB,IERR,IBLK,512,NUM)
      IF (NUM .LT. 0) GO TO 600
C
      DO 595 J=1,NUM
      IBLK(J) = (255./(RMAX-RMIN))*(FLOAT(IBLK(J))-RMIN)
      IF (IBLK(J) .LT. 0) IBLK(J) = 0
      IF (IBLK(J) .GT. 377B) IBLK(J) = 377B
595   CONTINUE
C
      IBLK(NUM+1) = SLU11
      CALL DRIVR(40002B,IBLK,NUM+1)
 600  CONTINUE
      IFRST = 0
      ILAST = 255
C
C
C  OUTPUT SOFT KEY FUNCTIONS

      WRITE(LU,29)
29    FORMAT(/"FUNCTION KEYS:"/)
      WRITE(LU,30)
      WRITE(LU,30)
30    FORMAT(4("dB              d@    ")/)
605   WRITE(LU,31)
31    FORMAT("
      WRITE(LU,32)
32    FORMAT(" << SCROLL   SCROLL >>                CURSOR   ",
     124X," NEW IMAGE       EXIT   ")
610   CALL EXEC(1,LU,INPT,1)
      INPT = INPT-7023
      IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 610
C
C   BRANCH TO APPROPRIATE SECTION
C
C
      GO TO (1000,2000,3000,4000,5000,6000,100,9000),INPT
C
C
C
C  SCROLL IMAGE BACK
C
1000  IERR = SCROL(IDCB,-9,NUML,IFRST,ILAST,RMAX,RMIN)
      IF (IERR .LT. 0) GO TO 991
      GO TO 610
```

```
C
C  SCROLL FORWARD
C
2000  IERR = SCROL(IDCB,17,NUML,IFRST,ILAST,RMAX,RMIN)
      IF (IERR .LT. 0) GO TO 991
      GO TO 610
3000  CONTINUE
C
C  POSITION CURSOR
C
4000  CALL EXEC(23,6HCURSR ,LU)
      GO TO 605
5000  CONTINUE
6000  CONTINUE
      GO TO 610
C
C  TERMINATE
C
9000  CALL CLOSE(IDCB)
      CALL RESET(LU)
      WRITE(LU,33)
33    FORMAT("END PROGRAM")
      CALL EXEC(6)
C
C  FILE NOT FOUND
C
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
$
```

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

```
FTN4
      PROGRAM CURSR
C
      DIMENSION LU(5),IBUF(2352),IZERO(2)
C
      INTEGER EA,LA
C
      DATA IZERO/44000B,64000B/
C
C
      CALL RMPAR(LU)
C
C
C  SAVE IMAGE LINES
C
C
      DO 50 I=0,20
      CALL RLINE(I,0,111,IBUF(112*I+1))
50    CONTINUE
      WRITE(LU,1)
1     FORMAT("
      WRITE(LU,2)
2     FORMAT("    LEFT          UP          RIGHT                    ",
     112X,"    DOWN     ",12X,"    RETURN    ")
      CALL MOVEC(0,255)
      EA = 0
      LA = 255
100   CALL EXEC(1,LU,INPT,1)
      INPT = INPT-7023
      IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 100
C
C
C  BRANCH TO APPROPRIATE SECTION
C
C
      GO TO (400,200,500,100,100,300,100,600), INPT
C
C  MOVE CURSOR UP
C
200   LA = MOD(LA+11,256)
      CALL MOVEC(EA,LA)
      GO TO 100
C
C  MOVE CURSOR DOWN
C
300   LA = MOD(LA+249,256)
      CALL MOVEC(EA,LA)
      GO TO 100
C
C  MOVE CURSOR LEFT
C
400   EA = MOD(EA+499,512)
      CALL MOVEC(EA,LA)
      GO TO 100
C
```

```
C  MOVE CURSOR RIGHT
C
500    EA = MOD(EA+17,512)
       CALL MOVEC(EA,LA)
       GO TO 100
C
C  RETURN TO PREVIOUS SCREEN
C
600    DO 610 I=0,20
       CALL WLINE(I,0,111,IBUF(112*I+1))
610    CONTINUE
C
       CALL DRIVR(2,IZERO,2)
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
200    ICMPW = I
       END
$
```

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

```
FTN4
       SUBROUTINE RESET(LU)
C
C
       WRITE(LU,1)
1      FORMAT("
C
C  WAIT 200 MSEC
C
       CALL EXEC(12,0,1,0,-20)
C
C  CLEAR DISPLAY
C
       WRITE(LU,2)
2      FORMAT("")
       END
$
```

&RLINE T=00004 IS ON CR00022 USING 00005 BLKS R=0039

```
FTN4,L
      SUBROUTINE RLINE(LINE,IPIX,JPIX,IDATA)
C
C   THIS SUBROUTINE READS A LINE FROM GMR-27.
C
C       WHERE
C           LINE = LINE # TO READ
C           IPIX = STARTING PIXEL
C           JPIX = ENDING PIXEL
C
C           IDATA = BUFFER IN WHICH DATA IS RETURNED  (1 PIXEL/WORD
C
C
      DIMENSION IDATA(512),INIT(5)
C
      EQUIVALENCE (LLA,INIT(2)),(LEA,INIT(3)),(LEB,INIT(4))
C
      DATA INIT/100377B,64000B,44000B,50000B,26002B/
C
C  COMPUTE DIRECTION
C
      IDIRC = 1
      IF (IPIX .GT. JPIX) IDIRC = -1
C
C  SET UP FOR READ BACK
C
      LLA = 64000B + IAND(LINE,377B)
      LEA = 44000B + IAND(IPIX,777B)
      LEB = 50000B + IDIRC + 512
      CALL DRIVR(2,INIT,5)
C
C  READ BACK LINE
C
      NUM = IDIRC*(JPIX-IPIX)+1
      CALL DRIVR(1,IDATA,NUM)
C
      RETURN
      END
$
```

&MOVEC T=00004 IS ON CR00022 USING 00004 BLKS R=0031

```
FTN4,L
      SUBROUTINE MOVEC(IX,IY)
C
C  THIS SUBROUTINE MOVES THE CURSOR ON THE GMR-27.  ITS POSITION I
C  INDICATED IN THE LOWER LEFT HAND CORNER OF THE SCREEN.
C
C       IX = X-COORDINATE
C       IY = Y-COORDINATE
C
C
      INTEGER WACO
C
      DIMENSION ICR(7),IP00(5),IPXY(3)
C
      EQUIVALENCE (ICR,ICR1),(ICR(2),ICR2),(ICR(3),ICR3),(ICR(5),I
     1 (ICR(6),ICR6),(ICR(7),ICR7),(IPXY,IPXY1),(IPXY(2),IPXY2)
C
      DATA IP00/44000B,64000B,24015B,50010B,26002B/
      DATA ICR/0,0,0,22054B,0,0,0/
      DATA IPXY/0,0,24001B/
      DATA WACO,LEAO,LLAO/22000B,44000B,64000B/
C
C
C
C  WRITE POSITION ON SCREEN
C
      CALL DRIVR(2,IP00,5)
C
      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
C
      CALL DRIVR(2,ICR,7)
C
C   POSITION CURSOR
C
      IPXY1 = IOR(LEAO,IAND(IX,777B))
      IPXY2 = IOR(LLAO,IAND(IY,377B))
      CALL DRIVR(2,IPXY,3)
      RETURN
      END
$
```

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

FTN4,Q,C,T
      PROGRAM IMAGE
C
C
C  THIS PROGRAM IS THE IMAGE FILE MANAGER FOR THE IMAGE DISPLAY
C  SUBSYSTEM.
C
C
      DIMENSION LU(5),IDCB1(272),IDCB2(528),IDCB3(144),JNAME(3),
     1 IFNAM(3),NAME(6),IDATA(512),KNAME(6)
C
      INTEGER ENTRY(256),ISIZE(2),TEXT1(19)
C
      EQUIVALENCE (ENTRY(7),NLINE),(ENTRY(8),NPIXL),(ENTRY(12),LOC
     1 (ENTRY(13),JNAME),(ENTRY(16),IFNAM),(ENTRY(19),IFNUM)
     2,(ENTRY,KNAME)
     2,(TEXT1,ENTRY(129))
      EQUIVALENCE (ISIZE(2),ISIZ2)
C
C
C  GET INPUT PARAMETERS
C
      CALL RMPAR(LU)
      IF (LU .LE. 0) LU = 1
C
C  OUTPUT HEADING
C
900   WRITE(LU,1)
1     FORMAT(//"      I M A G E   F I L E   M A N A G E R"//)
C
C  GET COMMAND INPUT
C
1000  WRITE(LU,2)
2     FORMAT(">_")
      READ(LU,3) ICMD
3     FORMAT(A2)
C
C  EXECUTE COMMAND
C
      IF (ICMD .NE. 2H??) GO TO 1010
C
C  COMMAND IS HELP
C
      WRITE(LU,4)
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
```

```
C
1010  IF (ICMD .NE. 2HBU) GO TO 1030
C
C  BUILD IMAGE COMMAND
C
      CALL EXEC(23+100000B,6HBLDIM ,LU)
      GO TO 1020
5     GO TO 900
C
C  PROGRAM NOT RP'ED
C
1020  WRITE(LU,6)
6     FORMAT(" BLDIM NOT RP'ED!")
      GO TO 1000
C
1030  IF (ICMD .NE. 2HDI) GO TO 1045
C
C  DISPLAY IMAGE COMMAND
C
      CALL EXEC(23+100000B,6HDSPLY ,LU)
      GO TO 1040
7     GO TO 900
C
C  DSPLY NO RP'ED
C
1040  WRITE(LU,8)
8     FORMAT("  DSPLY NOT RP'ED!")
      GO TO 1000
C
1045  IF (ICMD .NE. 2HWT) GO TO 1050
C
C  WRITE NASA TAPE
C
      CALL EXEC(23,6HWTAPE ,LU)
      GO TO 1000
C
C
1050  IF (ICMD .NE. 2HSA .AND. ICMD .NE. 2HRE .AND.
     1    ICMD .NE. 2HPU)   GO TO 1200
C
C  SAVE/RESTORE IMAGE TO/FROM TAPE AND PURGE IMAGE
C
C  OPEN DIRECTORY FILE
C
      CALL OPEN(IDCB1,IERR,6HIMDIRC,2,2HIM,23,272)
      IF (IERR .LT. 0) GO TO 9999
C
C  GET IMAGE NAME
C
      WRITE(LU,9)
9     FORMAT(" ENTER IMAGE NAME (12 CHARACTERS)?_")
      READ(LU,10) NAME
10    FORMAT(6A2)
C
C  FIND IMAGE
```

```
C
1060  CALL READF(IDCB1,IERR,ENTRY,256,LEN)
      IF (LEN .NE. -1) GO TO 1070
C
C  EOF REACHED
C
      WRITE(LU,11)
11    FORMAT(" IMAGE NOT FOUND!")
      CALL CLOSE(IDCB1)
      GO TO 1000
C
1070  IF (IERR .LT. 0) GO TO 9999
C
C  COMPARE NAME OF IMAGE
C
      IF (ICMPW(ENTRY,NAME,6) .NE. 0) GO TO 1060
C
C  IMAGE FOUND
C
      IF (ICMD .EQ. 2HRE) GO TO 1120
      IF (ICMD .EQ. 2HPU) GO TO 1300
C
C  TASK IS TO SAVE IMAGE
C
      IF (LOC .EQ. 1) GO TO 1090
C
C  IMAGE ALREADY ON TAPE
C
      WRITE(LU,12)
12    FORMAT(" IMAGE NOT ON DISC!")
      GO TO 1000
C
C  IMAGE  ON DISC
C
1090  CALL OPEN(IDCB2,IERR,JNAME,0,0,0,528)
      IF (IERR .LT. 0) GO TO 9999
C
C  GET TYPE 0 FILE
C
C     WRITE(LU,13)
C13   FORMAT(" TYPE MT LU 000# ?_")
C     READ(LU,14) IFNAM
C14   FORMAT(3A2)
      IFNAM=2HLU
      IFNAM(2) =2H00
      IFNAM(3) =2H08
      WRITE(LU,131)
131   FORMAT(" SELECT OPTION"/"  1.  8-BIT PACKED"/"  2.  UNPACKED
      READ(LU,*) IPACK
      CALL OPEN(IDCB3,IERR,IFNAM)
      IF (IERR .LT. 0) GO TO 9999
      CALL RWNDF(IDCB3,IERR)
      IF (IERR .LT. 0) GO TO 9999
      WRITE(LU,15)
15    FORMAT(" FILE #?_")
      READ(LU,*) IFNUM
      CALL SPACE(IDCB3,IERR,IFNUM-1)
      IF (IERR .LT. 0) GO TO 9999
```

```
C
C  WRITE HEADER ON TAPE
C
      CALL WRITF(IDCB3,IERR,ENTRY,11)
      IF (IERR .LT. 0) GO TO 9999
C
C  NOW TRANSFER DATA
C
      DO 1100 I = 1,NLINE
      CALL READF(IDCB2,IERR,IDATA,512)
      IF (IERR .LT. 0) GO TO 9999
C
      IF(IPACK .NE. 1) GO TO 1101
C
C  PACK DATA
C
      DO 1102 J =1,NPIXL,2
      K = 0.5*(J+1)
      IVAR=IDATA(J+1)
      CALL ROT8(IVAR,KVAR)
1102  IDATA(K)=IOR(IDATA(J),KVAR)
1101  CALL WRITF(IDCB3,IERR,IDATA,NPIXL)
      IF (IERR .LT. 0) GO TO 9999
1100  CONTINUE
C
C  WRITE EOF
C
      CALL WRITF(IDCB3,IERR,0,-1)
      IF (IERR .LT. 0) GO TO 9999
C
C  PURGE DISC FILE
C
      CALL PURGE(IDCB2,IERR,JNAME,2HIM)
      IF (IERR .LT. 0) GO TO 9999
C
C  UPDATE ENTRY
C
      LOC = 2
C
      CALL POSNT(IDCB1,IERR,-1)
      IF (IERR .LT. 0) GO TO 9999
      CALL WRITF(IDCB1,IERR,ENTRY,256)
      IF (IERR .LT. 0) GO TO 9999
C
      CALL CLOSE(IDCB1)
      CALL RWNDF(IDCB3)
      CALL CLOSE(IDCB3)
      GO TO 1000
C
C  RESTORE IMAGE FROM TAPE
C
1120  IF (LOC .EQ. 2) GO TO 1130
C
C  IMAGE ON DISC
C
      WRITE(LU,16)
16    FORMAT(" IMAGE ALREADY ON DISC!")
      CALL CLOSE(IDCB1)
      GO TO 1000
```

```
C
C  CREATE DISC FILE
C
1130  ISIZE = (FLOAT(NLINE)*FLOAT(NPIXL)+127.)/128.
      ISIZ2 = NPIXL
C
      CALL CREAT(IDCB2,IERR,JNAME,ISIZE,2,2HIM,23,528)
      IF (IERR .GE. 0) GO TO 1135
C
C  CAN'T CREATE DISC FILE
C
      WRITE(LU,19)
19    FORMAT(" CAN'T CREATE DISC FILE!!")
      CALL CLOSE(IDCB1)
      GO TO 1000
C
C OPEN TYPE 0 FILE
C
1135  CALL OPEN(IDCB3,IERR,IFNAM)
C
C GET LU OF TYPE 0 FILE
C
      CALL LOCF(IDCB3,IERR,IREC,IRB,IOFF,JSEC,MTLU)
      IF (IERR .LT. 0) GO TO 9999
C
C  TELL USER TO MOUNT TAPE
C
      WRITE(LU,17) MTLU
17    FORMAT(" MOUNT TAPE ON LU ",I2"  ENTER RETURN WHEN READY")
      CALL EXEC(1,LU,IREC,1)
C
C  REWIND TAPE
C
      CALL RWNDF(IDCB3,IERR)
      IF (IERR .LT. 0) GO TO 9999
C
C  SPACE FORWARD TO FILE
C
      CALL SPACE(IDCB3,IERR,IFNUM-1)
C
C  READ HEADER
C
      CALL READF(IDCB3,IERR,IDATA,11)
      IF (IERR .LT. 0) GO TO 9999
      IF(ICMPW(IDATA,ENTRY,11) .NE. 0) GO TO 1160
C
C  HEADER COMPARES
C
C  TRANSFER DATA
C
      DO 1140 I=1,NLINE
      CALL READF(IDCB3,IERR,IDATA,NPIXL)
      IF (IERR .LT. 0) GO TO 9999
      CALL WRITF(IDCB2,IERR,IDATA,NPIXL)
      IF (IERR .LT. 0) GO TO 9999
1140  CONTINUE
C
```

```
      CALL RWNDF(IDCB3)
      CALL CLOSE(IDCB3)
      CALL CLOSE(IDCB2)
C
C  UPDATE DIRECTORY ENTRY
C
      LOC = 1
C
      CALL POSNT(IDCB1,IERR,-1)
      IF (IERR .LT. 0) GO TO 9999
      CALL WRITF(IDCB1,IERR,ENTRY,256)
      IF (IERR .LT. 0) GO TO 9999
      CALL CLOSE(IDCB1,IERR)
      IF (IERR .LT. 0) GO TO 9999
C
      GO TO 1000
C
C  LABEL DOES NO MATCH
C
1160  WRITE(LU,18)
18    FORMAT(" WRONG FILE!!")
      CALL RWNDF(IDCB3)
      CALL CLOSE(IDCB3)
      CALL CLOSE(IDCB1)
      GO TO 1000
C
C
1200  IF (ICMD .NE. 2HDL) GO TO 1230
C
C  DIRECTORY LIST
C
C  OPEN DIRECTORY FILE
C
      CALL OPEN(IDCB1,IERR,6HIMDIRC)
      IF (IERR .LT. 0) GO TO 9999
C
C  OUTPUT HEADING
C
      WRITE(LU,30)
30    FORMAT(//"IMAGE NAME    #LINES  #PIXELS  LOC       TEXT"/)
C
C  OUTPUT INFO
C
1210  CALL READF(IDCB1,IERR,ENTRY,256,LEN)
      IF (LEN .NE. -1) GO TO 1220
C
C EOF REACHED
C
      CALL CLOSE(IDCB1)
      GO TO 1000
C
1220  IF (IERR .LT. 0) GO TO 9999
C
      IF (ENTRY .EQ. -1) GO TO 1210
      ICHR = 2HD
      IF (LOC .NE. 1) ICHR = 2HT
      WRITE(LU,31)KNAME,NLINE,NPIXL,ICHR,TEXT1
31    FORMAT(6A2,2X,I4,4X,I4,3X,A5,2X,19A2)
      GO TO 1210
```

```
C
1230  IF (ICMD .EQ. 2HEX) GO TO 1240
C
C  ILLEGAL COMMAND
C
      WRITE(LU,22)
22    FORMAT("ILLEGAL COMMAND!")
      GO TO 1000
C
1240  WRITE(LU,23)
23    FORMAT("END PROGRAM")
      CALL EXEC(6)
C
C  PURGE FILE
C
1300  CALL POSNT(IDCB1,IERR,-1)
      IF (IERR .LT. 0) GO TO 9999
      ENTRY = -1
      CALL WRITF(IDCB1,IERR,ENTRY,256)
      IF (IERR .LT. 0) GO TO 9999
C
C  PURGE DATA FILE
C
      CALL PURGE(IDCB2,IERR,JNAME,2HIM)
      CALL CLOSE(IDCB1)
      GO TO 1000
C
C
C   ERROR
C
C
9999  WRITE(LU,20) IERR
20    FORMAT(" FILE ERROR ",I6)
      CALL CLOSE(IDCB1)
      GO TO 1000
      END
$
```

&SPACE T=00004 IS ON CR00022 USING 00004 BLKS R=0029

```
FTN4
      SUBROUTINE SPACE(IDCB,IERR,NUM)
C
C   THIS SUBROUTINE IS USED TO SPACE FORWARD OR BACKWARD THE
C   NUMBER OF FILES SPECIFIED.
C
      DIMENSION IDCB(144)
C
      DATA IFRWD,IBACK/1300B,1400B/
C
C
C
      IERR = 0
      IF (NUM .EQ. 0) RETURN
C
      IDIR = IFRWD
      IF (NUM .GT. 0) GO TO 100
      IDIR = IBACK
      NUM = -NUM
C
C
100   DO 110 I=1,NUM
      CALL FCONT(IDCB,IERR,IDIR)
      IF (IERR .LT. 0) RETURN
110   CONTINUE
C
      RETURN
      END
$
```

