NASA Technical Memorandum 104612

FORTRAN Programs to Process Magsat Data for Lithospheric, External Field, and Residual Core Components

Douglas E. Alsdorf and Ralph R.B. von Frese
The Ohio State University
Columbus, Ohio

Geodynamics Branch
Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

1994

This publication is available from the NASA Center for AeroSpace Information,
800 Elkridge Landing Road, Linthicum Heights, MD 21090-2934, (301) 621-0390.

FORTRAN PROGRAMS TO PROCESS MAGSAT DATA FOR LITHOSPHERIC, EXTERNAL FIELD, AND RESIDUAL CORE COMPONENTS

by

Douglas E. Alsdorf and Ralph R.B. von Frese

ABSTRACT

The FORTRAN programs supplied in this document provide a complete processing package for statistically extracting residual core, external field and lithospheric components in Magsat observations. The data reduction method consists of two stages involving pass-to-pass and gridded map comparisons. To process the individual passes: 1) orbits are separated into dawn and dusk local times and by altitude, 2) passes are selected based on the variance of the magnetic field observations after a least-squares fit of the core field is removed from each pass over the study area, and 3) spatially adjacent passes are processed with a Fourier correlation coefficient filter to separate coherent and non-coherent features between neighboring tracks. In the second stage of map processing: 1) data from the passes are normalized to a common altitude and gridded into dawn and dusk maps with least squares collocation, 2) dawn and dusk maps are correlated with a Fourier correlation coefficient filter to separate coherent and non-coherent features; the coherent features are averaged to produce a total field grid, 3) total field grids from all altitudes are continued to a common altitude, correlation filtered for coherent anomaly features, and subsequently averaged to produce the final total field grid for the study region, and 4) the total field map is differentially reduced to the pole. Source code which provides standard statistical information is also supplied to quantify the performance of the data reduction procedures.

CONTENTS

I. INTRODUCTION

The National Aeronautics and Space Administration (NASA) magnetic field satellite (Magsat) has provided a global data set of geomagnetic field observations. Data retrieved from the satellite have been reviewed by NASA and made available for scientific investigations as the Chronicle and Investigator-B data sets (Langel et al., 1981). Both data sets have been used to evaluate magnetospheric effects, to define the core field and to determine magnetic anomalies at satellite altitudes associated with geologic features. The documentation and FORTRAN source code supplied in this technical memorandum describe a step by step method for processing the Investigator-B data set. The processing helps to define the magnetic anomaly field from lithospheric sources, the influence of external fields, and possible residual core field effects which are not included in current models (e.g., Alsdorf 1991).

The FORTRAN source code has been developed for processing the Investigator-B tapes. However, with adaptations the code could also be applied to the Chronicle series tapes or to other geophysical data sets (e.g., the magnetic field information from the POGO satellites). The code was developed in a UNIX-based environment on color graphics workstations such that compiled versions require user input at the terminal. Default values are noted for all user input variables, thereby facilitating the interactive nature of the processing. Graphics programs are not supplied in this document; however, the gridded data produced by the processing is formatted for standard contouring packages.

The data reduction process was broken into several steps so that modifications to the code could be more easily applied. Also, output from each step can be investigated for refinement purposes by changing the values supplied by the user. The flow chart in Figure 1 outlines each step of the data reduction process and should be referred to for the filenames described in this document. The first stage of processing operates on the individual passes while the second stage processes gridded forms of the data. For study areas covering about one fifth of the globe, it takes about two or three hours of operator time on a RISC-based computer to generate differentially reduced-to-the-pole grids from the orbital data contained on the Investigator-B tapes. Subsequent runs through the data to adjust the variables will take less time because several of the programs need to be run only once for a study area.

A full consideration of the theoretical details of these processing procedures is beyond the scope of this report. These details are found in the references cited in the software and this report. Additional discussions and explanations can be found in Alsdorf et al. (1991; 1992).

There are six programs (Appendix B) used to process the satellite magnetic data from the individual orbits and five programs (Appendix C) that refine the data in grid form. Several auxiliary programs (Appendix D) are also provided to evaluate the output at different steps of the processing. Appendix A outlines the compile- and run-time considerations, disc storage allocation, and notes several improvements that could be made to the various programs. Chapter II describes the processing of the profiles while Chapter III details the processing of the grids.

II. PASS PROCESSING

The programs used in this section include: 1) **subcore**, 2) **reorder**, 3) **massage**, 4) **movetrunc**, 5) **fourier1d** and 6) **combine**. Appendix A should be reviewed for initial set up information before executing these programs. The following programs are presented in the appropriate running order.

II.A SUBCORE

After compiling each program and copying the data from the Investigator-B tapes to disc (Appendix A), the first program to run is **subcore**. This program reads the data in either sequential or direct access and writes to disc in direct access. Refer to the comment statements in the program when changing the code from the sequential access to the direct access driver. Each written record corresponds to an individual observation point as recorded by the satellite. For each record the first two values are integers (fixed point numbers) indicating the pass number and modified Julian day, respectively, and the remaining values are reals (floating point numbers) indicating location coordinates, core field values and vector magnetic field observations. NASA Technical Memorandum 82160 (Langel et al., 1981) should be consulted for a complete description of these variables. The order of the input files from the tape to disc transfer should be in the same time order as recorded by the satellite. This order will maintain the time orientation of the data which is convenient for subsequent processing. However, this rule is not absolutely necessary because program **reorder** can readjust the data to any required time or space orientation, as explained in Section II.B.

A review of the program description and comment statements in the code provides a complete assessment of the adjustments that are made to the data in **subcore**. The major functions of **subcore** are: 1) acquire only the data within user-defined latitude and longitude limits, 2) separate an individual orbit into its dawn and dusk components, and 3) calculate the core field value at each observation point. The spherical harmonic coefficients through degree and order 13 are used to model the core field and are presented in Appendix B.

After the core field value is subtracted from the observed value a series of values is written to disc. Not all of these values are necessary for future processing and some could be removed from the write statements if disc storage is limited. The subtraction of the core field from the magnetic field observations is not a profile-least-squares procedure (Alsdorf, 1991). The least-squares method of subtraction is performed in program **massage** (Section II.C). After **subcore** is complete, the data files from the tape to disc transfer can be removed from disc storage because the processing no longer accesses these files.

Separate files containing dawn passes and dusk passes are written by **subcore** to disc. These files are run separately through the remaining programs that process the profiles (Figure 1).

II.B REORDER

The major function of **reorder** is to rearrange the input file from a time to a spatial orientation. Ordering the passes by location in space is according either to the average longitude of a pass (the usual choice) or by the average elevation. Reordering the passes by time or pass number is also an option, however, it is seldom used. This program reads the direct access output from **subcore** and also writes out direct access files. This program requires twice the disc space as the size of the input file because it is necessary to create a working file which can be deleted after the run is complete. Once **reorder** has been completed, the 2-integer and 27-real output file from **subcore** can be deleted because the file is no longer necessary for processing.

Subcore and **reorder** complete the standard preliminary processing of Magsat data. The output from **reorder** should be saved, even after running further programs, because data parameters may need adjustments when refining the final output. These adjustments often include rerunning the programs described below.

II.C MASSAGE

As originally designed, **massage** developed a combination of local and regional models of the data in an attempt to remove external field effects by Fourier correlation coefficient filtering. This method encompasses the construction of a "guide function" which is an approximate representation of the influence of external fields in an individual pass. The guide function and the observations are then transformed by the Fourier program (Section II.E) and wavenumber components which correlate within a user defined range of correlation coefficients are cut from the observed data. Those components which correlate represent the effects of unwanted external fields and are therefore cut from the observed spectrum. However, after many investigations it was found that bandpass filtering could provide acceptable results with less computation and disc storage requirements than the guide function method. However, the options to construct the guide function are still included in **massage** for any research that may require a cubic spline fit to the data.

The major functions of **massage** are now usually limited to the following: 1) remove "spikes" from a profile and linearly interpolate all values at latitude intervals of 0.33 degrees, 2) calculate and subtract a least-squares profile fit of the core field values from the observed values, and 3) write out two corresponding files of an individual profile where one file contains the latitude, longitude and radius (e.g., dk.llr in Figure 1) while the other file holds the magnetic field value for each interpolated observation point (e.g., dk.var). Note that output files from **massage** are sequential access and considerably smaller than previous files because each profile is marked by only one header and either three or one variable(s) depending on file type.

II.D MOVETRUNC

This is the step where the dawn and dusk data sets are subdivided further into altitude bands (Alsdorf et al., 1992). The number and distribution of passes for each altitude band must be maintained to ensure a small distance between adjacent passes as compared to the distance to the lithosphere. For example, over the south polar region there are over 2500 dawn and dusk passes available from Magsat. After separation into four distinct altitude bands, there are over 500 passes for each local time at each altitude (Alsdorf et al., 1992), thus maintaining the density of observations for each band. Non-polar regions will have fewer passes because of the orientation of the satellite, therefore, these regions will probably have less than four altitude bands. For the purposes of this report we will consider only two bands of altitude separation; lower and upper altitude passes. Therefore, after running **movetrunc**, four sets of data will exist including upper and lower altitude dawn data and upper and lower altitude dusk data.

Before **movetrunc** is run, the program **check** as described in Appendix D should be executed to find passes with unacceptably high variances. Also, the output file of "averaged sorted variables" from **reorder** should be copied and edited for profiles that are above or below the median elevation. Appendix A reviews the UNIX commands which can be used to create the files of pass numbers that separate the passes into upper and lower altitude sets.

Movetrunc reads the file of latitudes, longitudes and radii (e.g., dk.llr) as well as the corresponding file of magnetic field values (e.g., dk.var) produced from running **massage**. After removing unwanted passes, adjacent profiles are truncated to similar lengths according to the latitude value of each observation along a pass. Figure 2 schematically shows how the passes are truncated. Note that pass 6 is duplicated; one version (6w) is truncated to match the length which overlaps with pass 5 and the other version (6e) is truncated so that it has the same overlapping section as pass 7. This duplication and truncation procedure is repeated for every pass. Reviewing Figures 1 and 2, pass 6 is comparable to dk.llr and dk.var, 6w is similar to dk.low.llr.y and dk.low.y, and 6e is like dk.low.llr.x and dk.low.x. Therefore, two sets of files, offset by one pass, are written to disc so that program **fourier1d** (Section II.E) can correlate the immediately adjacent profiles.

II.E FOURIER1D

This program performs the fast Fourier transform (FFT) and inverse FFT as well as bandwidth and/or correlation coefficient filtering. Complex number notation is used to denote the wavenumber components in the memory of the computer. Options are provided for folding out the edges of the data, smoothing the edges to zero to minimize Gibbs energy effects, and centering the data within an array of zeros. Note that the subroutines of this program are one dimensional versions of those used in **fourier2d** (Section III.B)

The following example demonstrates the size of arrays to use, the percent of data to be folded out and the percent of data to be smoothed to zero in any application. First assume that the study area has a latitude range of 40 degrees. This range results in 121 data points:

$$121 \text{ data points} = (40 \text{ degrees})/(0.33 \text{ degrees per data point}) \quad (1)$$

The size of FFT array to use should then be set to a power of two greater than 121 data points (128 or 256). In this case 128 allows for minimal folding and smoothing, so that better performance is obtained when 256 is used. The percentage of data to be folded out can be then calculated by

$$(2 * 121 * X\%) + 121 < 256 \quad (2)$$

In using equation 2, the X percentage chosen must satisfy the less than sign. If for example we chose 10 percent, then for 121 data points, 12 values at each end of a pass will be folded out and added to the beginning or end of the profile, so that there are 145 data points (145=12+121+12). The following sequence illustrates the mirror folding of data points at each end of the profile obtained by **fourier1d**.

12, 11, ... 2, 1, 1, 2, ... 11, 12, 13, ... 109, 110, 111, ... 120, 121, 121, 120, ... 111, 110

folded data ---|------------------ original data ------------------|---------- folded data

The percent of data to be smoothed to zero must satisfy

$$(145 * Y\%) < 12 \quad (3)$$

where the Y percentage is chosen so that only the data folded out are smoothed and not the actual data. In this case we might chose a Y percent of 8% which smooths 11 values of each edge of 145 data points (11=0.08 * 145). Finally the program will center the 145 data points within the FFT array by adding 55 zeros to the beginning and 56 zeros to the end of the 145 data points (55,56={256-145} * 0.5). When the final data set is written to file after inverse transforming, only the original 121 data point positions are used.

This is the first application of the **fourier1d** program and the user should consider bandpass and correlation coefficient filtering of the data at this time.

II.F COMBINE

Combine is run twice when processing the data. In this first application of the program, the two output files of latitudes, longitudes and radii (e.g., dk.low.llr.x and dk.low.llr.y) from **movetrunc** were identical except for an offset of one pass between the files. After **fourier1d** is applied in II.E, **combine** is used to find the same pass in the two files and truncate both versions of the pass to a similar length. This application of **combine** as illustrated in Figure 2 is analogous to truncating 6w and 6e so that both of these versions of pass 6 are of similar overlapping lengths. Therefore, it is important to input the files in the correct order. However, **combine** will check to see that the user has input the files correctly and if not, will stop execution of the program and issue a warning to the screen.

II.G FOURIER1D

This is the second use of **fourier1d** on the profiles. Here, the passes should only be correlation filtered for similar wavelengths. Bandpass filtering is usually not performed.

II.H COMBINE

This is the second use of **combine** on the profiles. The output of **combine** is chosen as one file of latitudes, longitudes, radii and anomaly values (e.g., dk.low.llra) which will be input to **collocation** as described in section III.A for gridding. This single output file is written to disc in formatted-ASCII, sequential-access so that the file may be easily transferred from a workstation to a supercomputer.

This concludes the data processing as applied to passes.

III. MAP PROCESSING

Programs applied for map processing include: 1) **collocation**, 2) **fourier2d**, 3) **avgdifres**, 4) **sqrmap** and 5) **inversion**. Before executing these programs, Appendix A should be reviewed for initial set up information.

Before the programs in this chapter are run, the output file of latitudes, longitudes, radii and anomalies from **combine** (Section II.H) should be transferred from the workstation platform to a supercomputer. This transfer is not absolutely necessary, however the computing speed of a supercomputer facilitates faster processing of matrix inversions and large two-dimensional Fourier transforms.

The four sets of profile data (i.e., upper and lower altitude dawn and dusk orbits) were each processed independently as described in Chapter II. When processing the grids in this chapter, the lower altitude dawn and dusk grids are compared and the upper altitude dawn and dusk maps are compared. At the end of the processing, the lower and upper altitude total field maps can be continued to the same elevation and subsequently correlated and averaged (Alsdorf et al., 1992). The following discussion only addresses a single altitude set of dawn and dusk data, although the other altitude data will also be processed in the same manner to test for lithospheric anomaly features sets.

The program **statmat** included in Appendix D can be run using any combination of the following grids as input. **Statmat** determines a variety of standard statistics necessary for interpretation of the magnetic anomalies and the quality of processing.

The following programs are presented in correct running order.

III.A COLLOCATION

Collocation reads the ASCII free format file from **combine** (e.g., dk.low.llra), which includes arbitrarily distributed data points throughout the study region, and calculates node values at regular intervals over a grid at constant altitude (Goyal et al., 1991; Goyal, 1986). The output grid file from **collocation** is formatted ASCII, where the first row of values in the file is along the southern most latitude from west to east. Coordinates for the first data point in the array are the western most longitude and the southern most latitude. The next data point is just one grid interval to the east of the first data point and along the latitude coordinate. The remaining data points in the row are successively one grid interval east of the preceding point. The next row of values follows the same west to east orientation as the first row, however this row is one grid interval to the north of the first row. The remaining rows then fill the grid successively from south to north and west to east. All programs that work with the grids keep this same orientation.

The user supplied covariance matrix (Appendix C) that is used in **collocation** has been found to produce acceptable magnetic anomalies (Goyal et al., 1991; Goyal, 1986). The covariance matrix provides a function that is used to calculate weights based on distances between grid nodes and observation points.

Both a dawn grid and a dusk grid at the same altitude are produced by separate runs of **collocation**. These grids correspond to the respective sets of passes from the profile processing. The elevation of the grids should be the same and is commonly chosen as the average elevation of all observations in the dawn and dusk data sets. **Collocation** can be used to predict values at grid points which are separated by distances of equal degrees or equal lengths. For example, when working over the polar regions, the FFT algorithms and filtering routines work best with grids of equal areas denoted by the grid coordinates of equal length separations. The program comment statements should be reviewed for appropriate input parameters when choosing between the above grid coordinate options.

III.B FOURIER2D

The forward and inverse FFT subroutines of this program are the two dimensional versions of those in **fourier1d** (Section II.E). **Fourier2d** offers several data processing filters including both the bandpass and correlation coefficient routines. Additional filters are also included which are not typically used in the Magsat data processing; however, these routines are made available for expanded processing efforts. The directional filtering routine fashions a wedge-shaped filter to pass/reject directional trends of data features, whereas the remaining routines perform flat-earth upward and downward continuation, flat-earth reduction of magnetic total field anomalies to the pole, obtain flat-earth anomaly derivatives, and adjust the phase and amplitude of the individual wavenumber components. For spherical-earth applications, both the continuation and reduction to pole of data is more suited and better constrained by the matrix inversion methods of program **inversion** (Section III.I) and, therefore, Section III.I should be reviewed for these data processing methods.

The main calling routine of **fourier2d** also differs from **fourier1d** in that it does not loop through successive profiles or maps and it allows for multiple calls to the filters in any user-defined order. The comment statements of **fourier2d** should be reviewed for the correct user input values which control the order and number of times a particular filter is called.

The dawn grid is correlated with the dusk grid such that the correlation coefficient cutoffs are set to pass the coherent and consistent anomalies. The size of the FFT array, the percent of data to be folded out and the percent of data to be smoothed are calculated in similar fashion to those of the profiles as described in Section II.E. However, because these values are applied to the columns as well as to the rows of the matrix, the various percentages will be determined by both the number of rows and columns. The input Y% should be chosen so as not to smooth actual values within the array.

This is the first use of **fourier2d** as applied to the grids and the user should only choose to apply the coefficient filter to the two grids. Alsdorf et al. (1992) review appropriate correlation cutoff values for the south polar region where the auroral external field influences are significant; other regions may require different values depending on the effects of external fields in those areas.

III.C AUXILIARY MAP PROCESSING

The grid processing steps of this section are not necessary for standard map reductions; however, these steps are presented for completeness. The processing of this section removes possible external field influences manifested as coherent differences between the correlated dawn and dusk maps. Also, the standard deviations (ie. energy levels) of the correlated dawn and dusk maps are adjusted to nearly the same level.

III.C.1 AVGDIFRES

This program can be run up to three times during the data processing; once in this auxiliary section and possibly twice in standard processing. In this initial application, **avgdifres** is used to calculate the difference between the correlated dusk and dawn grids (e.g., low.diff1).

III.C.2 FOURIER2D

This is the second application of **fourier2d** on the grids. Here, the difference grid from **avgdifres** (e.g., low.diffl) is smoothed with a high-cut filter so that a long wavelength model of the possible influence of external fields is produced.

III.C.3 SQRMAP

At this point, both the difference grid and its low-pass filtered version should be visually inspected to determine if the differences can serve as a model of the expected effects of external fields for the study region. If so, then **sqrmap** subtracts the filtered difference matrix from the correlated dusk or dawn grid. Before the subtraction, the difference matrix is least- squares adjusted to more closely match the correlated dusk or dawn grid under consideration.

This concludes the auxiliary processing section.

III.D AVGDIFRES

This is the second application of **avgdifres** to the grids. As applied here, **avgdifres** finds the average and the difference of the correlated dusk and dawn grids produced from either **fourier2d** in III.B or **sqrmap** in III.C.3.

III.E FOURIER2D

This is the final application of **fourier2d** to the grids. With this application the averaged grid from **avgdifres** (III.D) is high-cut filtered to remove wavelengths shorter than the elevation of the data set. Magnetic anomaly wavelengths smaller than the magnitude of the elevation of the grid are not apparent at satellite altitudes.

The high-cut filtered output map from this execution of **fourier2d** represents the total field magnetic anomaly map at the particular altitude which is being considered.

III.F MODELING OF THE MAGNETIC ANOMALIES

The processing of the previous sections is repeated over each altitude band (e.g. both the lower and upper altitudes) producing a total field grid for each altitude. As discussed in this section, each total field grid is individually continued to a common altitude using the inversion program, then all continued grids are averaged to produce the final total field grid for the study region. This total field grid can then be differentially-reduced-to-the-pole (DRTP) for geologic interpretations and comparisons with gravitational anomalies.

III.F.1 AVGDIFRES

This is the final application of **avgdifres** to the grids. Here the smoothed grid from **fourier2d** (III.E) is resampled so that the output matrix can be inverted within the interactive memory allocation of a supercomputer. Generally, resampling should occur at a grid interval less than the high-cut wavelength used in **fourier2d**. This step is not necessary if precautions are taken as described below in section III.F.2.

III.F.2 INVERSION

This modeling program finds the effective susceptibilities which correspond to the total field grid supplied by **avgdifres** (III.F.1) or **fourier2d** (III.E) (von Frese et. al., 1981; 1988). These susceptibilities are then subjected to a core field model to produce the total field anomalies at a user defined altitude, or the susceptibilities can be subjected to a radial field of constant intensity to model the DRTP anomalies. These procedures are equivalent to spherical-earth continuation of the Magsat data. To find the susceptibilities, a core field model expanded through degree and order 13 which can be updated to the mission lifetime is necessary (e.g., Appendix B). If the subject area is large and results in more unknowns than the memory allocation of an interactive session on a supercomputer allows, then one of the following can be applied: 1) a boot strap inversion (von Frese et al., 1988), 2) the matrix inversion routines can be modified to write and read from disc rather than memory, or 3) use batch submission so that the code will be executed during a period of reduced user demand (Appendix A). Documentation in **inversion** describes cpu storage and time requirements in terms of the number of unknowns for any inversion.

After all of the total field grids are continued to an average altitude, the continued grids can be correlation filtered using **fourier2d** and subsequently averaged together to produce the final total field grid for the study area. The algorithms of the averaging code are rather straightforward, and we do not present them in this document. However, the code is available via email as outlined in Appendix A. The continued total field grids can be compared to test the self-consistency of anomaly features. Comparisons are facilitated by differencing the grids and statistical analyses.

This concludes the data processing as applied to the grids.

IV. CONCLUSION

The FORTRAN programs supplied in this document provide processing capabilities for investigating lithospheric, external field, and residual core components in the Magsat data. For extracting lithospheric anomalies, the data processing begins with reading the NASA Investigator-B files and finishes with a differentially-reduced-to-the-pole magnetic anomaly map of the study region.

V. ACKNOWLEDGEMENTS

Programming advise as well as several elements of the software were provided by Drs. Dhananjay Ravat, Gary P. Murdock and Daniel R.H. O'Connell. We also thank Dr. Saul A. Teukolsky for permission to use selected FORTRAN routines from Numerical Recipes. Subroutines **spline**, **splint** and **sort** in programs **reorder**, **massage** and **check** are based on routines in Numerical Recipes in Fortran: The Art of Scientific Computing, published by Cambridge University Press and are used by permission. This memorandum originated as Geological Sciences Computing and Graphics Laboratory Report #1 of the Department of Geological Sciences at the Ohio State University. Elements of the software were developed with funding provided by the NASA Center for Mapping (NAGW-973), Amoco, Arco, Exxon, Texaco and Unocal, and the support of the Ohio Supercomputer Center. The memorandum was completed as part of a NASA Summer research fellowship to DEA at the Goddard Space Flight Center with funding from Hughes-STX.

VI. REFERENCES CITED

Alsdorf, D.E., R.R.B. von Frese, J. Arkani-Hamed and H.C. Noltimier, Separation of lithospheric, external, and core components of the south polar geomagnetic field at satellite altitudes, submit. J. Geophys. Res. in February, 1992.

Alsdorf, D.E., FORTRAN Programs to Process Magsat Data, Geological Sciences Computing & Graphics Laboratory Rept. #1, Dept. of Geological Sciences, The Ohio State University, 1992.

Alsdorf, D.E., Statistical Processing of Magsat Data for Magnetic Anomalies of the Lithosphere, unpubl. M.Sc. Thesis, Dept. of Geological Sciences, The Ohio State University, Columbus, OH, 1991.

Goyal, H.K., Statistical Prediction of Satellite Magnetic Anomalies for Geologic Interpretation, unpubl. M.Sc. Thesis, Dept. of Geological Sciences, The Ohio State University, Columbus, OH, 1986.

Goyal, H.K., R.R.B. von Frese, W.J. Hinze and D.N. Ravat, Statistical prediction of satellite magnetic anomalies, Geophys. J. Int., 102, 101-111, 1990.

Langel, R., J. Berbert, T. Jennings and R. Horner, Magsat Data Processing: A Report for Investigators, NASA Technical Memorandum 82160, Goddard Space Flight Center, Greenbelt, Maryland, 1981.

Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Numerical Recipes, The Art of Scientific computing (FORTRAN version), Cambridge University Press, Cambridge, 1989.

von Frese, R.R.B., W.J. Hinze and L.W. Braile, Spherical earth gravity and magnetic anomaly analysis by equivalent point source inversion, Earth and Planet. Sci. Lett., 53, 69-83, 1981.

von Frese, R.R.B., D.N. Ravat, W.J. Hinze and C.A. McGue, Improved inversion of geopotential field anomalies for lithospheric investigations, Geophysics, 53-3, 375-385, 1988.

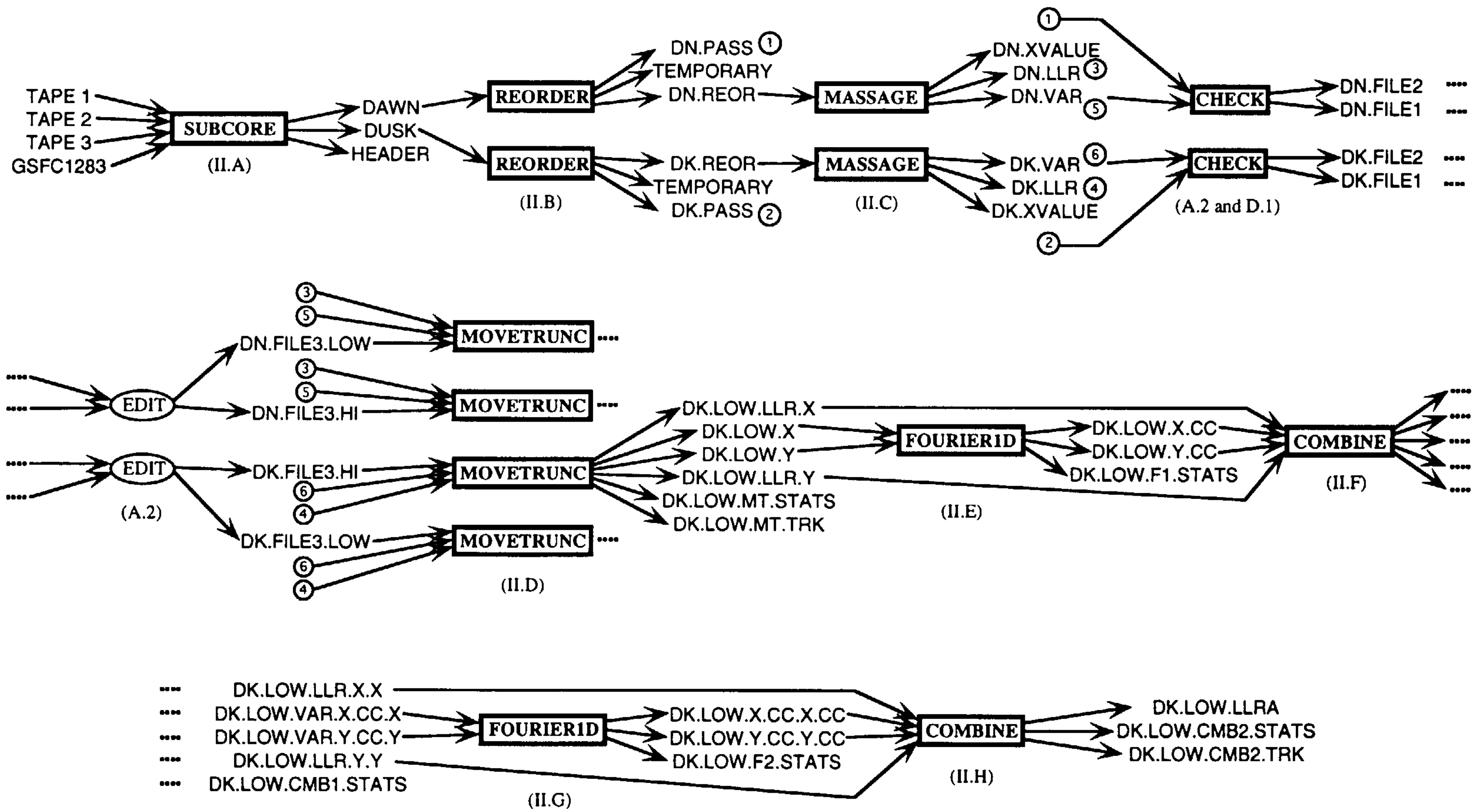

Figure 1: Processing flow chart. Program names are given in boxes and suggested file names follow the arrows. All names are used in the manuscript and values in parantheses are indexed to the appropriate chapter or appendix.

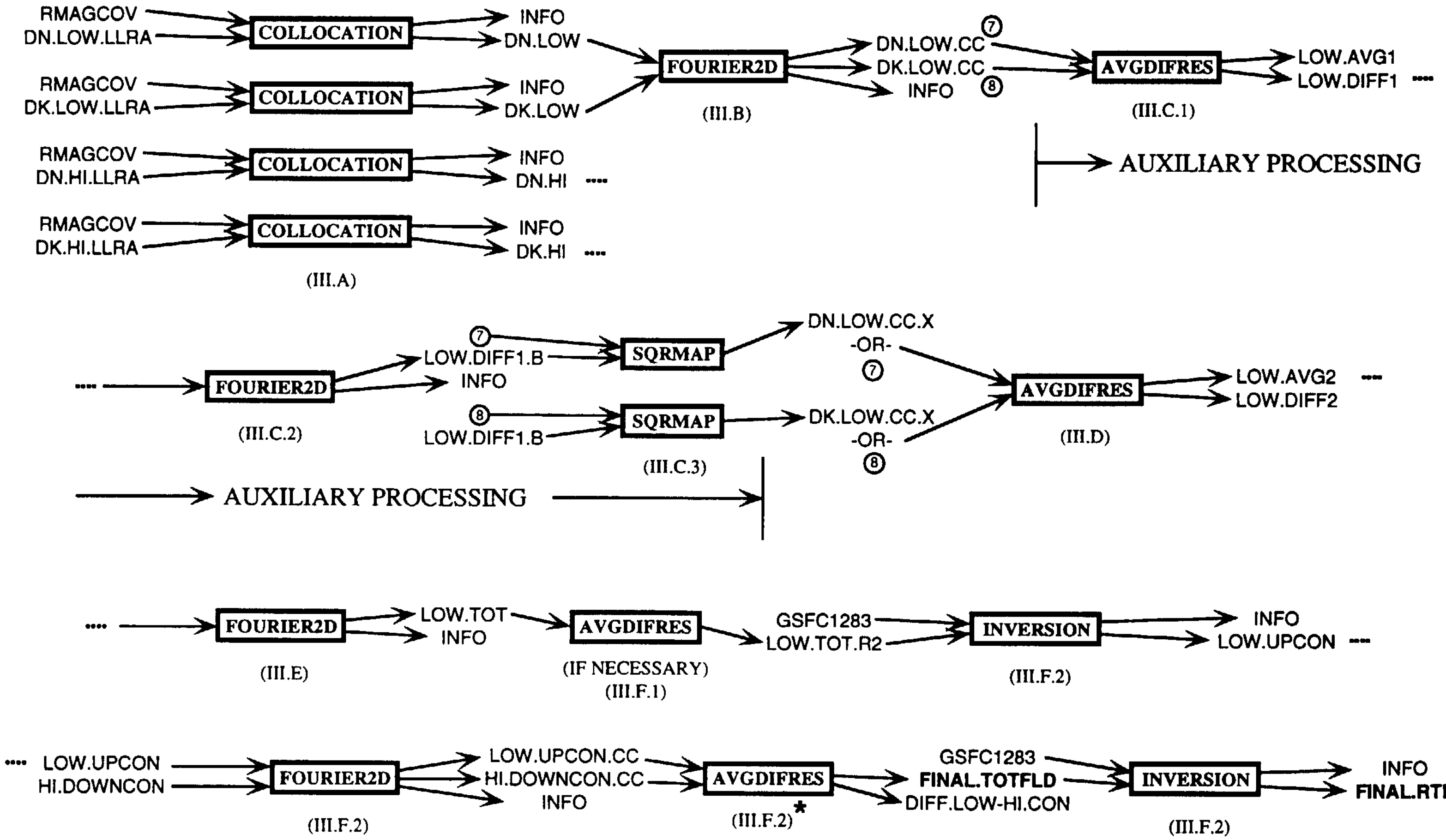

Figure 1 (continued from page 17)

* If more than two altitude bands are used, then **avgdifres** can not be used at this point. An auxiliary program to average all correlated continuations is available via email (as described in Appendix A).

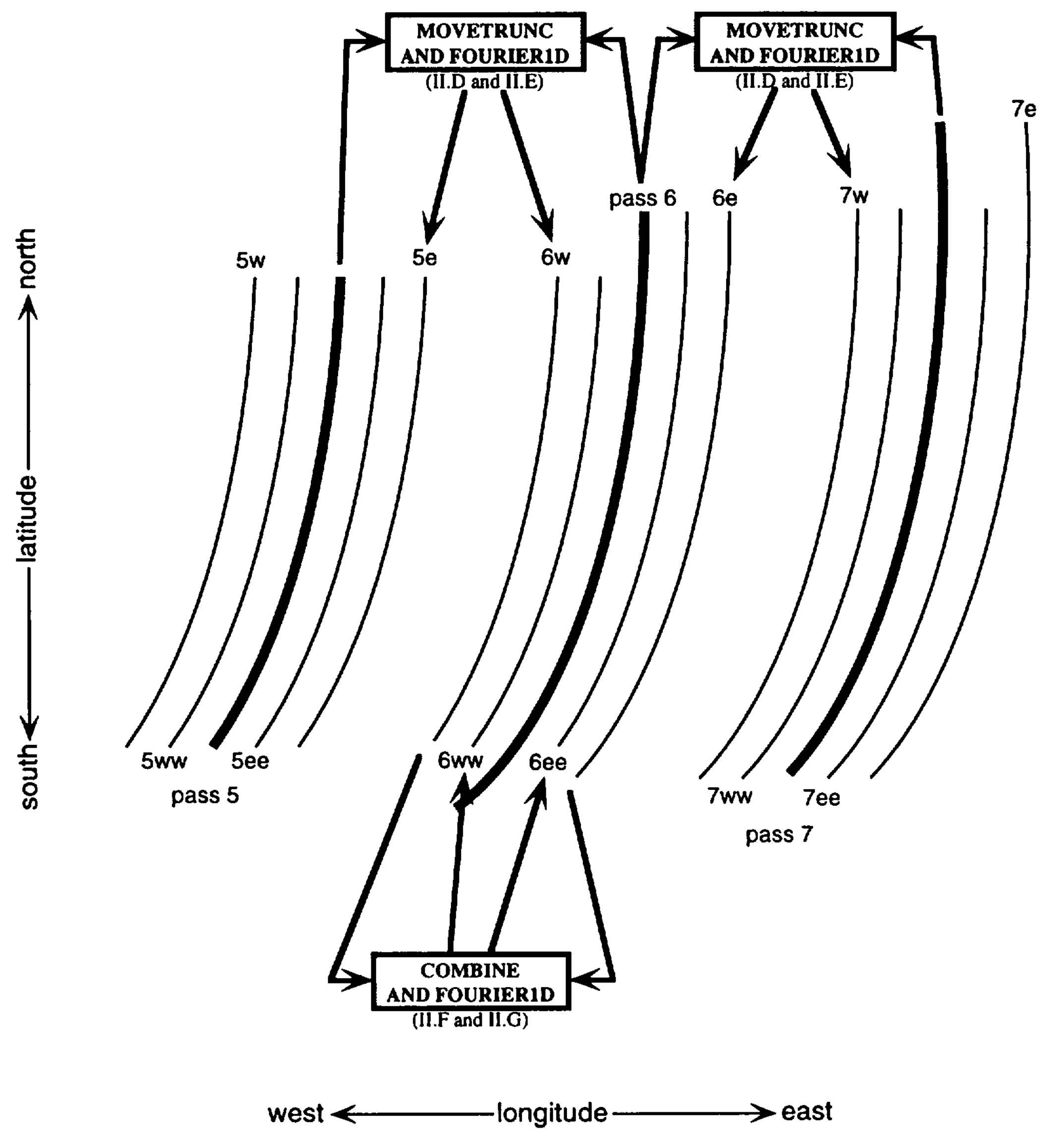

Figure 2 Pass-to-pass processing schematic showing the application of the correlation filter of program **fourier1d** and the truncation of passes by programs **movetrunc** and **combine**. Pass labeling convention follows from the text and program labels are from Figure 1.

APPENDIX A: DATA EDITING AND COMPUTING REQUIREMENTS

This section describes the computing environment necessary to compile and execute the FORTRAN source code as well as disc storage estimates for the files. Additional comments address UNIX commands used to create various input files and possible improvements to the code to increase speed and decrease total file storage requirements.

A.1 COMPUTING ENVIRONMENT

The source code was written, compiled and executed on DEC 3100 color graphics workstations with the Ultrix operating system. Source code in Appendix C was also compiled and executed on the Ohio Supercomputer Center's CRAY Y-MP8 which operates with UNICOS. Other computing systems with FORTRAN 77 compilers should compile, link and execute the code with little or no modifications. Programs **subcore.f**, **reorder.f** and **massage.f** use direct access for file reads and writes and need to be modified for the specific operating system. The comment statements in these programs should be reviewed for additional information. After the code has been transferred to a FORTRAN source code directory, the individual programs should be compiled with the following Ultrix command:

Ultrix prompt: f77 -static programname.f

or with the following UNICOS command:

UNICOS prompt: cf77 -Zp -Wf"-a static" programname.f

The -Zp option allows for optimal autotasking and vectorization and -static permits the local variables to be statically allocated. The executable file, a.out created by running the FORTRAN compilier command can be moved to the users bin directory and given the same name as the programname without the .f extension. When executing any of the programs, the user is prompted at the screen (standard input/output device) for filenames and parameters for variables. The variables all have default values listed inside parentheses and the user can type these values as desired. Filenames are suggested in the flow chart of Figure 1.

When running the code on the CRAY, it is convenient to use the batch submission procedures for the inversion code (section III.F.2) because it may require more memory than is allocated under an interactive session. The following is a typical example of the batch submission method:

UNICOS prompt: qsub -lM 10Mw -lT 3600 shellfile

The -lM 10Mw allocates 10 megawords (80 megabytes) of memory and -lT 3600 provides 60 minutes of cpu time to run the code in the shell file. Consult the CRAY manual pages for more information.

The total size of the three Investigator-B tapes is around 300 Megabytes (Mb) on a DEC 3100, however other machines may double the size depending on the default number of bytes used to define floating and fixed point numbers. For study areas that constitute about one fifth of the globe, disc storage requirements to run **subcore** range between 45 and 70 Mb depending on global location. To run **reorder**, 90 to 140 Mb are necessary. Between 3 and 6 Mb are needed to run any of the remaining programs in Appendix B. The dawn and dusk grids are generally less than 400 Kilobytes (Kb).

A.2 HELPFUL UNIX COMMANDS

This section describes how to produce files of pass numbers which remove the high variance passes as well as subdivide the passes into altitude sets (e.g. lower and upper altitudes). As defined here, file1 contains those passes with a variance above a threshold; file2.low and file2.hi hold pass numbers of either the lower or upper (respectively) altitude pass numbers; and file3.low and file3.hi include both the large variance pass numbers and the lower or upper altitude pass numbers. The following sequence of commands must be executed once for the dawn data and once again for the dusk data.

As discussed in section II.D, **check** is run immediately before **movetrunc**. Once a maximum variance cutoff has been established, **check** should be run in a UNIX script shell as follows:

UNIX prompt> script file1

script prompt> check

script prompt> 1 (type 1 and hit the return) script prompt> output file from **massage** (e.g., dk.var)

script prompt> 1

script prompt> 200.0   200.0   200.0   -200.0 500.0

(these are suggested values)

script prompt> yes

script prompt> output file from **reorder** (e.g., dk.pass)

script prompt> a new file (e.g., file2)

Now several hundred pass numbers will be written to the screen and to file1. After the program is finished, exit out of the script shell and use a text editor to access file1.

script prompt> exit

UNIX prompt> vi file1 (or use any other editor)

Once in the editor, remove every line except those with pass numbers on them so that the final version of file1 resembles the following line.

1074   129   75.982   -0.023   -29.244   18.222   1087.460

File1 now contains the pass numbers of those passes which have a variance above the user-defined maximum (in this case 500.0 nT^2).

Next, make two copies of file2 so that the pass numbers can be separated into low and high altitude sets.

UNIX prompt> cp file2 file2.low

UNIX prompt> cp file2 file2.hi

Edit file2.low with a text editor, removing all lines where the average elevation is above the median elevation. The median elevation occurs at the mid-line in the file (e.g., if there are 1400 lines in file2, then the median elevation occurs on line number 700). Similarly, edit file2.hi removing all lines where the average elevation is below the median elevation. Finally, combine the edited file2.low and file2.hi with file1 as follows:

UNIX prompt> cat file2.low file1 > file3.low

UNIX prompt> cat file2.hi file1 > file3.hi

While processing the lower altitude passes, file3.hi is input to **movetrunc** when the program asks for "input file of pass numbers not wanted". Conversely, file3.low is input to **movetrunc** when processing the upper altitude passes.

A.3 SUGGESTED IMPROVEMENTS

Several improvements to the code to lower run time and decrease file sizes could be made. The following improvements concentrate on the programs that process the individual profiles as described in Appendix B:

1. **subcore**: Arrays that are in part named "data" could be combined so that the same array is passed to each subroutine.

2. **reorder**: A better method of finding the average longitude of short passes may decrease total run time.

3. **movetrunc** and **combine**:

a) These two programs are very similar and with some modifications the programs could be combined.

b) Because the output lat-long-radii files are similar, only one file containing flags indicating the index locations where the passes overlap is necessary for output.

The following improvements concentrate on the programs in Appendix C that process the gridded maps.

1. **collocation**:

a) Invert only the symmetric half of the COVM array.

b) Use a faster sorting routine for finding the closest points to a grid node location.

2. **inversion**: For arrays larger than the allocated machine memory, an option should be inserted that uses disc space for the matrix inversion.

As a final note, the source code is intended as a framework that allows step-wise processing of the Magsat data. This framework is open for improvements which are heartily encouraged. For copies of the code in this technical memorandum as well as additional auxiliary programs not presented in this document, send an email request to:

alsdorf@geo1s.mps.ohio-state.edu -or- vonfrese@geo1s.mps.ohio-state.edu

Comments, criticisms, suggestions for code improvements, as well as requests for code updates should also be directed to the above email addresses.

APPENDIX B: PROFILE PROCESSING

PROGRAMS

subcore.f

reorder.f

massage.f

movetrunc.f

fourier1d.f

combine.f

DATA FILE

gsfc1283

```
      program subcore
      real*4 rbuff(3024/4),minlat,maxlat,minlon,maxlon,
     >        seconds(3000),rhead(2228/4)
      integer*4 ibuff(3024/4),flag1,flag2,headcnt,recnum,
     >           outnum,dnnum,dknum,
     >           dndkpass,datacnt,stop10,stop11,
     >           head10,data10,head11,data11,head12,data12,totrecord,
     >           dnrec,dkrec,nowant(50),ihead(2228/4)
      character*80 filename
      character*4 cbuff(3024/4),chead(2228/4)
      character*60 aaid
      equivalence (ibuff,rbuff),(cbuff,rbuff),(flag1,ibuff(1)),
     >             (flag2,ibuff(2)), (ihead,rhead), (flag1,ihead(1)),
     >             (flag2,ihead(2)), (chead,rhead)
      common /latlon/ minlat,maxlat,minlon,maxlon
      common /dndkdat/ dndata(1500,26),dkdata(1500,26),
     >                 idndata(1500,2),idkdata(1500,2)
      common /coeff/ gg(50,50),ggt(50,50),ggtt(50,50),jnum,knum,
     >                 ttzero,aaid,mmaxn,ttemp
      common /mainfld/ fld(1500,8),dawne,dawni,duske,duski,dndkpass
      common /thatsit/ outdawn(1500,27),outdusk(1500,27)
      COMMON /NASA/ TG(50,50)
      COMMON /FLDCOM/ ST,CT,SPH,CPH,R,NMAX,BT,BP,BR,B
      COMMON /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)
      common data(3000,26),idata(3000,2)
c
c------------------------------------ program description
c
c     subcore reads the 3 NASA INVESTIGATOR-B tapes from disc and obtains
c     the data for the user defined area.  the program performs the following
c     corrections to the data:  1) reorders the dataset from NASA's column
c     arrays to user manageable row arrays  2) removes all values at a
c     single sampling point if one of those values is flagged by NASA with
c     9999  3) selects the data for the user defined area  4) separates
c     the area into dawn and dusk datasets  5) calculates the core-field
c     value for every data point along the dawn or dusk profile and saves
c     that value in an array  6) removes the core-field values from the
c     data point along a dawn or dusk profile  7) removes NASA's ring-
c     current correction  and  8) writes several values to the output
c     files - these values can be determined by looking at subroutine
c     corering.
c         NOTE: output unformatted files are direct access
c               input unformatted files are sequential access
c               output formatted files are sequential access
c         NOTE: use NASA Technical Memorandum No. 82160  for a complete
c               description of each variable
c
c      program date: 16 apr 91
c
c      updates:
c          4 jun 92, added sequential access driver
c          NOTE: the input files are sequential access in this section
c                to use direct access input files, then swap the
c                driver at the end of the code with the currently used
c                sequential access driver.
c          NOTE: if you are working on an ibm rs6000 then the record
c                length for direct access on files 20 and 21 is:
c                           recl=116
c                for input of direct access to files 10,11 and 12 then
c                remove the /4 from 3024/4.
c                if you are working on a dec3100:
c                           recl=29
c                and keep 3024/4.
c
      write (*,*) 'INPUT FIRST DATASET FROM TAPE TO DISC TRANSFER'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT SECOND DATASET FROM TAPE TO DISC TRANSFER'
      read (*,9990) filename
      open (11, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT THIRD DATASET FROM TAPE TO DISC TRANSFER'
      read (*,9990) filename
      open (12, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT FIELD MODEL SPHERICAL HARMONIC'
      write (*,*) 'COEFFICIENTS (GSFC1283)'
      read (*,9990) filename
      open (13, file=filename,status='old',form='formatted')
c------------ use the following if you want to input your own nowant
c             file
c      write (*,*) '1 to remove certain pass numbers'
c      write (*,*) '0 do no remove any pass numbers'
c      read (*,*) nocnt
c      if (nocnt .ne. 0) then
c         write (*,*) 'input file of pass numbers not wanted'
c         read (*,9990) filename
c         open (14, file=filename,status='old',form='formatted')
c         do i=1,5000
c            read (14,*,end=10) nowant(i)
c         enddo
```

```
c 10      nocnt=i-1
c       endif
c--------------- the following lines automatically place the nowant
c                pass numbers in the nowant array.  these passes are
c                messed up for one reason or another.  i'm sure they
c                could be salvaged, but i'm lazy.
      nowant(1)=909
      nowant(2)=1079
      nowant(3)=1206
      nowant(4)=2602
      nowant(5)=2728
      nowant(6)=2744
      nowant(7)=2791
      nowant(8)=2854
      nowant(9)=3059
      nocnt=9
      write (*,*) 'OUTPUT DAWN DATA FILE OF 2-INTEGERS AND 27-REALS'
      read (*,9990) filename
      open (20, file=filename,form='unformatted',access='DIRECT',
     >      recl=116)
      write (*,*) 'OUTPUT DUSK DATA FILE OF 2-I AND 27-R'
      read (*,9990) filename
      open (21, file=filename,form='unformatted',access='DIRECT',
     >      recl=116)
      write (*,*) 'OUTPUT HEADERS FILE'
      read (*,9990) filename
      open (22, file=filename,form='formatted')
c
      write (*,*) '0 FOR NO ADDITIONAL INFORMATION'
      write (*,*) '1 FOR ONLY Dst INDEXES'
      write (*,*) '2 FOR COMPLETE INFORMATION... this is a big file'
      read (*,*) info
      if (info .gt. 0) then
         write (*,*) 'OUTPUT ADDITIONAL INFORMATION ORBIT FILE'
         read (*,9990) filename
         open (23, file=filename,form='formatted')
      endif
c
      write (*,*) 'MINIMUM AND MAXIMUM LATITUDE OF STUDY AREA'
      write (*,*) 'INPUT RANGE IS FROM -90.0 TO 90.0'
      read (*,*) minlat,maxlat
      write (*,*) 'MINIMUM AND MAXIMUM LONGITUDE OF STUDY AREA'
      write (*,*) 'INPUT RANGE IS FROM -180.0 TO 180.0'
      read (*,*) minlon,maxlon
c
c---------------------------------- the following arrays store the spherical
c                                   harmonic coefficients that describe the
c                                   core field
      READ (13,9926) Jnum,Knum,TtZERO, AaID
 9926 FORMAT (2I1,1X,F6.1,A60)
      MmAXN=0
      TtEMP=0.
 50   READ (13,9928) N,M,GNM,HNM,GTNM,HTNM,GTTNM,HTTNM
 9928 FORMAT (2I3,6F11.4)
      IF (N.LE.0) GO TO 80
      MmAXN=(MAX0(N,MmAXN))
      Gg(N,M)=GNM
      GgT(N,M)=GTNM
      GgTT(N,M)=GTTNM
      TtEMP=AMAX1(TtEMP,ABS(GTNM))
      IF (M.EQ.1) GO TO 50
      Gg(M-1,N)=HNM
      GgT(M-1,N)=HTNM
      GgTT(M-1,N)=HTTNM
      GO TO 50
 80   CONTINUE
c
      dnrec=0
      dkrec=0
      totrecord=0
      datacnt=0
      headcnt=0
      stop10=0
      stop11=0
c
      head10=0
      data10=0
      recnum=1
 100  num=1
      read (10,end=120) rhead
      head10=head10+1
      go to 220
c
 120  stop10=1
      close (10)
      write (*,*) 'done with file one'
      write (*,*) 'total headers on file one =',head10
      write (*,*) 'total data sets on file one =',data10
      head11=0
      data11=0
```

```
      totrecord=totrecord+recnum
      recnum=1
c---------------------------- use the go to 999 statement if you only
c                             want to input one file at a time
      go to 999
 125  num=1
      read (11,end=150) rhead
      head11=head11+1
      go to 220
c
 150  stop11=1
      close (11)
      write (*,*) 'done with file two'
      write (*,*) 'total headers on file two =',head11
      write (*,*) 'total data sets on file two =',data11
      head12=0
      data12=0
      totrecord=totrecord+recnum
      recnum=1
 155  num=1
      read (12,end=999) rhead
      head12=head12+1
c
 220  continue
c---------------------------------- if flag1=1 then this is header file
      if (flag1 .eq. 1) then
         if (info .eq. 1) then
            write (23,*) ihead(4),(rhead(i),i=23,34)
         elseif (info .eq. 2) then
            write (23,*) (ihead(i),i=1,4),(rhead(i),i=5,8),
     >                  (ihead(i),i=9,10),(rhead(i),i=11,14),
     >                  (ihead(i),i=15,16),(rhead(i),i=17,34),
     >                  (chead(i),i=35,64),(ihead(i),i=65,67),
     >                  (rhead(i),i=68,557)
         endif
         write (22,8880) ihead(4),(rhead(i),i=5,8),(ihead(i),i=15,16)
 8880    format (1x,i5,4e15.7,2i6)
         dndkpass=ihead(4)
         duske=rhead(5)
         duski=rhead(6)
         dawne=rhead(7)
         dawni=rhead(8)
         headcnt=headcnt+1
         recnum=recnum+1
c
         if (flag2 .eq. 2) then
c
 230        continue
c
            if (stop10.eq.0 .and. stop11.eq.0) then
               data10=data10+1
               read (10, end=120) rbuff
            elseif (stop10.eq.1 .and. stop11.eq.0) then
               data11=data11+1
               read (11, end=150) rbuff
            elseif (stop10.eq.1 .and. stop11.eq.1) then
               data12=data12+1
               read (12, end=999) rbuff
            endif
            recnum=recnum+1
            datacnt=datacnt+1
c------------------------------------- if flag1=2 then this is a data file
c
            if (flag1 .eq. 1) then
               write (*,*) 'problem with flag1 =',flag1,
     >                     'in a data record'
               stop
            endif
            do i=num,num+29
               idata(i,1)=ibuff(5)
               idata(i,2)=ibuff(3)
               xnum=real(i-num)
               seconds(i)=(real(ibuff(4)) + (rbuff(6)*xnum))
            enddo
c------------------------------------- these do 250 loops reorder the data
c                                      from column to row oriented data
            jj=7
            do 250 j=1,25
               do 250 i=num,num+29
                  data(i,j)=rbuff(jj)
                  jj=jj+1
 250        continue
c
            num=num+30
c
c------------------------------------- if flag2=1 then the next record is
c                                      a header and the data information is
c                                      complete for this orbit
            if (flag2 .eq. 1) then
               innum=num-1
```

```
c
c-------------------------------------- search the nowant array for passes
c                                       that just didn't happen.  the
c                                       following passes are doubled and
c                                       are considered not wanted:
c                                       909,1079,1206,2602,2728,2744,2791,
c                                       2854,3059
c
               do i=1,nocnt
                  if (nowant(i) .eq. idata(1,1)) then
                     write (*,*) 'removed pass number ',
     >                           nowant(i),idata(1,1)
                     if (stop10.eq.0 .and. stop11.eq.0) go to 100
                     if (stop10.eq.1 .and. stop11.eq.0) go to 125
                     if (stop10.eq.1 .and. stop11.eq.1) go to 155
                  endif
               enddo
c
               do 270 i=1,innum
                  data(i,26)=seconds(i)
 270           continue
c
               call nine (innum,outnum)
               innum=outnum
               call area (innum,outnum)
               innum=outnum
               call dawndusk (innum,dnnum,dknum)
c
               if (dnnum .le. 0) go to 310
               call pfligrf (dnnum,1)
               call corering (dnnum,1)
               do 300 i=1,dnnum
                  dnrec=dnrec+1
                  write (20,rec=dnrec) (idndata(i,j),j=1,2),
     >                  (outdawn(i,j),j=1,27)
 300           continue
 310           continue
c
               if (dknum .le. 0) go to 360
               call pfligrf (dknum,2)
               call corering (dknum,2)
               do 350 i=1,dknum
                  dkrec=dkrec+1
                  write (21,rec=dkrec) (idkdata(i,j),j=1,2),
     >                  (outdusk(i,j),j=1,27)
 350           continue
 360           continue
c
c----------------------- these ugly little go to's get back to a
c                         header record
               if (stop10.eq.0 .and. stop11.eq.0) go to 100
               if (stop10.eq.1 .and. stop11.eq.0) go to 125
               if (stop10.eq.1 .and. stop11.eq.1) go to 155
c
            endif
c----------------------- go back and get another data record
            go to 230
c
         endif
c
      elseif (flag2 .eq. 1) then
         write (*,*) 'a header file had no associated data record'
         write (*,*) 'this header file has pass number ',ihead(4)
      endif
c
      if (stop10.eq.0 .and. stop11.eq.0) go to 100
      if (stop10.eq.1 .and. stop11.eq.0) go to 125
      if (stop10.eq.1 .and. stop11.eq.1) go to 155
c
 999  continue
      write (*,*) 'total headers on file three =',head12
      write (*,*) 'total data sets on file three =',data12
      write (*,*) 'total headers on tapes =',headcnt
      write (*,*) 'total data sets on tapes =',datacnt
      totrecord=totrecord+recnum
      write (*,*) 'total records read =',totrecord
      write (*,*) 'total records written to dawn file =',dnrec
      write (*,*) 'total records written to dusk file =',dkrec
      close (12)
      close (13)
      close (14)
      close (20)
      close (21)
      close (22)
      close (23)
      stop
      end
c
c
c
```

```
      subroutine nine (innum,nincnt)
      common data(3000,26),idata(3000,2)
c
c----------------------------- subroutine description
c        this subroutine removes from the data array all variables
c        associated with a single sampling point if selected variables at
c        that sampling point are greater than 99999.0
c
      nincnt=0
c
      do 100 i=1,innum
         do j=1,3
            if (data(i,j) .ge. 99999.0) go to 200
         enddo
         do j=5,6
            if (data(i,j) .ge. 99999.0) go to 200
         enddo
         do j=8,23
            if (data(i,j) .ge. 99999.0) go to 200
         enddo
         if (idata(i,1) .ge. 9999) go to 200
c         if (data(i,1).eq.99999.0 .or. data(i,2).eq.99999.0 .or.
c     >        data(i,3).eq.99999.0 .or. data(i,8).eq.99999.0 .or.
c     >      data(i,12).eq.99999.0) go to 200
         nincnt=nincnt+1
         do 140 j=1,2
            idata(nincnt,j)=idata(i,j)
 140     continue
         do 150 j=1,26
            data(nincnt,j)=data(i,j)
 150     continue
 200     continue
 100  continue
c
      return
      end
c
c
c
      subroutine area(innum,outnum)
      integer*4 innum,outnum
      real*4 minlat,maxlat,minlon,maxlon
      common /latlon/ minlat,maxlat,minlon,maxlon
      common data(3000,26),idata(3000,2)
c
c-------------------------------------- subroutine description
c        this subroutine removes all data outside of the user defined
c        area.
c
      outnum=0
      do 200 i=1,innum
         if (data(i,1).gt.maxlat .or. data(i,1).lt.minlat .or.
     >       data(i,2).gt.maxlon .or. data(i,2).lt.minlon) go to 100
         outnum=outnum+1
         do 140 j=1,2
            idata(outnum,j)=idata(i,j)
 140     continue
         do 150 j=1,26
            data(outnum,j)=data(i,j)
 150     continue
 100     continue
 200  continue
c
      return
      end
c
c
c
      subroutine dawndusk (innum,dncnt,dkcnt)
      integer*4 innum,dncnt,dkcnt,totcnt
      common /dndkdat/ dndata(1500,26),dkdata(1500,26),
     >                 idndata(1500,2),idkdata(1500,2)
      common data(3000,26),idata(3000,2)
c
c---------------------------------- subroutine description
c        this subroutine separates the data array into dawn and dusk data
c        sets.
c
      data(innum+1,1) = -90.0
      dkcnt=0
      dncnt=0
c
      do 200 i=1,innum
         if (data(i,1) .lt. data(i+1,1)) then
            dkcnt=dkcnt+1
            do 90 j=1,2
               idkdata(dkcnt,j)=idata(i,j)
 90         continue
            do 100 j=1,26
               dkdata(dkcnt,j)=data(i,j)
```

```
 100        continue
         elseif (data(i,1) .gt. data(i+1,1)) then
            dncnt=dncnt+1
c------------------------- remove the first "dawn" data point because
c                          in reality this point could actually belong
c                          to a dusk profile.  a look at the longitudes
c                          will prove this point.  of course with some
c                          extra code this point could be saved -- but
c                          hey its only one in a thousand!!
c
            if (dncnt .eq. 1) go to 160
            do 140 j=1,2
               idndata(dncnt-1,j)=idata(i,j)
 140        continue
            do 150 j=1,26
               dndata(dncnt-1,j)=data(i,j)
 150        continue
 160        continue
         elseif (data(i,1) .eq. data(i+1,1)) then
            write (*,8880) data(i,1),data(i+1,1),idata(i,1)
 8880       format ('two latitudes are equal therefore program skips',
     >            /,'the first latitude ',f9.4,' and reviews the'/,
     >              'second latitude ',f9.4,' for pass number',i6)
         endif
 200  continue
c
      totcnt = dkcnt + dncnt
      if (totcnt .ne. innum) write (*,8881) dkcnt,dncnt,totcnt,innum
 8881 format ('total dusk observations =',i4,' total dawn obs. =',i4,
     >       /,'totals added =',i5,' which differs from the input',
     >       /,'of the area selected =',i5)
c
      dncnt=dncnt-1
      return
      end
c
c
c
      subroutine pfligrf (innum,idndk)
      common /dndkdat/ dndata(1500,26),dkdata(1500,26),
     >                 idndata(1500,2),idkdata(1500,2)
      COMMON /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)
C
C*******************************************************************
C
C
C     PROGRAM   P F L I G R F
C
C
C     THIS PROGRAM CALCULATES VALUES OF ALL OF THE FOLLOWING ALONG
C     MAGSAT PROFILES CREATED BY STEP1P4 FORTRAN AT PURDUE.
C             INDEX    VALUE
c                1  = pass number
C                2  = TOTAL FIELD
C                3  = X COMPONENT
C                4  = Y COMPONENT
C                5  = Z COMPONENT
C                6  = INCLINATION
C                7  = DECLINATION
C                8  = latitude point to assure correct points are
c                     compared
C
C  YEAR(I)=EPOCH IN YEARS AND DECIMAL FRACTION YEARS (E.G., 1965.75=
C      1 OCT. 65) FOR WHICH THE GEOMAGNETIC REFERENCE FIELD IS TO
C      BE COMPUTED AT OBSERVATION POINTS.
C      THEN THE GEOMAGNETIC FIELD OVER THE OBSER-
C      VATION POINT IS COMPUTED BY SUBROUTINE FIELDG FOR THE EPOCH
C      SPECIFIED BY THE YEAR-INPUT VARIABLE
C
C    TAPE UNITS:
C             4. (U/I).....DATA FILES CREATED BY STEP1P4
C
C             7. (U/O).....OUTPUT...WATCH THE ORDER OF VARIABLES
C
c     revised  25 AUG 90
c
c     this subroutine was modified to read the spherical harmonic
c     coefficients in the main program and transfer those coefficients
c     by a common block.  with these modifications, the file holding the
c     coefficients is only read once and not a thousand billion times
c     which decreases total run time on the program. (ok ok, so maybe
c     not a thousand billion times, but only the number of dawn and
c     dusk profiles given to the subroutine.)
C
c     along the way i've removed some useless code that would write
c     items to file 6 or any of a number of additional places
c     depending on which format the user supplied.  so if the original
c     is desired, it can be found in programs named the same as
c     the subroutines in this program.
c
```

```
C*****************************************************************
C
      if (idndk .eq. 1) then
         do 50 i=1,innum
            ipass1=idndata(i,1)
            mjd=idndata(i,2)
            secx=dndata(i,26)
            theta(i)=dndata(i,1)
            phi(i)=dndata(i,2)
            elvo(i)=dndata(i,3)
            ELVO(I) = ELVO(I) - 6371.2
            IDAYS = 44239 - MJD
            IF (IDAYS .GT. 0) THEN
               FRACY = FLOAT(365-(IDAYS)) / FLOAT(365)
               FRACYA = SECX / (3600000.*24.*365.)
               YEAR(I) = 1979.0 + FRACY + FRACYA
            ELSE
               FRACY = FLOAT(-IDAYS) / FLOAT(366)
               FRACYA = SECX / (3600000.*24.*366.)
               YEAR(I) = 1980.0 + FRACY + FRACYA
            ENDIF
 50      continue
      elseif (idndk .eq. 2) then
         do 70 i=1,innum
            ipass1=idkdata(i,1)
            mjd=idkdata(i,2)
            secx=dkdata(i,26)
            theta(i)=dkdata(i,1)
            phi(i)=dkdata(i,2)
            elvo(i)=dkdata(i,3)
            ELVO(I) = ELVO(I) - 6371.2
            IDAYS = 44239 - MJD
            IF (IDAYS .GT. 0) THEN
               FRACY = FLOAT(365-(IDAYS)) / FLOAT(365)
               FRACYA = SECX / (3600000.*24.*365.)
               YEAR(I) = 1979.0 + FRACY + FRACYA
            ELSE
               FRACY = FLOAT(-IDAYS) / FLOAT(366)
               FRACYA = SECX / (3600000.*24.*366.)
               YEAR(I) = 1980.0 + FRACY + FRACYA
            ENDIF
 70      continue
      endif
c
      np=innum
      LQ=1
      CALL FIELDG (0.,0.,0.,0.,50,LQ,Q1,Q2,Q3,Q4)
      CALL GEOMAG (NP,IPASS1)
c
      return
      end
c
c
c
      SUBROUTINE GEOMAG (NPTS,IPASS1)
      COMMON /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)
      common /mainfld/ fld(1500,8),dawne,dawni,duske,duski,dndkpass
      INTEGER*4 IPASS1,dndkpass
C
C
C        ************************************************************
C        THIS SUBROUTINE CALCULATES THE MAGNITUDE, INCLINATION, AND
C        DECLINATION OF THE GEOMAGNETIC FIELD ON A GRID NTHETA BY
C        NPHI
C        ************************************************************
C
C          THETA,PHI  **  ORIGIN OF THE GRID (DEG.)
C                ELV  **  ELAVATION OF GRID (KILO. ABOVE SEA LEVEL)
C        HTHETA,HPHI  **  GRID SPACING (DEG.)
C        NTHETA,NPHI  **  DIMENSIONS OF THE GRID
C                 NF  **  UNIT FILE WHICH WILL STORE THE FIELD
C
C        ************************************************************
C        SUBROUTINES USED
C           ** FIELDG **  (NASA)
C           ** FIELD **   (NASA)
C        ************************************************************
C
      LL=0
      RD=180./3.14159265
c
      DO 100 I=1,NPTS
         ATHETA = THETA(I)
         APHI = PHI(I)
         ELV = ELVO(I)
         YR = YEAR(I)
c
         CALL FIELDG (ATHETA,APHI,ELV,YR,50,LL,X,Y,Z,FF)
c
         H=SQRT(X*X+Y*Y)
         T=SQRT(H*H+Z*Z)
```

```
        finc=RD*ATAN2(Z,H)
        fdec=RD*ATAN2(Y,X)
        fld(i,1)=ipass1
        fld(i,2)=t
        fld(i,3)=x
        fld(i,4)=y
        fld(i,5)=z
        fld(i,6)=finc
        fld(i,7)=fdec
        thet=theta(i)
        fld(i,8)=thet
 100  continue
c
      RETURN
      END
c
c
c
      SUBROUTINE FIELDG (DLAT,DLONG,ALT,TM,NMX,L,X,Y,Z,F)
      EQUIVALENCE (SHMIT(1,1),TG(1,1))
      COMMON /NASA/ TG(50,50)
      COMMON /FLDCOM/ ST,CT,SPH,CPH,R,NMAX,BT,BP,BR,B
      common /coeff/ gg(50,50),ggt(50,50),ggtt(50,50),jnum,knum,
     >                 ttzero,aaid,mmaxn,ttemp
      DIMENSION G(50,50), GT(50,50), SHMIT(50,50), GTT(50,50)
      CHARACTER*60 AID,aaid
      DATA A/0./
C
C        ***************************************************************
C        FOR DOCUMENTATION OF THIS SUBROUTINE AND SUBROUTINE FIELD SEE :
C             NATIONAL SPACE SCIENCE DATA CENTER*S PUBLICATION
C             **COMPUTATION OF THE MAIN GEOMAGNETIC FIELD
C               FROM SPHERICAL HARMONIC EXPANSIONS**
C             DATA USERS' NOTE, NSSDC 68-11, MAY 1968
C             GODDARD SPACE FLIGHT CENTER, GREENBELT, MD.
C        ***************************************************************
C
C         DLAT ** LATITUDE IN DEGREES POSITIVE NORTH
C        DLONG ** LONGITUDE IN DEGREES POSITIVE EAST
C          ALT ** ELEVATION IN KM (POSITIVE ABOVE, NEGATIVE BELOW
C                 EARTH'S SURFACE)
C           TM ** EPOCH IN YEARS
C          NMX ** SET TO INTEGER GREATER THAN DEGREE OF EXPANSION
C            L ** SET TO 1 ON INITIAL DUMMY CALL, SET TO 0 ON SUBSEQUENT
C                 CALLS
C
C        SUBROUTINE RETURNS GEOMAGNETIC FIELD DIRECTIONS (X,Y,Z), POSI-
C        TIVE NORTH, EAST AND DOWN, RESPECTIVELY, AND MAGNITUDE OF TOTAL
C        FIELD, F---ALL VALUES ARE IN GAMMAS
C
C------ from the data statement above, A = 0.0 only and only on the first
c       call to this subroutine from anywhere within the program.  after
c       the first call, it is seen below that A = 6371.2 for all future
c       calls during the running of the program
c
C
      TLAST=0.0
C
C     IF(A.EQ.6378.139) IF(L) 210,100,110
      IF(A.EQ.6371.2) IF(L) 210,100,110
C
C     A=6378.139
      A = 6371.2
C     FLAT=1.-1./298.25
      FLAT = 1.
      A2=A**2
      A4=A**4
      B2=(A*FLAT)**2
      A2B2=A2*(1.-FLAT**2)
      A4B4=A4*(1.-FLAT**4)
      IF (L) 160,160,110
 100  IF (TM-TLAST) 190,210,190
c
 110  continue
      L=0
      j=jnum
      k=knum
      tzero=ttzero
      aid=aaid
      maxn=mmaxn
      temp=ttemp
      do 120 ii=1,maxn
         do 120 iii=1,maxn
            g(ii,iii)=gg(ii,iii)
            gt(ii,iii)=ggt(ii,iii)
            gtt(ii,iii)=ggtt(ii,iii)
 120     continue
c
c
c  110 READ (3,260) J,K,TZERO, AID
```

```
c       L=0
c      WRITE (7,270) J,K,TZERO, AID
c       MAXN=0
c       TEMP=0.
c  120 READ (3,280) N,M,GNM,HNM,GTNM,HTNM,GTTNM,HTTNM
c      WRITE(7,280) N,M,GNM,HNM,GTNM,HTNM
c       IF (N.LE.0) GO TO 130
c       MAXN=(MAX0(N,MAXN))
c       G(N,M)=GNM
c       GT(N,M)=GTNM
c       GTT(N,M)=GTTNM
c       TEMP=AMAX1(TEMP,ABS(GTNM))
c       IF (M.EQ.1) GO TO 120
c       G(M-1,N)=HNM
c       GT(M-1,N)=HTNM
c       GTT(M-1,N)=HTTNM
c       GO TO 120
c 130 WRITE (7,290)
c  130 CONTINUE
c       DO 150 N=2,MAXN
c       DO 150 M=1,N
c          MI=M-1
c          IF (M.EQ.1) GO TO 140
c         WRITE (7,300) N,M,G(N,M),G(MI,N),GT(N,M),GT(MI,N),GTT(N,M),GTT(
c      1   MI,N)
c          GO TO 150
c 140     WRITE (7,310) N,M,G(N,M),GT(N,M),GTT(N,M)
c  140 CONTINUE
c  150 CONTINUE
c      WRITE (7,320)
       IF (TEMP.EQ.0.) L=-1
c       REWIND 3
  160 IF (K.NE.0) GO TO 190
       SHMIT(1,1)=-1.
       DO 170 N=2,MAXN
          SHMIT(N,1)=SHMIT(N-1,1)*FLOAT(2*N-3)/FLOAT(N-1)
          SHMIT(1,N)=0.
          JJ=2
       DO 170 M=2,N
          SHMIT(N,M)=SHMIT(N,M-1)*SQRT(FLOAT((N-M+1)*JJ)/FLOAT(N+M-2))
          SHMIT(M-1,N)=SHMIT(N,M)
  170 JJ=1
       DO 180 N=2,MAXN
       DO 180 M=1,N
          G(N,M)=G(N,M)*SHMIT(N,M)
          GT(N,M)=GT(N,M)*SHMIT(N,M)
          GTT(N,M)=GTT(N,M)*SHMIT(N,M)
          IF (M.EQ.1) GO TO 180
          G(M-1,N)=G(M-1,N)*SHMIT(M-1,N)
          GT(M-1,N)=GT(M-1,N)*SHMIT(M-1,N)
          GTT(M-1,N)=GTT(M-1,N)*SHMIT(M-1,N)
  180 CONTINUE
  190 T=TM-TZERO
       DO 200 N=1,MAXN
       DO 200 M=1,N
          TG(N,M)=G(N,M)+T*(GT(N,M)+GTT(N,M)*T)
          IF (M.EQ.1) GO TO 200
          TG(M-1,N)=G(M-1,N)+T*(GT(M-1,N)+GTT(M-1,N)*T)
  200 CONTINUE
       TLAST=TM
  210 DLATR=DLAT/57.2957795
       SINLA=SIN(DLATR)
       RLONG=DLONG/57.2957795
       CPH=COS(RLONG)
       SPH=SIN(RLONG)
       IF (J.EQ.0) GO TO 220
       R=ALT+6371.2
       CT=SINLA
       GO TO 230
  220 SINLA2=SINLA**2
       COSLA2=1.-SINLA2
       DEN2=A2-A2B2*SINLA2
       DEN=SQRT(DEN2)
       FAC=(((ALT*DEN)+A2)/((ALT*DEN)+B2))**2
       CT=SINLA/SQRT(FAC*COSLA2+SINLA2)
       R=SQRT(ALT*(ALT+2.*DEN)+(A4-A4B4*SINLA2)/DEN2)
  230 ST=SQRT(1.-CT**2)
       NMAX=MINO(NMX,MAXN)
C
       CALL FIELD
C
       Y=BP
       F=B
       IF (J) 240,250,240
  240 X=-BT
       Z=-BR
       RETURN
C
C      TRANSFORMS FIELD TO GEODETIC DIRECTIONS
C
```

```
  250 SIND=SINLA*ST-SQRT(COSLA2)*CT
      COSD=SQRT(1.0-SIND**2)
      X=-BT*COSD-BR*SIND
      Z=BT*SIND-BR*COSD
      RETURN
C
  260 FORMAT (2I1,1X,F6.1,A60)
  270 FORMAT (2I3,5X,6HEPOCH ,F7.1,5X,A60)
  280 FORMAT (2I3,6F11.4)
  290 FORMAT (6HO N  M,6X,1HG,10X,1HH,9X,2HGT,9X,2HHT,8X,3HGTT,8X,3HHTT/
     1)
  300 FORMAT (2I3,6F11.4)
  310 FORMAT (2I3,F11.4,11X,F11.4,11X,F11.4)
  320 FORMAT (///)
C
      END
c
c
c
      SUBROUTINE FIELD
      COMMON /NASA/ G(50,50)
      COMMON /FLDCOM/ ST,CT,SPH,CPH,R,NMAX,BT,BP,BR,B
      DIMENSION P(50,50), DP(50,50), CONST(50,50), SP(50), CP(50),
     >          FN(50), FM(50)
      DATA P(1,1)/0./
c
      IF (P(1,1).EQ.1.0) GO TO 120
      P(1,1)=1.
      DP(1,1)=0.
      SP(1)=0.
      CP(1)=1.
      DO 110 N=2,nmax
         FN(N)=N
      DO 110 M=1,N
         FM(M)=M-1
  110 CONST(N,M)=FLOAT((N-2)**2-(M-1)**2)/FLOAT((2*N-3)*(2*N-5))
  120 SP(2)=SPH
      CP(2)=CPH
      DO 130 M=3,NMAX
         SP(M)=SP(2)*CP(M-1)+CP(2)*SP(M-1)
  130 CP(M)=CP(2)*CP(M-1)-SP(2)*SP(M-1)
      AOR=6371.2/R
      AR=AOR**2
      BT=0.
      BP=0.
      BR=0.
      DO 190 N=2,NMAX
         AR=AOR*AR
      DO 190 M=1,N
         IF (N-M) 150,140,150
  140    P(N,N)=ST*P(N-1,N-1)
         DP(N,N)=ST*DP(N-1,N-1)+CT*P(N-1,N-1)
         GO TO 160
  150    P(N,M)=CT*P(N-1,M)-CONST(N,M)*P(N-2,M)
         DP(N,M)=CT*DP(N-1,M)-ST*P(N-1,M)-CONST(N,M)*DP(N-2,M)
  160    PAR=P(N,M)*AR
         IF (M.EQ.1) GO TO 170
         TEMP=G(N,M)*CP(M)+G(M-1,N)*SP(M)
         BP=BP-(G(N,M)*SP(M)-G(M-1,N)*CP(M))*FM(M)*PAR
         GO TO 180
  170    TEMP=G(N,M)*CP(M)
         BP=BP-(G(N,M)*SP(M))*FM(M)*PAR
  180    BT=BT+TEMP*DP(N,M)*AR
  190 BR=BR-TEMP*FN(N)*PAR
      BP=BP/ST
      B=SQRT(BT*BT+BP*BP+BR*BR)
c
      RETURN
      END
c
c
c
      subroutine corering (innum,idndk)
      integer*4 dndkpass
      common /mainfld/ fld(1500,8),dawne,dawni,duske,duski,dndkpass
      common /dndkdat/ dndata(1500,26),dkdata(1500,26),
     >                 idndata(1500,2),idkdata(1500,2)
      common /thatsit/ outdawn(1500,27),outdusk(1500,27)
c
c------------------------------ subroutine description
c       this subroutine subtracts the core field at each data point and
c       calculates the ring current affect as defined by NASA's formula
c       which uses the E and I values for the entire orbit.  this ring
c       current value is also subtracted to yield the 'resid'ual value.
c       since the total field value, the core-field value and the ring
c       current values are written to file, any one value can be obtained
c       at the next processing step.
c           NOTE: the core field subtraction is not a least squares
c                 procedure.  the least squares removal is done in
c                 program massage
```

```
c
      pie=3.1415927
      radius=6371.2
c
      if (idndk .eq. 1) then
         do 100 i=1,innum
c
            if (dndata(i,1) .ne. fld(i,8)) then
               write (*,*) 'no match between latitudes in corering',
     >                     ' subroutine with dawn dataset'
               write (*,*) dndata(i,1),fld(i,8)
               stop
            elseif (dndkpass.ne.idndata(i,1) .or. dndkpass.ne.fld(i,1)
     >            .or.  idndata(i,1).ne.fld(i,1)) then
               write (*,*) 'no match between pass numbers in corering',
     >               ' subroutine with dawn dataset'
               write (*,*) dndkpass,idndata(i,1),fld(i,1)
               stop
            endif
c
            totmag=dndata(i,8)-fld(i,2)
            tavgmag=dndata(i,12)-fld(i,2)
            dip=dndata(i,6)*(pie/180.0)
            delbzz=(dawne*sin(dip))-(2.0*dawni*sin(dip)*
     >            ((radius/dndata(i,3))**3.0))
            delbxx=(-1.0*dawne*cos(dip))-(dawni*cos(dip)*
     >            ((radius/dndata(i,3))**3.0))
            ringcur=(sqrt(((fld(i,3)+delbxx)**2.0) + (fld(i,4)**2.0) +
     >              ((fld(i,5)+delbzz)**2.0))) - fld(i,2)
            resid=totmag-ringcur
            resavgmag=tavgmag-ringcur
            do 150 j=1,15
               outdawn(i,j)=dndata(i,j)
 150        continue
            do 170 j=16,21
               jj=j-14
               outdawn(i,j)=fld(i,jj)
 170        continue
            outdawn(i,22)=totmag
            outdawn(i,23)=tavgmag
            outdawn(i,24)=resid
            outdawn(i,25)=resavgmag
            outdawn(i,26)=ringcur
            outdawn(i,27)=dndata(i,26)
 100     continue
c
      elseif (idndk .eq. 2) then
         do 200 i=1,innum
c
            if (dkdata(i,1) .ne. fld(i,8)) then
               write (*,*) 'no match between latitudes in corering',
     >                     ' subroutine with dusk dataset'
               write (*,*) dkdata(i,1),fld(i,8)
               stop
            elseif (dndkpass.ne.idkdata(i,1) .or. dndkpass.ne.fld(i,1)
     >            .or. idkdata(i,1).ne.fld(i,1)) then
               write (*,*) 'no match between pass numbers in corering',
     >                     ' subroutine with dusk dataset'
               write (*,*) dndkpass,idkdata(i,1),fld(i,1)
               stop
            endif
c
            totmag=dkdata(i,8)-fld(i,2)
            tavgmag=dkdata(i,12)-fld(i,2)
            dip=dkdata(i,6)*(pie/180.0)
            delbzz=(duske*sin(dip))-(2.0*duski*sin(dip)*
     >            ((radius/dkdata(i,3))**3.0))
            delbxx=(-1.0*duske*cos(dip))-(duski*cos(dip)*
     >            ((radius/dkdata(i,3))**3.0))
            ringcur=(sqrt(((fld(i,3)+delbxx)**2.0) + (fld(i,4)**2.0) +
     >              ((fld(i,5)+delbzz)**2.0))) - fld(i,2)
            resid=totmag-ringcur
            resavgmag=tavgmag-ringcur
            do 250 j=1,15
               outdusk(i,j)=dkdata(i,j)
 250        continue
            do 270 j=16,21
               jj=j-14
               outdusk(i,j)=fld(i,jj)
 270        continue
            outdusk(i,22)=totmag
            outdusk(i,23)=tavgmag
            outdusk(i,24)=resid
            outdusk(i,25)=resavgmag
            outdusk(i,26)=ringcur
            outdusk(i,27)=dkdata(i,26)
 200     continue
c
      endif
c
```

```
      return
      end
c
c---------------- this is the driver for direct access
c
c      program subcore
c      real*4 rbuff(3024/4),minlat,maxlat,minlon,maxlon,
c     >       seconds(3000)
c      integer*4 ibuff(3024/4),flag1,flag2,headcnt,recnum,
c     >          outnum,dnnum,dknum,
c     >          dndkpass,datacnt,stop10,stop11,
c     >          head10,data10,head11,data11,head12,data12,totrecord,
c     >          dnrec,dkrec,nowant(50)
c      character*80 filename
c      character*4 cbuff(3024/4)
c      character*60 aaid
c      equivalence (ibuff,rbuff),(cbuff,rbuff),(flag1,ibuff(1)),
c     >            (flag2,ibuff(2))
c      common /latlon/ minlat,maxlat,minlon,maxlon
c      common /dndkdat/ dndata(1500,26),dkdata(1500,26),
c     >                 idndata(1500,2),idkdata(1500,2)
c      common /coeff/ gg(50,50),ggt(50,50),ggtt(50,50),jnum,knum,
c     >               ttzero,aaid,mmaxn,ttemp
c      common /mainfld/ fld(1500,8),dawne,dawni,duske,duski,dndkpass
c      common /thatsit/ outdawn(1500,27),outdusk(1500,27)
c      COMMON /NASA/ TG(50,50)
c      COMMON /FLDCOM/ ST,CT,SPH,CPH,R,NMAX,BT,BP,BR,B
c      COMMON /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)
c      common data(3000,26),idata(3000,2)
cc
c      write (*,*) 'INPUT FIRST DATASET FROM TAPE TO DISC TRANSFER'
c      read (*,9990) filename
c 9990 format (a80)
c      open (10, file=filename,status='old',form='unformatted',
c     >      access='DIRECT',recl=3024/4)
c      write (*,*) 'INPUT SECOND DATASET FROM TAPE TO DISC TRANSFER'
c      read (*,9990) filename
c      open (11, file=filename,status='old',form='unformatted',
c     >      access='DIRECT',recl=3024/4)
c      write (*,*) 'INPUT THIRD DATASET FROM TAPE TO DISC TRANSFER'
c      read (*,9990) filename
c      open (12, file=filename,status='old',form='unformatted',
c     >      access='DIRECT',recl=3024/4)
c      write (*,*) 'INPUT FIELD MODEL SPHERICAL HARMONIC'
c      write (*,*) 'COEFFICIENTS (GSFC1283)'
c      read (*,9990) filename
c      open (13, file=filename,status='old',form='formatted')
cc
cc-----------------------use the following if you want to input
cc                        your own file of nowant passes
cc      write (*,*) '1 to remove certain pass numbers'
cc      write (*,*) '0 do no remove any pass numbers'
cc      read (*,*) nocnt
cc      if (nocnt .ne. 0) then
cc         write (*,*) 'input file of pass numbers not wanted'
cc         read (*,9990) filename
cc         open (14, file=filename,status='old',form='formatted')
cc         do i=1,5000
cc            read (14,*,end=10) nowant(i)
cc         enddo
cc 10      nocnt=i-1
cc      endif
cc--------------- the following lines automatically place the nowant
cc                pass numbers in the nowant array.  these passes are
cc                messed up for one reason or another.  i'm sure they
cc                could be salvaged, but i'm lazy.
c      nowant(1)=909
c      nowant(2)=1079
c      nowant(3)=1206
c      nowant(4)=2602
c      nowant(5)=2728
c      nowant(6)=2744
c      nowant(7)=2791
c      nowant(8)=2854
c      nowant(9)=3059
c      nocnt=9
c------------------------------ recl=116 for an ibm rs6000
c
c      write (*,*) 'OUTPUT DAWN DATA FILE OF 2-INTEGERS AND 27-REALS'
c      read (*,9990) filename
c      open (20, file=filename,form='unformatted',access='DIRECT',
c     >      recl=29)
c      write (*,*) 'OUTPUT DUSK DATA FILE OF 2-I AND 27-R'
c      read (*,9990) filename
c      open (21, file=filename,form='unformatted',access='DIRECT',
c     >      recl=29)
c      write (*,*) 'OUTPUT HEADERS FILE'
c      read (*,9990) filename
c      open (22, file=filename,form='formatted')
cc
```

```
c       write (*,*) '0 FOR NO ADDITIONAL INFORMATION'
c       write (*,*) '1 FOR ONLY Dst INDEXES'
c       write (*,*) '2 FOR COMPLETE INFORMATION... this is a big file'
c       read (*,*) info
c       if (info .gt. 0) then
c          write (*,*) 'OUTPUT ADDITIONAL INFORMATION ORBIT FILE'
c          read (*,9990) filename
c          open (23, file=filename,form='formatted')
c       endif
cc
c       write (*,*) 'MINIMUM AND MAXIMUM LATITUDE OF STUDY AREA'
c       write (*,*) 'INPUT RANGE IS FROM -90.0 TO 90.0'
c       read (*,*) minlat,maxlat
c       write (*,*) 'MINIMUM AND MAXIMUM LONGITUDE OF STUDY AREA'
c       write (*,*) 'INPUT RANGE IS FROM -180.0 TO 180.0'
c       read (*,*) minlon,maxlon
cc
cc---------------------------------- the following arrays store the spherical
cc                                   harmonic coefficients that describe the
cc                                   core field
c       READ (13,9926) Jnum,Knum,TtZERO, AaID
c 9926  FORMAT (2I1,1X,F6.1,A60)
c       MnAXN=0
c       TtEMP=0.
c 50    READ (13,9928) N,M,GNM,HNM,GTNM,HTNM,GTTNM,HTTNM
c 9928  FORMAT (2I3,6F11.4)
c       IF (N.LE.0) GO TO 80
c       MnAXN=(MAX0(N,MnAXN))
c       Gg(N,M)=GNM
c       GgT(N,M)=GTNM
c       GgTT(N,M)=GTTNM
c       TtEMP=AMAX1(TtEMP,ABS(GTNM))
c       IF (M.EQ.1) GO TO 50
c       Gg(M-1,N)=HNM
c       GgT(M-1,N)=HTNM
c       GgTT(M-1,N)=HTTNM
c       GO TO 50
c 80    CONTINUE
cc
c       dnrec=0
c       dkrec=0
c       totrecord=0
c       datacnt=0
c       headcnt=0
c       stop10=0
c       stop11=0
cc
c       head10=0
c       data10=0
c       recnum=1
cc
c 100   num=1
c 110   read (10,rec=recnum,err=120) rbuff
c       go to 220
cc
c 120   stop10=1
c       write (*,*) 'done with file one'
c       write (*,*) 'total headers on file one =',head10
c       write (*,*) 'total data sets on file one =',data10
c       head11=0
c       data11=0
c       totrecord=totrecord+recnum
c       recnum=1
cc----------------------------- use the go to 999 statement if you only
cc                              want to input one file at a time
c       go to 999
c 125   num=1
c 130   read (11,rec=recnum,err=150) rbuff
c       go to 220
cc
c 150   stop11=1
c       write (*,*) 'done with file two'
c       write (*,*) 'total headers on file two =',head11
c       write (*,*) 'total data sets on file two =',data11
c       head12=0
c       data12=0
c       totrecord=totrecord+recnum
c       recnum=1
c 155   num=1
c 160   read (12,rec=recnum,err=999) rbuff
cc
c 220   continue
cc----------------------------------- if flag1=1 then this is header file
c       if (flag1 .eq. 1) then
c          if (info .eq. 0) then
c             go to 225
c          elseif (info .eq. 1) then
c             write (23,*) ibuff(4),(rbuff(i),i=23,34)
c          elseif (info .eq. 2) then
c             write (23,*) (ibuff(i),i=1,4),(rbuff(i),i=5,8),
```

```
c     >                (ibuff(i),i=9,10),(rbuff(i),i=11,14),
c     >               (ibuff(i),i=15,16),(rbuff(i),i=17,34),
c     >               (cbuff(i),i=35,64),(ibuff(i),i=65,67),
c     >              (rbuff(i),i=68,557)
c        endif
c 225    continue
c        write (22,8880) ibuff(4),(rbuff(i),i=5,8),(ibuff(i),i=15,16)
c 8880   format (1x,i5,4e15.7,2i6)
c        dndkpass=ibuff(4)
c        duske=rbuff(5)
c        duski=rbuff(6)
c        dawne=rbuff(7)
c        dawni=rbuff(8)
c        headcnt=headcnt+1
c        recnum=recnum+1
c        if (stop10 .eq. 0  .and.  stop11 .eq. 0) then
c           head10=head10+1
c           go to 100
c        elseif (stop10 .eq. 1  .and.  stop11 .eq. 0) then
c           head11=head11+1
c           go to 125
c        elseif (stop10 .eq .1  .and.  stop11 .eq. 1) then
c           head12=head12+1
c           go to 155
c        endif
cc------------------------------------ if flag1=2 then this is a data file
c     elseif (flag1 .eq. 2) then
c        do 230 i=num,num+29
c           idata(i,1)=ibuff(5)
c           idata(i,2)=ibuff(3)
c           xnum=real(i-num)
c           seconds(i)=(real(ibuff(4)) + (rbuff(6)*xnum))
c 230    continue
cc------------------------------------ these do 250 loops reorder the data
cc                                    from column to row oriented data
c        jj=7
c        do 250 j=1,25
c           do 250 i=num,num+29
c              data(i,j)=rbuff(jj)
c              jj=jj+1
c 250       continue
cc
c        num=num+30
cc
cc------------------------------------ if flag2=1 then the next record is
cc                                    a header and the data information is
cc                                    complete for this orbit
c        if (flag1 .eq. 2  .and.  flag2 .eq. 1) then
c           innum=num-1
cc
cc----------------------------------- search the nowant array for passes
cc                                   that just didn't happen.  the
cc                                   following passes are doubled and
cc                                   are considered not wanted:
cc                                   909,1079,1206,2602,2728,2744,2791,
cc                                   2854,3059
c
c           do i=1,nocnt
c              if (nowant(i) .eq. idata(1,1)) then
c                 write (*,*) 'removed pass number ',
c     >                       nowant(i),idata(1,1)
c                 go to 400
c              endif
c           enddo
cc
c           do 270 i=1,innum
c              data(i,26)=seconds(i)
c 270       continue
cc
c           call nine (innum,outnum)
c           innum=outnum
c           call area (innum,outnum)
c           innum=outnum
c           call dawndusk (innum,dnnum,dknum)
cc
c           if (dnnum .le. 0) go to 310
c           call pfligrf (dnnum,1)
c           call corering (dnnum,1)
c           do 300 i=1,dnnum
c              dnrec=dnrec+1
c              write (20,rec=dnrec) (idndata(i,j),j=1,2),
c     >                             (outdawn(i,j),j=1,27)
c 300       continue
c 310       continue
cc
c           if (dknum .le. 0) go to 360
c           call pfligrf (dknum,2)
c           call corering (dknum,2)
c           do 350 i=1,dknum
c              dkrec=dkrec+1
c              write (21,rec=dkrec) (idkdata(i,j),j=1,2),
```

```
c     >                                     (outdusk(i,j),j=1,27)
c 350        continue
c 360        continue
c         endif
cc
c 400     recnum=recnum+1
c         datacnt=datacnt+1
c         if (stop10 .eq. 0  .and.  stop11 .eq. 0) then
c            data10=data10+1
c            go to 110
c         elseif (stop10 .eq. 1  .and.  stop11 .eq. 0) then
c            data11=data11+1
c            go to 130
c         elseif (stop10 .eq .1  .and.  stop11 .eq. 1) then
c            data12=data12+1
c            go to 160
c         endif
cc
c      elseif (flag1 .ne. 1  .or.  flag1 .ne. 2) then
c         write (*,*) 'HOLD THE FORT MAN, BAD FIRST FLAG NUMBER'
c         write (*,*) flag1
c         go to 999
c      endif
cc
c 999  continue
c      write (*,*) 'total headers on file three =',head12
c      write (*,*) 'total data sets on file three =',data12
c      write (*,*) 'total headers on tapes =',headcnt
c      write (*,*) 'total data sets on tapes =',datacnt
c      totrecord=totrecord+recnum
c      write (*,*) 'total records read =',totrecord
c      write (*,*) 'total records written to dawn file =',dnrec
c      write (*,*) 'total records written to dusk file =',dkrec
c      close (10)
c      close (11)
c      close (12)
c      close (13)
c      close (14)
c      close (20)
c      close (21)
c      close (22)
c      close (23)
c      stop
c      end
cc
cc-------------- add on all subroutines from here on
```

```
      program reorder
      character*80 filename
      integer passmjd(4000,2),idata(400,2),istore(2),
     >        countall,jstop,choice,passno(4000),dndk,
     >        pntcnt(4000,2),shrtpas(3000,2),shrtcnt,
     >        passrec(4000,2),passrem,cnter,denum,pchoice,
     >        spknum,mincheck,spkvar,nowant(4000),outnum,
     >        minchk,cntsome,inrec,innum,outrec,oif,inf,onf,
     >        innumall,pntall(4000,2)
      real data(400,27),dstore(27),east,west,diffwe,hilat,
     >     lolat,north,south,percent,totlat,upper,lower,
     >     desdata(400,27),intpdata(400,27)
      double precision aver(4000,2),ra(4000),cross,passavg(4000,2),
     >                 savglon(4000,2),crosss
      common data(400,27),idata(400,2)
      common /order1/ aver(4000,2),passmjd(4000,2),cross
      common /order2/ passno(4000),pntcnt(4000,2),pntall(4000,2)
      common /order3/ savglon(4000,2),crosss
      common /hsort/ ra(4000)
      common /shorty/ shrtpas(3000,2),shrtcnt,passrec(4000,2)
      common /spike/ desdata(400,27),upper,lower,spkvar
      common /intb/ intpdata(400,27)
c
c------------------------------------ program description
c
c       this program takes data in the 2-integers and 27-reals format
c       and reorders the entire dataset into a sorted file according
c       to the variable that the user chooses.  that variable is usually
c       the average longitude of each individual pass, such that after
c       reordering, the dataset will have all passes arranged from LOWEST
c       average longitude (-179.99) to the HIGHEST average longitude
c       (+179.99).  if the dataset crosses the -180.0 180.0 meridian,
c       then the eastern (negative) longitudes are incremented to a
c       positive value by adding 360.0 (see write statement with
c       variable "cross").  the sorting variable can also be the
c       average elevation or the pass numbers of each individual pass
c       (sorting by pass numbers = sorting by time).  this program takes
c       a little time (about 15 minutes on the DECstation 3100).  the
c       program requires DIRECT ACCESS or else it just won't happen!
c       NOTE:  as crazy as it may seem, real*8 is necessary for the
c              averages because i found two dusk longitude averages to be
c              EXACTLY the same with real*4.  if two averages are the same
c              then passno(4000) will have the same pass twice which messes
c              up subroutine reorder2.
c       NOTE:  i usually try to keep all file reads and writes in the
c              main program but reorder2 and reorder3 are a deviation
c              from the rule
c       NOTE:  the dataset must be despiked and interpolated to
c              correctly calculate the longitude averages of the
c              extended passes.  however, i prefer to not write out
c              the despiked or interpolated data because this program
c              represents the end of the first processing step, after-
c              which the data is ready for more involved processing
c              (ie. correlation filtering, bandpassing ...).
c              therefore, the output from reorder should be the
c              original data, only reordered.  get it??
c       NOTE:  for direct access on an ibm rs6000, recl=116.
c              on a dec3100, recl=29
c
c     program date:  16 apr 91
c
      write (*,*) 'INPUT 2I-27R FILE:'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename, status='old',form='unformatted',
     >     access='direct',recl=116)
      write (*,*) '0 IF THIS IS A DUSK DATASET'
      write (*,*) '1 IF THIS IS A DAWN DATASET'
      read (*,*) dndk
      write (*,*) '1 TO AVERAGE AND REORDER ON LONGITUDE'
      write (*,*) '2 TO AVERAGE AND REORDER ELEVATION'
      write (*,*) '3 TO REORDER ON PASS NUMBER'
      read (*,*) choice
c
      if (choice .eq. 1) then
      write (*,*) 'OUTPUT FILE OF 2I-27R DATASET REORDERED'
      read (*,9990) filename
      open (20, file=filename,form='unformatted',access='DIRECT',
     >     recl=116)
      write (*,*) 'INTERMEDIATE I/O FILE NOT REORDERED'
      write (*,*) '------ DO NOT USE THIS FILE -------'
      read (*,9990) filename
      open (21, file=filename,form='unformatted',access='direct',
     >     recl=116)
      endif
      if (choice .gt. 1) then
      write (*,*) 'OUTPUT FILE OF 2I-27R DATASET REORDERED'
      read (*,9990) filename
      open (20, file=filename,form='unformatted',access='direct',
     >     recl=116)
```

```
      endif
      write (*,*) 'OUTPUT FILE OF PASS NUMBERS AND AVERAGED',
     >            ' SORTED VARIABLE'
      read (*,9990) filename
      open (22, file=filename,form='formatted')
c
      write (*,*) '0 IF YOU WANT ALL PASSES'
      write (*,*) '1 IF SOME PASSES NEED TO BE REMOVED'
      read (*,*) pchoice
      if (pchoice .eq. 1) then
         write (*,*) 'INPUT FILE OF PASSES YOU DO NOT WANT'
         read (*,9990) filename
         open (11, file=filename,form='formatted')
      endif
      write (*,*) 'WHAT IS THE MINIMUM NUMBER OF'
      write (*,*) 'OBSERVATIONS ALLOWABLE FOR EACH PASS (50)'
      read (*,*) mincheck
c
      write (*,*) '0 FOR NO DESPIKING OF DATA SET'
      write (*,*) '1 FOR DESPIKING ONCE'
      write (*,*) '2 FOR DESPIKING TWICE (this is the usual choice)'
      write (*,*) '3 ... AND SO ON'
      read (*,*) spknum
      if (spknum .gt. 0) then
         write (*,*) 'WHAT IS THE MAXIMUM nT: (1.0)'
         write (*,*) 'WHAT IS THE MINIMUM nT: (-1.0)'
         read (*,*) upper,lower
         write (*,*) 'WHICH VARIABLE TO DESPIKE: (23)'
         write (*,*) '1=LAT, 2=LONG,...23=totavgmag..25=resavgmag...'
         write (*,*) 'lat lon rad mlt invlat diplat bs bv x y z'
         write (*,*) 'bva xa ya za totfld xfld yfld zfld inc dec'
         write (*,*) 'totmag totavgmag resid resavgmag ringcur sec '
         read (*,*) spkvar
      endif
c---------------------------- the following if statement determines if
c                             the study area includes the -180.0 180.0
c                             longitude line.  for further comments see
c                             subroutine reorder1
      cross=0.0
      crosss=cross
      minchk=0
      if (choice .eq. 1) then
        write (*,*) 'WESTERN MOST LONGITUDE OF STUDY AREA'
        write (*,*) 'EASTERN MOST LONGITUDE OF STUDY AREA'
        write (*,*) '-180.0 to 180.0  NOT 0.0 to 360.0'
        read (*,*) west,east
        diffwe = west - east
        if (diffwe .gt. 0.0) then
           cross=360.0
           crosss=cross
           write (*,*) ' '
           write (*,*) 'the program has determined that this study'
           write (*,*) 'area crosses the 180.0, -180.0 meridian'
           write (*,*) ' '
        endif
        write (*,*) 'NORTHERN AND SOUTHERN MOST LATITUDES'
        write (*,*) '90.0 to -90.0  NOT 0.0 to 180.0'
        read (*,*) north,south
        write (*,*) 'PERCENT OF TOTAL LATITUDE LENGTH TO'
        write (*,*) 'TO BE CONSIDERED TO FIND SHORT PASSES (90.0)'
        read (*,*) percent
c-------------------------- percent is used to calculate the range that
c                           is used in subroutine shorts to determine
c                           if a pass is a short pass or a long pass.
c                           see shorts for more info.  also, since no
c                           passes go below or above 83.0 degrees, the
c                           program resets north and south if needed.
        if (north .gt. 83.0) north = 83.0
        if (south .lt. -83.0) south = -83.0
        totlat=abs(north-south)
        minchk=(int(((100.0-percent)/(2*100.0))*totlat)+1)*3
        percent=((100.0-percent)/(2*100.0))*totlat
        lolat=south+percent
        hilat=north-percent
      endif
c
      if (minchk .gt. mincheck) then
         mincheck=minchk
         write (*,*) 'minimum observation cut-off increased to',
     >               ' =',mincheck
      endif
c
      if (pchoice .eq. 1) then
         do 50 kk=1,4000
           read (11,*,end=55) nowant(kk)
 50      continue
 55      continue
         pchoice=kk-1
      endif
c
```

```
c--------------------------- the main program reads the data to find
c                            which 2i-27r lines belong to a specific pass
c                            number (idata(n,1)) and since it reads one line
c                            of the next pass it stores that line in
c                            memory.
c
      inrec=1
      outrec=0
      shrtcnt=0
      countall=0
      jstop=0
      passrem=0
      cntsame=0
c
      read (10,rec=inrec) (idata(1,i),i=1,2),(data(1,j),j=1,27)
 100  n=2
 105  inrec=inrec+1
      read (10,rec=inrec,err=110) (idata(n,i),i=1,2),(data(n,j),j=1,27)
      if (idata(n,1) .ne. idata(n-1,1)) go to 120
      n=n+1
      go to 105
 110  continue
      jstop=1
 120  continue
      do 130 i=1,2
      istore(i)=idata(n,i)
 130  continue
      do 140 i=1,27
      dstore(i)=data(n,i)
 140  continue
c
      countall=countall+1
      innum=n-1
      innumall=innum
c
c------------------------------------ if passes are NOT wanted, remove them
      if (pchoice .eq. 0) go to 145
      do 143 ii=1,pchoice
        if (idata(innum,1) .eq. nowant(ii)) then
           write (*,*) 'PASS NUMBER REMOVED ',nowant(ii),
     >                                         idata(innum,1)
           passrem=passrem+1
           go to 400
        endif
 143  continue
 145  continue
c
      if (innum .lt. mincheck) then
         write (*,9980) idata(innum,1),innum
 9980    format ('PASS REMOVED AT READ =',i6,' OBSERV COUNT =',i5)
         passrem=passrem+1
         go to 400
      endif
c------------------------------------ search for passes that cross from
c                                     -180.0 to 180.0 meridian
      imerpass=idata(1,1)
      call meridian (innum,imerpass)
c------------------------------------ despike the data if user chooses and
c                                     despike the number of times chosen
      if (spknum .eq. 0) go to 190
      cnter=0
 150  call despike (innum,denum)
      cnter=cnter+1
      innum=denum
      do 180 k=1,innum
         do 180 kk=1,27
            data(k,kk)=desdata(k,kk)
 180     continue
      if (cnter .lt. spknum) go to 150
c
 190  continue
      if (innum .lt. mincheck) then
         write (*,9981) idata(innum,1),innum
 9981    format ('PASS REMOVED AFTER DESPIKING =',i6,
     >           ' OBSERV COUNT =',i5)
         passrem=passrem+1
         go to 400
      endif
c
c------------------------------------- interpolate the dataset
      call interpl (dndk,innum,outnum)
      innum=outnum
      do 210 i=1,innum
         do 210 j=1,27
            data(i,j)=intpdata(i,j)
 210  continue
      if (innum .lt. mincheck) then
         write (*,9982) idata(1,1),innum
 9982    format ('PASS REMOVED AFTER INTERPOLATING =',i6,
     >           ' OBSERV COUNT =',i5)
```

```
         passrem=passrem+1
         go to 400
      endif
c
      if (choice .eq. 1) oif=21
      if (choice .gt. 1) go to 260
      do 250 i=1,innum
         outrec=outrec+1
         write (oif,rec=outrec) (idata(1,j),j=1,2),(data(i,j),j=1,27)
 250  continue
c
 260  continue
      cntsome=cntsome+1
      pntcnt(cntsome,2)=innum
      pntcnt(cntsome,1)=idata(1,1)
      pntall(cntsome,1)=idata(1,1)
      pntall(cntsome,2)=innumall
c
c------------------------------------ subroutine finds all short passes
      if (choice .eq. 1) call shorts (innum,hilat,lolat,dndk)
c
c------------------------------------ now call reorder1 which will find
c                                     the average longitude and elevation
c                                     (not radius) and store them in
c                                     aver(4000) as well as storing the
c                                     pass number and modified julian day for
c                                     the current pass.
      call reorder1 (innum,cntsome)
c------------------------------------ ok, now go back and get more passes
c                                     to average until done with the file
c
 400  continue
      do 410 i=1,2
         idata(1,i)=istore(i)
 410  continue
      do 420 i=1,27
         data(1,i)=dstore(i)
 420  continue
      if (jstop .eq. 1) go to 500
      go to 100
c
 500  continue
c
c------------------------------------ now sort the chosen variable
      call sort (cntsome,choice)
c
      do 550 i=1,cntsome
         do 530 j=1,cntsome
            if (choice .eq. 1) then
               if (aver(j,choice) .eq. ra(i)) then
                  passno(i)=passmjd(j,1)
                  passavg(i,1)=dble(passmjd(j,1))
                  passavg(i,2)=aver(j,choice)
                  write (22,*) (passmjd(j,ii),ii=1,2),
     >                         (aver(j,ii),ii=1,2)
                  go to 540
               endif
            elseif (choice .eq. 2) then
               if (aver(j,choice) .eq. ra(i)) then
                  write (22,*) (passmjd(j,ii),ii=1,2),
     >                         (aver(j,ii),ii=1,2)
                  passno(i)=passmjd(j,1)
                  go to 540
               endif
            elseif (choice .eq. 3) then
               if (passmjd(j,1) .eq. int(ra(i))) then
                  write (22,*) (passmjd(j,ii),ii=1,2),
     >                         (aver(j,ii),ii=1,2)
                  passno(i)=passmjd(j,1)
                  go to 540
               endif
            endif
 530     continue
 540     continue
 550  continue
c
c---------------------------- now reread the file in reorder2 and write
c                             it ordered according to the pass numbers
c                             given to reorder2.
c
      if (choice .eq. 1) then
         inf=21
         onf=20
      elseif (choice .gt. 1) then
         inf=10
         onf=20
      endif
      call reorder2 (cntsome,inf,onf)
c
c---------------------------- if sorting by average longitude, then must
```

```
c                              extend the shorter passes and calculate a
c                              new average.  see subroutine reorder3 for
c                              more information.
c
      if (choice .eq. 1) then
c
         call reorder3 (cntsome,dndk)
c
         do 600 i=1,cntsome
            do 610 j=1,shrtcnt
               if (savglon(j,1) .eq. passavg(i,1)) then
                  passavg(i,2)=savglon(j,2)
                  go to 620
               endif
 610        continue
 620        continue
            ra(i)=passavg(i,2)
 600     continue
c
         call sort (cntsome,4)
c
         write (22,*) ' '
         write (22,*) 'new reordering as follows'
         write (22,*) ' '
         do 640 i=1,cntsome
            do 650 j=1,cntsome
               if (ra(i) .eq. passavg(j,2)) then
                  passno(i)=int(passavg(j,1))
                  do 655 k=1,cntsome
                     if (passno(i) .eq. passmjd(k,1)) then
                        i22mjd=passmjd(k,2)
                        x22elev=aver(k,2)
                        go to 656
                     endif
 655              continue
 656              continue
                  write (22,*) int(passavg(j,1)),i22mjd,
     >                         passavg(j,2),x22elev
                  go to 660
               endif
 650        continue
 660        continue
 640     continue
         write (22,*) ' '
         write (22,*) shrtcnt,' short passes as follows'
         write (22,*) ' '
         do 680 i=1,shrtcnt
            write (22,*) (shrtpas(i,j),j=1,2)
 680     continue
c
         call reorder2 (cntsome,10,20)
c
      endif
c
 999  continue
      write (*,*) 'total passes read =',countall
      write (*,*) 'total passes written =',cntsome
      write (*,*) 'total passes removed =',passrem
      write (*,*) 'total passes considered to be short =',shrtcnt
      write (*,*) 'total records read for original input file=',inrec-1
      close (10)
      close (20)
      close (21)
      close (22)
      close (25)
      stop
      end
c
c
c
      subroutine meridian (innum,passnum)
      real data(400,27)
      integer idata(400,2),innum,passnum
      common data(400,27),idata(400,2)
c
      do 100 i=1,innum-1
         if (data(i,2) .lt. data(i+1,2)) then
            write (*,*) passnum,' CROSSES -180.0',data(i,2),data(i+1,2)
            do 150 ii=1,innum
               if (data(ii,2) .lt. 0.0) data(ii,2)=data(ii,2)+360.0
 150        continue
            go to 200
         endif
 100  continue
c
 200  continue
      return
      end
c
c
```

```
c
      subroutine reorder1 (nobs,num)
      real data(400,27)
      double precision nobss,along(4000),radd(4000),aalong,aavg,selev,
     >                 elev(4000),savg,aver(4000,2),cross
      integer nobs,num,idata(400,2),passmjd(4000,2),mjd,
     >        passnum
      common /order1/ aver(4000,2),passmjd(4000,2),cross
      common data(400,27),idata(400,2)
c
c---------------------------- subroutine description
c
c         reorder1 takes a given set of longitudes and elevations and
c         finds the average longitude and elevation for the set.  since
c         some longitudes cross FROM -180.0 TO +180.0 (that is, longitudes
c         always decrease unless crossing 180) it necessary to correct
c         the average to the more usual 360 method.  therefore the
c         dataset is ordered from westernmost longitude to eastern most.
c         NOTE: real*8 is necessary since on some rare occasions the
c               averages can be the same at real*4 precision.
c         NOTE: when the study area includes the -180.0 180.0 longitude
c               line (but does not include all other longitudes) it is
c               necessary to add 360.0 to the negative (or eastern)
c               longitudes so that eastern longitudes will be located after
c               the western longitudes.
c         NOTE: for datasets that are global (ie. polar datasets or the
c               whole blasted world) then variable 'west' should be input
c               as -180.0 and variable 'east' should be input as 180.0.
c               input as such will produce a map centered on 0.0
c               longitude.
c
      nobss=dble(nobs)
      passnum=idata(1,1)
      mjd=idata(1,2)
      do 50 n=1,nobs
         along(n)=dble(data(n,2))
         radd(n)=dble(data(n,3))
 50   continue
c
      aalong=0.0
      do 110 n=1,nobs-1
         if (along(n) .lt. along(n+1)) then
            write (*,*) passnum,' CROSSES -180.0 to 180.0',
     >                                    along(n),along(n+1)
            go to 130
         endif
 110  continue
      do 120 n=1,nobs
         aalong=aalong+along(n)
 120  continue
      aavg=aalong/nobss
      go to 150
c
 130  aalong=0.0
      do 135 n=1,nobs
         if (along(n) .lt. 0.0) then
            along(n)=along(n)+360.0
         endif
         aalong=aalong+along(n)
 135  continue
      aavg=aalong/nobss
c
 150  continue
      selev=0.0
      do 170 n=1,nobs
         elev(n)=radd(n)-6378.140
         selev=selev+elev(n)
 170  continue
      savg=selev/nobss
c
      if (aavg .gt. 180.0) aavg=aavg-360.0
      passmjd(num,1)=passnum
      passmjd(num,2)=mjd
      aver(num,1)=aavg + cross
      aver(num,2)=savg
c
 200  continue
      return
      end
c
c
c
      subroutine reorder2 (nlines,inf,onf)
      integer nrecord(4000),passno(4000),npoints(4000),
     >        idata(2),pntcnt(4000,2),passrec(4000,2),
     >        shrtpas(3000,2),shrtcnt,inrec2,inf,onf,
     >        pntall(4000,2)
      real data(27)
      common /order2/ passno(4000),pntcnt(4000,2),pntall(4000,2)
      common /shorty/ shrtpas(3000,2),shrtcnt,passrec(4000,2)
```

```
c
c----------------------- the basis for this subroutine was provided
c                         quite generously
c                         by:      Dr. D.R.H. O'Connell
c                                  Dept. of Geologcial Sci.
c                                  Ohio State University
c
c
c----------------------- determine which pass this point belongs to.
c
      if (inf .eq. 10) then
         do 20 i=1,nlines
            passrec(i,1)=passno(i)
            do 30 ii=1,nlines
               if (passno(i) .eq. pntall(ii,1)) then
                  npoints(i)=pntall(ii,2)
                  go to 35
               endif
 30         continue
 35         continue
 20      continue
      elseif (inf .eq. 21) then
         do 50 i=1,nlines
            passrec(i,1)=passno(i)
            do 40 ii=1,nlines
               if (passno(i) .eq. pntcnt(ii,1)) then
                  npoints(i)=pntcnt(ii,2)
                  go to 45
               endif
 40         continue
 45         continue
 50      continue
      endif
c
c----------------------- npoints = number of records to allocate
c----------------------- for each pass number. nrecord = the output file
c----------------------- record positions for each pass
      nrecord(1)=1
      passrec(1,2)=1
      do 60 i=2,nlines
         i1=i-1
         nrecord(i)=nrecord(i1)+npoints(i1)
         passrec(i,2)=nrecord(i)
 60   continue
c----------------------- read each data point
c
c     rewind (inf)
      inrec2=0
 70   inrec2=inrec2+1
      read (inf,rec=inrec2,err=90) (idata(j),j=1,2),(data(k),k=1,27)
c----------------------- determine the matching pass number and its
c----------------------- output record number by searching all pass numbers
      do 80 i=1,nlines
         if (idata(1) .eq. passno(i)) then
            write (onf,rec=nrecord(i))
     *                     (idata(j),j=1,2),(data(k),k=1,27)
c----------------------- increment output record number for this pass number
            nrecord(i)=nrecord(i)+1
c----------------------- read next data point
            goto 70
         endif
 80   continue
      go to 70
 90   continue
c
      write (*,*) inrec2 - 1,' TOTAL RECORDS READ FOR FILE',inf
      write (*,*) nrecord(nlines)-1,' TOTAL RECORDS WRITTEN',
     >               ' FOR FILE',onf
c----------------------- webe jammin
c
      return
      end
c
c
c
c
      SUBROUTINE SORT(N,choice)
      double precision ra(4000),aver(4000,2),rra,cross
      integer passmjd(4000,2),n,choice
      common /order1/ aver(4000,2),passmjd(4000,2),cross
      common /hsort/ ra(4000)
c-----------------------------this subroutine is written by the authors
c                              of:    Numerical Recipes (fortran);
c                                     The Art of Scientific Computing
c                                     Cambridge University Press
c                                     1989, p. 230
c                              the routine is referred to as "heapsort"
c     Copyright (C) 1986, 1992 Numerical Recipes Software
      if (choice .le. 2) then
         do 10 i=1,n
            ra(i)=aver(i,choice)
```

```
 10      continue
      elseif (choice .eq. 3) then
         do 30 i=1,n
            ra(i)=dble(passmjd(i,1))
 30      continue
      endif
c
      L=N/2+1
      IR=N
 100  CONTINUE
      IF(L.GT.1)THEN
         L=L-1
         RRA=RA(L)
      ELSE
         RRA=RA(IR)
         RA(IR)=RA(1)
         IR=IR-1
         IF(IR.EQ.1)THEN
            RA(1)=RRA
            RETURN
         ENDIF
      ENDIF
      I=L
      J=L+L
 200  IF(J.LE.IR)THEN
         IF(J.LT.IR)THEN
            IF(RA(J).LT.RA(J+1))J=J+1
         ENDIF
         IF(RRA.LT.RA(J))THEN
            RA(I)=RA(J)
            I=J
            J=J+J
         ELSE
            J=IR+1
         ENDIF
         GO TO 200
      ENDIF
      RA(I)=RRA
      GO TO 100
      END
c
c
c
      subroutine shorts (innum,hilat,lolat,dndk)
      integer innum,idata(400,2),dndk,shrtcnt,
     >          shrtpas(3000,2),passrec(4000,2)
      real data(400,27),hilat,lolat
      common data(400,27),idata(400,2)
      common /shorty/ shrtpas(3000,2),shrtcnt,passrec(4000,2)
c
c------------------------------ subroutine description
c         shorts determines if the pass is short.  a short pass is
c         a pass which does not extend above the northern-most
c         (hilat) or below the southern-most (lolat).
c
      if (dndk .eq. 0) then
         if (data(1,1) .gt. lolat  .or.
     >         data(innum,1) .lt. hilat) then
            shrtcnt=shrtcnt+1
            shrtpas(shrtcnt,1)=idata(1,1)
            shrtpas(shrtcnt,2)=innum
         endif
      elseif (dndk .eq. 1) then
         if (data(innum,1) .gt. lolat  .or.
     >         data(1,1) .lt. hilat) then
            shrtcnt=shrtcnt+1
            shrtpas(shrtcnt,1)=idata(1,1)
            shrtpas(shrtcnt,2)=innum
         endif
      endif
c
      return
      end
c
c
c
      subroutine reorder3 (allcnt,dndk)
      integer shrtpas(3000,2),shrtcnt,passrec(4000,2),
     >          allcnt,passnum,recnum,row,recnt,rrow,frow,
     >          isdata(400,2),frec,nrec,rrec,pass,ifdata(400,2),
     >          irdata(400,2),numcnt,minrrow,stocount,minfrow,
     >          dndk,fcnt,rcnt
      real sdata(400,2),fdata(400,2),rdata(400,2),sftdat(400,2),
     >       ftdata(400,2),rtdata(400,2),totfdiff,totrdiff,
     >       fdiff,rdiff,totsft,totsrt,srtdat(400,2)
      double precision avgslon,avgrdiff,avgfdiff,savglon(4000,2),crosss
      common /shorty/ shrtpas(3000,2),shrtcnt,passrec(4000,2)
      common /trunc/ sdata(400,2),fdata(400,2),rdata(400,2),
     >               sftdat(400,2),ftdata(400,2),srtdat(400,2),
     >               rtdata(400,2)
```

```
      common /order3/ savglon(4000,2),crosss
c
c---------------------------------------------subroutine description
c        this subroutine extends all short passes by adding or subtracting
c        the average longitude difference from the closest full length
c        pass, then calculates the true average longitude for the short
c        pass, and stores that true average in an array.  the short passes
c        are found by subroutine shorts with a user defined percentage of total
c        length to be considered as a long pass.  as this percentage is
c        increased the distance to the closest pass also increases such that
c        there is more chance for error due to a poor fit of the average
c        longitude.  the closest full length pass is either west or east
c        of the short pass.  if west, then the average difference is added
c        to the full length pass, however, if east then the average
c        difference is subtracted.  these values are then added to the
c        longitudes of the short pass to create a set of full length
c        longitudes which are averaged together to get the true average
c        longitude.  the fundamental principle involved here is that passes
c        are always parallel and do not actually cross over each other.
c        therfore, the difference between adjacent passes remains almost
c        -or at least pretty dog-gone close to almost- constant.
c
c
      icnt=1
 50   continue
      do 100 i=1,allcnt
         if (shrtpas(icnt,1) .eq. passrec(i,1)) then
            recnum=passrec(i,2)
            passnum=passrec(i,1)
            row=shrtpas(icnt,2)
            recnt=i
            go to 200
         endif
 100  continue
c
 200  continue
c---------------------------- read in the short pass
      do 220 i=1,row
         read (20,rec=recnum) (isdata(i,j),j=1,2),(sdata(i,j),j=1,2)
         recnum=recnum+1
 220  continue
      recnum=recnum-1
      if (passnum .ne. isdata(row,1)) then
         write (*,*) 'wrong pass number in reorder3'
         stop
      endif
c--------------------------- add 360.0 to longitudes that cross the
c                            -180.0 to 180.0 meridian so that averages
c                            are correct
      do 230 n=1,row-1
        if (sdata(n,2) .lt. sdata(n+1,2)) then
           do 235 i=1,row
              if (sdata(i,2) .lt. 0.0) sdata(i,2)=sdata(i,2)+360.0
 235       continue
           go to 238
        endif
 230  continue
 238  continue
c
c---------------------------- find the starting record number for the
c                             nearest east pass
      frec=0
      numcnt=recnt+1
      if (numcnt .gt. allcnt) go to 270
      nrec=passrec(numcnt,2)
 240  continue
      read (20,rec=nrec) pass
      do 260 i=1,shrtcnt
         if (pass .eq. shrtpas(i,1)) then
            numcnt=numcnt+1
            if (numcnt .gt. allcnt) go to 270
            nrec=passrec(numcnt,2)
            go to 240
         endif
 260  continue
      frec=nrec
c
c-------------------------------- find the starting record number for the
c                                 nearest west pass
 270  continue
      rrec=0
      numcnt=recnt-1
      if (numcnt .lt. 1) go to 300
      nrec=passrec(numcnt,2)
 280  continue
      read (20,rec=nrec) pass
      do 290 i=1,shrtcnt
         if (pass .eq. shrtpas(i,1)) then
            numcnt=numcnt-1
            if (numcnt .lt. 1) go to 300
```

```
          nrec=passrec(numcnt,2)
          go to 280
        endif
 290  continue
      rrec=nrec
c
c------------------------------------ calculate the average longitude
 300  continue
      avgfdiff=10000.0
      if (frec .gt. 0) then
         i=1
         read (20,rec=frec,err=340) (ifdata(i,j),j=1,2),
     >                              (fdata(i,j),j=1,2)
         i=i+1
         frec=frec+1
 320     read (20,rec=frec,err=340) (ifdata(i,j),j=1,2),
     >                              (fdata(i,j),j=1,2)
         if (ifdata(i-1,1) .ne. ifdata(i,1)) go to 340
         frec=frec+1
         i=i+1
         go to 320
 340     continue
         frow=i-1
         do 344 n=1,frow-1
            if (fdata(n,2) .lt. fdata(n+1,2)) then
               do 342 i=1,frow
                  if (fdata(i,2) .lt. 0.0) fdata(i,2)=fdata(i,2)+360.0
 342           continue
               go to 345
            endif
 344     continue
 345     continue
c------------------------------ truncate the short and forward passes
c                               to the same length
c
         call truncate (row,frow,dndk,minfrow,stocount,passnum,
     >                  ifdata(frow,1),1)
         totfdiff=0.0
         totsft=0.0
         do 350 i=1,minfrow
            fdiff=abs(sftdat(i,2)-ftdata(i,2))
            totfdiff=fdiff+totfdiff
            totsft=totsft+sftdat(i,2)
 350     continue
c------------------------------ calculate the average longitude difference
         avgfdiff=dble(totfdiff/real(minfrow))
      endif
c
c------------------------------ repeat the process for the closest west pass
      avgrdiff=10000.0
      if (rrec .gt. 0) then
         i=1
         read (20,rec=rrec) (irdata(i,j),j=1,2),(rdata(i,j),j=1,2)
         i=i+1
         rrec=rrec+1
 360     read (20,rec=rrec) (irdata(i,j),j=1,2),(rdata(i,j),j=1,2)
         if (irdata(i-1,1) .ne. irdata(i,1)) go to 380
         rrec=rrec+1
         i=i+1
         go to 360
 380     continue
         rrow=i-1
         do 384 n=1,rrow-1
            if (rdata(n,2) .lt. rdata(n+1,2)) then
               do 382 i=1,rrow
                  if (rdata(i,2) .lt. 0.0) rdata(i,2)=rdata(i,2)+360.0
 382           continue
               go to 385
            endif
 384     continue
 385     continue
         call truncate (row,rrow,dndk,minrrow,stocount,passnum,
     >                  irdata(rrow,1),-1)
         totrdiff=0.0
         totsrt=0.0
         do 390 i=1,minrrow
            rdiff=abs(srtdat(i,2)-rtdata(i,2))
            totrdiff=rdiff+totrdiff
            totsrt=totsrt+srtdat(i,2)
 390     continue
         avgrdiff=dble(totrdiff/real(minrrow))
      endif
c
c------------------------------------- if the east difference is the smallest
c                                      then use the east pass to calculate
c                                      average longitude of the extended short
c                                      pass
      if (avgfdiff .lt. avgrdiff) then
         do 400 i=1,frow
            if (ftdata(i,2) .eq. fdata(i,2)) then
```

```
            fcnt=i
            go to 410
         endif
 400    continue
 410    continue
        if (sftdat(1,2) .lt. ftdata(1,2)) then
           if (fcnt-1 .eq. 0) go to 425
           do 420 i=1,fcnt-1
              totsft=totsft+(fdata(i,2)-real(avgfdiff))
 420       continue
 425       continue
           if (minfrow+fcnt .eq. frow) go to 435
           do 430 i=minfrow+fcnt,frow
              totsft=totsft+(fdata(i,2)-real(avgfdiff))
 430       continue
 435       continue
        elseif (sftdat(1,2) .gt. ftdata(1,2)) then
           if (fcnt-1 .eq. 0) go to 445
           do 440 i=1,fcnt-1
              totsft=totsft+(fdata(i,2)+real(avgfdiff))
 440       continue
 445       continue
           if (minfrow+fcnt .eq. frow) go to 455
           do 450 i=minfrow+fcnt,frow
              totsft=totsft+(fdata(i,2)+real(avgfdiff))
 450       continue
 455       continue
        endif
        avgslon=dble(totsft)/dble(frow)
c
c------------------------------------ repeat the process if the west pass is
c                                     the closest
      elseif (avgrdiff .lt. avgfdiff) then
        do 500 i=1,rrow
           if (rtdata(1,2) .eq. rdata(i,2)) then
              rcnt=i
              go to 510
           endif
 500    continue
 510    continue
        if (srtdat(1,2) .lt. rtdata(1,2)) then
           if (rcnt-1 .eq. 0) go to 525
           do 520 i=1,rcnt-1
              totsrt=totsrt+(rdata(i,2)-real(avgrdiff))
 520       continue
 525       continue
           if (minrrow+rcnt .eq. rrow) go to 535
           do 530 i=minrrow+rcnt,rrow
              totsrt=totsrt+(rdata(i,2)-real(avgrdiff))
 530       continue
 535       continue
        elseif (srtdat(1,2) .gt. rtdata(1,2)) then
           if (rcnt-1 .eq. 0) go to 545
           do 540 i=1,rcnt-1
              totsrt=totsrt+(rdata(i,2)+real(avgrdiff))
 540       continue
 545       continue
           if (minrrow+rcnt .eq. rrow) go to 555
           do 550 i=minrrow+rcnt,rrow
              totsrt=totsrt+(rdata(i,2)+real(avgrdiff))
 550       continue
 555       continue
        endif
        avgslon=dble(totsrt)/dble(rrow)
      endif
c
c------------------------store the average in array
      if (avgslon .gt. 180.0) avgslon = avgslon - 360.0
      savglon(icnt,1)=dble(passnum)
      savglon(icnt,2)=avgslon + crosss
c
      icnt=icnt+1
      if (icnt .gt. shrtcnt) return
      go to 50
c
      end
c
c
c
      subroutine truncate (xrow,yrow,dndk,minrow,stocount,
     >                    xpassno,ypassno,fr)
      integer xrow,yrow,stocount,rowii,rowinc,minrow,fr,
     >        dndk,xpassno,ypassno
      real xdata(400,2),ydata(400,2),sftdat(400,2),ftdata(400,2),
     >     adata,bdata,diffab,abss,
     >     rtdata(400,2),srtdat(400,2)
      common /trunc/ sdata(400,2),fdata(400,2),rdata(400,2),
     >               sftdat(400,2),ftdata(400,2),srtdat(400,2),
     >               rtdata(400,2)
c
```

```
c------------------------------- subroutine description
c        truncate compares the input passes and truncates both
c        passes to the same overlapping length.
c
      do 20 j=1,xrow
         do 20 jj=1,2
            xdata(j,jj)=sdata(j,jj)
 20   continue
      if (fr) 50,50,60
 50   do 55 j=1,yrow
         do 55 jj=1,2
            ydata(j,jj)=rdata(j,jj)
 55   continue
      go to 80
 60   do 65 j=1,yrow
         do 65 jj=1,2
            ydata(j,jj)=fdata(j,jj)
 65   continue
c
 80   continue
      stocount=0
      jj=1
      rowii=xrow
      rowinc=yrow
c-----------------------loops from 90 to 200 increment through the
c                        two input passes and truncate the lengths
c                        to the same length
 90   continue
      adata=xdata(jj,1)
      bdata=ydata(jj,1)
      diffab=adata-bdata
      abss=abs(diffab)
      if (rowii .eq. 0  .or.  rowinc .eq. 0) then
c------------------- if this happens, then the findgap subroutine from
c                     movetrunc will have to be implemented to remove
c                     the appropriate pass.  so far, there hasn't been
c                     any problems.  another alternative would be to
c                     use only the east or west pass instead of
c                     comparing both to the short pass.
c
         write (*,*) 'xrows (s) =',rowii,' yrows (',fr,') =',rowinc
         write (*,*) 'big problem with pass number =',xpassno,ypassno
         write (*,*) 'xrow =',xrow,' yrow =',yrow
         stop
      endif
      minrow=min(rowii,rowinc)
c-----------------------if pass a (ii) matches pass b (inc) at
c                        beginning length then write to xdata and
c                        ydata and race to main program
      if (abss .lt. 0.33) then
         if (fr .eq. -1) then
            do 110 ll=1,minrow
               do 110 kk=1,2
                  srtdat(ll,kk)=xdata(ll,kk)
                  rtdata(ll,kk)=ydata(ll,kk)
 110        continue
         elseif (fr .eq. 1) then
            do 115 ll=1,minrow
               do 115 kk=1,2
                  sftdat(ll,kk)=xdata(ll,kk)
                  ftdata(ll,kk)=ydata(ll,kk)
 115        continue
         endif
         return
      endif
c-----------------------if pass a no matcha the b data then find new
c                        a or b depending on whether or not ascending
c                        or descending order of independent variable
      if (abss .ge. 0.33) then
         stocount=stocount+1
c-----------------------if this is a dusk pass then will count from
c                        -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (xdata(jj,1) .gt. ydata(jj,1)) then
               rowinc=rowinc-1
               do 130 mm=1,rowinc
                  do 130 kk=1,2
                     ydata(mm,kk)=ydata(mm+1,kk)
 130              continue
            elseif (xdata(jj,1) .lt. ydata(jj,1)) then
               rowii=rowii-1
               do 150 nn=1,rowii
                  do 150 kk=1,2
                     xdata(nn,kk)=xdata(nn+1,kk)
 150              continue
             endif
c-----------------------if this is a dawn pass then will count from
c                        the equator toward the south pole
c                        that is decreasing independent variable
         elseif (dndk .eq. 1) then
```

```
         if (xdata(jj,1) .lt. ydata(jj,1)) then
            rowinc=rowinc-1
            do 160 mm=1,rowinc
               do 160 kk=1,2
                  ydata(mm,kk)=ydata(mm+1,kk)
 160           continue
         elseif (xdata(jj,1) .gt. ydata(jj,1)) then
            rowii=rowii-1
            do 170 nn=1,rowii
               do 170 kk=1,2
                  xdata(nn,kk)=xdata(nn+1,kk)
 170           continue
         endif
      endif
   endif
c
   go to 90
c
   end
c
c
c
c
   subroutine interp1 (dndk,num,ii)
   real data(400,27),xdata(27),intpdata(400,27)
   integer num,idata(400,2),dndk
   common data(400,27),idata(400,2)
   common /inta/ xdata(27)
   common /intb/ intpdata(400,27)
c
c------------------------------------- subroutine description
c
c       the basic concept of this subroutine was provided by:
c                       Dr. D.N. "Tiku" Ravat
c                       Dept. of Geology
c                       Purdue University
c       this subroutine linearly interpolates ALL 27-r variables by
c       basing the interpolation on the latitudes which are interpolated
c       at every 0.33 degrees of starting latitude.
c
   ii=0
   xlat=real(int(data(1,1)*100.0))/100.0
   if (dndk .eq. 0) then
      if (xlat .lt. data(1,1)) xlat=xlat + 0.33
      i=1
 100  if (xlat.ge.data(i,1) .and. xlat.le.data(i+1,1)) then
         call interp2 (i,xlat)
         xdata(2)=real(int(xdata(2)*100.0))/100.0
         ii=ii+1
         do 150 j=1,27
            intpdata(ii,j)=xdata(j)
 150        continue
         xlat=xlat + 0.33
         if (xlat .gt. data(num,1)) return
         go to 100
      elseif (xlat .gt. data(i+1,1)) then
         i=i+1
         go to 100
      endif
c
   elseif (dndk .eq. 1) then
      if (xlat .gt. data(1,1)) xlat=xlat - 0.33
      i=1
 180  if (xlat.le.data(i,1) .and. xlat.ge.data(i+1,1)) then
         call interp2 (i,xlat)
         xdata(2)=real(int(xdata(2)*100.0))/100.0
         ii=ii+1
         do 200 j=1,27
            intpdata(ii,j)=xdata(j)
 200        continue
         xlat=xlat - 0.33
         if (xlat .lt. data(num,1)) return
         go to 180
      elseif (xlat .lt. data(i+1,1)) then
         i=i+1
         go to 180
      endif
   endif
c
   end
c
c
c
   subroutine interp2 (inum,xlat)
   real data(400,27),diffdata(27),xdata(27)
   integer inum,idata(400,2)
   common data(400,27),idata(400,2)
   common /inta/ xdata(27)
c
c---------------------------- this subroutine is also from Tiku and is
```

```
c                                the interpolator (not to confused with the
c                                terminator!)
c
      do 100 i=1,27
         diffdata(i)=data(inum,i)-data(inum+1,i)
 100     continue
      do 120 i=1,27
         xdata(i)=data(inum,i)+(xlat-data(inum,1))*
     >                     (diffdata(i)/diffdata(1))
 120     continue
c
      return
      end
c
c
c
      subroutine despike (npts,outnum)
      real data(400,27),desdata(400,27),upper,lower
      integer ic(400),outnum,var,idata(400,2)
      common data(400,27),idata(400,2)
      common /spike/ desdata(400,27),upper,lower,var
c
c--------------------------- this subroutine also provided by Tiku
c
c
c       PROGRAM DESPIKE
C*****************************************************************
C    PROGRAM DESPIKE REMOVES MOST SPIKES FROM THE INPUT DATA SET.
C    HOWEVER, FOR BEST RESULTS, IT IS SUGGESTED TO RUN DESPIKE
C    AT LEAST THREE TIMES --- FOR EXAMPLE:
C
C                   INPUT1   ---DESPIKE---> OUTPUT1
C        (OUTPUT1 = INPUT2)  ---DESPIKE---> OUTPUT2
C        (OUTPUT2 = INPUT3)  ---DESPIKE---> OUTPUT3.
C
C    STILL, AFTER RUNNING DESPIKE THREE TIMES, IT FAILS TO ELIMINATE
C    ORBITS WITH DISCONTINUOUS RESID VS LATITUDE PROFILES.
C    PROGRAM DEGAP ATTEMPTS TO TAKE CARE OF SUCH PASSES.
C
C
C    PARAMETERS TO CHECK:  "UPPER" AND "LOWER" (IN NANOTESLAS):
C
C    IF PROGRAM DESPIKE HAS DETERMINED OBSERVATION  N  TO BE
C    A GOOD POINT, IT THEN SETS OUT TO DETERMINE IF POINT  N+1
C    IS A GOOD POINT.  IT DOES THIS BY CHECKING THE POINTS
C    N, N+1, AND N+2.  OBSERVATION  N+1  WILL BE A GOOD POINT
C    IF THE RESIDUAL DIFFERENCE BETWEEN POINT  N+1  AND THE
C    POINT ABOVE IT (N OR N+2) IS LESS THAN  "UPPER"  AND
C    IF THE RESIDUAL DIFFERENCE BETWEEN IT AND THE POINT
C    BELOW IT (N+2 OR N) IS GREATER THAN  "LOWER".
C*****************************************************************
C
      DO 2 I=1,400
         IC(I)=1
    2 CONTINUE
C
C*****************************************************************
C  ARE THE FIRST NEW POINTS SPIKES?
C  NOTE:  DATA(U,23) = RESID2(U)
C
      I=1
   15 SL1=(DATA(I+1,var)-DATA(I,var))
      SL2=(DATA(I+2,var)-DATA(I,var))
      SL3=(DATA(I+3,var)-DATA(I,var))
      SL4=(DATA(I+4,var)-DATA(I,var))
      XSL=ABS(SL1+SL2+SL3+SL4)/4.0
C
      S2=(DATA(I+2,var)-DATA(I+1,var))
C
      IF(ABS(SL1).GT.(3.0*XSL).OR.ABS(SL1).GT.(ABS(3.0*S2))) IC(I)=0
      IF(IC(I).EQ.0) THEN
         I=I+1
         GO TO 15
      ENDIF
C
C*****************************************************************
C  ARE THE MID POINTS SPIKES?
C
      DO 20 J=I,NPTS-2
         SL2=(DATA(J+1,var)-DATA(J+2,var))
         IF(SL1.GT.UPPER.AND.SL2.LT.LOWER) IC(J+1)=0
         IF(SL1.LT.LOWER.AND.SL2.GT.UPPER) IC(J+1)=0
         SL1=SL2
   20 CONTINUE
C
C*****************************************************************
C  IS THE LAST POINT A SPIKE?
C
      K=NPTS-2
   25 IF(IC(K).EQ.0) THEN
         K=K-1
```

```
         GO TO 25
      ENDIF
c
      SL1=ABS(DATA(K,var)-DATA(NPTS-1,var))
      SL2=ABS(DATA(NPTS-1,var)-DATA(NPTS,var))
      SL3=ABS(DATA(K,var)-DATA(NPTS,var))
         IF(IC(NPTS-1).EQ.0) THEN
            IF(SL1.GT.(3.0*SL2)) IC(NPTS)=0
            IF(SL3.GT.(3.0*UPPER)) IC(NPTS)=0
         ENDIF
         IF(IC(NPTS-1).EQ.1) THEN
            IF(SL2.GT.(3.0*SL1)) IC(NPTS)=0
         ENDIF
c
c     NOBS=0
c     DO 30 I=1,NPTS
c        IF(IC(I).EQ.1) NOBS=NOBS+1
c  30 CONTINUE
c     WRITE(6,*) IDATA(1,1), NOBS
c
      outnum=0
      DO 35 I=1,NPTS
         IF(IC(I).EQ.1) THEN
            outnum=outnum+1
            do 32 m=1,27
               desdata(outnum,m)=data(i,m)
 32            continue
         ENDIF
 35   CONTINUE
c
      return
      END
```

```
      program massage
      character*80 filename
      character*10 dndk
      real data(400,27),intpdata(400,27),movdata(400,27),
     >     dstore(27),desdata(400,27),upper,lower,mean,
     >     varmax,mincc,strintdat(400)
      integer idata(400,2),countall,innum,outnum,winlen2,incwen2,
     >         istore(2),denum,cnter,spknum,spkvar,movvar,choice,
     >         vary,nowant(4000),col,eight,zero,gfchoice,
     >         col3,mincheck,gftype,passrem,recnum,gfcnt,winlen1,
     >         winlen2a,winlen2b
      common data(400,27)
      common /intb/ intpdata(400,27)
      common /move/ movdata(400,27),winlen2,incwen2,movvar,gftype,
     >              varmax,mincc,incwen1,winlen1,winlen2a,winlen2b
      common /spike/ desdata(400,27),upper,lower,spkvar
      common /trax/ x(400),y(400)
c
c---------------------------- program description
c
c        ok, first the name of the program.  to be frank, i just couldn't
c        think of shortened name for despike-moving-average-min-max-
c        cubic-spline-linear-interpolation... so i just called the
c        program "massage" because this program massages the data a bit!!
c        this program takes data in the 2-integers and 27-reals format
c        and after a bit of work writes the worked over data in either
c        2i-27r or the more usual format of a file of latitudes, longitudes
c        and radii separated by headers and a file of residuals separated
c        by headers.  the bit of work includes, 1) removing bad data points
c        characterized by a large change in magnitude from surrounding
c        data points. ie: removing "spikes".  2) fitting a moving average
c        to the despiked dataset.  3) interpolating every 0.33 degrees of
c        latitude on the moving average (guide function).  4) interpolating
c        the despiked dataset.  interpolating schemes are linear for
c        the initial run through on a pass, then the gf is a spline.
c        5) fit and remove a least squares core field from the data.
c        of course there are several variations on the above scheme which
c        can be figured out be reading the write (*,*) statements.
c        NOTE: if certain passes are NOT wanted, this program will
c              remove them from the processing.
c        NOTE: i've found that the bandpass filter works better than
c              removing a guide function or cubic spline, but u can
c              do as u like.
c        NOTE: for optimal results, make use of the least squares
c              core field removal as applied by massage.
c        NOTE: for ibm rs6000, recl=116.  for dec3100, recl=29
c
c       program date:  16 apr 91
c
      write (*,*) 'INPUT 2I-27R FILE'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename,status='old',form='unformatted',
     >      access='direct',recl=116)
c
      write (*,*) 'TYPE dawn OR dusk AS APPROPRIATE'
      read (*,9991) dndk
 9991 format (a10)
c
      write (*,*) '0 IF YOU WANT ALL PASSES'
      write (*,*) '1 IF CERTAIN PASSES NEED TO BE REMOVED'
      read (*,*) choice
      if (choice .eq. 1) then
         write (*,*) 'INPUT FILE OF PASSES YOU DO NOT WANT'
         read (*,9990) filename
         open (11, file=filename,status='old',form='formatted')
      endif
c
      write (*,*) '0 FOR CUBIC SPLINE AND DATA OUTPUT'
      write (*,*) '1 FOR ONLY CUBIC SPLINE OUTPUT'
      write (*,*) '2 FOR ONLY DATA OUTPUT (the usual choice)'
      read (*,*) gfchoice
      write (*,*) 'WHICH 27R VARIABLE TO BE WRITTEN TO FILE(S) (12)'
      write (*,*) '1=LAT, 2=LONG ,..12=bva,...23=totavgmag'
      write (*,*) 'lat lon rad mlt invlat  diplat  bs   bv    x   y  z'
      write (*,*) 'bva xa  ya  za  totfld  xfld    yfld zfld inc dec'
      write (*,*) 'totmag  totavgmag  resid  resavgmag  ringcur  sec '
      write (*,*) '0 IF YOU WANT 2I-27R OUTPUT'
      read (*,*) vary
      write (*,*) '1 FIT LEAST SQUARES CORE FIELD TO THIS VARIABLE'
      write (*,*) '0 DO NOT FIT CORE FIELD'
      write (*,*) 'choose 1'
      read (*,*) ixfit
      if (ixfit .eq. 1) then
         write (*,*) 'OUTPUT FILE FOR PASS NUMBERS AND X VALUES'
         read (*,9990) filename
         open (22, file=filename, form='formatted')
      endif
c
      if (gfchoice .eq. 0  .or.  gfchoice .eq. 2) then
```