&RESIZ T=00004 IS ON CR00022 USING 00060 BLKS R=0541

```
FTN4,Q,T,C
      PROGRAM RESIZ
C     WRITTEN BY W. E. ALEXANDER
C
C     PROGRAM FORMS A PART OF THE SPATIAL DOMAIN FILTERING PACKAGE
C
C     PROGRAM ALLOWS THE USER TO INTERPOLATE AND SCALE AN IMAGE AN
C     CHANGE ITS DATA TYPE. THUS A FLOATING POINT IMAGE CAN BE MAD
C     INTO AN EIGHT BIT IMAGE.
C
C
C
      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,RITEL
      EQUIVALENCE(G(1),IOP(1))
      EQUIVALENCE(F(1),G(1))
      EQUIVALENCE (DIRC(4),F(1)),(DIRC(1),INM(1)),(INM(1),NROW)
      EQUIVALENCE (INM(2),ICOLS),(DIRC(2),AMAX),(DIRC(3),AMIN)
C
      DATA NSON/2HLF,2HLT,2HR ,2HDI,2HNT,2HP /
      CALL RMPAR(IPRAM)
      LU=IPRAM(1)
      IF(LU.EQ.0) LU=1
C
C     INITIALIZE PARAMETERS
C
      ITYPE = 8
C
      CALL CODE
      WRITE (JMES,6)
6     FORMAT (" RESIZE")
      CALL TRMGN (JMES,LU,0)
      CALL BLANK (JMES)
      NTYPE=32
      IRTCD=0
      IMXX=512
      IMXP1=IMXX+1
      IFE=0
      ILE=511
      IFR=0
      ILR=511
C
5     CALL CODE
      WRITE (JMES,995)
995    FORMAT (" RESIZE IMAGE ")
      CALL TRMGN (JMES,LU,0)
      CALL BLANK (JMES)
C
C     SPECIFY DATA LENGTH FOR OUTPUT
C
```

```
   10 CALL CODE
      WRITE(JMES,11)
   11 FORMAT(" SPECIFY OUTPUT DATA TYPE")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,12)
   12 FORMAT("  1.  8 BIT IMAGE")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,13)
   13 FORMAT("  2.  15 BIT IMAGE")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
C
      IRTM=ICD
C
      CALL BLANK(JMES)
   15 CALL SPCHR (IRTM,IRT)
      GO TO (500,10,5,20,17,17),IRT
17    CALL CKFLD(2,ICD,IRS)
      GO TO (25,25,30,30,20),IRS
   20 IW=1
      GO TO 950
   25 ITYPE =8
      IMAX=255
      GO TO 32
   30 ITYPE =15
      IMAX=32767
C
C     SPECIFY WORK FILE
C
C
32    CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,450)
450   FORMAT("  SELECT OPTION")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,455)
455   FORMAT("  1.  SPECIFY NEW IMAGE")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,460)
460   FORMAT("  2.  USE CURRENT WORK FILE")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
465   CALL CKFLD(2,ICD,IRS)
      GO TO (475,475,480,485),IRS
485   IW=12
C
C     OPEN WORK FILE
C
```

```
      GO TO 950
  475 IGET=WFINT (NROW,ICOLS,AMAX,AMIN,LU)
      IF(IGET.LT.0) GO TO 999
  480 IGET=READL(-1,0,511,DIRC)
      IF(IGET.LT.0) GO TO 999
   40 CALL CODE
      WRITE(JMES,45) AMAX,AMIN
   45 FORMAT(" AMAX= ",E12.5,5X," AMIN= ",E12.5,5X,"FOR IMAGE")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
C
C     SPECIFY IMAGE SCALING OPTION
C
   50 CALL CODE
      WRITE(JMES,51)
   51 FORMAT("     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,ICD,IRTM)
      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
C
C     AUTOMATIC SCALING SELECTED
C
   65 SCL=AMAX-AMIN
      IF (ABS(SCL).LE.1.0E-5) GO TO 70
      SCL=FLOAT(IMAX)/SCL
       IOPT=1
       GO TO 190
C
C      SYSTEM DEFAULT SCALING OPTION SELECTED
C
 75    SCL=1.0
       IOPT=2
       GO TO 190
```

```
      CALL CODE
      WRITE(JMES,151)
  151 FORMAT(" SCALE TOO SMALL")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,152)
  152 FORMAT("  ENTER CR TO GO TO SYSTEM LEVEL MENU")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
       IRTCD=1HX
       GO TO 1000
 155   SCL=(1.0/SCL)*FLOAT(IMAX)
       GO TO 190
C
C      LOG COMPRESSION OPTION SELECTED
C
  160 CALL CODE
      WRITE(JMES,161)
  161 FORMAT(" LOG COMPRESSION OPTION SELECTED")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
       IOPT=5
       SCL=FLOAT(IMAX)/ALOG(AMAX-AMIN+1.0)
       GO TO 190
C
C      EXPONENTIATION OPTION SELECTED
C
  165 CALL CODE
      WRITE(JMES,166)
  166 FORMAT("  ENTER DESIRED EXPONENT")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
 170   CALL SPCHR (IRTM,IRT)
      POWER=RTM
      IF (IRT .EQ. 5) GO TO 180
  175 IW=6
      GO TO 950
 180   POWER=ABS(POWER)
      CALL CODE
      WRITE(JMES,185) POWER
 185   FORMAT("   EXPONENT= ",1F10.4)
       CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
       SCL=FLOAT(IMAX)/((AMAX-AMIN)**POWER)
       IOPT=6
C
C      OBTAIN PARAMETERS FOR RESIZING IMAGE
C
 190   INUM=64
       INCNT=0
       IMCNT=0
       NTST=INUM
  195 CALL CODE
      WRITE(JMES,196)
  196 FORMAT("  INDEPENDENT DIRECTIONAL SCALING")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
```

```
C
C       USER ENTERS SCALE FACTOR
C
   80   CALL CODE
        WRITE(JMES,81)
    81 FORMAT(" ENTER DESIRED SCALE FACTOR")
        CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
       CALL BLANK(JMES)
   85   CALL SPCHR (IRTM,IRT)
       GO TO (1000,1000,1000,1000,855),IRT
  855   GAIN=ABS(RTM)
       IF (IRT .EQ. 5) GO TO 95
   90   IW=3
       GO TO 950
    95 CALL CODE
        WRITE(JMES,100) GAIN
  100   FORMAT ("    SCALE FACTOR = ",F10.4)
        CALL TRMGN(JMES,LU,0)
       CALL BLANK(JMES)
        IOPT=3
        SCL=GAIN
        GO TO 190
C
C       USER SPECIFIED MAXIMUM AND MINIMUM
C
  105   IOPT=4
        CALL CODE
        WRITE(JMES,106)
   106 FORMAT("  ENTER MAXIMUM FOR IMAGE")
        CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
       CALL BLANK(JMES)
  110   CALL SPCHR (IRTM,IRT)
       GO TO (1000,1000,1000,1000,910),IRT
  910  AMXIN=RTM
        IF(RTM.NE.OB) GO TO 120
  115   IW=4
       GO TO 950
   120 CALL CODE
       WRITE(JMES,121) AMXIN
   121 FORMAT("    MAXIMUM FOR IMAGE = ",1PE15.8)
       CALL TRMGN(JMES,LU,0)
       CALL BLANK(JMES)
  130   CALL CODE
        WRITE(JMES,131)
   131 FORMAT("  ENTER MINIMUM FOR IMAGE")
       CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
       CALL BLANK(JMES)
  135   CALL SPCHR (IRTM,IRT)
       GO TO (1000,1000,1000,1000,935),IRT
  935  AMNIN=RTM
       IF (IRT .EQ. 5) GO TO 145
   140 IW=5
       GO TO 950
   145 CALL CODE
       WRITE(JMES,150) AMNIN
  150   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
```

```
931   CALL CODE
      WRITE(JMES,197)
  197 FORMAT(" ENTER ROW SCALE FACTOR")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
 200   CALL SPCHR (IRTM,IRT)
      YS=RTM
      IF (IRT .EQ. 5) GO TO 210
  205 IW=7
      GO TO 950
  210 CALL CODE
      WRITE(JMES,211)
  211 FORMAT("  ENTER COLUMN SCALE FACTOR ")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
 215   CALL SPCHR (IRTM,IRT)
      XS=RTM
      IF (IRT .EQ. 5) GO TO 225
  220 IW=8
      GO TO 950
C
C     CHECK TO SEE IF INTERPOLATION IS REQUIRED.  IF NOT BRANCH.
C
C
C     COMPUTE NEW SIZE OF IMAGE
C
  225  NNEW=YS*NROW+.5
       NCOLS=XS*ICOLS+.5
C
      IF(XS.EQ.1.0.AND.YS.EQ.1.0) GO TO 260
       IF(NCOLS.LE.512) GO TO 230
      CALL CODE
      WRITE(JMES,690) NCOLS,XS
 690  FORMAT("   CALCULATED COLUMN SIZE = ",115,"(SF =",1F5.2")")
       CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE (JMES,969)
969   FORMAT ("      512 IS MAXIMUM ALLOWABLE OUTPUT")
      CALL TRMGN (JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,226)
  226 FORMAT(" REENTER COLUMN SCALE FACTOR")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
       GO TO 215
  230 CALL CODE
      WRITE(JMES,695) NNEW,YS,NCOLS,XS
 695   FORMAT("-- OUTPUT IMAGE-",14X,"ROWS = ",115,"(SF=",1F5.2,")
     *14X,"COLUMNS = ",115," (SF = ",1F5.2,")")
 231   CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,1232)
 1232 FORMAT("  1.  VALUES OKAY")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
```

```
      CALL CODE
      WRITE(JMES,1233)
 1233 FORMAT("  2.  REENTER SCALE FACTOR")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
  232  CALL SPCHR (IRTM,IRT)
       CALL CKFLD(2,ICD,IRS)
      GO TO (234,234,931),IRS
  233 IW=9
      GO TO 950
C
C     COMPUTE INCREMENTS FOR INTERPOLATION
C
  234  MM=NNEW
       IFLT=0
       DY=FLOAT(NROW)/FLOAT(NNEW)
       DX=FLOAT(ICOLS)/FLOAT(NCOLS)
       IF(DY.LE.1.0) GO TO 235
  700 CALL CODE
      WRITE(JMES,1750)
 1750 FORMAT(" IMAGE SHOULD BE FILTERED BEFORE INTERPOLATION")
       CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE (JMES,9750)
 9750 FORMAT ("    TO PREVENT ALIASING")
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
       CALL CODE
       WRITE(JMES,760)
 760   FORMAT("  1.  CONTINUE ")
       CALL TRMGN(JMES,LU,0)
       CALL BLANK(JMES)
       WRITE(JMES,761)
  761 FORMAT("  2.  PREFILTER IMAGE")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
  704  CALL SPCHR (IRTM,IRT)
      GO TO (1000,1000,1000,1000,904),IRT
 904   CALL CKFLD(2,ICD,IRS)
      GO TO (702,702,710),IRS
  710 IW=10
      GO TO 950
C
C     SCHELULE FILTER TO PREVENT ALIASING
C
  702  FCX=0.8*FLOAT(NCOLS)/FLOAT(ICOLS)
       FCY=0.8*FLOAT(NNEW)/FLOAT(NROW)
       NX=2
       NY=2
       ITME=0
       CALL XYFLT(U,V,FCX,FCY,NX,NY,N,A,B)
C
      DO 705 II=1,3
  705 NAME(II)=NSON(II,1)
      CALL EXEC (9,NAME,LU,0,NCOLS-1)
C
C     INITIALIZE FOR RESIZING
```

```
      C
        235 CALL CODE
            WRITE(JMES,810)
        810 FORMAT("  RESIZING OF IMAGE IN PROGRESS")
            CALL TRMGN(JMES,LU,0)
            CALL BLANK(JMES)
      C
      C      SCHEDULE DINTP FOR RESIZING IMAGE
      C
            INCW=LU+200B
            DO 250 II=1,3
        250 NAME(II)=NSON(II,2)
      C
      C     CLOSE WORK FILE
      C
            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))
      C
      C     OPEN WORK FILE
      C
      C
      C
      C      CALL OPEN(IDCB,IGET,6HWF0000,2,0,0,528)
      C      IF(IGET.LT.0) GO TO 999
      C
      C      REMAP INTENSITY VALUES FOR IMAGE
      C
      C      OBTAIN NEW SIZE PARAMETERS
      C
      C
             IGET=READL(-1,0,511,DIRC)
             IF(IGET.LT.0) GO TO 999
      C
      260   IMCNT=0
            IZCNT=0
            DO 405 NN=1,NROW
      C
      C      READ IN NEW ROW
      C
            NNM1=NN-1
             IGET=READL(NNM1,IFE,ICOLS-1,F)
             IF(IGET.LT.0) GO TO 999
             IF (IOPT .GT. 6) GO TO 310
            GO TO (306,310,320,330,340,350),IOPT
      306    DO 305 I=1,ICOLS
       305   G(I)=(F(I)-AMIN)*SCL+0.5
             GO TO 360
       310   DO 315 I=1,ICOLS
       315   G(I)=F(I)+0.5
             GO TO 360
       320   DO 325 I=1,ICOLS
       325   G(I)=(F(I)*SCL+0.5)
             GO TO 360
       330   DO 335 I=1,ICOLS
       335   G(I)=(F(I)-AMNIN)*SCL+0.5
             GO TO 360
       340   DO 345 I=1,ICOLS
             F(I)=AMAX1(F(I),AMIN)
       345   G(I)=SCL*(ALOG(F(I)-AMIN+1.0))+0.5
             GO TO 360
       350   DO 355 I=1,ICOLS
       355   G(I)=SCL*(F(I)-AMIN)**POWER+0.5
```

```
C
C      WRITE OUTPUT TO WORK FILE
C
  360 IGET=RITEL(NNM1,0,ICOLS-1,G)
      IF(IGET.LT.0) GO TO 999
C
C     IF OUTPUT IS 8 BIT, WRITE TO DISPLAY
C
      IF(ITYPE.EQ.15) GO TO 365
C
       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)=MIN0(IFIX(G(I)),IMAX)
 370   IOP(I)=MAX0(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)
C
  400 IF(ITYPE.NE.8) GO TO 405
      IGET=WLINE(NNM1,00,ICOLS-1,IOP)
      IF(IGET.LT.0) GO TO 999
  405 CONTINUE
C
C      CLOSE WORK FILE
C
       AMAX=FLOAT(IMAX)
       AMIN=0.0
        CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
       IRTCD=0
       ITOT=NROW*ICOLS
       ATOT=100.0/FLOAT(ITOT)
       PZERO=ATOT*FLOAT(IZCNT)
       PMAX=ATOT*FLOAT(IMCNT)
       IF(PZERO.EQ.0.0 .AND.PMAX.EQ.0.0) GO TO 1000
      CALL CODE
      WRITE(JMES,410) PZERO,PMAX
      CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
 410   FORMAT("- PERCENT CLIPPED AT ZERO =",F6.2,
     1"     - PERCENT CLIPPED AT MAX =",F6.2)
 420   CALL TRMGN(JMES,LU,0)
      CALL BLANK(JMES)
      CALL CODE
      WRITE(JMES,421)
  421 FORMAT(" 1.  CONTINUE[  2.  RESCALE IMAGE")
      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
      CALL BLANK(JMES)
 425   CALL SPCHR (IRTM,IRT)
      GO TO (1000,1000,1000,1000,925),IRT
 925   CALL CKFLD(2,ICD,IRS)
      GO TO (1000,1000,440,430),IRS
  430 IW=11
      GO TO 950
```

```
440    CALL READL(-1,0,511,DIRC)
        GO TO 5
 70    CALL CODE
       WRITE(JMES,921)
921    FORMAT (' SCALING SIZE ERROR')
       CALL TRMGN(JMES,LU,0)
       CALL BLANK(JMES)
500    CONTINUE
       GO TO 999
950    CALL CODE
       WRITE (JMES,21)
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
999    IF(IGET.EQ.-8) CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
1000   CALL EXEC(6)
        END
$
$
```

&ROT8 T=00004 IS ON CR00022 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
```

&TRMGN T=00004 IS ON CR00022 USING 00056 BLKS R=0439

```
FTN4,L
      SUBROUTINE TRMGN(JMES,LU,IP,RTM,ICD,IRTM)
C
C  THIS SUBROUTINE IS USED TO WRITE OUT AND POSSIBLY READ BACK
C  FROM THE TERMINAL INFORMATION NECESSARY FOR PROGRAM CONTROL.
C  JMES IS TRHE MESSAGE TO BE OUTPUT TO THE LU. IP (IF =0) MEANS
C  WRIE ONLY, (IF =1) MEANS TO WAIT FOR A RESPONSE FROM THE OPERAT
C  RTM IC 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 MAXIMUN
C  OUTPUT MESSAGE IS 80 CHARACTERS LONG. THE MAXIMUN INPUT MESSAGE
C  10 CHARACTERS LONG.
C
      DIMENSION JMES(40),IRTM(5)
      ICNWD=400B+LU
C     WRITE THE MESSAGE TO THE LU
      CALL EXEC (2,ICNWD,JMES,40)
      IF (IP .EQ. 0) RETURN
C     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
C       WRITTEN BY W.E.ALEXANDER
C       SUBROUTINE FORM 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       FREQUENCY IN THE X DIRECTION AND S IS THE SAMPLING
C       INTERVAL (X DIRECTION)
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
        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
7     U(I)=0
6     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
C
              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
C
              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
C             CALCULATE COEFFICIENTS
C
              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
   22         IF(J.EQ.1)THT=112.5*PI/180.0
              IF(J.EQ.2)THT=157.5*PI/180.0
              GO TO 23
   20         CN=CNY
              DD = DY
              IF (NY.EQ.3) GO TO 21
              THT=135.0*PI/180.0
              GO TO 23
   21         IF(J.EQ.1)THT=112.5*PI/180.0
              IF(J.EQ.2)THT=157.5*PI/180.0
   23         ALP=COS(THT)
              BET = SIN(THT)
              S1 = 1.0+ALP*DD
              S2 = 1.0-ALP*DD
              S3 = BET*DD
              S4=-S3
              P=CMPLX(S1,S3)/CMPLX(S2,S4)
              S1 = -2*REAL(P)
              S2 = (CABS(P))**2
C
```

```
             AA = 0.25 * ( 1.0 + S1 +S2 )
             A(1,J,K) = AA
             A(2,J,K) = 2.0*AA
             A(3,J,K) = AA
             B(1,J,K) = 1.0
             B(2,J,K) = S1
      10     B(3,J,K)=S2
      C
      C      OBTAIN TWO DIMENSION FILTER
      C
             IF(NX.EQ.3)GO TO 30
             A(1,2,1) = 1.0
             B(1,2,1) = 1.0
      30     IF(NY.EQ.3)GO TO 31
             A(1,2,2) = 1.0
             B(1,2,2) = 1.0
      C
      31     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)
      40     V(I,J,K) = B(I,K,1)*B(J,K,2)
             RETURN
             END
      C  *****************SUBROUTINE INTRP************************
      C/L60
      C  *****************SUBROUTINE INTRP************************
      C  *                                                       *
      C  * THIS SUBROUTINE IS FOR INTERPOLATING POINTS IN AN ARRAY *
      C  *                                                       *
      C  *************SUBROUTINE VARIABLES************************
      C  *  AINT:STORAGE ARRAY FOR DATA POINTS                    *
      C  *  Y:THE DISTANCE BETWEEN LINES OF DATA POINTS           *
      C  *  DX:INTERVAL VALUE BETWEEN INTERPOLATING POINTS        *
      C  *  NCOLS:NUMBER OF POINTS REQUIRED PER ROW IN OUTPUT     *
      C  *  ICOLS:NUMBER OF POINTS TO PER ROW IN INPUT            *
      C  *  FOP:THE OUTPUT ARRAY                                  *
      C  *********************************************************
      C
      C
             SUBROUTINE INTRP(AINT,Y,DX,NCOLS,ICOLS,FOP)
             DIMENSION AINT(1), FOP(1),JMES(40)
           CALL CODE
           WRITE (JMES,150)
       150  FORMAT(' NOW IN INTRP')
           CALL TRMGN(JMES,LU,0)
      C
             IMAX=512
             IMXP=IMAX+1
             ICM1=ICOLS-1
             I=1
             M=I
       15    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
       25    E=(AINT(M+1)-AINT(M))*X+AINT(M)
             F=(AINT(M+IMXP)-AINT(M+IMAX))*X+AINT(M+IMAX)
             FOP(I)=(F-E)*Y+E
```

```
           I=I+1
           IF(I.LE.NCOLS) GO TO 15
           IF(X.LT.1.0.AND.I.GT.NCOLS) GO TO 51
C
C             COMPLETE INTERPOLATION
C
 50       FOP(NCOLS)=(AINT(ICOLS+IMAX)-AINT(ICOLS))*Y+AINT(ICOLS)
    51 CALL CODE
          WRITE(JMES,160)
 160      FORMAT(' NOW LEAVING INTRP')
         CALL TRMGN(JMES,LU,0)
           RETURN
           END
```

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

```
FTN4,L
         SUBROUTINE SPCHR (IRTCD,IRT)
C
C        THIS ROUTINE CHECKS FOR SPECIAL CHARACTERS IN THE INPUT DATA
C
         IRT=5
         IF (IRTCD.EQ. OB) IRT=0
         IF ((IRTCD.EQ.1HX) .OR. (IRTCD .EQ. 2HX ))IRT=1
         IF ((IRTCD.EQ.1HB) .OR. (IRTCD .EQ. 2HB ))IRT=2
         IF ((IRTCD.EQ.1HD) .OR. (IRTCD .EQ. 2HD ))IRT=3
         IF ((IRTCD.EQ.1HR) .OR. (IRTCD .EQ. 2HR ))IRT=4
         RETURN
         END
         SUBROUTINE CKFLD(IA,ICD,IRS)
C
C        SUBROUTINE TO CHECK FOR CARRIAGE RETURN OR NUMERIC VALUE
C
         IRS=ICD+1
         IF (ICD .EQ. OB) IRS=1
         RETURN
         END
         SUBROUTINE INFRM (IA,LU)
         DIMENSION IA(3)
         ICNWD=400B +LU
         CALL EXEC (2,ICNWD,IA,3)
         RETURN
         END
C
C
         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)
C        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)
C
```

```
C     INITIALIZE
C
      IMAX=512
      INT=ICOLS/2-1
      IMXP1=IMAX+1
      ASTT=AINT(INT)+AINT(INT+1)+AINT(INT+2)
      ASTT=ASTT/3
      DO 10 I=1,3
      WF(I)=ASTT
10    WG(I)=ASTT
C
C     START FORWARD FILTER
C
      MM=IMXP1
      WF(1)=AINT(IMXP1)
20    WG(1)=A1*WF(1)+A2*WF(3)+A3*WF(3)-B2*WG(2)-B3*WG(3)
C
C     UPDATE
C
      AINT(MM)=WG(1)
      WG(3)=WG(2)
      WG(2)=WG(1)
      WF(3)=WF(2)
      WF(2)=WF(1)
      MM=MM+1
      IF (MM.GT. ICOLS) GO TO 30
      WF(1)=AINT(MM)
      GO TO 20
C
C     START REVERSE FILTER
C
30    ASTT=AINT(INT)
      DO 40 I=1,3
      WF(I)=ASTT
40    WG(I)=ASTT
C
      MM=IMAX+ICOLS
      WF(I)=AINT(MM)
41    WG(1)=A1*WF(1)+A2*WF(2)+A3*WF(3)-B2*WG(2)-B3*WG(3)
C
C     UPDATE
C
      AINT(MM)=WG(1)
      WG(3)=WG(2)
      WG(2)=WG(1)
      WF(3)=WF(2)
      WF(2)=WF(1)
      MM=MM-1
      IF (MM .LE. IMAX) GO TO 50
      WF(1)=AINT(MM)
      GO TO 41
50    RETURN
      END
      SUBROUTINE BLANK (JMES)
      DIMENSION JMES(40)
      DO 10 I=1,40
10    JMES(I)=2H
      RETURN
      END
      END$
```

&WFINT T=00004 IS ON CR00022 USING 00006 BLKS R=0044

```
FTN4
      INTEGER FUNCTION WFINT(NLINE,NPIXL,PMAX,PMIN,LU)
C
C
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
C     LU    = INTERACTIVE TERMINAL LU
C
C     NLINE = # LINES IN IMAGE
C     NPIXL = # PIXELS/LINE
C     PMAX  = MAXIMUM PIXEL INTENSITY IN IMAGE (REAL)
C     PMIN  = MINIMUM PIXEL INTENSITY IN IMAGE (REAL)
C
C
      DIMENSION IDCB1(144),IRTN(5),IB(6)
C
      EQUIVALENCE (IB2,IB(2)),(IB(3),RMAX),(IB(5),RMIN)
C
C
C
C  SCHEDULE BUILD WORK FILE PROGRAM
C
      CALL EXEC(23,6HBLDWF ,LU)
C
C  GET RETURNED PARAMETERS
C
      CALL RMPAR(IRTN)
      WFINT = IRTN
      IF (IRTN .LT.0 ) RETURN
C
C  GET MAX MIN DATA
C
      CALL OPEN(IDCB1,IERR,6HWF0000)
      IF (IERR .LT. 0) GO TO 100
C
      CALL READF(IDCB1,IERR,IB,6)
      IF (IERR .LT. 0) GO TO 100
      NLINE = IB
      NPIXL = IB2
      PMAX =  RMAX
      PMIN = RMIN
      CALL CLOSE(IDCB1)
      WFINT = 0
      RETURN
C
100   WFINT = IERR
      CALL CLOSE(IDCB1)
      END
C
```

```
C
C  READ LINE FROM WORK FILE SUBROUTINE
C
C
      INTEGER FUNCTION READL(LINE,IPIXL,JPIXL,RBUF)
C
      COMMON /CBLK/IDCB(528),TBUF(512),IFLAG
C
      DIMENSION RBUF(512)
C
C
C CHECK IF FILE OPEN
C
      IF (IFLAG .EQ. 1) GO TO 100
C
C  MUST OPEN FILE
C
      CALL OPEN(IDCB,IERR,6HWF0000,2,0,0,528)
      IF (IERR .LT. 0) GO TO 999
      IFLAG = 1
C
C  FILE OPENED--READ APPROPRIATE LINE
C
100   CALL READF(IDCB,IERR,TBUF,1024,LEN,LINE+2)
      IF (IERR .LT. 0) GO TO 999
C
C POSITION DATA IN BUFFER
C
      ISTEP = 1
      IF (IPIXL .GT. JPIXL) ISTEP = -1
C
      J = 1
      DO 110 I=IPIXL+1,JPIXL+1,ISTEP
      RBUF(J) = TBUF(I)
110   J = J+1
      READL = 0
      RETURN
C
C  ERROR
C
999    READL = IERR
      END
C
C
C  WRITE WORK FILE SUBROUTINE
C
C
      INTEGER FUNCTION RITEL(LINE,IPIXL,JPIXL,RBUF)
C
      COMMON /CBLK/IDCB(528),TBUF(512),IFLAG
C
      DIMENSION RBUF(512)
C
C  CHECK IF FILE OPENED
C
      IF (IFLAG .EQ. 1) GO TO 100
C
```

```
C  MUST OPEN FILE
C
      CALL OPEN(IDCB,IERR,6HWF0000,2,0,0,528)
      IF (IERR .LT. 0) GO TO 999
      IFLAG = 1
C
C  FILE OPENED--WRITE APPROPRIATE LINE
C
100   CALL READF(IDCB,IERR,TBUF,1024,LEN,LINE+2)
      IF (IERR .LT. 0) GO TO 999
C
      ISTEP = 1
      IF (IPIXL .GT. JPIXL) ISTEP = -1
      J = 1
      DO 110 I=IPIXL+1,JPIXL+1,ISTEP
      TBUF(I) = RBUF(J)
110   J = J+1
C
      CALL WRITF(IDCB,IERR,TBUF,0,LINE+2)
      IF (IERR .LT. 0) GO TO 999
C
      RITEL = 0
      RETURN
C
C  ERROR RETURN
C
999   RITEL = IERR
      END
C
C
C  BLOCK DATA SUBROGRAM
C
C
      BLOCK DATA
C
      COMMON /CBLK/IDCB(528),TBUF(512),IFLAG
C
      DATA IFLAG/0/
C
      END
C
C
C  CLOSE WORK FILE SUBROUTINE
C
C
      SUBROUTINE CLSWF(NLINE,NPIXL,PMAX,PMIN)
C
      COMMON /CBLK/ IDCB(528)
C
      DIMENSION IB(6)
C
      EQUIVALENCE (IB2,IB(2)),(IB(3),RMAX),(IB(5),RMIN)
C
C
C  THIS ROUTINE IS USED TO CLOSE THE WORK FILE
C
C WRITE DATA ON WORK FILE
C
      IB = NLINE
      IB2 = NPIXL
      RMAX = PMAX
      RMIN = PMIN
C
      CALL WRITF(IDCB,IERR,IB,6,1)
      CALL CLOSE(IDCB)
```

&DINTP T=00004 IS ON CR00022 USING 00009 BLKS R=0087

```
FTN4,Q,T,C
      PROGRAM DINTP
C
C     THIS PROGRAM IS USED CHANGE THE PHYSICAL SIZE OF AN IMAGE
C
C     WRITTEN BY WINSER E. ALEXANDER
C
      DIMENSION AINT(1024),F(512),IPRAM(5),DIRC(515),INM(2)
      EQUIVALENCE (F(1),DIRC(4)),(DIRC(1),INM(1))
      EQUIVALENCE (INM(1),NROW),(INM(2),ICOLS),(DIRC(2),AMAX)
      EQUIVALENCE (DIRC(3),AMIN)
C
C     INPUT PARAMETERS (CALL RMPAR)
C     IPRAM(1)= LOGICAL UNIT FOR INTERACTIVE DEVICE
C     IPRAM(2) = NUMBER OF DESIRED ROWS IN OUTPUT IMAGE
C     IPRAM(3) = NUMBER OF DESIRED COLUMNS IN OUTPUT IMAGE
C
C     IMAGE TO BE USED IS ASSUMED TO BE IN IMAGE WORK FILE (WF0000
C
      CALL RMPAR(IPRAM)
      LU=IPRAM(1)
      IF(LU.LE.0) LU=1
      NNEW=IPRAM(2)
      NCOLS=IPRAM(3)
      NCM1=NCOLS-1
      IMX=512
      IMXP1=IMX+1
C
C     OBTAIN PARAMETERS FROM CURRENT IMAGE
C
      IGET=READL(-1,0,511,DIRC)
      ICM1=ICOLS-1
      IF(IGET.LT.0) GO TO 999
C
C     INTERPOLATE IMAGE
C
      DY=FLOAT(NROW)/FLOAT(NNEW)
      DX=FLOAT(ICOLS)/FLOAT(NCOLS)
      IFLT=0
      IF(DY.GT.1.0) STOP 111
      IF(DX.LT.0.0) IFLT=1
      IFR=0
      IFE=0
C
```

```
C     INITIALIZE ARRAYS
C
      IGET=READL(0,IFE,ICOLS-1,AINT(MXP1))
      IF(IGET.LT.0) GO TO 999
      IGET=READL(1,IFE,ICM1,AINT(1))
      IF(IGET.LT.0) GO TO 999
C
      MCNT=2
      MORG=NROW
      DO 100 KK=NNEW,1,-1
C
C     COMPUTE Y
C
   20 Y=(NNEW-KK)*DY-(NROW-MORG)
      IF(Y.LT.1.0) GO TO 50
C
C     BRING IN NEW ROW
C
      CALL MOVE(AINT,ICOLS,IMXP1)
C
      MCNT=MCNT+1
      IGET=0
      IF(MCNT.GT.NROW) IGET=-150
      IF(IGET.LT.0) GO TO 999
      IGET=READL(MCNT,IFE,ICM1,AINT)
      IF(IGET.LT.0) GO TO 999
      MORG=MORG-1
C
C     RECOMPUTE Y
C
      GO TO 20
C
C     INTERPOLATE FOR NEW ROW
C
   50 CALL INTRP(AINT,Y,DX,NCOLS,ICOLS,F)
C
C     OUTPUT CURRENT ROW
C
  100 CALL RITEL(KK-1,0,NCM1,F)
C
C    NOTE THAT WORK FILE IS NOT CLOSED BY THIS PROGRAM
C
C     INSERT PARAMETERS IN WORK FILE
C
      NROW=NNEW
      ICOLS=NCOLS
      CALL RITEL(-1,0,ICM1,DIRC)
  999 CONTINUE
C
C     ERROR PROCESSING
C
      WRITE(LU,1000) IGET
 1000 FORMAT(" ERROR CODE = ",I15)
      CALL EXEC(6)
      END
C
```