PRECEDING PAGE BLANK NOT FILMED

B-32

```
         if (vary .eq. 0) then
            write (*,*) 'OUTPUT 2I-27R DATA FILE'
            read (*,9990) filename
            open (20, file=filename,form='unformatted')
         endif
         if (vary .gt. 0) then
            write (*,*) 'OUTPUT DATA FILE OF HEADERS AND VARIABLE'
            read (*,9990) filename
            open (20, file=filename,form='unformatted')
            write (*,*) 'OUTPUT DATA FILE OF HEADERS'
            write (*,*) 'AND INTERP-LATS,LONGS,RADII'
            read (*,9990) filename
            open (21, file=filename,form='unformatted')
         endif
      endif
c
 20   continue
      write (*,*) '0 FOR NO DESPIKING OF DATA SET'
      write (*,*) '1 FOR DESPIKING ONCE'
      write (*,*) '2 FOR DESPIKING TWICE (this is the usual choice)'
      write (*,*) '3 ... AND SO ON'
      read (*,*) spknum
      if (spknum .gt. 0) then
         write (*,*) 'WHAT IS THE MAXIMUM nT: (1.0)'
         write (*,*) 'WHAT IS THE MINIMUM nT: (-1.0)'
         read (*,*) upper,lower
         write (*,*) 'WHICH VARIABLE TO DESPIKE: (23)'
         write (*,*) '1=LAT, 2=LONG,..12=bva,...23=totavgmag'
         write (*,*) 'lat lon rad mlt invlat diplat bs bv x y z'
         write (*,*) 'bva xa ya za totfld xfld yfld zfld inc dec'
         write (*,*) 'totmag totavgmag resid resavgmag ringcur sec '
         read (*,*) spkvar
      endif
c
      if (gfchoice .lt. 2) then
         write (*,*) 'WHICH VARIABLE TO WORK WITH IN CUBIC SPLINE:(12)'
         write (*,*) '1=LAT, 2=LONG,..12=bva,...23=totavgmag'
         write (*,*) 'lat lon rad mlt invlat diplat bs bv x y z'
         write (*,*) 'bva xa ya za totfld xfld yfld zfld inc dec'
         write (*,*) 'totmag totavgmag resid resavgmag ringcur sec '
         read (*,*) movvar
         if (vary .eq. 0) then
            write (*,*) 'OUTPUT 2I-27R CUBIC SPLINE FILE'
            read (*,9990) filename
            open (23, file=filename,form='unformatted')
         endif
         if (vary .gt. 0) then
            write (*,*) 'OUTPUT FILE OF CUBIC SPLINE VARIABLE'
            read (*,9990) filename
            open (23, file=filename,form='unformatted')
            write (*,*) 'OUTPUT FILE OF CUBIC SPLINE HEADERS AND'
            write (*,*) 'AND INTERP-LATS,LONGS,RADII'
            read (*,9990) filename
            open (24, file=filename,form='unformatted')
            write (*,*) 'OUTPUT FILE OF DATA - CUBIC SPLINE'
            read (*,9990) filename
            open (28, file=filename,form='unformatted')
         endif
      write (*,*) 'OUTPUT STATISTICS FILE'
      read (*,9990) filename
      open (25, file=filename,form='formatted')
      write (25,*)'PASS        VAR        STDEV       MIN        MEAN       MAX'
      write (*,*) 'OUTPUT FILE OF TRACKS FITTED WITH A CUBIC SPLINE'
      read (*,9990) filename
      open (26, file=filename,form='unformatted')
      write (*,*) 'OUTPUT FILE OF TRACKS NOT FITTED'
      read (*,9990) filename
      open (27, file=filename,form='unformatted')
      write (*,*) 'TYPE OF CUBIC SPLINE TO APPLY TO DATA: (2)'
      write (*,*) '1 FOR A MOVING AVERAGE'
      write (*,*) '2 FOR A MIN-MAX-AVERAGE FINDER'
      read (*,*) gftype
      if (gftype .eq. 1) then
      incwen2=0
      winlen2=0
      winlen2a=0
      winlen2b=0
      write (*,*) 'WHAT IS THE LENGTH OF THE WINDOW:'
      read (*,*) winlen1
      write (*,*) 'HOW MANY POINTS TO INCREMENT WINDOW LOCATION:'
      write (*,*) 'should be equal to or greater than 1'
      read (*,*) incwen1
      elseif (gftype .eq. 2) then
      write (*,*) 'LENGTH OF THE first AVERAGING WINDOW:'
      read (*,*) winlen2a
      write (*,*) 'LENGTH OF THE min-max AVERAGING WINDOW:'
      read (*,*) winlen2
      write (*,*) 'LENGTH OF THE last AVERAGING WINDOW:'
      read (*,*) winlen2b
      write (*,*) 'HOW MANY POINTS TO SEARCH FROM AN ENDPOINT TO FIND'
```

```
      write (*,*) 'IF THE MIN OR MAX POINT SHOULD BE REMOVED'
      read (*,*) incwen2
      write (*,*) 'IF MINCC CAN NOT BE MATCHED THEN'
      write (*,*) 'WHAT IS THE LENGTH OF THE AVERAGING WINDOW'
      read (*,*) winlen1
      write (*,*) 'IF MINCC CAN NOT BE MATCHED THEN'
      write (*,*) 'HOW MANY POINTS TO INCREMENT WINDOW LOCATION:'
      write (*,*) 'should be equal to or greater than 1'
      read (*,*) incwen1
      endif
      write (*,*) 'MAXIMUM VARIANCE WITHOUT FITTING A CUBIC SPLINE'
      read (*,*) varmax
      write (*,*) 'MINIMUM CORRELATION COEFFICIENT OF CUBIC SPLINE TO'
      write (*,*) 'ORIGINAL DATA'
      read (*,*) mincc
      endif
      write (*,*) 'AND FINALLY - WHAT IS THE MINIMUM NUMBER OF'
      write (*,*) 'OBSERVATIONS ALLOWABLE FOR EACH PASS (50)'
      read (*,*) mincheck
c
      if (choice .eq. 1) then
         do 50 kk=1,4000
           read (11,*,end=55) nowant(kk)
 50      continue
 55      continue
         choice=kk-1
      endif
c
c--------------------------------- read the data and find all lines
c                                  for each individual pass
      noc=0
      nocgf=0
      gfcnt=0
      itercnt=0
      passrem=0
      countall=0
      jstop=0
      ifirstcnt=0
      iseccnt=0
      iendcnt=0
      i3cnt=0
      i4cnt=0
      i5cnt=0
      i6cnt=0
      i7cnt=0
      i8cnt=0
      ibigcnt=0
      iswitch=0
      ilowcnt=0
      recnum=1
      read (10,rec=recnum) (idata(1,i),i=1,2),(data(1,j),j=1,27)
 100  n=2
 105  recnum=recnum+1
      read (10,rec=recnum,err=110) (idata(n,i),i=1,2),(data(n,j),j=1,27)
      if (idata(n,1) .ne. idata(n-1,1)) go to 120
      n=n+1
      go to 105
 110  continue
      jstop=1
 120  continue
      do 130 i=1,2

istore(i)=idata(n,i)
 130  continue
      do 140 i=1,27
         dstore(i)=data(n,i)
 140  continue
c
      countall=countall+1
c------------------------------------ if passes are NOT wanted, remove them
      if (choice .eq. 0) go to 145
      do 63 ii=1,choice
        if (idata(1,1) .eq. nowant(ii)) then
c           write (*,*) 'PASS NUMBER REMOVED ',nowant(ii),idata(1,1)
           passrem=passrem+1
           go to 400
        endif
 63   continue
 145  continue
c
      innum=n-1
c
      if (innum .lt. mincheck) then
         write (*,9980) idata(1,1),innum
 9980    format ('PASS REMOVED AT READ =',i6,' OBSERV COUNT =',i5)
         passrem=passrem+1
         go to 400
      endif
c------------------------------------ search for passes that cross the
c                                     -180.0 180.0 meridian
```

```
      call meridian (innum)
c
c------------------------------------ despike the data if user chooses and
c                                      despike the number of times chosen
      if (spknum .eq. 0) go to 190
      cnter=0
 150  call despike (innum,denum)
      cnter=cnter+1
      innum=denum
      do 180 k=1,innum
         do 180 kk=1,27
            data(k,kk)=desdata(k,kk)
 180        continue
      if (cnter .lt. spknum) go to 150
c
 190  continue
      if (innum .lt. mincheck) then
         write (*,9981) idata(1,1),innum
 9981    format ('PASS REMOVED AFTER DESPIKING =',i6,
     >           ' OBSERV COUNT =',i5)
         passrem=passrem+1
         go to 400
      endif
c
c------------------------------------- interpolate the dataset
      call interpl (dndk,innum,outnum)
      innum=outnum
      if (innum .lt. mincheck) then
         write (*,9982) idata(1,1),innum
 9982    format ('PASS REMOVED AFTER INTERPOLATING =',i6,
     >           ' OBSERV COUNT =',i5)
         passrem=passrem+1
         go to 400
      endif
c
      do 205 i=1,innum
         x(i)=intpdata(i,1)
         y(i)=intpdata(i,2)
 205  continue
c------------------------------------- fit the core field (#16) profile to
c                                       the chosen variable profile
      if (ixfit .eq. 1) then
         call sqrfit(innum,xxx,vary)
         write (22,*) idata(1,1),xxx
      endif
c
      if (gfchoice .eq. 1) go to 300
      if (vary .eq. 0) go to 230
      col=1
      col3=3
      zero=0
      mean=0.0
      eight=8888
      write (20) outnum,col,zero,mean,idata(1,1),eight
      write (21) outnum,col3,zero,mean,idata(1,1),eight

      do 210 m=1,outnum
         if (intpdata(m,2).gt.180.0) intpdata(m,2)=intpdata(m,2)-360.0
         write (20) intpdata(m,vary)
         write (21) intpdata(m,1),intpdata(m,2),intpdata(m,3)
         strintdat(m)=intpdata(m,vary)
 210  continue
 230  if (vary .gt. 0) go to 300
      do 250 i=1,outnum
         if (intpdata(i,2).gt.180.0) intpdata(i,2)=intpdata(i,2)-360.0
         write (20) (idata(i,j),j=1,2),(intpdata(i,j),j=1,27)
 250  continue
c
c-------------------------------- this if statement if chosen, will fit
c                                 the cubic spline, interpolate it and
c                                 write it to file
 300  continue
      if (gfchoice .lt. 2) then
         do 310 k=1,innum
            do 310 kk=1,27
               data(k,kk)=intpdata(k,kk)
 310        continue
c
         call moving (innum,num,idata(1,1),eight,gfcnt,dndk,itercnt,
     >                ifirstcnt,iseccnt,iendcnt,i3cnt,i4cnt,i5cnt,
     >                i6cnt,i7cnt,i8cnt,ibigcnt,iswitch,ilowcnt)
         outnum=innum
c
         if (eight .eq. 7777) call track (27,innum,noc,nocgf)
c
         if (vary .gt. 0) then
            col=1
            col3=3
            zero=0
            mean=0.0
```

```
            if (eight .eq. 8888) then
               write (28) outnum,col,zero,mean,idata(1,1),eight
               write (23) outnum,col,zero,mean,idata(1,1),eight
               write (24) outnum,col3,zero,mean,idata(1,1),eight

               do m=1,outnum
                  write (28) (strintdat(m)-intpdata(m,vary))
               enddo
            elseif (eight .eq. 7777) then
               ieight=8888
               write (28) outnum,col,zero,mean,idata(1,1),ieight
               write (23) zero,col,zero,mean,idata(1,1),eight
               write (24) zero,col3,zero,mean,idata(1,1),eight
               do m=1,outnum
                  write (28) strintdat(m)
               enddo
               go to 400
            endif
c
            call track (26,outnum,noc,nocgf)
c
            do 350 m=1,outnum
               if (intpdata(i,2).gt.180.0)
     >                        intpdata(i,2)=intpdata(i,2)-360.0
               write (23) intpdata(m,vary)
               write (24) intpdata(m,1),intpdata(m,2),intpdata(m,3)
 350        continue
         endif
         if (vary .eq. 0) then
            if (eight .eq. 7777) then
               write (23) eight
               go to 400
            endif
c
            call track (26,outnum,noc,nocgf)
c
            do 370 m=1,outnum
               if (intpdata(i,2).gt.180.0)
     >                        intpdata(i,2)=intpdata(i,2)-360.0
               write (23) (idata(1,mm),mm=1,2),
     >                          (intpdata(m,mm),mm=1,27)
 370        continue
         endif
      endif
c
c----------------------------- set idata(1,i) and data(1,i) to the
c                              previously read values and go back and
c                              get the rest of the values for the pass
 400  continue
      do 410 i=1,2
         idata(1,i)=istore(i)
 410     continue
      do 420 i=1,27
         data(1,i)=dstore(i)
 420  continue
      if (jstop .eq. 1) go to 999
      go to 100
c
 999  continue
      write (*,*) 'records read =',recnum-1
      write (*,*) 'passes read =',countall
      write (*,*) 'passes removed =',passrem
      write (*,*) 'passes fitted with a cubic spline =',gfcnt
      write (*,*) 'passes without a cubic spline =',
     >                  countall-gfcnt-passrem
      write (*,*) 'total extra iterations =',itercnt-gfcnt
      if (gftype .eq. 2) then
         write (*,*) ifirstcnt,' first points were removed'
         write (*,*) iseccnt,' second points were removed'
         write (*,*) iswitch,' orbits used a moving-average gf'
      endif
      write (*,*) iendcnt,' passes were 2nd order'
      write (*,*) i3cnt,' passes were 3rd order'
      write (*,*) i4cnt,' passes were 4th order'
      write (*,*) i5cnt,' passes were 5th order'
      write (*,*) i6cnt,' passes were 6th order'
      write (*,*) i7cnt,' passes were 7th order'
      write (*,*) i8cnt,' passes were 8th order'
      write (*,*) ibigcnt,' passes were larger than 8th order'
      write (*,*) ilowcnt,' passes were below cc of',mincc
      close (10)
      close (11)
      close (20)
      close (21)
      close (23)
      close (24)
      close (25)
      close (26)
      close (27)
      stop
```

```
      end
c
c
c
      subroutine meridian (innum)
      real data(400,27)
      integer innum
      common data(400,27)
c-------------------------------- subroutine description
c       this subroutine determines if a pass crosses the -180.0 180.0
c       meridian.  if a pass does cross, then 360.0 is added to the
c       negative values so that the interpolation scheme does not
c       try to interpolate from -180.0 to 180.0 every 0.33 degrees
c
      do 100 i=1,innum-1
         if (data(i,2) .lt. data(i+1,2)) then
            do 150 ii=1,innum
               if (data(ii,2) .lt. 0.0) data(ii,2)=data(ii,2)+360.0
 150        continue
            go to 200
         endif
 100  continue
c
 200  continue
      return
      end
c
c
c
c
      subroutine interp1 (dndk,num,ii)
      real data(400,27),xdata(27),intpdata(400,27)
      integer num
      character*10 dndk
      common data(400,27)
      common /inta/ xdata(27)
      common /intb/ intpdata(400,27)
c
c------------------------------------ subroutine description
c
c         the basic concept of this subroutine was provided by:
c                        Dr. D.N. "Tiku" Ravat
c                        Dept. of Geology
c                        Purdue University
c         this subroutine linearly interpolates ALL 27-r variables by
c         basing the interpolation on the latitudes which are interpolated
c         at every 0.33 degrees of starting latitude.
c
      ii=0
      xlat=real(int(data(1,1)*100.0))/100.0
      if (dndk .eq. 'dusk') then
         if (xlat .lt. data(1,1)) xlat=xlat + 0.33
         i=1
 100     if (xlat.ge.data(i,1) .and. xlat.le.data(i+1,1)) then
            call interp2 (i,xlat)
            xdata(2)=real(int(xdata(2)*100.0))/100.0
            ii=ii+1
            do 150 j=1,27
               intpdata(ii,j)=xdata(j)
 150           continue
            xlat=xlat + 0.33
            if (xlat .gt. data(num,1)) return
            go to 100
         elseif (xlat .gt. data(i+1,1)) then
            i=i+1
            go to 100
         endif
c
      elseif (dndk .eq. 'dawn') then
         if (xlat .gt. data(1,1)) xlat=xlat - 0.33
         i=1
 180     if (xlat.le.data(i,1) .and. xlat.ge.data(i+1,1)) then
            call interp2 (i,xlat)
            xdata(2)=real(int(xdata(2)*100.0))/100.0
            ii=ii+1
            do 200 j=1,27
               intpdata(ii,j)=xdata(j)
 200           continue
            xlat=xlat - 0.33
            if (xlat .lt. data(num,1)) return
            go to 180
         elseif (xlat .lt. data(i+1,1)) then
            i=i+1
            go to 180
         endif
      endif
c
      end
c
c
```

```
c
      subroutine interp2 (inum,xlat)
      real data(400,27),diffdata(27),xdata(27)
      integer inum
      common data(400,27)
      common /inta/ xdata(27)
c
c--------------------------- this subroutine is also from Tiku and is
c                            the interpolator (not to confused with the
c                            terminator!)
c
      do 100 i=1,27
         diffdata(i)=data(inum,i)-data(inum+1,i)
 100     continue
      do 120 i=1,27
         xdata(i)=data(inum,i)+(xlat-data(inum,1))*
     >                 (diffdata(i)/diffdata(1))
 120     continue
c
      return
      end
c
c
c
      SUBROUTINE SPLINE(X,Y,N,YP1,YPN,Y2)
      PARAMETER (NMAX=100)
      DIMENSION X(N),Y(N),Y2(N),U(NMAX)
c
c--------------------------------- subroutine description
c       this subroutine provided by the authors of Numerical Recipes
c                          Numerical Recipes (fortran)
c                          The Art of Scientific Computing
c                          Cambridge University Press, 1989
c     Copyright (C) 1986, 1992 Numerical Recipes Software
      IF (YP1.GT..99E30) THEN
        Y2(1)=0.
        U(1)=0.
      ELSE
        Y2(1)=-0.5
        U(1)=(3./(X(2)-X(1)))*((Y(2)-Y(1))/(X(2)-X(1))-YP1)
      ENDIF
      DO 11 I=2,N-1
        SIG=(X(I)-X(I-1))/(X(I+1)-X(I-1))
        P=SIG*Y2(I-1)+2.
        Y2(I)=(SIG-1.)/P
        U(I)=(6.*((Y(I+1)-Y(I))/(X(I+1)-X(I))-(Y(I)-Y(I-1))
     *       /(X(I)-X(I-1)))/(X(I+1)-X(I-1))-SIG*U(I-1))/P
11    CONTINUE
      IF (YPN.GT..99E30) THEN
        QN=0.
        UN=0.
      ELSE
        QN=0.5
        UN=(3./(X(N)-X(N-1)))*(YPN-(Y(N)-Y(N-1))/(X(N)-X(N-1)))
      ENDIF
      Y2(N)=(UN-QN*U(N-1))/(QN*Y2(N-1)+1.)
      DO 12 K=N-1,1,-1
        Y2(K)=Y2(K)*Y2(K+1)+U(K)
12    CONTINUE
      RETURN
      END
c
c
c
      SUBROUTINE SPLINT(XA,YA,Y2A,N,X,Y)
      DIMENSION XA(N),YA(N),Y2A(N)
c--------------------------------- subroutine description
c     Copyright (C) 1986, 1992 Numerical Recipes Software
      KLO=1
      KHI=N
1     IF (KHI-KLO.GT.1) THEN
        K=(KHI+KLO)/2
        IF(XA(K).GT.X)THEN
          KHI=K
        ELSE
          KLO=K
        ENDIF
      GOTO 1
      ENDIF
      H=XA(KHI)-XA(KLO)
      IF (H.EQ.0.) PAUSE 'Bad XA input.'
      A=(XA(KHI)-X)/H
      B=(X-XA(KLO))/H
      Y=A*YA(KLO)+B*YA(KHI)+
     *      ((A**3-A)*Y2A(KLO)+(B**3-B)*Y2A(KHI))*(H**2)/6.
      RETURN
      END
c
c
c
      subroutine moving (nobs,inum,passnum,eight,gfcnt,dndk,
```

```
     >                    itercnt,ifirstcnt,iseccnt,iendcnt,
     >                    i3cnt,i4cnt,i5cnt,i6cnt,i7cnt,i8cnt,
     >                    ibigcnt,iswitch,ilowcnt)
      integer winlen,incwen,nobs,minm,
     >        wwinlen2,iincwen2,iincwen1,var,maxm,
     >        gftype,passnum,eight,gfcnt,subwinlen,strnobs,
     >        ggftype,wwinlen1,wwinlen2a,wwinlen2b
      real data(400,27),movdata(400,27),minval,maxval,
     >     varmax,strdata(400,27),intpdata(400,27),mincc,
     >     x(100),y(100),y2(100),
     >     sdata(400,27)
      character*10 dndk
      common data(400,27)
      common /move/ movdata(400,27),wwinlen2,iincwen2,var,ggftype,
     >              varmax,mincc,iincwen1,wwinlen1,wwinlen2a,
     >              wwinlen2b
      common /intb/ intpdata(400,27)
c
c---------------------------- subroutine description
c
c     subroutine moving creates the cubic spline fit of each pass with
c     a variance above the user defined limit.  this cubic spline will
c     match the original pass to within the user defined correlation
c     coefficient.  the cubic spline is the time domain representation
c     of the non-lithospheric components in the pass.  the following
c     source was used as a reference for the statistical calculations:
c
c             Davis, Statistics and Data Analysis in
c             Geology, 2nd ed., 1986 pp. 41
c
c------------------------loops that sum x, x**2
c
      strnobs=nobs
      xsum=0.0
      xsumsqr=0.0
      xmin=data(1,var)
      xmax=xmin
      do 10 j=1,nobs
         xsum=xsum+data(j,var)
         xsumsqr=xsumsqr+(data(j,var))**2
         xmin=min(xmin,data(j,var))
         xmax=max(xmax,data(j,var))
 10   continue
      nobss=float(nobs)
c------------------------find corrected sum of squares and mean
c
      xcsumsqr=xsumsqr-((xsum**2)/nobss)
      xmean=xsum/nobss
c------------------------find variance, standard deviation
c
      xvar=xcsumsqr/(nobss-1.0)
      xstdev=sqrt(xvar)
c------------------------write out this mess for individual pass
c
      write (25,9992) passnum,xvar,xstdev,xmin,xmean,xmax
 9992 format (i6,5f14.5)
c
c------------------------- if the variance of the pass is below the user
c                          defined limit then race on back to main program
      if (xvar .le. varmax) then
         eight=7777
         return
      endif
      gftype=ggftype
      gfcnt=gfcnt+1
      do 15 i=1,nobs
         do 15 j=1,27
            strdata(i,j)=data(i,j)
 15   continue
      subwinlen=0
c------------------------------ use a moving average fit
 17   if (gftype .eq. 1) then
         itercnt=itercnt+1
         winlen=wwinlen1
         incwen=iincwen1
         inum=1
         if (subwinlen .gt. 0) then
            if (incwen .gt. 1) then
               incwen=incwen-subwinlen
               go to 25
            elseif (incwen .lt. winlen) then
               winlen=winlen-subwinlen
               if (winlen .lt. 2) stop 1110
            endif
 25         continue
         endif
         xdivide=real(nobs)/real(incwen+winlen-1)
         xp5=(real(int(xdivide)))+0.5
         if (xdivide .lt. xp5) idivide=int(xdivide)
         if (xdivide .ge. xp5) idivide=int(xdivide)+1
```

```
        xwinlen=real(nobs)/real(idivide)
        iwinadd=int((xwinlen-(real(int(xwinlen))))*real(idivide))
        winlen=int(xwinlen)-incwen
        istrwinlen=winlen
        iadd=0
        if (iwinadd .gt. 0  .and.  iadd .lt. iwinadd) then

   iadd=iadd+1
           winlen=winlen+1
        endif
        i1=(winlen-1)/2
        i3=i1
        i7=0
        if (i1 .gt. nobs) then
           i1=nobs/2
           i7=1
        endif
        avgdat=0.0
        do j=1,i1
           avgdat=data(j,var)+avgdat
        enddo
        do 20 j=1,27
           movdata(inum,j)=data(1,j)
20      continue

movdata(inum,var)=avgdat/(real(i1))
        if (i7 .eq. 1) go to 100
30      continue
        winlen=istrwinlen
        if (iwinadd .gt. 0  .and.  iadd .lt. iwinadd) then

   iadd=iadd+1
           winlen=winlen+1
        endif
        i2=i3+incwen
        i3=i2+winlen-1
        i4=i2+((i3-i2)/2)
        if (i3 .gt. nobs-i1+1) go to 100
        avgdat=0.0
        inum=inum+1
        do 50 j=i2,i3
           avgdat=avgdat+data(j,var)
50      continue
        do 70 j=1,27
           movdata(inum,j)=data(i4,j)
70      continue
        movdata(inum,var)=avgdat/(i3-i2+1)

go to 30
100     continue

inum=inum+1
        avgdat=0.0
        do j=(nobs-i1+1),nobs
           avgdat=avgdat+data(j,var)
        enddo
150     do 170 j=1,27
           movdata(inum,j)=data(nobs,j)
170     continue
        movdata(inum,var)=avgdat/(real(i1))
c------------------------------------ or use the minimum and maximum values
c                                     within the window length
c
      elseif (gftype .eq. 2) then
         itercnt=itercnt+1
         winlen=wwinlen2
         incwen=iincwen2
         winlena=wwinlen2a
         winlenb=wwinlen2b
         if (winlena.gt.nobs .or. winlenb.gt.nobs .or.
     >      incwen.gt.nobs .or. (winlena+1).ge.(nobs-winlenb)) then
            do i=1,27
               movdata(1,i)=data(1,i)
               movdata(2,i)=data(nobs,i)
            enddo
            avgdat=0.0
            do i=1,nobs/2
               avgdat=data(i,var)+avgdat
            enddo
            avgdat=avgdat/real(nobs/2)
            movdata(1,var)=avgdat
            avgdat=0.0
            do i=(nobs/2)+1,nobs
               avgdat=data(i,var)+avgdat
            enddo
            avgdat=avgdat/real(nobs-(nobs/2))
            movdata(2,var)=avgdat
            iendcnt=iendcnt+1
            inum=2
            go to 700
```

```
      endif
      avgdat=0.0
      do 500 i=1,winlena
500
   avgdat=data(i,var)+avgdat
      avgdat=avgdat/real(winlena)
      do 510 i=1,27
510      movdata(1,i)=data(1,i)
      movdata(1,var)=avgdat
      i1=winlena+1
      i2=nobs-winlenb
      i3=0
      i4=nobs-winlenb+1
      i5=nobs
      i6=nobs
      maxval=-1.0e10
      minval=1.0e10
      do 520 m=i1,i2
         minval=min(minval,data(m,var))
         if (minval .eq. data(m,var)) minm=m
         maxval=max(maxval,data(m,var))
         if (maxval .eq. data(m,var)) maxm=m
520   continue
      ilow=minm-((winlen-1)/2)
      ihi=minm+((winlen-1)/2)
      if (ilow .lt. 1) ilow=1
      if (ihi .gt. nobs) ihi=nobs
      avgdatmin=0.0
      do 530 i=ilow,ihi
530      avgdatmin=avgdatmin+data(i,var)
      avgdatmin=avgdatmin/real(ihi-ilow+1)
      ilow=maxm-((winlen-1)/2)
      ihi=maxm+((winlen-1)/2)
      if (ilow .lt. 1) ilow=1
      if (ihi .gt. nobs) ihi=nobs
      avgdatmax=0.0
      do 540 i=ilow,ihi
540      avgdatmax=avgdatmax+data(i,var)
      avgdatmax=avgdatmax/real(ihi-ilow+1)
      if (minm .lt. maxm) then
         do 570 j=1,27
            movdata(2+i3,j)=data(minm,j)
            movdata(3+i3,j)=data(maxm,j)
            movdata(2+i3,var)=avgdatmin
            movdata(3+i3,var)=avgdatmax
570      continue
      elseif (minm .gt. maxm) then
         do 590 j=1,27
            movdata(2+i3,j)=data(maxm,j)
            movdata(3+i3,j)=data(minm,j)
            movdata(2+i3,var)=avgdatmax
            movdata(3+i3,var)=avgdatmin
590      continue
      endif
      avgdat=0.0
      do 594 i=i4,i5
594      avgdat=data(i,var)+avgdat
      avgdat=avgdat/real(i5-i4+1)
      do 596 i=1,27
596      movdata(4+i3,i)=data(i6,i)
      movdata(4+i3,var)=avgdat
      inum=4+i3
      if (dndk .eq. 'dusk') then
         if (movdata(2+i3,1) .le.
     >         (movdata(1+i3,1)+(real(incwen)*0.33))) then
            ifirstcnt=ifirstcnt+1
            do 600 j=2,3
               do 600 i=1,27
600               movdata(j+i3,i)=movdata(j+1+i3,i)
            inum=inum-1
         endif
         if (movdata(inum-1+i3,1) .ge.
     >         (movdata(inum+i3,1)-(real(incwen)*0.33))) then
            iseccnt=iseccnt+1
            do 610 i=1,27
610            movdata(inum-1+i3,i)=movdata(inum+i3,i)
            inum=inum-1
         endif
      elseif (dndk .eq. 'dawn') then
         if (movdata(2+i3,1) .ge.
     >            (movdata(1+i3,1)-(real(incwen)*0.33))) then
            ifirstcnt=ifirstcnt+1
            do 620 j=2,3
               do 620 i=1,27
620               movdata(j+i3,i)=movdata(j+1+i3,i)
            inum=inum-1
         endif
         if (movdata(inum-1+i3,1) .le.
     >         (movdata(inum+i3,1)+(real(incwen)*0.33))) then
            iseccnt=iseccnt+1
```

```
            do 630 i=1,27
630            movdata(inum-1+i3,i)=movdata(inum+i3,i)
            inum=inum-1
         endif
      endif
   endif
c
700   continue
      if (dndk .eq. 'dusk') then
         do 401 i=1,inum
            x(i)=movdata(i,1)
            y(i)=movdata(i,var)
401      continue
      elseif (dndk .eq. 'dawn') then
         do 404 i=1,inum
            ii=inum-i+1
            x(i)=movdata(ii,1)
            y(i)=movdata(ii,var)
404      continue
      endif
      oneslope=(y(2)-y(1))/(x(2)-x(1))
      twoslope=(y(inum)-y(inum-1))/(x(inum)-x(inum-1))
      call spline (x,y,inum,oneslope,twoslope,y2)
      do 402 i=1,nobs
         ii=nobs-i+1
         xint=data(i,1)
         if (dndk .eq. 'dawn') xint=data(ii,1)
         call splint (x,y,y2,inum,xint,yint)
         do 403 j=1,27
            intpdata(i,j)=data(i,j)
            if (dndk .eq. 'dawn') intpdata(i,j)=data(ii,j)
403      continue
         intpdata(i,var)=yint
402   continue
      if (dndk .eq. 'dawn') then
         do 405 i=1,nobs
            ii=nobs-i+1
            do 405 j=1,27
               sdata(i,j)=intpdata(ii,j)
405      continue
         do 406 i=1,nobs
            do 406 j=1,27
               intpdata(i,j)=sdata(i,j)
406      continue
      endif
c
c------------------------- calculate the correlation coefficient between
c                          the original data and the cubic spline
c
c-------------------------loops that sum x, x**2, y, y**2 and xy
      if (nobs .ne. strnobs) stop 0002
      nobss=real(nobs)
      xsum=0.0
      xsumsqr=0.0
      ysum=0.0
      ysumsqr=0.0
      sumxy=0.0
      do 440 j=1,nobs
         xsum=xsum+strdata(j,var)
         xsumsqr=xsumsqr+(strdata(j,var))**2
         ysum=ysum+intpdata(j,var)
         ysumsqr=ysumsqr+(intpdata(j,var))**2
         sumxy=sumxy+(strdata(j,var)*intpdata(j,var))
440   continue
c------------------------find corrected sum of products, covariance
c                        and corrected sum of squares (x) (y)
c
      sumprod=sumxy-((xsum*ysum)/nobss)
      covarxy=sumprod/(nobss-1.0)
      xcsumsqr=xsumsqr-((xsum**2)/nobss)
      ycsumsqr=ysumsqr-((ysum**2)/nobss)
c
c------------------------find variance, standard deviation for x and y
c
      xvar=xcsumsqr/(nobss-1.0)
      yvar=ycsumsqr/(nobss-1.0)
      xstdev=sqrt(xvar)
      ystdev=sqrt(yvar)
c-----------------------find correlation coefficient by Davis method
      corrDxy=covarxy/(xstdev*ystdev)
c
c----------------------- if the fit between cubic spline and original data
c                        is below the minimum acceptable user defined
c                        correlation coefficient limit, then shorten the
c                        windows by a length equivalent to subwinlen.  then
c                        rerun the entire subroutine from statement 17
c
      if (corrDxy .lt. mincc .and. gftype .eq. 2) then
         write (*,*) passnum,' pass below cc limit',corrDxy
         gftype=1
```

```
         iswitch=iswitch+1
         do 450 i=1,nobs
            do 450 j=1,27
               data(i,j)=strdata(i,j)
 450     continue
         go to 17
      endif
      if (corrDxy .lt. mincc .and. gftype .eq. 1) then
         if (i7 .eq. 1) then
            write (*,*) passnum,' pass below cc limit',corrDxy
            ilowcnt=ilowcnt+1
            go to 999
         endif
         subwinlen=subwinlen+1
         do 460 i=1,nobs
            do 460 j=1,27
               data(i,j)=strdata(i,j)
 460     continue
         go to 17
      endif
c
 999  eight=8888
      write (60,*) passnum,corrDxy
      if (inum .eq. 2) iendcnt=iendcnt+1
      if (inum .eq. 3) i3cnt=i3cnt+1
      if (inum .eq. 4) i4cnt=i4cnt+1
      if (inum .eq. 5) i5cnt=i5cnt+1
      if (inum .eq. 6) i6cnt=i6cnt+1
      if (inum .eq. 7) i7cnt=i7cnt+1
      if (inum .eq. 8) i8cnt=i8cnt+1
      if (inum .gt. 8) ibigcnt=ibigcnt+1
      return
      end
c
c
c
c
      subroutine despike (npts,outnum)
      real data(400,27),desdata(400,27),upper,lower
      integer ic(400),outnum,var
      common data(400,27)
      common /spike/ desdata(400,27),upper,lower,var
c
c-------------------------- this subroutine also provided by Tiku
c
c
c       PROGRAM DESPIKE
C*****************************************************************
C    PROGRAM DESPIKE REMOVES MOST SPIKES FROM THE INPUT DATA SET.
C    HOWEVER, FOR BEST RESULTS, IT IS SUGGESTED TO RUN DESPIKE
C    AT LEAST THREE TIMES --- FOR EXAMPLE:
C
C                INPUT1   ---DESPIKE---> OUTPUT1
C       (OUTPUT1 = INPUT2) ---DESPIKE---> OUTPUT2
C       (OUTPUT2 = INPUT3) ---DESPIKE---> OUTPUT3.
C
C    STILL, AFTER RUNNING DESPIKE THREE TIMES, IT FAILS TO ELIMINATE
C    ORBITS WITH DISCONTINUOUS RESID VS LATITUDE PROFILES.
C    PROGRAM DEGAP ATTEMPTS TO TAKE CARE OF SUCH PASSES.
C
C
C    PARAMETERS TO CHECK:  "UPPER" AND "LOWER" (IN NANOTESLAS):
C
C    IF PROGRAM DESPIKE HAS DETERMINED OBSERVATION  N  TO BE
C    A GOOD POINT, IT THEN SETS OUT TO DETERMINE IF POINT  N+1
C    IS A GOOD POINT.  IT DOES THIS BY CHECKING THE POINTS
C    N, N+1, AND N+2.  OBSERVATION  N+1  WILL BE A GOOD POINT
C    IF THE RESIDUAL DIFFERENCE BETWEEN POINT  N+1  AND THE
C    POINT ABOVE IT (N OR N+2) IS LESS THAN  "UPPER"  AND
C    IF THE RESIDUAL DIFFERENCE BETWEEN IT AND THE POINT
C    BELOW IT (N+2 OR N) IS GREATER THAN  "LOWER".
C*****************************************************************
C
      DO 2 I=1,400
         IC(I)=1
    2 CONTINUE
C
C*****************************************************************
C  ARE THE FIRST NEW POINTS SPIKES?
C  NOTE:  DATA(U,23) = RESID2(U)
C
      I=1
   15 SL1=(DATA(I+1,var)-DATA(I,var))
      SL2=(DATA(I+2,var)-DATA(I,var))
      SL3=(DATA(I+3,var)-DATA(I,var))
      SL4=(DATA(I+4,var)-DATA(I,var))
      XSL=ABS(SL1+SL2+SL3+SL4)/4.0
C
      S2=(DATA(I+2,var)-DATA(I+1,var))
C
```

```
      IF(ABS(SL1).GT.(3.0*XSL).OR.ABS(SL1).GT.(ABS(3.0*S2))) IC(I)=0
      IF(IC(I).EQ.0) THEN
         I=I+1
         GO TO 15
      ENDIF
C
C*****************************************************************
C  ARE THE MID POINTS SPIKES?
C
      DO 20 J=I,NPTS-2
         SL2=(DATA(J+1,var)-DATA(J+2,var))
         IF(SL1.GT.UPPER.AND.SL2.LT.LOWER) IC(J+1)=0
         IF(SL1.LT.LOWER.AND.SL2.GT.UPPER) IC(J+1)=0
         SL1=SL2
   20 CONTINUE
C
C*****************************************************************
C  IS THE LAST POINT A SPIKE?
C
      K=NPTS-2
   25 IF(IC(K).EQ.0) THEN
         K=K-1
         GO TO 25
      ENDIF
C
      SL1=ABS(DATA(K,var)-DATA(NPTS-1,var))
      SL2=ABS(DATA(NPTS-1,var)-DATA(NPTS,var))
      SL3=ABS(DATA(K,var)-DATA(NPTS,var))
         IF(IC(NPTS-1).EQ.0) THEN
            IF(SL1.GT.(3.0*SL2)) IC(NPTS)=0
            IF(SL3.GT.(3.0*UPPER)) IC(NPTS)=0
         ENDIF
         IF(IC(NPTS-1).EQ.1) THEN
            IF(SL2.GT.(3.0*SL1)) IC(NPTS)=0
         ENDIF
C
C     NOBS=0
C     DO 30 I=1,NPTS
C        IF(IC(I).EQ.1) NOBS=NOBS+1
C  30 CONTINUE
C     WRITE(6,*) IDATA(1,1), NOBS
C
      outnum=0
      DO 35 I=1,NPTS
         IF(IC(I).EQ.1) THEN
            outnum=outnum+1
            do 32 m=1,27
               desdata(outnum,m)=data(i,m)
 32            continue
         ENDIF
 35   CONTINUE
C
      return
      END
c
c
c
      subroutine track (nf,innum,noc,nocgf)
      common /trax/ x(400),y(400)
c-------------------------------- subroutine description
c        this subroutine calculates the lat lon coordinates of each point
c        to be plotted in tplot for a map view of the footprint of the pass
c
      RADFAC=0.017453293
      nop=innum
      if (nf .eq. 26) nocgf=nocgf+1
      if (nf .eq. 27) noc=noc+1
      DO 200 J=1,NOP
         x(j)=90.0-x(j)
         if(y(j).lt.0.) y(j)=y(j)+360.
         X(J)=X(J)*RADFAC
         Y(J)=Y(J)*RADFAC
  200 CONTINUE
      if (nf .eq. 26) WRITE (26) NOCgf,NOP, (X(J),Y(J),J=1,NOP)
      if (nf .eq. 27) WRITE (27) NOC,NOP, (X(J),Y(J),J=1,NOP)
c
      return
      end
c
c
c
c
c
      subroutine sqrfit (innum,x,ifitvar)
      real core(400),xmag(400),intpdata(400,27)
      common /intb/ intpdata(400,27)
c------------------------------ subroutine description
c        sqrfit fits the core field values to the observed data
c        in a least squares manner.  that is, the subroutine finds
c        a value of x that is multiplied by all core field values
```

```
c        in a pass so that the core field model matches the
c        observed values closer.
c
      x=0.0
      cmean=0.0
      fmean=0.0
      do i=1,innum
         core(i)=intpdata(i,16)
         cmean=cmean+core(i)
         xmag(i)=intpdata(i,ifitvar)
         fmean=fmean+xmag(i)
      enddo
c
      cmean=cmean/real(innum)
      fmean=fmean/real(innum)
c
      do i=1,innum
         xmag(i)=xmag(i)-fmean
         core(i)=core(i)-cmean
      enddo
c
      ctc=0.0
      do 600 i=1,innum
 600     ctc=(core(i)*core(i))+ctc
c
      ctcinv=1.0/ctc
c
      ctf=0.0
      do 700 i=1,innum
 700     ctf=(core(i)*xmag(i))+ctf
c
      x=ctcinv*ctf
c
      do 800 i=1,innum
 800     intpdata(i,ifitvar)=xmag(i)-(core(i)*x)
c
c
 999  continue
      return
      end
```

```
      program movetrunc
      integer xrow,xcol,zero,eight,xpassno,ypassno,ycol,
     >        x3row,x3col,x3pass,y3row,y3col,y3pass,yrow,
     >        strcnt,nobs,dndk,tcount,paircnt,nowant(4000),nocnt,
     >        paircnt1,jstop,minobs,passrem,totpass,global,
     >        outto,nopass,type,noc
      real xmean,ymean,x3data(400,3),y3data(400,3),y3mean,
     >     x1data(400),y1data(400),xadata(400,4),ybdata(400,4),
     >     x3mean,xamean,ybmean,x(400),y(400)
      character*80 filename
      common /tplot/ x(400),y(400)
      common /nope/ nowant(4000),nocnt
      common /trunstat/ xadata(400,4),ybdata(400,4)
      common /rowcol/ x3row,y3row,xrow,yrow,x3col,y3col,xcol,ycol,
     >                x3mean,y3mean,xmean,ymean,x3pass,y3pass,
     >                xpassno,ypassno
      common x3data(400,3),x1data(400),y3data(400,3),y1data(400)
c
c-------------------------------------- program description
c
c       this program truncates, in the time domain, two adjacent passes
c       to the same length.  this program should be used only after the
c       passes have been reordered by program reorder.  truncation is
c       accurate in the time domain rather than in the frequency domain.
c       this program is used before program fourier which is used to
c       extract the similar wavelengths of adjacent passes.
c
c        program date: 16 apr 91
c
      write (*,*) 'INPUT FILE X OF LAT-LONG-RAD DATA'
      read (*,9990) filename
 9990 format(a80)
      open (10, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT FILE X OF MAGNETIC VARIABLES'
      read (*,9990) filename
      open (11, file=filename,status='old',form='unformatted')
c
      write (*,*) '0 IF THE DATA IS GLOBAL OR POLAR'
      write (*,*) '1 IF THE DATA DOES NOT INCLUDE ALL LONGITUDES'
      read (*,*) global
      if (global .eq. 0) then
        write (*,*) 'NOTE: FILE Y WILL HAVE THE FIRST PASS MOVED'
        write (*,*) 'TO THE BOTTOM OF THE FILE'
        write (*,*) ' '
      elseif (global .eq. 1) then
        write (*,*) 'NOTE: OUTPUT FILE X WILL NOT INCLUDE THE LAST PASS'
        write (*,*) '      FILE Y WILL NOT INCLUDE THE FIRST PASS'
        write (*,*) ' '
      endif
c
      write (*,*) 'OUTPUT FILE X OF TRUNCATED LAT-LONG-RAD DATA'
      read (*,9990) filename
      open (20, file=filename,form='unformatted')
      write (*,*) 'OUTPUT FILE X OF TRUNCATED MAGNETIC VARIABLES'
      read (*,9990) filename
      open (21, file=filename,form='unformatted')
      write (*,*) 'OUTPUT FILE Y OF TRUNCATED LAT-LONG-RAD DATA'
      read (*,9990) filename
      open (22, file=filename,form='unformatted')
      write (*,*) 'OUTPUT FILE Y OF TRUNCATED MAGNETIC VARIABLES'
      read (*,9990) filename
      open (23, file=filename,form='unformatted')
      write (*,*) 'OUTPUT FILE OF TRACK PROFILES TO BE RUN IN TPLOT'
      read (*,9990) filename
      open (24, file=filename,form='unformatted')
      write (*,*) 'OUTPUT FILE OF STATISTICS'
      read (*,9990) filename
      open (25, file=filename,form='formatted')
c
      write (*,*) '0 IF THESE ARE DUSK DATA SETS'
      write (*,*) '1 IF THESE ARE DAWN DATA SETS'
      read (*,*) dndk
      write (*,*) '0 IF ALL PASSES ARE WANTED'
      write (*,*) '1 IF SOME PASSES SHOULD BE REMOVED'
      read (*,*) nocnt
      if (nocnt .eq. 1) then
        write (*,*) 'INPUT FILE OF PASS NUMBERS NOT WANTED'
        read (*,9990) filename
        open (26, file=filename,status='old',form='formatted')
        do 10 i=1,4000
          read (26,*,end=15) nowant(i)
 10     continue
 15     continue
        nocnt=i-1
      endif
      write (*,*) 'MINIMUM NUMBER OF OBSERVATIONS FOR EACH PASS'
      read (*,*) minobs
      write (*,*) 'TYPE OF GAP FINDER (2)'
      write (*,*) '1 FOR ONLY FINDING GAPS'
      write (*,*) '2 FOR USING THE MINIMUM OBSERVATIONS'
```

```
      read (*,*) type
c
      write (*,*) '------------------------------------------ '
      write (*,*) 'running through dataset to find passes',
     >            ' that do not overlap'
c
c---------------- subroutine findgap locates passes that do not
c                 have overlapping segments and removes the
c                 shorter of the two passes.  the subroutine
c                 continues reading and rereading the dataset
c                 until all non-overlapping segments are removed
c
      call findgap (global,dndk,minobs,type)
      write (*,*) 'done with run through'
      write (*,*) '------------------------------------------ '
c
      write (25,*)'XPASS YPASS CCD       CCY        XVAR       YVAR  ',
     >           'COVARXY   XSTDEV     YSTDEV'
      write (25,*)'                  XMEAN   YMEAN      XSLOPE     YSLOPE',
     >           '     XINTCPT   YINTCPT'
      paircnt=0
      paircnt1=0
      tcount=0
      strcnt=0
      jstop=0
      passrem=0
      totpass=0
c
 30   continue
      read (10,end=90) y3row,y3col,zero,y3mean,y3pass,eight
      do 35 i=1,y3row
         read (10) (y3data(i,ii),ii=1,y3col)
 35   continue
      read (11) yrow,ycol,zero,ymean,ypassno,eight
      do 45 i=1,yrow
         read (11) yldata(i)
 45   continue
      totpass=totpass+1
      if (yrow .lt. minobs  .or.  y3row .lt. minobs) then
        write (*,*)y3pass,ypassno,' PASS REMOVED: ROWS=',y3row,yrow
        passrem=passrem+1
        go to 30
      endif
      if (nocnt .eq. 0) go to 55
      do 50 i=1,nocnt
         if (ypassno .eq. nowant(i)) then
           write (*,*) ypassno,y3pass,' PASS REMOVED'
           passrem=passrem+1
           go to 30
         endif
 50   continue
 55   continue
      go to 95
c-------------------------------- this little jump around is used
c                                 to get the last and first passes
c


   of global datasets truncatted
 90   jstop=1
      if (global .eq. 1) go to 999
c
 95   continue
      strcnt=strcnt+1
c-------------------------------- offset the data file in subroutine
c                                 movtrun
      if (strcnt .eq. 1) call movtrun (-1,outto,jstop)
      if (strcnt .gt. 1) call movtrun (0,outto,jstop)
      if (outto) 30,100,100
c---------------------------------- truncate the passes to the same length
 100  continue
      nopass=0
      call truncate (xrow,yrow,dndk,nobs,paircnt1,nopass,
     >              xpassno,ypassno)
      if (nopass .gt. 0) then
c------------------------------ if this happens, then subroutine
c


  findgap didn't work just right
         write (*,*) 'OH MAN HAVE YOU GOT TROUBLE NOW'

  stop
      endif
c---------------------------------- do a little statistical nonsense
c
      call statistics (nobs,xpassno,ypassno,xamean,ybmean)
c
```

```
c-------------------------write out the truncated lengths of passes
c
      write (20) nobs,x3col,zero,xamean,xpassno,eight
      write (21) nobs,xcol,zero,xamean,xpassno,eight
      write (22) nobs,y3col,zero,ybmean,ypassno,eight
      write (23) nobs,ycol,zero,ybmean,ypassno,eight
      do 200 j=1,nobs
         write (20) xadata(j,1),(xadata(j,i),i=3,4)
         write (21) xadata(j,2)
         write (22) ybdata(j,1),(ybdata(j,i),i=3,4)
         write (23) ybdata(j,2)
 200  continue
c
      call track (nobs,noc)
      WRITE (24) NOC,NObs,(X(J),Y(J),J=1,NObs)
c
      if (paircntl .gt. 0) tcount=tcount+1
      paircnt=paircnt+paircntl
      if (jstop .eq. 1) go to 999
c
      call movtrun (1,outto,jstop)
      if (outto) 30,999,999
c
c
 999  continue
      if (global .eq. 1) strcnt=strcnt-1
      write (*,*) 'corrected',paircnt,' pairs of latitudes in'
      write (*,*) tcount,' passes to beginning lengths'
      write (*,*) 'total passes read = ',totpass
      write (*,*) 'removed',passrem,' passes from processing'
      write (*,*) 'total passes written = ',strcnt
      close (10)
      close (11)
      close (20)
      close (21)
      close (22)
      close (23)
      close (25)
      stop
      end
c
c
c
      subroutine truncate (xrow,yrow,dndk,minrow,stocount,nopass,
     >                      xpassno,ypassno)
      integer xrow,yrow,stocount,rowii,rowinc,minrow,nocnt,
     >      dndk,nopass,minxyrow,nowant(4000),xpassno,ypassno
      real xdata(400,4),ydata(400,4),
     >     x3data(400,3),x1data(400),y3data(400,3),y1data(400),
     >     adata,bdata,diffab,abss,xadata(400,4),ybdata(400,4)
      common /trunstat/ xadata(400,4),ybdata(400,4)
      common /nope/ nowant(4000),nocnt
      common x3data(400,3),x1data(400),y3data(400,3),y1data(400)
c
c-------------------------------- subroutine description
c
c       this subroutine truncates two sets of values to the same length.
c       truncation is based on the independent variable which is the
c       latitude of each point along a pass.  these values must be
c       interpolated to every 0.33 degrees.
c
      do 70 j=1,xrow
         xdata(j,1)=x3data(j,1)
         xdata(j,2)=x1data(j)
         xdata(j,3)=x3data(j,2)
         xdata(j,4)=x3data(j,3)
 70   continue
      do 75 j=1,yrow
         ydata(j,1)=y3data(j,1)
         ydata(j,2)=y1data(j)
         ydata(j,3)=y3data(j,2)
         ydata(j,4)=y3data(j,3)
 75   continue
c
 80   continue
      stocount=0
      jj=1
      rowii=xrow
      rowinc=yrow
c-----------------------loops from 90 to 200 increment through the
c                       two input passes and truncate the lengths
c                       to the same length
 90   continue
      adata=xdata(jj,1)
      bdata=ydata(jj,1)
      diffab=adata-bdata
      abss=abs(diffab)
      if (rowii .eq. 0  .or.  rowinc .eq. 0) then
        minxyrow=min(xrow,yrow)
        if (minxyrow .eq. xrow) then
```

```
           nopass=xpassno
           nowant(nocnt+1)=xpassno
           write (*,*) 'xrows (ii) =',rowii,' yrows (inc) =',rowinc
           write (*,*) 'rerunning to remove x pass number =',xpassno
           write (*,*) 'xrow =',xrow,' yrow =',yrow
           return
        elseif (minxyrow .eq. yrow) then
           nopass=ypassno
           nowant(nocnt+1)=ypassno
           write (*,*) 'xrows (ii) =',rowii,' yrows (inc) =',rowinc
           write (*,*) 'rerunning to remove y pass number =',ypassno
           write (*,*) 'xrow =',xrow,' yrow =',yrow
           return
        endif
      endif
      minrow=min(rowii,rowinc)
c      write (*,*) rowii,rowinc,minrow
c      write (*,*) adata,bdata,abss
c------------------------if pass a (ii) matches pass b (inc) at
c                        beginning length then write to xdata and
c                        ydata and race to main program
      if (abss .lt. 0.33) then
         do 110 ll=1,minrow
            do 110 kk=1,4
               xadata(ll,kk)=xdata(ll,kk)
               ybdata(ll,kk)=ydata(ll,kk)
c               write (*,*) xdata(ll),ydata(ll)
 110     continue
         return
      endif
c------------------------if pass a no matcha the b data then find new
c                        a or b depending on whether or not ascending
c                        or descending order of independent variable
      if (abss .ge. 0.33) then
         stocount=stocount+1
c------------------------if this is a dusk pass then will count from
c                        -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (xdata(jj,1) .gt. ydata(jj,1)) then
               rowinc=rowinc-1
               do 130 mm=1,rowinc
                  do 130 kk=1,4
                     ydata(mm,kk)=ydata(mm+1,kk)
c                    write (*,*) ydata(mm,kk)
 130              continue
            elseif (xdata(jj,1) .lt. ydata(jj,1)) then
               rowii=rowii-1
               do 150 nn=1,rowii
                  do 150 kk=1,4
                     xdata(nn,kk)=xdata(nn+1,kk)
 150              continue
            endif
c------------------------if this is a dawn pass then will count from
c                        the equator toward the south pole
c                        that is decreasing independent variable
         elseif (dndk .eq. 1) then
            if (xdata(jj,1) .lt. ydata(jj,1)) then
               rowinc=rowinc-1
               do 160 mm=1,rowinc
                  do 160 kk=1,4
                     ydata(mm,kk)=ydata(mm+1,kk)
c                    write (*,*) ydata(mm,kk)
 160              continue
            elseif (xdata(jj,1) .gt. ydata(jj,1)) then
               rowii=rowii-1
               do 170 nn=1,rowii
                  do 170 kk=1,4
                     xdata(nn,kk)=xdata(nn+1,kk)
 170              continue
            endif
         endif
      endif
c
      go to 90
c
      end
c
c
c
c
      subroutine statistics (minrow,xpassno,ypassno,xamean,ybmean)
      integer minrow,nobs,xpassno,ypassno
      real xadata(400,4),ybdata(400,4),nobss
      common /trunstat/ xadata(400,4),ybdata(400,4)
c
c                          the statistical calculations using two
c                          references:
c                           1) Davis, Statistics and Data Analysis in
c                              Geology, 2nd ed., 1986 pp. 41
c                           2) Young, Statistical Treatment of Experi-
```

```
c                              mental Data, 1962, McGraw Hill, 115-132
c
c-----------------------loops that sum x, x**2, y, y**2 and xy
c                        and calculate new truncate means
      nobs=minrow
      nobss=float(nobs)
      xsum=0.0
      xsumsqr=0.0
      ysum=0.0
      ysumsqr=0.0
      sumxy=0.0
      do 240 j=1,nobs
         xsum=xsum+xadata(j,2)
         xsumsqr=xsumsqr+(xadata(j,2))**2
         ysum=ysum+ybdata(j,2)
         ysumsqr=ysumsqr+(ybdata(j,2))**2
         sumxy=sumxy+(xadata(j,2)*ybdata(j,2))
 240     continue
c      write (*,*) xsum,ysum,xsumsqr,ysumsqr,sumxy
c-----------------------find corrected sum of products, covariance
c                       and corrected sum of squares (x) (y)
c
      xamean=xsum/nobss
      ybmean=ysum/nobss
      sumprod=sumxy-((xsum*ysum)/nobss)
      covarxy=sumprod/(nobss-1.0)
      xcsumsqr=xsumsqr-((xsum**2)/nobss)
      ycsumsqr=ysumsqr-((ysum**2)/nobss)
c
c-----------------------find variance, standard deviation for x and y
c
      xvar=xcsumsqr/(nobss-1.0)
      yvar=ycsumsqr/(nobss-1.0)
      xstdev=sqrt(xvar)
      ystdev=sqrt(yvar)
c-----------------------find correlation coefficient by Davis method
      corrDxy=covarxy/(xstdev*ystdev)
c-----------------------find slopes, intercepts and correlation
c                       coefficient by Young method
      xslope=((nobss*sumxy)-(xsum*ysum))/((nobss*xsumsqr)-xsum**2)
      yslope=((nobss*sumxy)-(xsum*ysum))/((nobss*ysumsqr)-ysum**2)
      xintcpt=((ysum*xsumsqr)-(sumxy*xsum))/((nobss*xsumsqr)-xsum**2)
      yintcpt=((xsum*ysumsqr)-(sumxy*ysum))/((nobss*ysumsqr)-ysum**2)
      corrYxy=sqrt(xslope*yslope)
c
c-----------------------write out this mess for individual pass and
c                       overlapping lengths of passes
c
c      write (25,9992)xpassno,ypassno,xvar,yvar,xstdev,ystdev,
c     >               xamean,ybmean
 9992 format ('FOR OVERLAPPING LENGTHS   X=',i5,' Y=',i5,/,
     >        'X VARIANCE=',f9.3,' Y VARIANCE=',f9.3,' X STDEV=',
     >        f9.3,' Y STDEV=',f9.3,' XMEAN=',f9.3,' YMEAN=',f9.3)
c      write (25,9993) covarxy,corrDxy
 9993 format ('COVARIANCE XY=',f9.3,' Davis CORRELATION COEF=',f9.3)
c      write (25,9994) xslope,xintcpt,yslope,yintcpt,corrYxy
 9994 format ('X SLOPE=',f9.3,' X INTERCEPT=',f9.3,' Y SLOPE=',
     >         f9.3,' Y INTERCEPT=',f9.3,' Young CORRELATION COEF=',
     >         f9.3,/)
c
      write (25,9995) xpassno,ypassno,corrDxy,corrYxy,xvar,yvar,
     >               covarxy,xstdev,ystdev
 9995 format (2i5,7(f10.4))
      write (25,9996) xamean,ybmean,xslope,yslope,xintcpt,yintcpt
 9996 format (10x,6(f10.4))
      return
      end
c
c
c
      subroutine movtrun (into,outto,jstop)
      integer into,outto,sy3row,syrow,sy3col,sycol,
     >        sy3pass,sypassno,jstop,
     >        y3row,yrow,y3col,ycol,y3pass,ypassno,
     >        x3row,xrow,x3col,xcol,x3pass,xpassno,
     >        sx3row,sxrow,sx3col,sxcol,sx3pass,sxpassno
      real x3data(400,3),x1data(400),y3data(400,3),y1data(400),
     >     sy3mean,symean,savey3(400,3),savey1(400),y3mean,
     >     ymean,x3mean,xmean,storex3(400,3),storex1(400),
     >     sx3mean,sxmean
      common x3data(400,3),x1data(400),y3data(400,3),y1data(400)
      common /rowcol/ x3row,y3row,xrow,yrow,x3col,y3col,xcol,ycol,
     >                x3mean,y3mean,xmean,ymean,x3pass,y3pass,
     >                xpassno,ypassno
c
c-------------------------------- subroutine description
c        this subroutine stores one pass so that the offset
c        will occur.
      if (jstop) 10,10,350
 10   if (into) 40,85,290
```

```
c
 40    continue
       do 50 i=1,y3row
          do 55 ii=1,y3col
             savey3(i,ii)=y3data(i,ii)
 55       continue
          savey1(i)=y1data(i)
 50    continue
       sy3row=y3row
       syrow=yrow
       sy3col=y3col
       sycol=ycol
       sy3mean=y3mean
       symean=ymean
       sy3pass=y3pass
       sypassno=ypassno
c
       do 70 i=1,y3row
          do 75 ii=1,y3col
             x3data(i,ii)=y3data(i,ii)
 75       continue
          x1data(i)=y1data(i)
 70    continue
       x3row=y3row
       xrow=yrow
       x3col=y3col
       xcol=ycol
       x3mean=y3mean
       xmean=ymean
       x3pass=y3pass
       xpassno=ypassno
c
       outto = -1
       return
c
c
 85    continue
       do 90 i=1,y3row
          do 95 ii=1,y3col
             storex3(i,ii)=y3data(i,ii)
 95       continue
          storex1(i)=y1data(i)
 90    continue
       sx3row=y3row
       sxrow=yrow
       sx3col=y3col
       sxcol=ycol
       sx3mean=y3mean
       sxmean=ymean
       sx3pass=y3pass
       sxpassno=ypassno
c
       if (xrow.ne.x3row .or. xpassno.ne.x3pass .or.
     >     yrow.ne.y3row .or. ypassno.ne.y3pass) then
          write (*,*) 'WACKO, TRA-LA-LA, JOLLY-GOOD,  NO MATCH BETWEEN'
          write (*,*) 'ROWS OR PASSNOS  X= ',xrow,x3row,xpassno,x3pass
          write (*,*) '                 Y= ',yrow,y3row,ypassno,y3pass
          stop
       endif
c
       outto = 0
       return
c
c
 290   continue
       x3row=sx3row
       xrow=sxrow
       x3col=sx3col
       xcol=sxcol
       x3mean=sx3mean
       xmean=sxmean
       x3pass=sx3pass
       xpassno=sxpassno
       do 300 i=1,x3row
          do 305 ii=1,x3col
             x3data(i,ii)=storex3(i,ii)
 305      continue
          x1data(i)=storex1(i)
 300   continue
c
       outto = -1
       return
c
c
 350   continue
       do 360 i=1,sy3row
          do 365 ii=1,sy3col
             y3data(i,ii)=savey3(i,ii)
 365      continue
          y1data(i)=savey1(i)
```

```
 360  continue
      y3row=sy3row
      yrow=syrow
      y3col=sy3col
      ycol=sycol
      y3mean=sy3mean
      ymean=symean
      y3pass=sy3pass
      ypassno=sypassno
c
      outto = 0
      return
c
      end
c
c
c
      subroutine findgap (global,dndk,minobs,type)
      integer zero,eight,minxyrow,xrow,yrow,type,
     >        y3row,y3col,y3pass,xcnt,ycnt,strmincnt,
     >        dndk,nowant(4000),nocnt,xpassno,ypassno,
     >        minobs,totpass,global,both,nocnt2,
     >        nopass,strnocnt,allcnt
      real y3data(400,3),y3mean,abss,alldata(4000,4)
      common /nope/ nowant(4000),nocnt
c------------------------------------- subroutine description
c        findgap locates two adjacent passes that do not have
c        any common overlapping segment.  if it finds two such
c        passes, then it removes the shorter of the two, continues
c        running through the remainder of the data while searching
c        for non-overlapping passes, and finally reruns through
c        the data set to assure that all non-overlapping passes
c        have been located.
c
      totpass=0
      allcnt=0
      strnocnt=nocnt
c-------------------- read through the data only once and
c                     store the pass number, first lat and
c                     last lat in array alldata
 30   continue
      read (10,end=60) y3row,y3col,zero,y3mean,y3pass,eight
      do 35 i=1,y3row
         read (10) (y3data(i,ii),ii=1,y3col)
 35   continue
      totpass=totpass+1
      if (y3row .lt. minobs) go to 30
      if (nocnt .eq. 0) go to 55
      do 50 i=1,nocnt
         if (y3pass .eq. nowant(i)) go to 30
 50   continue
 55   continue
      allcnt=allcnt+1
      alldata(allcnt,1)=real(y3pass)
      alldata(allcnt,2)=real(y3row)
      alldata(allcnt,3)=y3data(1,1)
      alldata(allcnt,4)=y3data(y3row,1)
      go to 30
 60   continue
c------------------------------- depending on the type of gap
c


   finder chosen, the program
c                                 will proceed as appropriate
      if (type .eq. 2) go to 400
c
 70   continue
      jstop=0
c----------------------- xcnt and ycnt represent the two
c                        adjacent passes
      xcnt=1
      ycnt=2
c----------------------- the next two if statements check
c                        if one of the two adjacent passes
c                        is not wanted
      if (nocnt .eq. 0) go to 100
      do 80 i=1,nocnt
         if (int(alldata(xcnt,1)) .eq. nowant(i)) then
            do 90 jj=xcnt,allcnt-1
               do 90 j=1,4
                  alldata(jj,j)=alldata(jj+1,j)
 90         continue
            allcnt=allcnt-1
            go to 70
         endif
 80   continue
c
 100  continue
      if (nocnt .eq. 0) go to 140
```

```
      do 110 i=1,nocnt
         if (int(alldata(ycnt,1)) .eq. nowant(i)) then
            if (ycnt+1 .gt. allcnt) then
               ycnt=ycnt+1
               go to 195
            elseif (jstop .eq. 1) then
               ycnt=ycnt+1
               go to 100
            endif
            do 105 jj=ycnt,allcnt-1
               do 105 j=1,4
                  alldata(jj,j)=alldata(jj+1,j)
 105        continue
            allcnt=allcnt-1
            go to 100
         endif
 110  continue
 140  continue
      both=0
      abss=abs(alldata(xcnt,3)-alldata(ycnt,3))
      if (abss .lt. 0.33) go to 190
      if (abss .ge. 0.33) then
c-------------------------- truncation time!
         xrow=int(alldata(xcnt,2))
         yrow=int(alldata(ycnt,2))
         xpassno=int(alldata(xcnt,1))
         ypassno=int(alldata(ycnt,1))
         minxyrow=min(xrow,yrow)
         nopass=xpassno
         if (minxyrow .eq. yrow) nopass=ypassno
         if (xrow .eq. yrow) both=1
         nocnt2=nocnt
c-----------------------if this is a dusk pass then will count from
c                        -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
               if (alldata(xcnt,3) .gt. alldata(ycnt,4)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            elseif (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
               if (alldata(xcnt,4) .lt. alldata(ycnt,3)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            endif
c-----------------------if this is a dawn pass then will count from
c                        the equator toward the south pole
c                        that is decreasing independent variable
         elseif (dndk .eq. 1) then
            if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
               if (alldata(xcnt,3) .lt. alldata(ycnt,4)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            elseif (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
               if (alldata(xcnt,4) .gt. alldata(ycnt,3)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            endif
         endif
         if (nocnt .gt. nocnt2  .and.  both .eq. 1) then
            nocnt=nocnt+1
            nowant(nocnt)=xpassno
         endif
      endif
c
 190  continue
      xcnt=ycnt
      ycnt=ycnt+1
      if (jstop .eq. 1) go to 200
 195  if (ycnt .gt. allcnt) then
         if (global .eq. 1) go to 200
         ycnt=1
         jstop=1
      endif
      go to 100
c
 200  continue
      if (nocnt .eq. strnocnt) go to 999
      if (strnocnt .lt. nocnt) then
         strnocnt=nocnt
         go to 70
      endif
c
c
 400  continue
      mincnt=0
      strmincnt=mincnt
```

```
c
 470  continue
      jstop=0
c------------------------ xcnt and ycnt, see notes above
      xcnt=1
      ycnt=2
      if (int(alldata(xcnt,2)) .lt. minobs) then
         do 490 jj=xcnt,allcnt-1
            do 490 j=1,4
               alldata(jj,j)=alldata(jj+1,j)
 490     continue
         allcnt=allcnt-1
         go to 470
      endif
c
 500  continue
      if (int(alldata(ycnt,2)) .lt. minobs) then
         if (ycnt+1 .gt. allcnt) then
            ycnt=ycnt+1
            go to 595
         elseif (jstop .eq. 1) then
            ycnt=ycnt+1
            go to 500
         endif
         do 505 jj=ycnt,allcnt-1
            do 505 j=1,4
               alldata(jj,j)=alldata(jj+1,j)
 505     continue
         allcnt=allcnt-1
         go to 500
      endif
 510  continue
 540  continue
      abss=abs(alldata(xcnt,3)-alldata(ycnt,3))
      if (abss .lt. 0.33) go to 590
      if (abss .ge. 0.33) then
         xrow=int(alldata(xcnt,2))
         yrow=int(alldata(ycnt,2))
         minxyrow=min(xrow,yrow)
         mincnt2=mincnt
c------------------------if this is a dusk pass then will count from
c                        -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
               if (alldata(xcnt,3) .gt. alldata(ycnt,4)) mincnt=mincnt+1
            elseif (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
               if (alldata(xcnt,4) .lt. alldata(ycnt,3)) mincnt=mincnt+1
            endif
c------------------------if this is a dawn pass then will count from
c                        the equator toward the south pole
c                        that is decreasing independent variable
         elseif (dndk .eq. 1) then
            if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
               if (alldata(xcnt,3) .lt. alldata(ycnt,4)) mincnt=mincnt+1
            elseif (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
               if (alldata(xcnt,4) .gt. alldata(ycnt,3)) mincnt=mincnt+1
            endif
         endif
         if (mincnt .gt. mincnt2) minobs=minxyrow+1
      endif
c
 590  continue
      xcnt=ycnt
      ycnt=ycnt+1
      if (jstop .eq. 1) go to 600
 595  if (ycnt .gt. allcnt) then
         if (global .eq. 1) go to 600
         ycnt=1
         jstop=1
      endif
      go to 500
c
 600  continue
      if (mincnt .eq. strmincnt) go to 999
      if (strmincnt .lt. mincnt) then
         strmincnt=mincnt
         go to 470
      endif
c
 999  continue
      write (*,*) 'total passes read = ',totpass
      if (nocnt .gt. 0) then
         write (*,*) 'will remove the following passes from processing'
         do 1010 i=1,nocnt
            write (*,*) nowant(i),i
 1010    continue
      endif
      if (type .eq. 2) write (*,*) 'new minimum observation cutoff',
     >                    ' is =',minobs
      rewind (10)
```

```
c
      return
      end
c
c
c
      subroutine track (nop,noc)
      integer nop,noc
      real radfac,x(400),y(400),xadata(400,4),ybdata(400,4)
      common /tplot/ x(400),y(400)
      common /trunstat/ xadata(400,4),ybdata(400,4)
c-------------------------------- subroutine description
c         track stores the lat and long coordinates of each
c         data point along a pass.  these coordinates are then
c         used to plot with a graphics package, the track footprint
c         of the satellite.
c         NOTE: the lat and long values are converted to radian values
c               because the plotting package that i work with utilizes
c               radians.
      RADFAC=0.017453293
      noc=noc+1
      do 300 j=1,nop
         x(j) = 90.0 - xadata(j,1)
         y(j) = xadata(j,3)
         if (y(j) .lt. 0.0) y(j) = y(j) + 360.0
         x(j) = x(j) * radfac
         y(j) = y(j) * radfac
 300  continue
c
      return
      end
```

```
      program fourier1d
      character*80 filename
      real xdata(4096),ydata(4096),xmean,ymean,minccin,
     >     prcnt,delta,cuthi,cutlo,xlag,mincc,maxcc,short,long,
     >     maxccin,cxlag,strxdata(4096)
      integer xpassno,ypassno,zero,eight,file,xcol,xrow,ycol,yrow,
     >         trnsf,lhb,cc,trnsb,npass,imean,cnwind,seven,
     >         nwind,numfile,nxout,nyout,match,gfcnt,xnobs,ynobs
      complex xcdata(4096),ycdata(4096)
      common /rowcol/ xnobs,ynobs
      common /fftifft/ nobs,prcnt,imean,fold
      common /lhbflt/ delta,cuthi,cutlo,xlag,npass,nwind
      common /ccflt/ mincc,maxcc,match,minccin,maxccin,cnwind,cxlag
      common /reals/ xdata,ydata
      common /comps/ xcdata,ycdata
c
c-----------------------------program description
c
c       fourier1d is an all encompassing fourier analysis program!
c       subroutines include the fft for forward and inverse situations,
c       a bandpass filter which can be adjusted to perform low, high and
c       bandpass filtering of wave numbers and a correlation coefficient
c       filter which zeros out wavenumbers according to correlation
c       coefficients.  both the bandpass filter and the correlation
c       coefficient filter provide the user with several windowing options
c       (as well as no windowing option) to smooth wavenumbers prior
c       to inverse transformation.  both filters use the same subroutine
c       to window in the TIME (real number data) domain.  with respect to
c       run time considerations: if many different datasets are submitted
c       at the same time to the program, it will still calculate the same
c       BANDPASS windowing function for each dataset every time it
c       encounters a new dataset.  since this windowing function need only
c       be calculated once, it causes the program to do needless work.  i
c       hope to soon remedie this little time consuming "bug".  big note of
c       caution: the windowing function will change with each new dataset
c       for the CORRELATION FILTER and therefore, leave it alone!!
c
c       NOTE: ANY fourier analysis routine can be inserted
c             to this program as a subroutine.  maybe i'll put in
c             such features as upward continuation, etc..
c       NOTE: the only data variables absolutely necessary as INPUT are
c             the number of observations of input profile, the remaining
c             variables; zero,mean,pass-number and eight, are not needed.
c             but, mean can be an OUTPUT if desired.
c
c    updates:
c          30 Jan 91
c          this update pertained to removing calls to differing fft2d
c          subroutines so that now all calls are to the same fft2d
c          subroutine.  and more importantly, now the fft2d routines
c          will handle 1 row of data so that the bandpass filter works
c          correctly.  and even more importantly the zero filling option
c          now zero fills such that the data is located in the middle
c          of all those wonderful zeros.  and for those of you who are
c          really into this, you can now fold out a percentage of the
c          edge of your data, smooth the folded out part to zero, fft,
c          filter, and ifft such that edge effects are minimized.
c          ohh boyy!!
c
c          9 jul 92:
c          major revisions changing code from two-dimensionally based
c          ffts to one-dimensional ffts.  revisions include removal of
c          subroutines transpo and store.  major changes to subroutines
c          fft2d (which is now known as fft1d), datwnd, bndpas and
c          window.  now there is no longer a need for transposing the
c          arrays so that run time should be decreased.
c          NOTE: i removed a great deal of comments at the beginning of
c                the subroutines.  all removed comments discussed the
c                two dimensional sense of the routine.  i added comments
c                dealing specifically with the one dimensional changes.
c
c
       write (*,*) '0 IF YOU HAVE A FILE OF ALL VARIABLES'
       write (*,*) '1 IF YOU WANT TO TYPE THEM INTERACTIVELY -- ha ha'
       read (*,*) file
       if (file .eq. 0) then
          write (*,9988)
 9988     format ('USE THE FOLLOWING ORDER FOR INPUT FILE'/,
     >            'IF VARIABLE DOES NOT APPLY INPUT ANY BOGUS NUMBER'/,
     >            'numfile'/,
     >            'lhb  cc'/,
     >            'nobs  fold  prcnt  imean'/,
     >            'delta  short  long   npass'/,
     >            'nwind  xlag'/
     >            'mincc  maxcc  match  minccin maxccin cnwind cxlag'/
     >            'isub'/)
          write (*,*) 'INPUT FILE OF VARIABLES'
          read (*,9990) filename
          open (22, file=filename, form='formatted',status='old')
          go to 100
```

```
      elseif (file .eq. 1) then
         go to 50
      elseif (file .ne. 0  .or.  file .ne. 1) then
         write (*,*) 'HEY HEY HEY ITS BAD FILE NUMBER AND YOU GOT'
         write (*,*) 'TO MAKE A NEW CHOICE TOOOOOOO -- to the tune of'
         write (*,*) 'fat albert'
         go to 999
      endif
c
 50   continue
      write (*,9989)
 9989 format ('1 IF YOU HAVE ONLY ONE FILE TO BE FOURIERED'/,
     >        '2 IF YOU HAVE TWO FILES TO BE COMPARED')
      read (*,*) numfile
c
      write (*,9991)
 9991 format (/'1 FOR ONLY LOW-HIGH-BAND FILTER THE DATA'/,
     >        '2 FOR L-H-B FILTER THEN C-C FILTER THE DATA'/,
     >        '3 FOR L-H-B THEN C-C THEN L-H-B FILTER THE DATA'/,
     >        '4 FOR C-C FILTER THEN L-H-B FILTER THE DATA'/,
     >        '5 FOR C-C THEN L-H-B THEN C-C FILTER THE DATA'/,
     >        '6 DO NOT FILTER THE DATA'//,