```
C
      SUBROUTINE MOVE(AINT,ICOLS,IMXP1)
C
C       THIS SUBROUTINE MOVES ICOLS ELEMENTS IN ARRAY AINT FROM
C            A START POINT OF 1 TO A START POINT OF IMXP1
C
      DIMENSION AINT(1)
C
      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
C
C     THIS PROGRAM FROMS A PART OF THE IMAGE PROCESSING SYSTEM
C
C     IT IS USED TO STORE AN IMAGE ON TAPE AND THEN PURGE FROM DIS
C
C     THE IMAGE INVENTOPRY FILE IS UPDATED TO SHOW THAT THE IMAGE
C          TAPE
C
C     WRITTEN BY WINSER E. ALEXANDER
C
      DIMENSION IDCB1(272),IDCB2(528),IMAGE(6),IPRAM(5),JNAME(3)
      DIMENSION IDATA(512),ISIZE(2),IRTN(5),IBUF(15)
C
      EQUIVALENCE (IBUF(12),ILOC),(IBUF(13),JNAME),(IBUF(7),NLINE)
      EQUIVALENCE (IBUF(8),NPIXL),(IBUF(9),IPMIN),(IBUF(10),IPMAX)
C
C     GET INPUT PARAMETERS
C
      CALL RMPAR(IPRAM)
      LU = IPRAM(1)
      IF(LU.LE.0) LU=1
C
      LU2 = 8
C
C POSITION TAPE
C
      WRITE(LU,45)
      READ(LU,46) IOPT
      IF (IOPT .NE. 2HGO) GO TO 1000
      WRITE(LU,51)
      READ(LU,*) IFNUM
C
C  SPACE TO FILE POSITION
C
      CALL EXEC(3,400B+LU2)
      IF (IFNUM .LE. 0) GO TO 1000
      IF (IFNUM .EQ. 1) GO TO 5
C
```

```
      DO 55 I=1,IFNUM-1
      CALL EXEC(3,1300B+LU2)
55    CONTINUE
C
C     GET IMAGE NAME FROM USER
C
C
5     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
C
C     OPEN DIRECTORY FILE
C
   30 CALL OPEN(IDCB1,IERR,6HIMDIRC,1,2HIM,23,272)
      IF(IERR.LT.0) GO TO 999
C
C     FIND IMAGE FILE
C
   40 CALL READF(IDCB1,IERR,IBUF,15,LEN)
      IF(LEN.NE.-1) GO TO 35
      WRITE(LU,36)
36    FORMAT("IMAGE NOT FOUND")
      GO TO 5
35    IF(IERR.LT.0) GO TO 999
C
      IF(ICMPW(IMAGE,IBUF,6).NE.0) GO TO 40
C
C     IMAGE FOUND
C
C     CLOSE DIRECTORY FILE AND OPEN IMAGE FILE
C
      CALL CLOSE(IDCB1)
C
      CALL OPEN(IDCB2,IERR,JNAME,1,2HIM,23,528)
      IF(IERR.LT.0) GO TO 999
C
C     CHECK FOR TAPE ON TRANSPORT
C
45    FORMAT(" PUT TAPE ON TRANSPORT & PUT TAPE UNIT ON LINE."
     */"  ENTER -GO- WHEN READY"/)
   46 FORMAT(1A2)
C
C
      WRITE(LU,48) IPMAX,IPMIN
   48 FORMAT(" MAXIMUM VALUE = ",I18,".  MINIMUM = ",I18)
C
C     IF(IPMAX.LE.255) IMAGE WILL BE PACKED FOR OUTPUT (8 BIT IMAG
C
      ITYPE = 15
      IF(IPMAX.LE.255.AND.IPMIN.GE.0) ITYPE = 8
C
C
51    FORMAT("FILE #?_")
C
C
```

```
C
C OUTPUT DATA TO TAPE
C
60      DO 80 I =1,512
        CALL FILL(IDATA,0,512)
        IERR =0
        IF (I .LE. NLINE) CALL READF(IDCB2,IERR,IDATA,512)
        IF (IERR .LT. 0) GO TO 999
C
        NOUT = 512
        IF (ITYPE .EQ. 15) GO TO 70
C
C  PACK DATA
C
        DO 65 J=1,512,2
        CALL ROT8(IDATA(J),ITEMP)
65      IDATA(J) = IOR(ITEMP,IDATA(J+1))
        NOUT = 256
C
C  WRITE DATA
C
70      CALL EXEC(2,LU2,IDATA,NOUT)
80      CONTINUE
C
        CALL EXEC(3,100B+LU2)
        CALL CLOSE(IDCB2)
        GO TO 5
C
C       FILE ERROR
C
    999 WRITE(LU,996) IERR
    996 FORMAT(" FILE ERROR = ",I14)
        CALL CLOSE(IDCB1)
        CALL CLOSE(IDCB2)
1001    CALL EXEC(3,400B+LU2)
C
1000    CONTINUE
        END
C
C
        SUBROUTINE FILL(IARAY,ICHAR,NUM)
C
C       THIS SUBROUTINE IS USED TO FILL THE ARRAY "IARAY" WITH
C             THE CHARACTER "ICHAR"
        DIMENSION IARAY(NUM)
C
        DO 10 I=1,NUM
     10 IARAY(I)=ICHAR
        RETURN
        END
        END$
```

```
&EES3  T=00004 IS ON CR00022 USING 00006 BLKS R=0053

FTN4,L
      PROGRAM PLOTV
      DIMENSION LU(5)
      INTEGER IDCB(144 ),BUFF( 4 ), NAME(3)
      DATA NAME/2HDA,2HTA,2H1 /
C
C GET LU
C
      CALL RMPAR(LU)
C
      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
C20    CONTINUE
 20    CALL READF(IDCB,IERR,BUFF,4,IERR)
      IF (IERR .GE. 0) GOTO 40
      WRITE(LU,30) IERR
30    FORMAT("READ ERROR",F5.0)
      GO TO 55
40    CONTINUE
      CALL DVECT(BUFF,BUFF(2),BUFF(3),BUFF(4),LU)
 50   GO TO 20
55    STOP
      END
      SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)
      DIMENSION IBUFF(5)
C
C   DRAWS A VECTOR BETWEEN X1,Y1 AND X2,Y2
C
      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
C
      CALL DRIVR(2,IBUFF,4)
C
      RETURN
      END
      SUBROUTINE INITA(IBACK)
      DIMENSION INIT(6)
      DATA INIT/30000B,100377B,10377B,24021B,26000B/
C
C IBACK = 1 FOR REVERSE BACKGROUND
C  INITIALIZE
C
      IF(IBACK .EQ. 1) INIT(4)=24221B
      CALL DRIVR(2,INIT,5)
      RETURN
      END
 $
```

&DPLAM T=00004 IS ON CR00022 USING 00022 BLKS R=0210

```
FTN4,L
      PROGRAM DPLAM
C
C THIS PROGRAM DISPLAYS THE FILTER CHARACTERISTICS
C
C
      COMMON/CNT/XM(30,30)
      COMMON/WORK/WO(130)
      COMMON/QDCAZ/IQ(40)
      INTEGER BUFF
      COMMON/ /IDCB(144),BUFF(10)
      DIMENSION IBUF(80),ILU(5),A(25),B(25),AA(5,5),BB(5,5)
      DIMENSION XXX(31),YYY(31),XYP(31,2),LXY(15,3),AR(60)
      DIMENSION XERR(31),CZ( 9),IREG(2),U(3,3,2),V(3,3,2)
      COMPLEX HA,HB,Z(25)
      EQUIVALENCE (AA(1,1),XM(1,1)),(BB(1,1),XYP(1,1))
      EQUIVALENCE (IREG(1),REG)
      EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
C
      CALL RMPAR(ILU)
      LU=ILU(1)
      MN=ILU(2)
      AMAX = 0.0
      AMIN =1000.0
      MDIM = 30
      NDIM = 30
C
C
C GET FILTER COEFF'S
      CALL EXEC(14,1,IBUF,80)
C
       IF(MN.EQ.3) GO TO 100
       MNL=9
       DO 10 J=1,3
       DO 10 K=1,3
       II=J+(K-1)*3
       A(II)=U(J,K,1)
   10   B(II)=V(J,K,1)
       GO TO 101
  100 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)
  102 BB(IK,JL)=BB(IK,JL)+V(I,J,1)*V(K,L,2)
       DO 11 J=1,5
       DO 11 K=1,5
       II=J+(K-1)*5
       A(II)=AA(J,K)
   11   B(II)=BB(J,K)
       MNL=25
```

```
101   WRITE(LU,1011)
1011  FORMAT(21H COEFFICIENT MATRICES,/)
      WRITE(LU,105) (A(I),I=1,25)
      WRITE(LU,105)(B(I),I=1,25)
  105 FORMAT(5(1H ,5E10.2/)/)
C
C       COMPUTE THE CENTER OF OUTPUT ARRAY
C
      WRITE(LU,12)
   12   FORMAT(" ENTER MX FOR HORIZONTAL FREQUENCIES"/)
        READ(LU,13)MX
   13   FORMAT(1I2)
      WRITE(LU,14)
   14   FORMAT(" ENTER MY FOR VERTICAL FREQUENCIES"/)
        READ(LU,13) MY
  203   MXC=MX/2
        MXT=2*MXC
        NX=0
        IF(MXT.NE.MX) NX=1
        MYC=MY/2
        MYT=2*MYC
        NY=0
        IF(MYT.NE.MY) NY=1
      MXN=MXC+1
        MYN=MYC+1
      WRITE(LU,301)
  300 FORMAT(" COMPUTE SQUARED MAGNITUDE "/)
  301 FORMAT(" INITIALIZE ARRAY"/)
C
C       COMPUTE SQUARED MAGNITUDE CHARACTERISTIC
C
        MX=MXT+NX
         MY=MYT+NY
        FCX=2.0/FLOAT(MX)
        FCY=2.0/FLOAT(MY)
        IF(MX.LE.101.AND.MY.LE.61) GO TO 204
        IF(MX.LE.101) GO TO 202
        MX=101
      WRITE(LU,200)
  200   FORMAT(" SIZE OF ARRAY WAS REDUCED TO  31 FOR HORIZONTAL "
  202   IF(MY.LE.61) GO TO 203
        MY=61
        WRITE(LU,201)
        GO TO 203
  201   FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR VERTICAL "/)
  204 WRITE(LU,300)
        DO 20 I=1,MX+1
        DO 20 J=1,MY+1
        XF=FCX*(I-MXC-1)
         XXX(I)=XF
        YF=FCY*(J-MYC-1)
         YYY(J)=YF
        CALL ZWC(Z,XF,YF,MN)
        HA=CMPLX(0.0,0.0)
        HB=HA
        DO 21 K=1,MNL
        HA=HA+A(K)*Z(K)
      HB=HB+B(K)*Z(K)
```

```
21      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
C SQUARED MAGNITUDE NORMALIZED
C
C
C         OBTAIN W=1 PLOT FROM ARRAY
C
        DO 22 I=1,MXN
        XYP(I,2)=XM(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=SQRT(X)+1
        DO 30 I=1,MXN
        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
    30   XYP(I,1)=XF*1.414
       WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,MXN)
C
```

```
C      COMPUTE ERROR FUNCTION
C
       ERR=0.0
       DO 350 J=1,MDIM
   350 ERR=ERR+((XERR(J)-XYP(J,2))/AMAX)**2
       WRITE(LU,360) ERR
   360 FORMAT(" RELATIVE ERROR = ",1E15.7/)
C      COMPUTE CONTOURS
C
C
C      PLOT IMAGE  OF TRANSFER FUNCTION
C
         CALL  CONTR(XXX,YYY,AMAX,AMIN,MX+1,MY+1,LU)
C
 4443   STOP
        END
        SUBROUTINE ZWC(Z,XF,YF,MN)
C
C        THIS SUBROUTINE COMPUTES COMPLEX
C        VALUES FOR Z**I*W**J FOR
C        ZW TRANSFORM AND PLACES RESULTS
C        IN ONE DIMENTIONAL ARRAY Z
C        XF=HORIZONTAL RELATIVE FREQUENCY
C        YF=VERTICAL RELATIVE FREQUENCY
C
        COMPLEX Z(25),R,S
         IF(ABS(XF).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
        DO 21 K=1,5
        I=J+(K-1)*5
    21   Z(I)=S**(J-1)*R**(K-1)
    22   RETURN
        END
       BLOCK DATA WORK
       COMMON/WORK/WO(130)
       COMMON/CNT/XM(900)
       COMMON/QDCAZ/IQ(40)
       END
 $
```

&DPLA1 T=00004 IS ON CR00022 USING 00052 BLKS R=0437

```
FTN4,L
          SUBROUTINE  CONTR(XXX,YYY,AMAX,AMIN,MX,MY,LU)
          COMMON/CNT/XM(30,30)
        DIMENSION XXX(MX),YYY(MY),CZ(9),ISIZE(2)
        INTEGER BUFF,NAME9(3)
        COMMON / /IDCB(144),BUFF(4)
        DATA NAME9/2HDA,2HTA,2H1 /
          WRITE(LU,100)
100       FORMAT("  SELECT TYPE OF FILTER PLOT "/
       1"  1.  CONTOUR"/"  2.  PERSPECTIVE"/)
          READ(LU,*) IFLAG
C
C       GENERATE CZ
C
        CZ(1)=-1.
        DO 3 K=2,9
        CZ(K)=CZ(K-1)+.25
      3 CONTINUE
C
C CREATE A PLOT DATA FILE
C
        ITYPE=3
        ISIZE(1)=96
        CALL PURGE(IDCB,IERR,NAME9)
        IF(IERR .LT. 0) WRITE(LU,101) IERR
        CALL CREAT(IDCB,IERR,NAME9,ISIZE,ITYPE)
         IF(IERR .GE. 0) GO TO 201
        WRITE(LU,101) IERR
101     FORMAT("CREATE ERROR",F5.0)
        STOP
C
C       DO 3-D PLOTS
C
201       IF (IFLAG.EQ.1) GO TO 10
          IF (IFLAG.EQ.2) GO TO 20
20        CONTINUE
        CALL SET3D(1.,-1.,1.,-1.,AMAX,AMIN,2,0,.5,.5)
        CALL  PLT3D(XXX,YYY,XM,30,MX,MY,LU)
          IF(IFLAG.EQ.2) GOTO 30
C
C      DO ISOGRAMS
C
10        CONTINUE
          DO 11 I=1,MX
          DO 11 J=1,MY
          XM(I,J)=XM(I,J)/AMAX
11        CONTINUE
        CALL SET2D(1.,-1.,1.,-1.,3,0,1.)
        CALL  PLT2D(XXX,YYY,XM,30,MX,MY,CZ,9,LU)
30        CONTINUE
        CALL CLOSE(IDCB)
        RETURN
        END
```

```
      SUBROUTINE SET2D(ALPMAX,ALPMIN,BETMAX,BETMIN,IORGN,IALPCL,AL
      COMMON/ QDCAZ/ IXYXYB(4,4),XZ,AX,BX,YZ,AY,BY
      DATA XCNTR,YCNTR,EL/512.,512.,1000./
      XZ=XCNTR
      YZ=YCNTR
      IF(ALTOBL-1.)6,7,8
    6 CONTINUE
      ELALP=ALTOBL*EL
      ELBET=EL
      GO TO 9
    7 CONTINUE
      ELALP=EL
      ELBET=EL
      GO TO 9
    8 CONTINUE
      ELALP=EL
      ELBET=EL/ALTOBL
    9 CONTINUE
      IF (IORGN.EQ.1) GO TO 1
      IF (IORGN.EQ.2) GO TO 2
      IF (IORGN.EQ.3) GO TO 3
      GO TO 4
    1 CONTINUE
      IF (IALPCL.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(IALPCL.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*ELBET
      GO TO 5
   30 CONTINUE
      AX=0.
      BY=0.
      AY=ELALP/(ALPMAX-ALPMIN)
      BX=ELBET/(BETMAX-BETMIN)
      XZ=XZ-.5*ELBET
      YZ=YZ-.5*ELALP
      GO TO 5
    4 CONTINUE
      IF(IALPCL.EQ.1) GO TO 40
      AX=0.
      BY=0.
      AY=-ELALP/(ALPMAX-ALPMIN)
      BX=ELBET/(BETMAX-BETMIN)
      XZ=XZ-.5*ELBET
      YZ=YZ+.5*ELALP
      GO TO 5
   40 CONTINUE
      AY=0.
      BX=0.
      AX=ELALP/(ALPMAX-ALPMIN)
      BY=-ELBET/(BETMAX-BETMIN)
      XZ=XZ-.5*ELALP
      YZ=YZ+.5*ELBET
    5 CONTINUE
      XZ=XZ-AX*ALPMIN-BX*BETMIN
      YZ=YZ-AY*ALPMIN-BY*BETMIN
      IXYXYB(1,1)=IFIX(XZ+AX*ALPMIN+BX*BETMIN)
      IXYXYB(2,1)=IFIX(YZ+AY*ALPMIN+BY*BETMIN)
      IXYXYB(1,2)=IFIX(XZ+AX*ALPMIN+BX*BETMAX)
      IXYXYB(2,2)=IFIX(YZ+AY*ALPMIN+BY*BETMAX)
      IXYXYB(1,3)=IFIX(XZ+AX*ALPMAX+BX*BETMAX)
      IXYXYB(2,3)=IFIX(YZ+AY*ALPMAX+BY*BETMAX)
      IXYXYB(1,4)=IFIX(XZ+AX*ALPMAX+BX*BETMIN)
      IXYXYB(2,4)=IFIX(YZ+AY*ALPMAX+BY*BETMIN)
      IXYXYB(3,1)=IXYXYB(1,2)
      IXYXYB(4,1)=IXYXYB(2,2)
      IXYXYB(3,2)=IXYXYB(1,3)
      IXYXYB(4,2)=IXYXYB(2,3)
      IXYXYB(3,3)=IXYXYB(1,4)
      IXYXYB(4,3)=IXYXYB(2,4)
      IXYXYB(3,4)=IXYXYB(1,1)
      IXYXYB(4,4)=IXYXYB(2,1)
      RETURN
      END
```

```
C     ************************************************************************
C
C
      SUBROUTINE  PLT2D(ALPHA,BETA,GAMMA,IDMN,IALPHA,JBETA,C,NUMC
     1    ,IFILE,LU)
      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(IDCB,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)
      IMAXP2=IMAX+2
      IF (JBETA)4,1,5
    4 CONTINUE
      NOGRID=1
    5 CONTINUE
      JMAX=IABS(JBETA)
      IF (NOGRID.EQ.1) GO TO 6
      DO 7 K=1,4
      CALL FLBUF(IXYXYB(1,K),IXYXYB(2,K),
     1 IXYXYB(3,K),IXYXYB(4,K),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
    7 CONTINUE
    6 CONTINUE
      DO 8 N=1,NUMC
      DO 9 I=1,IMAXP2
      IXIY(1,I)=0
      JXJY(1,I)=0
    9 CONTINUE
      DO 10 J=1,JMAX
      IF(J.EQ.1) GO TO 111
      DO 12 I=1,IMAX
      IF(GAMMA(I,J).EQ.GAMMA(I,J-1)) GO TO 13
      IF (GAMMA(I,J).GE.C(N).AND.C(N).GE.GAMMA(I,J-1)) GO TO 14
      IF(GAMMA(I,J).LE.C(N).AND.C(N).LE.GAMMA(I,J-1)) GO TO 14
   13 CONTINUE
      JXJY(1,I+1)=0
      GO TO 12
   14 CONTINUE
      BETINT=BETA(J-1)+(BETA(J)-BETA(J-1))*(C(N)-GAMMA(I,J-1))/(GA
     1J)-GAMMA(I,J-1))
      ALPINT=ALPHA(I)
      IXR=IFIX(XZ+AX*ALPINT+BX*BETINT)
```

```
      IYR=IFIX(YZ+AY*ALPINT+BY*BETINT)
      IF(JXJY(1,I).EQ.0) GO TO 15
       CALL FLBUF(IXR,IYR,JXJY(1,I),JXJY(2,I),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   15 CONTINUE
      IF(IXIY(1,I+1).EQ.0) GO TO 16
       CALL FLBUF(IXR,IYR,IXIY(1,I+1),IXIY(2,I+1),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   16 CONTINUE
      IF(IXIY(1,I+2).EQ.0) GO TO 17
       CALL FLBUF(IXR,IYR,IXIY(1,I+2),IXIY(2,I+2),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   17 CONTINUE
      JXJY(1,I+1)=IXR
      JXJY(2,I+1)=IYR
   12 CONTINUE
111      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
   19 CONTINUE
      IXIY(1,I+1)=0
      GO TO 18
   20 CONTINUE
      ALPINT=ALPHA(I-1)+(ALPHA(I)-ALPHA(I-1))*(C(N)-GAMMA(I-1,J))/
     1(I,J)-GAMMA(I-1,J))
      BETINT=BETA(J)
      IXR=IFIX(XZ+AX*ALPINT+BX*BETINT)
      IYR=IFIX(YZ+AY*ALPINT+BY*BETINT)
      IF(JXJY(1,I).EQ.0) GO TO 21
      CALL FLBUF(IXR,IYR,JXJY(1,I),JXJY(2,I),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   21 CONTINUE
      IF (IXIY(1,I+1).EQ.0) GO TO 22
       CALL FLBUF(IXR,IYR,IXIY(1,I+1),IXIY(2,I+1),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   22 CONTINUE
      IF(JXJY(1,I+1).EQ.0) GO TO 23
       CALL FLBUF(IXR,IYR,JXJY(1,I+1),JXJY(2,I+1),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   23 CONTINUE
      IXIY(1,I+1)=IXR
      IXIY(2,I+1)=IYR
   18 CONTINUE
   10 CONTINUE
    8 CONTINUE
    1 CONTINUE
      RETURN
      END
```

```
C     ******************************************************************
C
C
C
      SUBROUTINE SET3D(ALPMAX,ALPMIN,BETMAX,BETMIN,GAMMAX,GAMMIN,
     *ORGN,IALPCL,GAMFAC,ALPFAC)
      COMMON/QDCAZ /IXYXYB(4,5),XZ,AX,BX,YZ,AY,BY,CY
      DATA ELX,ELY,EXLEFT,YBOTOM/1012.,856.,12.,156./
      AX=ALPFAC*ELX/(ALPMAX-ALPMIN)
      AY=ALPFAC*(1.-GAMFAC)*ELY/(ALPMAX-ALPMIN)
      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(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
      XZ=XZ+ELX
      AX=-AX
      BX=-BX
      IF(IALPCL.EQ.1)GO TO 10
      YZ=YZ+BY*(BETMAX-BETMIN)
      BY=-BY
      ALPVRT=ALPMAX
      BETVRT=BETMIN
      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(IALPCL.EQ.1)GO TO 30
      YZ=YZ+AY*(ALPMAX-ALPMIN)
      AY=-AY
      ALPVRT=ALPMIN
      BETVRT=BETMAX
      GO TO 5
   30 CONTINUE
      YZ=YZ+BY*(BETMAX-BETMIN)
      BY=-BY
      ALPVRT=ALPMAX
      BETVRT=BETMIN
      GO TO 5
    4 CONTINUE
      YZ=YZ+ELY*(1.-GAMFAC)
      ALPVRT=ALPMIN
      BETVRT=BETMIN
      AY=-AY
      BY=-BY
      IF(IALPCL.EQ.1)GO TO 40
      XZ=XZ+AX*(ALPMAX-ALPMIN)
      AX=-AX
      GO TO 5
   40 CONTINUE
      XZ=XZ+BX*(BETMAX-BETMIN)
      BX=-BX
    5 CONTINUE
      XZ=XZ-BX*BETMIN-AX*ALPMIN
      YZ=YZ-BY*BETMIN-AY*ALPMIN
      IXYXYB(1,1)=XZ+AX*ALPMIN+BX*BETMIN
      IXYXYB(2,1)=YZ+AY*ALPMIN+BY*BETMIN+CY*GAMMIN
      IXYXYB(3,1)=XZ+AX*ALPMAX+BX*BETMIN
      IXYXYB(4,1)=YZ+AY*ALPMAX+BY*BETMIN+CY*GAMMIN
      IXYXYB(1,2)=IXYXYB(3,1)
      IXYXYB(2,2)=IXYXYB(4,1)
      IXYXYB(3,2)=XZ+AX*ALPMAX+BX*BETMAX
      IXYXYB(4,2)=YZ+AY*ALPMAX+BY*BETMAX+CY*GAMMIN
      IXYXYB(1,3)=IXYXYB(3,2)
      IXYXYB(2,3)=IXYXYB(4,2)
      IXYXYB(3,3)=XZ+AX*ALPMIN+BX*BETMAX
      IXYXYB(4,3)=YZ+AY*ALPMIN+BY*BETMAX+CY*GAMMIN
      IXYXYB(1,4)=IXYXYB(3,3)
      IXYXYB(2,4)=IXYXYB(4,3)
      IXYXYB(3,4)=IXYXYB(1,1)
      IXYXYB(4,4)=IXYXYB(2,1)
      IXYXYB(1,5)=XZ+AX*ALPVRT+BX*BETVRT
      IXYXYB(2,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMIN
      IXYXYB(3,5)=IXYXYB(1,5)
      IXYXYB(4,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMAX
45      FORMAT (5(7X,I7))
      RETURN
      END
```

```
C     ********************************************************************
C
C
C
      SUBROUTINE  PLT3D(ALPHA,BETA,GAMMA,IDMN,IALPHA,JBETA,IFILE,L
      DIMENSION ALPHA(1),BETA(1),GAMMA(IDMN,1)
      COMMON/WORK/LASTXY(2,200)
      COMMON/QDCAL /IXYXYB(4,5),XZ,AX,BX,YZ,AY,BY,CY
      INTEGER BUFF(4)
      COMMON IDCB(144)
      NOGRID=0
      IF(IALPHA)1,2,3
    1 CONTINUE
      NOGRID=1
    3 CONTINUE
      IMAX=IABS(IALPHA)
      IF(JBETA)4,2,5
    4 CONTINUE
      NOGRID=1
    5 CONTINUE
      JMAX=IABS(JBETA)
      IF(NOGRID.EQ.1)GO TO 6
67      FORMAT (5(7X,I7))
68      FORMAT (10X,"GOOD",I5)
      DO 7 K=1,5
      CALL FLBUF( IXYXYB(1,K),IXYXYB(2,K),
     1  IXYXYB(3,K),IXYXYB(4,K),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
    7 CONTINUE
    6 CONTINUE
      DO 8 J=1,JMAX
      DO 8 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(I.EQ.1)GO TO 9
        CALL FLBUF(IXR,IYR,LASTXY(1,I-1),LASTXY(2,I-1),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
    9 CONTINUE
      IF(J.EQ.1)GO TO 10
        CALL FLBUF(IXR,IYR,LASTXY(1,I),LASTXY(2,I),BUFF)
      CALL WRITF(IDCB,IERR,BUFF,4)
   10 CONTINUE
        LASTXY(1,I)=IXR
        LASTXY(2,I)=IYR
    8 CONTINUE
    2 CONTINUE
      RETURN
      END
      SUBROUTINE FLBUF(IX1,IY1,IX2,IY2 ,BUFF)
      INTEGER BUFF(4)
      BUFF(1) = IX1
      BUFF(2) = IY1
      BUFF(3) = IX2
      BUFF(4) = IY2
      RETURN
      END
$
```

&FDIGN T=00004 IS ON CR00022 USING 00056 BLKS R=0498