     >        '1 FOR ONLY CORRELATION COEFICIENT FILTER THE DATA'/,
     >        '2 DO NOT C-C FILTER THE DATA'/,
     >        'choose 2 if 5 or less was chosen above'//,
     >        'lhb  cc')
      read (*,*) lhb,cc
c
      write (*,9992)
 9992 format ('THE FOLLOWING REFERS TO FFT AND IFFT'//,
     >        'NUMBER OF OBSERVATIONS FOR FFT ARRAY'/,
     >        'AT A POWER OF 2: (256)(2 16 32 64 128 256 etc)'//,
     >        'PERCENT OF DATA TO BE FOLDED OUT (0.1 TO 99.9)'/,
     >        'PERCENT OF EACH EDGE OF INPUT ARRAY OR FOLDED'/,
     >        'OUT ARRAY TO BE SMOOTHED TO ZERO (0.1 TO 49.9)'//,
     >        '0 DO NOT ADD MEAN TO IFFT DATA'/,
     >        '1 ADD MEAN OF INPUT DATA TO OUTPUT IFFT DATA'/,
     >        'nobs  fold  prcnt  imean')
      read (*,*) nobs,fold,prcnt,imean
c
      if (nobs .gt. 4096) then
         write (*,8999) nobs
 8999    format (1x,'SORRY',i6,1x,'IS GREATER THAN 4096 THE',
     >               ' SIZE OF ARRAYS SET'/' IN THE SOURCE CODE ',
     >               'YOU NEED TO ACCESS SOURCE CODE AND MAKE CHANGES')
         go to 999
      endif
c
      if (lhb .lt. 6) then
         write (*,9993)
 9993    format ('DELTA.....GRID INTERVAL IN MAP UNITS (0.33 degrees)'/
     >        'SHORT.....SHORTEST WAVELENGTH TO BE PASSED'/
     >        '          MUST BE AT LEAST 2*DELTA (0.66 degrees)'/
     >        'LONG......LONGEST WAVELENGTH TO BE PASSED'/
     >        '          MUST BE LARGER THAN SHORT ...no kidding!'/
c    >         'CUTHI.....HIGHEST WAVENUMBER TO PASSED .LE. NYQUIST'/
c    >         'CUTLO.....LOWEST WAVENUMBER TO BE PASSED .GE. 0.0 AND'/
c    >         '           .LT. CUTHI'//
     >        'NPASS.....-1 TO REJECT WAVELENGTHS BETWEEN SHORT'/
     >        '           AND LONG'/
     >        '          1 TO PASS WAVELENGTHS BETWEEN SHORT AND'/
     >        '           LONG'//
     >        '      NOTE : WAVENUMBER = 1/WAVELENGTH  AND IS '/
     >        '            CALCULATED BY THE PROGRAM'/
     >        ' INPUT ORDER IS  DELTA  SHORT  LONG  NPASS')
         read (*,*) delta,short,long,npass
         write (*,9994)
 9994    format (' NWIND..... TYPE OF WINDOW TO APPLY'/
     >   '             = 0 GIVES NO WINDOW'/
     >   '             = 1 RECTANGULAR WINDOW'/
     >   '             = 2 BARTLETT WINDOW (TRIANGULAR)'/
     >   '             = 3 HAMMING-TUKEY WINDOW'/
     >   '             = 4 PARZEN WINDOW'/
     >   '  XLAG.....SMOOTHING PARAMETER FOR WINDOWING IDEAL'/
     >   '           FILTER IN SPATIAL DOMAIN (95.0) (is disabled if'/
     >   '           no window was chosen above).'//
     >   ' nwind  xlag')
         read (*,*) nwind,xlag
      endif
c
      if (lhb .ge. 2 .and. lhb .le. 5 .or. cc .eq. 1) then
         write (*,9995)
 9995  format ('WHAT IS THE MINIMUM CORR COEF TO BE PASSED:(0.3)'/,
     >   'WHAT IS THE MAXIMUM CORR COEF TO BE PASSED:(1.0)'/,
     >   '0 DO NOT CHECK MATCH OF PASSNOS'/,
     >   '1 FOR PROGRAM TO CHECK MATCH OF PASSNOS'/,
     >   'MINIMUM INPUT CC WITHOUT WRITING WARNING'/,
     >   'MAXIMUM INPUT CC WITHOUT WRITING WARNING'/,
     >   'CNWIND..... TYPE OF WINDOW TO APPLY'/
     >   '             = 0 GIVES NO WINDOW'/
```

```
     >     '              = 1 RECTANGULAR WINDOW'/
     >     '              = 2 BARTLETT WINDOW (TRIANGULAR)'/
     >     '              = 3 HAMMING-TUKEY WINDOW'/
     >     '              = 4 PARZEN WINDOW'/
     >     'CXLAG......SMOOTHING PARAMETER FOR WINDOWING IDEAL'/
     >     '           FILTER IN SPATIAL DOMAIN (95.0) (is disabled if'/
     >     '           no window was chosen above).'//
     >     'mincc  maxcc  match  minccin  maxccin  cnwind  cxlag')
        read (*,*) mincc,maxcc,match,minccin,maxccin,cnwind,cxlag
      endif
      if (numfile .eq. 2) then
      write (*,*) 'AND FINALLY: 0 DO NOT SUBTRACT FILE1-FILE1 (use 0)'
      write (*,*) '1 TO WRITE A FILE OF TIME DOMAIN SUBTRACTION'
      write (*,*) 'OF FILE 3 - INPUT FILE 1 - OUTPUT FILE 1'
      read (*,*) isub
c------------------------ isub equals 1 if you want to subtract
c                         the filtered portions of file1 from the
c                         input of file1.  this option is not
c

    often used.
      endif
c
      go to 200
c
 100  continue
      read (22,*) numfile
      read (22,*) lhb,cc
      read (22,*) nobs,fold,prcnt,imean
      read (22,*) delta,short,long,npass
      read (22,*) nwind,xlag
      read (22,*) mincc,maxcc,match,minccin,maxccin,cnwind,cxlag
      read (22,*) isub
c
 200  continue
c
      write (*,*) 'all input files must have a header with:'
      write (*,*) 'row,column,zero,mean,pass number,eight'
      write (*,*) 'zero... can be bogus but row and col are necessary'
      write (*,*) 'INPUT FILE 1 (do not put guide function file here)'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename,status='old',form='unformatted')
      if (numfile .eq. 2) then
         write (*,*) 'INPUT FILE 2 (or the guide function file)'
         read (*,9990) filename
         open (11, file=filename,status='old',form='unformatted')
      endif
      write (*,*) 'OUTPUT OF FILE 1'
      read (*,9990) filename
      open (20, file=filename,form='unformatted')
      if (numfile .eq. 2) then
         write (*,*) 'OUTPUT OF FILE 2'
         read (*,9990) filename
         open (21, file=filename,form='unformatted')
      endif
      write (*,*) 'OUTPUT FILE OF STATISTICS AND INFORMATION'
      read (*,9990) filename
      open (25, file=filename,form='formatted')
      if (isub .eq. 1) then
         write (*,*) 'OUTPUT FILE 3 OF SUBTRACTION'
         read (*,9990) filename
         open (23, file=filename, form='unformatted')
      endif
c
      if (lhb .lt. 6) then
         cuthi=1.0/short
         cutlo=1.0/long
         RCUTLO=999999.99
         IF(CUTLO .GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
         RCUTHI=1.0/CUTHI
         WAVLEN=2.0*DELTA
         FNQ1=1.0/WAVLEN
         WRITE (25,9987) FNQ1,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI
 9987 FORMAT('NYQUIST WAVENUMBER =',F10.5,' CYCLES PER DATA INTERVAL'/,
     >        'NYQUIST WAVELENGTH = ',F10.5,' LENGTH INTERVALS'/,
     >        'LOW WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >        ' CYCLES PER DATA INTERVAL '/,F15.5,
     >        ' WAVELENGTH EQUIVALENT'/,
     >        'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >        ' CYCLES PER DATA INTERVAL '/,F15.5,
     >        ' WAVELENGTH EQUIVALENT',//)
      endif
c
      gfcnt=0
 210  continue
      read (10,end=999) xrow,xcol,zero,xmean,xpassno,eight
      xnobs=xrow
      do i=1,xrow
```

```
         read (10) xdata(i)
      enddo
 230  if (numfile .eq. 2) then
         read (11,end=999) yrow,ycol,zero,ymean,ypassno,seven
         if (seven .eq. 7777) go to 500
         ynobs=yrow
         do i=1,yrow
            read (11) ydata(i)
         enddo
      endif
c
      if (isub .eq. 1) then
         do i=1,xrow
            strxdata(i)=xdata(i)
         enddo
      endif
c
      xmean=0.0
      call forwardft (1,xmean,xpassno)
      ymean=0.0
      if (numfile .eq. 2) call forwardft (2,ymean,ypassno)
c
      if (lhb .le. 3) then
        call filter (1)
        if (numfile .eq. 2) call filter (2)
      endif
c
      if (lhb .ge. 2 .and. lhb .le. 5 .or. cc .eq. 1) then
        call correlate(xpassno,ypassno)
      endif
c
      if (lhb .ge. 3 .and. lhb .le. 5) then
        call filter (1)
        if (numfile .eq. 2) call filter (2)
      endif
c
      if (lhb .eq. 5) call correlate(xpassno,ypassno)
c
      call inverseft (1,xmean,xpassno)
      if (numfile .eq. 2) call inverseft (2,ymean,ypassno)
c
c
 500  write (20) xrow,xcol,zero,xmean,xpassno,eight
      do i=1,xrow
         write (20) xdata(i)
      enddo
      if (numfile .eq. 2) then
 530     write (21) yrow,ycol,zero,ymean,ypassno,seven
         if (seven .eq. 7777) then
            gfcnt=gfcnt+1
            go to 570
         endif
         do i=1,yrow
            write (21) ydata(i)
         enddo
 570     if (isub .eq. 1) then
            if (seven .eq. 7777) then
               write (23) xrow,xcol,zero,xmean,xpassno,eight
               do i=1,xrow
                  write (23) xdata(i)
               enddo
            elseif (seven .ne. 7777) then
               write (23) xrow,xcol,zero,xmean,xpassno,seven
               do i=1,xrow
                  write (23) ( strxdata(i)-xdata(i) )
               enddo
            endif
         endif
      endif
c
      go to 210
c
 999  continue
      if (gfcnt .gt. 0)
     >   write (*,*) 'total passes without a guide function =',gfcnt
      close (10)
      close (11)
      close (20)
      close (21)
      close (22)
      close (25)
      stop
      end
c
c
c
      subroutine forwardft (num,mean,passno)
      integer num,xnobs,ynobs,xynobs,nobs,passno
      real xdata(4096),ydata(4096),prcnt,mean
      complex xcdata(4096),ycdata(4096)
```

```
      common /fftifft/ nobs,prcnt,imean,fold
      common /rowcol/ xnobs,ynobs
      common /reals/ xdata,ydata
      common /comps/ xcdata,ycdata
      COMMON  H(4096)
      DIMENSION X(2,4096)
      COMPLEX H
      double precision TSUM
      EQUIVALENCE (X(1,1),H(1))
c
      TSUM=0.D0
      if (num .eq. 1) then
         xynobs=xnobs
         do i=1,xynobs
            x(1,i) = xdata(i)
            tsum=tsum+x(1,i)
         enddo
      elseif (num .eq. 2) then
         xynobs=ynobs
         do i=1,xynobs
            x(1,i) = ydata(i)
            tsum=tsum+x(1,i)
         enddo
      endif
c
c
c---------------------------------- subroutine description
c
C.....REQUIRED SUBROUTINES :
C
C        FFT1D, FORK, DATWND
C
C.....DIMENSIONING REQUIREMENTS :
C
C        X(2,N).....WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
C        H(N)       OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
C                   POWER OF TWO (2,4,8,16...).
C                   NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
C                   SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C.....AUTHOR :SUBROUTINES FFT2D AND FORK ARE MODIFIED FROM JON REED,
C             PURDUE UNIVERSITY, DECEMBER 1980.
C             ALL OTHER CODE WRITTEN BY:
C             JEFFREY E. LUCIUS
C             GEOPHYSICAL INTERACTIVE COMPUTING LABORATORY
C             DEPARTMENT OF GEOLOGY AND MINERALOGY
C             THE OHIO STATE UNIVERSITY
C             COLUMBUS, OHIO 43210
C
C             MARCH 25, 1985   (REVISED DEC 5, 1986)
C
C
c     revised once again for DEC workstations on 6 APR 90 so that
c     that this beast is actually user friendly.
c     These revisions will almost always be lower case letters.
c
c     revised again (judas priest this is getting old) on
c     1 AUG 90  into this present format of all fourier programs
c     combined into this program.  for a full listing of all
c     comments in the 6 APR 90 version, see that version. no
c     kidding.
c
C*****************************************************************
C
      IF(2**INT(ALOG(FLOAT(nobs))/ALOG(2.0)+0.01).NE.nobs) THEN
        WRITE(6,1030)
        STOP
      ENDIF
C
C.....CALCULATE AND REMOVE THE MEAN
C
      XMEAN1=TSUM/FLOAT(xynobs)
      DO IY=1,xynobs
         X(1,IY)=X(1,IY)-XMEAN1
      enddo
c      WRITE(25,1020) XMEAN1
C
C.....WINDOW THE EDGES VIA DATWND
C
      CALL DATWND(PRCNT,xynobs,nobs,fold)
c
c.....MATRIX IS NOW ZERO FILLED TO NXOUT BY NYOUT SIZE
C     CALCULATE AND REMOVE THE MEAN INTRODUCED BY TAPERING
C
      TSUM=0.D0
      DO IY=1,nobs
         TSUM=TSUM+X(1,IY)
      enddo
      XMEAN2=TSUM/FLOAT(nobs)
      DO IY=1,nobs
```

```
      X(1,IY)=X(1,IY)-XMEAN2
     enddo
c     WRITE(25,1020) XMEAN2
     XMEAN=XMEAN2+XMEAN1
c     WRITE(25,1080) XMEAN
     write (25,*) passno,xmean1,xmean2,xmean
c
c.....TRANSFORM DATA TO THE WAVENUMBER DOMAIN
c
     CALL FFT1D(nobs,-1)
c
     mean=xmean
     if (num .eq. 1) then
        do ix=1,nobs
           xcdata(ix) = h(ix)
        enddo
     elseif (num .eq. 2) then
        do ix=1,nobs
           ycdata(ix) = h(ix)
        enddo
     endif
c
     return
c
1020  FORMAT('MEAN REMOVED ',F15.7)
1030  FORMAT(1H , 'nobs MUST BE A POWER OF 2: SPA2FRQ FATAL')
1080  FORMAT('TOTAL MEAN REMOVED ',F15.7)
c
     END
c
c*********************************************************************
c
     SUBROUTINE FFT1D (nobs,NSIGN)
c
c ********************************************************************
c                                                                    *
c        "FFT1D" PERFORMS BOTH A FORWARD OR INVERSE FAST FOURIER     *
c     TRANSFORM. "FFT1D" IS THE DRIVER THAT PASSES THE CORRECT VECTORS *
c     TO "FORK" WHICH PERFORMS THE ACTUAL TRANSFORMING.              *
c    THE DIMENSIONING OF "H" MUST BE THE SAME AS IN THE MAIN PROGRAM *
c                                                                    *
c  "nobs"    - NUMBER OF fft observations IN DATA MATRIX             *
c  "NSIGN" - DIRECTION OF DESIRED TRANSFORMATION                     *
c          =+1   INVERSE TRANSFORM (FREQUENCY TO SPATIAL)            *
c          =-1   FORWARD TRANSFORM (SPATIAL TO FREQUENCY)            *
c                                                                    *
c ********************************************************************
c
     COMMON  H(4096)
     COMPLEX H
c
     SIGNI=FLOAT(NSIGN)
     IF(IABS(NSIGN).NE.1) THEN
        WRITE(6,105)
        STOP
     ENDIF
c
     CALL FORK (nobs,H,SIGNI)
c
     RETURN
105  FORMAT(5X,'"NSIGN" MUST EQUAL +1 OR -1 FOR "FFT2D", FATAL')
     END
c
c*********************************************************************
c
     SUBROUTINE FORK (LXX,CX,SIGNI)
c
c*********************************************************************
c                                                                    *
c  FAST FOURIER TRANSFORM, MODIFIED FROM CLAERBOUT, J.F., FUNDAMENTAL *
c  OF GEOPHYSICAL DATA PROCESSING, MCGRAW-HILL, 1976                 *
c     FORK USES COOLEY-TUKEY ALGORITHM.                              *
c                                                                    *
c "CX"    - DATA VECTOR TO BE TRANSFORMED                            *
c "LXX"   - LENGTH OF DATA VECTOR "CX" TO BE TRANSFORMED,            *
c            MUST BE A POWER OF 2 (LXX=2**INTEGER)                   *
c "SIGNI"- DIRECTION OF DESIRED TRANSFORMATION                       *
c          =+1.   INVERSE TRANSFORM (FREQUENCY TO SPATIAL)           *
c          =-1.   FORWARD TRANSFORM (SPATIAL TO FREQUENCY)           *
c                                                                    *
c  NORMALIZATION PERFORMED BY DIVIDING BY                            *
c      DATA LENGTH UPON THE FORWARD TRANSFORM.                       *
c*********************************************************************
c
     COMPLEX CX(LXX),CW,CTEMP,CON2
c
     LX=LXX
     LXH=LX/2
     J=1
     DO 103 I=1,LX
```

```
        IF (I.LT.J) THEN
          CTEMP=CX(J)
          CX(J)=CX(I)
          CX(I)=CTEMP
        ENDIF
        M=LXH
102     IF (J.GT.M) THEN
          J=J-M
          M=M/2
          IF (M.GE.1) GO TO 102
        ENDIF
        J=J+M
103   CONTINUE
      L=1
104   ISTEP=2*L
      CON2=(0.0,3.14159265)/FLOAT(L)*SIGNI
      DO 105 M=1,L
        CW=CEXP(CON2*FLOAT(M-1))
        DO 105 I=M,LX,ISTEP
          CTEMP=CW*CX(I+L)
          CX(I+L)=CX(I)-CTEMP
105   CX(I)=CX(I)+CTEMP
      L=ISTEP
      IF (L.LT.LX) GO TO 104
      IF (SIGNI.GT.0.0) RETURN
         SC=1./FLOAT(LX)
c
      DO 106 I=1,LX
106      CX(I)=CX(I)*SC
c
       RETURN
       END
c
c***********************************************************************
c
      SUBROUTINE DATWND (PRCNT,xynobs,nobs,fold)
      integer xynobs
c
c***********************************************************************
c                                                                      *
c  "DATWND" MULTIPLIES THE INPUT F(1,xynobs) BY A HALF BELL OF A HAMMIN*
c       TUKEY WINDOW ON ALL EDGES AND ZEROS OUT THE REMAINDER OF THE   *
c       (NX,NY) ARRAY.                                                 *
c                                                                      *
c "PRCNT" =PERCENTAGE OF DATA TO BE ALTERED IN SMOOTHING TO ZERO       *
c            0.0 .LT. "PRCNT" .LE. 50.0                                *
c
c  update 2 feb 91
c     datwnd has been considerably improved such that now the subroutine
c     performs three (count them, three !!) functions.  one; a percentage
c     of the input matrix can be folded out.  two; after folding out,
c     a new percentage of the folded out matrix (or regular data if
c     folding was not performed) can be smoothed to zero.  three; the
c     manipulated data is centered within zeros to finish filling the
c     matrix to nx by ny size.  because the actual data is now centered
c     within the transformed array, it is necessary to use the
c     do loops in subroutine inverseft to correctly extract the actual
c     data
c
c***********************************************************************
c
      dimension holdme(4096)
      COMMON  F(2,4096)
c
c-------------------------- fold out the data based on percentage
c
      nx1=xynobs
      if (fold.gt.0.0 .and. fold.lt.100.0) then
         KX=Int(fold*FLOAT(xynobs)/100.0+0.5)
         if (kx+xynobs .gt. nobs) kx=(nobs-xynobs)/2
         do i=1,xynobs
            holdme(i)=f(1,i)
         enddo
c-------------------------- fold out the observations
         do i=1,xynobs+kx+kx
            if (i.le.kx) f(1,i)=holdme(kx-i+1)
            if (i.gt.kx .and. i.le.(kx+xynobs)) f(1,i)=holdme(i-kx)
            if (i.gt.(kx+xynobs)) f(1,i)=holdme((2*xynobs+kx+1-i))
         enddo
         nx1=xynobs+2*kx
      endif
c
      if (prcnt.gt.0.0 .and. prcnt.lt.50.0) then
         KX=IFIX(PRCNT*FLOAT(NX1)/100.0+0.5)
c
         IF(KX.NE.0) THEN
            RKXPI= 3.14159265/FLOAT(KX)
            DO IX=1,KX
               FACTOR=0.5*(1.0+COS(FLOAT(KX-IX+1)*RKXPI))
               IXX=NX1-IX+1
               F(1,IXX)=F(1,IXX)* FACTOR
```

```
         enddo
       ENDIF
c
c       WRITE(25,150) KX, KY
          write (25,*) kx,ky
      endif
c
c.....center and ZERO OUT REMAINDER OF ARRAY IF NECESSARY
c
      nxhalf=(nobs-nx1)/2
      do i=1,nx1
         holdme(i)=f(1,i)
      enddo
      do i=1,nxhalf
         f(1,i)=0.0
      enddo
      do i=nxhalf+1,nxhalf+nx1
         f(1,i)=holdme(i-nxhalf)
      enddo
      do i=nxhalf+nx1+1,nobs
         f(1,i)=0.0
      enddo
c
      RETURN
c
150   FORMAT('smoothed',i4,' values on both x edges'/,
     >       '        ',i4,'                 y       ')
c
      END
c
c
c
      subroutine filter (num)
      integer num,npass,imean,nwind,nobs
      real prcnt,xlag,delta,cuthi,cutlo
      complex xcdata(4096),ycdata(4096),cdata(4096)
      common /rowcol/ xnobs,ynobs
      common /fftifft/ nobs,prcnt,imean,fold
      common /comps/ xcdata,ycdata
      common /lhbflt/ delta,cuthi,cutlo,xlag,npass,nwind
      COMMON  H(4096)
      COMPLEX H
      DIMENSION D1(2,4096)
      EQUIVALENCE (D1(1,1),H(1))
c
      if (num .eq. 1) then
         do i=1,nobs
            cdata(i) = xcdata(i)
         enddo
      elseif (num .eq. 2) then
         do i=1,nobs
            cdata(i) = ycdata(i)
         enddo
      endif
C*********************************************************************
C
C                               PROGRAM BANDPASS
C
C     PROGRAM BANDPASS PERFORMS HIGH, LOW, OR BANDPASS WAVENUMBER
C     FILTERING OF UNIFORMLY GRIDDED ARRAYS.  AN IDEAL FILTER
C     IS CONSTUCTED IN THE WAVENUMBER DOMAIN, WINDOWED IN THE SPACE
C     DOMAIN, THEN TRANSFORMED BACK INTO THE WAVENUMBER DOMAIN TO BE
C     MULTIPLIED BY THE INPUT TRANSFORM.
C
C.....REQUIRED SUBROUTINES :
C
C         BNDPAS, FFT2D, FORK, STORE, WINDOW
C
C.....DIMENSIONING REQUIREMENTS :
C
C         D1(2,N).....WHERE N IS THE NUMBER OF observations OF THE
C         H(N)            INPUT AND OUTPUT TRANSFORMED MATRIX. N MUST BE AN
C                         INTEGRAL POWER OF TWO (2,4,8,16...).
C                         NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
C                         SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C.....AUTHOR : JON REED, PURDUE UNIVERSITY, DECEMBER 1980.
C              REVISIONS BY STEVE MATESKON AND JEFF LUCIUS,
C                       OHIO STATE UNIVERSITY, JULY 1984.
C
c
c     this program, like others in the fft series, has been updated
c     to the DEC workstation system and now the program is actually
c     usable to just about anybody!  revised 21 apr 90
c
c     well, like the other programs in this package, this has been
c     updated on 4 AUG 90.  few comments have been removed - mainly
c     those comments about i/o operations not necessary to this
c     package have been removed.
c
```

```
c     update: 2 feb 90, removed need for cstore array
c
c
C*********************************************************************
C
C.....CREATE IDEAL CONTINUATION FILTER AND STORE IN ARRAY H
C
      CALL BNDPAS (CUTLO,CUTHI,NPASS,DELTA,nobs)
C
C.....CREATE SMOOTHED FILTER
C
      IF(XLAG.GT.0.0 .AND. XLAG.LE.100.0) THEN
        IF(NWIND.GT.0.AND.NWIND.LE.4) THEN
          CALL FFT1D (nobs,1)
          CALL WINDOW (nobs,XLAG,NWIND)
          CALL FFT1D (nobs,-1)
        ENDIF
      ENDIF
C
C.....WRITE FILTER (WAVENUMBER DOMAIN) ONTO UNIT 30 IF IOFIL = 1
C
c      IF(IOFIL.EQ.1) THEN
c         WRITE (30,*) nobs,IZERO,XMEAN
c         WRITE (30,*) (H(IX),IX=1,nobs)
c      ENDIF
C
      if (num .eq. 1) then
         do i=1,nobs
            xcdata(i) = cdata(i)*h(i)
         enddo
      elseif (num .eq. 2) then
         do i=1,nobs
            ycdata(i) = cdata(i)*h(i)
         enddo
      endif
c
      return
      end
c
C*********************************************************************
C
      SUBROUTINE BNDPAS (CCUTLO,CCUTHI,NPASS,DELTA,nobs)
C
C*********************************************************************
C                                                                    *
C     "BNDPAS" CALCULATES THE WAVE# RESPONSE OF AN IDEAL BANDPASS
c     FILTER OF A (nobs) MATRIX.  ARRAY "H" MUST BE DIMENSIONED THE
c     SAME AS IN THE MAIN PROGRAM
C                                                                    *
C "CCUTLO"   LOWEST  WAVE# TO BE PASSED, GE 0.0                      *
C "CCUTHI"   HIGHEST WAVE# TO BE PASSED, LE NYQUIST                  *
C "NPASS"    SWITCHES EITHER A PASS OR REJECTION BETWEEN             *
C            "CUTLO" & "CUTHI"                                       *
C             =-1 REJECT WAVENUMBERS BETWEEN THE 2 WAVENUMBERS       *
C             = 1 PASS   WAVENUMBERS BETWEEN THE 2 WAVENUMBERS       *
C "DELTA"    DATA GRID INTERVAL, IN MAP UNITS                        *
c  nobs      number of fft observations
C                                                                    *
C*********************************************************************
C
      COMMON  H(4096)
      COMPLEX H,ZERO,ONE
C
      CUTHI=CCUTHI
      CUTLO=CCUTLO
      RCUTLO=999999.99
      IF(CUTLO.GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
      RCUTHI=1.0/CUTHI
      WAVLEN=2.0*DELTA
      FNQ1=1.0/WAVLEN
 c     WRITE (25,112) FNQ1,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI,NPASS
C
      IF(IABS(NPASS).NE.1) THEN
        WRITE(6,151)
        STOP
      ENDIF
      IF(CUTHI.GT.FNQ1.OR.CUTHI.LE.CUTLO.OR.CUTLO.LT.0.0) THEN
        WRITE(6,151)
        STOP
      ENDIF
 C
      NXX=nobs+2
      NX2=(nobs/2)+1
      ZERO = (0.0,0.0)
      ONE  = (1.0,0.0)
      IF(NPASS.NE.1) THEN
         ZERO = (1.0,0.0)
         ONE  = (0.0,0.0)
      ENDIF
 C
```

```
      CLOWX=FLOAT(NX2)*WAVLEN*CUTLO
      CHIX =FLOAT(NX2)*WAVLEN*CUTHI
C
C....."ZERO" OUT THE entire ARRAY
C
      DO IX=1,nobs
         H(IX) = ZERO
      enddo
C
C.....OPERATE ON ROWS WHERE WAVENUMBERS ARE .LE. CUTLO
C
      MINS=1
      MAXS=NX2
      IF(CUTLO.GT.0.000001) MINS= int(clowx + 2.0001)
      IF(FNQ1-CUTHI.GT.0.000001) MAXS=int(chix + 1.0001)
      if (mins .le. maxs) then
         DO IX=MINs,MAXs
            H(NXX-IX)= ONE
            H(IX)= ONE
         enddo
      ENDIF
C
      RETURN
C
112   FORMAT(/1X,'NYQUIST WAVENUMBER =',F10.5,'CYCLES PER DATA INTERVAL'
     >       /1X,'NYQUIST WAVELENGTH = ',F10.5,' LENGTH INTERVALS'/
     >        1X,'LOW  WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >   ' CYCLES PER DATA INTERVAL',3X,F15.5,' WAVELENGTH EQUIVALENT'/
     >        1X,'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >        ' CYCLES PER DATA INTERVAL',3X,F15.5,
     >        ' WAVELENGTH EQUIVALENT',/1X,'NPASS= ',I1//)
151   FORMAT(5X ,'IMPOSSIBLE FILTER CONSTRUCTION IS SPECIFIED. FATAL')
C
      END
C
C
C**********************************************************************
C
      SUBROUTINE WINDOW (nobs,XLAG,NWIND)
C
C**********************************************************************
C "WINDOW" PERFORMS 1-DIMENSION WINDOWING OVER the DATA ARRAY
C  EACH QUAD. IS SEPERATELY WINDOWED. THE 1.0 COEFFICENT IS ALWAYS
C  THE OUTER MOST CORNER OF THE ARRAY.
C
C  "nobs"    = NUMBER OF observations  IN DATA MATRIX
C  "XLAG"  = SMOOTHING PARAMETER FOR WINDOWING IDEAL FILTER IN SPATIAL *
C               DOMAIN. DETERMINES WHAT PERCENTAGE OF DATA IS WINDOWED *
C               (nobs*XLAG/100.0) THE REMAINDER IS SET TO 0.0. I.E. THE
C                    SMALLER "XLAG" THE SMOOTHER THE WINDOWING.        *
C            "XLAG" MUST BE .GT. 0.0 .AND. LE. 100.0 FOR WINDOWING     *
C               VALUES OUTSIDE OF THIS RESULTS IN NO WINDOWING         *
C               THE SMALLER THE "XLAG" THE SMOOTHER THE FILTER.        *
C                                                                      *
C  "NWIND" = TYPE OF WINDOW TO APPLY                                   *
C            =0 GIVES NO WINDOW                                        *
c            =1 gives a rectangular window
C            =2 GIVES BARTLETT WINDOW(TRIANGLE WINDOW)                 *
C            =3 GIVES HAMMING-TUKEY WINDOW                             *
C            =4 GIVES PARZEN WINDOW                                    *
C                                                                      *
C **********************************************************************
C
      COMMON  H(4096)
      COMPLEX H
C
      LAG=FLOAT(nobs)*XLAG/200.0+0.5
      LAG=AMAX0(LAG,2)
      PI=3.14159265
      NXX=nobs+2
      NX2=(nobs/2)+1
      XNXR=FLOAT(NX2)
      XNX=1.0/FLOAT(NX2)
C
      RADIUS=FLOAT(LAG)*XNX
      RADI=1.0/RADIUS
      RAD2= RADIUS*RADIUS
C
C.....APPLY RECTANGULAR WINDOW TO FILTER
C
      IF(NWIND.EQ.1) THEN
         MAX=int(RADIUS*XNXR+1.0001)
         LL=MAX+1
         DO II=LL,NX2
            H(NXX-II)=(0.0,0.0)
            H(II)= (0.0,0.0)
         enddo
c        WRITE(25,660) XLAG,LAG
         RETURN
C
```

```
C.....APPLY BARTLETT WINDOW TO FILTER
C
      ELSEIF(NWIND.EQ.2) THEN
          MAX=RADIUS*XNXR+1.0001
          IF(MAX.GE.2) THEN
            DO 253 LL=2,MAX
              XI=FLOAT(LL-1)*XNX
              FACTOR=1.0-XI*RADI
              H(LL)=H(LL)*FACTOR
              MX=NXX-LL
253         H(MX)=H(MX)*FACTOR
          ENDIF
C
          LL=MAX+1
          DO 255 II=LL,NX2
            H(NXX-II)=(0.0,0.0)
 255      H(II)=(0.0,0.0)
C
c          WRITE(25,661) XLAG,LAG
          RETURN
C
C.....APPLY HAMMING-TUKEY WINDOW TO FILTER
C
      ELSEIF(NWIND.EQ.3) THEN
          PIRADI=PI*RADI
          MAX=RADIUS*XNXR+1.0001
          IF(MAX.GE.2) THEN
            DO 353 LL=2,MAX
              XI=FLOAT(LL-1)*XNX
              FACTOR=0.5*(1.0+COS(PIRADI*XI))
              H(LL)=H(LL)*FACTOR
              MX=NXX-LL
353         H(MX)=H(MX)*FACTOR
          ENDIF
C
          LL=MAX+1
          DO 355 II=LL,NX2
            H(NXX-II)=(0.0,0.0)
355       H(II)=(0.0,0.0)
C
c          WRITE(25,662) XLAG,LAG
          RETURN
C
C.....APPLY PARZEN WINDOW TO FILTER
C
      ELSEIF(NWIND.EQ.4) THEN
          MAX=RADIUS*XNXR+1.0001
          MAX2=SQRT(RAD2/4.0)*XNXR+1.0001
          FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
          H(1)=H(1)*FACTOR
C
          IF(MAX2.GE.2) THEN
            DO 453 LL=2,MAX2
              XI=FLOAT(LL-1)*XNX
              FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
              H(LL)=H(LL)*FACTOR
              MX=NXX-LL
453         H(MX)=H(MX)*FACTOR
          ENDIF
C
          KOUNT=MAX2+1
          DO 457 LL=KOUNT,MAX
            XI=FLOAT(LL-1)*XNX
            FACTOR=2.0*(1.0-(XI*RADI))**3
            H(LL)=H(LL)*FACTOR
            MX=NXX-LL
457       H(MX)=H(MX)*FACTOR
          LL=MAX+1
          DO 455 II=LL,NX2
            H(NXX-II)=(0.0,0.0)
455       H(II)=(0.0,0.0)
C
c          WRITE(25,663) XLAG,LAG
          RETURN
C
C.....DO NOT APPLY A WINDOW TO FILTER
C
      ELSEIF(NWIND.EQ.0) THEN
c        WRITE(25,664)
      ENDIF
C
      RETURN
C
660   FORMAT ('RECTANGULAR WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
661   FORMAT ('BARTLETT WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
662   FORMAT ('HAMMING-TUKEY WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
663   FORMAT ('PARZEN WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
664   FORMAT ('NO WINDOWING HAS BEEN APPLIED ; XLAG= ',F7.3)
C
      END
```

```
c
c
      subroutine correlate(xpassno,ypassno)
      integer xnobs,ynobs,nobs,match,xpassno,ypassno,
     >        zerocnt(4096),cnwind
      real mincc,maxcc,ccwinout,prcnt,
     >     pctpr3,pctpr4,minccin,maxccin,cxlag
      complex h(4096),power5,power6,totpwr
      COMPLEX  X(4096),Y(4096),zero,
     >        POWER1,POWER2,POWER3,POWER4,XPOWER,TPOWER
      REAL CCOEF,CCIN,CCOUT
      DATA ZERO/(0.000000,0.000000)/
      common /rowcol/ xnobs,ynobs
      common /comps/ x,y
      common /ccflt/ mincc,maxcc,match,minccin,maxccin,cnwind,cxlag
      common /fftifft/ nobs,prcnt,imean,fold
      common h
c
c-------------------------- subroutine description
c
c     correlate finds the correlation coefficient between each
c     wavenumber component of the two input arrays.  each cc is
c     normalized to range between -1.0 through 0.0 to 1.0.  the
c     cc is the cosine of the phase angle difference between two
c     wavenumber components.
c
c     revisions:
c
c     well, by now you know the story... revised 4 AUG 90
c     i've added the windowing functions available from the
c     bandpassing subroutines to this cc-filter.  try them if
c     you like!
c
c     updates 1 feb 91:  change calculation of correlation
c     coefficient from a summation based formula to the cosine of
c     the phase angle difference.
c
c
      if (match .eq. 1) then
         if (xpassno .ne. ypassno) then
            write (*,*) 'NO MATCH BETWEEN PASS NUMBERS',xpassno,ypassno
         endif
      endif
      if (xnobs .ne. ynobs ) then
         write (*,*) 'NO MATCH BETWEEN NUMBER OF OBSERVATIONS'
         write (*,*) 'CORRELATION COEF MAY NOT BE CORRECT'
         write (*,*) 'PASSNUMBERS=',xpassno,ypassno
         write (*,*) 'FILE 1: OBSERVATIONS =',xnobs
         write (*,*) 'FILE 2: OBSERVATIONS =',ynobs
      endif
c
      pi=3.141592654
      twopi=6.283185307
      POWER1=ZERO
      POWER2=ZERO
      POWER3=ZERO
      POWER4=ZERO
      XPOWER=ZERO
      TPOWER=ZERO
c
      DO 110 i=1,nobs
c-------------------------------- zerocnt array is a flagging array used to
c                                 set the windowing array h to equal
c                                 (0.0,0.0) or (1.0,0.0).  a little inspection
c                                 of subroutine BNDPAS will help illuminate
c                                 the principle.
         zerocnt(i)=1
C
C  SUM THE POWERS & CROSS PRODUCTS OF THE INPUT MAPS.
C
         POWER1=POWER1+(X(I)*CONJG(X(I)))
         POWER2=POWER2+(Y(I)*CONJG(Y(I)))
         XPOWER=XPOWER+(X(I)*CONJG(Y(I)))
c
c------- xrad is the phase angle of the x array wavenumber and
c        yrad is the phase angle of the y array wavenumber.  the
c        cosine of the minimum phase difference is the correlation
c        of the two wavenumbers.  to find the minimum phase difference
c        it is necessary to adjust xrad or yrad with integer values
c        of pi.  so....do not change the order of the if statements !!
c
         xrad=atan(aimag(x(i))/(real(x(i))))
         if (real(x(i)).lt.0.0) xrad=xrad+pi
         if (aimag(x(i)).lt.0.0) xrad=xrad+twopi
         yrad=atan(aimag(y(i))/(real(y(i))))
         if (real(y(i)).lt.0.0) yrad=yrad+pi
         if (aimag(y(i)).lt.0.0) yrad=yrad+twopi
         delrad=abs(xrad-yrad)
         ccoef=cos(delrad)
c
```

```
         IF (CCOEF .GT. maxcc  .or.  CCOEF .LT. mincc) THEN
            X(I)=ZERO
            Y(I)=ZERO
            zerocnt(i)=0
         ENDIF
c
C  SUM THE POWERS & CROSS PRODUCTS FOR THE OUTPUT MAPS.
c
         POWER3=POWER3+(X(I)*CONJG(X(I)))
         POWER4=POWER4+(Y(I)*CONJG(Y(I)))
         TPOWER=TPOWER+(X(I)*CONJG(Y(I)))
 110  CONTINUE
c
C  CALCULATE THE C.C. FOR THE INPUT MAPS.
c
      if (power1 .eq. zero  .or. power2 .eq. zero) then
         write (*,*) 'power1 =',power1,xpassno
         write (*,*) 'power2 =',power2,ypassno
         ccin=9999.9
      else
         CCIN=REAL(XPOWER/SQRT(POWER1*POWER2))
      endif
c
C  CALCULATE THE C.C. FOR THE OUTPUT MAPS.
c
      if (power3 .eq. zero  .or.  power4 .eq. zero) then
         write (*,*) 'power3 =',power3,xpassno
         write (*,*) 'power4 =',power4,ypassno
         ccout=9999.9
      else
         CCOUT=REAL(TPOWER/SQRT(POWER3*POWER4))
      endif
c
C  CALCULATE THE PERCENTAGE OF THE POWER RETAINED IN THE FILTERED
C  MAPS.
c
      if (power1 .eq. zero  .or. power2 .eq. zero) then
         pctpr1=9999.9
         pctpr2=9999.9
      else
         PCTPR1=(POWER3/POWER1)*100.0
         PCTPR2=(POWER4/POWER2)*100.0
      endif
c
C  WRITE THE C.C. FOR THE INPUT & OUTPUT MAPS TO FILE 6.
c
c      WRITE (6,444) CCIN
c      WRITE (6,555) CCOUT
c
C  WRITE THE POWER PERCENTAGES TO FILE 6.
c
c      WRITE (6,666) PCTPR1,PCTPR2
c 444  FORMAT (' ','THE CORRELATION COEFFICIENT BETWEEN THE INPUT '
c     *     ,'MAPS IS ',F6.3)
c 555  FORMAT (' ','THE CORRELATION COEFFICIENT BETWEEN THE OUTPUT '
c     *     ,'MAPS IS ',F6.3)
c 666  FORMAT (' ','THE PERCENTAGE OF THE TOTAL POWER IN MAP ONE',
c     *' PASSED IS',F7.3,'%',/,' THE PERCENTAGE OF THE TOTAL POWER',
c     *' IN MAP TWO PASSED IS',F7.3,'%')
c
c
      write (25,888) xpassno,ypassno,ccin,ccout,pctpr1,pctpr2
 888  format (2i6,4f10.3)
      if (ccin .lt. minccin) write (*,*)xpassno,ypassno,ccin,' <min'
      if (ccin .gt. maxccin) write (*,*)xpassno,ypassno,ccin,' >max'
c
c----------------------------------- the following if statement controls
c                                    the windowing functions for smoothing
c                                    the output arrays and calculates a new
c                                    output correlation coefficient and
c                                    percents of power retained in the
c                                    windowed arrays because
c                                    the data will change slightly with
c                                    windowing
c
      if (cnwind .ge. 1  .and.  cnwind .le. 4) then
         power5=zero
         power6=zero
         totpwr=zero
         do i=1,nobs
            h(i)= (1.0,0.0)
            if (zerocnt(i) .eq. 0) h(i) = (0.0,0.0)
         enddo
         call fft1d (nobs,1)
         call window (nobs,cxlag,cnwind)
         call fft1d (nobs,-1)
         do i=1,nobs
            x(i) = x(i)*h(i)
            y(i) = y(i)*h(i)
            power5=power5+(x(i)*conjg(x(i)))
```

```
            power6=power6+(y(i)*conjg(y(i)))
            totpwr=totpwr+(x(i)*conjg(y(i)))
         enddo
         if (power5 .eq. zero  .or.  power6 .eq. zero) then
            write (*,*) 'power5 =',power5,xpassno
            write (*,*) 'power6 =',power6,ypassno
            ccwinout=9999.9
            go to 340
         endif
         if (power1 .eq. zero  .or.  power2 .eq. zero) then
            pctpr3=9999.9
            pctpr4=9999.9
            go to 340
         endif
         ccwinout=real(totpwr/sqrt(power5*power6))
         pctpr3=(power5/power1)*100.0
         pctpr4=(power6/power2)*100.0
 340     continue
         write (25,888) xpassno,ypassno,ccin,ccwinout,pctpr3,pctpr4
c 888    format (2i6,4f10.3)
      endif
c
      return
      end
c
c
c
      subroutine inverseft (num,mean,passno)
      integer num,xnobs,ynobs,xynobs,row,col,passno
      real xdata(4096),ydata(4096),mean
      complex xcdata(4096),ycdata(4096)
      common /rowcol/ xnobs,ynobs
      common /reals/ xdata,ydata
      common /comps/ xcdata,ycdata
      common /fftifft/ nobs,prcnt,imean,fold
      COMMON  H(4096)
      DIMENSION X(2,4096),holdme(4096)
      COMPLEX H
      EQUIVALENCE (X(1,1),H(1))
c
      if (num .eq. 1) then
         xynobs=xnobs
         do i=1,nobs
            h(i) = xcdata(i)
         enddo
      elseif (num .eq. 2) then
         xynobs=ynobs
         do i=1,nobs
            h(i) = ycdata(i)
         enddo
      endif
c
C-------------------------------subroutine description
C.....REQUIRED SUBROUTINES :
C
C        FFT1D, FORK
C
C.....DIMENSIONING REQUIREMENTS :
C
C        X(2,N).....WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
C        H(N)       OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
C                   POWER OF TWO (2,4,8,256...).
C                   NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
C                   SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C.....AUTHOR : JEFF LUCIUS
C              DEPARTMENT OF GEOLOGY AND MINERALOGY
C              OHIO STATE UNIVERSITY, DECEMBER 1984.
C
c     revised: 8 AUG 90
c     updated: 2 feb 91
c              added do loops that find the data portion of the
c              zero centered inverse transformed data.  a look at
c              subroutine datwnd will help figure this out.
c
c
C.....INVERSE TRANSFORM DATA TO THE SPACE DOMAIN
c
      CALL FFT1D (nobs,+1)
c
      nxhalf=(nobs-xynobs)/2
      do i=nxhalf+1,nxhalf+xynobs
         holdme(i-nxhalf)=x(1,i)
      enddo
c
      total=0.0
      DO I=1,xynobs
         x(1,i)=holdme(i)
         total=total+x(1,i)
      enddo
      xmean=total/float(xynobs)
```

```
      IF(IMEAN .EQ. 1) THEN
         do i=1,xynobs
            x(1,i)=x(1,i)+mean
         enddo
      ENDIF
c
      XMIN= 1.0E20
      XMAX=-1.0E20
      if (num .eq. 1) then
         do i=1,xynobs
            xdata(i) = x(1,i)
         enddo
      elseif (num .eq. 2) then
         do i=1,xynobs
            ydata(i) = x(1,i)
         enddo
      endif
      DO I=1,xynobs
          XMIN=AMIN1(XMIN,X(1,I))
          XMAX=AMAX1(XMAX,X(1,I))
          IF(XMAX.EQ.X(1,I)) IMAX=I
          IF(XMIN.EQ.X(1,I)) IMIN=I
      enddo
c       WRITE(25,1020) XMAX,IMAX,XMIN,IMIN,xmean,passno
      write (25,9980) passno,xmean,xmax,imax,xmin,imin
 9980 format (i5,2x,f13.5,2x,f13.5,2x,i4,f13.5,2x,i4)
c
1020  FORMAT('MAXIMUM OF IFFT = ',E15.7,' AT (',I3,')'/,
     >       'MINIMUM = ',E15.7,' AT (',I3,')'/,
     >       'MEAN AFTER IFFT = ',e15.7,'  FOR PASS',i6,/)
c
      return
      END
```

```
      program combine
      integer xrow,xcol,zero,eight,xpassno,ypassno,ycol,
     >        x3row,x3col,x3pass,y3row,y3col,y3pass,yrow,
     >        strcnt,syrow,sy3row,sycol,sy3col,nobs,noc,
     >        sypassno,sy3pass,dndk,tcount,paircnt,minobs,
     >        paircnt1,choice,totobs,prime,global,crosscnt,
     >        passrem,nowant(4000),nocnt,type
      real xmean,ymean,x3data(400,3),y3data(400,3),y3mean,
     >     x1data(400),y1data(400),xadata(400,4),ybdata(400,4),
     >     x3mean,sy3mean,symean,x(400),y(400),
     >     savey1(400),savey3(400,3),xamean,
     >     ybmean,avgdata(400,4)
      character*80 filename
      common /trunstat/ xadata(400,4),ybdata(400,4)
      common x3data(400,3),x1data(400),y3data(400,3),y1data(400)
      common /aver/ avgdata(400,4)
      common /tplot/ x(400),y(400)
      common /nope/ nowant(4000),nocnt
c
c---------------------------------------- program description
c
c        combine is very similar to movetrunc in that both
c        programs truncate adjacent passes to the same over-
c        lapping segments.  both programs also provide
c        statistics and track output files.  the major
c        difference is that movetrunc has only one file
c        as input whereas combine has two files as input.
c        combine can output one file of two passes averaged
c        together to make the one file or it can output
c        two similar length passes that can be further
c        processed by fourier methods.  movetrunc and combine
c        could be cluged together to make one program
c        so why don't you go ahead and jam them together??
c        good luck!!
c
c      program date: 16 apr 91
c
      write (*,*) '1 TO HAVE ONE OUTPUT FILE'
      write (*,*) '2 TO HAVE TWO OUTPUT FILES'
      read (*,*) ifilenum
      if (ifilenum .eq. 2) then
         prime=1
         goto 3
      endif
      write (*,*) '1 TO AVERAGE A-east AND A-west'
      write (*,*) '2 TO AVERAGE A-east AND B-west (choose 2)'
      read (*,*) prime
 3    write (*,*) '0 IF THE DATA SET IS GLOBAL OR POLAR'
      write (*,*) '1 IF THE DATA SET DOES NOT INCLUDE ALL LONGITUDES'
      read (*,*) global
c
      if (prime .eq. 1) then
         write (*,*) 'NOTE: FILE Y WILL HAVE THE FIRST PASS MOVED'
         write (*,*) 'TO THE BOTTOM OF THE FILE'
         write (*,*) ' '
         if (global .eq. 1) then
           write (*,*) 'AND THE FIRST PASS WILL NOT BE INCLUDED IN'
           write (*,*) 'THE PROCESSING.  FILE X WILL HAVE THE LAST'
           write (*,*) 'PASS REMOVED AND THIS PASS WILL NOT BE INCLUDED'
           write (*,*) 'IN THE PROCESSING'
         endif
      endif
c
      write (*,*) 'INPUT FILE X OF LAT-LONG-RAD DATA'
      read (*,9990) filename
 9990 format(a80)
      open (10, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT FILE X OF MAGNETIC VARIABLES'
      read (*,9990) filename
      open (11, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT FILE Y OF LAT-LONG-RAD DATA'
      read (*,9990) filename
      open (12, file=filename,status='old',form='unformatted')
      write (*,*) 'INPUT FILE Y OF MAGNETIC VARIABLES'
      read (*,9990) filename
      open (13, file=filename,status='old',form='unformatted')
c
      if (ifilenum .eq. 1) then
      write (*,*) 'OUTPUT FILE OF TRUNCATED LAT-LONG-RAD-ANOM DATA'
      write (*,*) 'AND  ---------NO HEADERS TO BE WRITTEN---------'
      read (*,9990) filename
      open (20, file=filename,form='formatted')
      write (*,*) 'OUTPUT FILE OF LAT-LON TO BE RUN IN TPLOT'
      read (*,9990) filename
      open (21, file=filename,form='unformatted')
c
      elseif (ifilenum .eq. 2) then
      write (*,*) 'OUTPUT FILE X OF LAT-LON-RAD'
      read (*,9990) filename
      open (30, file=filename, form='unformatted')
```

```
      write (*,*) 'OUTPUT FILE X OF VARIABLE'
      read (*,9990) filename
      open (31, file=filename, form='unformatted')
      write (*,*) 'OUTPUT FILE Y OF LAT-LON-RAD'
      read (*,9990) filename
      open (32, file=filename, form='unformatted')
      write (*,*) 'OUTPUT FILE Y OF VARIABLE'
      read (*,9990) filename
      open (33, file=filename, form='unformatted')
      endif
c
      write (*,*) 'OUTPUT FILE OF STATISTICS'
      read (*,9990) filename
      open (25, file=filename,form='formatted')
c
      write (*,*) '0 IF THESE ARE DUSK DATA SETS'
      write (*,*) '1 IF THESE ARE DAWN DATA SETS'
      read (*,*) dndk
      if (ifilenum .eq. 1) then
      write (*,*) '0 DO NOT REMOVE THE MEAN FROM THE AVERAGED DATASET'
      write (*,*) '1 REMOVE THE MEAN'
      read (*,*) choice
      endif
      write (*,*) '0 IF ALL PASSES ARE WANTED'
      write (*,*) '1 TO REMOVE THE PASSES THAT ARE NOT WANTED'
      read (*,*) nocnt
      if (nocnt .eq. 1) then
         write (*,*) 'INPUT FILE OF PASSES NOT WANTED'
         read (*,9990) filename
         open (14, file=filename,form='formatted',status='old')
         do 5 i=1,4000
            read (14,*,end=6) nowant(i)
 5       continue
 6       continue
         nocnt=i-1
      endif
      if (prime .eq. 1) then
         write (*,*) '1 FOR A LATITUDE GAP FINDER'
         write (*,*) '2 FOR A MINIMUM OBSERVATION GAP FINDER'
         read (*,*) type
      endif
      write (*,*) 'AND FINALLY! MINIMUM NUMBER OBSERVATIONS PER PASS'
      read (*,*) minobs
c
      if (prime .eq. 1) then
         write (*,*)'---------------------------------------------'
         write (*,*)'running through dataset to find passes that',
     >               ' do not overlap'
c------------------------------------- findgap locates adjacent
c                                       passes that do not overlap
         call findgap (global,dndk,minobs,type)
         write (*,*) 'done with run through'
         write (*,*)'---------------------------------------------'
      endif
c
      write (25,*) 'XPASS YPASS CCD        CCY        XVAR       YVAR ',
     >             '     COVARXY    XSTDEV     YSTDEV'
      write (25,*) '                         XMEAN     YMEAN      XSLOPE     YSLOPE',
     >             '     XINTCPT    YINTCPT'
c
      paircnt=0
      paircnt1=0
      tcount=0
      strcnt=0
      totobs=0
      noc=0
      crosscnt=0
      passrem=0
c
 10   continue
      read (10,end=999) x3row,x3col,zero,x3mean,x3pass,eight
      do 15 i=1,x3row
         read (10) (x3data(i,ii),ii=1,x3col)
 15   continue
      read (11) xrow,xcol,zero,xmean,xpassno,eight
      do 25 i=1,xrow
         read (11) x1data(i)
 25   continue
 30   continue
      read (12,end=150) y3row,y3col,zero,y3mean,y3pass,eight
      do 35 i=1,y3row
         read (12) (y3data(i,ii),ii=1,y3col)
 35   continue
      read (13) yrow,ycol,zero,ymean,ypassno,eight
      do 45 i=1,yrow
         read (13) y1data(i)
 45   continue
c
      strcnt=strcnt+1
c------------------------- this if statement offsets the passes
```

```
c                          and saves the offset for the end
      if (strcnt .eq. 1  .and. prime .eq. 1) then
         do 50 i=1,y3row
            do 55 ii=1,y3col
               savey3(i,ii)=y3data(i,ii)
 55         continue
            savey1(i)=y1data(i)
 50      continue
         sy3row=y3row
         syrow=yrow
         sy3col=y3col
         sycol=ycol
         sy3mean=y3mean
         symean=ymean
         sy3pass=y3pass
         sypassno=ypassno
         go to 30
      endif
c
      go to 190
c
 150  continue
      if (global .eq. 1) go to 999
      do 160 i=1,sy3row
         do 165 ii=1,sy3col
            y3data(i,ii)=savey3(i,ii)
 165     continue
         y1data(i)=savey1(i)
 160  continue
      y3row=sy3row
      yrow=syrow
      y3col=sy3col
      ycol=sycol
      y3mean=sy3mean
      ymean=symean
      y3pass=sy3pass
      ypassno=sypassno
c
 190  continue
c
      if (xrow.ne.x3row .or. xpassno.ne.x3pass .or.
     >    yrow.ne.y3row .or. ypassno.ne.y3pass) then
         write (*,*) 'WACKO, TRA-LA-LA, JOLLY-GOOD,  NO MATCH BETWEEN'
         write (*,*) 'ROWS OR PASSNOS  X= ',xrow,x3row,xpassno,x3pass
         write (*,*) '                 Y= ',yrow,y3row,ypassno,y3pass
         go to 999
      elseif (xrow .lt. minobs  .or.  yrow .lt. minobs) then
        write(*,*)'FILE X PASS',xpassno,' REMOVED: OBSERVATIONS=',xrow
        write(*,*)'FILE Y PASS',ypassno,' REMOVED: OBSERVATIONS=',yrow
        passrem=passrem+1
        go to 10
      endif
      do 195 i=1,nocnt
         if (nowant(i).eq.xpassno .or. nowant(i).eq.ypassno) then
            write (*,*) 'FILE X PASS',xpassno,' REMOVED  NOT WANTED'
            write (*,*) 'FILE Y PASS',ypassno,' REMOVED  NOT WANTED'
            passrem=passrem+1
            go to 10
         endif
 195  continue
c
      call truncate (xrow,yrow,dndk,nobs,paircnt1,minobs)
c
      call statistics (nobs,xpassno,ypassno,xamean,ybmean)
c
      if (ifilenum .eq. 1) call average (nobs,choice,prime,crosscnt)
c
      if (ifilenum .eq. 2) then
         if (xpassno .ne. ypassno) then
            write (*,*) xpassno,ypassno,' no match pass numbers'
            stop 444
         endif
         ithree=3
         ione=1
         xxavg=0.0
         ieight=8888
         izero=0
         write (30) nobs,ithree,zero,xxavg,xpassno,ieight
         write (31) nobs,ione,zero,xxavg,xpassno,ieight
         write (32) nobs,ithree,zero,xxavg,ypassno,ieight
         write (33) nobs,ione,zero,xxavg,ypassno,ieight
         do j=1,nobs
            if (xadata(j,3) .gt. 180.0) xadata(j,3)=xadata(j,3)-360.0
            write (30) xadata(j,1),(xadata(j,i),i=3,4)
            write (31) xadata(j,2)
            if (ybdata(j,3) .gt. 180.0) ybdata(j,3)=ybdata(j,3)-360.0
            write (32) ybdata(j,1),(ybdata(j,i),i=3,4)
            write (33) ybdata(j,2)
         enddo
         go to 300
```

```
      endif
c
c--------------------------write out the truncated lengths passes
c
      if (prime .eq. 1) then
         do 200 j=1,nobs
            if (xadata(j,3) .gt. 180.0) xadata(j,3)=xadata(j,3)-360.0
            write (20,*) xadata(j,1),(xadata(j,i),i=3,4),avgdata(j,4)
 200     continue
         do 205 j=1,nobs-1
            if (xadata(j,3) .lt. xadata(j+1,3)) then
               crosscnt=crosscnt+1
               go to 207
            endif
 205     continue
 207     continue
      elseif (prime .eq. 2) then
         do 220 j=1,nobs
            if (avgdata(j,2).gt.180.0) avgdata(j,2)=avgdata(j,2)-360.0
            write (20,*) (avgdata(j,k),k=1,4)
 220     continue
      endif
c
c------------------------- write out the trace of the pass for plotting
c                          in tplot
      call track (nobs,noc,prime)
      WRITE (21) NOC,NObs,(X(J),Y(J),J=1,NObs)
c
 300  continue
      if (paircnt1 .gt. 0) tcount=tcount+1
      paircnt=paircnt+paircnt1
      totobs=totobs+nobs
c
      go to 10
c
 999  continue
      write (*,*) 'corrected',paircnt,' pairs of latitudes in'
      write (*,*) tcount,' passes to beginning lengths'
      write (*,*) 'total passes read = ',strcnt
      write (*,*) 'removed',passrem,' passes from the file'
      write (*,*) 'total observations in the written dataset =',totobs
      write (*,*) 'study area includes',crosscnt,' pairs of',
     >            ' longitudes that cross -180.0 180.0'
      close (10)
      close (11)
      close (12)
      close (13)
      close (20)
      close (21)
      close (25)
      close (30)
      close (31)
      close (32)
      close (33)
      stop
      end
c
c
c
      subroutine truncate (xrow,yrow,dndk,minrow,stocount,minobs)
      integer xrow,yrow,stocount,rowii,rowinc,minrow,
     >        dndk,minobs
      real xdata(400,4),ydata(400,4),
     >     x3data(400,3),x1data(400),y3data(400,3),y1data(400),
     >     adata,bdata,diffab,abss,xadata(400,4),ybdata(400,4)
      common /trunstat/ xadata(400,4),ybdata(400,4)
      common x3data(400,3),x1data(400),y3data(400,3),y1data(400)
c
c--------------------------- subroutine description
c          truncate locates the overlapping segment between two
c          adjacent passes and stores that segment in the
c          appropriate arrays
c
      do 70 j=1,xrow
         xdata(j,1)=x3data(j,1)
         xdata(j,2)=x1data(j)
         xdata(j,3)=x3data(j,2)
         xdata(j,4)=x3data(j,3)
 70   continue
      do 75 j=1,yrow
         ydata(j,1)=y3data(j,1)
         ydata(j,2)=y1data(j)
         ydata(j,3)=y3data(j,2)
         ydata(j,4)=y3data(j,3)
 75   continue
c
 80   continue
      stocount=0
      jj=1
      rowii=xrow
```

```
      rowinc=yrow
c-----------------------loops from 90 to 200 increment through the
c                       two input passes and truncate the lengths
c                       to the same length
 90   continue
      adata=xdata(jj,1)
      bdata=ydata(jj,1)
      diffab=adata-bdata
      abss=abs(diffab)
      if (rowii .eq. 0  .or.  rowinc .eq. 0) then
         write (*,*) 'xrows (ii) =',rowii,' yrows (inc) =',rowinc
         write (*,*) 'xrow =',xrow,' yrow =',yrow
         stop
      endif
      minrow=min(rowii,rowinc)
c-----------------------if pass a (ii) matches pass b (inc) at
c                       beginning length then write to xdata and
c                       ydata and race to main program
      if (abss .lt. 0.33) then
         do 110 ll=1,minrow
            do 110 kk=1,4
               xadata(ll,kk)=xdata(ll,kk)
               ybdata(ll,kk)=ydata(ll,kk)
 110     continue
         return
      endif
c-----------------------if pass a no matcha the b data then find new
c                       a or b depending on whether or not ascending
c                       or descending order of independent variable
      if (abss .ge. 0.33) then
         stocount=stocount+1
c-----------------------if this is a dusk pass then will count from
c                       -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (xdata(jj,1) .gt. ydata(jj,1)) then
               rowinc=rowinc-1
               do 130 mm=1,rowinc
                  do 130 kk=1,4
                     ydata(mm,kk)=ydata(mm+1,kk)
 130              continue
            elseif (xdata(jj,1) .lt. ydata(jj,1)) then
               rowii=rowii-1
               do 150 nn=1,rowii
                  do 150 kk=1,4
                     xdata(nn,kk)=xdata(nn+1,kk)
 150              continue
            endif
c-----------------------if this is a dawn pass then will count from
c                       the equator toward the south pole
c                       that is decreasing independent variable
         elseif (dndk .eq. 1) then
            if (xdata(jj,1) .lt. ydata(jj,1)) then
               rowinc=rowinc-1
               do 160 mm=1,rowinc
                  do 160 kk=1,4
                     ydata(mm,kk)=ydata(mm+1,kk)
 160              continue
            elseif (xdata(jj,1) .gt. ydata(jj,1)) then
               rowii=rowii-1
               do 170 nn=1,rowii
                  do 170 kk=1,4
                     xdata(nn,kk)=xdata(nn+1,kk)
 170              continue
            endif
         endif
      endif
c
      go to 90
c
      end
c
c
c
c
      subroutine statistics (minrow,xpassno,ypassno,xamean,ybmean)
      integer minrow,nobs,xpassno,ypassno
      real xadata(400,4),ybdata(400,4),nobss
      common /trunstat/ xadata(400,4),ybdata(400,4)
c-------------------------from 200 to write statement of variables is
c                         the statistical calculations using two
c                         references:
c                          1) Davis, Statistics and Data Analysis in
c                             Geology, 2nd ed., 1986 pp. 41
c                          2) Young, Statistical Treatment of Experi-
c                             mental Data, 1962, McGraw Hill, 115-132
c
c-------------------------loops that sum x, x**2, y, y**2 and xy
c                         and calculate new truncate means
      nobs=minrow
      nobss=float(nobs)
```

```
      xsum=0.0
      xsumsqr=0.0
      ysum=0.0
      ysumsqr=0.0
      sumxy=0.0
      do 240 j=1,nobs
         xsum=xsum+xadata(j,2)
         xsumsqr=xsumsqr+(xadata(j,2))**2
         ysum=ysum+ybdata(j,2)
         ysumsqr=ysumsqr+(ybdata(j,2))**2
         sumxy=sumxy+(xadata(j,2)*ybdata(j,2))
 240     continue
c        write (*,*) xsum,ysum,xsumsqr,ysumsqr,sumxy
c------------------------find corrected sum of products, covariance
c                        and corrected sum of squares (x) (y)
c
      xamean=xsum/nobss
      ybmean=ysum/nobss
      sumprod=sumxy-((xsum*ysum)/nobss)
      covarxy=sumprod/(nobss-1.0)
      xcsumsqr=xsumsqr-((xsum**2)/nobss)
      ycsumsqr=ysumsqr-((ysum**2)/nobss)
c
c------------------------find variance, standard deviation for x and y
c
      xvar=xcsumsqr/(nobss-1.0)
      yvar=ycsumsqr/(nobss-1.0)
      xstdev=sqrt(xvar)
      ystdev=sqrt(yvar)
c-----------------------find correlation coefficient by Davis method
      corrDxy=covarxy/(xstdev*ystdev)
c-----------------------find slopes, intercepts and correlation
c                       coefficient by Young method
      xslope=((nobss*sumxy)-(xsum*ysum))/((nobss*xsumsqr)-xsum**2)
      yslope=((nobss*sumxy)-(xsum*ysum))/((nobss*ysumsqr)-ysum**2)
      xintcpt=((ysum*xsumsqr)-(sumxy*xsum))/((nobss*xsumsqr)-xsum**2)
      yintcpt=((xsum*ysumsqr)-(sumxy*ysum))/((nobss*ysumsqr)-ysum**2)
      corrYxy=sqrt(xslope*yslope)
c
c------------------------write out this mess for individual pass and
c                        overlapping lengths of passes
c
c     write (25,9992)xpassno,ypassno,xvar,yvar,xstdev,ystdev,
c    >               xamean,ybmean
 9992 format ('FOR OVERLAPPING LENGTHS   X=',i5,' Y=',i5,/,
     >       'X VARIANCE=',f9.3,' Y VARIANCE=',f9.3,' X STDEV=',
     >       f9.3,' Y STDEV=',f9.3,' XMEAN=',f9.3,' YMEAN=',f9.3)
c     write (25,9993) covarxy,corrDxy
 9993 format ('COVARIANCE XY=',f9.3,' Davis CORRELATION COEF=',f9.3)
c     write (25,9994) xslope,xintcpt,yslope,yintcpt,corrYxy
 9994 format ('X SLOPE=',f9.3,' X INTERCEPT=',f9.3,' Y SLOPE=',
     >        f9.3,' Y INTERCEPT=',f9.3,' Young CORRELATION COEF=',
     >        f9.3,/)
c
      write (25,9995) xpassno,ypassno,corrDxy,corrYxy,xvar,yvar,
     >               covarxy,xstdev,ystdev
 9995 format (2i5,7(f10.4))
      write (25,9996) xamean,ybmean,xslope,yslope,xintcpt,yintcpt
 9996 format (10x,6(f10.4))
      return
      end
c
c
c
c
      subroutine average (nobs,choice,prime,crosscnt)
      real xadata(400,4),ybdata(400,4),avgdatmean,avgdatsum,
     >     avgdata(400,4),nobss
      integer nobs,choice,prime,crosscnt
      common /trunstat/ xadata(400,4),ybdata(400,4)
      common /aver/ avgdata(400,4)
c
c------------------------------ subroutine description
c       average calculates the average magnetic value of the
c       input passes.  it will also find the average position
c       of the input passes if so directed.
c
      avgdatsum=0.0
      nobss=real(nobs)
c
      if (prime .eq. 2) then
         do 100 i=1,nobs
            avgdata(i,1)=((xadata(i,1)+ybdata(i,1))/2.0)
            avgdata(i,2)=((xadata(i,3)+ybdata(i,3))/2.0)
            addxy = abs(xadata(i,3)) + abs(ybdata(i,3))
            if (addxy .gt. 270.0) then
               crosscnt=crosscnt+1
               if (ybdata(i,3).gt.0.0 .and. xadata(i,3).lt.0.0) then
                  xadata(i,3) = xadata(i,3) + 360.0
               elseif (ybdata(i,3).lt.0.0 .and. xadata(i,3).gt.0.0) then
```

```
              ybdata(i,3) = ybdata(i,3) + 360.0
            endif
            avgdata(i,2)=((xadata(i,3)+ybdata(i,3))/2.0)
            if (avgdata(i,2) .gt. 180.0)
     >           avgdata(i,2)=avgdata(i,2)-360.0
         endif
         avgdata(i,3)=((xadata(i,4)+ybdata(i,4))/2.0)
         avgdata(i,4)=((xadata(i,2)+ybdata(i,2))/2.0)
         avgdatsum=avgdatsum+avgdata(i,4)
100    continue
       avgdatmean=avgdatsum/nobss
       if (choice .eq. 1) then
          do 150 i=1,nobs
             avgdata(i,4)=avgdata(i,4)-avgdatmean
150       continue
       endif
c
     elseif (prime .eq. 1) then
        do 200 i=1,nobs
           avgdata(i,4)=((xadata(i,2)+ybdata(i,2))/2.0)
           avgdatsum=avgdatsum+avgdata(i,4)
200     continue
        avgdatmean=avgdatsum/nobss
        if (choice .eq. 1) then
           do 250 i=1,nobs
              avgdata(i,4)=avgdata(i,4)-avgdatmean
250        continue
        endif
     endif
c
     return
     end
c
c
c
     subroutine track (nop,noc,prime)
     integer nop,noc,prime
     real radfac,avgdata(400,4),x(400),y(400),xadata(400,4),
    >     ybdata(400,4)
     common /aver/ avgdata(400,4)
     common /tplot/ x(400),y(400)
     common /trunstat/ xadata(400,4),ybdata(400,4)
c
c------------------------------- subroutine description
c      track finds the lat and long coordinates of the observations
c      along an orbit.  these coordinates can be plotted as the
c      trace of the pass along the earth.
c      NOTE: lat and long are changed to radians for the plotting
c            package that i use.
c
     RADFAC=0.017453293
     noc=noc+1
     if (prime .eq. 2) then
        DO 200 J=1,NOP
           x(j)=90.0-avgdata(j,1)
           y(j)=avgdata(j,2)
           if(y(j).lt.0.) y(j)=y(j)+360.
           X(J)=X(J)*RADFAC
           Y(J)=Y(J)*RADFAC
200     CONTINUE
     elseif (prime .eq. 1) then
        do 300 j=1,nop
           x(j) = 90.0 - xadata(j,1)
           y(j) = xadata(j,3)
           if (y(j) .lt. 0.0) y(j) = y(j) + 360.0
           x(j) = x(j) * radfac
           y(j) = y(j) * radfac
300     continue
     endif
c
     return
     end
c
c
c
     subroutine findgap (global,dndk,minobs,type)
     integer zero,eight,strtotpass,x3row,x3col,x3pass,
    >        y3row,y3col,y3pass,cnt,type,strmincnt,
    >        dndk,nowant(4000),nocnt,mincnt,
    >        minobs,totpass,global,
    >        nopass,strnocnt,allcnt,strallcnt
     real y3data(400,3),y3mean,abss,allxdat(4000,4),
    >     allydat(4000,4),x3mean,x3data(400,3),stryone(4)
     common /nope/ nowant(4000),nocnt
c-------------------------------------- subroutine description
c
c      findgap locates the overlapping segment in each of the
c      two adjacent passes.  this is done by looking at the
c      first and last latitudes in each pass and comparing
c      the values.  if there are two passes that do not have
c      overlapping segments, then one of the two passes
```

```
c       is removed.
c       NOTE: the difference between a latitude gap finder and
c             a minimum observation gap finder is:  a latitude
c             finder allows extremely short passes to be worked
c             with in the processing whereas a minimum observation
c             gap finder removes all short passes.  the trade off
c             occurs because a minimum observation finder actually
c             allows a higher number of observations to be worked
c             with in collocation.  therefore a minobs gap finder
c             is usually best.  experiment to see what u like.
c
      totpass=0
      allcnt=0
      strnocnt=nocnt
c
 10   continue
      read (10,end=30) x3row,x3col,zero,x3mean,x3pass,eight
      do 15 i=1,x3row
         read (10) (x3data(i,ii),ii=1,x3col)
 15   continue
      totpass=totpass+1
      if (x3row .lt. minobs) then
         nowant(nocnt+1)=x3pass
         nocnt=nocnt+1
      endif
      allcnt=allcnt+1
      allxdat(allcnt,1)=real(x3pass)
      allxdat(allcnt,2)=real(x3row)
      allxdat(allcnt,3)=x3data(1,1)
      allxdat(allcnt,4)=x3data(x3row,1)
      go to 10
c
 30   continue
      strtotpass=totpass
      strallcnt=allcnt
      totpass=0
      allcnt=0
 31   read (12,end=50) y3row,y3col,zero,y3mean,y3pass,eight
      do 35 i=1,y3row
         read (12) (y3data(i,ii),ii=1,y3col)
 35   continue
      totpass=totpass+1
      if (y3row .lt. minobs) then
         nowant(nocnt+1)=y3pass
         nocnt=nocnt+1
      endif
      allcnt=allcnt+1
      allydat(allcnt,1)=real(y3pass)
      allydat(allcnt,2)=real(y3row)
      allydat(allcnt,3)=y3data(1,1)
      allydat(allcnt,4)=y3data(y3row,1)
      go to 31
c
 50   continue
      if (totpass.ne.strtotpass .or. allcnt.ne.strallcnt) then
         write (*,*) 'FILES DO NOT HAVE THE SAME NUMBER OF PASSES'
         write (*,*) 'FILE X PASS COUNT=',strtotpass,strallcnt
         write (*,*) 'FILE Y PASS COUNT=',totpass,allcnt
         stop
      endif
c
      do 52 ii=1,4
         stryone(ii)=allydat(1,ii)
 52   continue
      do 55 i=1,allcnt-1
         do 55 ii=1,4
            allydat(i,ii)=allydat(i+1,ii)
 55   continue
      do 57 ii=1,4
         allydat(allcnt,ii)=stryone(ii)
 57   continue
      if (global .eq. 1) allcnt=allcnt-1
c
      if (type .eq. 2) go to 400
c
c-------------------- work a latitude gap finder
 70   continue
      cnt=1
c
 100  continue
      if (allxdat(cnt,1) .ne. allydat(cnt,1)) then
         write (*,*) 'PASSES DO NOT MATCH FOR A-east A-west'
         write (*,*) 'REVERSE THE ORDER OF INPUT FILES AND RERUN'
         stop
      endif
      if (nocnt .eq. 0) go to 140
      do 110 i=1,nocnt
         if (int(allxdat(cnt,1)) .eq. nowant(i)) then
            if (cnt+1 .gt. allcnt) go to 190
            do 105 jj=cnt,allcnt-1
```

```
            do 105 j=1,4
               allxdat(jj,j)=allxdat(jj+1,j)
               allydat(jj,j)=allydat(jj+1,j)
 105        continue
            allcnt=allcnt-1
         endif
 110  continue
 140  continue
      abss=abs(allxdat(cnt,3)-allydat(cnt,3))
      if (abss .lt. 0.33) go to 190
      if (abss .ge. 0.33) then
         nopass=int(allxdat(cnt,1))
c-------------------------if this is a dusk pass then will count from
c                          -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (allxdat(cnt,3) .gt. allydat(cnt,3)) then
               if (allxdat(cnt,3) .gt. allydat(cnt,4)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            elseif (allxdat(cnt,3) .lt. allydat(cnt,3)) then
               if (allxdat(cnt,4) .lt. allydat(cnt,3)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            endif
c-------------------------if this is a dawn pass then will count from
c                          the equator toward the south pole
c                          that is decreasing independent variable
         elseif (dndk .eq. 1) then
            if (allxdat(cnt,3) .lt. allydat(cnt,3)) then
               if (allxdat(cnt,3) .lt. allydat(cnt,4)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            elseif (allxdat(cnt,3) .gt. allydat(cnt,3)) then
               if (allxdat(cnt,4) .gt. allydat(cnt,3)) then
                  nocnt=nocnt+1
                  nowant(nocnt)=nopass
               endif
            endif
         endif
      endif
c
 190  continue
      cnt=cnt+1
      if (cnt .gt. allcnt) go to 200
      go to 100
c
 200  continue
      if (nocnt .eq. strnocnt) go to 999
      if (strnocnt .lt. nocnt) then
         strnocnt=nocnt
         go to 70
      endif
c
c
c--------------------- work a minimum observations gap finder
 400  continue
      mincnt=0
      strmincnt=mincnt
c
 470  continue
      cnt=1
c
 500  continue
      if (allxdat(cnt,1) .ne. allydat(cnt,1)) then
         write (*,*) 'PASSES DO NOT MATCH FOR A-east A-west'
         write (*,*) 'REVERSE THE ORDER OF INPUT FILES AND RERUN'
         stop
      endif
      if (int(allxdat(cnt,2)) .lt. minobs  .or.
     >     int(allydat(cnt,2)) .lt. minobs) then
         if (cnt+1 .gt. allcnt) go to 590
         do 505 jj=cnt,allcnt-1
            do 505 j=1,4
               allxdat(jj,j)=allxdat(jj+1,j)
               allydat(jj,j)=allydat(jj+1,j)
 505     continue
         allcnt=allcnt-1
         go to 500
      endif
 510  continue
 540  continue
      abss=abs(allxdat(cnt,3)-allydat(cnt,3))
      if (abss .lt. 0.33) go to 590
      if (abss .ge. 0.33) then
         xrow=int(allxdat(cnt,2))
         yrow=int(allydat(cnt,2))
         minxyrow=min(xrow,yrow)
```

```
         mincnt2=mincnt
c------------------------if this is a dusk pass then will count from
c                          -90.0 lat degrees toward the equator
         if (dndk .eq. 0) then
            if (allxdat(cnt,3) .gt. allydat(cnt,3)) then
               if (allxdat(cnt,3) .gt. allydat(cnt,4)) mincnt=mincnt+1
            elseif (allxdat(cnt,3) .lt. allydat(cnt,3)) then
               if (allxdat(cnt,4) .lt. allydat(cnt,3)) mincnt=mincnt+1
            endif
c------------------------if this is a dawn pass then will count from
c                          the equator toward the south pole
c                          that is decreasing independent variable
         elseif (dndk .eq. 1) then
            if (allxdat(cnt,3) .lt. allydat(cnt,3)) then
               if (allxdat(cnt,3) .lt. allydat(cnt,4)) mincnt=mincnt+1
            elseif (allxdat(cnt,3) .gt. allydat(cnt,3)) then
               if (allxdat(cnt,4) .gt. allydat(cnt,3)) mincnt=mincnt+1
            endif
         endif
         if (mincnt .gt. mincnt2) minobs=minxyrow+1
      endif
c
c
 590  continue
      cnt=cnt+1
      if (cnt .gt. allcnt) go to 600
      go to 500
c
 600  continue
      if (mincnt .eq. strmincnt) go to 999
      if (strmincnt .lt. mincnt) then
         strmincnt=mincnt
         go to 470
      endif
c
 999  continue
      write (*,*) 'total passes read = ',totpass
      if (nocnt .gt. 0) then
         write (*,*) 'will remove the following passes from processing'
         do 1010 i=1,nocnt
            write (*,*) nowant(i),i
 1010    continue
      endif
      if (type .eq. 2) write (*,*) 'new minimum observation cutoff',
     >                   ' is =',minobs
      rewind (10)
      rewind (12)
c
      return
      end
```

00 1980. MAGSAT12/83 FIELD MODEL

2	1	-29986.5977	0	26.5047	0
2	2	-1956.0098	5603.8984	11.9441	-15.9892
3	1	-1996.6699	0	-17.7715	0
3	2	3027.3098	-2129.2998	3.5560	-14.9880
3	3	1662.6899	-199.7370	03.1459	-24.1879
4	1	1281.4099	0	01.2007	0
4	2	-2180.5198	-335.5408	-3.6188	2.2614
4	3	1250.8599	270.8979	-1.1017	2.5249
4	4	832.8589	-252.2290	1.8474	-4.1066
5	1	937.6270	0	-.6511	0
5	2	782.2798	212.1880	-2.0523	2.9464
5	3	397.8979	-256.6948	-5.8275	02.0422
5	4	-418.9170	52.8622	-2.4301	1.9796
5	5	198.8960	-297.1729	-2.7610	-2.4318
6	1	-217.8750	0	.0000	0
6	2	357.2400	45.9779	-0.0000	1.4379
6	3	261.0259	149.8220	-1.2862	0.4783
6	4	-74.2543	-150.6240	-4.1370	-0.9792
6	5	-161.9080	-77.7266	-1.3131	0.5830
6	6	-48.0216	92.1005	00.0000	.1310
7	1	47.9521	0	1.5744	0
7	2	65.5112	-14.7623	0.3746	00.4169
7	3	41.9219	93.0835	.9571	-0.7149
7	4	-192.1090	70.5787	1.2512	0.0000
7	5	3.5873	-43.0699	.6488	-0.4089
7	6	13.8212	-2.2228	.8309	.3552
7	7	-107.7240	17.2125	01.5749	1.5578
8	1	71.9145	0	1.3718	0
8	2	-59.2225	-82.4974	-0.7318	-0.8626
8	3	1.6950	-27.4258	.3371	.7545
8	4	20.7407	-4.9118	.0000	.3567
8	5	-12.4245	16.2844	0.7764	0.1974
8	6	.6345	17.7860	-0.0000	0.3800
8	7	10.5919	-23.0138	0.0000	0.4037
8	8	-1.6744	-9.6949	1.7731	0.0000
9	1	18.4829	0	0.7492	0
9	2	6.5464	6.8482	0	-0.6295
9	3	-.4431	-17.7074	0.3060	
9	4	-10.9846	4.2139		
9	5	-6.9316	-22.1996	-.8890	
9	6	4.2292	9.1278	.2806	.6764
9	7	2.7169	16.0818		-.8512
9	8	6.0254	-13.2103		
9	9	-1.3743	-14.5602	-2.2246	-1.7360
10	1	5.2888	0	-.2980	
10	2	10.3522	-20.8464		.4655
10	3	1.3838	15.5210	.4694	
10	4	-12.3472	8.7402		.8824
10	5	9.4401	-5.3002		.5640
10	6	-3.4208	-6.3179	-.0717	
10	7	-1.1873	9.0043		
10	8	6.7100	9.6459		
10	9	1.4932	-5.9579	-.4223	.3562
10	10	-4.9898	1.9550	-1.0560	2.0960
11	1	-3.4630	0	.8585	
11	2	-4.0062	1.2249		-.4876
11	3	2.2272	.5099	-.4740	
11	4	-5.4032	2.6612	-.2979	-.4653
11	5	-1.9878	5.7745	.7246	-.9477
11	6	4.5775	-4.2379		-.6276
11	7	3.1480	-.4134		
11	8	.9039	-1.3354	-.6724	
11	9	1.9700	3.5658	1.2551	
11	10	2.8069	-.4729	.1466	-.2740
11	11	-.2741	-6.1331	-.5215	
12	1	2.4795	0		
12	2	-1.1335	.6132		
12	3	-1.6582	1.7210		
12	4	2.1526	-1.3117		
12	5	.0610	-3.1316		
12	6	-.6072	.7970		
12	7	-.3238	-.0409		
12	8	1.6151	-2.4801		
12	9	1.7163	-.3808		
12	10	-.7606	-1.6667		
12	11	2.0084	-1.5290		
12	12	3.4451	0.7629		
13	1	-1.7069	0		
13	2	-.1409	.2182		
13	3	-.2764	.7604		
13	4	-.2045	2.5462		
13	5	.6790	-1.4956		
13	6	.6367	.3952		
13	7	-.4485	.2423		
13	8	-.1787	-.2355		
13	9	.2656	.0192		
13	10	-.4602	-.0133		
13	11	0.1407	-1.2097		
13	12	.5953	.3708		

13	13	-.1378	.4564			
14	1	-.2049	0			
14	2	-.5967	-.4816			
14	3	.3725	.3039			
14	4	-.8791	1.4826			
14	5	-.1829	-.2216			
14	6	1.1048	-.4758			
14	7	-.4404	-.2088			
14	8	.3847	.7995			
14	9	-.4823	.0535			
14	10	.1867	.8530			
14	11	-.1106	.0633			
14	12	.3455	-.1350			
14	13	-.0851	0.2500			
14	14	.3797	-.3348			
-1	0	0.	0.	0.	0.	0.

APPENDIX C: MAP PROCESSING

PROGRAMS

collocation.f

fourier2d.f

avgdifres.f

sqrmap.f

inversion.f

DATA FILE

rmagcov

```
      program collocation
c  ******************************************************
c  THIS PROGRAM READS A FILE OF IRREGULARLY DISTRIBUTED
c  DATA POINTS (LATITUDE,LONGITUDE,ELEVATION,ANOMALY
c  VALUE) AND PREDICTS THE GRIDDED ANOMALIES ON A GRID OF
c  SPECIFIED DENSITY. THE ANOMALY ESTIMATE FOR EACH GRID
c  LOCATION IS OBTAINED FROM NCPP NEAREST SAMPLE POINTS
c  USING A LOCAL COVARIANCE MODEL. THE PROCEDURE KNOWN AS
c  LEAST-SQUARES COLLOCATION INVOLES THE FOLLOWING STEPS:
c
c  1. REMOVE THE MEAN OF THE ANOMALIES IN THE PREDICTION
c     AREA TO OBTAIN RESIDUALS CENTERED AROUND MEAN
c  2. SEARCH FOR THE NCPP CLOSEST DATA POINTS TO THE GRID
c     LOCATION AND STORE THEM IN VECTOR (M1)
c  3. FORM THE COVARIANCE MATRIX (COVM) OF THE NCPP DATA
c     POINTS
c  4. ADD THE ERROR VARIANCE OF THE DATA POINTS TO THE
c     DIAGONAL OF 'COVM' MATRIX, TO FORM THE FINAL 'COVM'
c     MATRIX
c  5. INVERT 'COVM' MATRIX AND STORE THE RESULT AGAIN IN
c     'COVM'
c  6. FORM THE CROSS-COVARIANCE VECTOR (T1) BETWEEN THE
c     GRID VALUE TO BE PREDICTED AND THE NCPP DATA POINTS
c  7. BY LEAST SQUARES COLLOCATION, THE ANOMALY ESTIMATE
c     IS GIVEN AS:
c
c      TP=T1*COVM*M1
c
c     AND THE STANDARD ERROR OF PREDICTION IS GIVEN BY,
c
c      SEP=DSQRT(VAR-T1*COVM*T1)
c
c     VAR...COVARIANCE AT ZERO SEPARATION (I.E. VARIANCE)
c
c
c
c     THE ABOVE EQUATIONS INVOLVE THE MATRIX OPERATIONS.
c
c  ******************************************************
c
c   PRELIMINARY SOFTWARE EXPLICITLY DEVELOPED FOR GRAVITY
c   PREDICTION OVER A SPHERICAL SURFACE WAS MADE AVAILABLE
c   BY GEODETIC SCIENCE DEPARTMENT AT THE OHIO STATE
c   UNIVERSITY. IT WAS MODIFIED FOR NASA MAGNETIC SATELLITE
c   APPLICATION FOR 3-D PREDICTION AND THEREBY ALTITUDE
c   NORMALIZATION.
c
c   MODIFICATIONS BY:  HARISH K. GOYAL
c                      DEPT. OF GEOL & MIN, OSU
c                      TEL. 422-1434, CAMPUS
c                      MAR, 1986
c
c     1. LATITUDES ARE CHANGED TO CO-LATITUDES TO COMPLY
c        WITH SPHERICAL COORDINATES.
c     2. SEPARATION DISTANCES ARE THE RADIAL VECTORS TO
c        ACCOUNT FOR THREE DIMENSIONAL VARIABILITY
c     3. COVARIANCES ARE AUTOMATICALLY SCALED IN THE
c        PROGRAM
c
c     further modifications 11 may 90
c     these modifications are all lower case letters
c
c     well'p just a few more modifications on 8 sep 90
c     these changes are:  1) removal of unnecessary arrays
c     2) changing all arrays to real*4 and not real*8
c     3) keeping all arrays that work with the inversion as
c     real*8.  4) removing every blasted 'nbug' statement i could
c     get my hands on!!  5) reading the data once storing everything in
c     memory -ie. not reading the data twice.  6) changing all logical
c     true and false statements to user friendly statements.
c     7) this program could be faster by inverting only the half of
c     the symmetric covariance matrix (covm) and there probably is a
c     faster routine for searching for the closest points.
c
c  ******************************************************
c
c   INPUT PARAMETERS .....................................
c
c   NORTH...NORTH LATITUDE OF DATA AREA
c   SOUTH...SOUTH LATITUDE OF DATA AREA
c   WEST...WEST  LONGITUDE OF DATA AREA
c   EAST...EAST  LONGITUDE OF DATA AREA
c   note: use the following example if you want an equal-
c         area projection.  say you are working at the south
c         pole from -40S to -83S and including all longitudes.
c         then instead of choosing north=-40, south=-83,
c         west=-180 and east=180 (which are appropriate for
c         non-equal area degrees)  choose instead:
c         north=55, south=-55, west=-55, east=55.  55 comes
c         from the following calculation.  from -40 to
```

```
c         -83 degrees is 43 points, however, an equal area
c         projection will go all the way to the pole because
c         the pole will be centered at the middle of the grid
c         (unlike the non-equal area degree projection where
c         the pole is at the southern edge of the grid) so,
c         from -40 to -90 is 50 points and you should add a few
c         points for a rind around the edge, say 5 points.
c         55=50+5.  get it?  now, if you are going to work with
c         an equal area projection, you MUST transform the
c         coordinates of the input data points from the degree
c         domain to the spatial domain.  program getllraspc.f does
c         this transformation.  also, if you want to convert a
c         grid from/to equal-area to/from non-equal degree then
c         use program deg2spc.f to do the coordinate transform
c         and use collocation to get the values at the new grid
c         coordinates.  clear as mud, eh?  email me at
c               alsdorf@geols.mps.ohio-state.edu
c         if you need help.
c
C   NX...NO. OF GRID PTS IN THE LONG. DIRECTION, MINUS ONE
C   NY...NO. OF GRID PTS IN THE LAT. DIRECTION, MINUS ONE
C   NCOV...NUMBER OF ENTRIES IN COVARIANCE FUNCTION
c          and is determined by the program
C   NCPP...NO. OF NEAREST POINTS USED FOR PREDICTION
C   ELEV...COMMON ALTITUDE FOR GRIDDED ANOMALY DATA
c          and is in kilometers from --SURFACE-- of the earth
C   ESTD...STANDARD DEVIATION OF OBSERVATIONAL ERROR
C          (ERROR VARIANCE=ESTD**2)
c   from read of file 10
C              THI...LATITUDE COORDINATE
C              PHI...LONGITUDINAL COORDINATE
C              RAD...RADIUS VECTOR FROM --CENTER-- OF THE EARTH
C              ANO...ANOMALY VALUE
C
C
C
C  GRID DIMENSIONS AS FOLLOWS:
c  the following arrays = (ny+1)*(nx+1):
c        x,y,tp,ifc,ih,sep
c  the following arrays must be equal to or greater than
c  the maximum number of data points:
c        rrad,rthi,rphi,tano,dist,thii,phii,radd,anom
C  ARRAYS CNN,DNUM MUST ACCOMMODATE THE NUMBER OF ENTRIES
C  IN COVARIANCE FUNCTION
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      IMPLICIT REAL(A-H,O-Z)
      DIMENSION X(20000),Y(20000),TP(20000),CNN(500),
     .     DNUM(500),RRAD(200000),RTHI(200000),RPHI(200000),
     .     TANO(200000),
     .     DIST(200000),IFC(20000),thii(200000),phii(200000),
     >     radd(200000),anom(200000),IH(20000),SEP(20000)
      REAL    NORTH
      real thi,phi,rad,anom,cross
      integer totpts
      character*80 filename
      character*5 yesno
      real dnum,cnn,scale,sumsqr
      COMMON/ONE/ CNN,DNUM,NCOV
      COMMON/TWO/ SCALE
      DATA RHO,NPTS/57.2957795,0/
c
      write (*,*) 'INPUT COVARIANCE MATRIX'
      read (*,9990) filename
 9990 format (a80)
      open (1, file=filename,status='old',form='formatted')
      write (*,*) 'INPUT FILE OF ALL DATA POINTS LAT LON RAD ANOM'
      read (*,9990) filename
      open (10, file=filename,status='old',form='formatted')
      write (*,*) 'OUTPUT FILE OF GRIDDED DATA POINTS'
      read (*,9990) filename
      open (20, file=filename,form='formatted')
      write (*,*) 'OUTPUT INFORMATION FILE'
      read (*,9990) filename
      open (21, file=filename,form='formatted')
c
c
      write (*,*) 'BARF-- NORTH SOUTH EAST WEST AS 90.0 TO -90.0',
     >           ' AND 180.0 TO -180.0'
      read (*,*) north,south,east,west
      write (*,*) 'NUMBER OF GRID POINTS MINUS ONE IN THE NS DIRECTION'
      write (*,*) 'NUMBER ------------------------------ EW ---------'
      write (*,*) 'NS, EW '
      read (*,*) ny,nx
      write (*,*) 'POINT SIZE OF WINDOW FOR SEARCH AREA (20)'
      read (*,*) ncpp
      write (*,*) 'ELEVATION OF PREDICTION FOR GRID (350.0 Km)'
      write (*,*) '(--NOT-- radius = 6378.140 Km)'
      read (*,*) elev
      elev=elev+6378.140
```

```
      write (*,*) 'STANDARD DEVIATION OF OBSERVATIONAL ERROR (1.0)'
      read (*,*) estd
      write (*,*) 'REMOVE THE MEAN FROM THE GRID BEFORE WRITTING'
      write (*,*) 'y OR n'
      read (*,9991) yesno
 9991 format (a5)
c
      If(EAST.EQ.WEST.OR.NORTH.LE.SOUTH.OR.NY.LE.0.OR.NX.
     .     LE.0) STOP 9999
C
C     CHANGE LATITUDES TO SPHERICAL COORDINATES
C
      NORTH=90.0-NORTH
      SOUTH=90.0-SOUTH
C
c-------------- the program is checking for the -180.0 180.0 meridian
C
      cross=0.0
      IF(WEST.GT.EAST) cross=360.0
      EAST=EAST + cross
C
C     INPUT THE COVARIANCE TABLE
C
      I=1
 5    read (1,*,end=7) dnum(i),cnn(i)
      i=i+1
      go to 5
 7    ncov=i-1
      SCALE=CNN(1)
C
C  GRID SPACING ..ie: interval between grid nodes = dl and dp
C
      DP=(NORTH-SOUTH)/FLOAT(NY)
      DL=( EAST-WEST )/FLOAT(NX)
C
C ..  DETERMINE THE OVERLAPS IN  X AND Y-DIRECTIONS
C
      YOVLAP=DP/2.
      XOVLAP=DL/2.
c
C ..  NXP1,NYP1 = NUMBER OF SORT ELEMENTS IN X AND Y-DIR.
C ..  XLL,XUP = LOWER AND UPPER X LIMITS OF SORT RANGE
C ..  YLL,YUP = LOWER AND UPPER Y LIMITS OF THE SORT RANGE
C ..  NX,NY = NUMBER OF DIVISIONS ALONG X AND Y AXES
C  DETERMINE X,Y COORDINATES OF GRID INTERSECTIONS
C
      NXP1=NX + 1
      NYP1=NY + 1
      DO 10 I=1,NYP1
         YY=DP*(I-1)+YOVLAP
         DO 10 J=1,NXP1
            K=J+NXP1*(I-1)
            Y(K)=YY
            X(K)=DL*(J-1)+XOVLAP
 10   CONTINUE
C
C     DETERMINE THE  X,Y  COORDINATES OF THE DATA AREA AND
C     GRID SPACING
C
      XLL=0.
      XUP=X(NXP1)+XOVLAP
      YLL=0.
      YUP=Y(NXP1*NYP1)+YOVLAP
      DXX=NXP1/(XUP-XLL)
      DYY=NYP1/(YUP-YLL)
C
C  DETERMINE BOUNDARIES FOR DATA SELECTION
C
      THIS=SOUTH-YOVLAP
      THIN=NORTH+YOVLAP
      EPHI=EAST+XOVLAP
      WPHI=WEST-XOVLAP
c------------------------ varn is a constant for input to subroutine
c                         prdt.  varn should be change to an array for
c                         corresponding individual data points if each
c                         data point or group of data points need to have
c                         individually different error variances.
      varn=estd**2
c
      write (21,*) 'north colatitude =',north,' south colat =',south
      write (21,*) 'east longitude =',east,' west long =',west
      write (21,*) 'dp =',dp,' dl =',dl,' xovlap =',xovlap,
     >           ' yovlap =',yovlap
      write (21,*) 'thin =',thin,' this =',this,' ephi =',ephi,
     >           ' wphi =',wphi
      write (21,*) 'xll =',xll,' xup =',xup,' yll =',yll,' yup =',yup
      write (21,*) 'dxx =',dxx,' dyy =',dyy
      write (21,*) 'error variance =',varn,
     >                    ' standard deviation error =',estd
c
```

```
C  INPUT ADJUSTED MAGNETIC DATA AND SELECT DATA FOR
C  THE PREDICTION
C
      np=(nxp1+1)*(nyp1+1)
      do 15 i=1,np
         ifc(i)=0
 15   continue
c
      AMEAN=0.0
      totpts=0
 20   read (10,*,end=30) thi,phi,rad,anomaly
      totpts=totpts+1
      THI=90.0-THI
      IF(THI.GE.THIS) GO TO 20
      IF(THI.LE.THIN) GO TO 20
      IF(PHI .LT. 0.0) PHI=PHI + cross
      IF(PHI.LE.WPHI) GO TO 20
      IF(PHI.GE.EPHI) GO TO 20
      NPTS=NPTS + 1
      AMEAN=AMEAN + anomaly
      RX=PHI-WPHI
      RY=THI-THIS
C
C   IX AND IY IDENTIFIES THE BLOCK TO WHICH DATA FALL INTO
C   AND IYJX ASSIGNS AN IDENTIFIER TO DATA CORRESPONDING TO
C   THAT BLOCK
C
      IY=INT((RY-YLL)*DYY)+1
      JX=INT((RX-XLL)*DXX)+1
      IYJX=(IY-1)*NXP1+JX
C
C ..  IFC = COUNTER VECTOR , STORES NUMBER OF DATA PER
C             SORT ELEMENT
C
      IFC(IYJX)=IFC(IYJX)+1
c
      thii(npts)=thi
      phii(npts)=phi
      radd(npts)=rad
      anom(npts)=anomaly
c
      GO TO 20
c
 30   continue
      if (npts .le. 1) stop 111
C
      AMEAN=AMEAN/FLOAT(NPTS)
c
      WRITE(21,*) 'total points selected =',npts
      write (21,*) 'total points read =',totpts
      write (21,*) 'mean of selected points =',amean
      write (*,*) 'finished reading data set'
c
C     IH = POINTER VECTOR, FOR CORRESPONDING BLOCK ATTAINS
C     A VALUE EQUAL TO SUM OF DATA IN PREVIOUS BLOCKS + 1
C
      ND=NXP1*NYP1
      IH(1)=1
      DO 87 I=2,ND
         I1=I-1
         IH(I)=IFC(I1)+IH(I1)
 87   CONTINUE
C
C  TX,TY,RTHI,RPHI,RRAD,TANO,VARN ARE NUMBERED FOR
C  CORRESPONDING DATA, IN EACH BLOCK NUMBERING STARTS
C  WITH IH VALUE FOR THAT BLOCK AND INCREMENTED BY 1 FOR
C  NEXT DATA IN THE BLOCK
c
c  the mean anomaly value is removed here rather than in subroutine prdt.
c  also the sum of squares is calculated here and transfered to
c  subroutine prdt.
C
      sumsqr=0.0
      DO 85 I=1,NPTS
         rx=phii(i)-wphi
         ry=thii(i)-this
         IY=INT((RY-YLL)*DYY)+1
         JX=INT((RX-XLL)*DXX)+1
         IYJX=(IY-1)*NXP1+JX
         NUM=IH(IYJX)
         TANO(NUM)=anom(i)-amean
         sumsqr=sumsqr+(dble(tano(num))*dble(tano(num)))
         RTHI(NUM)=thii(i)
         RPHI(NUM)=phii(i)
         RRAD(NUM)=radd(i)
         IH(IYJX)=IH(IYJX)+1
 85   CONTINUE
C
C  IH(I) ATTAINS THE VALUE EQUAL TO NUMBER OF SAMPLE
C  POINTS IN PREVIOUS BLOCKS + 1
```

```
c
      DO 86 I=1,ND
         IH(I)=IH(I)-IFC(I)
 86   CONTINUE
      IH(ND+1)=NPTS+1
C
C  SUBROUTINE PRDT PREDICTS ANOMALIES AND ERRORS OF
C  STANDARD DEVIATION AT EACH GRID LOCATION
C
c
      write (*,*) 'calculating anomaly values'
c
      CALL PRDT(NPTS,NYP1,NXP1,X,Y,TP,RTHI,RPHI,DIST,
     .     VARN,THIS,TANO,NORTH,SOUTH,EAST,WEST,IH,DXX,DYY,DP,
     .     DL,AMEAN,RRAD,NCPP,ELEV,SEP,sumsqr)
c
C
C  WRITE THE VALUES OF THE PREDICTED Z-AXIS VALUES
C  (ANOMALIES) AND THEIR ERROR OF STANDARD DEVIATIONS.
C  THE ROWS ARE LISTED WITH LATITUDES STARTING SOUTH
C  AND INCREMENTING TO NORTH.
c
      if (yesno .eq. 'y') then
         np=nxp1*nyp1
         do 380 i=1,np
            totgrid=tp(i)+totgrid
 380     continue
         avgrid=totgrid/real(np)
         do 390 i=1,np
            tp(i)=tp(i)-avgrid
 390     continue
         write (21,*) 'total mean removed from the grid =',avgrid
      endif
c
      WRITE(21,9600) WEST,DL
      write (20,*) nxp1
      write (20,*) nyp1
      write (20,*) south
      write (20,*) west
      write (20,*) dl
      P=SOUTH
      DO 400 I=1,NYP1
         ID1=(I-1)*NXP1+1
         ID2=ID1+NXP1-1
         WRITE(21,9601) P,(TP(J),J=ID1,ID2)
         WRITE(20,'(6F13.5)') (TP(J),J=ID1,ID2)
 400  P=P + DP
C
      WRITE(21,9602) WEST,DL
      P=SOUTH
      DO 420 I=1,NYP1
         ID1=(I-1)*NXP1+1
         ID2=ID1+NXP1-1
         WRITE(21,'(6F13.5)') (SEP(J),J=ID1,ID2)
 420  P=P + DP
C
 9600 FORMAT(/,'PREDICTED Z-AXIS VALUES',/,
     .       'STARTING  LONGITUDE=',F9.3,'; INCREMENT=',F5.2,/)
 9601 FORMAT('LAT=',F8.2,2X,80(8F8.2,/,14X))
 9602 FORMAT(/,'PREDICTED STANDARD DEVIATIONS',/,
     .       'STARTING LONGITUDE=',F9.3,'; INCREMENT=',F5.2,/)
c
      STOP
      END
c
c
c
c
      SUBROUTINE PRDT(NPTP,MY,MX,X,Y,TP,LAT,LONG,DIST,
     .     NSEe,THIS,M1,THI1,THI2,PHI1,PHI2,IH,DXX,DYY,DP,DL,
     .     AMEAN,RC,NCPP,ELEV,SEP,sumsqr)
      IMPLICIT REAL (A-H,O-Z)
      REAL NSEe,LAT,LONG,M1
      real scale,covm,fact,sumsqr,var,nse,dummy,b,
     >       t1,t2,tem,tem2,cov
      DIMENSION LAT(1),LONG(1),DIST(1),M1(1),RC(1),
     >          X(1),Y(1),TP(1),IH(1),SEP(1)
      DIMENSION COVM(110,110),B(110),T1(110),T2(110),LCC(110)
      COMMON/TWO/ SCALE
      DATA DMAX/6E6/
      EQUIVALENCE (B(1),T1(1))
c------------------------------ subroutine description
c       this subroutine is one big do loop that progresses through
c       the grid nodes to determine the magnitude at each node.
c
      nse=dble(nsee)
      CON1=57.2957795
      ND=MY*MX
      NBAD=0
      YLL=0.0
```

```
      XLL=0.0
C
C     CALCULATE THE SCALING FACTOR FOR THE COVARIANCE TABLE
C
      fact = (sumsqr/dble(nptp))/scale
c
      CALL COVINT(0.0,FACT,VAR)
c
      write (21,*)'sum of squares =',sumsqr,' scaling factor =',FACT
      write (21,*)'zero separation variance =',var
c
      NCPM1=NCPP-1
      IND=0
C
C   P AND Q ARE THE LATITUDE AND LONGITUDE OF THE
C   PREDICTED POINT
C
      write (21,*) ' '
      write (21,*) 'the following data indicate areas where the'
      write (21,*) 'prediction was bad'
      write (21,*) 'lat  lon  x grid node  y gridnode  id1  id2  ',
     >             'bad point number of total points'
c
      DO 10 IPT=1,MY
         IM=IPT-1
         P=THI2+IM*DP
         DO 10 JPT=1,MX
            JM=JPT-1
            Q=PHI2+JM*DL
            IND=IND+1
            XPP=X(IND)
            YPP=Y(IND)
            DO 7 I=1,NCPP
 7             LCC(I)=0
C
C  DISTANCE TO ALL POINTS IN THE WINDOW FROM PREDICTION
C  POINT
C
            IY=INT((YPP-YLL)*DYY)+1
            JX=INT((XPP-XLL)*DXX)+1
            IYJX=(IY-1)*MX+JX
            ID1=IH(IYJX)
            ID2=IH(IYJX+1)
            IF((ID2-ID1).GE.NCPP) GO TO 100
C
C     NOT ENOUGH DATA IN FIRST WINDOW,SO CONSIDER NEXT
C     WINDOW
C
            IY1=IY
            IY2=IY
            DO 17 IC=1,MY
               NDATA=0
               IY1=IY-IC
               IY2=IY+IC
               IF(IY1.LT.1) IY1=1
               IF(IY2.GT.MY) IY2=MY
               JX1=JX-IC
               JX2=JX+IC
               IF(JX1.LT.1) JX1=1
               IF(JX2.GT.MX) JX2=MX
               DO 18 IL=IY1,IY2
                  IYJX1=(IL-1)*MX+JX1
                  ID1=IH(IYJX1)
                  IYJX2=IYJX1+JX2-JX1+1
                  ID2=IH(IYJX2)
                  NDATA=NDATA+ID2-ID1
 18            CONTINUE
               IF(NDATA.GE.NCPP) GO TO 100
               NBAD=NBAD+1
c-------------------------------------- write the bad point to file
c
               WRITE(21,800) P,Q,XPP,YPP,ID1,ID2,NBAD,ND
 800           FORMAT(4(1x,F12.5),4I6)
 17         CONTINUE
 100        CONTINUE
            DO 211 IC=IY1,IY2
               IF(IY1.EQ.IY2) GO TO 106
               IYJX1=(IC-1)*MX+JX1
               ID1=IH(IYJX1)
               IYJX2=IYJX1+JX2-JX1+1
               ID2=IH(IYJX2)-1
               IF(ID2.LT.ID1) GO TO 211
 106           DO 210 I=ID1,ID2
                  DELC=COS(LAT(I)/CON1)*COS(P/CON1)+SIN(LAT(I)/CON1)*
     >                 SIN(P/CON1)*COS((LONG(I)-Q)/CON1)
                  PART1=ELEV**2 + RC(I)**2
                  PART2=2.0*ELEV*RC(I)
                  DARGU=PART1-PART2*DELC
                  IF(DARGU.LE.0.0) THEN
                     DIST(I)=0.0
```

```
                  ELSE
                     DIST(I)=SQRT(DARGU)
                  END IF
 210           CONTINUE
 211        CONTINUE
C
C SEARCH FOR NCPP NEAREST POINTS TO PREDICTION POINT
C
 220        DO 260 J=1,NCPP
               IF(LCC(J).GT.0) GO TO 260
               DMIN=DMAX
               DO 253 IC=IY1,IY2
                  IF(IY1.EQ.IY2) GO TO 107
                  IYJX1=(IC-1)*MX+JX1
                  ID1=IH(IYJX1)
                  IYJX2=IYJX1+JX2-JX1+1
                  ID2=IH(IYJX2)-1
                  IF(ID2.LT.ID1) GO TO 253
 107              DO 250 I=ID1,ID2
                     IF(DIST(I)-DMIN) 230,250,250
 230                 IF(DIST(I)) 250,240,240
 240                 DMIN=DIST(I)
                     LMIN=I
 250              CONTINUE
 253           CONTINUE
               DIST(LMIN)=-DMIN
               LCC(J)=LMIN
 260        CONTINUE
            IF(LCC(NCPP).NE.0) GO TO 258
            GO TO 220
 258        CONTINUE
C
C  FORM COVARIANCE MATRIX
C
            DO 280 I=1,NCPM1
               M=LCC(I)
               COVM(I,I)=dble(VAR+NSE)
               K=I+1
               DO 280 J=K,NCPP
                  N=LCC(J)
                  DELC=COS(LAT(M)/CON1)*COS(LAT(N)/CON1)+SIN(LAT(M)/
     .                   CON1)*SIN(LAT(N)/CON1)*COS((LONG(M)
     >                   -LONG(N))/CON1)
                  PART1=RC(M)**2 + RC(N)**2
                  PART2=2.0*RC(M)*RC(N)
                  DARGU=PART1-PART2*DELC
                  IF(DARGU.LE.0.0) THEN
                     DIS=0.0
                  ELSE
                     DIS=SQRT(DARGU)
                  END IF
c
                  CALL COVINT(DIS,FACT,COV)
c
                  COVM(J,I)=dble(COV)
                  COVM(I,J)=dble(COV)
 280        CONTINUE
            COVM(NCPP,NCPP)=dble(VAR+NSE)
c---------------------------------------- covm array dimensioned at
c                                          (ncpp,ncpp) is the input matrix
C
C  INVERT COVARIANCE MATRIX
C
            COVM(1,1)=1.0/COVM(1,1)
            DO 340  I=1,NCPM1
               L=I+1
               DO 300 J=1,I
 300              B(J)=0.0
               DO 310 J=1,I
                  DO 310 K=1,I
 310                 B(J)=B(J)-COVM(K,J)*COVM(L,K)
               Dummy=COVM(L,L)
               DO 320 J=1,I
 320              Dummy=B(J)*COVM(L,J)+Dummy
               Dummy=1.0/Dummy
               DO 330 J=1,I
                  COVM(J,L)=B(J)*Dummy
                  COVM(L,J)=B(J)*Dummy
                  DO 330 K=1,I
 330                 COVM(J,K)=COVM(J,K)+B(K)*B(J)*Dummy
 340        COVM(L,L)=Dummy
C
C---------------------------------------- covm array dimensioned at
c                                          (ncpp,ncpp) is now inverted
C
            DO 410 I=1,NCPP
               N=LCC(I)
               DELC=COS(LAT(N)/CON1)*COS(P/CON1)+SIN(LAT(N)/CON1)*
     >                SIN(P/CON1)*COS((LONG(N)-Q)/CON1)
               PART1=RC(N)**2 + ELEV**2
```

```
                PART2=2.0*ELEV*RC(N)
                DARGU=PART1-PART2*DELC
                IF(DARGU.LE.0.0) THEN
                   DIS=0.0
                ELSE
                   DIS=SQRT(DARGU)
                END IF
c
                CALL COVINT(DIS,FACT,COV)
c
 410         T1(I)=dble(COV)
c
             DO 430 I=1,NCPP
                TEM=0.0
                DO 420 J=1,NCPP
 420               TEM=COVM(J,I)*T1(J)+TEM
 430         T2(I)=TEM
c
             TEM=0.0
             TEM2=0.0
             DO 440 I=1,NCPP
                M=LCC(I)
                TEM=dble(M1(M))*T2(I)+TEM
                TEM2=T1(I)*T2(I)+TEM2
 440         CONTINUE
C
C  COMPUTE ANOMALY ESTIMATE  AND STANDARD ERROR OF PREDICTION
C
             TP(IND)=real(TEM)+AMEAN
             IF(VAR.LT.TEM2) GO TO 444
             SEP(IND)=SQRT(VAR-TEM2)
             GO TO 445
 444         SEP(IND)=1.5
             WRITE(21,446) VAR,TEM2
 446         FORMAT('VARIANCE =',F7.2,'TEM2 =',F7.2,'SEP',
     .             ' TAKEN AS 1.5 NT')
 445         CONTINUE
 10    CONTINUE
c
       RETURN
       END
c
c
c
       SUBROUTINE COVINT(DIS,FACT,COV)
       IMPLICIT REAL (A-H,O-Z)
       real dis
       integer ncov
       DIMENSION CNN(500),DNUM(500)
       COMMON/ONE/ CNN,DNUM,NCOV
c
c-------------------------------------- INTERPOLATION OF COVARIANCES
       TCOV=dble(NCOV-1)
       R1=dble(DIS)
       IF(R1.LT.TCOV)GO TO 1
       COV=CNN(NCOV)*FACT
       WRITE(21,100) R1
       RETURN
c
     1 IF(R1.GT.0.0001) GO TO 6
       COV=CNN(1)*FACT
       RETURN
c
     6 DO 2 I=1,NCOV
          IF(R1.LT.DNUM(I)) GO TO 3
     2 CONTINUE
     3 I=I-1
       FPINT=R1-DNUM(I)
       FDINT=DNUM(I+1)-DNUM(I)
       COV=CNN(I)+(CNN(I+1)-CNN(I))*FPINT/FDINT
       COV=COV*FACT
       RETURN
c
   100 FORMAT('SEPARATION =',F10.3,' >500 KM  COVARIANCE VALUE FOR',
     >        ' 500 KM USED.')
c
       END
```