```
FTN4,L
      PROGRAM FDIGN
C
C THIS PROGRAM SCHEDULE  FILTER DESIGN,STABILITY,AND DISPLAY
C
      COMMON/WORK/WO(75)
       DIMENSION NAME1(3),NAME2(3),NAME3(3),IRTN(5)
      DIMENSION IDCB(144),NAME4(3),NAME5(3)
      DIMENSION U(3,3,2),V(3,3,2),ILU(5),IBUF(80),IBUF2(80)
      EQUIVALENCE (ILU(1),LU),(IBUF,IBUF2)
      EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
      DATA NAME1/2HST,2HAB,2HI /
      DATA NAME2/2HDP,2HLA,2HM /
      DATA NAME3/2HCO,2HEF,2HFS/
      DATA NAME4/2HFI,2HRO,2H  /
      DATA NAME5/2HPL,2HOT,2HV /
      DATA V/18*0./
      DATA U/18*0./
      DATA IBUF/80*0/
C
C GET LU
      CALL RMPAR(ILU)
C
C     GET FILTER PARAMETERS
C
      MN=1
4      WRITE(LU,400)
  400 FORMAT(" SELECT FILTER DESIGN"/"  1.  LOWPASS"/"  2.  BANDPA
     1 3.  HIGHPASS"/ "  4.  BOOST FILTER"/"  5.  TDLPF (LOWPASS)"
     2"  6.  ROTATING FILTER  "/"  7.  NON-RECURSIVE FILTERS " )
      READ(LU,401) IFIL
      IF(IFIL.EQ.4) GO TO 500
      IF(IFIL.EQ.3)GO TO 410
        IF(IFIL.EQ.6) GO TO 408
      IF(IFIL .EQ. 7) GO TO 1102
      WRITE(LU,402)
  402 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS"/)
      READ(LU,403)F2
  403 FORMAT(F2.2)
      IF(IFIL.NE.2) GO TO 407
      WRITE(LU,404)
  404 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR HIGHPASS"/)
      READ(LU,403) F1
  407 WRITE(LU,405)
  405 FORMAT(" ENTER NUMBER OF FILTER STAGES"/)
      READ(LU,401) MN
      IBUF(40) =MN
  401 FORMAT(1I1)
      GO TO 411
  500 WRITE(LU,510)
  510 FORMAT(" SELECT OPTION"/,"  1.  LOW BOOST FILTER"/,"  2.  HI
     *ST FILTER"/)
      READ(LU,515) IOPT
  515 FORMAT(1I1)
      IF(IOPT.GE.0.AND.IOPT.LE.2) GO TO 530
```

```
      WRITE(LU,520)
  520 FORMAT(" INVALID RESPONSE"/)
       GO TO 500
C
  530 WRITE(LU,535)
  535 FORMAT(" ENTER BOOST MAGNITUDE"/)
      READ(LU,*) BF
      WRITE(LU,540)
  540 FORMAT(" ENTER RELATIVE BREAK FREQUENCY"/)
      READ(LU,403) F1
      IF(IOPT.EQ.0.OR.IOPT.EQ.1) WRITE(LU,545) BF,F1
  545 FORMAT(" BOOST MAGNITUDE = ",1E15.5," FREQUENCY = ",1F10.5,"
     *OWBOOST FILTER."/,"   IS THIS CORRECT?"/)
      IF(IOPT.EQ.2) WRITE(LU,550) BF,F1
  550 FORMAT(" BOOST MAGNITUDE = ",1E15.5," BREAK FREQUENCY = ",1F
     * FOR HIGH BOOST FILTER."/,"   IS THIS CORRECT"/)
      READ(LU,551) IRES
  551 FORMAT(1A1)
      IF(IRES.EQ.1HY.OR.IRES.EQ.1Hy) GO TO 552
      GO TO 530
  552 BF=SQRT(ABS(BF))
      IF(IOPT.EQ.2) GO TO 560
      ALP=1.0
      BET=BF-1.0
      GO TO 412
  560 ALP=BF
      BET=1.0-BF
      GO TO 412
  410 WRITE(LU,404)
      READ(LU,403) F1
  412 WRITE(LU,405)
      READ(LU,401) MN
  411 MN=MN+1
  408 IF(IFIL.EQ.1) CALL LPFLT(U,V,F2,MN,LU)
      IF(IFIL.EQ.2) CALL BPFLT(U,V,F1,F2,MN,LU)
      IF(IFIL.EQ.3) CALL  BSTFT(U,V,F1,MN,1.0,-1.0,LU)
      IF(IFIL.EQ.4) CALL  BSTFT(U,V,F1,MN,ALP,BET,LU)
      IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
 IL.E     CALL BPFLT(U,V,F1,F2,MN,LU)
      IF(IFIL.EQ.3) CALL  BSTFT(U,V,F1,MN,1.0,-1.0,LU)
      IF(IFIL.EQ.4) CALL  BSTFT(U,V,F1,MN,ALP,BET,LU)
      IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
 IL.E       IF(IFIL.EQ.6) CALL  ROTAE(U,V,MN,LU)
C     IF(IFIL .EQ. 7) CALL FIR(U,MN,WR,LU)
      CONTINUE
      ILU(2) = MN
      IF(IFIL.EQ.2) ILU(2) =MN + 1
      IF(IFIL.EQ.6) ILU(2)=MN-1
C
C SCHEDULE  STABILITY TEST-STABT
C
      CALL EXEC(23,NAME1,LU,ILU(2),0,0,0,IBUF,80)
C
C
C  SCHEDULE DISPLAY PROGRAM-DPLAY
      IF(IFIL.EQ.3) ILU(2) =MN + 1
      IF(IFIL.EQ.4) ILU(2) =MN+1
      CALL EXEC(23,NAME2,LU,ILU(2),0,0,0,IBUF2,80)
C
```

```
      IF(IFIL .EQ. 3) IBUF(40) =MN
      IF(IFIL .EQ. 4) IBUF(40) =MN
      IF(IFIL .EQ. 6) IBUF(40)=MN-1
C
      CALL PURGE(IDCB,IERR,NAME3,2HES)
      IF(IERR .LT. 0) WRITE(LU,1101) IERR
      CALL CREAT(IDCB,IERR,NAME3,2,3,2HES)
      IF(IERR .LT. 0) WRITE(LU,1101) IERR
1101  FORMAT("CREATE ERROR",F5.0)
      CALL WRITF(IDCB,IERR,IBUF,80)
      IF(IERR .LT. 0) WRITE(LU,1101) IERR
      CALL CLOSE(IDCB,IERR)
      WRITE(LU,1105)
1105  FORMAT(" ENTER DISPLAY DEVICE "//"  1. TV"/"  2. HP2648A")
      READ(LU,*) IDEV
C
      IF(IDEV .EQ. 2) GO TO 1106
      CALL EXEC(23,NAME5,LU,0,0,0,0)
      GO TO 1107
1106  CONTINUE
       CALL HP48A(LU)
1107  CONTINUE
      CALL EXEC(6)
C
C SCHEDULE NON-RECURSIVE FILTERS
1102  CONTINUE
      CALL EXEC(23,NAME4,LU,0,0,0,0)
      STOP
      END
      SUBROUTINE BPFLT(U,V,F1,F2,N,LU)
C
C          WRITTEN BY W. E. ALEXANDER
C
C     F1----BREAK FREQUENCY FOR LOW FREQUENCY CUTOFF
C     F2----BREAK FREQUENCY FOR HIGH FREQUENCY CUTOFF
C
C     SUBROUTINE DESIGNS BANDPASS FILTER FROM LPFLT AND HPFLT
C
      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
C
      FC=AMAX1(F1,F2)
      CALL LPFLT(AA,BB,FC,N,LU)
C
      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)
C
      FC=AMIN1(F1,F2)
      CALL  BSTFT(AA,BB,FC,N,1.0,-1.0,LU)
```

```
C
      DO 21 I=1,3
      DO 21 J=1,3
      U(I,J,2)=AA(I,J,1)
   21 V(I,J,2)=BB(I,J,1)
C

      RETURN
      END
      SUBROUTINE  BSTFT(U,V,FC,N,ALP,BET,LU)
C     FREQUENCY BOOST DESIGN ROUTINE
C     FC=RC*S/PI WHERE RC IS THE CUTOFF FREQUENCY IN RADIANS AND
C            S IS THE SAMPLING INTERVAL.  THUS FC=0.5 GIVES A CUTO
C            FREQUENCY AT ONE FOURTH SAMPLING FREQUENCY.
C
C     FOR HIGH PASS FILTER, LET ALP=1.0 AND BET=-1.0
C     FOR HIGH FREQUENCY BOOST FILTER, ALP=BF AND BET=-1.0*(BF-1.0
C          WHERE BF=SQRT(DESIRED FILTER GAIN AT ONE HALF SAMPLING
C     FOR LOW FREQUENCY BOOST FILTER, ALP=1.0 AND BET=(BF-1.0)
C
C
      DIMENSION U(3,3,2),V(3,3,2)
C
      DO 21 K=1,2
      DO 21 I=1,3
      DO 21 J=1,3
      U(I,J,K)=0.0
   21 V(I,J,K)=0.0
      WRITE(LU,14) FC
   14 FORMAT(1H0," FC = ",1E22.5," FOR BOOST FILTER"/)
      PI=3.141592654
      D=1.0E-10
      PWR=0.25
      IF (N.EQ.3) PWR=0.125
      EPS=2.0**PWR-1.0
      IF(N.EQ.2.AND.BET.LT.0.0) EPS=1.50702
      IF(N.EQ.3.AND.BET.LT.0.0) EPS=2.4711
       XP=PI*FC*0.5
      T=(SIN(XP)/COS(XP))**2.0
      IF(T.GT.D) GO TO 10
      AAA=EPS/D
      GO TO 11
   10 AAA=EPS/T
   11 SALP=SQRT(AAA)
      DEM=1.0-2.0*AAA
      IF(DEM.LT.D) GO TO 12
   13 P1=(+2.0*AAA-2.0*SQRT(2.0)*SALP+1.0)/DEM
      GO TO 20
   12 F=-1.0*DEM
      IF(F.GT.D) GO TO 13
      P1=0.0
C
   20 A=((1.0+P1)**2)/4.0
      AS=A**2
      POS=P1**2
      R=4.0*(POS-AS)+((1.0+POS)**2-4.0*AS)-4.0*(P1*(1.0+POS)-2.0*A
      IF(ABS(R).LT.D) R=SIGN(D,R)
      S=((1.0-P1)**4)/R
C
```

```
      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+4.0*BET*AS)
      U(1,3,1)=S*(ALP*POS+BET*AS)
      U(3,1,1)=U(1,3,1)
      U(2,3,1)=S*(ALP*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)
C
      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
C
      IF(N.EQ.2) GO TO 27
      DO 26 I=1,3
      DO 26 J=1,3
      U(I,J,2)=U(I,J,1)
   26 V(I,J,2)=V(I,J,1)
27    N = N-1
      RETURN
      END
      SUBROUTINE LPFLT(U,V,FC,N,LU)
C     LOW PASS RECURSIVE FILTER DESIGN ROUTINE
C     FC#RC*S/PI WHERE RC IS THE CUTOFF FREQUENCY IN RADIANS AND
C            S IS THE SAMPLING INTERVAL.  THUS FC#0.5 GIVES A CUTO
C            FREQUENCY AT ONE FOURTH SAMPLING FREQUENCY.
C
      DIMENSION U(3,3,2),V(3,3,2)
      COMMON/WORK/A(5,5),B(5,5)
C
      DO 21 K=1,2
      DO 21 I=1,3
      DO 21 J=1,3
      U(I,J,K)=0.0
   21 V(I,J,K)=0.0
      IF(FC.GE.0.99) FC=0.99
      WRITE(LU,14)FC
   14 FORMAT(1H0," 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
```

```
      P2=P1
      IF(FC.GT.0.3)GO TO 10
      P2=(SQRT(T)-EPS)/(SQRT(T)+EPS)
      GO TO 20
   12 F=-1.0*DEM
      IF(F.GT.D) GO TO 13
      P1=0.0
      P2=0.0
   20 V(1,1,1)=1.0
      V(1,2,1)=P1+P2
      V(2,1,1)=V(1,2,1)
      V(1,3,1)=P1*P2
      V(3,1,1)=V(1,3,1)
      V(2,2,1)=V(1,2,1)**2
      V(2,3,1)=P1*P2**2+P2*P1**2
      V(3,2,1)=V(2,3,1)
      V(3,3,1)=V(1,3,1)**2
C
      SUM=0.0
      DO 25 I=1,3
      DO 25 J=1,3
   25 SUM=SUM+V(I,J,1)
C
      U(1,1,1)=SUM/16.0
      U(1,3,1)=U(1,1,1)
      U(3,1,1)=U(1,1,1)
      U(3,3,1)=U(1,1,1)
      U(1,2,1)=SUM/8.0
      U(2,1,1)=U(1,2,1)
      U(2,3,1)=U(1,2,1)
      U(3,2,1)=U(1,2,1)
      U(2,2,1)=SUM/4.0
      IF(N.EQ.2) GO TO 1
      DO 26 I=1,3
      DO 26 J=1,3
      U(I,J,2)=U(I,J,1)
   26 V(I,J,2)=V(I,J,1)
      GO TO 2
    1 U(1,1,2)=1.0
      V(1,1,2)=1.0
    2 DO 30 I=1,5
      DO 30 J=1,5
      A(I,J)=0.0
   30 B(I,J)=0.0
      DO 31 I=1,5
      DO 31 J=1,5
      DO 31 K=1,3
      DO 31 L=1,3
      IK=I-K+1
      JL=J-L+1
      IF(IK.LE.0.OR.IK.GT.3)GO TO 31
      IF(JL.LE.0.OR.JL.GT.3)GO TO 31
      A(I,J)=A(I,J)+U(IK,JL,1)*U(K,L,2)
      B(I,J)=B(I,J)+V(IK,JL,1)*V(K,L,2)
   31 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TDLPF(A,B,MN,RC,NDIM,LU)
C     INPUTS
C          N - NUMBER OF FILTER STAGES
C          RC - RELATIVE CUTOFF FREQUENCY FOR FILTER
C          NDIM - ARRAY DIMENSION IN CALLING PROGRAM
C     OUTPUTS
C          A - COEFFICIENT ARRAY (NUMERATOR)
C          B - COEFFICIENT ARRAY (DENOMINATOR)
C
      DIMENSION A(3,3,NDIM),B(3,3,NDIM)
      COMPLEX P(10),PK,Q,Z1,Z2
C
C     INITIALIZE
C
      N=MN-1
      PI=3.141592654
      D=1.0E-10
      IF(N.GT.NDIM) GO TO 300
      IF(0.01.GT.RC.OR.0.99.LT.RC) GO TO 400
      AA=0.5*PI*RC
      AA=SIN(AA)/COS(AA)
      PWR=1.0/FLOAT(N)
C      EPS=SQRT(2.0)-1.0
      EPS=1.0
      EPS=EPS**PWR
      C=AA**2/EPS
C
C     FIND ROOTS
C
      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))
C      WRITE(LU,14) PK
   14 FORMAT(" PK = ",1E15.5," + J",1E15.5/)
      Q=2.0-PK
      IF(CABS(Q).LE.D) Q=D
      IF(CABS(PK).LE.D) PK=SIGN(D,REAL(PK))
      Z1=(2.0+PK+2.0*CSQRT(2.0*PK))/Q
C      WRITE(LU,12) Z1
   12 FORMAT(" Z1 = ",1E15.5," + J",1E15.5/)
      IF(CABS(Z1).GE.1.0) GO TO 15
      P(L)=Z1
C      WRITE(LU,11) L,P(L)
   11 FORMAT(" P(",1I2,") = ",1E15.5," + J",1E15.5/)
      L=L+1
   15 Z2=(2.0+PK-2.0*CSQRT(2.0*PK))/Q
C      WRITE(LU,13) Z2
   13 FORMAT(" Z2 = ",1E15.5," + J",1E15.5/)
      IF(CABS(Z2).GE.1.0) GO TO 20
      P(L)=Z2
C      WRITE(LU,11) L,P(L)
      L=L+1
   20 IF((L-1).EQ.NN) GO TO 25
   10 CONTINUE
```

```
C
C     PAIR COMPLEX PAIRS OF ROOTS
C
   25 L=1
      DO 30 K=1,NN
      S1=AIMAG(P(K))
      IF(S1.LT.0.0) GO TO 30
      P(L)=P(K)
      L=L+1
   30 CONTINUE
C
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=CABS(P(K))**2
      AM=(1.0+C1+C2)**2/16.0
C
      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
C
      GO TO 600
  300 WRITE(LU,310)
  310 FORMAT(" NUMBER OF STAGES TOO LARGE FOR DIMENSION"/)
      GO TO 600
  400 WRITE(LU,410)
  410 FORMAT(" FREQUENCY SPECIFICATION OUT OF RANGE"/)
      GO TO 600
  500 WRITE(LU,510)
  510 FORMAT(" NUMBER OF ROOTS LESS THAN EXPECTED"/)
  600 RETURN
      END
```

```
      BLOCK DATA WORK
      COMMON/WORK/WO(75)
      END
      SUBROUTINE HP48A(LU)
      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    FORMAT  ("OPEN ERROR",F5.0)
      STOP
30    CALL GRAFC(1,LU)
20    CALL READF(IDCB,IERR,BUFF,4,ILOG)
      IF(ILOG .EQ. -1) CO TO 55
      IF (IERR .GE. 0) GOTO 40
      WRITE(LU,31) IERR
31    FORMAT("READ ERROR",F5.0)
      GO TO 55
40    CONTINUE
      CALL DVECT(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
C  GRAPHICS OFF=0; GRAPHICS  ON NOT=0
C
        IF(IFLAG.EQ.0) GO TO 100
C
C  GRAPHIC ON
C
        WRITE(LU,10) IESC
10      FORMAT(1R2,"*dC")
        WRITE(LU,12) IESC
12      FORMAT(1R2,"*dF")
        WRITE(LU,14) IESC
14      FORMAT(1R2,"*dA")
C
```

```
              GO TO 200
        C
        C  GRAPHICS OFF
        C
        100     WRITE(LU,30) IESC
        30      FORMAT(1R2,"*dd")
                WRITE(LU,40) IESC
        40      FORMAT(1R2,"*dE")
        200     RETURN
                END
              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 (IX0,IY0) DEFINES THE
        C         THE ORIGIN.
        C
                IX0=0
                IY0=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 + IY0
                JY2 = Y2 + IY0
              WRITE(LU,10) JX1,JY1,JX2,JY2
        10    FORMAT("pa",1I3,1H,,1I3,1H,,1I3,1H,,1I3,"Z")
              RETURN
              END
              END$
        $
```

&BLDIM T=00004 IS ON CR00022 USING 00034 BLKS R=0330

```
FTN4
      PROGRAM BLDIM
C
C  THIS PROGRAM BUILDS AN IMAGE FILE FOR THE NCA&T IMAGE DISPLAY
C  SYSTEM.  IMAGE FILES MAY BE GENERATED FROM THE GMR-27 DISPLAY,
C  TAPES OR DISC (TYPE 2 FILES).
C
C  PROGRAMMER: DLJ
C
      DIMENSION LU(5),IDCB1(272),IDCB2(528),NAME(6),ISIZE(2),IDATA
      DIMENSION JNAME(3),IBUF(6)
C
      INTEGER ENTRY(256),TEXT1(40),TEXT2(40),TEXT3(40),RDREC
C
      EQUIVALENCE (ENTRY,NAME),(ENTRY(7),NLINE),(ENTRY(8),NPIXL),
     1 (ENTRY(9),IPMIN),(ENTRY(10),IPMAX),(ENTRY(11),ISRC),
     2 (ENTRY(13),JNAME),(ENTRY(129),TEXT1),(ENTRY(169),TEXT2),
     3 (ENTRY(209),TEXT3),(ENTRY(12),ILOC)
      EQUIVALENCE (JNAME(2),JNAM2),(JNAME(3),JNAM3),(ISIZE(2),ISIZ
C
C  CONSTANTS
C       MPIXL = MAXIMUM PIXELS/LINE (WHEN CHANGING BE SURE TO MODIF
C                                    ARRAY SIZES)
C
      DATA MPIXL/512/
C
C  GET INPUT PARAMETERS
C
      CALL RMPAR(LU)
      IF (LU .LE. 0) LU = 1
C
C  OUTPUT HEADING
C
      WRITE(LU,1)
1     FORMAT(//"       B U I L D   I M A G E   S U B S Y S T E M"
C
C  OPEN DIRECTORY FILE
C
      CALL OPEN(IDCB1,IERR,6HIMDIRC,0,2HIM,23,272)
      IF (IERR .LT. 0) GO TO 9999
C
C  GET IMAGE NAME
C
1000  WRITE(LU,2)
2     FORMAT("ENTER 12 CHARACTER IMAGE NAME?(/E TO EXIT)_")
      READ(LU,3) NAME
3     FORMAT(6A2)
      IF (NAME .EQ. 2H/E) GO TO 1060
C
C  CHECK FOR DUPLICATE NAME
C
```

```
      IREC = 0
      KREC = 0
      CALL RWNDF(IDCB1,IERR)
      IF (IERR .LT. 0) GO TO 9999
1010  IREC = IREC + 1
      CALL READF(IDCB1,IERR,IBUF,6,LEN)
      IF (IERR .LT. 0) GO TO 9999
      IF (LEN .EQ. -1) GO TO 1030
C
C  COMPARE NAME
C
      IF (IBUF .EQ. -1) KREC = IREC
C
      DO 1020 I=1,6
      IF (NAME(I) .NE. IBUF(I)) GO TO 1010
1020  CONTINUE
C
C  DUPLICATE NAME FOUND
C
      WRITE(LU,4)
4     FORMAT("ERROR-DUPLICATE NAME")
      CALL RWNDF(IDCB1,IERR)
      IF (IERR .LT. 0) GO TO 9999
      GO TO 1000
C
C  EOF REACHED AND NO DUPLICATE FOUND
C
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
C  GET 3-LINES OF DESCRIPTIVE TEXT
C
      WRITE(LU,7)
7     FORMAT(" ENTER UP TO 3 LINES OF DESCRIPTIVE TEXT"/)
      TEXT1 = 2H
      CALL MVW(TEXT1,TEXT1(2),119)
      CALL EXEC(1,400B+LU,TEXT1,40)
      CALL EXEC(1,400B+LU,TEXT2,40)
      CALL EXEC(1,400B+LU,TEXT3,40)
C
C  GET SOURCE OF IMAGE
C
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
C
```

```
C  CREATE DATA FILE
C
      ISIZE = (FLOAT(NPIXL)*FLOAT(NLINE) + 127.)/ 128.
      ISIZ2 = NPIXL
      IF (KREC .EQ. 0) KREC = IREC
      JNAME = 2HIM
      CALL DCODE(KREC,JNAM2,JNAM3)
      CALL PURGE(IDCB2,IERR,JNAME,2HIM,23)
      CALL CREAT(IDCB2,IERR,JNAME,ISIZE,2,2HIM,23,528)
      IF (IERR .LT. 0) GO TO 9999
C
C  INITIALIZE INPUT ROUTINE
C
      IERR = RDREC(-LU,ISRC,NLINE,NPIXL)
      IF (IERR .LT. 0) GO TO 9999
C
C  GET EACH LINE AND WRITE TO FILE
C
      IPMAX = 0
      IPMIN = 377B
      DO 1050 I=1,NLINE
      IERR = RDREC(1,IDATA,IPMAX,IPMIN)
      IF (IERR .LT. 0) GO TO 9999
      CALL WRITF(IDCB2,IERR,IDATA,NPIXL)
      IF (IERR .LT. 0) GO TO 9999
C     WRITE(LU,1051) IPMAX,IPMIN
1051  FORMAT(2I12)
1050  CONTINUE
C
      CALL CLOSE(IDCB2)
C
C  WRITE DIRECTORY ENTRY
C
      IF (KREC .EQ. IREC) GO TO 1055
      CALL OPEN(IDCB1,IERR,6HIMDIRC,2,2HIM,23,272)
      IF (IERR .LT. 0) GO TO 9999
      CALL POSNT(IDCB1,IERR,KREC)
      IF (IERR .LT. 0) GO TO 9999
1055  ILOC = 1
      CALL WRITF(IDCB1,IERR,ENTRY,256)
      IF (IERR .LT. 0) GO TO 9999
      GO TO 1000
C
C  TERMINATE
C
1060  CALL CLOSE(IDCB1)
      CALL EXEC(6)
C
C  ERROR
C
9999  WRITE(LU,9)IERR
9     FORMAT(" FILE ERROR-",I6)
      CALL CLOSE(IDCB1)
      END
```

```
      INTEGER FUNCTION RDREC(ICODE,IBUF,IP1,IP2)
C
C THIS SUBROUTINE IS USED TO INPUT IMAGE FROM DISC,TAPE OR DISPLA
C
      DIMENSION IBUF(1),IDATA(1024),NAME(3),RDATA(512),IDCB(1040)
C
      LOGICAL PACKED
C
      EQUIVALENCE (IDATA,RDATA)
C
C
      IF (ICODE .GT. 0) GO TO 120
C
C INITIALIZATION
C
      NLINE = IP1
      NPIXL = IP2
      LU = -ICODE
C
      IF (LU .GT. 0) GO TO 90
C
C SPACE FOR CALL WITH NO INTERACTION
C
C
C INTERACTIVE CALL
C
90    IF (IBUF .NE. 1) GO TO 100
C
C GET DISC FILE NAME
C
      WRITE(LU,1)
1     FORMAT("ENTER DISC FILE NAME?_")
      READ(LU,2) NAME
2     FORMAT(3A2)
C
C OPEN FILE
C
      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
      IBCOD = 1
      RETURN
C
100   IF(IBUF .NE. 2) GO TO 110
C
C TAPE INPUT
C
      WRITE(LU,4)
4     FORMAT("TAPE LU?_")
      READ(LU,*) MTLU
C
```

```
C  REWIND TAPE
C
         CALL EXEC(3,MTLU+400B)
         WRITE(LU,9)
9        FORMAT(" FILE #?_")
         READ(LU,*) IFILE
         IF (IFILE .LE. 0) CALL EXEC(6)
         IF (IFILE .EQ. 1) GO TO 107
         DO 105 I=1,IFILE-1
         CALL EXEC(3,MTLU+1300B)
105      CONTINUE
C
107      WRITE(LU,3)
         READ(LU,*) IFMT
         PACKED = .TRUE.
         IF (IFMT .NE. 2) PACKED = .FALSE.
         NUM = NPIXL
         IF (PACKED) NUM = (NPIXL+1)/2
         IF (IFMT .EQ. 3) NUM = 2*NPIXL
         IBCOD = 2
         RETURN
C
110      IF (IBUF .NE. 3) GO TO 115
C
C  DISPLAY INPUT
C
         WRITE(LU,5)
5        FORMAT("ENTER START LINE,END LINE,START PIXEL,END PIXEL?_")
         READ(LU,*)ISTRTL,IENDL,ISTRTP,IENDP
         ISTEP = 1
         IF (ISTRTL .GT. IENDL) ISTEP = -1
         PACKED = .FALSE.
         NUM = NPIXL
         IBCOD = 3
         RETURN
C
C  INPUT IS WORK FILE
C
115      CALL OPEN(IDCB,IERR,6HWF0000,0,0,0,1040)
         IF (IERR .LT. 0) GO TO 999
         PACKED = .FALSE.
         NUM = 2*NPIXL
         IBCOD = 1
C
C  POSITION FILE
C
         CALL READF(IDCB,IERR,IDATA,0)
         IF (IERR .LT. 0) GO TO 999
C
         RETURN
C
C
C  DATA INPUT SECTION
C
C  BRANCH TO APPROPRIATE SUB SECTION
C
```

```
C
120   GO TO (130,140,150),IBCOD
C
C FILE INPUT
C
130   CALL READF(IDCB,IERR,RDATA,NUM)
      IF (IERR .LT. 0) GO TO 999
      IFMT=3
      GO TO 160
C
C  TAPE INPUT
C
140   CALL EXEC(1,MTLU,IDATA,NUM)
      GO TO 160
C
C  DISPLAY INPUT
C
150    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 RLINE(ISTRTL,ISTRTP,IENDP,IDATA)
      ISTRTL = ISTRTL + ISTEP
C
C   MOVE DATA TO OUTPUT ARRAY AND UNPACK IF NECESSARY
C
160   IF (.NOT. PACKED) GO TO 180
C
C  DATA IN PACKED FORMAT
C
      DO 170 I=1,NUM
      ITEMP = IDATA(I)
      CALL ROT8(ITEMP,JTEMP)
      JTEMP = IAND(JTEMP,377B)
      IF (JTEMP .GT. IP1) IP1 = JTEMP
      IF (JTEMP .LT. IP2) IP2 = JTEMP
      ITEMP = IAND(ITEMP,377B)
      IF (ITEMP .GT. IP1) IP1 = ITEMP
      IF (ITEMP .LT. IP2) IP2 = ITEMP
      IBUF(2*I-1) = JTEMP
170   IBUF(2*I) = ITEMP
      RETURN
C
C  DATA IS UNPACKED
C
180   DO 190 I=1,NPIXL
      ITEMP = IDATA(I)
      IF (IFMT .EQ. 3) ITEMP = RDATA(I)
      IF (ITEMP .GT. IP1) IP1 = ITEMP
      IF (ITEMP .LT. IP2) IP2 = ITEMP
190   IBUF(I) = ITEMP
      RETURN
C
999   RDREC = IERR
      END
$
$
```

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

```
FTN4,L
          PROGRAM LFLTR
C
C     WRITTEN BY E. E. SHERROD
C
C         PROGRAM DOES LINEAR FILTERING USING SPATIAL DOMAIN
C                RECURSIVE DIGITAL FILTERS
C
C
C
C
C
          DIMENSION A(3,3,2),B(3,3,2),ILU(5),SUM(3,2)
          DIMENSION F1(524),F2(524),F3(524)
          DIMENSION G1(1),G2(1),G3(1),IX1(3)
          DIMENSION X1(524),X2(524),X3(524)
        DIMENSION IDCB(144),NAME(3),IRTN(5)
        COMMON /IBLK/IBUF(80)
        INTEGER READL,RITEL,WFINT
          EQUIVALENCE(IBUF(1),A(1,1,1)),(IBUF(41),B(1,1,1))
        EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
        DATA NAME/2HCO,2HEF,2HFS/
C
C NROW X 512 IMAGE
C
         CALL RMPAR(ILU)
C
         LU=ILU(1)
        IPIXL=ILU(2)
        JPIXL=ILU(3)
C
C GET FILTER COEFF'S
        CALL OPEN(IDCB,IERR,NAME)
        IF(IERR .LT. 0) GO TO 9999
        CALL READF(IDCB,IERR,IBUF,80,IERR)
        IF(IERR .LT. 0) GO TO 9999
        NSTAG = IBUF(40)
        N = NSTAG + 1
        CALL CLOSE(IDCB,IERR)
C
C GET CONTROL BLOCK INFORMATION
C
          IERR=WFINT(NROW,ICOLS,RMAX,RMIN,LU)
          IF(IERR .LT. 0)GOTO 9999
        IPIXL = 2
        ICOLS=ICOLS-2
        JPIXL =ICOLS - 1
C
```

```
C
C     INITIALIZE FILTER TO MID LINE-COL AVG
C
      NMID=NROW/2
      CNST=0.0
      IERR=READL(NMID,0,511,F1)
      IF(IERR .LT. 0) GO TO 9999
701   DO 110 I=1,ICOLS
110   CNST=CNST+F1(I)
602   CNST=CNST/FLOAT(ICOLS)
C
      DO 13 I=1,524
      F3(I)=CNST
      F2(I)=CNST
13    F1(I)=CNST
C
C     CALCULATE FINAL VALUE FOR EACH STAGE
C
      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)
11    SUM(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,16HFILTER UNSTABLE ,8)
10    SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)
C
C     CALCULATE INITIAL CONDITIONS FOR EACH STAGE
C
      SUM(1,2)=CNST
      DO 12 NSTG=2,N
12    SUM(NSTG,2)=SUM(NSTG,1)*SUM(NSTG-1,2)
C
C     INITIALIZE FILTER
C
      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)
14  CONTINUE
    RMX=-1.0E38
    RMI= 1.0E38
C
C     FILTER REVERSE
C
      IERR=READL(8,IPIXL,JPIXL,F3)
      IF(IERR .LT. 0) GO TO 9999
      IERR=READL(7,IPIXL,JPIXL,F2)
      IF(IERR .LT. 0) GO TO 9999
      IERR=READL(6,IPIXL,JPIXL,F1)
      IF(IERR .LT. 0) GO TO 9999
```

```
C
          LNCK = 1
          DO 300 NRO=-6,NROW - 1,3
          CALL FILTR(2,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 7) GO TO 301
        LINE = IABS(NRO)
          CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,2,LU,RMX,RMI)
301        LNCK =LNCK +1
        LINE=IABS(NRO+1)
        IF(LINE .GT. NROW-1) GO TO 300
          IERR=READL(LINE,IPIXL,JPIXL,F3)
          IF(IERR .LT. 0) GO TO 9999
          CALL FILTR(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 7) GO TO 302
          CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,2,LU,RMX,RMI)
302        LNCK =LNCK +1
        LINE=IABS(NRO+2)
        IF(LINE .GT. NROW-1) GO TO 300
          IERR=READL(LINE,IPIXL,JPIXL,F2)
          IF(IERR .LT. 0) GO TO 9999
          CALL FILTR(2,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 7) GO TO 303
        IF(LINE .GT. NROW-1) GO TO 300
          CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,2,LU,RMX,RMI)
303        LNCK =LNCK +1
        LINE=IABS(NRO+3)
        IF(LINE .GT. NROW-1) GO TO 300
          IERR=READL(LINE,IPIXL,JPIXL,F1)
          IF(IERR .LT. 0) GO TO 9999
300       CONTINUE
C
C REINITIALIZE FILTER
C
        CONST=(RMX-RMI)/2.
        DO 15 II=1,524
        F1(II) = CONST
        F2(II) = CONST
        F3(II) = CONST
15      CONTINUE
C
C         FILTER FORWARD
C
        RMX=-0.1E38
        RMI= 0.1E38
        LINE =NROW-9
          IERR=READL(LINE,IPIXL,JPIXL,F3(12))
          IF(IERR .LT. 0) GO TO 9999
        LINE=LINE+1
          IERR=READL(LINE,IPIXL,JPIXL,F2(12))
          IF(IERR .LT. 0) GO TO 9999
        LINE=LINE+1
          IERR=READL(LINE,IPIXL,JPIXL,F1(12))
          IF(IERR .LT. 0) GO TO 9999
```

```
C
          LNCK =-6
          DO 400 NRO= -6,NROW - 1,3
          CALL FILTR(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)
401     LNCK=LNCK+1
        LINE=(NROW-1)-IABS(NRO+1)
          IERR=READL(LINE,IPIXL,JPIXL,F3(12))
          IF(IERR .LT. 0) GO TO 9999
          CALL FILTR(1,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 0) GO TO 402
          CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,1,LU,RMX,RMI)
402        LNCK =LNCK +1
        LINE=(NROW-1)-IABS(NRO+2)
        IF(LINE .LT. 0) GO TO 400
          IERR=READL(LINE,IPIXL,JPIXL,F2(12))
          IF(IERR .LT. 0) GO TO 9999
          CALL FILTR(1,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 0) GO TO 403
          CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,1,LU,RMX,RMI)
403        LNCK =LNCK +1
        LINE=(NROW-1)-IABS(NRO+3)
        IF(LINE .LT.0) GO TO 400
          IERR=READL(LINE,IPIXL,JPIXL,F1(12))
          IF(IERR .LT. 0) GO TO 9999
400       CONTINUE
C
51        CONTINUE
        RMAX=RMX
        RMIN=RMI
        CALL CLSWF(NROW,ICOLS,RMAX,RMIN)
        CALL PRTN(IRTN)
          CALL EXEC(6)
9999      CALL EXEC(2,LU,16HREAD FILE ERROR ,8)
          END
          SUBROUTINE FILTR(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)
C
        EQUIVALENCE (IBUF,A),(IBUF(41),B)
C     IFLAG =1 FOR FORWARD FILTERING, = 2 FOR REVERSE
C
C REVERSE FILTERING
C
         IF(IFLAG .EQ. 1) GO TO 200
         DO 20 I=1,11
         L =ICOLS+12 - I
         J = ICOLS-12 + I
         F1(L) = F1(J)
         F2(L) = F2(J)
20         F3(L) = F3(J)
C
```

```
          DO 10 M = ICOLS+9,1,-1
           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 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)
30          F3(L) = F3(J)
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
```

```
C
C    COMMON BLOCK SUBPROGRAM
C
       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
       IF(ITEMP .GT. 377B) ITEMP=377B
120    IX1(I) = ITEMP
         GO TO 200
100      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)
200    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$
```

&HFLTR T=00004 IS ON CR00022 USING 00036 BLKS R=0289

```
FTN4,L
      PROGRAM HFLTR
C
C    WRITTEN BY E. E. SHERROD
C
C      PROGRAM DOES HOMOMORPHIC FILTERING USING SPATIAL DOMAIN
C              RECURSIVE DIGITAL FILTERS
C
     COMMON /IBLK/IBUF(80)
     DIMENSION IF1(2),IF2(523),R1(523)
       DIMENSION A(3,3,2),B(3,3,2),ILU(5),SUM(3,2)
       DIMENSION F1(523),F2(523),F3(523)
       DIMENSION G1(1),G2(1),G3(1),IX1(3)
       DIMENSION X1(523),X2(523),X3(523)
     DIMENSION IDCB(144),NAME(3),IRTN(5)
     INTEGER READL,RITEL,WFINT
       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),ICO
    1(R1(2),RMAXX),(R1(3),RMINN)
     DATA NAME/2HCO,2HEF,2HFS/
C
      CALL RMPAR(ILU)
      LU=ILU(1)
     IF(LU .EQ. 0) LU=1
     IPIXL = ILU(2)
     IF(IPIXL .EQ. 0) IPIXL =0
     JPIXL = ILU(3)
     IF(JPIXL .EQ. 0) JPIXL = 511
C
C GET FILTER COEFF'S
     CALL OPEN(IDCB,IERR,NAME)
     IF(IERR .LT. 0) GO TO 9999
     CALL READF(IDCB,IERR,IBUF,80,IERR)
     IF(IERR .LT. 0) GO TO 9999
     NSTAG = IBUF(40)
     N = NSTAG + 1
     CALL CLOSE(IDCB,IERR)
C
C GET CONTROL BLOCK INFORMATION
       IERR=WFINT(NROW,ICOLS,RMAX,RMIN,LU)
       IF(IERR .LT. 0)GOTO 9999
     IPIXL=2
       ICOLS=ICOLS-2
     JPIXL = ICOLS -1
C
C      INITIALIZE FILTER TO MID LINE-COL AVG
       NMID=NROW/2
       CNST=0.0
       IERR=READL(NMID,IPIXL,JPIXL,F1)
       IF(IERR .LT. 0) GO TO 9999
     CALL BIAS(F1,RMIN,ICOLS)
701    DO 110 I=1,ICOLS
110    CNST=CNST+AMAX0(F1(I),1)
602    CNST=(CNST/FLOAT(ICOLS))
     CNST = ALOG(CNST)
```

```
C
      DO 9 I=1,523
      F1(I) = CNST
      F2(I) = CNST
      F3(I) = CNST
9     CONTINUE
C
C       CALCULATE FINAL VALUE FOR EACH STAGE
        DO 10 NSTG=2,N
        SUM(NSTG,1)=0.0
        SUM(NSTG,2)=0.0
        DO 11 I=1,3
        DO 11 J=1,3
        SUM(NSTG,1)=SUM(NSTG,1)+A(I,J,NSTG-1)
11      SUM(NSTG,2)=SUM(NSTG,2)+B(I,J,NSTG-1)
        DEL=ABS(SUM(NSTG,2))
        IF(DEL.LT.1.0E-30)CALL EXEC(2,LU,16HFILTER UNSTABLE ,8)
10      SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)
C
C       CALCULATE INITIAL CONDITIONS FOR EACH STAGE
        SUM(1,2)=CNST
        DO 12 NSTG=2,N
12      SUM(NSTG,2)=SUM(NSTG,1)*SUM(NSTG-1,2)
C
C       INITIALIZE FILTER
        DO 14 I=1,523
        X3(I)=(SUM(2,2))
        X2(I)=(SUM(2,2))
        X1(I)=(SUM(2,2))
      IF (NSTAG .EQ. 1) GO TO 14
      G3(I) =( SUM(3,2))
        G2(I)=(SUM(3,2))
        G1(I)=(SUM(3,2))
14    CONTINUE
      RMX=-1.0E38
      RMI= 1.0E38
C
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
C
```

```
          LNCK = 1
          DO 300 NRO=-6,NROW - 1,3
        CALL BIAS(F1,RMIN,ICOLS)
          CALL HFILT(2,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 7) GO TO 301
        LINE = IABS(NRO)
          CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,2,LU,RMX,RMI,SCL)
301        LNCK =LNCK +1
        LINE=IABS(NRO+1)
        IF(LINE .GT. NROW-1) GO TO 300
          IERR=READL(LINE,IPIXL,JPIXL,F3)
          IF(IERR .LT. 0) GO TO 9999
        CALL BIAS(F3,RMIN,ICOLS)
          CALL HFILT(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 7) GO TO 302
          CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,2,LU,RMX,RMI,SCL)
302        LNCK =LNCK +1
        LINE=IABS(NRO+2)
        IF(LINE .GT. NROW-1) GO TO 300
          IERR=READL(LINE,IPIXL,JPIXL,F2)
          IF(IERR .LT. 0) GO TO 9999
        CALL BIAS(F2,RMIN,ICOLS)
          CALL HFILT(2,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
          IF(LNCK .LT. 7) GO TO 303
        IF(LINE .GT. NROW-1) GO TO 300
          CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,2,LU,RMX,RMI,SCL)
303        LNCK =LNCK +1
        LINE=IABS(NRO+3)
        IF(LINE .GT. NROW-1) GO TO 300
          IERR=READL(LINE,IPIXL,JPIXL,F1)
          IF(IERR .LT. 0) GO TO 9999
300       CONTINUE
C
C  REINITIALIZE FILTER
        CONST = (RMX-RMI)/2.
        DO 15 J=1,523
        F1(J) = CONST
        F2(J) = CONST
        F3(J) = CONST
15      CONTINUE
C
C         FILTER FORWARD
C
        CALL EXEC(2,LU,16HFORWARD FILTERIN,8)
C
C SCALE FOR LN(32766)
        SCL = 10.397147 /(RMX)
        RMI=0.1E38
        RMX=-0.1E38
        JPIXL=JPIXL-1
        LINE =NROW-9
          IERR=READL(LINE,IPIXL,JPIXL,F3(12))
          IF(IERR .LT. 0) GO TO 9999
        LINE=LINE+1
          IERR=READL(LINE,IPIXL,JPIXL,F2(12))
          IF(IERR .LT. 0) GO TO 9999
        LINE=LINE+1
          IERR=READL(LINE,IPIXL,JPIXL,F1(12))
          IF(IERR .LT. 0) GO TO 9999
```

```
C
          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)
          IERR=READL(LINE,IPIXL,JPIXL,F3(12))
          IF(IERR .LT. 0) GO TO 9999
          CALL HFILT(1,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
          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
          IERR=READL(LINE,IPIXL,JPIXL,F2(12))
          IF(IERR .LT. 0) GO TO 9999
          CALL HFILT(1,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
          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
          IERR=READL(LINE,IPIXL,JPIXL,F1(12))
          IF(IERR .LT. 0) GO TO 9999
400       CONTINUE
C
51        CONTINUE
        CALL EXEC(2,LU,10HCOMPLETED ,5)
C
        CALL CLSWF(NROW,ICOLS,RMX,RMI)
C
        RMAX = RMX
        RMIN = RMI
        CALL PRTN(IRTN)
          CALL EXEC(6)
9999      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)
C
        EQUIVALENCE (IBUF,A),(IBUF(41),B)
C     IFLAG =1 FOR FORWARD FILTERING, = 2 FOR REVERSE
C
C REVERSE FILTERING
C
         IF(IFLAG .EQ. 1) GO TO 200
         DO 20 I=1,11
         L =ICOLS+12 - I
         J = ICOLS-12 + I
         F1(L) = F1(J)
         F2(L) = F2(J)
20         F3(L) = F3(J)
C
```

```
          DO 10 M = ICOLS+9,1,-1
           J = M + 1
           K = M +2
           X1(M) = A(1) * ALOG(F1(M))
     1          + A(2) * ALOG(F1(J))-B(2)*X1(J)
     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)
30          F3(L) = F3(J)
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
```

```
C
C   COMMON BLOCK SUBPROGRAM
C
        BLOCK DATA IBLK
        COMMON/IBLK/IBUF(80)
        END
        SUBROUTINE RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,IFLAG,LU,RMX,R
       1SCL)
        DIMENSION X1(1),G1(1),XX1(523)
        INTEGER RITEL
C
C IFLAG =1 FOR FORWARD  =2 FOR REVERSE
C REV
          IF(NSTAG .EQ. 2) GO TO 12
        IF(IFLAG .EQ. 1) GO TO 11
        DO 10 M=1,JPIXL-IPIXL+1
        IF(X1(M) .GT. RMX) RMX=X1(M)
        IF(X1(M) .LT. RMI) RMI=X1(M)
10      CONTINUE
        IERR=RITEL(LINE,IPIXL,JPIXL,X1)
        IF(IERR .LT. 0) GO TO 9999
        GO TO 12
C
11      CONTINUE
C FORWARD
        DO 20 M=12,JPIXL-IPIXL+12
        X=SCL*(X1(M))
        IF(X .GT. 10.397147) X = 10.397177
        XX1(M) = EXP(X)
        IF(XX1(M) .GT. RMX) RMX=XX1(M)
        IF(XX1(M) .LT. RMI) RMI=XX1(M)
20      CONTINUE
        IERR= RITEL(LINE,IPIXL,JPIXL,XX1(12))
        IF(IERR .LT. 0) GO TO 9999
12      CONTINUE
          RETURN
9999    CALL EXEC(2,LU,16HWRITE FILE ERROR,8)
          END
        SUBROUTINE BIAS(F1,RMIN,ICOLS)
        DIMENSION F1(1)
        DO 10 I=1,ICOLS + 11
        F1(I) = F1(I) - RMIN +1.0
        IF(F1(I) .LT. 1.) F1(I) = 1.0
10      CONTINUE
        RETURN
        END
$
```

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

FTN4
      PROGRAM SHOW
C
      DIMENSION RDATA(512),IDATA(512),LU(5)
C
      INTEGER READL
      EQUIVALENCE (RDATA,LU(2)),(LU(2),ILINE),(LU(3),IPIXL),
     1 (RDATA(2),RMAX),(RDATA(3),RMIN)
C
C GET INPUT PARAMETERS
C
      CALL RMPAR(LU)
C
C  GET SCALE
C
      WRITE(LU,1)
1     FORMAT("INPUT RANGE?_")
      READ(LU,*)RL,RH
C
C  READ WORK FILE HEADER
C
      IERR = READL(-1,0,511,RDATA)
      IF (IERR .LT. 0) GO TO 999
      NLINE = ILINE
      NPIXL = IPIXL
      PMAX = RMAX
      PMIN = RMIN
      DO 100 I=0,NLINE-1
      IF (READL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
      DO 90 J =1,NPIXL
      IDATA(J) = RL +((RH-RL)/(PMAX-PMIN))*(RDATA(J)-PMIN)
      IF (IDATA(J) .GT. 255) IDATA(J) = 255
      IF (IDATA(J) .LT. 0) IDATA(J) = 0
90    CONTINUE
C
      CALL WLINE(I,0,511,IDATA)
100   CONTINUE
      CALL CLSWF(NLINE,NPIXL,PMAX,PMIN)
      CALL EXEC(6)
999   WRITE(LU,2) IERR
2     FORMAT("FILE ERROR",I7)
      END
$
```

&FIRO T=00004 IS ON CR00022 USING 00003 BLKS R=0023

```
FTN4,L
      PROGRAM FIRO
      DIMENSION ILU(5),IBUF(80),A(3,3,2),H(5,5),NAME(3),IDCB(144)
      DIMENSION NAME1(3),NAME2(3)
      EQUIVALENCE (IBUF(1),A(1,1,1))
      DATA H/25*0./
      DATA IBUF/80*0/
      DATA NAME/2HCO,2HEF,2HFS/
      DATA NAME1/2HDP,2HLA,2HM /
      DATA NAME2/2HPL.2HOT,2HV /
C
C GET LU
      CALL RMPAR(ILU)
      LU=ILU
      WRITE(LU,10)
10    FORMAT(" ENTER NUMBER OF STAGES _")
      READ(LU,*) NSTG
      IBUF(40)=NSTG
C
      WRITE(LU,11)
11    FORMAT(" ENTER ALPHA VALUE _")
      READ(LU,*) ALPHA
C
      H(1,1)=1.0
      DO 100 I=1,3
      DO 100 J=1,3
      CALL WINDO(ALPHA,I,J,WIN)
      A(I,J,NSTG)=WIN*H(I,J)
100   CONTINUE
      CALL PURGE(IDCB,IERR,NAME,2HES)
      IF(IERR .LT. 0) WRITE(LU,999) IERR
      CALL CREAT(IDCB,IERR,NAME,2,3,2HES)
      IF(IERR .LT. 0) WRITE(LU,999) IERR
      CALL WRITF(IDCB,IERR,IBUF,80)
      CALL CLOSE(IDCB,IERR)
C
C SCHEDULE DISPLAY
      CALL EXEC(23,NAME1,LU,NSTG,0,0,0,IBUF,80)
C
      WRITE(LU,40)
40    FORMAT(//" ENTER DISPLAY DEVICE "//"  1. TV"/"  2. HP2648A")
      READ(LU,*) IDEV
      IF(IDEV .EQ. 2) GO TO 41
      CALL EXEC(23,NAME2)
      GO TO 42
41    CONTINUE
      CALL HP48A(LU)
42    CONTINUE
999   FORMAT(" FILE ERROR ")
      STOP
      END
```

```
      SUBROUTINE HP48A(LU)
      DIMENSION IB(14),IA(4)
      INTEGER 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    FORMAT  ("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 DVECT(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
C  GRAPHICS OFF=0; GRAPHICS  ON NOT=0
C
        IF(IFLAG.EQ.0) GO TO 100
C
C  GRAPHIC ON
C
        WRITE(LU,10) IESC
10      FORMAT(1R2,"*dC")
        WRITE(LU,12) IESC
12      FORMAT(1R2,"*dF")
        WRITE(LU,14) IESC
14      FORMAT(1R2,"*dA")
C
        GO TO 200
C
C  GRAPHICS OFF
C
100     WRITE(LU,30) IESC
30      FORMAT(1R2,"*dd")
        WRITE(LU,40) IESC
40      FORMAT(1R2,"*dE")
200     RETURN
        END
```

```
      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 (IX0,IY0) DEFINES THE
C         THE ORIGIN.
C
        IX0=0
        IY0=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 + IY0
        JY2 = Y2 + IY0
      WRITE(LU,10) JX1,JY1,JX2,JY2
10    FORMAT("pa",1I3,1H,,1I3,1H,,1I3,1H,,1I3,"Z")
      RETURN
      END
      END$
$
$
```

&WINDO T=00004 IS ON CR00022 USING 00003 BLKS R=0012

```
FTN4
      SUBROUTINE WINDO(ALPHA,N,M,WIN)
      XN=SQRT(M**2 + N**2)
      BETA=ALPHA*SQRT(1.-XN)
      CALL BESIO(ALPHA,BIAA)
      CALL BESIO(BETA,BIBB)
      BETA1=ALPHA*SQRT(2)
      CALL BESIO(BETA1,BIB)
      ZMIN=BIB/BIAA
      WIN=(BIBB/BIAA-ZMIN)/(1.0-ZMIN)
      RETURN
      END
$
```