PAGE C-9 INTENTIONALLY BLANK

```
      program fourier2d
      character*80 filename
      real xdata(361,361),ydata(361,361),xmean,ymean,
     >      prcnt,delta,cuthi,cutlo,xlag,mincc,maxcc,short,long,
     >      xcolat,ycolat,xlong,ylong,xgridspc,ygridspc,
     >      minccin,maxccin,cxlag
      integer xpassno,ypassno,file,xcol,xrow,ycol,yrow,
     >          lhb,cc,npass,imean,head,spass,swind,
     >          nwind,numfile,nxout,nyout,cnwind,udnwind,
     >          numcc(10),numlhb(10),numsd(10),numud(10),numrtp(10),
     >          numderiv(10)
      complex xcdata(512,512),ycdata(512,512)
      common /rowcol/ xrow,xcol,yrow,ycol
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      common /lhbflt/ delta,cuthi,cutlo,xlag,npass,nwind
      common /ccflt/ mincc,maxcc,minccin,maxccin,cnwind,cxlag
      common /reals/ xdata,ydata
      common /striking/ ang1,ang2,spass,swind,slag
      common /udcont/ uddelta,zcon,udxlag,udnwind
      common /xyzderiv/ xyzdelta,nth,nway
      common /rtp/ azm,xinc,dec
      common /comps/ xcdata,ycdata
c
c-----------------------------program description
c
c        fourier2d is an all encompassing fourier analysis program!
c        subroutines include the fft for forward and inverse situations,
c        a bandpass filter which can be adjusted to perform low, high and
c        bandpass filtering of wave numbers, a correlation coefficient
c        filter which zeros out wavenumbers according to correlation
c        coefficients, a strike-dip filter to remove wavelengths in
c        degrees of direction, an upward-downward continuation filter,
c        a derivative filter for any of the three directions and a
c        reduction-to-pole filter for magnetic total field intensity data.
c
c        NOTE: the only data variables absolutely necessary as INPUT are
c              the number of rows and number of columns, the remaining
c              variables; zero,mean,pass-number and eight, are not needed.
c              but, mean can be an OUTPUT if desired.
c        NOTE: fourier2d is for two-dimensional data.  if you have a
c              one-dimensional data set then use fourier1d.  however,
c              in 1-d i have not yet implemented the continuation,
c              derivative or rtp filters.
c
c     program date: 10 jul 92
c        this code was an extensive modification of an earlier code
c        named fourmat.
c
c
c       NOTE:  because there are 6 filters, there are about forty
c              zillion different combinations of filtering the fft'd
c              data.  so, to accomadate all of these, the user must
c              first state how many times to run the filter (ilhb,
c              icc, etc) then must state the order where to run each
c              filter (arrays numlhb, numcc, num etc.. hold these
c              user defined positions).  note that with this scheme
c              any filter can be run multiple times.  be sure to enter
c              only one filter for each position value.  the
c              following example should clear things up:
c              say you want to first bandpass filter, then cc
c              filter and then bandpass filter again.  then you
c              should set ilhb=2 and icc=1.  then array numlhb should
c              have the values 1 and 3 (for the first and third positions)
c              and array numcc should only have the value 2 (for the
c              second position.
c
      write (*,*) 'OUTPUT FILE OF STATISTICS AND INFORMATION'
      read (*,9990) filename
      open (25, file=filename,form='formatted')
c
      write (*,9989)
 9989 format ('1 IF YOU HAVE ONLY ONE FILE TO BE FOURIERED'/,
     >        '2 IF YOU HAVE TWO FILES TO BE COMPARED')
      read (*,*) numfile
c
      write (*,9988)
 9988 format ('ENTER THE MAXIMUM NUMBER OF TIMES TO RUN EACH FILTER'/,
     >        'ENTER 0 TO NOT RUN THE FILTER'/,
     > 'BANDPASS, CORR.COEFF., STRIKE-DIP, UP/DN CONT., RTP, DERIV')
      read (*,*) ilhb,icc,isd,iud,irtp,ideriv
c
      if (ilhb .gt. 0) then
         write (*,*) 'BANDPASS SECTION:'
         write (*,*) 'enter placement values in order ie. 2 3 5 '
         write (*,*) 'do not repeat these values elsewhere'
         write (*,*) '1=first, 3=third, 5=fifth etc...'
         read (*,*) (numlhb(i),i=1,ilhb)
         write (*,9993)
9993  format ('DELTA.....GRID INTERVAL IN MAP UNITS (1.0 degrees)'/
     >        'SHORT.....SHORTEST WAVELENGTH TO BE PASSED'/
```

```
     >         '              MUST BE AT LEAST 2*DELTA (2.0 degrees)'/
     >         'LONG......LONGEST WAVELENGTH TO BE PASSED'/
     >         '              MUST BE LARGER THAN SHORT'/
     >         'NPASS.....-1 TO REJECT WAVELENGTHS BETWEEN SHORT'/
     >         '               AND LONG'/
     >         '              1 TO PASS WAVELENGTHS BETWEEN SHORT AND'/
     >         '               LONG'/
     >         '       NOTE : WAVENUMBER = 1/WAVELENGTH  AND IS '/
     >         '                CALCULATED BY THE PROGRAM'/
     >         ' INPUT ORDER IS  DELTA  SHORT  LONG  NPASS')
        read (*,*) delta,short,long,npass
        write (*,9994)
 9994 format (' NWIND..... TYPE OF WINDOW TO APPLY'/
     >         '               = 0 GIVES NO WINDOW'/
     >         '               = 1 RECTANGULAR WINDOW'/
     >         '               = 2 BARTLETT WINDOW (TRIANGULAR)'/
     >         '               = 3 HAMMING-TUKEY WINDOW'/
     >         '               = 4 PARZEN WINDOW'/
     >         '  XLAG.....SMOOTHING PARAMETER FOR WINDOWING IDEAL'/
     >         '             FILTER IN SPATIAL DOMAIN (is disabled if'/
     >         '             no window was chosen above).'/
     >         ' nwind  xlag')
        read (*,*) nwind,xlag
c
        cuthi=1.0/short
        cutlo=1.0/long
        RCUTLO=999999.99
        IF(CUTLO .GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
        RCUTHI=1.0/CUTHI
        WAVLEN=2.0*DELTA
        FNQ1=1.0/WAVLEN
        WRITE (25,9987) FNQ1,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI
 9987 FORMAT('NYQUIST WAVENUMBER =',F10.5,' CYCLES PER DATA INTERVAL'/,
     >       'NYQUIST WAVELENGTH = ',F10.5,' LENGTH INTERVALS'/,
     >       'LOW WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >       ' CYCLES PER DATA INTERVAL '/,F15.5,
     >       ' WAVELENGTH EQUIVALENT'/,
     >       'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >       ' CYCLES PER DATA INTERVAL '/,F15.5,
     >       ' WAVELENGTH EQUIVALENT',//)
      endif
c
      if (icc .gt. 0) then
         if (numfile .ne. 2) then
            write (*,*) 'you must enter two files to run cc filter'
            stop
         endif
         write (*,*) 'CORRELATION COEFFICIENT SECTION:'
         write (*,*) 'enter placement values in order ie. 2 3 5 '
         write (*,*) 'do not repeat these values elsewhere'
         write (*,*) '1=first, 3=third, 5=fifth etc...'
         read (*,*) (numcc(i),i=1,icc)
         write (*,9995)
 9995 format ('WHAT IS THE MINIMUM CORR COEF TO BE PASSED:(0.4)'/,
     >         'WHAT IS THE MAXIMUM CORR COEF TO BE PASSED:(1.0)'/,
     >         'MINIMUM INPUT CC WITHOUT WRITING WARNING'/,
     >         'MAXIMUM INPUT CC WITHOUT WRITING WARNING'/,
     >         'CNWIND..... TYPE OF WINDOW TO APPLY'/
     >         '               = 0 GIVES NO WINDOW'/
     >         '               = 1 RECTANGULAR WINDOW'/
     >         '               = 2 BARTLETT WINDOW (TRIANGULAR)'/
     >         '               = 3 HAMMING-TUKEY WINDOW'/
     >         '               = 4 PARZEN WINDOW'/
     >         'CXLAG......SMOOTHING PARAMETER FOR WINDOWING IDEAL'/
     >         '             FILTER IN SPATIAL DOMAIN (is disabled if'/
     >         '             no window was chosen above).'/
     >         'mincc  maxcc  minccin  maxccin  cnwind  cxlag')
         read (*,*) mincc,maxcc,minccin,maxccin,cnwind,cxlag
      endif
c
      if (isd .gt. 0) then
         write (*,*) 'STRIKE-DIP SECTION:'
         write (*,*) 'enter placement values in order ie. 2 3 5 '
         write (*,*) 'do not repeat these values elsewhere'
         write (*,*) '1=first, 3=third, 5=fifth etc...'
         read (*,*) (numsd(i),i=1,isd)
         write (*,9986)
 9986 format ('ANGLE 1: >= 0.0  AND  < ANGLE2'/,
     >         'ANGLE 2: > ANGLE 1  AND <= 180.0'/,
     >         ' 1 TO PASS AZIMUTHS BETWEEN ANGLES'/,
     >         '-1 TO REJECT AZIMUTHS BETWEEN ANGLES'/,
     >         'TYPE OF WINDOW TO APPLY TO FILTER'/,
     >' 0 (NONE), 1 (RECTANGULAR), 2 (TRIANGULAR), 3 (H-T), 4 (PARZEN)'/,
     >         'LAG VALUE ON SMOOTHING WINDOW (0.1 TO 99.9)'/,
     >         ' ang1  ang2  spass  swind  slag')
         read (*,*) ang1,ang2,spass,swind,slag
      endif
c
      if (iud .gt. 0) then
         write (*,*) 'UP/DOWN CONTINUATION SECTION:'
```

```
         write (*,*) 'enter placement values in order ie. 2 3 5 '
         write (*,*) 'do not repeat these values elsewhere'
         write (*,*) '1-first, 3-third, 5-fifth etc...'
         read (*,*) (numud(i),i=1,iud)
         write (*,9985)
 9985 format('udDELTA = GRID INTERVAL IN MAP UNITS (1.0)'/,
     >       'ZCON = DISTANCE TO CONTINUE THE DATA SET'/,
     >       '       IN THE SAME LENGTH UNITS AS DELTA'/,
     >       '       (1.0 degree = 111 km)'/,
     >       '       NEGATIVE FOR UPWARD CONTINUATION'/,
     >       '       POSITIVE FOR DOWNWARD CONTINUATION'/,
     >       'udNWIND = TYPE OF WINDOW TO APPLY TO FILTER'/,
     >       '          0 (NONE), 1 (RECTANGULAR), 2 (TRIANGULAR)'/,
     >       '          3 (H-T), 4 (PARZEN)'/,
     >       'udXLAG = SMOOTHING PARAMETER FOR WINDOWING FILTER'/,
     >       '         IN SPATIAL DOMAIN. DETERMINES WHAT PERCENTAGE'/,
     >       '         DATA IS WINDOWED'/,
     >       ' delta  zcon  udnwind  udxlag')
         read (*,*) uddelta,zcon,udnwind,udxlag
      endif
c
      if (irtp .gt. 0) then
         write (*,*) 'REDUCTION TO POLE SECTION:'
         write (*,*) 'enter placement values in order ie. 2 3 5 '
         write (*,*) 'do not repeat these values elsewhere'
         write (*,*) '1-first, 3-third, 5-fifth etc...'
         read (*,*) (numrtp(i),i=1,irtp)
         write (*,9983)
 9983 format ('AZM = AZIMUTH OF Y AXIS MEASURED IN DEGREES'/,
     >       '       CLOCKWISE FROM TRUE NORTH'/,
     >       'DEC = AVERAGE DECLINATION OF THE INPUTTED'/,
     >       '       ANOMALY DATA'/,
     >       'XINC = AVERAGE INCLINATION OF THE INPUTTED'/,
     >       '        ANOMALY DATA'/,
     >       'azm  dec  xinc')
         read (*,*) azm,dec,xinc
      endif
c
      if (ideriv .gt. 0) then
         write (*,*) 'DERIVATIVE SECTION:'
         write (*,*) 'enter placement values in order ie. 2 3 5 '
         write (*,*) 'do not repeat these values elsewhere'
         write (*,*) '1-first, 3-third, 5-fifth etc...'
         read (*,*) (numderiv(i),i=1,ideriv)
         write (*,9984)
 9984 format('xyzDELTA = GRID INTERVAL'/,
     >       'NTH = ORDER OF SPATIAL DERIVATIVE TO PERFORM'/,
     >       '      ON THE DATA'/,
     >       'NWAY = DIRECTION IN WHICH TO CALCULATE THE'/,
     >       '       DERIVATIVE'/,
     >       '       0 = VERTICAL DERIVATIVE'/,
     >       '       1 = HORIZONTAL DERIVATIVE IN "X" DIRECTION'/,
     >       '       2 = HORIZONTAL DERIVATIVE IN "Y" DIRECTION'/,
     >       'xyzdelta  nth  nway')
         read (*,*) xyzdelta,nth,nway
      endif
c
      write (*,9992)
 9992 format ('THE FOLLOWING REFERS TO FFT AND IFFT'//,
     >        'NUMBER OF COLUMNS AND ROWS OF FFT ARRAY'/,
     >        'AT A POWER OF 2: (256  128)(2 16 32 64 128 256 etc)'/,
     >   'TYPE OF INPUT ARRAY INDICATES TYPE OF FOLDING TO BE USED'/,
     >        '   0  IF A POLAR REGION, ie. E AND W EDGES ARE SAME'/,
     >        '   1  IF A NON-POLAR REGION, ie. E AND W EDGES NOT SAME'/,
     >        'PERCENT OF EACH EDGE OF INPUT ARRAY TO'/,
     >        'BE FOLDED OUT: (0.1 TO 99.9)'/,
     >        'PERCENT OF EACH EDGE OF FOLDED OUT OR NORMAL ARRAY'/,
     >        'TO BE SMOOTHED TO ZERO:(0.1 TO 49.9)'/,
     >        '0 DO NOT ADD MEAN TO IFFT DATA'/,
     >        '1 ADD MEAN TO IFFT DATA'/,
     >        'nxout  nyout  itypefold  fold  prcnt  imean')
      read (*,*) nxout,nyout,itypefold,fold,prcnt,imean
c
      if (nxout .gt. 512 .or. nyout .gt. 512) then
         write (*,8999) nxout,nyout
 8999    format (1x,'SORRY',i6,1x,'OR',i6,1x,'IS GREATER THAN 512 THE',
     >           ' SIZE OF ARRAYS SET'/' IN THE SOURCE CODE ',
     >           'YOU NEED TO ACCESS SOURCE CODE AND MAKE CHANGES')
         stop
      endif
c
c
      write (*,*) 'INPUT FILE 1'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename,status='old',form='formatted')
      if (numfile .eq. 2) then
         write (*,*) 'INPUT FILE 2'
         read (*,9990) filename
         open (11, file=filename,status='old',form='formatted')
```

```
      endif
      write (*,*) 'OUTPUT OF FILE 1'
      read (*,9990) filename
      open (20, file=filename,form='formatted')
      if (numfile .eq. 2) then
         write (*,*) 'OUTPUT OF FILE 2'
         read (*,9990) filename
         open (21, file=filename,form='formatted')
      endif
c
c
 210  continue
      read (10,*) xcol
      read (10,*) xrow
      read (10,*) xcolat
      read (10,*) xlong
      read (10,*) xgridspc
      if (numfile .eq. 2) then
         read (11,*) ycol

 read (11,*) yrow

 read (11,*) ycolat
         read (11,*) ylong

 read (11,*) ygridspc
      endif
c
      xpassno=1
      ypassno=2
      do i=1,xrow
         read (10,*) (xdata(j,i),j=1,xcol)
      enddo
      if (numfile .eq. 2) then
         do i=1,yrow
            read (11,*) (ydata(j,i),j=1,ycol)
         enddo
      endif
c
      xmean=0.0
      call forwardft (1,xmean,xpassno)
      ymean=0.0
      if (numfile .eq. 2) call forwardft (2,ymean,ypassno)
c
      itottime=ilhb+icc+isd+iud+irtp+ideriv
      do i=1,itottime
         do j=1,ilhb
            if (numlhb(j) .eq. i) then
               call filter (1)
               if (numfile .eq. 2) call filter (2)
               goto 888
            endif
         enddo
         do j=1,icc
            if (numcc(j) .eq. i) then
               call correlate(xpassno,ypassno)
               goto 888
            endif
         enddo
         do j=1,isd
            if (numsd(j) .eq. i) then
               call strkpas(1)
               if (numfile .eq. 2) call strkpas(2)
               goto 888
            endif
         enddo
         do j=1,iud
            if (numud(j) .eq. i) then
               call upcon(1)
               if (numfile .eq. 2) call upcon(2)
               goto 888
            endif
         enddo
         do j=1,irtp
            if (numrtp(j) .eq. i) then
               call mag2pol(1)
               if (numfile .eq. 2) call mag2pol(2)
               goto 888
            endif
         enddo
         do j=1,ideriv
            if (numderiv(j) .eq. i) then
               call deriva(1)
               if (numfile .eq. 2) call deriva(2)
               goto 888
            endif
         enddo
c
 888     continue
      enddo
```

```
c
c
      call inverseft (1,xmean,xpassno)
      if (numfile .eq. 2) call inverseft (2,ymean,ypassno)
c
      write (20,*) xcol
      write (20,*) xrow
      write (20,*) xcolat
      write (20,*) xlong
      write (20,*) xgridspc
      do i=1,xrow
         write (20,9981) (xdata(j,i),j=1,xcol)
      enddo
 9981 format (5(f12.6,1x))
      if (numfile .eq. 2) then
         write (21,*) ycol
         write (21,*) yrow
         write (21,*) ycolat
         write (21,*) ylong

 write (21,*) ygridspc
         do i=1,yrow
            write (21,9981) (ydata(j,i),j=1,ycol)
         enddo
      endif
c
 999  continue
      close (10)
      close (11)
      close (20)
      close (21)
      close (25)
      stop
      end
c
c
c
      subroutine forwardft (num,mean,passno)
      integer num,xrow,xcol,yrow,ycol,nxout,nyout,passno,
     >        row,col
      real xdata(361,361),ydata(361,361),prcnt,mean
      complex xcdata(512,512),ycdata(512,512)
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      common /rowcol/ xrow,xcol,yrow,ycol
      common /reals/ xdata,ydata
      common /comps/ xcdata,ycdata
      COMMON  H(512,512)
      DIMENSION X(2,512,512)
      COMPLEX H
      double precision TSUM
      EQUIVALENCE (X(1,1,1),H(1,1))
c
      TSUM=0.D0
      if (num .eq. 1) then
         row=xrow
         col=xcol
         do 50 i=1,row
            do 50 j=1,col
               x(1,j,i) = xdata(j,i)
               tsum=tsum+x(1,j,i)
 50      continue
      elseif (num .eq. 2) then
         row=yrow
         col=ycol
         do 80 i=1,row
            do 80 j=1,col
               x(1,j,i) = ydata(j,i)
               tsum=tsum+x(1,j,i)
 80      continue
      endif
c
C*********************************************************************
C
C                          PROGRAM SPA2FRQ
C
C     PROGRAM SPA2FRQ TRANSFORMS AN N X N MATRIX OF SPACE-DOMAIN
C     AMPLITUDES INTO THE N X N MATRIX OF WAVE NUMBER DOMAIN
C     COEFFICIENTS.  THE TRANSFORM MAY BE USED BY FIVE
C     IN-CORE PROGRAMS TO PERFORM SPECTRAL OPERATIONS (UPCON,MAGPOL,
C     BANDPASS,STRKPASS,DERIV). FUNCTIONS PERFORMED BY THIS PROGRAM
C     INCLUDE :
C        - REMOVAL OF THE MEAN FROM THE DATA
C        - OPTIONAL WINDOWING OF THE EDGES OF THE DATA SET
C        - PADDING OF THE DATA SET WITH ZEROES TO ACHIEVE NECCESSARY
C          SIZE (A POWER OF TWO)
C        - FORWARD TRANSFORM OF THE DATA
C
C.....REQUIRED SUBROUTINES :
C
C        FFT2D, FORK, DATWND
```

```
C
C.....DIMENSIONING REQUIREMENTS :
C
C        X(2,N,N).....WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
C        H(N,N)             OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
C                           POWER OF TWO (2,4,8,16...).
C                           NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
C                           SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C.....AUTHOR :SUBROUTINES FFT2D AND FORK ARE MODIFIED FROM JON REED,
C              PURDUE UNIVERSITY, DECEMBER 1980.
C              ALL OTHER CODE WRITTEN BY:
C              JEFFREY E. LUCIUS
C              GEOPHYSICAL INTERACTIVE COMPUTING LABORATORY
C              DEPARTMENT OF GEOLOGY AND MINERALOGY
C              THE OHIO STATE UNIVERSITY
C              COLUMBUS, OHIO 43210
C
C              MARCH 25, 1985   (REVISED DEC 5, 1986)
C
c     revised once again for DEC workstations on 6 APR 90 so that
c     that this beast is actually user friendly!
c
c     revised again (judas priest this is getting old) on
c     1 AUG 90  into this present format of all fourier programs
c     combined into this program.
C*******************************************************************
C
      IF(2**INT(ALOG(FLOAT(NXOUT))/ALOG(2.0)+0.01).NE.NXOUT) THEN
        WRITE(6,1030)
        STOP
      ENDIF
      IF(2**INT(ALOG(FLOAT(NYOUT))/ALOG(2.0)+0.01).NE.NYOUT) THEN
        WRITE(6,1040)
        STOP
      ENDIF
C
C.....CALCULATE AND REMOVE THE MEAN
C
      nxin=col
      nyin=row
      ICOL=nxin
      IROW=nyin
      XMEAN1=TSUM/FLOAT(NXIN*NYIN)
      DO 210 IY=1,NYIN
        DO 210 IX=1,NXIN
          X(1,IX,IY)=X(1,IX,IY)-XMEAN1
210   CONTINUE
c      WRITE(25,1020) XMEAN1
C
C.....WINDOW THE EDGES VIA DATWND
C
      CALL DATWND(PRCNT,NXIN,NYIN,NXOUT,NYOUT,fold,itypefold)
c
c.....MATRIX IS NOW ZERO FILLED TO NXOUT BY NYOUT SIZE
C     CALCULATE AND REMOVE THE MEAN INTRODUCED BY TAPERING
C
      TSUM=0.D0
      DO 214 IY=1,NYOUT
        DO 214 IX=1,NXOUT
          TSUM=TSUM+X(1,IX,IY)
214   CONTINUE
      XMEAN2=TSUM/FLOAT(NXOUT*NYOUT)
      DO 215 IY=1,NYOUT
        DO 215 IX=1,NXOUT
          X(1,IX,IY)=X(1,IX,IY)-XMEAN2
215   CONTINUE
c      WRITE(25,1020) XMEAN2
      XMEAN=XMEAN2+XMEAN1
c      WRITE(25,1080) XMEAN
      write (25,*) passno,xmean1,xmean2,xmean
C
C.....TRANSFORM DATA TO THE WAVENUMBER DOMAIN
C
      NX=NXOUT
      NY=NYOUT
      CALL FFT2D(NX,NY,-1)
C
      mean=xmean
      if (num .eq. 1) then
         do 500 iy=1,ny
            do 500 ix=1,nx
               xcdata(ix,iy) = h(ix,iy)
 500     continue
      elseif (num .eq. 2) then
         do 580 iy=1,ny
            do 580 ix=1,nx
               ycdata(ix,iy) = h(ix,iy)
 580     continue
      endif
```

```
c
      return
c
1020  FORMAT('MEAN REMOVED ',F15.7)
1030  FORMAT(1H , 'NXOUT MUST BE A POWER OF 2: SPA2FRQ FATAL')
1040  FORMAT(1H , 'NYOUT MUST BE A POWER OF 2: SPA2FRQ FATAL')
1080  FORMAT('TOTAL MEAN REMOVED ',F15.7)
C
      END
C
C*********************************************************************
C
      SUBROUTINE FFT2D (NX,NY,NSIGN)
C
C ********************************************************************
C                                                                    *
C         "FFT2D" PERFORMS BOTH A FORWARD OR INVERSE FAST FOURIER    *
C     TRANSFORM. "FFT2D" IS THE DRIVER THAT PASSES THE CORRECT VECTORS *
C     TO "FORK" WHICH PERFORMS THE ACTUAL TRANSFORMING.              *
C    THE DIMENSIONING OF "H" MUST BE THE SAME AS IN THE MAIN PROGRAM *
C                                                                    *
C  "NSIGN" = DIRECTION OF DESIRED TRANSFORMATION                     *
C           =+1   INVERSE TRANSFORM (FREQUENCY TO SPATIAL)           *
C           =-1   FORWARD TRANSFORM (SPATIAL TO FREQUENCY)           *
C                                                                    *
C ********************************************************************
C
      COMMON  H(512,512)
      COMMON  CTEMP(512)
      COMPLEX H,CTEMP
C
      SIGNI=FLOAT(NSIGN)
      IF(IABS(NSIGN).NE.1) THEN
         WRITE(6,105)
         STOP
      ENDIF
C
C.....OPERATE BY ROWS
C
      DO 101 IY=1,NY
101      CALL FORK (NX,H(1,IY),SIGNI)
C
C.....OPERATE BY COLUMNS
C
      DO 104 IX=1,NX
         DO 102 IY=1,NY
102         CTEMP(IY)=H(IX,IY)
         CALL FORK (NY,CTEMP,SIGNI)
         DO 103 IY=1,NY
103         H(IX,IY)=CTEMP(IY)
104   CONTINUE
c
      RETURN
C
105   FORMAT(5X,'"NSIGN" MUST EQUAL +1 OR -1 FOR "FFT2D", FATAL')
C
      END
C
C*********************************************************************
C
      SUBROUTINE FORK (LXX,CX,SIGNI)
C
C*********************************************************************
C                                                                    *
C  FAST FOURIER TRANSFORM, MODIFIED FROM CLAERBOUT, J.F., FUNDAMENTAL *
C  OF GEOPHYSICAL DATA PROCESSING, MCGRAW-HILL, 1976                 *
C      FORK USES COOLEY-TUKEY ALGORITHM.
C                                                                    *
C "CX"    = DATA VECTOR TO BE TRANSFORMED                            *
C "LXX"   = LENGTH OF DATA VECTOR "CX" TO BE TRANSFORMED,            *
C              MUST BE A POWER OF 2 (LXX=2**INTEGER)                 *
C "SIGNI"= DIRECTION OF DESIRED TRANSFORMATION                       *
C           =+1.   INVERSE TRANSFORM (FREQUENCY TO SPATIAL)          *
C           =-1.   FORWARD TRANSFORM (SPATIAL TO FREQUENCY)          *
C                                                                    *
C  NORMALIZATION PERFORMED BY DIVIDING BY                            *
C      DATA LENGTH UPON THE FORWARD TRANSFORM.                       *
C*********************************************************************
C
      COMPLEX CX(LXX),CW,CTEMP,CON2
C
      LX=LXX
      LXH=LX/2
      J=1
      DO 103 I=1,LX
        IF (I.LT.J) THEN
          CTEMP=CX(J)
          CX(J)=CX(I)
          CX(I)=CTEMP
        ENDIF
        M=LXH
```

```
102       IF (J.GT.M) THEN
            J=J-M
            M=M/2
            IF (M.GE.1) GO TO 102
          ENDIF
          J=J+M
103     CONTINUE
        L=1
104     ISTEP=2*L
        CON2=(0.0,3.14159265)/FLOAT(L)*SIGNI
        DO 105 M=1,L
          CW=CEXP(CON2*FLOAT(M-1))
          DO 105 I=M,LX,ISTEP
            CTEMP=CW*CX(I+L)
            CX(I+L)=CX(I)-CTEMP
105     CX(I)=CX(I)+CTEMP
        L=ISTEP
        IF (L.LT.LX) GO TO 104
        IF (SIGNI.GT.0.0) RETURN
           SC=1./FLOAT(LX)
C
        DO 106 I=1,LX
106        CX(I)=CX(I)*SC
C
         RETURN
         END
C
C*********************************************************************
C
        SUBROUTINE DATWND (PRCNT,NX11,NY11,NX,NY,fold,itypefold)
C
C*********************************************************************
C                                                                   *
C  "DATWND" MULTIPLIES THE INPUT F(1,X,Y) BY A HALF BELL OF A HAMMIN-  *
C        TUKEY WINDOW ON ALL EDGES AND ZEROS OUT THE REMAINDER OF THE  *
C        (NX,NY) ARRAY.                                             *
C                                                                   *
C "PRCNT" =PERCENTAGE OF DATA TO BE ALTERED IN SMOOTHING TO ZERO      *
C              0.0 .LT. "PRCNT" .LE. 50.0                           *
C
c  update 2 feb 91
c     datwnd has been considerably improved such that now the subroutine
c     performs three (count them, three !!) functions.  one; a percentage
c     of the input matrix can be folded out.  two; after folding out,
c     a new percentage of the folded out matrix (or regular data if
c     folding was not performed) can be smoothed to zero.  three; the
c     manipulated data is centered within zeros to finish filling the
c     matrix to nx by ny size.  because the actual data is now centered
c     within the transformed array, it is necessary to use the
c     do loops in subroutine inverseft to correctly extract the actual
c     data
c
C*********************************************************************
C
        dimension holdme(512,512)
        COMMON  F(2,512,512)
C
        nx1=nx11
        ny1=ny11
C
c--------------------------- fold out the data based on percentage
c
        if (fold.gt.0.0 .and. fold.lt.100.0) then
c
        KX=Int(fold*FLOAT(NX1)/100.0+0.5)
        KY=Int(fold*FLOAT(NY1)/100.0+0.5)
        if (kx+nx1 .gt. nx) kx=(nx-nx1)/2
        if (ky+ny1 .gt. ny) ky=(ny-ny1)/2
        do j=1,ny1
           do i=1,nx1
              holdme(i,j)=f(1,i,j)
           enddo
        enddo
c--------------------------- fold out the columns in each row:
c                             if itypefold is 1 then the data is considered to
c                             to be a rectangular style of projection where
c                             the east and west edges of the data are not
c                             covering the same geographic region.  therefore,
c                             the folding out of data along a row is symetric
c                             with respect to the individual edge.
c
        if (itypefold .eq. 1 ) then
           do j=1,ny1
              do i=1,nx1+kx+kx
                 if (i.le.kx) f(1,i,j)=holdme(kx-i+1,j)
                 if (i.gt.kx .and. i.le.(kx+nx1)) f(1,i,j)=holdme(i-kx,j)
                 if (i.gt.(kx+nx1)) f(1,i,j)=holdme((2*nx1+kx+1-i),j)
              enddo
           enddo
c
```

```
c-------------------------- if itypefold is 0 then the array is considered to
c                          be a polar style of projection where the east and
c                          west edges are covering the same geographic area.
c                          therefore, the folding along each row is actually
c                          adding the western edge of data to the eastern
c                          edge and eastern data to western edge.
c
      elseif (itypefold .eq. 0) then
         do j=1,ny1
            do i=1,nx1+kx+kx
               if (i.le.kx) f(1,i,j)=holdme(nx1+i-kx,j)
               if (i.gt.kx .and. i.le.(kx+nx1)) f(1,i,j)=holdme(i-kx,j)
               if (i.gt.(kx+nx1)) f(1,i,j)=holdme((i-nx1-kx),j)
            enddo
         enddo
      endif
c
      if (ny1 .eq. 1) go to 333
      do j=1,ny1
         do i=1,nx1+kx+kx
            holdme(i,j)=f(1,i,j)
         enddo
      enddo
c-------------------------- fold out the rows in each column
      do i=1,nx1+kx+kx
         do j=1,ny1+ky+ky
            if (j.le.ky) f(1,i,j)=holdme(i,ky-j+1)
            if (j.gt.ky .and. j.le.(ky+ny1)) f(1,i,j)=holdme(i,j-ky)
            if (j.gt.(ky+ny1)) f(1,i,j)=holdme(i,(2*ny1+ky+1-j))
         enddo
      enddo
      ny1=ny11+2*ky
 333  nx1=nx11+2*kx
      endif
c
      if (prcnt.gt.0.0 .and. prcnt.lt.50.0) then
         KX=IFIX(PRCNT*FLOAT(NX1)/100.0+0.5)
         KY=IFIX(PRCNT*FLOAT(NY1)/100.0+0.5)
C
C.....APPLY WINDOW TO COLUMNS
C
         IF(NY1.NE.1 .AND. KY.NE.0)  THEN
            RKYPI= 3.14159265/FLOAT(KY)
            DO 10 IY=1,KY
               FACTOR=0.5*(1.0+COS(FLOAT(KY-IY+1)*RKYPI))
               IYY=NY1-IY+1
               DO 10 IX=1,NX1
                  F(1,IX,IY) = F(1,IX,IY) * FACTOR
10             F(1,IX,IYY)= F(1,IX,IYY)* FACTOR
         ENDIF
C
C.....APPLY WINDOW TO ROWS
C
         IF(KX.NE.0) THEN
            RKXPI= 3.14159265/FLOAT(KX)
            DO 30 IX=1,KX
               FACTOR=0.5*(1.0+COS(FLOAT(KX-IX+1)*RKXPI))
               IXX=NX1-IX+1
               DO 30 IY=1,NY1
                  F(1,IX,IY)=F(1,IX,IY) * FACTOR
30          F(1,IXX,IY)  =F(1,IXX,IY)* FACTOR
         ENDIF
C
c       WRITE(25,150) KX, KY
         write (25,*) kx,ky
      endif
C
C.....center and ZERO OUT REMAINDER OF ARRAY
C
      nxhalf=(nx-nx1)/2
      nyhalf=(ny-ny1)/2
      do i=1,nx1
         do j=1,ny1
            holdme(i,j)=f(1,i,j)
         enddo
      enddo
      do i=1,nxhalf
         do j=1,ny
            f(1,i,j)=0.0
         enddo
      enddo
      do i=nxhalf+1,nxhalf+nx1
         do j=1,ny
            if (j .le. nyhalf) f(1,i,j)=0.0
            if (j.gt.nyhalf  .and. j.le.nyhalf+ny1)
     >           f(1,i,j)=holdme(i-nxhalf,j-nyhalf)
            if (j .gt. nyhalf+ny1) f(1,i,j)=0.0
         enddo
      enddo
      do i=nxhalf+nx1+1,nx
```

```
        do j=1,ny
           f(1,1,j)=0.0
        enddo
     enddo
c
     RETURN
C
150  FORMAT('smoothed',i4,' values on both x edges'/,
    >       '           ',i4,'               y       ')
160  FORMAT(1H-,'"PRCNT"= ',F7.3,' OUTSIDE OF PROPER RANGE',
    >        ' NO WINDOWING PERFORMED: "DATWND"'/)
C
     END
c
c
c
     subroutine filter (num)
     integer num,npass,imean,nwind,nxout,nyout,
    >        row,col
     real prcnt,xlag,delta,cuthi,cutlo
     complex xcdata(512,512),ycdata(512,512)
     common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
     common /comps/ xcdata,ycdata
     common /lhbflt/ delta,cuthi,cutlo,xlag,npass,nwind
     COMMON  H(512,512)
     COMPLEX H
c
     if (num .eq. 1) then
        row=nyout
        col=nxout
        nx=nxout
        ny=nyout
     elseif (num .eq. 2) then
        row=nyout
        col=nxout
        nx=nxout
        ny=nyout
     endif
C ********************************************************************
C
C                            PROGRAM BANDPASS
C
C     PROGRAM BANDPASS PERFORMS HIGH, LOW, OR BANDPASS WAVENUMBER
C     FILTERING OF UNIFORMLY GRIDDED ARRAYS. INPUT MATRIX IS THE
C     WAVENUMBER DOMAIN TRANSFORM AS OUTPUT BY SPA2FRQ. AN IDEAL FILTER
C     IS CONSTUCTED IN THE WAVENUMBER DOMAIN, WINDOWED IN THE SPACE
C     DOMAIN, THEN TRANSFORMED BACK INTO THE WAVENUMBER DOMAIN TO BE
C     MULTIPLIED BY THE INPUT TRANSFORM.
C
C.....REQUIRED SUBROUTINES :
C
C         BNDPAS, FFT2D, FORK, STORE, WINDOW
C
C.....DIMENSIONING REQUIREMENTS :
C
C         H(N,N)        WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
C                       INPUT AND OUTPUT TRANSFORMED MATRIX. N MUST BE AN
C                       INTEGRAL POWER OF TWO (2,4,8,16...).
C                       NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
C                       SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C.....AUTHOR : JON REED, PURDUE UNIVERSITY, DECEMBER 1980.
C              REVISIONS BY STEVE MATESKON AND JEFF LUCIUS,
C                       OHIO STATE UNIVERSITY, JULY 1984.
C
c
c     this program, like others in the fft series, has been updated
c     to the DEC workstation system and now the program is actually
c     usable to just about anybody!  revised 21 apr 90
c
c     well, like the other programs in this package, this has been
c     updated on 4 AUG 90.  few comments have been removed - mainly
c     those comments about i/o operations not necessary to this
c     package have been removed.
c
c     update: 2 feb 90, removed need for cstore array
c
C
C*********************************************************************
C
C.....CREATE FILTER AND STORE IN ARRAY H
C
      CALL BNDPAS (CUTLO,CUTHI,NPASS,DELTA,NX,NY)
      CALL STORE(NX,NY)
C
C.....CREATE SMOOTHED FILTER
C
      IF(XLAG.GT.0.0 .AND. XLAG.LE.100.0) THEN
        IF(NWIND.GT.0.AND.NWIND.LE.4) THEN
          CALL FFT2D (NX,NY,1)
```

```
          CALL WINDOW (NX,NY,XLAG,NWIND)
          CALL FFT2D (NX,NY,-1)
        ENDIF
      ENDIF
C
C.....WRITE FILTER (WAVENUMBER DOMAIN) ONTO UNIT 30 IF IOFIL = 1
C
c       IF(IOFIL.EQ.1) THEN
c         WRITE (30,*) NX,NY,IZERO,XMEAN
c         DO 200 IY=1,NY
c           WRITE (30,*) (H(IX,IY),IX=1,NX)
c200     CONTINUE
c         WRITE(6,1040) NX,NY
c       ENDIF
C
      if (num .eq. 1) then
         do 500 i=1,row
            do 500 j=1,col
               xcdata(j,i) = xcdata(j,i)*h(j,i)
 500        continue
      elseif (num .eq. 2) then
         do 580 i=1,row
            do 580 j=1,col
               ycdata(j,i) = ycdata(j,i)*h(j,i)
 580        continue
      endif
c
      return
      end
c
C**********************************************************************
C
      SUBROUTINE BNDPAS (CCUTLO,CCUTHI,NPASS,DELTA,NX,NY)
C
C**********************************************************************
C                                                                     *
C     "BNDPAS" CALCULATES TWO QUADRANTS OF THE WAVE# RESPONSE OF      *
C     AN IDEAL BANDPASS FILTER OF A (NX,NY) MATRIX.                   *
C   ARRAY "H" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM     *
C                                                                     *
C "CCUTLO"   LOWEST  WAVE# TO BE PASSED, GE 0.0                       *
C "CCUTHI"   HIGHEST WAVE# TO BE PASSED, LE NYQUIST                   *
C "NPASS"    SWITCHES EITHER A PASS OR REJECTION BETWEEN              *
C            "CUTLO" & "CUTHI"                                        *
C             =-1 REJECT WAVENUMBERS BETWEEN THE 2 WAVENUMBERS        *
C             = 1 PASS   WAVENUMBERS BETWEEN THE 2 WAVENUMBERS        *
C "DELTA"    DATA GRID INTERVAL, IN MAP UNITS                         *
C  "NX"      NUMBER OF ROWS (POWER OF 2 GE "ICOL", 16,32,ETC)         *
C  "NY"      NUMBER OF ROWS (POWER OF 2 GE "IROW", 16,32,ETC)         *
C                                                                     *
C**********************************************************************
C
      COMMON  H(512,512)
      COMPLEX H,ZERO,ONE
      DIMENSION A(2)
      DATA A/4HPASS,4HCUT /
C
      CUTHI=CCUTHI
      CUTLO=CCUTLO
      RCUTLO=999999.99
      IF(CUTLO.GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
      RCUTHI=1.0/CUTHI
      WAVLEN=2.0*DELTA
      FNQ1=1.0/WAVLEN
c      WRITE (25,112) FNQ1,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI,NPASS
C
      IF(IABS(NPASS).NE.1) THEN
        WRITE(6,151)
        STOP
      ENDIF
      IF(CUTHI.GT.FNQ1.OR.CUTHI.LE.CUTLO.OR.CUTLO.LT.0.0) THEN
        WRITE(6,151)
        STOP
      ENDIF
C
      NXX=NX+2
      NX2=(NX/2)+1
      NY2=(NY/2)+1
      ANY2=FLOAT(NY2)
      ZERO = (0.0,0.0)
      ONE  = (1.0,0.0)
      WAY  = A(1)
C
      IF(NPASS.NE.1) THEN
        ZERO = (1.0,0.0)
        ONE  = (0.0,0.0)
        WAY  = A(2)
      ENDIF
C
      RHIY2 = (FNQ1/(ANY2*CUTHI))**2
```

```
      NHIY  = CUTHI*WAVLEN*ANY2+1.0
      NHIY=AMINO(NHIY,NY2)
      CLOWX=FLOAT(NX2)*WAVLEN*CUTLO
      CHIX =FLOAT(NX2)*WAVLEN*CUTHI
C
      IF (CUTLO.LE.0.000001) THEN
c         WRITE(6,152) WAY
        RLOWY2=0.0
        NLOWY=0
      ELSE
        IF(FNQ1-CUTHI.LT.0.00001) THEN
c           WRITE(6,153) WAY
          RLOWY2=(FNQ1/(ANY2*CUTLO))**2
          NLOWY=CUTLO*WAVLEN*ANY2+1.0
        ELSE
c           WRITE(6,154)
          RLOWY2=(FNQ1/(ANY2*CUTLO))**2
          NLOWY=CUTLO*WAVLEN*ANY2+1.0
        ENDIF
      ENDIF
C
C
C....."ZERO" OUT THE PART OF ARRAY TO BE ALTERED
C
      DO 35 IY=1,NY2
        DO 35 IX=1,NX
          H(IX,IY) = ZERO
35    CONTINUE
C
C.....OPERATE ON ROWS WHERE SOME WAVENUMBERS ARE .LE. CUTLO
C
      IF(NLOWY.NE.0) THEN
         MINS=1
         MAXS=NX2
         IF(NLOWY.EQ.1) THEN
            IF(CUTLO.GT.0.00001) MINS=NX2*CUTLO*WAVLEN+2.0001
            IF(FNQ1-CUTHI.GT.0.00001) MAXS=NX2*CUTHI*WAVLEN+1.0001
         ENDIF
         DO 102 IY=1,NLOWY
            Y2=FLOAT(IY-1)**2
            MINX=MINS
            IF(CUTLO.GT.0.000001) THEN
               MINX=CLOWX*SQRT(1.0-Y2*RLOWY2)+2.0001
            ENDIF
            MAXX=MAXS
            IF(FNQ1-CUTHI.GE.0.00001) THEN
               MAXX=CHIX*SQRT(1.0-Y2*RHIY2)+1.0001
            ENDIF
            IF(MINX.EQ.1) THEN
               H(1,IY)= ONE
               MINX=MINX+1
            ENDIF
            IF(MINX.LE.MAXX) THEN
               DO 150 IX=MINX,MAXX
                  H(NXX-IX,IY)= ONE
 150           H(IX,IY)= ONE
            ENDIF
 102     CONTINUE
      ENDIF
C
C.....OPERATE ON ROWS WHERE ALL WAVENUMBERS ARE .GT. CUTLO
C
      IF(NHIY.NE.NY2) THEN
         LL=NLOWY+1
         DO 200 IY=LL,NHIY
            Y2=FLOAT(IY-1)**2
            MAXX=CHIX*SQRT(1.0-Y2*RHIY2)+1.0001
            H(1,IY)= ONE
            IF(MAXX.NE.1) THEN
               DO 180 IX=2,MAXX
                  H(NXX-IX,IY)= ONE
 180           H(IX,IY)=ONE
            ENDIF
 200     CONTINUE
      ELSE
         IF(NLOWY+1.GT.NY2) RETURN
         LL=NLOWY+1
         DO 215 IY=LL,NY2
            DO 215 IX=1,NX
               H(IX,IY)= ONE
 215     CONTINUE
      ENDIF
C
      RETURN
C
112   FORMAT(/1X,'NYQUIST WAVENUMBER =',F10.5,'CYCLES PER DATA INTERVAL'
     >        /1X,'NYQUIST WAVELENGTH = ',F10.5,' LENGTH INTERVALS'/
     >          1X,'LOW  WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
     >     ' CYCLES PER DATA INTERVAL',3X,F15.5,' WAVELENGTH EQUIVALENT'/
     >          1X,'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
```

```
     >         ' CYCLES PER DATA INTERVAL',3X,F15.5,
     >         ' WAVELENGTH EQUIVALENT',/1X,'NPASS= ',I1//)
151   FORMAT(5X ,'IMPOSSIBLE FILTER CONSTRUCTION IS SPECIFIED. FATAL')
152   FORMAT(1H-,'LOW  ',A4,' FILTER IS BEING CONSTRUCTED IN "BNDPAS"'/)
153   FORMAT(1H-,'HIGH ',A4,' FILTER IS BEING CONSTRUCTED IN "BNDPAS"'/)
154   FORMAT(1H-,'BAND PASS FILTER IS BEING CONSTRUCTED IN "BNDPAS"'/)
C
      END
C
C
C*********************************************************************
C
      SUBROUTINE STORE(NNX,NNY)
C
      COMMON  H(512,512)
      COMPLEX H
C
      IF(NNY.EQ.1) RETURN
      NX=NNX
      NY=NNY
      NXH=NX/2+1
      NXX=NX+2
      NYH=NY/2+2
      NYL=NYH-1
      DO 15 IY=NYH,NY
         NYL=NYL-1
         H(1,IY)=H(1,NYL)
         DO 10 IX=2,NX
            H(IX,IY)=CONJG(H(NXX-IX,NYL))
 10      CONTINUE
         H(NXH,IY)=H(NXH,NYL)
 15   CONTINUE
C
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE WINDOW (NX,NY,XLAG,NWIND)
C
C*********************************************************************
C "WINDOW" PERFORMS 2-DIMENSION WINDOWING OVER A 4 QUAD. DATA ARRAY    *
C      EACH QUAD. IS SEPERATELY WINDOWED. THE 1.0 COEFFICENT IS ALWAYS *
C      THE OUTER MOST CORNER OF THE ARRAY.                              *
C  FOR ONE DIMENSIONAL WINDOW, LET NY=1                                 *
C                                                                       *
C  "NX"     = NUMBER OF COLUMNS IN DATA MATRIX                          *
C  "NY"     = NUMBER OF ROWS    IN DATA MATRIX                          *
C  "XLAG"   = SMOOTHING PARAMETER FOR WINDOWING IDEAL FILTER IN SPATIAL *
C                DOMAIN. DETERMINES WHAT PERCENTAGE OF DATA IS WINDOWED *
C                (NX*XLAG/100.0) THE REMAINDER IS SET TO 0.0. I.E. THE  *
C                     SMALLER "XLAG" THE SMOOTHER THE WINDOWING.        *
C             "XLAG" MUST BE .GT. 0.0 .AND. LE. 100.0 FOR WINDOWING     *
C                VALUES OUTSIDE OF THIS RESULTS IN NO WINDOWING         *
C                THE SMALLER THE "XLAG" THE SMOOTHER THE FILTER.        *
C                                                                       *
C  "NWIND" = TYPE OF WINDOW TO APPLY                                    *
C            =0 GIVES NO WINDOW                                         *
c            =1 gives a rectangular window
C            =2 GIVES BARTLETT WINDOW(TRIANGLE WINDOW)                  *
C            =3 GIVES HAMMING-TUKEY WINDOW                              *
C            =4 GIVES PARZEN WINDOW                                     *
C                                                                       *
C *********************************************************************
C
      COMMON  H(512,512)
      COMPLEX H
C
      IF(XLAG.LE.0.0 .OR. XLAG.GT.100.0) THEN
        WRITE(6,50) XLAG
        RETURN
      ENDIF
C
      LAG=FLOAT(NX)*XLAG/200.0+0.5
      LAG=AMAX0(LAG,2)
      PI=3.14159265
      NXX=NX+2
      NYY=NY+2
      NX2=(NX/2)+1
      XNXR=FLOAT(NX2)
      XNX=1.0/FLOAT(NX2)
      NY2=(NY/2)+1
      IF (NY.EQ.1) NY2=1
      YNY=1.0/FLOAT(NY2)
C
      RADIUS=FLOAT(LAG)*XNX
      RADI=1.0/RADIUS
      RAD2= RADIUS*RADIUS
      NRAD=FLOAT(NY2)*RADIUS+1.0001
C
```

```
C.....APPLY RECTANGULAR WINDOW TO FILTER
C
      IF(NWIND.EQ.1) THEN
C
        IF(NRAD.NE.0) THEN
          MAX=RADIUS*XNXR+1.0001
C
          IF(MAX.EQ.0) THEN
            H(1,1)=(0.0,0.0)
            MAX=MAX+1
          ENDIF
C
          LL=MAX+1
          DO 155 II=LL,NX2
            H(NXX-II,1)=(0.0,0.0)
155       H(II,1)= (0.0,0.0)
        ENDIF
C
        IF(NRAD.NE.1.AND.NRAD.NE.0) THEN
          DO 102 IY=2,NRAD
            IYY=NYY-IY
            YLEN2=(FLOAT(IY-1)*YNY)**2
            MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
C
            IF(MAX.EQ.0) THEN
              H(1,IY)=(0.0,0.0)
              H(IY,IYY)=(0.0,0.0)
              MAX=MAX+1
            ENDIF
C
            LL=MAX+1
            DO 105 II=LL,NX2
              H(NXX-II,IY)=(0.0,0.0)
              H(II,IY)=(0.0,0.0)
              H(NXX-II,IYY)=(0.0,0.0)
105         H(II,IYY)=(0.0,0.0)
102       CONTINUE
        ENDIF
C
C          WRITE(25,660) XLAG,LAG
         IF(NRAD.EQ.NY2) RETURN
         LL=NRAD+1
         DO 108 I=LL,NY2
           IYY=NYY-I
           H(1,I)=(0.0,0.0)
           H(1,IYY)=(0.0,0.0)
           DO 109 J=2,NX2
             H(J,I)=(0.0,0.0)
             H(NXX-J,I)=(0.0,0.0)
             H(J,IYY)=(0.0,0.0)
109        H(NXX-J,IYY)=(0.0,0.0)
108      CONTINUE
C
C.....APPLY BARTLETT WINDOW TO FILTER
C
      ELSEIF(NWIND.EQ.2) THEN
C
        IF(NRAD.NE.0) THEN
          MAX=RADIUS*XNXR+1.0001
C
          IF(MAX.GE.2) THEN
            DO 253 LL=2,MAX
              XI=FLOAT(LL-1)*XNX
              FACTOR=1.0-XI*RADI
              H(LL,1)=H(LL,1)*FACTOR
              MX=NXX-LL
253         H(MX,1)=H(MX,1)*FACTOR
          ENDIF
C
          LL=MAX+1
          DO 255 II=LL,NX2
            H(NXX-II,1)=(0.0,0.0)
255       H(II,1)=(0.0,0.0)
        ENDIF
C
        IF(NRAD.NE.1.AND.NRAD.NE.0) THEN
          DO 202 IY=2,NRAD
            IYY=NYY-IY
            XI=FLOAT(IY-1)*YNY
            YLEN2= XI*XI
            MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
            FACTOR=1.0-XI*RADI
            H(1,IYY)=H(1,IYY)*FACTOR
            H(1,IY)=H(1,IY)*FACTOR
C
            IF(MAX.GE.2) THEN
              DO 203 LL=2,MAX
                XI=SQRT((FLOAT(LL-1)*XNX)**2+YLEN2)
                FACTOR=1.0-XI*RADI
                H(LL,IYY)=H(LL,IYY)*FACTOR
                H(LL,IY)=H(LL,IY)*FACTOR
```

```
               MX=NXX-LL
               H(MX,IYY)=H(MX,IYY)*FACTOR
203          H(MX,IY)=H(MX,IY)*FACTOR
           ENDIF
C
           LL=MAX+1
           DO 205 II=LL,NX2
             H(NXX-II,IYY)=(0.0,0.0)
             H(II,IYY)=(0.0,0.0)
             H(NXX-II,IY)=(0.0,0.0)
205        H(II,IY)=(0.0,0.0)
202      CONTINUE
       ENDIF
C
C        WRITE(25,661) XLAG,LAG
       IF(NRAD.EQ.NY2) RETURN
       LL=NRAD+1
       DO 208 I=LL,NY2
         IYY=NYY-I
         H(1,I)=(0.0,0.0)
         H(1,IYY)=(0.0,0.0)
         DO 209 J=2,NX2
           H(J,IYY)=(0.0,0.0)
           H(NXX-J,IYY)=(0.0,0.0)
           H(J,I)=(0.0,0.0)
209      H(NXX-J,I)=(0.0,0.0)
208    CONTINUE
C
C.....APPLY HAMMING-TUKEY WINDOW TO FILTER
C
      ELSEIF(NWIND.EQ.3) THEN
C
        IF(NRAD.NE.0) THEN
          PIRADI=PI*RADI
          MAX=RADIUS*XNXR+1.0001
C
          IF(MAX.GE.2) THEN
            DO 353 LL=2,MAX
              XI=FLOAT(LL-1)*XNX
              FACTOR=0.5*(1.0+COS(PIRADI*XI))
              H(LL,1)=H(LL,1)*FACTOR
              MX=NXX-LL
353         H(MX,1)=H(MX,1)*FACTOR
          ENDIF
C
          LL=MAX+1
          DO 355 II=LL,NX2
            H(NXX-II,1)=(0.0,0.0)
355       H(II,1)=(0.0,0.0)
        ENDIF
C
        IF(NRAD.NE.1.AND.NRAD.NE.0) THEN
          DO 302 IY=2,NRAD
            XI=FLOAT(IY-1)*YNY
            YLEN2= XI*XI
            IYY=NYY-IY
            MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
            FACTOR=0.5*(1.0+COS(PIRADI*XI))
            H(1,IYY)=H(1,IYY)*FACTOR
            H(1,IY)=H(1,IY)*FACTOR
C
            IF(MAX.GE.2) THEN
              DO 303 LL=2,MAX
                XI=SQRT((FLOAT(LL-1)*XNX)**2+YLEN2)
                FACTOR=0.5*(1.0+COS(PIRADI*XI))
                H(LL,IYY)=H(LL,IYY)*FACTOR
                H(LL,IY)=H(LL,IY)*FACTOR
                MX=NXX-LL
                H(MX,IYY)=H(MX,IYY)*FACTOR
303           H(MX,IY)=H(MX,IY)*FACTOR
            ENDIF
C
            LL=MAX+1
            DO 305 II=LL,NX2
              H(NXX-II,IYY)=(0.0,0.0)
              H(II,IYY)=(0.0,0.0)
              H(NXX-II,IY)=(0.0,0.0)
305         H(II,IY)=(0.0,0.0)
302       CONTINUE
        ENDIF
C
C         WRITE(25,662) XLAG,LAG
        IF(NRAD.EQ.NY2) RETURN
        LL=NRAD+1
        DO 308 I=LL,NY2
          IYY=NYY-I
          H(1,I)=(0.0,0.0)
          H(1,IYY)=(0.0,0.0)
          DO 309 J=2,NX2
            H(J,IYY)=(0.0,0.0)
```

```
            H(NXX-J,IYY)=(0.0,0.0)
            H(J,I)=(0.0,0.0)
309         H(NXX-J,I)=(0.0,0.0)
308       CONTINUE
C
C.....APPLY PARZEN WINDOW TO FILTER
C
        ELSEIF(NWIND.EQ.4) THEN
C
          IF(NRAD.NE.0) THEN
            MAX=RADIUS*XNXR+1.0001
            N2RAD=FLOAT(NY2)*RADIUS*0.5+1.0001
            MAX2=SQRT(RAD2/4.0)*XNXR+1.0001
            FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
            H(1,1)=H(1,1)*FACTOR
C
            IF(MAX2.GE.2) THEN
              DO 453 LL=2,MAX2
                XI=FLOAT(LL-1)*XNX
                FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
                H(LL,1)=H(LL,1)*FACTOR
                MX=NXX-LL
453           H(MX,1)=H(MX,1)*FACTOR
            ENDIF
C
            KOUNT=MAX2+1
            DO 457 LL=KOUNT,MAX
              XI=FLOAT(LL-1)*XNX
              FACTOR=2.0*(1.0-(XI*RADI))**3
              H(LL,1)=H(LL,1)*FACTOR
              MX=NXX-LL
457         H(MX,1)=H(MX,1)*FACTOR
            LL=MAX+1
            DO 455 II=LL,NX2
              H(NXX-II,1)=(0.0,0.0)
455         H(II,1)=(0.0,0.0)
          ENDIF
C
          IF(NRAD.NE.1.AND.NRAD.NE.0) THEN
            DO 402 IY=2,NRAD
              IYY=NYY-IY
              XI=FLOAT(IY-1)*YNY
              YLEN2= XI*XI
              MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
              N2RAD=FLOAT(NY2)*RADIUS*0.5+1.0001
C
              IF(IY.GT.N2RAD) THEN
                KOUNT=2
                FACTOR=2.0*(1.0-(XI*RADI))**3
                H(1,IYY)=H(1,IYY)*FACTOR
                H(1,IY)=H(1,IY)*FACTOR
              ELSE
                MAX2=SQRT(RAD2/4.0-YLEN2)*XNXR+1.0001
                FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
                H(1,IYY)=H(1,IYY)*FACTOR
                H(1,IY)=H(1,IY)*FACTOR
C
                IF(MAX2.GE.2) THEN
                  DO 403 LL=2,MAX2
                    XI=SQRT((FLOAT(LL-1)*XNX)**2+YLEN2)
                    FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
                    H(LL,IYY)=H(LL,IYY)*FACTOR
                    H(LL,IY)=H(LL,IY)*FACTOR
                    MX=NXX-LL
                    H(MX,IYY)=H(MX,IYY)*FACTOR
403               H(MX,IY)=H(MX,IY)*FACTOR
                ENDIF
C
                KOUNT=MAX2+1
              ENDIF
C
              DO 407 LL=KOUNT,MAX
                XI=SQRT((FLOAT(LL-1)*XNX)**2+YLEN2)
                FACTOR=2.0*(1.0-(XI*RADI))**3
                H(LL,IYY)=H(LL,IYY)*FACTOR
                H(LL,IY)=H(LL,IY)*FACTOR
                MX=NXX-LL
                H(MX,IYY)=H(MX,IYY)*FACTOR
407           H(MX,IY)=H(MX,IY)*FACTOR
              LL=MAX+1
              DO 405 II=LL,NX2
                H(NXX-II,IYY)=(0.0,0.0)
                H(II,IYY)=(0.0,0.0)
                H(NXX-II,IY)=(0.0,0.0)
405           H(II,IY)=(0.0,0.0)
402         CONTINUE
          ENDIF
C
c          WRITE(25,663) XLAG,LAG
          IF(NRAD.EQ.NY2) RETURN
```

```
         LL=NRAD+1
         DO 408 I=LL,NY2
           IYY=NYY-I
           H(1,I)=(0.0,0.0)
           H(1,IYY)=(0.0,0.0)
           DO 409 J=2,NX2
             H(J,IYY)=(0.0,0.0)
             H(NXX-J,IYY)=(0.0,0.0)
             H(J,I)=(0.0,0.0)
409        H(NXX-J,I)=(0.0,0.0)
408      CONTINUE
C
C.....DO NOT APPLY A WINDOW TO FILTER
C
       ELSEIF(NWIND.EQ.0) THEN
c         WRITE(25,664)
       ENDIF
C
       RETURN
C
50     FORMAT(3X,'INPUTTED XLAG OF ',F5.2,' EXCEEDS PERMISSIBLE ',
     >       'RANGE OF.GT. 0.0 .AND. .LE. 100.0, NO WINDOWING PERFORMED')
660    FORMAT ('RECTANGULAR WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
661    FORMAT ('BARTLETT WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
662    FORMAT ('HAMMING-TUKEY WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
663    FORMAT ('PARZEN WINDOW USED XLAG= ',F7.3,4X,'LAG= ',I5)
664    FORMAT ('NO WINDOWING HAS BEEN APPLIED ; XLAG= ',F7.3)
C
       END
C
c
       subroutine correlate(xpassno,ypassno)
       integer xrow,xcol,yrow,ycol,nxout,nyout,
     >         xpassno,ypassno,zerocnt(512,512),cnwind
       real mincc,maxcc,ccwinout,prcnt,
     >     pctpr3,pctpr4,minccin,maxccin,cxlag
       complex h(512,512),power5,power6,totpwr
       COMPLEX  X(512,512),Y(512,512),zero,
     >          POWER1,POWER2,POWER3,POWER4,XPOWER,TPOWER
       REAL CCOEF,CCIN,CCOUT
       DATA ZERO/(0.000000,0.000000)/
       common /rowcol/ xrow,xcol,yrow,ycol
       common /comps/ x,y
       common /ccflt/ mincc,maxcc,minccin,maxccin,cnwind,cxlag
       common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
       common h
c
c---------------------------------- subroutine description
c
c      correlate finds the correlation coefficient between each
c      wavenumber component of the two input arrays.  each cc is
c      normalized to range between -1.0 through 0.0 to 1.0.  the
c      cc is the cosine of the phase angle difference between
c      two wavenumber components.
c
c      revised 4 aug 90:  i've added the windowing functions
c      available from the bandpassing subroutines to this cc-
c      filter.  try them if you like.
c
c      updates 1 feb 91:  change calculation of correlation
c      coefficient from a summation based formula to the cosine of
c      the phase angle difference.
c
c
       nx=nxout
       ny=nyout
c
       if (xcol .ne. ycol  .or.  xrow .ne. yrow ) then
          write (*,*) 'NO MATCH BETWEEN ROW OR COLUMN'
          write (*,*) 'CORRELATION COEF MAY NOT BE CORRECT'
          write (*,*) 'PASSNUMBERS=',xpassno,ypassno
          write (*,*) 'FILE 1: ROW COL =',xrow,xcol
          write (*,*) 'FILE 2: ROW COL =',yrow,ycol
       endif
c
       pi=3.141592654
       twopi=6.283185307
       POWER1=ZERO
       POWER2=ZERO
       POWER3=ZERO
       POWER4=ZERO
       XPOWER=ZERO
       TPOWER=ZERO
c
       DO 110 j=1,NY
          DO 120 i=1,NX
c-------------------------------- zerocnt array is a flagging array used to
c                                 set the windowing array h to equal
c                                 (0.0,0.0) or (1.0,0.0).  a little inspection
c                                 of subroutine BNDPAS will help illuminate
```

```
c                               the principle.
            zerocnt(i,j)=1
C
C  SUM THE POWERS & CROSS PRODUCTS OF THE INPUT MAPS.
C
            POWER1=POWER1+(X(I,J)*CONJG(X(I,J)))
            POWER2=POWER2+(Y(I,J)*CONJG(Y(I,J)))
            XPOWER=XPOWER+(X(I,J)*CONJG(Y(I,J)))
C
c------- xrad is the phase angle of the x array wavenumber and
c        yrad is the phase angle of the y array wavenumber.  the
c        cosine of the minimum phase difference is the correlation
c        of the two wavenumbers.  to find the minimum phase difference
c        it is necessary to adjust xrad or yrad with integer values
c        of pi.  so....do not change the order of the if statements !!
c
            xrad=atan(aimag(x(i,j))/(real(x(i,j))))
            if (real(x(i,j)).lt.0.0) xrad=xrad+pi
            if (aimag(x(i,j)).lt.0.0) xrad=xrad+twopi
            yrad=atan(aimag(y(i,j))/(real(y(i,j))))
            if (real(y(i,j)).lt.0.0) yrad=yrad+pi
            if (aimag(y(i,j)).lt.0.0) yrad=yrad+twopi
            delrad=abs(xrad-yrad)
            ccoef=cos(delrad)
c
            IF (CCOEF .GT. maxcc  .or.  CCOEF .LT. mincc) THEN
               X(I,J)=ZERO
               Y(I,J)=ZERO
               zerocnt(i,j)=0
            ENDIF
c
C  SUM THE POWERS & CROSS PRODUCTS FOR THE OUTPUT MAPS.
C
            POWER3=POWER3+(X(I,J)*CONJG(X(I,J)))
            POWER4=POWER4+(Y(I,J)*CONJG(Y(I,J)))
            TPOWER=TPOWER+(X(I,J)*CONJG(Y(I,J)))
 120     CONTINUE
 110  CONTINUE
C
C  CALCULATE THE C.C. FOR THE INPUT MAPS.
C
      if (power1 .eq. zero  .or. power2 .eq. zero) then
         write (*,*) 'power1 =',power1,xpassno
         write (*,*) 'power2 =',power2,ypassno
         ccin=9999.9
      else
         CCIN=REAL(XPOWER/SQRT(POWER1*POWER2))
      endif
C
C  CALCULATE THE C.C. FOR THE OUTPUT MAPS.
C
      if (power3 .eq. zero  .or.  power4 .eq. zero) then
         write (*,*) 'power3 =',power3,xpassno
         write (*,*) 'power4 =',power4,ypassno
         ccout=9999.9
      else
         CCOUT=REAL(TPOWER/SQRT(POWER3*POWER4))
      endif
C
C  CALCULATE THE PERCENTAGE OF THE POWER RETAINED IN THE FILTERED
C  MAPS.
C
      if (power1 .eq. zero  .or. power2 .eq. zero) then
         pctpr1=9999.9
         pctpr2=9999.9
      else
         PCTPR1=(POWER3/POWER1)*100.0
         PCTPR2=(POWER4/POWER2)*100.0
      endif
C
C  WRITE THE C.C. FOR THE INPUT & OUTPUT MAPS TO FILE 6.
C
c      WRITE (6,444) CCIN
c      WRITE (6,555) CCOUT
C
C  WRITE THE POWER PERCENTAGES TO FILE 6.
C
c      WRITE (6,666) PCTPR1,PCTPR2
c 444  FORMAT (' ','THE CORRELATION COEFFICIENT BETWEEN THE INPUT '
c     *     ,'MAPS IS ',F6.3)
c 555  FORMAT (' ','THE CORRELATION COEFFICIENT BETWEEN THE OUTPUT '
c     *     ,'MAPS IS ',F6.3)
c 666  FORMAT (' ','THE PERCENTAGE OF THE TOTAL POWER IN MAP ONE',
c     *' PASSED IS',F7.3,'%',/,' THE PERCENTAGE OF THE TOTAL POWER',
c     *' IN MAP TWO PASSED IS',F7.3,'%')
C
c
      write (25,888) xpassno,ypassno,ccin,ccout,pctpr1,pctpr2
 888  format (2i6,4f10.3)
      if (ccin .lt. minccin) write (*,*)xpassno,ypassno,ccin,' <min'
```

```
      if (ccin .gt. maxccin) write (*,*)xpassno,ypassno,ccin,' >max'
c
c---------------------------------- the following if statement controls
c                                    the windowing functions for smoothing
c                                    the output arrays and calculates a new
c                                    output correlation coefficient and
c                                    percents of power retained in the
c                                    windowed arrays because
c                                    the data will change slightly with
c                                    windowing
c
c      write (*,*) '1 = zerocnt, 0 != zerocnt'
c      read (*,*) i
c      if (i .eq. 1) then
c         write (50,*) nx
c         write (50,*) ny
c         do i=1,ny
c            write (50,9970) (zerocnt(j,i),j=1,nx)
c         enddo
c 9970    format (20(i3,1x))
c      endif
c
      if (cnwind .ge. 1  .and.  cnwind .le. 4) then
         power5=zero
         power6=zero
         totpwr=zero
         do 300 i=1,ny
            do 300 j=1,nx
               h(j,i)= (1.0,0.0)
               if (zerocnt(j,i) .eq. 0) h(j,i) = (0.0,0.0)
 300        continue
         call store (nx,ny)
         call fft2d (nx,ny,1)
         call window (nx,ny,cxlag,cnwind)
         call fft2d (nx,ny,-1)
         do 330 iy=1,ny
            do 330 ix=1,nx
               x(ix,iy) = x(ix,iy)*h(ix,iy)
               y(ix,iy) = y(ix,iy)*h(ix,iy)
               power5=power5+(x(ix,iy)*conjg(x(ix,iy)))
               power6=power6+(y(ix,iy)*conjg(y(ix,iy)))
               totpwr=totpwr+(x(ix,iy)*conjg(y(ix,iy)))
 330     continue
         if (power5 .eq. zero  .or.  power6 .eq. zero) then
            write (*,*) 'power5 =',power5,xpassno
            write (*,*) 'power6 =',power6,ypassno
            ccwinout=9999.9
            go to 340
         endif
         if (power1 .eq. zero  .or.  power2 .eq. zero) then
            pctpr3=9999.9
            pctpr4=9999.9
            go to 340
         endif
         ccwinout=real(totpwr/sqrt(power5*power6))
         pctpr3=(power5/power1)*100.0
         pctpr4=(power6/power2)*100.0
 340     continue
      write (25,888) xpassno,ypassno,ccin,ccwinout,pctpr3,pctpr4
c 888  format (2i6,4f10.3)
      endif
c
      return
      end
c
c
c
      subroutine inverseft (num,mean,passno)
      integer num,xrow,xcol,yrow,ycol,row,col,passno
      real xdata(361,361),ydata(361,361),mean
      complex xcdata(512,512),ycdata(512,512)
      common /rowcol/ xrow,xcol,yrow,ycol
      common /reals/ xdata,ydata
      common /comps/ xcdata,ycdata
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      COMMON  H(512,512)
      DIMENSION X(2,512,512),holdme(361,361)
      COMPLEX H
      EQUIVALENCE  (X(1,1,1),H(1,1))
c
      if (num .eq. 1) then
         ny=nyout
         nx=nxout
         row=xrow
         col=xcol
         do 50 j=1,ny
            do 50 i=1,nx
               h(i,j) = xcdata(i,j)
 50         continue
      elseif (num .eq. 2) then
```

```
         ny=nyout
         nx=nxout
         row=yrow
         col=ycol
         do 80 j=1,ny
            do 80 i=1,nx
               h(i,j) = ycdata(i,j)
 80         continue
      endif
c
c*******************************************************************
c
c                          PROGRAM FRQ2SPA
c
c     PROGRAM FRQ2SPA INVERSE TRANSFORMS AN N X N MATRIX OF WAVENUMBER
c     DOMAIN COEFFICIENTS INTO THE N X N MATRIX OF SPACE DOMAIN
c     AMPLITUDES.  FUNCTIONS PERFORMED BY THIS PROGRAM INCLUDE :
c        - INVERSE TRANSFORM OF THE DATA
c        - RESTORING THE MEAN TO THE DATA
c        - CALCULATION OF SPACE DOMAIN MAXIMUM AND MINIMUM AMPLITUDES
c
c.....REQUIRED SUBROUTINES :
c
c        FFT2D, FORK
c
c.....DIMENSIONING REQUIREMENTS :
c
c        X(2,N,N).....WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
c        H(N,N)           OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
c                         POWER OF TWO (2,4,8,256...).
c                         NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
c                         SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
c
c
c.....AUTHOR : JEFF LUCIUS
c              DEPARTMENT OF GEOLOGY AND MINERALOGY
c              OHIO STATE UNIVERSITY, DECEMBER 1984.
c
c     revised: 8 AUG 90
c     updated: 2 feb 91
c              added do loops that find the data portion of the
c              zero centered inverse transformed data.  a look at
c              subroutine datwnd will help figure this out.
c
c*******************************************************************
c
c.....INVERSE TRANSFORM DATA TO THE SPACE DOMAIN
c
      irow=row
      icol=col
c
      CALL FFT2D (NX,NY,+1)
c
      nxhalf=(nx-icol)/2
      nyhalf=(ny-irow)/2
      do i=nxhalf+1,nxhalf+icol
         do j=nyhalf+1,nyhalf+irow
             holdme(i-nxhalf,j-nyhalf)=x(1,i,j)
         enddo
      enddo
c
      total=0.0
      DO 210 J=1,Irow
         DO 210 I=1,Icol
            x(1,i,j)=holdme(i,j)

    total=total+x(1,i,j)
 210  CONTINUE
      xmean=total/float(icol*irow)
      IF(IMEAN .EQ. 1) THEN
         do 215 j=1,irow
            do 215 i=1,icol
               x(1,i,j)=x(1,i,j)+mean
 215     continue
      ENDIF
c
      XMIN= 1.0E20
      XMAX=-1.0E20
      DO 220 J=1,irow
         if (num .eq. 1) then
            do i=1,icol
               xdata(i,j) = x(1,i,j)
            enddo
         elseif (num .eq. 2) then
            do i=1,icol
               ydata(i,j) = x(1,i,j)
            enddo
         endif
         DO 220 I=1,Icol
            XMIN=AMIN1(XMIN,X(1,I,J))
```

```
            XMAX=AMAX1(XMAX,X(1,I,J))
            IF(XMAX.EQ.X(1,I,J)) THEN
              IMAX=I
              JMAX=J
            ENDIF
            IF(XMIN.EQ.X(1,I,J)) THEN
              IMIN=I
              JMIN=J
            ENDIF
220   CONTINUE
c      WRITE(25,1020) XMAX,jMAX,iMAX,XMIN,jMIN,iMIN,xmean,passno
      write (25,9980) passno,xmean,xmax,jmax,imax,xmin,jmin,imin
 9980 format (i5,2x,f13.5,2x,f13.5,2x,i4,2x,i4,f13.5,2x,i4,2x,i4)
c
1020  FORMAT('MAXIMUM OF IFFT = ',E15.7,' AT (',I3,',',I3,')'/,
     >        'MINIMUM = ',E15.7,' AT (',I3,',',I3,')'/,
     >        'MEAN AFTER IFFT = ',e15.7,'  FOR PASS',i6,/)
c
      return
      END
c
c
c
c
      subroutine STRKPAS (num)
      integer num,xrow,xcol,yrow,ycol,spass,swind,
     >        imean,nxout,nyout,row,col
      real prcnt,slag
      complex xcdata(512,512),ycdata(512,512)
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      common /comps/ xcdata,ycdata
      common /striking/ ang1,ang2,spass,swind,slag
      COMMON  H(512,512)
      COMPLEX H
 c
      if (num .eq. 1) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      elseif (num .eq. 2) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      endif
C ********************************************************************
C                                                                    *
C       "STRIKE" PERFORMS A STRIKE SENSITIVE FILTERING (FAN FILTER)  *
C    ON UNIFORMLY GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.          *
C                                                                    *
C  "sPASS" = CONTROLS IF DATA IS TO BE PASSED OR REJECTED BETWEEN    *
C             "ANG1" AND "ANG2".                                     *
C                = 1  PASS   AZIMUTHS BETWEEN ANGLES "ANG1" & "ANG2" *
C                =-1  REJECT AZIMUTHS BETWEEN ANGLES "ANG1" & "ANG2" *
C  "ANG1"  = SMALLEST ANGLE, GE 0.0   .AND. LT "ANG2"
C  "ANG2"  = LARGEST ANGLE, GT "ANG1" .AND. LE 180.00
C                                                                    *
c
c     updates and revisions:
c
c     23 dec 91; added this strike pass routine to the fourier
c                program.  required removal of write statements
c
C ********************************************************************
C
C
      CALL STRIKE (ANG1,ANG2,sPASS,NX,NY)
C
C------------- CREATE SMOOTHED FILTER
C
      IF(sLAG.gt.0.0 .and. sLAG.lt.100.0) then
         if (swind.gt.0  .and.  swind.le.4) then
            CALL FFT2D (NX,NY,1)
            CALL WINDOW (NX,NY,sLAG,sWIND)
            CALL FFT2D (NX,NY,-1)
         endif
      endif
 C
 C------------- SET UP TO WRITE 30 if desired
 C
 c      DO 356 IY=1,NY
 c         WRITE (30) (H(IX,IY),IX=1,NX)
 c 356  CONTINUE
 C
 C------------- ACCESS TRANSFORM OF DATA  & MULTIPLY *FILTER (CONVOLVING)
 C
      if (num .eq. 1) then
         do 500 i=1,row
            do 500 j=1,col
               xcdata(j,i) = xcdata(j,i)*h(j,i)
```

```
 500        continue
      elseif (num .eq. 2) then
         do 580 i=1,row
            do 580 j=1,col
               ycdata(j,i) = ycdata(j,i)*h(j,i)
 580        continue
      endif
c
      return
      end
c
C
      SUBROUTINE STRIKE (AANG1,AANG2,NNPASS,NNX,NNY)
c
      COMPLEX H,ZERO,ONE
      COMMON H(512,512)
      DATA DG2RAD,DG90 /0.017453293,1.570796327/
C
C ********************************************************************
C     "STRIKE" CREATES A STRIKE SENSITIVE FILTER (FAN FILTER)         *
C   FOR 2 QUADRANTS OF THE (NX,NY) MATRIX                             *
C   ARRAY "H" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM     *
C                                                                     *
C            (ANGLES ARE MEASURED IN DEGREES CLOCKWISE FROM NORTH)    *
C  "ANG1 " = SMALLEST ANGLE, GE 0.0  .AND. LT ANG2                    *
C  "ANG2 " = LARGEST  ANGLE, GT ANG1 .AND. LE 180.0                   *
C  "NPASS" = STATES IF DATA IS PASSED OR REJECTED BETWEEN THE 2 ANGLES *
C               =-1  REJECT AZIMUTHS BETWEEN ANGLES "ANG1" & "ANG2"   *
C               = 1  PASS   AZIMUTHS BETWEEN ANGLES "ANG1" & "ANG2"   *
C  "NX"     = NUMBER OF ROWS (POWER OF 2 GE "NX1", 16,32,ETC) MAX=128  *
C  "NY"     = NUMBER OF ROWS (POWER OF 2 GE "NY1", 16,32,ETC) MAX=128  *
C                                                                     *
C ********************************************************************
C
      NX=NNX
      NY=NNY
      ANG1  = AANG1
      ANG2  = AANG2
      NPASS = NNPASS
C
      IF(ANG1.LT.ANG2 .AND. ANG1.GE.0.0 .AND. ANG2.LE.180.0) GOTO 109
         WRITE(6,125)
 125     FORMAT(5X, 'ILLEGAL SPECIFICATION OF STRIKE ANGLES, FATAL')
         STOP
 109  CONTINUE
C
      NX2=NX/2+1
      NY2=NY/2+1
      NX1=NX+1
      NXX=NX+2
      NYY=NY+2
      XY=FLOAT(NX2)/FLOAT(NY2)
C
      ZERO= (0.0,0.0)
      ONE = (1.0,0.0)
      IF (NPASS.NE.-1) GOTO 160
         ZERO=(1.0,0.0)
         ONE =(0.0,0.0)
 160  CONTINUE
C
C ********************************************************************
C      'ZERO' OUT ARRARY
C ********************************************************************
C
      DO 15 IY=1,NY
         DO 15 IX=1,NX
 15         H(IX,IY)=ZERO
C
C ********************************************************************
C        COMPUTE PARAMETERS FOR SOUTHWEST QUADRANT OF MATRIX
C ********************************************************************
C
      IF(ANG2.GE.90.0) GOTO 20
         IYMAX=0
         GOTO 200
C
   20 IF(ANG1.GT.90.0) GOTO 114
         A1=0.0
         TA1=0.0
         IYMAX=NY2
         ADA1=1.5
         GOTO 113
  114 CONTINUE
         A1=ATAN(TAN((ANG1-90.0)*DG2RAD)*XY)
         TA1=TAN(A1)
         IYMAX=FLOAT(NX2)/TA1+1.0
         IYMAX=AMIN0(IYMAX,NY2)
         ADA1=1.0
  113 CONTINUE
C
```

```
      IF(ANG2.LT.180.0) GOTO 115
        A2=DG90
        TA2=0.0
        ADA2=FLOAT(NX2)+0.5
        GOTO 116
 115  CONTINUE
        A2=ATAN(TAN((ANG2-90.0)*DG2RAD)*XY)
        TA2=TAN(A2)
        ADA2=1.0
 116  CONTINUE
C
 200  IF(ANG1.LE.90.0) GOTO 25
        IYMAXX=0
        GOTO 300
C
C **********************************************************************
C       COMPUTE PARAMETERS FOR SOUTHEAST QUADRANT OF MATRIX
C **********************************************************************
C
  25  IF(ANG1.GT.0.0) GOTO 45
        A1=0.0
        TTA1=1.0E20
        ADDA1=FLOAT(NX2)-2.5
        GOTO 60
  45  IF(ANG1.NE.90.0) GOTO 55
        A1=0.0
        A2=0.0
        TTA1=0.0
        TTA2=0.0
        ADDA1= 0.0
        ADDA2= 0.0
        IYMAXX=NY2
        GOTO 122
  55  A1=ATAN(TAN(ANG1*DG2RAD)/XY)
        TTA1= 1.0/TAN(A1)
        ADDA1=1.0
C
  60  IF(ANG2.LT.90.0) GOTO 121
        IYMAXX=NY2
        TTA2=0.0
        ADDA2= 0.0
        GOTO 122
 121  CONTINUE
        A2=ATAN(TAN(ANG2*DG2RAD)/XY)
        TTA2= 1.0/TAN(A2)
        ADDA2=1.0
        IYMAXX= ABS(FLOAT(NX2)/TTA2+1.5)
        IYMAXX=AMIN0(IYMAXX,NY2)
 122  CONTINUE
C
C **********************************************************************
C     CALCULATE THE FILTER COEFFICENTS
C **********************************************************************
C
 300    NYMAX=AMAX0(IYMAX,IYMAXX)
      DO 50 IY=1,NYMAX
        Y=FLOAT(IY-1)
C
C **********************************
C     DEFINE SOUTHWEST QUADRANT
C **********************************
C
        IF(IYMAX.LT.IY) GOTO 30
        MIN=Y*TA1+ADA1
        MAX=Y*TA2+ADA2
        MAX=AMINO(MAX,NX2)
C
        IF(MIN.GT.MAX) GOTO 30
        DO 75 IX=MIN,MAX
 75        H(IX,IY)=ONE
C
C **********************************
C     DEFINE SOUTHEAST QUADRANT
C **********************************
C
 30     IF(IYMAXX.LT.IY) GOTO 35
        MIN=(NX1-(Y*TTA2))+ADDA2
        MIN=AMINO(MIN,NX)
        MAX=(NX1-(Y*TTA1))+ADDA1
        MAX=AMAX0(MAX,NX2+1)
C
        IF(MAX.GT.NX) GO TO 35
        DO 275 IX=MAX,MIN
 275       H(IX,IY)=ONE
C
C ****************************************************
C     USE ANTI-SYMMETRY TO DEFINE QUADRANTS # 2 & 3
C ****************************************************
C
 35     IF(IY.EQ.1 .OR. IY.EQ.NY2) GOTO 50
```

```
         IYY=NYY-IY
         DO 40 IX=2,NX
 40         H(NXX-IX,IYY)=H(IX,IY)
         H(1,IYY)=H(1,IY)
 50   CONTINUE
      H(1,1)=(1.0,0.0)
C
      RETURN
      END
c
c
c
c
      subroutine UPCON (num)
      integer num,npass,imean,nwind,nxout,nyout,udnwind,
     >        row,col
      complex xcdata(512,512),ycdata(512,512)
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      common /comps/ xcdata,ycdata
      common /udcont/ delta,zcon,udxlag,udnwind
      COMMON  H(512,512)
      COMPLEX H
c
      if (num .eq. 1) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      elseif (num .eq. 2) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      endif
C
C *********************************************************************
C
C    "UPCON" PERFORMS UPWARD OR DOWNWARD CONTINUATION ON UNIFORMLY
C    GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.
C
C *********************************************************************
C
C  CREATE CONTINUATION FILTER
C
      CALL CONTIN (DELTA,ZCON,NX,NY)
      CALL STORE(NX,NY)
c
c  smooth the continuation filter
C
      IF(udXLAG.gt.0.0 .and. udXLAG.le.100.0) then
        if (udnwind .le. 4 .and. udnwind .gt. 0) then
           CALL FFT2D (NX,NY,1)
           CALL WINDOW (NX,NY,udXLAG,udNWIND)
           CALL FFT2D (NX,NY,-1)
        endif
      endif
C
C  ACCESS TRANSFORM OF DATA  & MULTIPLE *FILTER (CONVOLVING)
C
      if (num .eq. 1) then
         do 500 i=1,row
            do 500 j=1,col
               xcdata(j,i) = xcdata(j,i)*h(j,i)
 500        continue
      elseif (num .eq. 2) then
         do 580 i=1,row
            do 580 j=1,col
               ycdata(j,i) = ycdata(j,i)*h(j,i)
 580        continue
      endif
c
      return
      end
c
c
c
      SUBROUTINE CONTIN (DELTA,ZCON,NX,NY)
      COMMON H(512,512)
      COMPLEX H
C
C**********************************************************************
C                                                                     *
C   "CONTIN" COMPUTES TWO QUADRANTS (NX/2+1 BY NY) OF AN IDEA         *
C  UPWARD OF DOWNWARD CONTINUATION FILTER DIMENSIONED "NX" BY "NY".   *
C          FOR ONE DIMENSION LET NY=1.                                *
C   ARRAY "H" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM     *
C                                                                     *
C  "DELTA"=  GRID INTERVAL IN LENGTH UNITS                            *
C  "ZCON" =  THE DEPTH OR HEIGHT AT WHICH CONTINUATION IS DESIRED.    *
C            (IN THE SAME LENGTH UNITS AS "DELTA", I.E. MILES,KM)     *
```

```
C     IF "ZCON" IS NEGATIVE FILTER WILL BE UPWARD CONTINUATION         *
C     IF "ZCON" IS POSITIVE FILTER WILL BE DOWNWARD CONTINUATION       *
C                                                                      *
C***********************************************************************
C
      PI2Z= 3.14159265 *ZCON/DELTA
      NXX=NX+2
      NX2=NX/2+1
      RNX2=1.0/FLOAT(NX2)
      NY2=NY/2+1
      RNY2=1.0/FLOAT(NY2)
C
      DO 101 IY=1,NY2
         CON1=(FLOAT(IY-1)*RNY2)**2
         S=SQRT(CON1)
         X1=EXP(S*PI2Z)
         H(1,IY)=CMPLX(X1,0.0)
         DO 101 IX=2,NX2
            S=SQRT((FLOAT(IX-1)*RNX2)**2+CON1)
            X1=EXP(S*PI2Z)
            H(IX,IY)=CMPLX(X1,0.0)
            H(NXX-IX,IY)=H(IX,IY)
  101 CONTINUE
      H(1,1)=(1.0,0.0)
C
      RETURN
      END
c
c
c
c
      subroutine DERIVA (num)
      integer num,npass,imean,nwind,nxout,nyout,
     >        row,col
      complex xcdata(512,512),ycdata(512,512)
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      common /comps/ xcdata,ycdata
      common /xyzderiv/ delta,nth,nway
      COMMON  H(512,512)
      COMPLEX H
c
      if (num .eq. 1) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      elseif (num .eq. 2) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      endif
C
C **********************************************************************
C                                                                      *
C        "DERIVA" CALCULATES THE "NTH" DERIVATIVE IN THE WAVE#         *
C    DOMAIN OF UNIFORMLY GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.     *
C                                                                      *
C **********************************************************************
C
C  CREATE DERIVATIVE FILTER
C
      CALL DERIV(NX,NY,NTH,NWAY,DELTA)
C
C  ACCESS TRANSFORM OF DATA  & MULTIPLE *FILTER (CONVOLVING)
C
      if (num .eq. 1) then
         do 500 i=1,row
            do 500 j=1,col
               xcdata(j,i) = xcdata(j,i)*h(j,i)
 500        continue
      elseif (num .eq. 2) then
         do 580 i=1,row
            do 580 j=1,col
               ycdata(j,i) = ycdata(j,i)*h(j,i)
 580        continue
      endif
c
      return
      end
C
c
      SUBROUTINE DERIV(NNX,NNY,NNTH,NWAY,DELTA)
      COMMON C(512,512)
      COMPLEX C,CON,CON2,CON3
C
C***********************************************************************
C                                                                      *
C     "DERIV" CALCULATES THE VALUES OF 2 QUADRANTS FOR THE             *
C  "NTH" DERIVATIVE OF A (NX,NY) MATRIX. THESE VALUES ARE TO           *
```

```
C  BE MULTIPLIED BY THE WAVE# SPECTRUM OF THE GIVEN FIELD.              *
C     FOR A 1 DIMENSIONAL ARRAY SET "NY"=1                               *
C  ARRAY "C" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM.       *
C                                                                        *
C    "NNTH"  = THE ORDER OF DERIVATIVE DESIRED                           *
C    "NWAY"  = THE DIRECTION THE DERIVATIVE IS TO TAKEN                  *
C               0  VERTICAL DIRECTION                                    *
C               1  HORIZONTAL DIRECTION (X)                              *
C               2  HORIZONTAL DIRECTION (Y)                              *
C    "DELTA" = GRID INTERVAL IN LENGTH UNITS                             *
C                                                                        *
C*************************************************************************
C
      NX=NNX
      NY=NNY
      NTH=NNTH
      NXX=NX+2
      NX2=NX/2+1
      NYY=NY+2
      NY2=NY/2+1
      CON=(6.2831853,0.0)
      IF(NWAY.GE.1) CON=(0.0,6.2831853)
      RNXDEL=1.0/(FLOAT(NX)*DELTA)
      RNYDEL=1.0/(FLOAT(NY)*DELTA)
C
C *******************************************************************
C     TAKE VERTICAL DERIVATIVE IN WAVE# DOMAIN
C *******************************************************************
C
      if (nway .eq. 0) then
         DO 105 IX=2,NX2
            FX2=(FLOAT(IX-1)*RNXDEL)
            C(IX,1)=(CON*FX2)**NTH
  105    C(NXX-IX,1) = (CON*FX2)**NTH
         C(1,1) = (0.0,0.0)
C
         DO 110 IY=2,NY2
            IYY=NYY-IY
            FY2=(FLOAT(IY-1)*RNYDEL)
            C(1,IYY)=(CON*FY2)**NTH
            C(1,IY) =(CON*FY2)**NTH
            FY2=FY2**2
            DO 110 IX=2,NX2
               FX2=(FLOAT(IX-1)*RNXDEL)**2
               S=SQRT(FX2+FY2)
               CON2=(CON*S)**NTH
               C(NXX-IX,IYY) = CON2
               C(IX,IYY)  = CON2
               C(NXX-IX,IY) = CON2
  110    C(IX,IY) = CON2
         RETURN
C
C *******************************************************************
C     TAKE HORIZONTAL(Y) DERIVATIVE IN WAVE# DOMAIN
C *******************************************************************
C
      elseif (nway .eq. 2) then
         DO 205 IX=1,NX
  205    C(IX,1) = (0.0,0.0)
C
         DO 210 IY=2,NY2
            IYY=NYY-IY
            CON2=(FLOAT(IY-1)*RNYDEL*CON)**NTH
            CON3=CONJG(CON2)
            DO 210 IX=1,NX
               C(IX,IYY) = CON3
  210    C(IX,IY)  = CON2
         RETURN
C
C *******************************************************************
C     TAKE HORIZONTAL(X) DERIVATIVE IN WAVE# DOMAIN
C *******************************************************************
C
      elseif (nway .eq. 1) then
         DO 305 IY=1,NY2
            C(1,NYY-IY) = (0.0,0.0)
  305    C(1,IY) = (0.0,0.0)
C
         DO 310 IX=2,NX2
            IXX=NXX-IX
            CON2=(FLOAT(IX-1)*RNXDEL*CON)**NTH
            CON3=CONJG(CON2)
            C(IXX,1) =  CON3
            C(IX,1)  =  CON2
            DO 310 IY=2,NY2
               C(IXX,NYY-IY)=CON3
               C(IXX,IY) =CON3
               C(IX,NYY-IY)=CON2
  310    C(IX,IY)  =CON2
         RETURN
```

```
      else
         write (*,*) 'nway is not equal to 0,1 or 2'
         stop
      endif
c
      end
c
c
c
c
      subroutine MAG2POL (num)
      integer num,npass,imean,nwind,nxout,nyout,
     >        row,col
      complex xcdata(512,512),ycdata(512,512)
      common /fftifft/ nxout,nyout,prcnt,imean,fold,itypefold
      common /comps/ xcdata,ycdata
      common /rtp/ azm,xinc,dec
      COMMON  H(512,512)
      COMPLEX H
c
      if (num .eq. 1) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      elseif (num .eq. 2) then
         row=nyout
         col=nxout
         nx=nxout
         ny=nyout
      endif
C
C ********************************************************************
C
C        "MAG2POL" APPROXIMATLY CALCULATES THE CORRESPONDING MAGNETIC  *
C     ANOMALY MAP DUE TO AN EARTH'S FIELD VECTOR OF 0.0 DECLINATION,   *
C     AND 90.0 INCLINATION FROM AN INPUTTED MAGNETIC ANOMALY MAP WITH  *
C     A KNOWN FIELD VECTOR.
C
C ********************************************************************
c
      DEC=DEC+AZM
      CALL MAGPOL(XINC,DEC,NX,NY)
C
C  ACCESS TRANSFORM OF DATA  & MULTIPLE *FILTER (CONVOLVING)
C
      if (num .eq. 1) then
         do 500 i=1,row
            do 500 j=1,col
               xcdata(j,i) = xcdata(j,i)*h(j,i)
 500        continue
      elseif (num .eq. 2) then
         do 580 i=1,row
            do 580 j=1,col
               ycdata(j,i) = ycdata(j,i)*h(j,i)
 580        continue
      endif
c
      return
      end
c
c
C
C
      SUBROUTINE MAGPOL(AINC,DDEC,NX,NY)
      COMMON X(512,512)
      COMPLEX X,CONA,CONB
C
C*********************************************************************
C                                                                    *
C       "MAGPOL" CREATES A WAVE# REDUCTION-TO-MAGNETIC-POLE FILTER   *
C    ONTO THE ARRAY "X" IN BLANK COMMON                              *
C    THE DIMENSIONS OF THE ARRAY "X" MUST BE IDENTICAL TO THAT IN    *
C    THE MAIN PROGRAM.                                               *
C                                                                    *
C    "AINC"    THE AVERAGE MAGNETIC INCLINATION OF THE AREA IN DEGREES. *
C    "DDEC"    THE AVERAGE MAGNETIC DECLINATION OF THE AREA IN DEGREES. *
C                                                                    *
C*********************************************************************
C
      RR=3.14159265/180.0
      NX2=NX/2+1
      NX1=NX/2
      NXX=NX+2
      NY2=NY/2+1
      NYY=NY+2
      RNX=1.0/FLOAT(NX)
      RNY=1.0/FLOAT(NY)
C
      SINI=SIN(AINC*RR)
```

```
      COSI=COS(AINC*RR)
      SIND=SIN(DDEC*RR)
      COSD=COS(DDEC*RR)
C
      CON3=COSI*COSD
      CON2=COSI*SIND
C
      CONA=1.0/CMPLX(SINI,CON2)**2
      CONB=CONJG(CONA)
      X(1,1)=(0.0,0.0)
      DO 30 IX=2,NX1
         X(IX,1)=CONA
   30 X(NXX-IX,1)=CONB
      X(NX2,1)=CONA
C
      CONA=1.0/CMPLX(SINI,CON3)**2
      CONB=CONJG(CONA)
      DO 50 IY=2,NY2
         IYY=NYY-IY
         FY=FLOAT(IY-1)*RNY
         FY2=FY*FY
         CFY=CON3*FY
         X(1,IYY)=CONB
         X(1,IY) =CONA
         DO 40 IX=2,NX1
            IXX=NXX-IX
            FX=FLOAT(IX-1)*RNX
            CON4=SQRT(FX**2+FY2)
            X(IXX,IY)=1.0/CMPLX(SINI,(CFY-FX*CON2)/CON4)**2
            X(IX,IYY)=CONJG(X(IXX,IY))
            X(IX,IY)  =1.0/CMPLX(SINI,(FX*CON2+CFY)/CON4)**2
   40    X(IXX,IYY)=CONJG(X(IX,IY))
         CON4=SQRT(0.25+FY2)
         X(NX2,IYY)=1.0/CMPLX(SINI,(0.5*CON2-CFY)/CON4)**2
   50 X(NX2,IY) =1.0/CMPLX(SINI,(0.5*CON2+CFY)/CON4)**2
C
      RETURN
      end
```

```
      program avgdifres
      character*80 filename
      integer coln,rown,colk,rowk,var,row,col,set
      dimension  dndata(550,550),dkdata(550,550),avgdata(550,550),
     >           difdata(550,550),ddata(550,550)
c
c---------------------------------------- program description
c       avgdifres is a bashing of matrix manipulation.  the program
c       will find the average of two input data sets, the difference
c       between them, and will resample (hence avg-dif-res  aint science
c       great??) every other point, every third, fourth etc.. point.
c       if you want the average and difference of the input matrices
c       make sure you choose to resample every data point, ie choose
c       1 for 1x1.
c
      write (*,*) '1 FOR ONE DATA SET'
      write (*,*) '2 FOR TWO DATA SETS'
      read (*,*) set
c
      write (*,*) 'INPUT FILE OF DAWN GRIDDED ANOMALIES OR ONE DATA SET'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename,status='old',form='formatted')
      if (set .eq. 1) go to 10
      write (*,*) 'INPUT FILE OF DUSK GRIDDED ANOMALIES'
      read (*,9990) filename
      open (12, file=filename,status='old',form='formatted')
 10   write (*,*) 'OUTPUT AVERAGED AND/OR RESAMPLED GRIDDED FILE'
      read (*,9990) filename
      open (20, file=filename,form='formatted')
      if (set .eq. 1) go to 20
      write (*,*) 'OUTPUT DIFFERENCE GRID OF (DUSK) - (DAWN)'
      read (*,9990) filename
      open (22, file=filename,form='formatted')
c
 20   continue
      write (*,*) 'RESAMPLE BY THIS NUMBER  USE:'
      write (*,*) '1  for keeping the average grid as is'
      write (*,*) '2  for  2 degrees by 2 degrees'
      write (*,*) '3  for  3 degrees by 3 degrees   and so on'
      read (*,*) var
c
      read (10,*) coln
      read (10,*) rown
      read (10,*) xcolat
      read (10,*) xlong
      read (10,*) xgridspc
 40   do 50 i=1,rown
         read (10,*) (dndata(i,j),j=1,coln)
 50      continue
      if (set .eq. 1) go to 170
      read (12,*) colk
      read (12,*) rowk
      read (12,*) ycolat
      read (12,*) ylong
      read (12,*) ygridspc
 80   do 100 i=1,rowk
         read (12,*) (dkdata(i,j),j=1,colk)
 100  continue
c
      if (coln.ne.colk .or. rown.ne.rowk) then
         write (*,*) 'HEY NOW KIDS GRIDS ARE DIFFERENT SIZES'
         write (*,*) coln,rown,colk,rowk
         go to 999
      endif
c
c--------------------- find the average and difference matrices also
c                      calculate the total and average RMS difference.
      totrms=0.0
      do 150 i=1,rown
         do 150 j=1,coln
            avgdata(i,j)=((dndata(i,j)+dkdata(i,j))/2.0)
            difdata(i,j)=dkdata(i,j)-dndata(i,j)
            totrms=totrms+((sqrt((dndata(i,j)-dkdata(i,j))**2))/2.0)
 150     continue
      avgrms=totrms/(real(rown)*real(coln))
c
 170  if (set .eq. 1) then
         do 180 i=1,rown
            do 180 j=1,coln
               avgdata(i,j)=dndata(i,j)
 180        continue
      endif
c
c---------------------------- this is the section that resamples
      row=1
      ii=1
 200  jj=1
      col=1
 250  ddata(ii,jj)=avgdata(row,col)
```


PRECEDING PAGE BLANK NOT FILMED

```
      col=col+var
      if (col .gt. coln) go to 300
      jj=jj+1
      go to 250
 300  row=row+var
      if (row .gt. rown) go to 400
      ii=ii+1
      go to 200
 400  continue
c
      write (*,*) 'new rows=',ii,' new cols=',jj
      write (20,*) jj
      write (20,*) ii
      write (20,*) xcolat
      write (20,*) xlong
      write (20,*) real(var)
      do 450 k=1,ii
         write (20,9992) (ddata(k,l),l=1,jj)
 450     continue
 9992 format (6(f11.4,1x))
c
      if (set .eq. 1) go to 999
      write (22,*) coln
      write (22,*) rown
      write (22,*) ycolat
      write (22,*) ylong
      write (22,*) ygridspc
      do 500 m=1,rown
         write (22,9992) (difdata(m,n),n=1,coln)
 500     continue
      write (*,*) 'total rms difference=',totrms
      write (*,*) 'average rms difference=',avgrms
c
 999  continue
      close (10)
      close (12)
      close (20)
      close (22)
      stop
      end
```

```
      program sqrmap
      character*80 filename
      dimension xcore(100000),xxmag(100000),
     >          core(361,361),xmag(361,361),xmultmag(400)
c
c---------------------------- program description
c      this program will find a least squares value of x that
c      can be multiplied by the difference matrix so that the
c      difference closer fits the dawn or dusk matrix.  the
c      value of x is multiplied by the difference and subtracted
c      from the dawn or dusk grid to make an output grid.
c      NOTE: i realize i'm making the 2-D arrays
c            into 1-D arrays
c
      write (*,*) 'input difference matrix'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename,form='formatted',status='old')
      write (*,*) 'input dawn or dusk matrix of total field values'
      read (*,9990) filename
      open (11, file=filename,form='formatted',status='old')
      write (*,*) 'output (total field) - (x)(difference)'
      read (*,9990) filename
      open (20, file=filename,form='formatted')
c
      cmean=0.0
      fmean=0.0
      read (10,*) icol
      read (10,*) irow
      read (10,*) south
      read (10,*) west
      read (10,*) gridspc
      do i=1,irow
         read (10,*) (core(i,j),j=1,icol)
         do j=1,icol
            cmean=cmean+core(i,j)
         enddo
      enddo
c
      read (11,*) imcol
      read (11,*) imrow
      read (11,*) south
      read (11,*) west
      read (11,*) gridspc
      do i=1,imrow
         read (11,*) (xmag(i,j),j=1,imcol)
         do j=1,imcol
            fmean=fmean+xmag(i,j)
         enddo
      enddo
      if (irow.ne.imrow .or. icol.ne.imcol ) then
         write (*,*) 'rows cor mag ',irow,imrow
         write (*,*) 'cols cor mag ',icol,imcol
         stop 10
      endif
      cmean=cmean/real(irow*icol)
      fmean=fmean/real(irow*icol)
c
c--------------------------- remove the mean values
      do 500 i=1,irow
         do 500 j=1,icol
            xmag(i,j)=xmag(i,j)-fmean
 500        core(i,j)=core(i,j)-cmean
c
c--------------------------- turn the 2-D arrays into 1-D (cheater)
      do i=1,irow
         do j=1,icol
            ii=((i-1)*icol)+j
            xxmag(ii)=xmag(i,j)
            xcore(ii)=core(i,j)
         enddo
      enddo
c---------------------------- find a scalar value of c transpose c
      ctc=0.0
      do 600 i=1,ii
 600     ctc=(xcore(i)*xcore(i))+ctc
c
c--------------------------- find (c transpose c) inverse
      ctcinv=1.0/ctc
c
      ctf=0.0
      do 700 i=1,ii
 700     ctf=(xcore(i)*xxmag(i))+ctf
c---------------------------- find x
      x=ctcinv*ctf
      write (*,*) 'the value of x = ', x
c
      write (20,*) imcol
      write (20,*) imrow
      write (20,*) south
```

```
      write (20,*) west
      write (20,*) gridspc
      do i=1,irow
         do j=1,icol
            xmultmag(j)=xmag(i,j)-(x)*(core(i,j))
         enddo
         write (20,9991) (xmultmag(j),j=1,icol)
      enddo
 9991 format (6(f11.4,1x))
c
 999  continue
      stop
      end
```

```
      program inversion
      DIMENSION DW(3982),DP(3982),DFS(361,44)
      COMMON XORD,YORD,DXD,DYD,NXD,NYD,RO,RD,E,G,COST,SINT,YC,THETA,PHI,
     >       P(3982),TMAG(3982),DF(361,44),S(7930153)
      EQUIVALENCE (DW(1),P(1)),  (DP(1),TMAG(1))
      REAL I,INC(361),D,DEC(361),FLD(361),I1,D1,F1,myear,mi,md,
     >     mi1,md1,mf1,mxoro,myoro,mdxo,mdyo,melvo
      real fldd,fldo,incd,inco,decd,deco
      dimension fldd(44,181),fldo(44,361),incd(44,181),inco(44,361),
     >          decd(44,181),deco(44,361)
      common /gmf/ fldd,fldo,incd,inco,decd,deco
      character*80 filename
      integer choice
c
C***********************************************************************
c
c     PROGRAM NVERTSM CALCULATES A SET OF CGS-SUSCEPTIBILITIES FOR AN
c     NXD-BY-NYD SPHERICAL ARRAY OF POINT DIPOLES SUCH THAT THE RESUL-
c     TANT EQUIVALENT SOURCE FIELD IS A LEAST-SQUARES BEST FIT TO MAG-
c     NETIC DATA OBSERVED OVER AN NXO-BY-NYO SPHERICAL GRID.  OUTPUT
c     CONSISTS OF LISTINGS AND/OR PUNCHED DECKS OF EQUIVALENT SOURCE
c     SUSCEPTIBILITIES AND/OR ANOMALY VALUES (SEE DATA CARD 1 BELOW).
c
c     DIMENSIONING REQUIREMENTS.....
c     DIMENSION DW(NXD*NYD),DP(NXD*NYD),P(NXD*NYD),TMAG(NXD*NYD),
c     DF(NXO,NYO),S(NIJ) WHERE NIJ=(NXD*NYD)*(NXD*NYD+1)/2, FLD1(NXD),
c     INC1(NXD),D1(NXD),FLD(NXO),I(NXO),D(NXO)
c
c     this program has been slightly modified from a view point of
c     lines of code.  these changes are all lower case.  however from
c     the view point of run time it should now take less than 1/4 !!!
c     of the time that it took in the past.  this is because no reads
c     from files 10 or 11 are necessary as all arrays are stored in
c     memory.
c
c     modifications made  15 may 90
c
c     further changes on 22 Sep 90
c          these changes are mostly removal of unnecessary write
c          statements and general cleanup of the program.
c
c     more changes on 2 Jan 91
c          this update included the addition of a few lines of code
c          that allows for input of a file of susceptibilities and
c          output of a magnetic field map.
c
C******************  DATA INPUT SEQUENCE  *****************************
c
      write (*,*) '0 IF YOU HAVE A FILE FOR THE VARIABLES'
      write (*,*) '1 IF YOU WANT TO TYPE VARIABLES INTERACTIVELY'
      read (*,*) choice
      if (choice .eq. 0) go to 10
      write (*,9991)
 9991 format('NFLD=0 DO NOT CALCULATE EQUIVALENT SOURCE M-FIELD'/
     >       '     =1 CALCULATE EQUIVALENT SOURCE M-FIELD'/
     >       '     =2 GIVEN THE CGS-SUSCEPTIBILITIES THE PROGRAM'/
     >       '        WILL CALCULATE THE EQUIV SRC M-FLD'/
     >       ' NIO=0 DO NOT WRITE A FILE SUSCEPTIBILITIES'/
     >       '     =1 WRITE OUT A SEPARATE FILE OF SUSCEPTIBILITIES'/
     >       ' NFLD(1)  NIO(0)')
      read (*,*) nfld,nio
c
      if (nfld .eq. 1) then
         write (*,*) ' '
         write (*,*) 'the following refers to calculation of the',
     >               ' susceptibilities'
         write (*,*) ' '
      endif
      write (*,9992)
 9992 format('NXO= NUMBER OF LONGITUDINAL COLS OF OBSERVATION GRID'/
     >'NYO= NUMBER OF LATITUDINAL ROWS  OF OBSERVATION GRID'/
     >'XORO= WESTERN-MOST LONGITUDINAL COORDINATE OF OBSERVATION'/
     >'      GRID IN -180.0 to 180.0 DEGREES'/
     >'YORO= SOUTHERN-MOST LATITUDINAL COORDINATE OF OBSERVATION'/
     >'      GRID IN -90.0 to 90.0 DEGREES'/
     >'DXO= LONGITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/
     >'DYO= LATITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/
     >'ELVO= ELEVATION OF OBSERVATION GRID IN KILOMETERS (350.0)'//
     >'NXO  NYO  XORO  YORO  DXO  DYO  ELVO ')
      read (*,*) nxo,nyo,xoro,yoro,dxo,dyo,elvo
c
      write (*,9993)
 9993 format ('NXD= NUMBER OF LONGITUDINAL COLS OF SOURCE GRID'/
     >'NYD= NUMBER OF LATITUDINAL ROWS  OF SOURCE GRID'/
     >'XORD= WESTERN-MOST LONGITUDINAL COORDINATE OF SOURCE'/
     >'      GRID IN -180.0 to 180.0 DEGREES'/
     >'YORD= SOUTHERN-MOST LATITUDINAL COORDINATE OF SOURCE'/
     >'      GRID IN -90.0 to 90.0 DEGREES'/
     >'DXD= LONGITUDINAL STATION SPACING OF SOURCE GRID IN DEGREES'/
     >'DYD= LATITUDINAL STATION SPACING OF SOURCE GRID IN DEGREES'/
```

```
     >'ELVD- ELEVATION OF SOURCE GRID IN KILOMETERS (-50.0)'//
     >'NXD  NYD  XORD  YORD  DXD  DYD  ELVD ')
      read (*,*) nxd,nyd,xord,yord,dxd,dyd,elvd
c
      write (*,9994)
 9994 format ('YEAR- EPOCH IN YEARS AND DECIMAL FRACTION YEARS'/
     >'       E.G., 1965.75 = 1 OCT 65  FOR WHICH THE'/
     >'       GEOMAGNETIC REFERENCE FIELD IS TO'/
     >'       BE COMPUTED AT  SOURCE AND/OR OBSERVATION POINTS'/
     >'     = 0 USER SUPPLIES CHARACTERISTICS OF SOURCE POLARIZATION'/
     >'       FIELD (F1,I1,D1) AND GEOMAGNETIC FIELD (I,D) OVER'/
     >'       OBSERVATION GRID'/
     >'F1- SCALAR MAGNETIC INTENSITY IN GAMMAS OF SOURCE POLARIZATION'/
     >'    FIELD.'/
     >'I1- INCLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
     >'D1- DECLINATION IN DEGREES OF SOURCE POLARIZING FIELD'//
     >'NOTE---IF (F1+I1+D1).EQ.0.0, THEN THE SOURCE POLARIZING'/
     >'        FIELD IS COMPUTED BY SUBROUTINE FIELDG FOR EPOCH'/
     >'        SPECIFIED BY THE YEAR-INPUT VARIABLE'//
     >' YEAR (1980.0)  F1  (0.0)  I1  (0.0)  D1  (0.0)')
      read (*,*) year,f1,i1,d1
c
      write (*,9995)
 9995 format ('I- INCLINATION IN DEGREES OF THE GEOMAGNETIC FIELD'/
     >'   OVER THE OBSERVATION POINTS'/
     >'D- DECLINATION IN DEGREES OF THE GEOMAGNETIC FIELD OVER THE'/
     >'   OBSERVATION POINTS'//
     >'NOTE---IF (I+D).EQ.0.0, THEN THE GEOMAGNETIC FIELD OVER THE'/
     >'       OBSERVATION POINT IS COMPUTED BY SUBROUTINE FIELDG'/
     >'       FOR THE EPOCH SPECIFIED BY THE YEAR-INPUT VARIABLE'//
     >'I (0.0)    D (0.0)')
      read (*,*) i,d
      write (*,*) 'ERROR FACTOR FOR VARIANCE (fak)(0.10E-7)'
      read (*,*) fak
c
      if (nfld .eq. 0) go to 15
      write (*,*) ' '
      write (*,*) 'the following refers to calculation of the'
      write (*,*) 'equivalent source magnetic field'
      write (*,*) ' '
c
      write (*,9998)
 9998 format('mNXO- NUMBER OF LONGITUDINAL COLS OF OBSERVATION GRID'/
     >'mNYO- NUMBER OF LATITUDINAL ROWS  OF OBSERVATION GRID'/
     >'mXORO- WESTERN-MOST LONGITUDINAL COORDINATE OF OBSERVATION'/
     >'       GRID IN -180.0 to 180.0 DEGREES'/
     >'mYORO- SOUTHERN-MOST LATITUDINAL COORDINATE OF OBSERVATION'/
     >'       GRID IN -90.0 to 90.0 DEGREES'/
     >'mDXO- LONGITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/
     >'mDYO- LATITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/
     >'mELVO- ELEVATION OF OBSERVATION GRID IN KILOMETERS (350.0)'//
     >'mNXO  mNYO  mXORO  mYORO  mDXO  mDYO  mELVO ')
      read (*,*) mnxo,mnyo,mxoro,myoro,mdxo,mdyo,melvo
c
      write (*,9996)
 9996 format ('mYEAR- EPOCH IN YEARS AND DECIMAL FRACTION YEARS'/
     >'       E.G., 1965.75 = 1 OCT 65  FOR WHICH THE'/
     >'       GEOMAGNETIC REFERENCE FIELD IS TO'/
     >'       BE COMPUTED AT  SOURCE AND/OR OBSERVATION POINTS'/
     >'     = 0 USER SUPPLIES CHARACTERISTICS OF SOURCE POLARIZATION'/
     >'       FIELD (F1,I1,D1) AND GEOMAGNETIC FIELD (I,D) OVER'/
     >'       OBSERVATION GRID'/
     >'mF1- SCALAR MAGNETIC INTENSITY IN GAMMAS OF SOURCE POLARIZATION'/
     >'    FIELD.'/
     >'mI1- INCLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
     >'mD1- DECLINATION IN DEGREES OF SOURCE POLARIZING FIELD'//
     >'NOTE---IF (mF1+mI1+mD1).EQ.0.0, THEN THE SOURCE POLARIZING'/
     >'        FIELD IS COMPUTED BY SUBROUTINE FIELDG FOR EPOCH'/
     >'        SPECIFIED BY THE YEAR-INPUT VARIABLE'//
     >'mYEAR (0.0)  mF1  (60000.0)  mI1  (90.0)  mD1  (0.0)')
      read (*,*) myear,mf1,mi1,md1
c
      write (*,9997)
 9997 format ('mI- INCLINATION IN DEGREES OF THE GEOMAGNETIC FIELD'/
     >'   OVER THE OBSERVATION POINTS'/
     >'mD- DECLINATION IN DEGREES OF THE GEOMAGNETIC FIELD OVER THE'/
     >'   OBSERVATION POINTS'//
     >'NOTE---IF (mI+mD).EQ.0.0, THEN THE GEOMAGNETIC FIELD OVER THE'/
     >'       OBSERVATION POINT IS COMPUTED BY SUBROUTINE FIELDG'/
     >'       FOR THE EPOCH SPECIFIED BY THE YEAR-INPUT VARIABLE'//
     >'mI (90.0)    mD (0.0)')
      read (*,*) mi,md
      write (*,*) '0 DO NOT SUBTRACT MEAN OF FINAL R-T-P MAP'
      write (*,*) '1 SUBTRACT THE MEAN'
      read (*,*) isub
      go to 15
c
 10   write (*,*) 'INPUT FILE OF NUMBERS FOR VARIABLES'
      read (*,9990) filename
      open (17, file=filename,status='old',form='formatted')
```

```
      READ (17,*) NFLD,NIO
      READ (17,*) NXO,NYO,XORO,YORO,DXO,DYO,ELVO
      READ (17,*) NXD,NYD,XORD,YORD,DXD,DYD,ELVD
      READ (17,*) YEAR,F1,I1,D1,I,D
      read (17,*) fak
      read (17,*) mnxo,mnyo,mxoro,myoro,mdxo,mdyo,melvo
      read (17,*) myear,mf1,mi1,md1,mi,md
      read (17,*) isub
c
 15   write (*,*) 'INPUT FILE OF GRIDDED ANOMALY DATA OR SUSC DATA'
      write (*,*) 'DATA SHOULD BE WEST TO EAST AND SOUTH TO NORTH'
c-------------------------- array df(ix,jy) reads the anomaly data
c                            in gammas or nanoteslas with the input
c                            starting with the southernmost latitude
c                            at the westernmost longitude.
      read (*,9990) filename
 9990 format (a80)
      open (13, file=filename,status='old',form='formatted')
      write (*,*) 'INPUT FILE OF SPHERICAL HARMONIC COEFFICIENTS'
      write (*,*) 'SUCH AS  GSFC1283'
      read (*,9990) filename
      open (3, file=filename,status='old',form='formatted')
c      open (3, file='../../data/mgst1283',status='old',
c     >                                          form='formatted')
      if (nfld .eq. 1  .or.  nfld .eq. 2) then
         write (*,*) 'OUTPUT FILE OF EQUIVALENT SOURCE M-FIELD'
         read (*,9990) filename
         open (30, file=filename,form='formatted')
      endif
      if (nio .eq. 1) then
         write (*,*) 'OUTPUT FILE OF SUSCEPTIBILITIES'
         read (*,9990) filename
         open (33, file=filename,form='formatted')
      endif
      write (*,*) 'OUTPUT INFORMATION FILE OF A BUNCH OF STUFF!!'
      read (*,9990) filename
      open (6, file=filename,form='formatted')
c
      if (nfld .eq. 2) go to 155
c
      WRITE(6,520) NXO,NYO,XORO,YORO,DXO,DYO,ELVO
      WRITE(6,540) NXD,NYD,XORD,YORD,DXD,DYD,ELVD
c
      IF (YEAR.LT.1.E-9) WRITE(6,420) F1,I1,D1,I,D
      IF (F1+I1+D1.LT.1.E-9.AND.I+D.GT.0.0) WRITE(6,430) YEAR,I,D
      IF (I+D.LT.1.E-9.AND.F1+I1+D1.GT.0.0) WRITE(6,440) YEAR,F1,I1,D1
      IF (I+D.LT.1.E-9.AND.F1+I1+D1.LT.1.E-9) WRITE(6,450) YEAR
C
C     CALCULATE THE GEOMAGNETIC FIELD OVER THE SOURCE AND/OR OBSERVATION
C     GRIDS, RESPECTIVELY
C
      IF (YEAR.LT.1.E-9) GO TO 120
      LQ=1
C
      CALL FIELDG (0.,0.,0.,0.,55,LQ,Q1,Q2,Q3,Q4)
C
      IF (F1+I1+D1.GT.0.0) GO TO 110
C
      CALL GEOMAG (YEAR,ELVD,YORD,XORD,DYD,DXD,NYD,NXD,11)
C
  110 IF (I+D.GT.0.0) GO TO 120
C
      CALL GEOMAG (YEAR,ELVO,YORO,XORO,DYO,DXO,NYO,NXO,10)
C
  120 CONTINUE
C
      read (13,*) iew
      read (13,*) ins
      read (13,*) scolat
      read (13,*) west
      read (13,*) gridspc
      DO 130 JY=1,NYO
         READ (13,*) (DF(IX,JY),IX=1,NXO)
  130 continue
C
C     COMPUTE MAXIMUM, MINIMUM AND AVERAGE AMPLITUDE VALUES FOR M-FIELD
C     INPUT DATA
C
      DSUM=0.0
      DMIN=DF(1,1)
      DMAX=DMIN
      DO 140 JY=1,NYO
      DO 140 IX=1,NXO
         DSUM=DSUM+DF(IX,JY)
         IF (DMAX.LT.DF(IX,JY)) DMAX=DF(IX,JY)
         IF (DMIN.GT.DF(IX,JY)) DMIN=DF(IX,JY)
  140 CONTINUE
      DSUM=DSUM/FLOAT(NXO*NYO)
      WRITE(6,640) DMAX,DMIN,DSUM
C
```

```
c     save values for the reduction-to-pole
c
 155  istrnxd=nxd
      istrnyd=nyd
      strxord=xord
      stryord=yord
      strdxd=dxd
      strdyd=dyd
      strelvd=elvd
      if (nfld .eq. 2) go to 840
c
C     CONVERT LAT AND LONG TO RADIANS AND ELEVATIONS TO EARTH RADII
C
      PI=3.1415926536
      FACT=PI/180.0
C
      XORO=XORO*FACT
      XORD=XORD*FACT
      YORO=(90.0-YORO)*FACT
      YORD=(90.0-YORD)*FACT
      DXO=DXO*FACT
      DXD=DXD*FACT
      DYO=-DYO*FACT
      DYD=-DYD*FACT
C
      I=I*FACT
      D=-D*FACT
      I1=I1*FACT
      D1=-D1*FACT
C
      REARTH=6371.0
      RO=ELVO+REARTH
      RD=ELVD+REARTH
      E=RD**2+RO**2
      G=2.0*RD*RO
      NP=(NXD*NYD)
      DO 150 JY=1,NP
         P(JY)=0.0
         DW(JY)=0.0
  150 CONTINUE
C
      NIJ=NP*(NP+1)/2
      DO 160 JY=1,NIJ
  160 S(JY)=0.0
C
C     COMPUTE WEIGHTING COEFFICIENTS, DW(L), L=1,NP
C
C     COMPUTE UPPER HALF OF SYMMETRIC A(TRANSPOSE)A MATRIX OF 1-ST ORDER
C     PARTIALS AND STORE AS 1-DIMENSIONAL ARRAY S(LLL), LLL=1,NIJ---I.E.
C     A(TRANSPOSE)A(I,J)=S(IJ), WHERE IJ=I*(I-1)/2+J
C
      IF (YEAR.LT.1.E-9) GO TO 180
      IF (F1+I1+D1.GT.0.0.AND.I+D.LT.1.E-9) GO TO 200
      IF (F1+I1+D1.LT.1.E-9.AND.I+D.GT.0.0) GO TO 220
C
C     COMPUTE A(TRANSPOSE)A MATRIX FOR CASE WHERE (I,D) AND (F1,I1,D1)
C     ARE DERIVED FROM FIELDG
C
      DO 170 JY=1,NYO
         THETA=YORO+(FLOAT(JY)-1.0)*DYO
         COST=COS(THETA)
         SINT=SIN(THETA)
C
         do 175 l=1,nxo
            fld(l)=fldo(jy,l)
            inc(l)=inco(jy,l)
            dec(l)=deco(jy,l)
 175        continue
      DO 170 IX=1,NXO
         PHI=XORO+(FLOAT(IX)-1.0)*DXO
         AI=INC(IX)
         AD=DEC(IX)
C
         CALL MAGS1 (AI,AD)
C
         DY=DF(IX,JY)-YC
         LLL=0
      DO 170 L=1,NP
         DW(L)=DW(L)+TMAG(L)*DY
      DO 170 K=1,L
         LLL=LLL+1
         S(LLL)=S(LLL)+TMAG(L)*TMAG(K)
  170 CONTINUE
      GO TO 240
C
C     COMPUTE A(TRANSPOSE)A MATRIX FOR CASE WHERE USER SUPPLIES (I,D)
C     AND (F1,I1,D1)
C
  180 DO 190 JY=1,NYO
         THETA=YORO+(FLOAT(JY)-1.0)*DYO
```

```
         COST=COS(THETA)
         SINT=SIN(THETA)
      DO 190 IX=1,NXO
         PHI=XORO+(FLOAT(IX)-1.0)*DXO
C
         CALL MAGS2 (I,D,F1,I1,D1)
C
         DY=DF(IX,JY)-YC
C
         LLL=0
      DO 190 L=1,NP
         DW(L)=DW(L)+TMAG(L)*DY
      DO 190 K=1,L
         LLL=LLL+1
         S(LLL)=S(LLL)+TMAG(L)*TMAG(K)
  190 CONTINUE
      GO TO 240
C
C     COMPUTE A(TRANSPOSE)A MATRIX FOR CASE WHERE (I,D) ARE DERIVED
C     FROM FIELDG AND USER SUPPLIES (F1,I1,D1)
C
  200 DO 210 JY=1,NYO
         THETA=YORO+(FLOAT(JY)-1.0)*DYO
         COST=COS(THETA)
         SINT=SIN(THETA)
C
         do 185 l=1,nxo
            fld(l)=fldo(jy,l)
            inc(l)=inco(jy,l)
            dec(l)=deco(jy,l)
 185        continue
c
      DO 210 IX=1,NXO
         PHI=XORO+(FLOAT(IX)-1.0)*DXO
         AI=INC(IX)
         AD=DEC(IX)
C
         CALL MAGS2 (AI,AD,F1,I1,D1)
C
         DY=DF(IX,JY)-YC
C
         LLL=0
      DO 210 L=1,NP
         DW(L)=DW(L)+TMAG(L)*DY
      DO 210 K=1,L
         LLL=LLL+1
         S(LLL)=S(LLL)+TMAG(L)*TMAG(K)
  210 CONTINUE
      GO TO 240
C
C     COMPUTE A(TRANSPOSE)A MATRIX FOR CASE WHERE USER SUPPLIES (I,D)
C     AND (F1,I1,D1) ARE DERIVED FROM FIELDG
C
  220 DO 230 JY=1,NYO
         THETA=YORO+(FLOAT(JY)-1.0)*DYO
         COST=COS(THETA)
         SINT=SIN(THETA)
      DO 230 IX=1,NXO
         PHI=XORO+(FLOAT(IX)-1.0)*DXO
C
         CALL MAGS1 (I,D)
C
         DY=DF(IX,JY)-YC
         LLL=0
      DO 230 L=1,NP
         DW(L)=DW(L)+TMAG(L)*DY
      DO 230 K=1,L
         LLL=LLL+1
         S(LLL)=S(LLL)+TMAG(L)*TMAG(K)
  230 CONTINUE
  240 CONTINUE
C-------------------------------- if a transpose a without error
c                                  variance is wanted then use
c                                  write (20).
c     WRITE(20)  (S(J),J=1,LLL)
c-------------------------------- if a transpose observations without
c                                  error variance is wanted then use
c                                  write (25).
c     WRITE(25)  (DW(L),L=1,NP)
c
c-----------------  the following loop adds fak to the diagonal
c                   of aTa
      II=0
      DO 888 J=1,NP
         II=II+J
         S(II)=S(II)+FAK
  888 CONTINUE
C
C     COMPUTE INVERSE OF S-ARRAY
C
```

```
      CALL SPPCO (S,NP,RCOND,DP,INFO)
C
      WRITE(6,480) RCOND
      IF (INFO.NE.0) WRITE(6,490) INFO
      DO 250 IX=1,NP
  250 DP(IX)=DW(IX)
C
      CALL SPPSL (S,NP,DP)
C
C     WRITE OUT COORDINATE CHARACTERISTICS OF M-DIPOLES
C
      WRITE(6,590)
      L=0
      DO 260 JY=1,NYD
         YLAT=90.0-(YORD+FLOAT(JY-1)*DYD)/FACT
      DO 260 IX=1,NXD
         L=L+1
         XLON=(XORD+FLOAT(IX-1)*DXD)/FACT
         WRITE(6,600) DP(L),XLON,YLAT,ELVD
  260 CONTINUE
      DO 270 JY=1,NP
  270 P(JY)=DP(JY)
C
C  IF NIO = 1 WRITE SUSCEPTIBILITIES ONTO USER DEFINED UNIT 33
C
      if (nio .eq. 1) then
         write (33,*) iew
         write (33,*) ins
         write (33,*) scolat
         write (33,*) west
         write (33,*) gridspc
         IJK=1
         DO 375 JY=1,NYD
            IJK1=IJK+NXD-1
            WRITE(33,'(4(E18.8,1x))') (DP(L),L=IJK,IJK1)
            IJK=IJK1+1
375      CONTINUE
      endif
C
      IF (NFLD.eq.0) GO TO 410
      WRITE (6,620)
C
C     COMPUTE EQUIVALENT SOURCE M-FIELD
C
 840  continue
      WRITE(6,630)
      nxd=istrnxd
      nyd=istrnyd
      xord=strxord
      yord=stryord
      dxd=strdxd
      dyd=strdyd
      elvd=strelvd
      if (nfld .eq. 2) then
         read (13,*) iew
         read (13,*) ins
         read (13,*) scolat
         read (13,*) west
         read (13,*) gridspc
         do jy=1,nyd
            read (13,*) (df(ix,jy),ix=1,nxd)
         enddo
         DO JY=1,NYd
            do ix=1,nxd
               L=((jy-1)*nxd)+ix
               P(L)=df(ix,jy)
            enddo
         enddo
      endif
c
      IF (mYEAR.LT.1.E-9) GO TO 880
      call fieldg (0.,0.,0.,0.,55,1,q1,q2,q3,q4)
      if (mfl+mil+mdl .gt. 0.0) go to 870
      call geomag (myear,elvd,yord,xord,dyd,dxd,nyd,nxd,11)
 870  if (mi+md .gt. 0.0) go to 880
      call geomag (myear,melvo,myoro,mxoro,mdyo,mdxo,mnyo,mnxo,10)
c
 880  continue
c
      PI=3.1415926536
      FACT=PI/180.0
c
      mXORO=mXORO*FACT
      XORD=XORD*FACT
      mYORO=(90.0-mYORO)*FACT
      YORD=(90.0-YORD)*FACT
      mDXO=mDXO*FACT
      DXD=DXD*FACT
      mDYO=-mDYO*FACT
      DYD=-DYD*FACT
```

```
C
      mI=mI*FACT
      mD=-mD*FACT
      mI1=mI1*FACT
      mD1=-mD1*FACT
C
      REARTH=6371.0
      RO=mELVO+REARTH
      RD=ELVD+REARTH
      E=RD**2+RO**2
      G=2.0*RD*RO
C
      IF (mYEAR.LT.1.E-9) GO TO 300
      IF (mF1+mI1+mD1.GT.0.0 .AND. mI+mD.LT.1.E-9) GO TO 330
      IF (mF1+mI1+mD1.LT.1.E-9 .AND. mI+mD.GT.0.0) GO TO 360
C
C     COMPUTE M-FIELD FOR CASE WHERE (mI,mD) AND (mF1,mI1,mD1)
C     ARE DERIVED FROM FIELDG
C
      DO 290 JY=1,mNYO
         THETA=mYORO+(FLOAT(JY)-1.0)*mDYO
         COST=COS(THETA)
         SINT=SIN(THETA)
C
         do 195 l=1,mnxo
            fld(l)=fldo(jy,l)
            inc(l)=inco(jy,l)
            dec(l)=deco(jy,l)
 195     continue
C
         DO 280 IX=1,mNXO
            PHI=mXORO+(FLOAT(IX)-1.0)*mDXO
            aI=INC(IX)
            aD=DEC(IX)
C
            CALL MAGS1 (aI,aD)
C
  280    DFS(IX,JY)=YC
  290 WRITE(6,580) (DFS(IX,JY),IX=1,mNXO)
      GO TO 390
C
C     COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (mI,mD)
C     AND (mF1,mI1,mD1)
C
  300 DO 320 JY=1,mNYO
         THETA=mYORO+(FLOAT(JY)-1.0)*mDYO
         COST=COS(THETA)
         SINT=SIN(THETA)
         DO 310 IX=1,mNXO
            PHI=mXORO+(FLOAT(IX)-1.0)*mDXO
C
            CALL MAGS2 (mI,mD,mF1,mI1,mD1)
C
  310    DFS(IX,JY)=YC
  320 WRITE(6,580) (DFS(IX,JY),IX=1,mNXO)
      GO TO 390
C
C     COMPUTE M-FIELD FOR CASE WHERE (mI,mD) ARE DERIVED FROM
C     FIELDG AND USER SUPPLIES (mF1,mI1,mD1)
C
  330 DO 350 JY=1,mNYO
         THETA=mYORO+(FLOAT(JY)-1.0)*mDYO
         COST=COS(THETA)
         SINT=SIN(THETA)
C
         do 205 l=1,mnxo
            fld(l)=fldo(jy,l)
            inc(l)=inco(jy,l)
            dec(l)=deco(jy,l)
 205     continue
C
         DO 340 IX=1,mNXO
            PHI=mXORO+(FLOAT(IX)-1.0)*mDXO
            aI=INC(IX)
            aD=DEC(IX)
C
            CALL MAGS2 (aI,aD,mF1,mI1,mD1)
C
  340    DFS(IX,JY)=YC
  350 WRITE(6,580) (DFS(IX,JY),IX=1,mNXO)
      GO TO 390
C
C     COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (mI,mD) AND
C     (mF1,mI1,mD1) ARE DERIVED FROM FIELDG
C
  360 DO 380 JY=1,mNYO
         THETA=mYORO+(FLOAT(JY)-1.0)*mDYO
         COST=COS(THETA)
         SINT=SIN(THETA)
         DO 370 IX=1,mNXO
            PHI=mXORO+(FLOAT(IX)-1.0)*mDXO
```

```
C
            CALL MAGS1 (mI,mD)
C
  370    DFS(IX,JY)=YC
  380 WRITE(6,580)  (DFS(IX,JY),IX=1,mNXO)
  390 CONTINUE
C
C     COMPUTE MAXIMUM, MINIMUM AND AVERAGE AMPLITUDE VALUES FOR EQUIVA-
C     LENT SOURCE M-FIELD
C
      DSUM=0.0
      DMIN=DFS(1,1)
      DMAX=DMIN
      DO 400 JY=1,mNYO
      DO 400 IX=1,mNXO
         DSUM=DSUM+DFS(IX,JY)
         IF (DMAX.LT.DFS(IX,JY)) DMAX=DFS(IX,JY)
         IF (DMIN.GT.DFS(IX,JY)) DMIN=DFS(IX,JY)
  400 CONTINUE
      DSUM=DSUM/FLOAT(mNXO*mNYO)
      WRITE(6,640) DMAX,DMIN,DSUM
C
      if (isub .eq. 0) dsum=0.0
      dmax=dfs(1,1)
      dmin=dmax
      write (30,*) mnxo
      write (30,*) mnyo
      write (30,*) myoro/fact
      write (30,*) mxoro/fact
      write (30,*) mdxo/fact
      DO 385 JY=1,mNYO
         WRITE(30,580) ((DFS(IX,JY)-dsum),IX=1,mNXO)
         do 385 ix=1,mnxo
            dmax=max(dmax,(dfs(ix,jy)-dsum))
            dmin=min(dmin,(dfs(ix,jy)-dsum))
 385  CONTINUE
      if (isub .eq. 1) write (6,*) 'new max=',dmax,' new min=',dmin
c
  410 CONTINUE
      SSUM=0.0
      DO 900 II=1,mNXO
         DO 900 J=1,mNYO
            SSUM=SSUM+(DF(II,J)-DFS(II,J))**2
 900  CONTINUE
      WRITE(6,790) SSUM
 790  FORMAT('  SUM OF SQUARES = ',E15.10)
 901  CONTINUE
      STOP
C
  420 FORMAT (/,1X,3HF1=,F10.3,5X,3HI1=,F6.2,5X,3HD1=,F6.2,5X,2HI=,F6.2,
     15X,2HD=,F6.2,/)
  430 FORMAT (/,1X,5HYEAR=,F10.5,5X,2HI=,F6.2,5X,2HD=,F6.2,/)
  440 FORMAT (/,1X,5HYEAR=,F10.5,5X,3HF1=,F10.3,5X,3HI1=,F6.2,5X,3HD1=,F
     16.2,/)
  450 FORMAT (/,1X,5HYEAR=,F10.5,/)
  480 FORMAT (//,2X,  7HRCOND =,E20.8)
  490 FORMAT (//,2X,26HATA LEADING MINOR OF ORDER,I5,1X,24HIS NOT POSITI
     1VE DEFINITE)
  520 FORMAT (2X,4HNXO=,I5,5X,4HNYO=,I5,5X,5HXORO=,F10.5,5X,5HYORO=,F10.
     15,5X,4HDXO=,F10.5,5X,4HDYO=,F10.5,5X,5HELVO=,F10.5,//)
  540 FORMAT (2X,4HNXD=,I5,5X,4HNYD=,I5,5X,5HXORD=,F10.5,5X,5HYORD=,F10.
     15,5X,4HDXD=,F10.5,5X,4HDYD=,F10.5,5X,5HELVD=,F10.5,//)
  580 FORMAT (8(1X,F9.3))
  590 FORMAT (//,'CGS-SUSCEPTIBILITIES',5X,'E-LONGITUDE',5X,'N-',
     >'LATITUDE',5X,'KM-ELEVATION',/)
  600 FORMAT (1X,E20.5,5X,F10.4,5X,F10.4,5X,F10.4)
  620 FORMAT (///)
  630 FORMAT (//,2X,25HEQUIVALENT SOURCE M-FIELD,/)
  640 FORMAT (//,2X,5HDMAX=,F10.3,10X,5HDMIN=,F10.3,10X,5HDAVG=,F10.3)
C
      END
      SUBROUTINE MAGS1 (I,D)
C
C***********************************************************************
C     SUBROUTINE MAGS1 CALCULATES THE TOTAL MAGNETIC FIELD AT A SPHERICAL
C      OBSERVATION POINT (R,THETA,PHI) DUE TO A SPHERICAL ARRAY OF POINT
C      DIPOLES WITH CGS-SUSCEPTIBILITIES P(L) AT SOURCE GRID COORDINATES
C      (R1,THETA1,PHI1).  POLARIZING FIELD CHARACTERISTICS (F1,I1,D1)
C      ARE READ FROM TAPE11.
C***********************************************************************
C
      COMMON XORD,YORD,DXD,DYD,NXD,NYD,RO,RD,E,G,COST,SINT,YC,THETA,PHI,
     >P(3982),TMAG(3982),DF(361,44),S(7930153)
      real fldd,fldo,incd,inco,decd,deco
      dimension fldd(44,181),fldo(44,361),incd(44,181),inco(44,361),
     >          decd(44,181),deco(44,361)
      common /gmf/fldd,fldo,incd,inco,decd,deco
      REAL I,I1,JR,JTHETA,JPHI,INC1(181),FLD1(181),DEC1(181)
      YC=0.0
      L=0
```

```
      DO 110 JY=1,NYD
         THETA1=YORD+FLOAT(JY-1)*DYD
         COST1=COS(THETA1)
         SINT1=SIN(THETA1)
         CT1T=COST1*COST
         ST1T=SINT1*SINT
         SC1T=SINT1*COST
         CS1T=COST1*SINT
C
         do 730 k=1,nxd
            fld1(k)=fldd(jy,k)
            inc1(k)=incd(jy,k)
            dec1(k)=decd(jy,k)
 730        continue
C
      DO 110 IX=1,NXD
         L=L+1
C
C     COMPUTE POINT DIPOLE POLARIZATION VECTOR IN SPHERICAL ORTHONORMAL
C     UNIT VECTORS (ER,ETHETA,EPHI) WITH SUCEP(L)=1.0
C
         F1=FLD1(IX)
         I1=INC1(IX)
         D1=DEC1(IX)
C
         JR=F1*SIN(I1)
         JTHETA=F1*COS(I1)*COS(D1)
         JPHI=F1*COS(I1)*SIN(D1)
C
C     COMPUTE MAGNETIC FIELD VECTOR (U,V,W) AT THE POINT (R,THETA,PHI)
C     DUE TO POINT DIPOLES AT (R1,THETA1(JY),PHI1(IX))
C
         PHI1=XORD+FLOAT(IX-1)*DXD
         SINP=SIN(PHI-PHI1)
         COSP=COS(PHI-PHI1)
         A=CT1T+ST1T*COSP
         B=SC1T-CS1T*COSP
         C=SINT*SINP
         R2=E-G*A
         R15=R2**1.5
         R25=R2**2.5
         XX=JR*(RD-RO*A)+JTHETA*RO*B-JPHI*RO*C
         ATHETA=-CS1T+SC1T*COSP
         BTHETA=-ST1T-CT1T*COSP
         CTHETA=COST*SINP
         APHI=-SINT1*SINP
         BPHI=COST1*SINP
         CPHI=COSP
C
         U1=-JR*A+JTHETA*B-JPHI*C
         U2=-3.0*XX*(RO-RD*A)
         U3=U1/R15
         U4=U2/R25
         U=U3+U4
C
         V1=-JR*ATHETA+JTHETA*BTHETA-JPHI*CTHETA
         V2=3.0*XX*RD*ATHETA
         V3=V1/R15
         V4=V2/R25
         V=V3+V4
C
         W1=-JR*APHI+JTHETA*BPHI-JPHI*CPHI
         W2=3.0*XX*RD*APHI
         W3=W1/R15
         W4=W2/R25
         W=W3+W4
C
C     CALCULATE THE COMPONENT OF THE ANOMALOUS FIELD IN THE DIRECTION
C     OF THE GEOMAGNETIC FIELD AT THE OBVSERVATION POINT---I.E.,
C     ((U,V,W)*(UR,UTHETA,UPHI))*P(L) = TOTAL MAGNETIC FIELD
C
         UR=SIN(I)
         UTHETA=COS(I)*COS(D)
         UPHI=COS(I)*SIN(D)
C
         TMAG(L)=UR*U+UTHETA*V+UPHI*W
         YC=YC+P(L)*TMAG(L)
  110 CONTINUE
C
      RETURN
C
      END
      SUBROUTINE MAGS2 (I,D,F1,I1,D1)
C
C*********************************************************************
C     SUBROUTINE MAGS2 CALCULATES THE TOTAL MAGNETIC FIELD AT A SPHERICAL
C     OBSERVATION POINT (R,THETA,PHI) DUE TO A SPHERICAL ARRAY OF POINT
C     DIPOLES WITH CGS-SUSCEPTIBILITIES P(L) AT SOURCE GRID COORDINATES
C     (R1,THETA1,PHI1).
C*********************************************************************
```

```
C
      COMMON XORD,YORD,DXD,DYD,NXD,NYD,RO,RD,E,G,COST,SINT,YC,THETA,PHI,
     >P(3982),TMAG(3982),DF(361,44),S(7930153)
      REAL I,I1,JR,JTHETA,JPHI
      YC=0.0
      L=0
      DO 110 JY=1,NYD
         THETA1=YORD+FLOAT(JY-1)*DYD
         COST1=COS(THETA1)
         SINT1=SIN(THETA1)
         CT1T=COST1*COST
         ST1T=SINT1*SINT
         SC1T=SINT1*COST
         CS1T=COST1*SINT
C
      DO 110 IX=1,NXD
         L=L+1
C
C     COMPUTE POINT DIPOLE POLARIZATION VECTOR IN SPHERICAL ORTHONORMAL
C     UNIT VECTORS (ER,ETHETA,EPHI) WITH SUSCEP(L)=1.0
C
         JR=F1*SIN(I1)
         JTHETA=F1*COS(I1)*COS(D1)
         JPHI=F1*COS(I1)*SIN(D1)
C
C     COMPUTE MAGNETIC FIELD VECTOR (U,V,W) AT THE POINT (R,THETA,PHI)
C     DUE TO POINT DIPOLES AT (R1,THETA1(JY),PHI1(IX))
C
         PHI1=XORD+FLOAT(IX-1)*DXD
         SINP=SIN(PHI-PHI1)
         COSP=COS(PHI-PHI1)
         A=CT1T+ST1T*COSP
         B=SC1T-CS1T*COSP
         C=SINT*SINP
         R2=E-G*A
         R15=R2**1.5
         R25=R2**2.5
         XX=JR*(RD-RO*A)+JTHETA*RO*B-JPHI*RO*C
         ATHETA=-CS1T+SC1T*COSP
         BTHETA=-ST1T-CT1T*COSP
         CTHETA=COST*SINP
         APHI=-SINT1*SINP
         BPHI=COST1*SINP
         CPHI=COSP
C
         U1=-JR*A+JTHETA*B-JPHI*C
         U2=-3.0*XX*(RO-RD*A)
         U3=U1/R15
         U4=U2/R25
         U=U3+U4
C
         V1=-JR*ATHETA+JTHETA*BTHETA-JPHI*CTHETA
         V2=3.0*XX*RD*ATHETA
         V3=V1/R15
         V4=V2/R25
         V=V3+V4
C
         W1=-JR*APHI+JTHETA*BPHI-JPHI*CPHI
         W2=3.0*XX*RD*APHI
         W3=W1/R15
         W4=W2/R25
         W=W3+W4
C
C     CALCULATE THE COMPONENT OF THE ANOMALOUS FIELD IN THE DIRECTION
C     OF THE GEOMAGNETIC FIELD AT THE OBVSERVATION POINT---I.E.,
C     ((U,V,W)*(UR,UTHETA,UPHI))*P(L) = TOTAL MAGNETIC FIELD
C
         UR=SIN(I)
         UTHETA=COS(I)*COS(D)
         UPHI=COS(I)*SIN(D)
C
         TMAG(L)=UR*U+UTHETA*V+UPHI*W
         YC=YC+P(L)*TMAG(L)
  110 CONTINUE
C
      RETURN
C
      END
      SUBROUTINE GEOMAG (YEAR,ELV,THETA,PHI,HTHETA,HPHI,NTHETA,NPHI,NF)
      real fldd,fldo,incd,inco,decd,deco
      dimension fldd(44,181),fldo(44,361),incd(44,181),inco(44,361),
     >          decd(44,181),deco(44,361)
      common /gmf/ fldd,fldo,incd,inco,decd,deco
C
C
C        ****************************************************************
C        THIS SUBROUTINE CALCULATES THE MAGNITUDE, INCLINATION, AND
C        DECLINATION OF THE GEOMAGNETIC FIELD ON A GRID NTHETA BY
C        NPHI
C        ****************************************************************
```

```
c
c
c          THETA,PHI  **  ORIGIN OF THE GRID (DEG.)
c                ELV  **  ELEVATION OF GRID (KILO. ABOVE SEA LEVEL)
c        HTHETA,HPHI  **  GRID SPACING (DEG.)
c        NTHETA,NPHI  **  DIMENSIONS OF THE GRID
c                 NF  **  UNIT FILE WHICH WILL STORE THE FIELD
c
c
c        ************************************************************
c        SUBROUTINES USED
c           ** FIELDG **  (NASA)
c           ** FIELD **   (NASA)
c           ** WRITEB **  (SYSTEM)
c        ************************************************************
c
      REAL F(361),INC(361),DEC(361)
c
      write (6,*) ' '
      if (nf .eq. 10) then
         write (6,*) 'the following is the geomagnetic field'
         write (6,*) 'over the observation grid'
      elseif (nf .eq. 11) then
         write (6,*) 'the following is the geomagnetic field'
         write (6,*) 'over the source grid'
      endif
      write (6,*) ' '
c
      LL=0
      YYYYYY=THETA
      XXXXXX=PHI
c
      DO 120 I=1,NTHETA
         PHI=XXXXXX
         DO 110 J=1,NPHI
c
            CALL FIELDG (THETA,PHI,ELV,YEAR,50,LL,X,Y,Z,F(J))
c
            H=SQRT(X**2+Y**2)
            INC(J)=ATAN2(Z,H)
            DEC(J)=ATAN2(Y,X)
            PHI=PHI+HPHI
  110    CONTINUE
c
c---------------------------- change 6 to nf in write statement
c                             if separate files for g.m.f.o.o.
c                             and g.m.f.o.s is wanted.  don't forget
c                             to also use open statements.
       WRITE(6,9981)  (F(L),L=1,NPHI)
       WRITE(6,9981)  (INC(L),L=1,NPHI)
       WRITE(6,9981)  (DEC(L),L=1,NPHI)
       write (67,9981)  (f(l),l=1,nphi)
       write (68,9981)  ((inc(l)*57.295828),l=1,nphi)
       write (69,9981)  ((dec(l)*57.295828),l=1,nphi)
9981   format (4(1x,f19.9))
c
       if (nf .eq. 11) then
          do 710 l=1,nphi
             fldd(i,l)=f(l)
             incd(i,l)=inc(l)
             decd(i,l)=dec(l)
 710         continue
       else if (nf .eq. 10) then
          do 720 l=1,nphi
             fldo(i,l)=f(l)
             inco(i,l)=inc(l)
             deco(i,l)=dec(l)
 720         continue
       endif
         THETA=THETA+HTHETA
  120 CONTINUE
c
      PHI=XXXXXX
      THETA=YYYYYY
      RETURN
c
      END
      SUBROUTINE FIELDG (DLAT,DLONG,ALT,TM,NMX,L,X,Y,Z,F)
c
c
c         ***************************************************************
c         FOR DOCUMENTATION OF THIS SUBROUTINE AND SUBROUTINE FIELD SEE :
c              NATIONAL SPACE SCIENCE DATA CENTER*S PUBLICATION
c              **COMPUTATION OF THE MAIN GEOMAGNETIC FIELD
c                FROM SPHERICAL HARMONIC EXPANSIONS**
c              DATA USERS' NOTE, NSSDC 68-11, MAY 1968
c              GODDARD SPACE FLIGHT CENTER, GREENBELT, MD.
c         ***************************************************************
c
c
c          DLAT ** LATITUDE IN DEGREES POSITIVE NORTH
```

```
C        DLONG ** LONGITUDE IN DEGREES POSITIVE EAST
C          ALT ** ELEVATION IN KM (POSITIVE ABOVE, NEGATIVE BELOW
C                 EARTH'S SURFACE)
C           TM ** EPOCH IN YEARS
C          NMX ** SET TO INTEGER GREATER THAN DEGREE OF EXPANSION
C            L ** SET TO 1 ON INITIAL DUMMY CALL, SET TO 0 ON SUBSEQUENT
C                 CALLS
C
C        SUBROUTINE RETURNS GEOMAGNETIC FIELD DIRECTIONS (X,Y,Z), POSI-
C        TIVE NORTH, EAST AND DOWN, RESPECTIVELY, AND MAGNITUDE OF TOTAL
C        FIELD, F---ALL VALUES ARE IN GAMMAS
C
C
      EQUIVALENCE (SHMIT(1,1),TG(1,1))
      COMMON /NASA/ TG(55,55)
      COMMON /FLDCOM/ CPH,SPH,R,CT,ST,BT,BP,BR,B
      COMMON /MAX/  NMAX
      DIMENSION G(55,55), GT(55,55), SHMIT(55,55), AID(55), GTT(55,55)
      DATA A/0./
C
      TLAST=0.0
C
      IF(A.EQ.6378.16) IF(L) 210,100,110
C
      A=6378.16
      FLAT=1.-1./298.25
      A2=A**2
      A4=A**4
      B2=(A*FLAT)**2
      A2B2=A2*(1.-FLAT**2)
      A4B4=A4*(1.-FLAT**4)
      IF (L) 160,160,110
  100 IF (TM-TLAST) 190,210,190
  110 READ (3,260) J,K,TZERO,(AID(I),I=1,11)
      L=0
c     WRITE(6,270) J,K,TZERO,(AID(I),I=1,11)
      MAXN=0
      TEMP=0.
  120 READ (3,280) N,M,GNM,HNM,GTNM,HTNM,GTTNM,HTTNM
      IF (N.LE.0) GO TO 130
      MAXN=(MAX0(N,MAXN))
      G(N,M)=GNM
      GT(N,M)=GTNM
      GTT(N,M)=GTTNM
      TEMP=AMAX1(TEMP,ABS(GTNM))
      IF (M.EQ.1) GO TO 120
      G(M-1,N)=HNM
      GT(M-1,N)=HTNM
      GTT(M-1,N)=HTTNM
      GO TO 120
 130  continue
      rewind (3)
c 130 WRITE(6,290)
c     DO 150 N=2,MAXN
c     DO 150 M=1,N
c        MI=M-1
c        IF (M.EQ.1) GO TO 140
c        WRITE(6,300) N,M,G(N,M),G(MI,N),GT(N,M),GT(MI,N),GTT(N,M),GTT(
c    1   MI,N)
c        GO TO 150
c 140    WRITE(6,310) N,M,G(N,M),GT(N,M),GTT(N,M)
c 150 CONTINUE
      WRITE (6,320)
      IF (TEMP.EQ.0.) L=-1
  160 IF (K.NE.0) GO TO 190
      SHMIT(1,1)=-1.
      DO 170 N=2,MAXN
         SHMIT(N,1)=SHMIT(N-1,1)*FLOAT(2*N-3)/FLOAT(N-1)
         SHMIT(1,N)=0.
         JJ=2
      DO 170 M=2,N
         SHMIT(N,M)=SHMIT(N,M-1)*SQRT(FLOAT((N-M+1)*JJ)/FLOAT(N+M-2))
         SHMIT(M-1,N)=SHMIT(N,M)
  170 JJ=1
      DO 180 N=2,MAXN
      DO 180 M=1,N
         G(N,M)=G(N,M)*SHMIT(N,M)
         GT(N,M)=GT(N,M)*SHMIT(N,M)
         GTT(N,M)=GTT(N,M)*SHMIT(N,M)
         IF (M.EQ.1) GO TO 180
         G(M-1,N)=G(M-1,N)*SHMIT(M-1,N)
         GT(M-1,N)=GT(M-1,N)*SHMIT(M-1,N)
         GTT(M-1,N)=GTT(M-1,N)*SHMIT(M-1,N)
  180 CONTINUE
  190 T=TM-TZERO
      DO 200 N=1,MAXN
      DO 200 M=1,N
         TG(N,M)=G(N,M)+T*(GT(N,M)+GTT(N,M)*T)
         IF (M.EQ.1) GO TO 200
         TG(M-1,N)=G(M-1,N)+T*(GT(M-1,N)+GTT(M-1,N)*T)
```

```
 200 CONTINUE
     TLAST=TM
 210 DLATR=DLAT/57.2957795
     SINLA=SIN(DLATR)
     RLONG=DLONG/57.2957795
     CPH=COS(RLONG)
     SPH=SIN(RLONG)
     IF (J.EQ.0) GO TO 220
     R=ALT+6371.0
     CT=SINLA
     GO TO 230
 220 SINLA2=SINLA**2
     COSLA2=1.-SINLA2
     DEN2=A2-A2B2*SINLA2
     DEN=SQRT(DEN2)
     FAC=(((ALT*DEN)+A2)/((ALT*DEN)+B2))**2
     CT=SINLA/SQRT(FAC*COSLA2+SINLA2)
     R=SQRT(ALT*(ALT+2.*DEN)+(A4-A4B4*SINLA2)/DEN2)
 230 ST=SQRT(1.-CT**2)
     NMAX=MIN0(NMX,MAXN)
C
     CALL FIELD
C
     Y=BP
     F=B
     IF (J) 240,250,240
 240 X=-BT
     Z=-BR
     RETURN
C
C    TRANSFORMS FIELD TO GEODETIC DIRECTIONS
C
 250 SIND=SINLA*ST-SQRT(COSLA2)*CT
     COSD=SQRT(1.0-SIND**2)
     X=-BT*COSD-BR*SIND
     Z=BT*SIND-BR*COSD
     RETURN
C
 260 FORMAT (2I1,1X,F6.1,10A6,A3)
 270 FORMAT (2I3,5X,6HEPOCH ,F7.1,5X,10A6,A3)
 280 FORMAT (2I3,6F11.4)
 290 FORMAT(6H0 N  M,6X,1HG,10X,1HH,9X,2HGT,9X,2HHT,8X,3HGTT,8X,3HHTT//
    1)
 300 FORMAT (2I3,6F11.4)
 310 FORMAT (2I3,F11.4,11X,F11.4,11X,F11.4)
 320 FORMAT (///)
C
     END
C
C
     SUBROUTINE FIELD
C
     COMMON /NASA/ G(55,55)
     COMMON /FLDCOM/ CPH,SPH,R,CT,ST,BT,BP,BR,B
     COMMON /MAX/ NMAX
     DIMENSION P(55,55), DP(55,55), CONST(55,55), SP(55), CP(55),
    > FN(55), FM(55)
     DATA P(1,1)/0./
C
     IF (P(1,1).EQ.1.0) GO TO 120
     P(1,1)=1.
     DP(1,1)=0.
     SP(1)=0.
     CP(1)=1.
     DO 110 N=2,18
        FN(N)=N
     DO 110 M=1,N
        FM(M)=M-1
 110 CONST(N,M)=FLOAT((N-2)**2-(M-1)**2)/FLOAT((2*N-3)*(2*N-5))
 120 SP(2)=SPH
     CP(2)=CPH
     DO 130 M=3,NMAX
        SP(M)=SP(2)*CP(M-1)+CP(2)*SP(M-1)
 130 CP(M)=CP(2)*CP(M-1)-SP(2)*SP(M-1)
     AOR=6371.0/R
     AR=AOR**2
     BT=0.
     BP=0.
     BR=0.
     DO 190 N=2,NMAX
        AR=AOR*AR
     DO 190 M=1,N
        IF (N-M) 150,140,150
 140    P(N,N)=ST*P(N-1,N-1)
        DP(N,N)=ST*DP(N-1,N-1)+CT*P(N-1,N-1)
        GO TO 160
 150    P(N,M)=CT*P(N-1,M)-CONST(N,M)*P(N-2,M)
        DP(N,M)=CT*DP(N-1,M)-ST*P(N-1,M)-CONST(N,M)*DP(N-2,M)
 160    PAR=P(N,M)*AR
        IF (M.EQ.1) GO TO 170
```

```
          TEMP=G(N,M)*CP(M)+G(M-1,N)*SP(M)
          BP=BP-(G(N,M)*SP(M)-G(M-1,N)*CP(M))*FM(M)*PAR
          GO TO 180
  170     TEMP=G(N,M)*CP(M)
          BP=BP-(G(N,M)*SP(M))*FM(M)*PAR
  180     BT=BT+TEMP*DP(N,M)*AR
  190 BR=BR-TEMP*FN(N)*PAR
      BP=BP/ST
      B=SQRT(BT*BT+BP*BP+BR*BR)
      RETURN
C
      END
      SUBROUTINE SPPCO(AP,N,RCOND,Z,INFO)
      INTEGER N,INFO
      REAL AP(1),Z(1)
      REAL RCOND
C
C     SPPCO FACTORS A REAL SYMMETRIC POSITIVE DEFINITE MATRIX
C     STORED IN PACKED FORM
C     AND ESTIMATES THE CONDITION OF THE MATRIX.
C
C     IF  RCOND  IS NOT NEEDED, SPPFA IS SLIGHTLY FASTER.
C     TO SOLVE  A*X = B , FOLLOW SPPCO BY SPPSL.
C     TO COMPUTE  INVERSE(A)*C , FOLLOW SPPCO BY SPPSL.
C     TO COMPUTE  DETERMINANT(A) , FOLLOW SPPCO BY SPPDI.
C     TO COMPUTE  INVERSE(A) , FOLLOW SPPCO BY SPPDI.
C
C     ON ENTRY
C
C        AP      REAL (N*(N+1)/2)
C                THE PACKED FORM OF A SYMMETRIC MATRIX  A .  THE
C                COLUMNS OF THE UPPER TRIANGLE ARE STORED SEQUENTIALLY
C                IN A ONE-DIMENSIONAL ARRAY OF LENGTH  N*(N+1)/2 .
C                SEE COMMENTS BELOW FOR DETAILS.
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        AP      AN UPPER TRIANGULAR MATRIX  R , STORED IN PACKED
C                FORM, SO THAT  A = TRANS(R)*R .
C                IF  INFO .NE. 0 , THE FACTORIZATION IS NOT COMPLETE.
C
C        RCOND   REAL
C                AN ESTIMATE OF THE RECIPROCAL CONDITION OF  A .
C                FOR THE SYSTEM  A*X = B , RELATIVE PERTURBATIONS
C                IN  A  AND  B  OF SIZE  EPSILON  MAY CAUSE
C                RELATIVE PERTURBATIONS IN  X  OF SIZE  EPSILON/RCOND .
C                IF  RCOND  IS SO SMALL THAT THE LOGICAL EXPRESSION
C                           1.0 + RCOND .EQ. 1.0
C                IS TRUE, THEN  A  MAY BE SINGULAR TO WORKING
C                PRECISION.  IN PARTICULAR,  RCOND  IS ZERO  IF
C                EXACT SINGULARITY IS DETECTED OR THE ESTIMATE
C                UNDERFLOWS.  IF INFO .NE. 0 , RCOND IS UNCHANGED.
C
C        Z       REAL(N)
C                A WORK VECTOR WHOSE CONTENTS ARE USUALLY UNIMPORTANT.
C                IF  A  IS SINGULAR TO WORKING PRECISION, THEN  Z  IS
C                AN APPROXIMATE NULL VECTOR IN THE SENSE THAT
C                NORM(A*Z) = RCOND*NORM(A)*NORM(Z) .
C                IF  INFO .NE. 0 , Z  IS UNCHANGED.
C
C        INFO    INTEGER
C                = 0  FOR NORMAL RETURN.
C                = K  SIGNALS AN ERROR CONDITION.  THE LEADING MINOR
C                     OF ORDER  K  IS NOT POSITIVE DEFINITE.
C
C     PACKED STORAGE
C
C          THE FOLLOWING PROGRAM SEGMENT WILL PACK THE UPPER
C          TRIANGLE OF A SYMMETRIC MATRIX.
C
C                K = 0
C                DO 20 J = 1, N
C                   DO 10 I = 1, J
C                      K = K + 1
C                      AP(K) = A(I,J)
C             10    CONTINUE
C             20 CONTINUE
C
C     LINPACK.  THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     LINPACK SPPFA
C     BLAS SAXPY,SDOT,SSCAL,SASUM
C     FORTRAN ABS,AMAX1,REAL,SIGN
C
C     INTERNAL VARIABLES
```

```
C
      REAL SDOT,EK,T,WK,WKM
      REAL ANORM,S,SASUM,SM,YNORM
      INTEGER I,IJ,J,JM1,J1,K,KB,KJ,KK,KP1
C
C
C     FIND NORM OF A
C
      J1 = 1
      DO 30 J = 1, N
         Z(J) = SASUM(J,AP(J1),1)
         IJ = J1
         J1 = J1 + J
         JM1 = J - 1
         IF (JM1 .LT. 1) GO TO 20
         DO 10 I = 1, JM1
            Z(I) = Z(I) + ABS(AP(IJ))
            IJ = IJ + 1
   10    CONTINUE
   20    CONTINUE
   30 CONTINUE
      ANORM = 0.0E0
      DO 40 J = 1, N
         ANORM = AMAX1(ANORM,Z(J))
   40 CONTINUE
C
C     FACTOR
C
      CALL SPPFA(AP,N,INFO)
      IF (INFO .NE. 0) GO TO 180
C
C        RCOND = 1/(NORM(A)*(ESTIMATE OF NORM(INVERSE(A)))) .
C        ESTIMATE = NORM(Z)/NORM(Y) WHERE  A*Z = Y  AND  A*Y = E .
C        THE COMPONENTS OF  E  ARE CHOSEN TO CAUSE MAXIMUM LOCAL
C        GROWTH IN THE ELEMENTS OF W  WHERE  TRANS(R)*W = E .
C        THE VECTORS ARE FREQUENTLY RESCALED TO AVOID OVERFLOW.
C
C        SOLVE TRANS(R)*W = E
C
         EK = 1.0E0
         DO 50 J = 1, N
            Z(J) = 0.0E0
   50    CONTINUE
         KK = 0
         DO 110 K = 1, N
            KK = KK + K
            IF (Z(K) .NE. 0.0E0) EK = SIGN(EK,-Z(K))
            IF (ABS(EK-Z(K)) .LE. AP(KK)) GO TO 60
               S = AP(KK)/ABS(EK-Z(K))
               CALL SSCAL(N,S,Z,1)
               EK = S*EK
   60       CONTINUE
            WK = EK - Z(K)
            WKM = -EK - Z(K)
            S = ABS(WK)
            SM = ABS(WKM)
            WK = WK/AP(KK)
            WKM = WKM/AP(KK)
            KP1 = K + 1
            KJ = KK + K
            IF (KP1 .GT. N) GO TO 100
               DO 70 J = KP1, N
                  SM = SM + ABS(Z(J)+WKM*AP(KJ))
                  Z(J) = Z(J) + WK*AP(KJ)
                  S = S + ABS(Z(J))
                  KJ = KJ + J
   70          CONTINUE
               IF (S .GE. SM) GO TO 90
                  T = WKM - WK
                  WK = WKM
                  KJ = KK + K
                  DO 80 J = KP1, N
                     Z(J) = Z(J) + T*AP(KJ)
                     KJ = KJ + J
   80             CONTINUE
   90          CONTINUE
  100       CONTINUE
            Z(K) = WK
  110    CONTINUE
         S = 1.0E0/SASUM(N,Z,1)
         CALL SSCAL(N,S,Z,1)
C
C        SOLVE R*Y = W
C
         DO 130 KB = 1, N
            K = N + 1 - KB
            IF (ABS(Z(K)) .LE. AP(KK)) GO TO 120
               S = AP(KK)/ABS(Z(K))
               CALL SSCAL(N,S,Z,1)
  120       CONTINUE
```

```
            Z(K) = Z(K)/AP(KK)
            KK = KK - K
            T = -Z(K)
            CALL SAXPY(K-1,T,AP(KK+1),1,Z(1),1)
  130    CONTINUE
         S = 1.0E0/SASUM(N,Z,1)
         CALL SSCAL(N,S,Z,1)
C
         YNORM = 1.0E0
C
C        SOLVE TRANS(R)*V = Y
C
         DO 150 K = 1, N
            Z(K) = Z(K) - SDOT(K-1,AP(KK+1),1,Z(1),1)
            KK = KK + K
            IF (ABS(Z(K)) .LE. AP(KK)) GO TO 140
               S = AP(KK)/ABS(Z(K))
               CALL SSCAL(N,S,Z,1)
               YNORM = S*YNORM
  140       CONTINUE
            Z(K) = Z(K)/AP(KK)
  150    CONTINUE
         S = 1.0E0/SASUM(N,Z,1)
         CALL SSCAL(N,S,Z,1)
         YNORM = S*YNORM
C
C        SOLVE R*Z = V
C
         DO 170 KB = 1, N
            K = N + 1 - KB
            IF (ABS(Z(K)) .LE. AP(KK)) GO TO 160
               S = AP(KK)/ABS(Z(K))
               CALL SSCAL(N,S,Z,1)
               YNORM = S*YNORM
  160       CONTINUE
            Z(K) = Z(K)/AP(KK)
            KK = KK - K
            T = -Z(K)
            CALL SAXPY(K-1,T,AP(KK+1),1,Z(1),1)
  170    CONTINUE
C        MAKE ZNORM = 1.0
         S = 1.0E0/SASUM(N,Z,1)
         CALL SSCAL(N,S,Z,1)
         YNORM = S*YNORM
C
         IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
         IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0
  180 CONTINUE
      RETURN
      END
      SUBROUTINE SPPFA(AP,N,INFO)
      INTEGER N,INFO
      REAL AP(1)
C
C     SPPFA FACTORS A REAL SYMMETRIC POSITIVE DEFINITE MATRIX
C     STORED IN PACKED FORM.
C
C     SPPFA IS USUALLY CALLED BY SPPCO, BUT IT CAN BE CALLED
C     DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C     (TIME FOR SPPCO) = (1 + 18/N)*(TIME FOR SPPFA) .
C
C     ON ENTRY
C
C        AP      REAL (N*(N+1)/2)
C                THE PACKED FORM OF A SYMMETRIC MATRIX  A .  THE
C                COLUMNS OF THE UPPER TRIANGLE ARE STORED SEQUENTIALLY
C                IN A ONE-DIMENSIONAL ARRAY OF LENGTH  N*(N+1)/2 .
C                SEE COMMENTS BELOW FOR DETAILS.
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        AP      AN UPPER TRIANGULAR MATRIX  R , STORED IN PACKED
C                FORM, SO THAT  A = TRANS(R)*R .
C
C        INFO    INTEGER
C                = 0  FOR NORMAL RETURN.
C                = K  IF THE LEADING MINOR OF ORDER  K  IS NOT
C                     POSITIVE DEFINITE.
C
C
C     PACKED STORAGE
C
C          THE FOLLOWING PROGRAM SEGMENT WILL PACK THE UPPER
C          TRIANGLE OF A SYMMETRIC MATRIX.
C
C                K = 0
C                DO 20 J = 1, N
```

```
C                  DO 10 I = 1, J
C                     K = K + 1
C                     AP(K) = A(I,J)
C            10    CONTINUE
C            20 CONTINUE
C
C     LINPACK.  THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS SDOT
C     FORTRAN SQRT
C
C     INTERNAL VARIABLES
C
      REAL SDOT,T
      REAL S
      INTEGER J,JJ,JM1,K,KJ,KK
C     BEGIN BLOCK WITH ...EXITS TO 40
C
C
         JJ = 0
         DO 30 J = 1, N
            INFO = J
            S = 0.0E0
            JM1 = J - 1
            KJ = JJ
            KK = 0
            IF (JM1 .LT. 1) GO TO 20
            DO 10 K = 1, JM1
               KJ = KJ + 1
               T = AP(KJ) - SDOT(K-1,AP(KK+1),1,AP(JJ+1),1)
               KK = KK + K
               T = T/AP(KK)
               AP(KJ) = T
               S = S + T*T
   10       CONTINUE
   20       CONTINUE
            JJ = JJ + J
            S = AP(JJ) - S
C     ......EXIT
            IF (S .LE. 0.0E0) GO TO 40
            AP(JJ) = SQRT(S)
   30    CONTINUE
         INFO = 0
   40 CONTINUE
      RETURN
      END
      SUBROUTINE SAXPY  (N,SA,SX,INCX,SY,INCY)
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - CDC/SINGLE
C
C   LATEST REVISION     - JANUARY 1, 1978
C
C   PURPOSE             - COMPUTE A CONSTANT TIMES A VECTOR PLUS
C                           A VECTOR, ALL SINGLE PRECISION
C
C   USAGE               - CALL SAXPY (N,SA,SX,INCX,SY,INCY)
C
C   ARGUMENTS    N      - LENGTH OF VECTORS X AND Y. (INPUT)
C                SA     - REAL SCALAR. (INPUT)
C                SX     - REAL VECTOR OF LENGTH MAX(N*IABS(INCX),1).
C                           (INPUT)
C                INCX   - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
C                           X(I) IS DEFINED TO BE..
C                           SX(1+(I-1)*INCX) IF INCX.GE.0 OR
C                           SX(1+(I-N)*INCX) IF INCX.LT.0.
C                SY     - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1).
C                           (INPUT/OUTPUT)
C                           SAXPY REPLACES Y(I) WITH SA*X(I)+Y(I)
C                           FOR I=1,...,N.
C                           X(I) AND Y(I) REFER TO SPECIFIC ELEMENTS
C                           OF SX AND SY, RESPECTIVELY. SEE INCX AND
C                           INCY ARGUMENT DESCRIPTIONS.
C                INCY   - DISPLACEMENT BETWEEN ELEMENTS OF SY. (INPUT)
C                           Y(I) IS DEFINED TO BE..
C                           SY(1+(I-1)*INCY) IF INCY.GE.0 OR
C                           SY(1+(I-N)*INCY) IF INCY.LT.0.
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
C   NOTATION            - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
```

```
C   COPYRIGHT           - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
C
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER            N,INCX,INCY
      REAL               SX(1),SY(1),SA
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER            I,IX,IY,M,MP1,NS
C                                  FIRST EXECUTABLE STATEMENT
      IF (N.LE.0.OR.SA.EQ.0.E0) RETURN
      IF (INCX.EQ.INCY) IF (INCX-1) 5,15,35
    5 CONTINUE
C                                  CODE FOR NONEQUAL OR NONPOSITIVE
C                                    INCREMENTS.
      IX = 1
      IY = 1
      IF (INCX.LT.0) IX = (-N+1)*INCX+1
      IF (INCY.LT.0) IY = (-N+1)*INCY+1
      DO 10 I=1,N
         SY(IY) = SY(IY)+SA*SX(IX)
         IX = IX+INCX
         IY = IY+INCY
   10 CONTINUE
      RETURN
C                                  CODE FOR BOTH INCREMENTS EQUAL TO 1
C                                    CLEAN-UP LOOP SO REMAINING VECTOR
C                                    LENGTH IS A MULTIPLE OF 4.
   15 M = N-(N/4)*4
      IF (M.EQ.0) GO TO 25
      DO 20 I=1,M
         SY(I) = SY(I)+SA*SX(I)
   20 CONTINUE
      IF (N.LT.4) RETURN
   25 MP1 = M+1
      DO 30 I=MP1,N,4
         SY(I) = SY(I)+SA*SX(I)
         SY(I+1) = SY(I+1)+SA*SX(I+1)
         SY(I+2) = SY(I+2)+SA*SX(I+2)
         SY(I+3) = SY(I+3)+SA*SX(I+3)
   30 CONTINUE
      RETURN
C                                  CODE FOR EQUAL, POSITIVE, NONUNIT
C                                    INCREMENTS.
   35 CONTINUE
      NS = N*INCX
      DO 40 I=1,NS,INCX
         SY(I) = SA*SX(I)+SY(I)
   40 CONTINUE
      RETURN
      END
      REAL FUNCTION SDOT (N,SX,INCX,SY,INCY)
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - CDC/SINGLE
C
C   LATEST REVISION     - JANUARY 1, 1978
C
C   PURPOSE             - COMPUTE SINGLE PRECISION DOT PRODUCT
C
C   USAGE               - FUNCTION SDOT (N,SX,INCX,SY,INCY)
C
C   ARGUMENTS    SDOT   - SUM FROM I=1 TO N OF X(I)*Y(I). (OUTPUT)
C                           X(I) AND Y(I) REFER TO SPECIFIC ELEMENTS
C                           OF SX AND SY, RESPECTIVELY. SEE INCX AND
C                           INCY ARGUMENT DESCRIPTIONS.
C                N      - LENGTH OF VECTORS X AND Y. (INPUT)
C                SX     - REAL VECTOR OF LENGTH MAX(N*IABS(INCX),1).
C                           (INPUT)
C                INCX   - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
C                           X(I) IS DEFINED TO BE..
C                           SX(1+(I-1)*INCX) IF INCX.GE.0 OR
C                           SX(1+(I-N)*INCX) IF INCX.LT.0.
C                SY     - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1).
C                           (INPUT)
C                INCY   - DISPLACEMENT BETWEEN ELEMENTS OF SY. (INPUT)
C                           Y(I) IS DEFINED TO BE..
C                           SY(1+(I-1)*INCY) IF INCY.GE.0 OR
C                           SY(1+(I-N)*INCY) IF INCY.LT.0.
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
```

```
C   NOTATION            - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C   COPYRIGHT           - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
C
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER            N,INCX,INCY
      REAL               SX(1),SY(1)
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER            I,M,MP1,NS,IX,IY
C                                  FIRST EXECUTABLE STATEMENT
      SDOT = 0.0E0
      IF (N.LE.0) RETURN
      IF (INCX.EQ.INCY) IF (INCX-1) 5,15,35
    5 CONTINUE
C                                  CODE FOR UNEQUAL INCREMENTS OR
C                                    NONPOSITIVE INCREMENTS.
      IX = 1
      IY = 1
      IF (INCX.LT.0) IX = (-N+1)*INCX+1
      IF (INCY.LT.0) IY = (-N+1)*INCY+1
      DO 10 I=1,N
         SDOT = SDOT+SX(IX)*SY(IY)
         IX = IX+INCX
         IY = IY+INCY
   10 CONTINUE
      RETURN
C                                  CODE FOR BOTH INCREMENTS EQUAL TO 1
C                                    CLEAN-UP LOOP SO REMAINING VECTOR
C                                    LENGTH IS A MULTIPLE OF 5.
   15 M = N-(N/5)*5
      IF (M.EQ.0) GO TO 25
      DO 20 I=1,M
         SDOT = SDOT+SX(I)*SY(I)
   20 CONTINUE
      IF (N.LT.5) RETURN
   25 MP1 = M+1
      DO 30 I=MP1,N,5
         SDOT = SDOT+SX(I)*SY(I)+SX(I+1)*SY(I+1)+SX(I+2)*SY(I+2)+SX(I
     1   +3)*SY(I+3)+SX(I+4)*SY(I+4)
   30 CONTINUE
      RETURN
C                                  CODE FOR POSITIVE EQUAL INCREMENTS
C                                    .NE.1.
   35 CONTINUE
      NS = N*INCX
      DO 40 I=1,NS,INCX
         SDOT = SDOT+SX(I)*SY(I)
   40 CONTINUE
      RETURN
      END
      SUBROUTINE SSCAL  (N,SA,SX,INCX)
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - CDC/SINGLE
C
C   LATEST REVISION     - JANUARY 1, 1978
C
C   PURPOSE             - COMPUTE A SINGLE PRECISION CONSTANT
C                           TIMES A SINGLE PRECISION VECTOR
C
C   USAGE               - CALL SSCAL (N,SA,SX,INCX)
C
C   ARGUMENTS    N      - LENGTH OF VECTOR X. (INPUT)
C                SA     - REAL SCALAR. (INPUT)
C                SX     - REAL VECTOR OF LENGTH N*INCX. (INPUT/OUTPUT)
C                           SSCAL REPLACES X(I) WITH SA*X(I) FOR
C                           I=1,...,N.
C                           X(I) REFERS TO A SPECIFIC ELEMENT OF SX.
C                           SEE INCX ARGUMENT DESCRIPTION.
C                INCX   - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
C                           X(I) IS DEFINED TO BE SX(1+(I-1)*INCX).
C                           INCX MUST BE GREATER THAN ZERO.
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
C   NOTATION            - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
```

```
C
C   COPYRIGHT           - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
C
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER            INCX,N
      REAL               SA,SX(1)
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER            I,M,MP1,NS
C                                  FIRST EXECUTABLE STATEMENT
      IF (N.LE.0) RETURN
      IF (INCX.EQ.1) GO TO 10
C                                  CODE FOR INCREMENTS NOT EQUAL TO 1.
      NS = N*INCX
      DO 5 I=1,NS,INCX
         SX(I) = SA*SX(I)
    5 CONTINUE
      RETURN
C                                  CODE FOR INCREMENTS EQUAL TO 1.
C                                    CLEAN-UP LOOP SO REMAINING VECTOR
C                                    LENGTH IS A MULTIPLE OF 5.
   10 M = N-(N/5)*5
      IF (M.EQ.0) GO TO 20
      DO 15 I=1,M
         SX(I) = SA*SX(I)
   15 CONTINUE
      IF (N.LT.5) RETURN
   20 MP1 = M+1
      DO 25 I=MP1,N,5
         SX(I) = SA*SX(I)
         SX(I+1) = SA*SX(I+1)
         SX(I+2) = SA*SX(I+2)
         SX(I+3) = SA*SX(I+3)
         SX(I+4) = SA*SX(I+4)
   25 CONTINUE
      RETURN
      END
      REAL FUNCTION SASUM (N,SX,INCX)
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - CDC/SINGLE
C
C   LATEST REVISION     - JANUARY 1, 1978
C
C   PURPOSE             - COMPUTE SINGLE PRECISION SUM OF ABSOLUTE
C                           VALUES
C
C   USAGE               - FUNCTION SASUM (N,SX,INCX)
C
C   ARGUMENTS    SASUM  - SUM FROM I=1 TO N OF ABS(X(I)). (OUTPUT)
C                           X(I) REFERS TO A SPECIFIC ELEMENT OF SX.
C                           SEE INCX ARGUMENT DESCRIPTION.
C                N      - LENGTH OF VECTOR X. (INPUT)
C                SX     - REAL VECTOR OF LENGTH N*INCX. (INPUT)
C                INCX   - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
C                           X(I) IS DEFINED TO BE SX(1+(I-1)*INCX).
C                           INCX MUST BE GREATER THAN ZERO.
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
C   NOTATION            - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C   COPYRIGHT           - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER            N,INCX
      REAL               SX(1)
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER            I,M,MP1,NS
C                                  FIRST EXECUTABLE STATEMENT
      SASUM = 0.0E0
      IF (N.LE.0) RETURN
      IF (INCX.EQ.1) GO TO 10
```

```
C                                    CODE FOR INCREMENTS NOT EQUAL TO 1.
      NS = N*INCX
      DO 5 I=1,NS,INCX
         SASUM = SASUM+ABS(SX(I))
    5 CONTINUE
      RETURN
C                                    CODE FOR INCREMENTS EQUAL TO 1.
C                                      CLEAN-UP LOOP SO REMAINING VECTOR
C                                      LENGTH IS A MULTIPLE OF 6.
   10 M = N-(N/6)*6
      IF (M.EQ.0) GO TO 20
      DO 15 I=1,M
         SASUM = SASUM+ABS(SX(I))
   15 CONTINUE
      IF (N.LT.6) RETURN
   20 MP1 = M+1
      DO 25 I=MP1,N,6
         SASUM = SASUM+ABS(SX(I))+ABS(SX(I+1))+ABS(SX(I+2))+ABS(SX(I
     1   +3))+ABS(SX(I+4))+ABS(SX(I+5))
   25 CONTINUE
      RETURN
      END
      SUBROUTINE SPPSL(AP,N,B)
      INTEGER N
      REAL AP(1),B(1)
C
C     SPPSL SOLVES THE REAL SYMMETRIC POSITIVE DEFINITE SYSTEM
C     A * X = B
C     USING THE FACTORS COMPUTED BY SPPCO OR SPPFA.
C
C     ON ENTRY
C
C        AP      REAL (N*(N+1)/2)
C                THE OUTPUT FROM SPPCO OR SPPFA.
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C        B       REAL(N)
C                THE RIGHT HAND SIDE VECTOR.
C
C     ON RETURN
C
C        B       THE SOLUTION VECTOR  X .
C
C     ERROR CONDITION
C
C        A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS
C        A ZERO ON THE DIAGONAL.  TECHNICALLY THIS INDICATES
C        SINGULARITY BUT IT IS USUALLY CAUSED BY IMPROPER SUBROUTINE
C        ARGUMENTS.  IT WILL NOT OCCUR IF THE SUBROUTINES ARE CALLED
C        CORRECTLY AND  INFO .EQ. 0 .
C
C     TO COMPUTE  INVERSE(A) * C  WHERE  C  IS A MATRIX
C     WITH  P  COLUMNS
C           CALL SPPCO(AP,N,RCOND,Z,INFO)
C           IF (RCOND IS TOO SMALL .OR. INFO .NE. 0) GO TO ...
C           DO 10 J = 1, P
C              CALL SPPSL(AP,N,C(1,J))
C        10 CONTINUE
C
C     LINPACK.  THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS SAXPY,SDOT
C
C     INTERNAL VARIABLES
C
      REAL SDOT,T
      INTEGER K,KB,KK
C
      KK = 0
      DO 10 K = 1, N
         T = SDOT(K-1,AP(KK+1),1,B(1),1)
         KK = KK + K
         B(K) = (B(K) - T)/AP(KK)
   10 CONTINUE
      DO 20 KB = 1, N
         K = N + 1 - KB
         B(K) = B(K)/AP(KK)
         KK = KK - K
         T = -B(K)
         CALL SAXPY(K-1,T,AP(KK+1),1,B(1),1)
   20 CONTINUE
      RETURN
      END
```