&BESIO T=00004 IS ON CR00022 USING 00002 BLKS R=0015

```
FTN4
      SUBROUTINE BESIO(X,RIO)
      RIO=ABS(X)
      IF(RIO-3.75) 1,1,2
1     Z=X*X*7.111111E-2
      RIO=(((((4.5813E-3*Z+3.60768E-2)*Z+2.659732E-1)*Z+1.206749E0
     1089942E0)*Z+3.515623E0)*Z+1.
      RETURN
2     Z=3.75/RIO
      RIO=EXP(RIO)/SQRT(RIO)*((((((((3.92377E-3*Z-1.647633E-2)*Z+2
     17E-2)*Z-2.057706E-2)*Z+9.16281E-3)*Z-1.57565E-3)*Z+2.25319E-
     2+1.328592E-2)*Z+3.989423E-1)
      RETURN
      END
      END$
```

&BESJ T=00004 IS ON CR00022 USING 00014 BLKS R=0129

```
C
C     ..........................................................
C
C        SUBROUTINE BESJ
C
C        PURPOSE
C           COMPUTE THE J BESSEL FUNCTION FOR A GIVEN ARHUMENT AND
C
C        USAGE
C           CALL BESJ(X,N,BJ,D,IER)
C
C        DESCRIPTION OF PARAMETERS
C           X  -THE ARGUMENT OF THE J BESSEL FUNCTION DESIRED
C           N  -THE ORDER OF THE J BESSEL FUNCTION DESIRED
C           BJ -THE RESULTANT J BESSEL FUNCTION
C           D  -REQUIRED ACCURACY
C           IER-RESULTANT ERROR CODE WHERE,
C               IER=0  NO ERROR
C               IER=1  N IS NEGATIVE
C               IER=2  X IS NEGATIVE OR ZERO
C               IER=3  REQUIRED ACCURACY NOT OBTAINED
C               IER=4  RANGE OF N COMPARED TO X NOT CORRECT (SEE R
C
C        REMARKS
C           N MUST BE GREATER THAN OR EQUAL TO ZERO, BUT IT MUST B
C           LESS THAN
C               20+10*X-X** 2/3   FOR X LESS THAN OR EQUAL TO 15
C               90+X/2           FOR X GREATER THAN 15
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           RECURRENCE RELATION TECHNIQUE DESCRIBED BY H. GOLDSTEI
C           R.M. THALER,'RECURRENCE TECHNIQUES FOR THE CALCULATION
C           BESSEL FUNCTIONS',M.T.A.C.,V.13,PP.102-108 AND I.A. ST
C           AND M. ABRAMOWITZ,'GENERATION OF BESSEL FUNCTIONS ON H
C           SPEED COMPUTERS',M.T.A.C.,V.11,1957,PP.255-257
C
C     ..........................................................
C
      SUBROUTINE BESJ(X,N,BJ,D,IER)
C
      BJ=.0
      IF(N)10,20,20
   10 IER=1
      RETURN
   20 IF(X)30,30,31
   30 IER=2
      RETURN
   31 IF(X-15.)32,32,34
   32 NTEST=20.+10.*X-(X**(2/3))
      GO TO 36
   34 NTEST=90.+X/2.
   36 IF(N-NTEST)40,38,38
   38 IER=4
      RETURN
```

```
40    IER=0
      N1=N+1
      BPREV=0.
C
C     COMPUTE STARTING VALUE OF M
C
      IF(X-5.)50,60,60
50    MA=X+6.
      GO TO 70
60    MA=1.4*X+60./X
70    MB=N+IFIX(X)/4+2
      MZERO=MAX0(MA,MB)
C
C     SET UPPER LIMIT OF M
C
      MMAX=NTEST
100   DO 190 M=MZERO,MMAX,3
C
C     SET F(M),F(M-1)
C
      FM1=1.0E-28
      FM=.0
      ALPHA=.0
      IF(M-(M/2)*2)120,110,120
110   JT=-1.
      GO TO 130
120   JT=1.
130   M2=M-2
      DO 160 K=1,M2
      MK=M-K
      BMK=2.*FLOAT(MK)*FM1/X-FM
      FM=FM1
      FM1=BMK
      IF(MK-N-1.)150,140,150
140   BJ=BMK
150   JT=-JT
      S=1+JT
160   ALPHA=ALPHA+BMK*S
      BMK=2.*FM1/X-FM
      IF(N)180,170,180
170   BJ=BMK
180   ALPHA=ALPHA+BMK
      BJ=BJ/ALPHA
      IF(ABS(BJ-BPREV)-ABS(D*BJ))200,200,190
190   BPREV=BJ
      IER=3
200   RETURN
      END
$
```

&BLDWF T=00004 IS ON CR00022 USING 00022 BLKS R=4113

```
FTN4
      PROGRAM BLDWF
C
C
C  THIS PROGRAM 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
C
C
      DIMENSION IDCB1(272),IDCB2(1040),IDCB3(528),IMAGE(6),LU(5)
      DIMENSION IRTN(5),JNAME(3),IBUF(15),RDATA(512),IDATA(512),
     1 ISIZE(2)
C
      EQUIVALENCE (ILINE,IRTN(4)),(IPIXL,IRTN(5)),(ILINE,RDATA),
     1(RDATA(2),RMAX),(RDATA(3),RMIN),(IBUF(12),ILOC),(IBUF(13),JN
      EQUIVALENCE (IBUF(7),NLINE),(IBUF(8),NPIXL),(IBUF(9),IPMIN),
     1 (IBUF(10),IPMAX),(ISIZE(2),ISIZ2)
C
C
C
C
C  GET INPUT PARAMETERS
C
      CALL RMPAR(LU)
      IF (LU .LE. 0) LU = 1
C
C REUSE WORK FILE
C
      WRITE(LU,7)
7     FORMAT(//,"DO YOU WANT TO REUSE THE CURRENT WORK FILE? Y OR
      READ(LU,2) IANS
      IF(IANS .EQ. 1HY )GO TO 200
C  GET IMAGE NAME FROM USER
C
      WRITE(LU,1)
1     FORMAT("ENTER IMAGE NAME (12 CHARACTER)?_")
      READ(LU,2) IMAGE
2     FORMAT(6A2)
C
C  CHECK IF WORK FILE WANTED
C
      IF (ICMPW(IMAGE,12H            ,6) .EQ. 0) GO TO 140
C
C  OPEN DIRECTORY FILE
C
90    CALL OPEN(IDCB1,IERR,6HIMDIRC,1,2HIM,23,272)
      IF (IERR .LT. 0) GO TO 9999
C
```

```
C  FIND IMAGE
C
100   CALL READF(IDCB1,IERR,IBUF,15,LEN)
      IF (IERR .LT. 0) GO TO 9999
      IF (LEN .EQ. -1) GO TO 9990
C
      IF (ICMPW(IMAGE,IBUF,6) .NE. 0) GO TO 100
C
C IMAGE FOUND
C
C
      IF (ILOC .NE. 1) GO TO 9980
C
C  IMAGE IS ON DISC
C
C CREATE WORK FILE
C
      CALL OPEN(IDCB2,IERR,6HWF0000)
      IF (IERR .EQ. -6) GO TO 110
      IF (IERR .LT. 0) GO TO 9999
C
C  ASK IF USER WANTS TO SAVE WORK FILE
C
      WRITE(LU,6)
6     FORMAT(" DO YOU WANT TO SAVE IMAGE IN CURRENT WORK FILE?_")
      READ(LU,2) IANS
      IF (IANS .EQ. 2HNO) GO TO 110
C
C SCHEDULE BUILD IMAGE PROGRAM
C
      CALL CLOSE(IDCB2)
      CALL EXEC(23,6HBLDIM ,LU)
C
110   CALL PURGE(IDCB2,IERR,6HWF0000)
      IF (NPIXL .LT. 3) NPIXL = 3
      ISIZE = (2.0*FLOAT(NLINE+1)*FLOAT(NPIXL)+127.)/128.
      ISIZ2 = 2*NPIXL
      CALL CREAT(IDCB2,IERR,6HWF0000,ISIZE,2,0,0,1040)
      IF (IERR .LT. 0) GO TO 9999
C
C OPEN IMAGE DATA FILE
C
      CALL OPEN(IDCB3,IERR,JNAME,1,2HIM,23,528)
      IF (IERR .LT. 0) GO TO 9999
C
C  COPY DATA AND CONVERT TO REAL
C
C  POSITION TO RECORD # 2
C
      CALL WRITF(IDCB2,IERR,RDATA,1)
C
      DO 120 I=1,NLINE
      CALL READF(IDCB3,IERR,IDATA,512,LEN)
      IF (IERR .LT. 0) GO TO 9999
C
      DO 115 J=1,NPIXL
115   RDATA(J) = IDATA(J)
C
      CALL WRITF(IDCB2.IERR.RDATA)
```

```
      IF (IERR .LT. 0) GO TO 9999
120   CONTINUE
C
      RPMAX = IPMAX
      RPMIN = IPMIN
C
C  CLOSE ALL IMAGE FILES
C
130   CALL CLOSE(IDCB1)
      CALL CLOSE(IDCB3)
C
C  WRITE INFO IN WORK FILE RECORD 1
C
      ILINE = NLINE
      IPIXL = NPIXL
      RMAX = RPMAX
      RMIN = RPMIN
      CALL WRITF(IDCB2,IERR,RDATA,6,1)
      IF (IERR .LT. 0) GO TO 9999
C
      CALL CLOSE(IDCB2)
C
140   IRTN = 0
200   CALL PRTN(IRTN)
      CALL EXEC(6)
C
C  ERRORS
C
C
C  IMAGE NOT ON DISC
C
9980  WRITE(LU,4)
4     FORMAT(" IMAGE NOT ON DISC!")
      IRTN = -100
      GO TO 200
C
C  IMAGE NOT FOUND
C
9990  WRITE(LU,3)
3     FORMAT(" IMAGE NOT FOUND!")
      IRTN = -101
      GO TO 200
C
C  FILE ERROR
C
9999  WRITE(LU,5) IERR
5     FORMAT("FILE ERROR =",I6)
201   IF(IERR.EQ.-8) CALL CLOSE(IDCB1,IERR)
      IRTN = -103
      GO TO 200
      END
$
```

```
FTN4
      INTEGER FUNCTION SCROL(IDCB,IDIRC,NLINE,IFRST,ILAST,RMAX,RMI
C
C THIS SUBROUTINE IS USED TO SCROLL AN IMAGE ON THE GMR-27
C
C     IDCB = OPENED DATA CONTROL BLOCK FOR THE IMAGE
C     IDIRC = DIRECTION TO SCROLL (-N= BACK N LINES N= FORWARD N LI
C     NLINE = # LINES IN IMAGE
C     IFRST = LOWEST IMAGE LINE DISPLAYED
C     ILAST = HIGHEST IMAGE LINE DISPLAYED
C
C
      DIMENSION IDCB(144),IDATA(512)
C
      INTEGER SCROL
C
      DATA IUP,IDOWN/34060B,34040B/
C
C  CHECK IF NO WORK NECESSARY
C
      IF(IDIRC .EQ. 0) RETURN
C
      IF (IDIRC .GT. 0) GO TO 200
C
C  SCROLL IMAGE UP
C
      DO 100 I=-1,IDIRC,-1
      IF (IFRST .LE. 0) RETURN
      CALL READF(IDCB,SCROL,IDATA,512,LEN,IFRST)
C
      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
C
      IF (SCROL .LT. 0) RETURN
      CALL DRIVR(2,IUP,1)
      CALL WLINE(0,0,LEN-1,IDATA)
      IFRST = IFRST-1
100   ILAST = ILAST-1
      RETURN
C
C  SCROLL IMAGE DOWN
C
200   DO 210 I=1,IDIRC
      IF (ILAST .GE. NLINE-1) RETURN
      CALL READF(IDCB,SCROL,IDATA,512,LEN,ILAST+1)
C
      DO 220 J=1,LEN
      IDATA(J) = (255./(RMAX-RMIN))*(IDATA(J)-RMIN)
      IF (IDATA(J) .LT. 0) IDATA(J) = 0
      IF (IDATA(J) .GT. 255) IDATA(J) = 255
220   CONTINUE
C
      IF (SCROL .LT. 0) RETURN
      CALL DRIVR(2,IDOWN,1)
      CALL WLINE(255,0,LEN-1,IDATA)
      ILAST = ILAST+1
210   IFRST = IFRST+1
C
      RETURN
      END
```

```
$WLINE T=00004 IS ON CR00022 USING 00005 BLKS R=0036

FTN4,L
      SUBROUTINE WLINE(LINE,IPIX,JPIX,IDATA)
C
C  THIS SUBROUTINE WRITES A DESIGNATED LINE TO THE GMR-27
C
C          LINE = LINE NUMBER
C          IPIX = STARTING PIXEL
C          JPIX = ENDING PIXEL
C          IDATA = BUFFER CONTAINING IMAGE DATA FOR LINE
C
C
      DIMENSION IDATA(512),INIT(6)
C
      EQUIVALENCE (LLA,INIT(2)),(LEA,INIT(3)),(LEB,INIT(4))
C
      DATA INIT/100377B,64000B,44000B,50000B,24041B,26002B/
C
C  COMPUTE DIRECTION
C
      IDIRC = 1
      IF (IPIX .GT. JPIX) IDIRC = -1
C
C  SET UP TO WRITE LINE
C
      LLA = 64000B + IAND(LINE,377B)
      LEA = 44000B + IAND(IPIX,777B)
      LEB = 50000B + IDIRC + 512
      CALL DRIVR(2,INIT,6)
C
C  WRITE LINE
C
      NUM = IDIRC*(JPIX-IPIX)+1
      CALL DRIVR(2,IDATA,NUM)
C
      RETURN
      END
$
```

&DRIVR T=00004 IS ON CR00022 USING 00012 BLKS R=0241

```
ASMB,R,L,C
      NAM DRIVR,6
      ENT DRIVR
      EXT .ENTR,$LIBR,$LIBX
*
*
OPCOD BSS 1
BUFR  BSS 1
LEN   BSS 1
*
DRIVR NOP              ENTRY
      JSB .ENTR        GET
      DEF OPCOD        PARAMETERS.
      LDA LEN,I        GET # WORDS
      CMA,INA          NEGATE
      STA CNT          & SAVE.
      SSA,RSS          IF NOT NEGATIVE
      JMP EXIT         EXIT
*
      JSB $LIBR        TURN OFF
      NOP              INTERRUPTS.
      LDA OPCOD,I      CHECK REQUEST
      SLA,ELA          IF READ
      JMP D.2          GO PROCESS
*
*  WRITE REQUEST
*
      SSA,RSS          IF DMA NOT REQUIRED
      JMP D.1          GO DO PROGRAMMED I O
*
*  DMA OUTPUT
*
      LDA CW1          GET CONTROL WORD 1
      OTA DMA2         USE CHANNEL 2
      CLC 3B           PREPARE TO SEND ADDRESS
      LDA BUFR
      OTA 3B
      STC 3B           PREPARE TO SEND COUNT
      LDA CNT
      OTA 3B
      LDA BUFR,I
      OTA SC
      STC SC,C         START DEVICE
      STC DMA2,C       START DMA
      SFS DMA2
      JMP *-1
      CLF DMA2
      JMP EXIT+1
*
*
```

```
D.1   LDA BUFR,I    GET DATA WORD
      OTA SC        OUTPUT IT.
      STC SC,C      TURN ON DEVICE
      SFS SC        WAIT 'TIL
      JMP *-1       DONE
      ISZ BUFR      BUMP BUFFER ADDRESS
      ISZ CNT       LAST WORD?
      JMP D.1       NO GO BACK.
      JMP EXIT      GO EXIT
*
*  READ ENTRY
*
D.2   SSA           SKIP IF SPECIAL
      JMP D.3       MODE
      LDA SPD8      SET UP
      OTA SC
      STC SC,C      FOR
      SFS SC
      JMP *-1       READ.
D.3   LDA RDPD      GET READ DATA CODE
      OTA SC
      STC SC,C      START DEVICE
      SFS SC        WAIT 'TIL
      JMP *-1
D.4   LDA RDPD
      OTA SC
      STC SC,C
      SFS SC
      JMP *-1
      LIA SC        DONE.  GET WORD.
      STA BUFR,I    STUFF IN BUFFER
      ISZ BUFR      BUMP BUFFER
      ISZ CNT       DONE?
      JMP D.4       NO GO BACK.
*
EXIT  CLC SC        TURN OFF DEVICE
      JSB $LIBX     RESTORE RTE AND
      DEF DRIVR     RETURN
*
*
*
A     EQU 0
*
SC    EQU 22B
RDPD  OCT 160000
SPD8  OCT 120400
CNT   BSS 1
CW1   OCT 120022      *  HAVE TO CHANGE WITH SELECT CODE
DMA2  EQU 7
      END
```

```
&FDIGI T=00004 IS ON CR00022 USING 00018 BLKS R=0132

FTN4,L
      SUBROUTINE  ROTAE(U,V,MN,LU)
      COMPLEX P(10),Q(10),QQ,PP
      DIMENSION U(3,3,2),V(3,3,2)
     COMMON/WORK/AMAG(10),A(3,3),B(3,3)
      WRITE(LU,100)
100   FORMAT("  SELECT FILTER  "/,"  1.  BUTTERWORTH  "/,
     1  "  2.  CHEBYSHEV  "/,"  3.  LINEAR PHASE  "/)
      READ(LU,*) ITYPE
      WRITE(LU,110)
110   FORMAT("  ENTER THE NUMBER OF FILTER STAGES  "/)
      READ(LU,*) NSTG
      WRITE(LU,120)
120   FORMAT("  ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS "/)
      READ(LU,*) WR
      WRITE(LU,140)
140   FORMAT("  ARE ALL ZEROS LOCATED AT INFINITY  "/,
     1  "  1 =  YES  "/,"  2 =  NO  "/)
      READ(LU,*) IFLAG
      WRITE(LU,151)
151   FORMAT("  ENTER RIPPLE FACTOR "/)
      READ(LU,*) ELP
C
C     IF(ITYPE.EQ.1) CALL BUTTER
      IF(ITYPE.EQ.2) CALL  CHEB1(NSTG,WR,P,AMAG,ELP)
C     IF(ITYPE.EQ.3) CALL LINEAR PHASE
C
20    DO 10 J=1,NSTG
30    WRITE(LU,130) J
130   FORMAT("  ENTER ROTATION ANGLE IN NEG. DEGREES FOR STAGE #
     1,I2/)
      READ(LU,*) THETA
C
      PMAG = AMAG(J)
      Q(J) = CMPLX(-1.,0.)
      QQ = Q(J)
      PP = P(J)
      CALL  SROTT(A,B,PMAG,PP,QQ,IFLAG,THETA)
     DO 1111 I=1,3
     DO 1111 K=1,3
     U(I,K,J) = A(I,K)
1111  V(I,K,J) = B(I,K)
      WRITE(LU,40) P(J),AMAG(J)
40    FORMAT(1X,1("  P=",1E15.5,"  +J",1E15.5,/)," PMAG= ",E15.5
10    CONTINUE
      MN = NSTG + 1
     WRITE(1,1112) U
     WRITE(1,1112) V
1112 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 "/)
34    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=-2.0*COS(X)*SIN(X)
        C3=SIN(X)**2
        C7=-2.0*REAL(PP)*COS(X)
        C8=2.0*REAL(PP)*SIN(X)
        C9=CABS(PP)**2
        B(1,1)=C1+C2+C3+C7+C8+C9
        B(1,2)=2.0*(C1-C3+C7+C9)*ADJ
        B(1,3)=(C1-C2+C3+C7-C8+C9)*ADJ**2
        B(2,1)=2.0*(C3-C1+C8+C9)*ADJ
        B(2,2)=4.0*(C9-C1-C3)*ADJ**2
        B(2,3)=2.0*(C3-C1-C8+C9)*ADJ**3
        B(3,1)=(C1-C2+C3-C7+C8+C9)*ADJ**2
        B(3,2)=2.0*(C1-C3-C7+C9)*ADJ**3
        B(3,3)=(C1+C2+C3-C7-C8+C9)*ADJ**4
        IF(IFLAG.EQ.1) GO TO 10
        C4=-2.0*REAL(QQ)*COS(X)
        C5=2.0*REAL(QQ)*SIN(X)
        C6=CABS(QQ)**2
        A(1,1)=C1+C2+C3+C4+C5+C6
        A(1,2)=2.0*(C1-C3+C4+C6)
        A(1,3)=C1-C2+C3+C4-C5+C6
        A(2,1)=2.0*(C3-C1+C5+C6)
        A(2,2)=4.0*(C6-C1-C3)
        A(2,3)=2.0*(C3-C1-C5+C6)
        A(3,1)=C1-C2+C3-C4+C5+C6
        A(3,2)=2.0*(C1-C3-C4+C5+C6)
        A(3,3)=C1+C2+C3-C4-C5+C6
        GO TO 20
10      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
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)
30    CONTINUE
        RETURN
        END
$
$
```

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

FTN4,L
      PROGRAM STABI
C
C THIS PROGRAM EVALUATES THE FILTER STABILITY CHARACTERISTICS
C
C
      COMMON/WORK/WO(130)
C     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))
C
      CALL RMPAR(ILU)
      LU=ILU(1)
      MN=ILU(2) + 1
C
C
C GET FILTER COEFF'S
      CALL EXEC(14,1,IBUF,80)
C
C
      CALL STABT(V,MN,IRTCD,LU)
      IRTN = IRTCD
C
      CALL PRTN(IRTN)
      END
      SUBROUTINE STABT(V,MN,IRTCD,LU)
C     SUBROUTINE CHECKS STABILITY OF SYSTEM EQUATION-
C     Y(M,N)=A*Y(M-1,N)+B*Y(M,N-1)
C
C     C---COEFFICIENT MATRIX OF DENOMINATON OF ZW-TRANSFORM OF SYS
C          IMPULSE FUNCTION
C
      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)
      MDIM=25
      N=2*(MN-1)+1
      M=N**2
      IF(MN.EQ.3) GO TO 5
C
C     PUT COEFFICENTS IN STABILITY ARRAY
C
      DO 6 I=1,3
      DO 6 J=1,3
    6 C(I,J)=V(I,J,1)
      GO TO 13
    5 DO 10 I=1,5
      DO 10 J=1,5
      DO 10 K=1,3
      DO 10 L=1,3
      IK=I-K+1
      JL=J-L+1
      IF((IK .LE. 0) .OR. (IK .GT. 3)) GO TO 10
      IF((JL .LE. 0) .OR. (JL .GT. 3)) GO TO 10
      C(I,J)=C(I,J)+V(IK,JL,1)*V(K,L,2)
   10 CONTINUE
```

```
C
   13 CONTINUE
C     WRITE(LU,11)
   11 FORMAT(20H0 COEFFICIENT MATRIX,/)
C     DO 21 I=1,N
C  21 WRITE(LU,12) (C(I,J),J=1,N),N
   12 FORMAT(1H ,5F15.6)
C
C     FORM A AND B MATRICES
C
      DO 22 I=1,M
      DO 22 J=1,M
      A(I,J)=0.0
      B(I,J)=0.0
   22 S(I,J)=0.0
      NNOW=N-1
      DO 23 J=1,N
      DO 23 I=1,NNOW
      K=I+(J-1)*N
      IF(J.EQ.1) GO TO 24
   24 A(1,K)=-C(I+1,J)
        IF(J.GT.1) A(1,K)=-0.5*C(I+1,J)
        IF(J.GT.1) A(K+1,K)=0.5
        IF(J.EQ.1) A(K+1,K)=1.0
   23 CONTINUE
      DO 25 J=1,NNOW
      DO 25 I=1,N
      K=I+(J-1)*N
      KN=K+N
      IF(I.EQ.1) GO TO 26
   26 B(1,K)=-C(I,J+1)
        IF(I.GT.1) B(1,K)=-0.5*C(I,J+1)
        IF(I.GT.1) B(KN,K)=0.5
        IF(I.EQ.1) B(KN,K)=1.0
   25 CONTINUE
C
C
C     FIND EIGENVALUES OF A AND B
C
      DO 27 I=1,M
      DO 27 J=1,M
   27 S(I,J)=A(I,J)
      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
      WRITE(LU,71)
   71 FORMAT(/,10X,19HEIGEN VALUES OF (A))
C
      TEST=1.0
      IONE=0
C
      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
      IF(ISTAB) GOTO 405
  400 FORMAT(" FILTER IS UNSTABLE!"/)
  401 FORMAT(" FILTER IS STABLE"/)
C
        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,19HEIGEN 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
C
      DO 29 I=1,M
      DO 29 J=1,M
   29 S(I,J)=A(I,J)+B(I,J)

      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
      WRITE(LU,73)
   73 FORMAT(/,10X,21HEIGEN VALUES OF (A+B))
      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
 405  IF(ISTAB) WRITE(LU,400)
      IF(ISTAB) GO TO 404
        WRITE(LU,401)
 404    IRTCD = 0
      GO TO 500
C
C     FIND EIGENVALUES OF A*S
C
      DO 30 I=1,M
      DO 30 J=1,M
   30 S(I,J)=0.0
      DO 31 I=1,N
      DO 31 J=1,N
      K=J+(I-1)*N
      L=I+(J-1)*N
   31 S(K,L)=1.0
      CALL MLTMX(A,S,M,MDIM)
      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
      WRITE(LU,74)
   74 FORMAT(/,10X,21HEIGEN VALUES OF (A*S))
C
      IONE=1
      TEST=0.5
C
      ICNT=0
      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
      IF(ISTAB) ICNT=1
      DO 230 I=1,M
      DO 230 J=1,M
  230 S(I,J)=0.0
      DO 231 I=1,N
      DO 231 J=1,N
      K=J+(I-1)*N
      L=I+(J-1)*N
  231 S(K,L)=1.0
      CALL MLTMX(B,S,M,MDIM)
      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
       WRITE(LU,75)
   75 FORMAT(/,10X,21HEIGEN VALUES OF (B*S))
      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
      IF(ISTAB) ICNT=ICNT+1
      IF(ICNT.EQ.2) WRITE(LU.401)
```

```
C
C     FIND EIGENVALUES OF ABS(A)+ABS(B)
C
      DO 33 I=1,M
      DO 33 J=1,M
   33 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))
C
      TEST=1.0
C
      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
      IF(ISTAB) WRITE(LU,401)
C
C     FIND EIGENVALUES OF A*B
C
      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 PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
        IF(ISTAB) WRITE(LU,401)
        GO TO 501
  500   IF(ISTAB) IRTCD = 1000
  501    RETURN
      END
      SUBROUTINE PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
      LOGICAL ISTAB
      DIMENSION EVR(MDIM),EVI(MDIM),IERR(MDIM)
C
      ISTAB=.FALSE.
      D=1.0E-20
      RMX=0.0
      DO 20 I=1,M
      R=EVR(I)**2+EVI(I)**2
      R=SQRT(R)
      RMX=AMAX1(RMX,R)
      IF(R.LT.D) GO TO 20
      IF(IERR(I).LT.0) WRITE(LU,93) I,IERR(I)
   20 CONTINUE
       WRITE(LU,30) RMX
C
      IF(IONE.EQ.0.AND.RMX.GE.TEST) ISTAB=.TRUE.
      IF(IONE.EQ.1.AND.RMX.LE.TEST) ISTAB=.TRUE.
   10 FORMAT(1H ,E14.7,4X,2H+J,E14.7)
   11 FORMAT(13H ABS(LMDA) = ,E14.7)
   30 FORMAT(19H SPECTRAL RADIUS = ,E14.7/)
   93 FORMAT(//,10X,"IERR(",I2,") = ",I2/)
      RETURN
      END
      SUBROUTINE RNAN(N,M,S,EVR,EVI,IERR)
C     SUBROUTINE WAS WRITTEN TO CALL HSBG AND ATEIG IBM SCIENTIFIC
C          SUBROUTINES TO CALCULATE THE EIGENVALUES OF A REAL MATR
C     M----ORDER OF THE MATRIX S
C     N----SIZE OF FIRST DIMENSION ASSIGNED TO THE ARRAY S IN THE
C          CALLING PROGRAM
```

```
C
      DIMENSION S(25,25),EVR(25),EVI(25)
      COMMON/WORK/IANA(25)
      CALL HSBG(M,S,N)
      CALL ATEIG(M,S,EVR,EVI,IANA,N)
      RETURN
      END
C        SUBROUTINE ATEIG
C        PURPOSE
C           COMPUTE THE EIGENVALUES OF A REAL ALMOST TRIANGULAR MA
C
C        USAGE
C           CALL ATEIG(M,A,RR,RI,IANA,IA)
C
C        DESCRIPTION OF THE PARAMETERS
C           M      ORDER OF THE MATRIX
C           A      THE INPUT MATRIX, M BY M
C           RR     VECTOR CONTAINING THE REAL PARTS OF THE EIGENVA
C                  ON RETURN
C           RI     VECTOR CONTAINING THE IMAGINARY PARTS OF THE EI
C                  VALUES ON RETURN
C           IANA   VECTOR WHOSE DIMENSION MUST BE GREATER THAN OR
C                  TO M, CONTAINING ON RETURN INDICATIONS ABOUT TH
C                  THE EIGENVALUES APPEARED (SEE MATH. DESCRIPTION
C           IA     SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARR
C                  IN THE CALLING PROGRAM WHEN THE MATRIX IS IN DO
C                  SUBSCRIPTED DATA STORAGE MODE.
C                  IA=M WHEN THE MATRIX IS IN SSP VECTOR STORAGE M
C
C        REMARKS
C           THE ORIGINAL MATRIX IS DESTROYED
C           THE DIMENSION OF RR AND RI MUST BE GREATER OR EQUAL TO
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           QR DOUBLE ITERATION
C
C        REFERENCES
C           J.G.F. FRANCIS - THE QR TRANSFORMATION---THE COMPUTER
C           JOURNAL, VOL. 4, NO. 3, OCTOBER 1961, VOL. 4, NO. 4, J
C           1962.  J. H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROB
C           CLARENDON PRESS, OXFORD, 1965.
C
C     ..............................................................
C
      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
C
      E7=1.0E-8
      E6=1.0E-6
      E10=1.0E-10
      DELTA=0.5
      MAXIT=30
```

```
C
C          INITIALIZATION
C
          N=M
       20 N1=N-1
          IN=N1*IA
          NN=IN+N
          IF(N1) 30,1300,30
       30 NP=N+1
C
C          ITERATION COUNTER
C
          IT=0
C
C          ROOTS OF THE 2ND ORDER MAIN SUBMATRIX AT THE PREVIOUS
C          ITERATION
C
          DO 40 I=1,2
          PRR(I)=0.0
       40 PRI(I)=0.0
C
C          LAST TWO SUBDIAGONAL ELEMENTS AT THE PREVIOUS ITERATION
C
          PAN=0.0
          PAN1=0.0
C
C          ORIGIN SHIFT
C
          R=0.0
          S=0.0
C
C          ROOTS OF THE LOWER MAIN 2 BY 2 SUBMATRIX
C
          N2=N1-1
          IN1=IN-IA
          NN1=IN1+N
          N1N=IN+N1
          N1N1=IN1+N1
       60 T=A(N1N1)-A(NN)
          U=T*T
          V=4.0*A(N1N)*A(NN1)
          IF(ABS(V)-U*E7) 100,100,65
       65 T=U+V
          IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67,67,68
       67 T=0.0
       68 U=(A(N1N1)+A(NN))/2.0
          V=SQRT(ABS(T))/2.0
          IF(T)140,70,70
       70 IF(U) 80,75,75
       75 RR(N1)=U+V
          RR(N)=U-V
          GO TO 130
       80 RR(N1)=U-V
          RR(N)=U+V
          GO TO 130
      100 IF(T)120,110,110
      110 RR(N1)=A(N1N1)
          RR(N)=A(NN)
          GO TO 130
```

```
  120 RR(N1)=A(NN)
      RR(N)=A(N1N1)
  130 RI(N)=0.0
      RI(N1)=0.0
      GO TO 160
140     RR(N1) = U
        RR(N)=U
      RI(N1)=V
      RI(N)=-V
  160 IF(N2)1280,1280,180
C
C        TESTS OF CONVERGENCE
C
  180 N1N2=N1N1-IA
      RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
      EPS=E10*SQRT(RMOD)
      IF(ABS(A(N1N2))-EPS)1280,1280,240
  240 IF(ABS(A(NN1))-E10*ABS(A(NN))) 1300,1300,250
  250 IF(ABS(PAN1-A(N1N2))-ABS(A(N1N2))*E6) 1240,1240,260
  260 IF(ABS(PAN-A(NN1))-ABS(A(NN1))*E6)1240,1240,300
  300 IF(IT-MAXIT) 320,1240,1240
C
C        COMPUTE THE SHIFT
C
  320 J=1
        DO 360 I = 1,2
      K=NP-I
      IF(ABS(RR(K)-PRR(I))+ABS(RI(K)-PRI(I))-DELTA*(ABS(RR(K))
     1      ABS(RI(K)))) 340,360,360
  340 J=J+I
  360 CONTINUE
      GO TO (440,460,460,480),J
  440 R=0.0
      S=0.0
      GO TO 500
  460 J=N+2-J
      R=RR(J)*RR(J)
      S=RR(J)+RR(J)
      GO TO 500
  480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
      S=RR(N)+RR(N1)
C
C        SAVE THE LAST TWO SUBDIAGONAL TERMS AND THE ROOTS OF THE
C        SUBMATRIX BEFORE ITERATION
C
  500 PAN=A(NN1)
      PAN1=A(N1N2)
      DO 520 I=1,2
      K=NP-I
      PRR(I)=RR(K)
  520 PRI(I)=RI(K)
C
C        SEARCH FOR A PARTITION OF THE MATRIX, DEFINED BY P AND Q
C
      P=N2
      IF (N-3)600,600,525
  525 IPI=N1N2
      DO 580 J=2,N2
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS) 600,600,530
  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(IP
     1 )) -ABS(D)*EPS) 620,620,560
  560 P=N1-J
  580 CONTINUE
  600 Q=P
      GO TO 680
  620 P1=P-1
      Q=P1
      IF (P1-1) 680,680,650
  650 DO 660 I=2, P1
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS)680,680,660
  660 Q=Q-1
C
C        QR DOUBLE ITERATION
C
  680 II=(P-1)*IA+P
      DO 1220 I=P,N1
      II1=II-IA
      IIP=II+IA
      IF(I-P)720,700,720
  700 IPI=II+1
      IPIP=IIP+1
C
C        INITIALIZATION OF THE TRANSFORMATION
C
      G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
      G2=A(IPI)*(A(IPIP)+A(II)-S)
      G3=A(IPI)*A(IPIP+1)
      A(IPI+1)=0.0
      GO TO 780
  720 G1=A(II1)
      G2=A(II1+1)
      IF(I-N2)740,740,760
  740 G3=A(II1+2)
      GO TO 780
  760 G3=0.0
  780 CAP=SQRT(G1*G1+G2*G2+G3*G3)
      IF(CAP)800,860,800
  800 IF(G1)820,840,840
  820 CAP=-CAP
  840 T=G1+CAP
      PSI1=G2/T
      PSI2=G3/T
      ALPHA=2.0/(1.0+PSI1*PSI1+PSI2*PSI2)
      GO TO 880
  860 ALPHA=2.0
      PSI1=0.0
      PSI2=0.0
  880 IF(I-Q)900,960,900
  900 IF(I-P)920,940,920
  920 A(II1)=-CAP
      GO TO 960
  940 A(II1)=-A(II1)
C
C        ROW OPERATION
C
```

```
  960 IJ=II
      DO 1040 J=I,N
      T=PSI1*A(IJ+1)
      IF(I-N1)980,1000,1000
  980 IP2J=IJ+2
      T=T+PSI2*A(IP2J)
 1000 ETA=ALPHA*(T+A(IJ))
      A(IJ)=A(IJ)-ETA
      A(IJ+1)=A(IJ+1)-PSI1*ETA
      IF(I-N1)1020,1040,1040
 1020 A(IP2J)=A(IP2J)-PSI2*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
      JIP=IP+J
      JI=JIP-IA
      T=PSI1*A(JIP)
      IF(I-N1)1120,1140,1140
 1120 JIP2=JIP+IA
      T=T+PSI2*A(JIP2)
 1140 ETA=ALPHA*(T+A(JI))
      A(JI)=A(JI)-ETA
      A(JIP)=A(JIP)-ETA*PSI1
      IF(I-N1)1160,1180,1180
 1160 A(JIP2)=A(JIP2)-ETA*PSI2
 1180 CONTINUE
      IF(I-N2)1200,1220,1220
 1200 JI=II+3
      JIP=JI+IA
      JIP2=JIP+IA
      ETA=ALPHA*PSI2*A(JIP2)
      A(JI)=-ETA
      A(JIP)=-ETA*PSI1
      A(JIP2)=A(JIP2)-ETA*PSI2
 1220 II=IIP+1
      IT=IT+1
      GO TO 60
C
C           END OF ITERATION
C
 1240 IF(ABS(A(NN1))-ABS(A(N1N2))) 1300,1280,1280
C
C           TWO EIGENVALUES HAVE BEEN FOUND
C
 1280 IANA(N)=0
      IANA(N1)=2
      N=N2
      IF(N2)1400,1400,20
C
C           ONE EIGENVALUE HAS BEEN FOUND
C
```

```
 1300 RR(N)=A(NN)
      RI(N)=0.0
      IANA(N)=1
      IF(N1)1400,1400,1320
 1320 N=N1
      GO TO 20
 1400 RETURN
      END
C        SUBROUTINE HSBG
C
C        PURPOSE
C           TO REDUCE A REAL MATRIX INTO UPPER ALMOST TRIANGULAR F
C
C        USAGE
C           CALL HSBG(N,A,IA)
C
C        DESCRIPTION OF THE PARAMETERS
C           N      ORDER OF THE MATRIX
C           A      THE INPUT MATRIX, N BY N
C           IA     SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARR
C                  A IN THE CALLING PROGRAM WHEN THE MATRIX IS IN
C                  DOUBLE SUBSCRIPTED DATA STORAGE MODE.  IA=N WHE
C                  THE MATRIX IS IN SSP VECTOR STORAGE MODE.
C
C        REMARKS
C           THE HESSENBERG FORM REPLACES THE ORIGINAL MATRIX IN TH
C           ARRAY A.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           SIMILARITY TRANSFORMATIONS USING ELEMENTARY ELIMINATIO
C           MATRICES, WITH PARTIAL PIVOTING.
C
C        REFERENCES
C           J.H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROBLEM -
C           CLARENDON PRESS, OXFORD, 1965.
C
C     ..................................................................
C
      SUBROUTINE HSBG(N,A,IA)
      DIMENSION A(1)
      DOUBLE PRECISION S
      L=N
      NIA=L*IA
      LIA=NIA-IA
C
C        L IS THE ROW INDEX OF THE ELIMINATION
C
   20 IF(L-3) 360,40,40
   40 LIA=LIA-IA
      L1=L-1
      L2=L1-1
C
C        SEARCH FOR THE PIVOTAL ELEMENT IN THE LTH ROW
C
```

```
      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
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
C         INTERCHANGE THE ROWS
C
      M=L2-M/IA
      DO 160 I=L1,NIA,IA
      T=A(I)
      J=I-M
      A(I)=A(J)
  160 A(J)=T
C
C         TERMS OF THE ELEMENTARY TRANSFORMATION
C
  180 DO 200 I=L,LIA,IA
  200 A(I)=A(I)/A(ISUB)
C
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
C         LEFT TRANSFORMATION
C
```

```
      K=-IA
      DO 300 I=1,M
      K=K+IA
      LK=K+L1
      S=A(LK)
      LJ=L-IA
      DO 280 J=1,L2
      JK=K+J
      LJ=LJ+IA
  280 S=S+A(LJ)*A(JK)*1.0D0
  300 A(LK)=S
C
C        SET THE LOWER PART OF THE MATRIX TO ZERO
C
      DO 310 I=L,LIA,IA
  310 A(I)=0.0
  320 L=L1
      GO TO 20
  360 RETURN
        END
      SUBROUTINE MLTMX(A,S,M,MDIM)
C
C     SUBROUTINE OBTAINS THE MATRIX MULTIPLICATION OF A AND S AND
C         THE RESULTS IN S.
C
      DIMENSION S(MDIM,MDIM),A(MDIM,MDIM)
      COMMON/WORK/T(25,25)
      DO 10 I=1,M
      DO 10 J=1,M
      C=0.0
      DO 20 K=1,M
   20 C=C+A(I,K)*S(K,J)
   10 T(I,J)=C
      DO 50 I=1,M
      DO 50 J=1,M
   50 S(I,J)=T(I,J)
      RETURN
      END
      BLOCK DATA WORK
      COMMON /WORK/ WO(625)
      END
$
```

```
&FILTR T=00004 IS ON CR00022 USING 00004 BLKS R=0022

FTN4,L
      PROGRAM FILTR
C WRITTEN BY E. E. SHERROD
C
C THIS PROGRAM SELECTS THE FILTERING TYPE
C
      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
C GET LU
C
      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 RMPAR(IRTN)
      IF(IFITR .EQ. 1) GO TO 30
      IF(IFITR .EQ. 2) CALL EXEC(23,NAME2,LU,IPIXL,JPIXL,0,0)
      CALL RMPAR(IRTN)
30    WRITE(LU,40) RMAX,RMIN
40    FORMAT(" MAX PIXEL = ",F12.2,10X," MIN PIXEL = ",1F12.2)
      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
      END$
```

```
&NOISE T=00004 IS ON CR00022 USING 00010 BLKS R=0097

FTN4,L
      PROGRAM NOISE
C
      DIMENSION RDATA(512),GNOISE(512),LU(5),IU(5),IBUF(40)
C
      INTEGER READL
      EQUIVALENCE (RDATA,LU(2)),(LU(2),ILINE),(LU(3),IPIXL),
     1 (RDATA(2),RMAX),(RDATA(3),RMIN)
      DATA RDATA/512*0.0/
C
C GET INPUT PARAMETERS
C
      CALL RMPAR(LU)
C
C SCHEDULE BUILD WORK FILE PROGRAM
      CALL EXEC(23,6HBLDWF ,IU)
C
C  READ WORK FILE HEADER
C
      IERR = READL(-1,0,511,RDATA)
      IF (IERR .LT. 0) GO TO 999
      NLINE=ILINE
      NPIXL=IPIXL
      PMAX=RMAX
      PMIN=RMIN
C
C GET NOISE INFO
      WRITE(LU,13)
13    FORMAT(" ENTER NOISE MEAN VALUE _")
      READ(LU,*) AM
      WRITE(LU,14)
14    FORMAT(" ENTER STANDARD DEVIATION VALUE _")
      READ(LU,*) S
      IF(S .LE. 0) GO TO 1000
C
      DO 100 I=0,NLINE-1
      IF (READL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
C
C GET NOISE
C
      DO 101 JA=0,51
      CALL EXEC(1,8,IBUF,40)
      JJ=10*JA
      CALL CODE (80)
      READ( IBUF,12) (GNOISE(K+JJ),K=1,10 )
12    FORMAT(10F8.5)
101   CONTINUE
```

```
C
      DO 90 J =1,NPIXL
      RDATA(J)=RDATA(J) +GNOISE(J)*S+AM
600   FORMAT( F20.3)
90    CONTINUE
C
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
1000  CALL CLSWF(NLINE,NPIXL,PMAX,PMIN)
      CALL CLOSE(IDCB1)
      CALL EXEC(6)
999   WRITE(LU,2) IERR
2     FORMAT("FILE ERROR",I7)
      END
$
$
```

APPENDIX E

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 multiinput–multioutput (MIMO) system similar to the state-space representation of the one-dimensional digital recursive filter. In this paper, a pseudo-state representation is used and three coefficient matrices are obtained. A general theorem for stability of two-dimensional digital recursive filters is derived and a very useful theorem is presented which expresses sufficient requirements for instability in terms of the spectral radii of these matrices.

I. Introduction

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

$$g(m,n)=\sum_{J=0}^{L}\sum_{K=0}^{L}a_{JK}f(m-J,n-K)-\sum_{\substack{J=0\\J+K>0}}^{L}\sum_{K=0}^{L}b_{JK}g(m-J,n-K) \quad (1)$$

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].

Manuscript received October 10, 1975; revised March 7, 1979. This work was supported in part by the U.S. Department of Energy and also by the Naval Air Systems Command.

W. E. Alexander is with the Department of Electrical Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411.

S. A. Pruess is with the Department of Mathematics, The University of New Mexico, Albuquerque, NM 87106.

0098-4094/80/0100-0011$00.75 ©1980 IEEE

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

$$\left.\begin{aligned} G_{m,n} &= \bar{B}_1 G_{m,n-1} + \bar{B}_2 G_{m-1,n} + \bar{A} F_{m,n} \\ g(m,n) &= D G_{m,n} \end{aligned}\right\} \qquad (2)$$

$G_{m,n}$ 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

$$\begin{aligned} g(m,n) = {} & a_{00} f(m,n) + a_{10} f(m-1,n) + a_{01} f(m,n-1) \\ & + a_{11} f(m-1,n-1) - b_{10} g(m-1,n) \\ & - b_{01} g(m,n-1) - b_{11} g(m-1,n-1). \end{aligned} \qquad (3)$$

For this example, with $G_{m,n}$ and $F_{m,n}$ given in transpose form, we have

$$G_{m,n} = [\, g(m,n) \;\; g(m-1,n) \;\; g(m,n-1) \;\; g(m-1,n-1) \,]^T \qquad (4)$$

$$F_{m,n} = [\, f(m,n) \;\; f(m-1,n) \;\; f(m,n-1) \;\; f(m-1,n-1) \,]^T \qquad (5)$$

$$\bar{B}_1 = \begin{bmatrix} -b_{10} & 0 & -\frac{1}{2} b_{11} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

$$\bar{B}_2 = \begin{bmatrix} -b_{01} & -\frac{1}{2} b_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix} \qquad (6)$$

$$\bar{A} = \begin{bmatrix} a_{00} & a_{10} & a_{01} & a_{11} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \qquad (7)$$

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., $\sum_{m,n=0}^{\infty} |h(m,n)| < \infty$ [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

$$\begin{aligned} G_{J,K} &= \bar{A} H_{J,K} \\ h(J,K) &= D G_{J,K}. \end{aligned} \qquad (8)$$

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

$$C(\bar{B}_1^J, \bar{B}_2^K) = \binom{J+K}{K} \bar{B}_1^J \bar{B}_2^K = (J+K)!\, \bar{B}_1^J \bar{B}_2^K / (J!\,K!).$$

In general, the matrices do not commute. Therefore, we give as an example $C(\bar{B}_1^2, \bar{B}_2^1) = \bar{B}_1^2 \bar{B}_2 + \bar{B}_1 \bar{B}_2 \bar{B}_1 + \bar{B}_2 \bar{B}_1^2$.

Lemma 1

The contribution to the output vector, $G_{m,n}$, for a single input vector, $H_{J,K}$, which corresponds to a unit impulse at the (J,K) position where $J \le m$ and $K \le n$, is given by $G_{m,n} = C(\bar{B}_1^{m-J}, \bar{B}^{n-K}) \bar{A} H_{J,K}$ for the LSI system represented by (2).

The proof of Lemma 1 is given in the Appendix. Lemma 1 provides a convenient means of finding the output of the two-dimensional digital recursive filter for all values of m and n when the filter is excited by a single input at any point in the array. Since the filter is linear and shift invariant, we can use the principle of superposition to find the output for any particular sequence of inputs. Thus the unit impulse response of the filter is given by

$$G_{m,n} = C(\bar{B}_1^m, \bar{B}_2^n)\bar{A} H_{0,0} \quad (9)$$

$$h(m,n) = DG_{m,n} = DC(\bar{B}_1^m, \bar{B}_2^n)\bar{A} H_{0,0}. \quad (10)$$

Lemma 2

Given the discrete LSI system represented by (2) for which the corresponding transfer function has mutually prime numerator and denominator polynomials. If the contribution to the output vector $G_{m,n}$ by a bounded sequence of input vectors $F_{J,K}$ where $0 < J < m$ and $0 < K < n$ can be expressed by $G_{m,n} = \bar{Q}^m \bar{A} F_{J,K}$ or $G_{m,n} = \bar{Q}^n \bar{A} F_{J,K}$, then the system is unstable if $\rho(\bar{Q})$, the spectral radius of $\bar{Q}$, is greater than one. The proof of Lemma 2 is given in the Appendix.

Theorem 1

The discrete LSI system represented by (2) is stable if and only if for at least one matrix norm

$$S = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \| DC(\bar{B}_1^m, \bar{B}_2^n) A H_{0,0} \| < \infty. \quad (11)$$

Theorem 1 follows directly from (10) and the requirement that the discrete impulse response be absolutely summable. Since $h(m,n)$ is a scalar, its matrix norm is equivalent to its absolute value and the proof of Theorem 1 is obvious.

Theorem 2

The discrete LSI system 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 $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ is greater than or equal to one. The proof of Theorem 2 is given in the Appendix.

In the practical application of two-dimensional digital recursive filters, any filter with $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ equal to one can be considered to be unstable and should be avoided [8]. Goodman [5] has shown by clever examples that two-dimensional filters with nonessential singularities of the second kind on the unit bidisc may be stable. Such a filter may have $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ equal to one. However, roundoff errors and coefficient truncation would prevent satisfactory performance by such a filter for most applications.

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 $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ 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.

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 $G_{J+1,K}$, $G_{J,K+1}$, and $G_{J+1,K+1}$ for input vector $H_{J,K}$ and if all initial condition vectors are null vectors, we obtain

$$\left.\begin{aligned} G_{J+1,K} &= \bar{B}_1 G_{J,K} = \bar{B}_1 \bar{A} H_{J,K} \\ G_{J,K+1} &= \bar{B}_2 G_{J,K} = \bar{B}_2 \bar{A} H_{J,K} \\ G_{J+1,K+1} &= \bar{B}_1 G_{J,K+1} + \bar{B}_2 G_{J+1,K} = (\bar{B}_1\bar{B}_2 + \bar{B}_2\bar{B}_1)\bar{A} H_{J,K} \end{aligned}\right\}. \quad (A1)$$

If we use Lemma 1, we obtain

$$\left.\begin{aligned} G_{J+1,K} &= C(\bar{B}_1^1, \bar{B}_2^0)\bar{A} H_{J,K} = \bar{B}_1 \bar{A} H_{J,K} \\ G_{J,K+1} &= C(\bar{B}_1^0, \bar{B}_2^1)\bar{A} H_{J,K} = \bar{B}_2 \bar{A} H_{J,K} \\ G_{J+1,K+1} &= C(\bar{B}_1^1, \bar{B}_2^1)\bar{A} H_{J,K} = (\bar{B}_1\bar{B}_2 + \bar{B}_2\bar{B}_1)\bar{A} H_{J,K} \end{aligned}\right\}. \quad (A2)$$

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

$$G_{m+1,n} = \bar{B}_1 G_{m,n} + \bar{B}_2 G_{m+1,n-1}. \quad (A3)$$

Then using (9) to find expressions for $G_{m,n}$ and $G_{m+1,n-1}$, we have

$$G_{m+1,n} = \left[\bar{B}_1 C(\bar{B}_1^{m-J}, \bar{B}_2^{n-K}) + \bar{B}_2 C(\bar{B}_1^{m-J+1}, \bar{B}_2^{n-K-1})\right]\bar{A} H_{J,K}. \quad (A4)$$

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

$$C(\bar{B}_1^J, \bar{B}_2^K) = \bar{B}_1 C(\bar{B}_1^{J-1}, \bar{B}_2) + \bar{B}_2 C(B_1^J, \bar{B}_2^{K-1}) \quad (A5)$$

for all J and K, such that both J and K are greater than or equal to one. It follows directly that

$$G_{m+1,n}=C\left(\bar{B}_1^{m+1-J},\bar{B}_2^{n-K}\right)\bar{A}H_{J,K}. \qquad (A6)$$

Similarly from (2) we can write

$$G_{m,n+1}=\bar{B}_1G_{m-1,n+1}+\bar{B}_2G_{m,n}. \qquad (A7)$$

Using (9) to find expressions for $G_{m-1,n+1}$ and $G_{m,n}$, we have

$$G_{m,n+1}=\left[\bar{B}_1C\left(\bar{B}_1^{m-1-J},\bar{B}_2^{n+1-K}\right)+\bar{B}_2C\left(\bar{B}_1^{m-J},\bar{B}_2^{n-K}\right)\right]\bar{A}H_{J,K}. \qquad (A8)$$

It follows that

$$G_{m,n+1}=C\left(\bar{B}_1^{m-J},\bar{B}_2^{n+1-K}\right)\bar{A}H_{J,K}. \qquad (A9)$$

Finally, from (2) we obtain

$$G_{m+1,n+1}=\bar{B}_1G_{m,n+1}+\bar{B}_2G_{m+1,n}. \qquad (A10)$$

Using Lemma 1 to express $G_{m,n+1}$ and $G_{m+1,n}$, we obtain

$$G_{m+1,n+1}=\left[\bar{B}_1C\left(\bar{B}_1^{m-J},\bar{B}_2^{n+1-K}\right)+\bar{B}_2C\left(\bar{B}_1^{m+1-J},\bar{B}_2^{n-K}\right)\right]\bar{A}H_{J,K}. \qquad (A11)$$

It follows from (A5) and (A11) that

$$G_{m+1,n+1}=C\left(\bar{B}_1^{m+1-J},\bar{B}_2^{n+1-K}\right)\bar{A}H_{J,K} \qquad (A12)$$

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 $\rho(\bar{Q})>1$ [9].

Define the eigenvalue corresponding to the spectral radius of $\bar{Q}$ as λ_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

$$\bar{A}F_{J,n}=\epsilon P_Q+R_{J,n} \qquad \text{for all } J \text{ and } n. \qquad (A13)$$

where ϵ is an arbitrary nonzero finite constant and $R_{J,n}$ is not in the direction of P_Q. We then have

$$G_{m,n}=\bar{Q}^m\bar{A}F_{J,n}=\epsilon\bar{Q}^mP_Q+\bar{Q}^mR_{J,n}. \qquad (A14)$$

Then since λ_Q is the eigenvalue corresponding to the spectral radius, the norm of $G_{m,n}$ is dominated by the term $\epsilon\bar{Q}^mP_Q$ in the limit as m approaches infinity. Thus

$$S=\lim_{m\to\infty}\|G_{m,n}\|=\lim_{m\to\infty}\|\epsilon\bar{Q}^mP_Q\|=\lim_{m\to\infty}\|\epsilon\lambda_Q^mP_Q\|. \qquad (A15)$$

Note that S is infinite if λ_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

$$G_{M,N}=f(J,K)C\left(\bar{B}_1^{M-J},\bar{B}_2^{N-K}\right)\bar{A}H_{J,K}$$
$$g(M,N)=DG_{M,N} \qquad (A16)$$

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

$$G_{M,N}=f(0,N)C\left(\bar{B}_1^M,\bar{B}_2^0\right)\bar{A}H_{0,N}=f(0,N)\bar{B}_1^M\bar{A}H_{0,N}. \qquad (A17)$$

If we apply Lemma 2, we see that the system is unstable if $\rho(\bar{B}_1)>1$. If we let $J=M$ and $K=0$ in (A16), we have

$$G_{M,N}=f(M,0)C\left(\bar{B}_1^0,\bar{B}_2^N\right)\bar{A}H_{M,0}=f(M,0)\bar{B}_2^N\bar{A}H_{M,0}. \qquad (A18)$$

If we apply Lemma 2, we see that the system is unstable if $\rho(\bar{B}_2)>1$.

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

$$G_{M,M}=\sum_{J=0}^{M}f(J,M-J)C\left(\bar{B}_1^{M-J},\bar{B}_2^J\right)\bar{A}H_{J,M-J}. \qquad (A19)$$

Since all inputs are equal, we can write

$$G_{M,M}=f(0,M)\left[\sum_{J=0}^{M}C\left(\bar{B}_1^{M-J},\bar{B}_2^J\right)\right]\bar{A}H_{0,M} \qquad (A20)$$

$$G_{M,M}=f(0,M)\left(\bar{B}_1+\bar{B}_2\right)^M\bar{A}H_{0,M} \qquad (A21)$$

since

$$\left(\bar{B}_1+\bar{B}_2\right)^M=\sum_{J=0}^{M}C\left(\bar{B}_1^{M-J},\bar{B}_2^J\right) \qquad (A22)$$

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

Acknowledgment

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

References

[1] Lawrence R. Rabiner and Bernard Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975, pp. 442–455.

[2] Samuel Stearns, *Digital Signal Analysis*. Hayden Publishing Company, 1975, p. 134.

[3] E. I. Jury, "Theory and application of inners," *Proc. IEEE*, vol. 63, pp. 1044–1068, 1975.

[4] Katsuhiko Ogata, *State Space Analysis of Control Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1967, p. 487.

[5] Dennis Goodman, "Some stability properties of two-dimensional linear shift-invariant digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 201–208, 1977.

[6] E. Fornasini and G. Marchesini, "State space realization theory for two-dimensional filters," *IEEE Trans. Automat. Contr.*, vol. AC-21, pp. 484–492, Aug. 1976.

[7] Winser E. Alexander, "Stability and synthesis of two-dimensional digital recursive filters," Ph.D. dissertation, Univ. of New Mexico, Albuquerque, NM, 1974 (University Microfilms, Ann Arbor, MI).

[8] N. K. Bose, "Problems and progress in multidimensional system theory", *Proc. IEEE*, vol. 65, pp. 824–840, 1977.
[9] Alston S. Householder, *The Theory of Matrices in Numerical Analysis*. New York, NY: Blaisdell, 1964, ch. 2.

Winser 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

Winser E. Alexander

Department of Electrical Engineering
North Carolina A&T State University

ABSTRACT

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

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

INTRODUCTION

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

$$g(m,n)=\sum_{J=0}^{L}\sum_{K=0}^{L} a_{JK}f(m-J,n-K) - \sum_{\substack{J=0\\J+K>0}}^{L}\sum_{K=0}^{L} b_{JK}g(m-J,n-K) \qquad (1)$$

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

The Matrix Recursive Form

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

$$G_{m,n} = \underline{B}_1 G_{m-1,n} + \underline{B}_2 G_{m,n-1} + \underline{A} F_{m,n} \qquad (2)$$

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 $\underline{B}_1$, $\underline{B}_2$ and $\underline{A}$ are appropriate matrices to make (1) and (2) equivalent. The matrices $\underline{B}_1$, $\underline{B}_2$ and $\underline{A}$ are all of order $(L+1)^2$ by $(L+1)^2$. The current output is then given by $g(m,n) = DG_{m,n}$ where D is a row vector with (L+1) elements.

The ordering of the outputs in $G_{m,n}$ and of the inputs in $F_{m,n}$ is not unique. However, the ordering does affect the relative position of the elements of the corresponding $\underline{B}_1$ and $\underline{B}_2$ matrices. Also note that there are identical elements in $\underline{B}_1$ and $\underline{B}_2$. Where this occurs, the corresponding elements of $\underline{B}_1$ and $\underline{B}_2$ 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 eq[illegible]y and choose a particular ordering scheme.

Example 1

Consider the recursive digital filter with bivariate difference equation given by

$$g(m,n) = f(m,n) - b_{10}g(m-1,n) - b_{01}g(m,n-1) - b_{11}g(m-1,n-1) \qquad (3)$$

For this example, we have

$$G_{m,n} = \begin{bmatrix} g(m,n) \\ g(m-1,n) \\ g(m,n-1) \\ g(m-1,n-1) \end{bmatrix}; \quad F_{m,n} = \begin{bmatrix} f(m,n) \\ f(m-1,n) \\ f(m,n-1) \\ f(m-1,n-1) \end{bmatrix}$$

$$\underline{B}_1 = \begin{bmatrix} -b_{10} & 0 & \tfrac{1}{2}b_{11} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \tfrac{1}{2} & 0 \end{bmatrix} ; \quad \underline{B}_2 = \begin{bmatrix} -b_{01} & -\tfrac{1}{2}b_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \tfrac{1}{2} & 0 & 0 \end{bmatrix} \quad (5)$$

and

$$\bar{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6)$$

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 [3]. The usual approach for stability analysis of two dimensional digital recursive filters involves examining regions of analyticity for bivariate polynomials [4] which is computational 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 corresponsing transfer function are mutually prime is unstable if any one of the spectral radii $\rho(\underline{B}_1)$, $\rho(\underline{B}_2)$, $\rho(\underline{B}_1 + \underline{B}_2)$ is greater than or equal to one. The spectral ratius of a given matrix is the magnitude of the largest magnitude eigenvalue of that matrix).

Theroem 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 $\underline{B}_1$ and $\underline{B}_2$ is less than one ($\rho\,[\text{abs}\,(\underline{B}_1) + \text{abs}\,(\underline{B}_2)] < 1$).

Theorem 3

There is a particular permutation matrix $\underline{S}$ [5] such that if $\rho(\underline{B}_1)$, $\rho(\underline{B}_2)$ $\rho(\underline{B}_1 + \underline{B}_2)$ are all less than one, then the LSI digital recursive filter is stable if both $\rho(\underline{B}_1\underline{S})$ and $\rho(\underline{B}_2\underline{S})$ 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 $\rho(\underline{B}_1)$, $\rho(\underline{B}_2)$, and $\rho(\underline{B}_1+\underline{B}_2)$ are all less than one.

Example 2

From Theorem 1, we obtain the results that the filter represented by (3) is unstable if $|b_{01}| \geq 1$, $|b_{10}| \geq 1$, or if

$$\max\left[\left|\frac{-(b_{10}+b_{01})}{2} \pm \frac{1}{2}\sqrt{(b_{10}+b_{01})^2-4b_{11}}\right|\right] \geq 1 \quad (7)$$

Example 3

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

$$g(m,n) = f(m,n) + 0.95\, g(m-1,n) + 0.95\, g(m,n-1) -0.5\, g(m-1,n-1) \quad (8)$$

If we apply Theorem 1, we obtain $\rho(\underline{B}_1) = 0.95$, $\rho(\underline{B}_2) = 0.95$ and $\rho(\underline{B}_1 + \underline{B}_2) = 1.584$. 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

$$g(m,n) = f(m,n) + 0.75\, g(m-1,n) - 1.5\, g(m,n-1) + 0.9\, g(m-2,n) - 1.2\, g(m,n-2) - 1.3\, g(m-2,n-1) -0.9\, g(m-1,n-2) - 0.5\, g(m-2,n-2) \quad (9)$$

If we apply Theroem 1, we obtain $\rho(\underline{B}_1) = 1.095$, $\rho(\underline{B}_2) = 0.949$ and $\rho(\underline{B}_1+\underline{B}_2) = 1.284$. We conclude as did Read and Treitel that this filter is unstable.

Example 5

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

$$g(m,n) = f(m,n) - 0.5\, g(m-1,n) -0.5\, g(m,n-1) -0.25\, g(m-1,n-1 - 0.25\, g(m-2,n) -0.25\, g(m,n-2) \quad (10)$$

If we apply Theorem 3, we obtain $\rho(\underline{B}_1) = 0.5$, $\rho(\underline{B}_2) = 0.5$, $\rho(\underline{B}_1 + \underline{B}_2) = 0.866$; $\rho(\underline{B}_1\underline{S}) = \rho(\underline{B}_2\underline{S}) = 0.35355$. Therefore, we conclude that this filter is stable. This filter was verified to be stable by Maria and Fahmy [8].

Example 6

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

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

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

$$\left| b_{10} \right| + \left| b_{01} \right| < 1 \quad (12)$$

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

$$\underline{V} = \begin{bmatrix} 1.0 & b_{01} \\ b_{10} & b_{11} \end{bmatrix} \quad (13)$$

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

Example 7

Consider the example used by Read and Treitel where $\underline{V}$ is given by

$$\underline{V} = \begin{bmatrix} 1.0 & 1.5 & -1.9 & -0.8 & 1.1 \\ 1.4 & 2.1 & -2.6 & -1.1 & 1.5 \\ -1.8 & -2.4 & 3.3 & 1.3 & -1.6 \\ -0.7 & -0.9 & 1.1 & 0.5 & -0.8 \\ -0.9 & 1.3 & -1.6 & -0.6 & 1.0 \end{bmatrix} \quad (14)$$

For this example, $\rho(\underline{B}_1) = 2.169$; $\rho(\underline{B}_2) = 2.104$ and $\rho(\underline{B}_1 + \underline{B}_2) = 2.599$. 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 $\underline{B}_1$ and $\underline{B}_2$ matrices is very simple and straight forward 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

[1] Samuel Stearns, Digital Signal Analysis, Hayden Publishing Company, 1975, p. 134.

[2] E. I. Jury, "Theory and Applications of Inners", Proc. IEEE, Vol. 63, No. 7, 1975, pp 1044-1068.

[3] Katsuhiko Ogata, State Space Analysis of Control Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967, p. 487.

[4] Dennis Goodman, "Some Stability Properties of Two Dimensional Linear-Shift Invariant Digital Filters", IEEE Transactions on Circuits and Systems, Vol CAS24, No. 4, 1977, pp 201-208.

[5] Winser E. Alexander, Stability and Synthesis of Two-Dimensional Digital Recursive Filters, Ph.D. Dissertation, University of New Mexico, Albuquerque, N. M., 1974 (Available from University Microfilms, Ann Arbor, Mich.).

[6] J. L. Shanks, Sven Treitel and J. H. Justice, "Stability and Synthesis of Two Dimensional Recursive Filters", IEEE Transactions on Audio and Electroacoustics, Vol. Au-20, No. 2, 1972 pp 115-128.

[7] Randel R. Read and Sven Treitel, "The Stabilization of Two Dimensional Recursive Filters Via the Discrete Hilbert Transform", IEEE Transactions on Geoscience Electronics, Vol. GE-11, No. 3, 1973, pp 153-160.

[8] G. A. Maria and M. M. Fahmy, "On the Stability of Two Dimensional Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. Au-21, 1973, pp 470-472.

[9] Thomas S. Huang, "Stability of Two Dimensional Recursive Filters", IEEE Transactions on Audio and Electroacoustics, Vol. Au-20, No. 2, 1972, pp 158-163.

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 pseudo-state 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 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

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

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

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

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

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

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

$$H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n) z^{-m} w^{-n} \qquad (2.4)$$

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

$$H(z,w) = \frac{\sum_{j=0}^{L}\sum_{k=0}^{L} a(j,k)z^{-m}w^{-n}}{\sum_{j=0}^{L}\sum_{k=0}^{L} b(j,k)z^{-m}w^{-n}} \qquad (2.5)$$

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

$$G(z,w) = H(z,w)F(z,w) \qquad (2.6)$$

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

$$g(m,n) = \sum_{j=0}^{L}\sum_{k=0}^{L} a(j,k)f(m-j,n-k) - \sum_{\substack{j=0 \\ j+k>0}}^{L}\sum_{k=0}^{L} b(j,k)g(m-j,n-k) \qquad (2.7)$$

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 straight forward. 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 [3]. Stability analysis is not straight forward for the two dimensional problem because a two variable polynomial is not generally factorable into distinct roots. When the polynomial in the denominator of the two dimensional Z-Transform of the impulse response is factorable into distinct roots, the stability analysis procedure is the same as for the one dimensional problem.

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

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

$$G_{m,n} = \bar{B}G_{m,n-1} + \bar{C}G_{m-1,n} + \bar{A}F_{m,n} \qquad (3.1)$$

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

If the filter is unstable, then either $\bar{B}$, $\bar{C}$ or $(\bar{B} + \bar{C})$ has at least one eigenvalue with a magnitude greater than or equal to one. Thus, stability analysis involves setting up the matrices $\bar{B}$ and $\bar{C}$ 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

$$H(z,w) = H1(z)H2(w) \qquad (4.1)$$

or as the sum

$$H(z,w) = H1(z) + H2(w) \qquad (4.2)$$

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 Transfor (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.

$$H(r,s) = \frac{1}{1 + (-1)^n(r^2 + s^2)^n/R^{2n}} \qquad (4.3)$$

where r and s 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,s) to obtain

$$H1(r,s) = \frac{1}{1 + a^{2n}(r^2 + s^2)^n} \qquad (4.4)$$

where

$$a^2 = (-1)/\tan^2(RT/2) \qquad (4.5)$$

(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 ZW-Transform for the squared magnitude characteristic of the desired filter.

$$H(z,w) = \frac{1}{1+a^{2n}[((z-1)/(z+1))^2+((w-1)/(w+1))^2]^n} \qquad (4.6)$$

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.

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

However, the polynomial in the denominator of (4.6) is not factorable into distinct roots. Therefore, forming of the minimum phase version of H(z,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 $\bar{B}$ and $\bar{C}$ of (3.1) which corresponds to (4.6).

2. Calculate the eigenvalues of the matrix sum ($\bar{B}$ + $\bar{C}$). They occur in reciprocal pairs.

3. Form the minimum phase approximation of the filter by using the smaller magnitude eigenvalue of each of the reciprocal pairs as a root of z and of w for the denominator polynomial and by using the minimum phase version of the numerator polynomial.

The resulting filter ZW-Transform is given by

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

where

$$p = \frac{(2a - (2\sqrt{2})a + 1)}{1 - 2a} \qquad (4.8)$$

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:

$$H(z,w) = \prod_i \frac{Ai[z^2+q(1i)z+q(2i)][w^2+q(1i)w+q(2i)]}{[z^2+p(1i)z+p(2i)][w^2+p(1i)w+p(2i)]} \qquad (5.1)$$

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

$$LP(z,w) = \frac{(1+p)^4(z^2+2z+1)(w^2+2w+1)}{16(z^2+2pz+p^2)(w^2+2pw+p^2)} \qquad (5.2)$$

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:

$$H(z,w) = c + d|LP(z,w)|^2 \qquad (5.3)$$

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:

$$H(z,w) = \frac{N(z,w)}{D(z,w)} \qquad (5.4)$$

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

A high frequency boost filter can be designed by changing the values of c and d in (5.3). For the high 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:

$$c = 1.0 \qquad d = BF - 1.0 \qquad (5.5)$$

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:

$$c = BF \qquad d = 1.0 - BF \qquad (5.6)$$

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

6.0 FILTER DESIGN EXAMPLES

Figure 1 gives the perspective plot of a lowpass filter designed with the described technique with a cut off frequency which is 0.3 times the Nyquist frequency. Figure 2 gives the contour plot for this filter design. Figure 3 gives the perspective plot for a high frequency boost filter with a break frequency of 0.5 times the Nyquist frequency and a boost magnitude of 25.6. Figure 4 gives the contour plot for this filter design. Note that these examples are essentially circularly symmetric. Some degradation is observed as the break frequency approaches the Nyquist frequency. This 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 polynominal of the ZW-Transform is minimum phase.

REFERENCES

1. Ernest L. Hall, "A Comparison of Computations for Spatial Filtering", Proceedings of the IEEE, Vol. 60, no. 7, 1972, pp 887-891.

2. Lawrence R. Rabiner and Benard Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1975, pp 442-455.

3. Samuel Stearns, Digital Signal Analysis, Hayden Publishing Co., Inc., Edison, N. J. p 134.

4. N. K. Bose, "Problems and Pr[illegible]ress in Multidimensional System Theory", Pr[illegible]dings of IEEE, Vol. 65, No. 6, 1977, pp [illegible].

5. Katsuhiko Ogata, State Space [illegible]sis of Control Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., p. 487.

6. Winser E. Alexander and Steven A. Pruess, "Stability Analysis of Two Dimensional Digital Recursive Filters", accepted for publication in IEEE Transactions on Circuits and Systems.

7. Winser E. Alexander, "Stability Analysis of Two Dimensional Digital Recursive Filters", 12th Asilomar Conference on Circuits, Systems, and Computers, November, 1978.

8. E. Fornasini and G. Marchesini, "State Space Realization Theory for Two Dimensional Filters", IEEE Transactions on Automatic Control, Vol. AC-21 1976, pp 484-492.

9. Bernard Gold and Charles Rader, Digital Processing of Signals, McGraw Hill, Inc., New York, N. Y., 1969.

10. Andreas Antoniou, Digital Filters; Analysis and Design, McGraw Hill, Inc., New York, N. Y., 1979.

11. Athanasios Papoulis, Systems and Transforms with Applications in Optics, McGraw Hill, Inc., New York, N. Y., 1968.

12. Dennis Goodman, "A Design Technique for Circularly Symmetric Low Pass Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 4, 1978, pp 290-304.

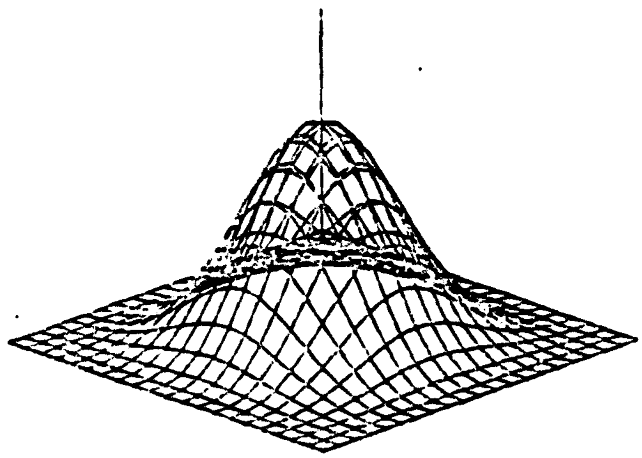

FIGURE 1. LOW PASS FILTER
STAGE = 1 RC = Ø.3

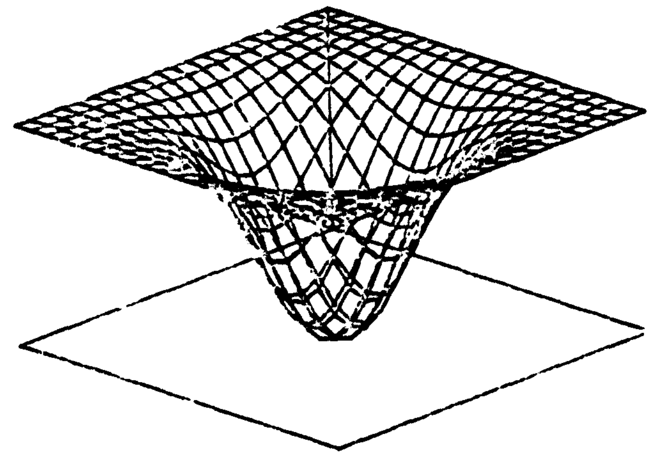

FIGURE 3. HIGH BOOST FILTER
BOOST FACTOR = 25.6 RC = Ø.5

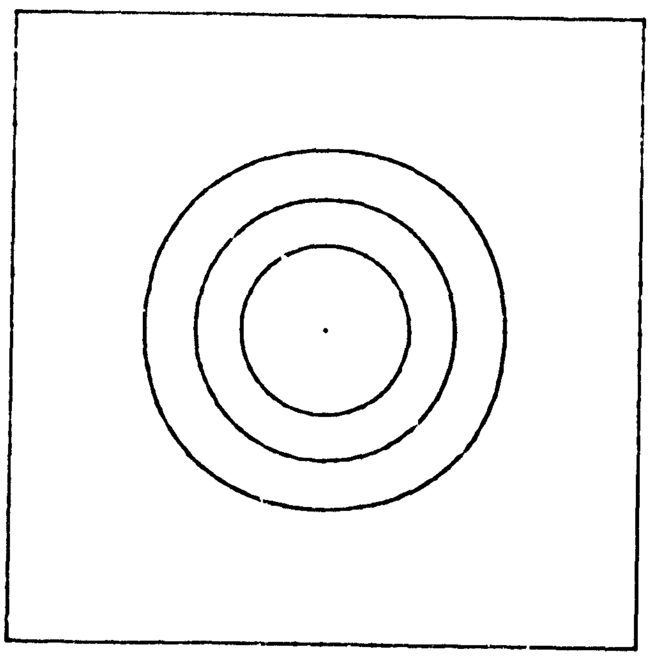

FIGURE 2. LOW PASS FILTER
STAGE = 1 RC = Ø.3

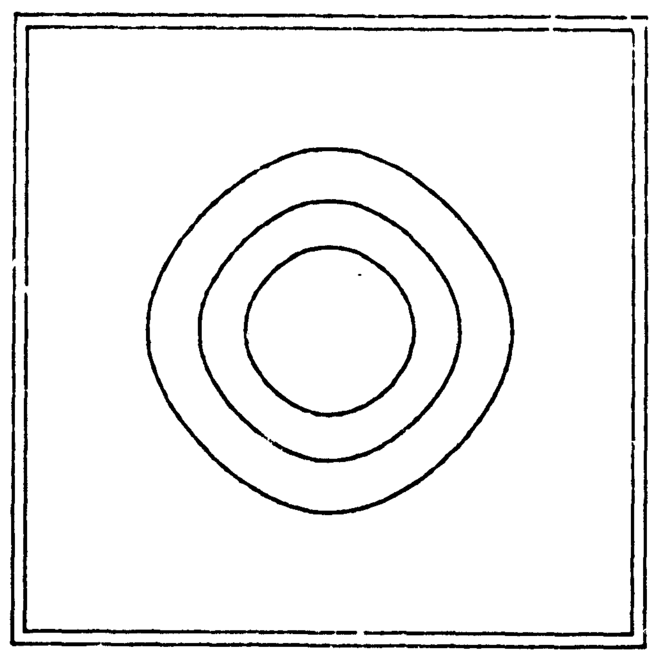

FIGURE 4. HIGH BOOST FILTER
BOOST FACTOR = 25.6 RC = Ø.5

SIMULTANEOUS LINEAR ALGEBRAIC EQUATION FORMULATIONS OF 2D FILTERS

D. E. OLSON, W. E. ALEXANDER, E. E. SHERROD
Elec. Eng. Dept., NC A&T SU, Greensboro, NC

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:

$$b_{ij}\, g_{p+i,q+j} = a_{ij}\, f_{p+i,q+j} \tag{1}$$

where $1 \le p \le N$, $1 \le q \le M$, $-m \le i \le m$, $-m \le j \le m$; 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:

$$B^{kl}_{pq}\, g_{kl} = A^{kl}_{pq}\, f_{kl} \tag{2}$$

where $1 \le k \le N$, $1 \le l \le M$, and the non-zero components of the coefficient tensors given by $A^{kl}_{pq} = a_{k-p,l-q}$; and $B^{kl}_{pq} = b_{k-p,l-q}$; for $-m \le k-p \le m$, $-m \le l-q \le m$.

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

$$g_{uv} = C^{pq}_{uv}\, A^{kl}_{pq}\, f_{kl} \tag{3}$$

Tensor equation (2) can also be interpreted as a matrix equation with the A^{kl}_{pq}, B^{kl}_{pq} taken as NMxNM dimensional coefficient matrices with row index "pq", column index "kl"; and g_{kl}, f_{kl} taken as column matrices.

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

$$LGR + \sum_{k=-m,k\ne 0}^{m} S_k G\, T_k = c\, PFQ + c \sum_{k=-m,k\ne 0}^{m} S_k F\, U_k \tag{4}$$

where $c = a_{00}/b_{00}$, the matrices $G = [g_{pq}]$, $F = [f_{pq}]$; and the non-zero components of the coefficient matrices L, R, P, Q, S_k, T_k and U_k given by:

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

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

$$T_{kpq} = b_{k,p-q}/b_{00} - b_{k,0}b_{0,p-q}/b^2_{00};\quad U_{kpq} = a_{k,p-q}/a_{00} - a_{k,0}a_{0,p-q}/a^2_{00}.$$

(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.