0.0000000000000D+00	0.6259999871254D+00
0.1000000000000D+01	0.6259951659277D+00
0.2000000000000D+01	0.6259807025574D+00
0.3000000000000D+01	0.6259565976827D+00
0.4000000000000D+01	0.6259228524177D+00
0.5000000000000D+01	0.6258794683214D+00
0.6000000000000D+01	0.6258264473985D+00
0.7000000000000D+01	0.6257637920986D+00
0.8000000000000D+01	0.6256915053163D+00
0.9000000000000D+01	0.6256095903911D+00
0.1000000000000D+02	0.6255180511067D+00
0.1100000000000D+02	0.6254168916912D+00
0.1200000000000D+02	0.6253061168165D+00
0.1300000000000D+02	0.6251857315978D+00
0.1400000000000D+02	0.6250557415938D+00
0.1500000000000D+02	0.6249161528054D+00
0.1600000000000D+02	0.6247669716761D+00
0.1700000000000D+02	0.6246082050909D+00
0.1800000000000D+02	0.6244398603760D+00
0.1900000000000D+02	0.6242619452984D+00
0.2000000000000D+02	0.6240744680650D+00
0.2100000000000D+02	0.6238774373220D+00
0.2200000000000D+02	0.6236708621544D+00
0.2300000000000D+02	0.6234547520854D+00
0.2400000000000D+02	0.6232291170752D+00
0.2500000000000D+02	0.6229939675208D+00
0.2600000000000D+02	0.6227493142548D+00
0.2700000000000D+02	0.6224951685447D+00
0.2800000000000D+02	0.6222315420922D+00
0.2900000000000D+02	0.6219584470318D+00
0.3000000000000D+02	0.6216758959306D+00
0.3100000000000D+02	0.6213839017867D+00
0.3200000000000D+02	0.6210824780285D+00
0.3300000000000D+02	0.6207716385137D+00
0.3400000000000D+02	0.6204513975281D+00
0.3500000000000D+02	0.6201217697846D+00
0.3600000000000D+02	0.6197827704219D+00
0.3700000000000D+02	0.6194344150038D+00
0.3800000000000D+02	0.6190767195172D+00
0.3900000000000D+02	0.6187097003719D+00
0.4000000000000D+02	0.6183333743984D+00
0.4100000000000D+02	0.6179477588472D+00
0.4200000000000D+02	0.6175528713873D+00
0.4300000000000D+02	0.6171487301050D+00
0.4400000000000D+02	0.6167353535020D+00
0.4500000000000D+02	0.6163127604947D+00
0.4600000000000D+02	0.6158809704124D+00
0.4700000000000D+02	0.6154400029956D+00
0.4800000000000D+02	0.6149898783950D+00
0.4900000000000D+02	0.6145306171697D+00
0.5000000000000D+02	0.6140622402854D+00
0.5100000000000D+02	0.6135847691134D+00
0.5200000000000D+02	0.6130982254283D+00
0.5300000000000D+02	0.6126026314069D+00
0.5400000000000D+02	0.6120980096262D+00
0.5500000000000D+02	0.6115843830618D+00
0.5600000000000D+02	0.6110617750861D+00
0.5700000000000D+02	0.6105302094665D+00
0.5800000000000D+02	0.6099897103638D+00
0.5900000000000D+02	0.6094403023301D+00
0.6000000000000D+02	0.6088820103071D+00
0.6100000000000D+02	0.6083148596243D+00
0.6200000000000D+02	0.6077388759968D+00
0.6300000000000D+02	0.6071540855235D+00
0.6400000000000D+02	0.6065605146854D+00
0.6500000000000D+02	0.6059581903430D+00
0.6600000000000D+02	0.6053471397350D+00
0.6700000000000D+02	0.6047273904756D+00
0.6800000000000D+02	0.6040989705528D+00
0.6900000000000D+02	0.6034619083261D+00
0.7000000000000D+02	0.6028162325247D+00
0.7100000000000D+02	0.6021619722448D+00
0.7200000000000D+02	0.6014991569479D+00
0.7300000000000D+02	0.6008278164583D+00
0.7400000000000D+02	0.6001479809611D+00
0.7500000000000D+02	0.5994596809996D+00
0.7600000000000D+02	0.5987629474735D+00
0.7700000000000D+02	0.5980578116360D+00
0.7800000000000D+02	0.5973443050921D+00
0.7900000000000D+02	0.5966224597955D+00
0.8000000000000D+02	0.5958923080471D+00
0.8100000000000D+02	0.5951538824917D+00
0.8200000000000D+02	0.5944072161163D+00
0.8300000000000D+02	0.5936523422472D+00
0.8400000000000D+02	0.5928892945476D+00
0.8500000000000D+02	0.5921181070153D+00
0.8600000000000D+02	0.5913388139798D+00
0.8700000000000D+02	0.5905514501004D+00
0.8800000000000D+02	0.5897560503628D+00
0.8900000000000D+02	0.5889526500771D+00

0.900000000000D+02	0.588141284875OD+00
0.910000000000D+02	0.5873219907072D+00
0.920000000000D+02	0.5864948038409D+00
0.930000000000D+02	0.5856597608566D+00
0.940000000000D+02	0.5848168986460D+00
0.950000000000D+02	0.5839662544091D+00
0.960000000000D+02	0.5831078656513D+00
0.970000000000D+02	0.5822417701807D+00
0.980000000000D+02	0.5813680061057D+00
0.990000000000D+02	0.5804866118315D+00
0.100000000000D+03	0.5795976260581D+00
0.101000000000D+03	0.5787010877766D+00
0.102000000000D+03	0.5777970362672D+00
0.103000000000D+03	0.5768855110959D+00
0.104000000000D+03	0.5759665521114D+00
0.105000000000D+03	0.5750401994427D+00
0.106000000000D+03	0.5741064934960D+00
0.107000000000D+03	0.5731654749515D+00
0.108000000000D+03	0.5722171847609D+00
0.109000000000D+03	0.5712616641440D+00
0.110000000000D+03	0.5702989545861D+00
0.111000000000D+03	0.5693290978346D+00
0.112000000000D+03	0.5683521358965D+00
0.113000000000D+03	0.5673681110348D+00
0.114000000000D+03	0.5663770657660D+00
0.115000000000D+03	0.5653790428566D+00
0.116000000000D+03	0.5643740853202D+00
0.117000000000D+03	0.5633622364147D+00
0.118000000000D+03	0.5623435396386D+00
0.119000000000D+03	0.5613180387285D+00
0.120000000000D+03	0.5602857776556D+00
0.121000000000D+03	0.5592468006225D+00
0.122000000000D+03	0.5582011520606D+00
0.123000000000D+03	0.5571488766262D+00
0.124000000000D+03	0.5560900191979D+00
0.125000000000D+03	0.5550246248732D+00
0.126000000000D+03	0.5539527389653D+00
0.127000000000D+03	0.5528744069999D+00
0.128000000000D+03	0.5517896747120D+00
0.129000000000D+03	0.5506985880427D+00
0.130000000000D+03	0.5496011931360D+00
0.131000000000D+03	0.5484975363355D+00
0.132000000000D+03	0.5473876641810D+00
0.133000000000D+03	0.5462716234057D+00
0.134000000000D+03	0.5451494609324D+00
0.135000000000D+03	0.5440212238706D+00
0.136000000000D+03	0.5428869595129D+00
0.137000000000D+03	0.5417467153322D+00
0.138000000000D+03	0.5406005389776D+00
0.139000000000D+03	0.5394484782722D+00
0.140000000000D+03	0.5382905812086D+00
0.141000000000D+03	0.5371268959465D+00
0.142000000000D+03	0.5359574708091D+00
0.143000000000D+03	0.5347823542794D+00
0.144000000000D+03	0.5336015949974D+00
0.145000000000D+03	0.5324152417565D+00
0.146000000000D+03	0.5312233435002D+00
0.147000000000D+03	0.5300259493188D+00
0.148000000000D+03	0.5288231084458D+00
0.149000000000D+03	0.5276148702550D+00
0.150000000000D+03	0.5264012842568D+00
0.151000000000D+03	0.5251824000948D+00
0.152000000000D+03	0.5239582675429D+00
0.153000000000D+03	0.5227289365014D+00
0.154000000000D+03	0.5214944569937D+00
0.155000000000D+03	0.5202548791633D+00
0.156000000000D+03	0.5190102532702D+00
0.157000000000D+03	0.5177606296874D+00
0.158000000000D+03	0.5165060588978D+00
0.159000000000D+03	0.5152465914907D+00
0.160000000000D+03	0.5139822781582D+00
0.161000000000D+03	0.5127131696923D+00
0.162000000000D+03	0.5114393169812D+00
0.163000000000D+03	0.5101607710061D+00
0.164000000000D+03	0.5088775828375D+00
0.165000000000D+03	0.5075898036323D+00
0.166000000000D+03	0.5062974846303D+00
0.167000000000D+03	0.5050006771504D+00
0.168000000000D+03	0.5036994325880D+00
0.169000000000D+03	0.5023938024109D+00
0.170000000000D+03	0.5010838381565D+00
0.171000000000D+03	0.4997695914281D+00
0.172000000000D+03	0.4984511138918D+00
0.173000000000D+03	0.4971284572729D+00
0.174000000000D+03	0.4958016733528D+00
0.175000000000D+03	0.4944708139656D+00
0.176000000000D+03	0.4931359309945D+00
0.177000000000D+03	0.4917970763691D+00
0.178000000000D+03	0.4904543020614D+00
0.179000000000D+03	0.4891076600828D+00

0.180000000000D+03	0.487757202480 9D+00
0.181000000000D+03	0.4864029813359D+00
0.182000000000D+03	0.4850450487576D+00
0.183000000000D+03	0.4836834568820D+00
0.184000000000D+03	0.4823182578680D+00
0.185000000000D+03	0.4809495038939D+00
0.186000000000D+03	0.4795772471547D+00
0.187000000000D+03	0.4782015398583D+00
0.188000000000D+03	0.4768224342225D+00
0.189000000000D+03	0.4754399824718D+00
0.190000000000D+03	0.4740542368340D+00
0.191000000000D+03	0.4726652495370D+00
0.192000000000D+03	0.4712730728059D+00
0.193000000000D+03	0.4698777588592D+00
0.194000000000D+03	0.4684793599063D+00
0.195000000000D+03	0.4670779281436D+00
0.196000000000D+03	0.4656735157521D+00
0.197000000000D+03	0.4642661748934D+00
0.198000000000D+03	0.4628559577073D+00
0.199000000000D+03	0.4614429163083D+00
0.200000000000D+03	0.4600271027823D+00
0.201000000000D+03	0.4586085691839D+00
0.202000000000D+03	0.4571873675330D+00
0.203000000000D+03	0.4557635498118D+00
0.204000000000D+03	0.4543371679618D+00
0.205000000000D+03	0.4529082738804D+00
0.206000000000D+03	0.4514769194184D+00
0.207000000000D+03	0.4500431563765D+00
0.208000000000D+03	0.4486070365022D+00
0.209000000000D+03	0.4471686114874D+00
0.210000000000D+03	0.4457279329648D+00
0.211000000000D+03	0.4442850525050D+00
0.212000000000D+03	0.4428400216139D+00
0.213000000000D+03	0.4413928917293D+00
0.214000000000D+03	0.4399437142181D+00
0.215000000000D+03	0.4384925403737D+00
0.216000000000D+03	0.4370394214124D+00
0.217000000000D+03	0.4355844084712D+00
0.218000000000D+03	0.4341275526044D+00
0.219000000000D+03	0.4326689047813D+00
0.220000000000D+03	0.4312085158825D+00
0.221000000000D+03	0.4297464366981D+00
0.222000000000D+03	0.4282827179240D+00
0.223000000000D+03	0.4268174101595D+00
0.224000000000D+03	0.4253505639046D+00
0.225000000000D+03	0.4238822295570D+00
0.226000000000D+03	0.4224124574095D+00
0.227000000000D+03	0.4209412976472D+00
0.228000000000D+03	0.4194688003449D+00
0.229000000000D+03	0.4179950154642D+00
0.230000000000D+03	0.4165199928510D+00
0.231000000000D+03	0.4150437822327D+00
0.232000000000D+03	0.4135664332158D+00
0.233000000000D+03	0.4120879952829D+00
0.234000000000D+03	0.4106085177906D+00
0.235000000000D+03	0.4091280499664D+00
0.236000000000D+03	0.4076466409064D+00
0.237000000000D+03	0.4061643395729D+00
0.238000000000D+03	0.4046811947913D+00
0.239000000000D+03	0.4031972552485D+00
0.240000000000D+03	0.4017125694895D+00
0.241000000000D+03	0.4002271859156D+00
0.242000000000D+03	0.3987411527814D+00
0.243000000000D+03	0.3972545181930D+00
0.244000000000D+03	0.3957673301049D+00
0.245000000000D+03	0.3942796363183D+00
0.246000000000D+03	0.3927914844783D+00
0.247000000000D+03	0.3913029220715D+00
0.248000000000D+03	0.3898139964241D+00
0.249000000000D+03	0.3883247546990D+00
0.250000000000D+03	0.3868352438943D+00
0.251000000000D+03	0.3853455108403D+00
0.252000000000D+03	0.3838556021976D+00
0.253000000000D+03	0.3823655644548D+00
0.254000000000D+03	0.3808754439266D+00
0.255000000000D+03	0.3793852867510D+00
0.256000000000D+03	0.3778951388880D+00
0.257000000000D+03	0.3764050461165D+00
0.258000000000D+03	0.3749150540331D+00
0.259000000000D+03	0.3734252080495D+00
0.260000000000D+03	0.3719355533903D+00
0.261000000000D+03	0.3704461350916D+00
0.262000000000D+03	0.3689569979984D+00
0.263000000000D+03	0.3674681867628D+00
0.264000000000D+03	0.3659797458421D+00
0.265000000000D+03	0.3644917194967D+00
0.266000000000D+03	0.3630041517883D+00
0.267000000000D+03	0.3615170865781D+00
0.268000000000D+03	0.3600305675246D+00
0.269000000000D+03	0.3585446380821D+00

0.270000000000D+03	0.3570593414984D+00
0.271000000000D+03	0.3555747208137D+00
0.272000000000D+03	0.3540908188580D+00
0.273000000000D+03	0.3526076782501D+00
0.274000000000D+03	0.3511253413953D+00
0.275000000000D+03	0.3496438504839D+00
0.276000000000D+03	0.3481632474896D+00
0.277000000000D+03	0.3466835741677D+00
0.278000000000D+03	0.3452048720535D+00
0.279000000000D+03	0.3437271824605D+00
0.280000000000D+03	0.3422505464793D+00
0.281000000000D+03	0.3407750049755D+00
0.282000000000D+03	0.3393005985885D+00
0.283000000000D+03	0.3378273677295D+00
0.284000000000D+03	0.3363553525809D+00
0.285000000000D+03	0.3348845930938D+00
0.286000000000D+03	0.3334151289872D+00
0.287000000000D+03	0.3319469997464D+00
0.288000000000D+03	0.3304802446216D+00
0.289000000000D+03	0.3290149026264D+00
0.290000000000D+03	0.3275510125368D+00
0.291000000000D+03	0.3260886128892D+00
0.292000000000D+03	0.3246277419800D+00
0.293000000000D+03	0.3231684378636D+00
0.294000000000D+03	0.3217107383514D+00
0.295000000000D+03	0.3202546810106D+00
0.296000000000D+03	0.3188003031629D+00
0.297000000000D+03	0.3173476418834D+00
0.298000000000D+03	0.3158967339994D+00
0.299000000000D+03	0.3144476160893D+00
0.300000000000D+03	0.3130003244814D+00
0.301000000000D+03	0.3115548952530D+00
0.302000000000D+03	0.3101113642289D+00
0.303000000000D+03	0.3086697669811D+00
0.304000000000D+03	0.3072301388271D+00
0.305000000000D+03	0.3057925148292D+00
0.306000000000D+03	0.3043569297935D+00
0.307000000000D+03	0.3029234182691D+00
0.308000000000D+03	0.3014920145469D+00
0.309000000000D+03	0.3000627526589D+00
0.310000000000D+03	0.2986356663774D+00
0.311000000000D+03	0.2972107892137D+00
0.312000000000D+03	0.2957881544182D+00
0.313000000000D+03	0.2943677949784D+00
0.314000000000D+03	0.2929497436193D+00
0.315000000000D+03	0.2915340328017D+00
0.316000000000D+03	0.2901206947222D+00
0.317000000000D+03	0.2887097613119D+00
0.318000000000D+03	0.2873012642362D+00
0.319000000000D+03	0.2858952348940D+00
0.320000000000D+03	0.2844917044167D+00
0.321000000000D+03	0.2830907036683D+00
0.322000000000D+03	0.2816922632443D+00
0.323000000000D+03	0.2802964134712D+00
0.324000000000D+03	0.2789031844060D+00
0.325000000000D+03	0.2775126058360D+00
0.326000000000D+03	0.2761247072778D+00
0.327000000000D+03	0.2747395179772D+00
0.328000000000D+03	0.2733570669084D+00
0.329000000000D+03	0.2719773827742D+00
0.330000000000D+03	0.2706004940049D+00
0.331000000000D+03	0.2692264287584D+00
0.332000000000D+03	0.2678552149196D+00
0.333000000000D+03	0.2664868801002D+00
0.334000000000D+03	0.2651214516384D+00
0.335000000000D+03	0.2637589565986D+00
0.336000000000D+03	0.2623994217710D+00
0.337000000000D+03	0.2610428736716D+00
0.338000000000D+03	0.2596893385417D+00
0.339000000000D+03	0.2583388423481D+00
0.340000000000D+03	0.2569914107826D+00
0.341000000000D+03	0.2556470692618D+00
0.342000000000D+03	0.2543058429273D+00
0.343000000000D+03	0.2529677566454D+00
0.344000000000D+03	0.2516328350070D+00
0.345000000000D+03	0.2503011023275D+00
0.346000000000D+03	0.2489725826471D+00
0.347000000000D+03	0.2476472997301D+00
0.348000000000D+03	0.2463252770656D+00
0.349000000000D+03	0.2450065378672D+00
0.350000000000D+03	0.2436911050730D+00
0.351000000000D+03	0.2423790013459D+00
0.352000000000D+03	0.2410702490733D+00
0.353000000000D+03	0.2397648703675D+00
0.354000000000D+03	0.2384628870660D+00
0.355000000000D+03	0.2371643207312D+00
0.356000000000D+03	0.2358691926507D+00
0.357000000000D+03	0.2345775238377D+00
0.358000000000D+03	0.2332893350312D+00
0.359000000000D+03	0.2320046466958D+00

0.360000000000D+03	0.2307234790226D+00
0.361000000000D+03	0.2294458519287D+00
0.362000000000D+03	0.2281717850583D+00
0.363000000000D+03	0.2269012977824D+00
0.364000000000D+03	0.2256344091995D+00
0.365000000000D+03	0.2243711381355D+00
0.366000000000D+03	0.2231115031446D+00
0.367000000000D+03	0.2218555225093D+00
0.368000000000D+03	0.2206032142411D+00
0.369000000000D+03	0.2193545960805D+00
0.370000000000D+03	0.2181096854979D+00
0.371000000000D+03	0.2168684996937D+00
0.372000000000D+03	0.2156310555991D+00
0.373000000000D+03	0.2143973698761D+00
0.374000000000D+03	0.2131674589188D+00
0.375000000000D+03	0.2119413388529D+00
0.376000000000D+03	0.2107190255372D+00
0.377000000000D+03	0.2095005345637D+00
0.378000000000D+03	0.2082858812580D+00
0.379000000000D+03	0.2070750806805D+00
0.380000000000D+03	0.2058681476263D+00
0.381000000000D+03	0.2046650966265D+00
0.382000000000D+03	0.2034659419483D+00
0.383000000000D+03	0.2022706975959D+00
0.384000000000D+03	0.2010793773114D+00
0.385000000000D+03	0.1998919945751D+00
0.386000000000D+03	0.1987085626064D+00
0.387000000000D+03	0.1975290943646D+00
0.388000000000D+03	0.1963536025493D+00
0.389000000000D+03	0.1951820996018D+00
0.390000000000D+03	0.1940145977053D+00
0.391000000000D+03	0.1928511087858D+00
0.392000000000D+03	0.1916916445132D+00
0.393000000000D+03	0.1905362163017D+00
0.394000000000D+03	0.1893848353110D+00
0.395000000000D+03	0.1882375124469D+00
0.396000000000D+03	0.1870942583623D+00
0.397000000000D+03	0.1859550834580D+00
0.398000000000D+03	0.1848199978838D+00
0.399000000000D+03	0.1836890115390D+00
0.400000000000D+03	0.1825621340737D+00
0.401000000000D+03	0.1814393748894D+00
0.402000000000D+03	0.1803207431405D+00
0.403000000000D+03	0.1792062477344D+00
0.404000000000D+03	0.1780958973335D+00
0.405000000000D+03	0.1769897003552D+00
0.406000000000D+03	0.1758876649737D+00
0.407000000000D+03	0.1747897991203D+00
0.408000000000D+03	0.1736961104851D+00
0.409000000000D+03	0.1726066065177D+00
0.410000000000D+03	0.1715212944279D+00
0.411000000000D+03	0.1704401811876D+00
0.412000000000D+03	0.1693632735311D+00
0.413000000000D+03	0.1682905779566D+00
0.414000000000D+03	0.1672221007272D+00
0.415000000000D+03	0.1661578478718D+00
0.416000000000D+03	0.1650978251865D+00
0.417000000000D+03	0.1640420382357D+00
0.418000000000D+03	0.1629904923530D+00
0.419000000000D+03	0.1619431926426D+00
0.420000000000D+03	0.1609001439804D+00
0.421000000000D+03	0.1598613510150D+00
0.422000000000D+03	0.1588268181691D+00
0.423000000000D+03	0.1577965496405D+00
0.424000000000D+03	0.1567705494036D+00
0.425000000000D+03	0.1557488212101D+00
0.426000000000D+03	0.1547313685907D+00
0.427000000000D+03	0.1537181948561D+00
0.428000000000D+03	0.1527093030982D+00
0.429000000000D+03	0.1517046961917D+00
0.430000000000D+03	0.1507043767946D+00
0.431000000000D+03	0.1497083473503D+00
0.432000000000D+03	0.1487166100883D+00
0.433000000000D+03	0.1477291670257D+00
0.434000000000D+03	0.1467460199685D+00
0.435000000000D+03	0.1457671705128D+00
0.436000000000D+03	0.1447926200462D+00
0.437000000000D+03	0.1438223697489D+00
0.438000000000D+03	0.1428564205954D+00
0.439000000000D+03	0.1418947733552D+00
0.440000000000D+03	0.1409374285949D+00
0.441000000000D+03	0.1399843866790D+00
0.442000000000D+03	0.1390356477712D+00
0.443000000000D+03	0.1380912118362D+00
0.444000000000D+03	0.1371510786406D+00
0.445000000000D+03	0.1362152477545D+00
0.446000000000D+03	0.1352837185528D+00
0.447000000000D+03	0.1343564902166D+00
0.448000000000D+03	0.1334335617344D+00
0.449000000000D+03	0.1325149319039D+00

0.450000000000D+03	0.131600599332 8D+00
0.451000000000D+03	0.1306905624408D+00
0.452000000000D+03	0.1297848194604D+00
0.453000000000D+03	0.1288833684389D+00
0.454000000000D+03	0.1279862072394D+00
0.455000000000D+03	0.1270933335422D+00
0.456000000000D+03	0.1262047448465D+00
0.457000000000D+03	0.1253204384716D+00
0.458000000000D+03	0.1244404115582D+00
0.459000000000D+03	0.1235646610704D+00
0.460000000000D+03	0.1226931837962D+00
0.461000000000D+03	0.1218259763500D+00
0.462000000000D+03	0.1209630351731D+00
0.463000000000D+03	0.1201043565357D+00
0.464000000000D+03	0.1192499365382D+00
0.465000000000D+03	0.1183997711125D+00
0.466000000000D+03	0.1175538560237D+00
0.467000000000D+03	0.1167121868713D+00
0.468000000000D+03	0.1158747590909D+00
0.469000000000D+03	0.1150415679556D+00
0.470000000000D+03	0.1142126085770D+00
0.471000000000D+03	0.1133878759075D+00
0.472000000000D+03	0.1125673647410D+00
0.473000000000D+03	0.1117510697147D+00
0.474000000000D+03	0.1109389853108D+00
0.475000000000D+03	0.1101311058574D+00
0.476000000000D+03	0.1093274255303D+00
0.477000000000D+03	0.1085279383544D+00
0.478000000000D+03	0.1077326382055D+00
0.479000000000D+03	0.1069415188111D+00
0.480000000000D+03	0.1061545737523D+00
0.481000000000D+03	0.1053717964654D+00
0.482000000000D+03	0.1045931802429D+00
0.483000000000D+03	0.1038187182355D+00
0.484000000000D+03	0.1030484034532D+00
0.485000000000D+03	0.1022822287667D+00
0.486000000000D+03	0.1015201869094D+00
0.487000000000D+03	0.1007622704782D+00
0.488000000000D+03	0.1000084719356D+00
0.489000000000D+03	0.9925878361054D-01
0.490000000000D+03	0.9851319770041D-01
0.491000000000D+03	0.9777170627220D-01
0.492000000000D+03	0.9703430126409D-01
0.493000000000D+03	0.9630097448686D-01
0.494000000000D+03	0.9557171762539D-01
0.495000000000D+03	0.9484652224011D-01
0.496000000000D+03	0.9412537976844D-01
0.497000000000D+03	0.9340828152630D-01
0.498000000000D+03	0.9269521870950D-01
0.499000000000D+03	0.9198618239525D-01

APPENDIX D: STATISTICS AND DATA CONVERSIONS

D.1 CHECK

Check is used for quickly defining the basic statistics (minimum, maximum, average, difference between adjacent points and variance) of an individual pass. If any one of the parameters is outside of the acceptable limits, then the pass number and all parameters are written to the screen. Rerunning this program several times while varying the cutoff limits helps to assess the general quality of the input data.

D.2 STATMAT

This program finds the standard statistics of any of the grids of Chapter III. If more than one grid is input, then the correlation coefficients between all possible map-to-map comparisons are also found. **Statmat** should be used to determine the similarities and differences between dawn and dusk maps and between continued maps.

D.3 PART2

This program was written by Dr. Gary P. Murdock and is used to convert the three Investigator-B tapes supplied by NASA on an IBM platform to a Digital Equipment Corporation (DEC) platform. The program converts IBM text and fixed point and floating point numbers to their respective representation on the DEC machine. Similar code can be written to convert IBM values to other platforms.

```
      program check
      real*4 mean,maxval,minval,diff,maxdiff,total,
     >       maxmax,minmin,meanmax,diffmax,dummy,xmean,maxvar
      double precision passord(4000,2),ra(4000)
      character*80 filename
      character*4 choice,test
      dimension idata(1500,2),data(1500,27)
      integer*4 count,var,countall,type,row,col,zero,pass,eight,
     >          recnum,passnum(4000),passmjd(4000,2)
      common /hsort/ ra
c
c----------------------------------- program description
c
c      check locates all passes with a minimum below a user defined
c      value, a maximum above a user defined value, a variance above
c      a user defined maximum, a mean value beyond a user defined value
c      and a difference between adjacent observations greater than a
c      user defined value.  if a pass is selected, then the pass number
c      and the above values are written to the screen.  this program
c      can be used on direct access 2i-27r files or on files with a
c      header and one variable per pass.  this program is used to find
c      passes which are influenced by external fields as noted by
c      their variance properties.
c
c        program date: 22 apr 91
c
c        updates: 6 jun 92; added sort subroutine
c          NOTE: check now removes the checked passes (ie. the high
c                variance passes) from the ordered pass number file
c                written by reorder.
c          NOTE: now check also orders the output file according to
c                the average elevation of the pass.
c          NOTE: these new options are not available for 2i-27r input.
c                 20 jul 92; added output file on unit 21
c          NOTE: this update simplifies the useage of check
c
c
      write (*,*) '0 IF INPUT FILE IS 2I-27R'
      write (*,*) '1 IF INPUT FILE IS HEADER AND VARIABLE'
      read (*,*) type
      if (type) 50,50,200
c
 50   write (*,*) 'INPUT 2I-27R  FILE:'
      read (*,9990) filename
 9990 format (a80)
      open (10, file=filename, status='old',form='unformatted',
     >      access='direct',recl=116)
c---------------------------------- recl=29 for a dec3100
      write (*,*) 'WHICH VARIABLE DO YOU WANT FROM THE 27'
      write (*,*) 'lat lon rad mlt invlat diplat bs bv x y z'
      write (*,*) 'bva xa ya za totfld xfld yfld zfld inc dec'
      write (*,*) 'totmag totavgmag resid resavgmag ringcur sec'
      read (*,*) var
      write (*,*) 'MAXIMUM ABSOLUTE VALUE FOR MEAN WITHOUT TELLING YOU'
      write (*,*) 'MAXIMUM VALUE FOR DIFFERENCE'
      write (*,*) 'MAXIMUM VALUE FOR VARIABLE'
      write (*,*) 'MINIMUM VALUE FOR VARIABLE'
      write (*,*) 'MAXIMUM VARIANCE FOR VARIABLE'
      read (*,*) meanmax,diffmax,maxmax,minmin,maxvar
c
      write (*,*)'PASSNO CNT     MAX               MEAN           MIN',
     >           '        DIFF    MAXVAR'
c
 21   jstop=0
      recnum=1
      countall=0
      count=0
      read (10,rec=recnum) (idata(1,i),i=1,2),(data(1,j),j=1,27)
 22   n=2
 23   recnum=recnum+1
      read (10,rec=recnum,err=24) (idata(n,i),i=1,2),(data(n,j),j=1,27)
      if (idata(n,1) .ne. idata(n-1,1)) go to 25
      n=n+1
      go to 23
 24   continue
      jstop=1
 25   continue
c
      countall=countall+1
      total=0.0
      xsumsqr=0.0
      maxval=-10e10
      minval=10e10
      maxdiff=10e-10
      dummy=data(n,var)
      data(n,var)=data(n-1,var)
      do 100 i=1,n-1
         total=total+data(i,var)
         xsumsqr=xsumsqr+(data(i,var)**2)
         maxval=max(maxval,data(i,var))
```

```
         minval=min(minval,data(i,var))
         diff=data(i,var)-data(i+1,var)
         maxdiff=max(maxdiff,abs(diff))
100   continue
      mean=total/(n-1)
      xvar=(xsumsqr-(((total)**2)/real(n-1)))/real(n-2)
      absmean=abs(mean)
      if (absmean.gt.meanmax .or. maxval.gt.maxmax .or.
     >    minval.lt.minmin .or. maxdiff.gt.diffmax .or.
     >    xvar.gt.maxvar) then
         write (*,9992) idata(n-1,1),n-1,maxval,mean,minval,
     >                           maxdiff,xvar
9992     format (2i5,5(1x,f12.6))
         count=count+1
      endif
      data(n,var)=dummy
c
 27   continue
      do 28 i=1,2
         idata(1,i)=idata(n,i)
 28   continue
      do 29 j=1,27
         data(1,j)=data(n,j)
 29   continue
      if (jstop .eq. 1) go to 999
      go to 22
c
 200  continue
      countall=0
      count=0
      write (*,*) 'INPUT HEADER AND ONE VARIABLE FILE:'
      read (*,9990) filename
      open (10, file=filename, status='old',form='unformatted')
      write (*,*) '1 to work with magnetic variable or latitude'
      write (*,*) '2 to work with longitude'
      write (*,*) '3 to work with radius'
      read (*,*) var
      write (*,*) 'MAXIMUM ABSOLUTE VALUE FOR MEAN WITHOUT TELLING YOU'
      write (*,*) 'MAXIMUM VALUE FOR DIFFERENCE'
      write (*,*) 'MAXIMUM VALUE FOR VARIABLE'
      write (*,*) 'MINIMUM VALUE FOR VARIABLE'
      write (*,*) 'MAXIMUM VALUE FOR VARIANCE'
      read (*,*) meanmax,diffmax,maxmax,minmin,maxvar
      write (*,*) 'REMOVE THESE PASSES FROM THE ORDERED PASS FILE'
      write (*,*) 'yes OR no (choose yes after running several times'
      write (*,*) '            to determine the variance cutoff)'
      read (*,9991) choice
 9991 format (a4)
      if (choice.eq.'yes' .or. choice.eq.'YES') then
         write (*,*) 'INPUT FILE OF ORDERED PASS NUMBERS'
         read (*,9990) filename
         open (11, file=filename,form='formatted',status='old')
         write (*,*) 'OUTPUT FILE - PASS NUMBERS - CHECKED NUMBERS'
         read (*,9990) filename
         open (20, file=filename,form='formatted')
         icnt=0
         do i=1,10000
            read (11,9991) test
            icnt=icnt+1
            if (test .eq. ' new') then
               read (11,9991) test
               go to 205
            end if
         end do
 205     icnt=icnt-2
         write (*,*) 'read ',icnt,' passes from pass file'
         write (*,*) 'OUTPUT FILE OF CHECKED PASSES'
         read (*,9990) filename
         open (21, file=filename,form='formatted')
c
         do i=1,icnt
            read (11,*) (passmjd(i,j),j=1,2),(passord(i,j),j=1,2)
         end do
      endif

c
      write (*,*)'PASSNO CNT     MAX                MEAN            MIN',
     >           '                DIFF  MAXVAR'
c
 210  read (10,end=500) row,col,zero,xmean,pass,eight
      do 220 i=1,row
         read (10) (data(i,j),j=1,col)
 220  continue
      countall=countall+1
      total=0.0
      xsumsqr=0.0
      maxval=-10e10
      minval=10e10
      maxdiff=10e-10
      data(row+1,var)=data(row,var)
```

```
      do 240 i=1,row
         total=total+data(i,var)
         xsumsqr=xsumsqr+(data(i,var)**2)
         maxval=max(maxval,data(i,var))
         minval=min(minval,data(i,var))
         diff=data(i,var)-data(i+1,var)
         maxdiff=max(maxdiff,abs(diff))
 240  continue
      mean=total/row
      xvar=(xsumsqr-(((total)**2)/real(row)))/real(row-1)
      absmean=abs(mean)
      if (absmean.gt.meanmax .or. maxval.gt.maxmax .or.
     >    minval.lt.minmin .or. maxdiff.gt.diffmax .or.
     >    xvar.gt.maxvar) then
c
         write (*,9992) pass,row,maxval,mean,minval,maxdiff,xvar
         if (choice.eq.'yes' .or. choice.eq.'YES')
     >      write (21,9992) pass,row,maxval,mean,minval,maxdiff,xvar
c
         count=count+1
         passnum(count)=pass
      endif
      go to 210
c
 500  continue
      if (choice.eq.'yes' .or. choice.eq.'YES') then
         jcnt=0
         do i=1,icnt
            do j=1,count
               if (passmjd(i,1) .eq. passnum(j)) go to 510
            enddo
            jcnt=jcnt+1
            ra(jcnt)=passord(i,2)
 510        continue
         enddo
c
         call sort (jcnt)
c
         do i=1,jcnt
            do j=1,icnt
               if ( ra(i) .eq. passord(j,2) ) then
                  write (20,*) (passmjd(j,jj),jj=1,2),passord(j,1),
     >                         real(passord(j,2))
                  go to 520
               endif
            enddo
 520        continue
         enddo
      endif
 999  continue
      close (10)
      close (11)
      close (20)
      write (*,*) 'total passes counted = ',countall
      write (*,*) 'total passes checked = ',count
      write (*,*) 'total passes written to file = ',jcnt
      stop
      end
c
c
      SUBROUTINE SORT(N)
      double precision ra,rra
      common /hsort/ ra(4000)
c-----------------------------this subroutine is written by the authors
c                             of:    Numerical Recipes (fortran);
c                                    The Art of Scientific Computing
c                                    Cambridge University Press
c                                    1989, p. 230
c                             the routine is referred to as "heapsort"
c     Copyright (C) 1986, 1992 Numerical Recipes Software
c
      L=N/2+1
      IR=N
 100  CONTINUE
      IF(L.GT.1)THEN
         L=L-1
         RRA=RA(L)
      ELSE
         RRA=RA(IR)
         RA(IR)=RA(1)
         IR=IR-1
         IF(IR.EQ.1)THEN
            RA(1)=RRA
            RETURN
         ENDIF
      ENDIF
      I=L
      J=L+L
 200  IF(J.LE.IR)THEN
         IF(J.LT.IR)THEN
```

```
      IF(RA(J).LT.RA(J+1))J=J+1
    ENDIF
    IF(RRA.LT.RA(J))THEN
      RA(I)=RA(J)
      I=J
      J=J+J
    ELSE
      J=IR+1
    ENDIF
    GO TO 200
ENDIF
RA(I)=RRA
GO TO 100
END
```

```
      program statmat
      character*70 filename(5),statfile
      character*5 done
      integer*4 col(5),row(5),count
      real*4 nobss,data(500,500,5),colat(5),long(5),space(5),
     >       xmean,ymean
      dimension xdata(250000),ydata(250000)
c
c-------------------------------- program description
c   statmat defines the basic statistics of the input grids.  see
c   the write statements for the specific values calculated.  also,
c   the code loops through all input grids and calculates the
c   correlation coefficients between all combinations of input
c   data.
c
c
      write (*,*) 'OUTPUT STATISTICS FILE'
      read (*,9990) statfile
      open (25, file=statfile, form='formatted')
c
c----------------read all data into array (row,col,layer or map number)
      i=1
 10   write (*,*) 'INPUT MATRIX WITH TPLOT HEADER'
      read (*,9990) filename(i)
 9990 format (a70)
      open (10, file=filename(i),status='old',form='formatted')
c
      read (10,*) col(i)
      read (10,*) row(i)
      read (10,*) colat(i)
      read (10,*) long(i)
      read (10,*) space(i)
      read (10,*) ((data(j,k,i),k=1,col(i)),j=1,row(i))
      close (10)
      write (*,*) 'ARE YOU THROUGH YET???'
      read (*,9991) done
 9991 format (a5)
      if (done .eq. 'y'  .or.  done .eq. 'yes') go to 50
      i=i+1
      go to 10
c------------------------maximum number of input data sets =count
c                        loops from line 80 to 400 increment through all
c                        maps comparing all possible logical
c                        combinations of map to map
 50   count=i
      write (*,*) 'total number of input data sets =',count
      do 60 i=1,count-1
         if (col(i) .ne. col(i+1)  .or.  row(i) .ne. row(i+1)) then
            write (*,*) filename(i),col(i),row(i)
            write (*,*) filename(i),col(i),row(i)
            write (*,*) 'rows or columns do not match'
            stop 0001
         endif
 60   continue
c
c-------------------------from 200 to write statement of variables is
c                         the statistical calculations using two
c                         references:
c                          1) Davis, Statistics and Data Analysis in
c                             Geology, 2nd ed., 1986 pp. 41
c                          2) Young, Statistical Treatment of Experi-
c                             mental Data, 1962, McGraw Hill, 115-132
c
 80   continue
      do 400 icnt=1,count
         do 400 jcnt=icnt,count
c
            do 100 j=1,row(icnt)
               do 100 k=1,col(icnt)
                  ij=(col(icnt)*(j-1))+k
                  xdata(ij)=data(j,k,icnt)
                  ydata(ij)=data(j,k,jcnt)
 100        continue
c-------------------------loops that sum x, x**2, y, y**2 and xy
      nobs=row(icnt)*col(icnt)
      if (nobs .ne. ij) stop 0002
      nobss=float(nobs)
      xsum=0.0
      xsumsqr=0.0
      ysum=0.0
      ysumsqr=0.0
      sumxy=0.0
      xmax=xdata(1)
      xmin=xmax
      ymax=ydata(1)
      ymin=ymax
      do 240 j4=1,nobs
         xsum=xsum+xdata(j4)
         xsumsqr=xsumsqr+(xdata(j4))**2
         ysum=ysum+ydata(j4)
```

```
         ysumsqr=ysumsqr+(ydata(j4))**2
         sumxy=sumxy+(xdata(j4)*ydata(j4))
         xmax=max(xmax,xdata(j4))
         xmin=min(xmin,xdata(j4))
         ymax=max(ymax,ydata(j4))
         ymin=min(ymin,ydata(j4))
 240  continue
c-----------------------find corrected sum of products, covariance
c                        and corrected sum of squares (x) (y)
c
      xmean=xsum/nobss
      ymean=ysum/nobss
      sumprod=sumxy-((xsum*ysum)/nobss)
      covarxy=sumprod/(nobss-1.0)
      xcsumsqr=xsumsqr-((xsum**2)/nobss)
      ycsumsqr=ysumsqr-((ysum**2)/nobss)
c
c-----------------------find variance, standard deviation for x and y
c
      xvar=xcsumsqr/(nobss-1.0)
      yvar=ycsumsqr/(nobss-1.0)
      xstdev=sqrt(xvar)
      ystdev=sqrt(yvar)
c----------------------find correlation coefficient by Davis method
      corrDxy=covarxy/(xstdev*ystdev)
c----------------------find slopes, intercepts and correlation
c                       coefficient by Young method
      xslope=((nobss*sumxy)-(xsum*ysum))/((nobss*xsumsqr)-xsum**2)
      yslope=((nobss*sumxy)-(xsum*ysum))/((nobss*ysumsqr)-ysum**2)
      xintcpt=((ysum*xsumsqr)-(sumxy*xsum))/((nobss*xsumsqr)-xsum**2)
      yintcpt=((xsum*ysumsqr)-(sumxy*ysum))/((nobss*ysumsqr)-ysum**2)
      corrYxy=sqrt(xslope*yslope)
c
c-----------------------write out this mess for individual pass and
c                        overlapping lengths of passes
c
      write (25,*) 'X = ',filename(icnt)
      write (25,*) 'Y = ',filename(jcnt)
      write (25,9992) xmean,ymean,xvar,yvar,xstdev,ystdev
 9992 format('X MEAN =',f9.3,' Y MEAN =',f9.3,/,
     >       'X VARIANCE=',f9.1,' Y VARIANCE=',f9.1,' X STDEV=',
     >        f8.3,' Y STDEV=',f8.3)
      write (25,9993) covarxy,corrDxy
 9993 format ('COVARIANCE XY=',f9.1,' Davis CORRELATION COEF=',f8.3)
      write (25,9994) xslope,xintcpt,yslope,yintcpt,corrYxy
 9994 format ('X SLOPE=',f8.3,' X INTERCEPT=',f8.3,' Y SLOPE=',
     >         f8.3,' Y INTERCEPT=',f8.3,/,'Young CORRELATION COEF=',
     >         f8.3,)
      write (25,9995) xmax,xmin,ymax,ymin
      write (*,*) corrDxy
 9995 format('X-MAX=',f9.3,' X-MIN=',f9.3,' Y-MAX=',f9.3,
     >       ' Y-MIN=',f9.3,/)
c
c
c----------------------increment to next set of passes
 400  continue
c
 999  continue
      close (10)
      close (25)
      stop
      end
```

```
      program part2
c
c---- convert magsat text from ebcdic to ascii, reorder integer bytes,
c----  and translate ibm real to dec real
c
c    editorial note: this code was supplied quite generously by Dr. Gary P.
c                    Murdock
c
      implicit none
c
c---- parameter storage:
      integer reclen
      parameter (reclen=3024)
c
c---- common storage:
      character*1 ascconv(256)
      common /e2acom/ ascconv
      integer*4 recnum,position
      common /xxyyzz/recnum,position
c
c---- equivalence storage:
      integer*4 inbufi(reclen/4),outbufi(reclen/4)
      character*1 inbufc(reclen),outbufc(reclen)
      equivalence (inbufi,inbufc),(outbufi,outbufc)
c
c---- local storage:
      integer i1,i2
      character*80 filename
      character*1 c1
c
c---- data: (0=no translate, 1=real*4, 2=integer*4, 3=ebcdic)
      integer*2 headtyp(557) /4*2,4*1,2*2,4*1,2*2,6*0,12*1,30*3,3*2,
     > 490*1/
      integer*2 datatyp(756) /5*2,691*1,30*2,30*0/
c
c---- functions:
      integer*4 realconv
c
c---- constants:
      c1 = char(1)
c
      write (*,*) 'input file:'
      read (*,99901) filename
99901 format (a80)
      open (21,file=filename,status='old',
     > access='direct',form='formatted',recl=reclen)
      write (*,*) 'output file:'
      read (*,99901) filename
      open (31,file=filename,
     > access='direct',form='formatted',recl=reclen)
c
      recnum = 1
100   read (21,92101,rec=recnum,err=200) inbufc
92101 format (50000a)
      if (inbufc(4).eq.c1) then
        do position = 1,557
          goto (104,101,102,103),headtyp(position)+1
101       outbufi(position) = realconv(inbufi(position))
          goto 104
102       i1 = position*4
          outbufc(i1-3) = inbufc(i1)
          outbufc(i1-2) = inbufc(i1-1)
          outbufc(i1-1) = inbufc(i1-2)
          outbufc(i1) = inbufc(i1-3)
          goto 104
103       i1 = position*4
          do i2 = i1-3,i1
            outbufc(i2) = ascconv(ichar(inbufc(i2))+1)
          end do
104     end do
      else
        do position = 1,756
          goto (108,105,106,107),datatyp(position)+1
105       outbufi(position) = realconv(inbufi(position))
          goto 108
106       i1 = position*4
          outbufc(i1-3) = inbufc(i1)
          outbufc(i1-2) = inbufc(i1-1)
          outbufc(i1-1) = inbufc(i1-2)
          outbufc(i1) = inbufc(i1-3)
          goto 108
107       i1 = position*4
          do i2 = i1-3,i1
            outbufc(i2) = ascconv(ichar(inbufc(i2))+1)
          end do
108     end do
      end if
      write (31,92101,rec=recnum) outbufc
      recnum = recnum+1
      goto 100
```

```
c
200    stop
       end

       BLOCK DATA EBC2ASC
       INTEGER*4 LOOKUP(64) /
      >    50462976, 2139490716,  193891735,  252579084,  319951120,
      > -2029484643,-1886250728,  522067228,-2088599168,  454494852,
      > -1953855096,  117835148,-1827237488,   76977556,-1684366952,
      >   446567700,-1566466016,-1499093853,  777758887,  556476476,
      > -1414878938,-1347506772,  610120112, 1580935466,-1280168147,
      > -1212762700,  746371512, 1061052197,-1111704646,-1044332610,
      >   591028418,  574433088, 1667391939, 1734763876, -976983704,
      >  -909588538, 1818979018, 1886350957, -859082127, -791687475,
      >  1953726161, 2021095029, -741180807, -673786412, -606414376,
      >  -539042340, -471670304, -404298268, 1128415611, 1195787588,
      >  -370652856, -303240214, 1280002685, 1347374669, -269594031,
      >  -202182160, 1414766428, 1482118741, -168535463, -101124106,
      >   858927408,  926299444,  -67487432,     -66052 /
       COMMON /E2ACOM/ LOOKUP
       END

       INTEGER*4 FUNCTION REALCONV (IBM)
C
       IMPLICIT NONE
C
C---- DUMMY STORAGE:
       INTEGER*4 IBM
C
C---- EQUIVALENCE STORAGE:
       INTEGER*4 IIBM
       CHARACTER*1 CIBM(4)
       EQUIVALENCE (IIBM,CIBM)
       INTEGER*4 IDEC
       CHARACTER*1 CDEC(4)
       EQUIVALENCE (IDEC,CDEC)
C
c---- common storage:
       integer*4 recnum,position
       common /xxyyzz/recnum,position
c
C---- LOCAL STORAGE:
       INTEGER COUNT
       LOGICAL SIGNFLAG
       CHARACTER*1 C0
C
C---- "CONSTANTS"
       C0 = CHAR(0)
C
C---- MOVE ARGUMENT TO EQUIVALENCE AREA
       IIBM = IBM
C
C---- SWITCH MANTISSA BYTES INTO dec
       CDEC(1) = CIBM(4)
       CDEC(2) = CIBM(3)
       CDEC(3) = CIBM(2)
C
C---- ZERO NON-MANTISSA BYTE
       CDEC(4) = C0
C
C---- CHECK FOR 0.0
       IF (CDEC(1).EQ.C0 .AND. CDEC(2).EQ.C0 .AND. CDEC(3).EQ.C0)
      > GOTO 120

C---- SHIFT MANTISSA BITS LEFT UNTIL A 1 IS FOUND, DISCARD THE 1,
C----  KEEP COUNT
       COUNT = 0
100    IDEC = ISHFT(IDEC,1)
       IF (CDEC(4).NE.C0) GOTO 110
       COUNT = COUNT+1
       GOTO 100
C
C---- EXTRACT AND CLEAR SIGN BIT
110    SIGNFLAG = BTEST(IIBM,7)
       IIBM = IBCLR(IIBM,7)
C
C---- CALCULATE NEW EXPONENT
       CIBM(2) = C0
       CIBM(3) = C0
       CIBM(4) = C0
       IIBM = IIBM*4-COUNT-130
       IF (IIBM.GT.255 .OR. IIBM.LT.0) THEN
         write (*,99901) recnum,position
99901    format ('ibm value out of range in',i6,',',i3)
         cdec(1) = c0
         cdec(2) = c0
         cdec(3) = c0
         cdec(4) = c0
         goto 120
       END IF
```

```
C
C---- MERGE DEC SIGN, EXPONENT AND MANTISSA
      CDEC(4) = CIBM(1)
      IDEC = ISHFT(IDEC,-1)
      IF (SIGNFLAG) IDEC = IBSET(IDEC,31)
C
120   REALCONV = IDEC
      RETURN
      END
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Technical Memorandum

4. TITLE AND SUBTITLE

FORTRAN Programs to Process Magsat Data for Lithospheric, External Field, and Residual Core Components

5. FUNDING NUMBERS

921

6. AUTHOR(S)

Douglas E. Alsdorf, Ralph R.B. von Frese and Geodynamics Branch

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Goddard Space Flight Center
Greenbelt, Maryland 20771

8. PERFORMING ORGANIZATION REPORT NUMBER

94B00142

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

TM-104612

11. SUPPLEMENTARY NOTES

Douglas E. Alsdorf and Ralph R.B. von Frese: Department of Geological Sciences, The Ohio State University, Columbus, Ohio.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified–Unlimited
Subject Category 46
Report available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD 21090; (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The FORTRAN programs supplied in this document provide a complete processing package for statistically extracting residual core, external field and lithospheric components in Magsat observations. To process the individual passes: 1) orbits are separated into dawn and dusk local times and by altitude, 2) passes are selected based on the variance of the magnetic field observations after a least-squares fit of the core field is removed from each pass over the study area, and 3) spatially adjacent passes are processed with a Fourier correlation coefficient filter to separate coherent and non-coherent features between neighboring tracks. In the second state of map processing: 1) data from the passes are normalized to a common altitude and gridded into dawn and dusk maps with least squares collocation, 2) dawn and dusk maps are correlated with a Fourier correlation efficient filter to separate coherent and non-coherent features; the coherent features are averaged to produce a total field grid, 3) total field grids from all altitudes are continued to a common altitude, correlation filtered for coherent anomaly features, and subsequently averaged to produce the final total field grid for the study region, and 4) the total field map is differentially reduced to the pole.

14. SUBJECT TERMS

Magsat, Fourier, Geomagnetic

15. NUMBER OF PAGES

196

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited