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

1994
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

ABSTRACT .................................................................................................. ii
I. INTRODUCTION ...................................................................................... 1
II. PASS PROCESSING .................................................................................. 3
III. MAP PROCESSING ................................................................................ 9
IV. CONCLUSIONS ..................................................................................... 15
V. ACKNOWLEDGEMENTS ....................................................................... 15
VI. REFERENCES CITED ............................................................................. 16
Figure 1: Data Processing Flow Chart .................................................... 17
Figure 2: Pass-to-Pass Processing Schematic .......................................... 19
APPENDIX A: DATA EDITING AND COMPUTING REQUIREMENTS
  Computing Environment ............................................................................. A-1
  Helpful UNIX Commands ......................................................................... A-2
  Suggested Improvements ......................................................................... A-4
APPENDIX B: PROFILE PROCESSING
  subcore.f .................................................................................................. B-1
  reorder.f .................................................................................................. B-17
  massage.f ................................................................................................. B-33
  movetrunf.f ............................................................................................. B-48
  fourier1d.f ............................................................................................... B-59
  combine.f ................................................................................................. B-75
  gsfc1283 coefficients ............................................................................... B-86
APPENDIX C: MAP PROCESSING
  collocation.f ........................................................................................... C-1
  fourier2d.f ............................................................................................... C-10
  avgdifres.f ............................................................................................... C-39
  sqrmf.p .................................................................................................... C-42
  inversion.f ............................................................................................... C-45
  rmagcov .................................................................................................. C-67
APPENDIX D: STATISTICS AND DATA CONVERSIONS
  check.f ..................................................................................................... D-2
  statmat.f ................................................................................................. D-7
  part2.f ..................................................................................................... D-10
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) fourierld and 6) combine. Appendix A should be reviewed for initial setup 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:

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

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

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

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 fourierld.

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

folded data ---I----------------- original data -----------------I-------- folded data

The percent of data to be smoothed to zero must satisfy

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

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 ($11=0.08 \times 145$). 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 ($55,56=(256-145) \times 0.5$). 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 fourierld 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.
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.diffl).

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

lithospheric, external, and core components of the south polar geomagnetic field at

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
University, Columbus, OH, 1991.

Goyal, H.K., Statistical Prediction of Satellite Magnetic Anomalies for Geologic
University, Columbus, OH, 1986.

Goyal, H.K., R.R.B. von Frese, W.J. Hinze and D.N. Ravat, Statistical prediction of

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

Art of Scientific computing (FORTRAN version), Cambridge University Press,

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.

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-
**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 parentheses 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 fourierid 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 compiler 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 -IT 3600 shellfile`

The `-lM 10Mw` allocates 10 megawords (80 megabytes) of memory and `-IT 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)
```
output file from message (e.g., dk.var)

1

200.0  200.0  200.0  -200.0  500.0

(thes are suggested values)

yes

output file from reorder (e.g., dk.pass)

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.

exit

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 file2.1ow

UNIX prompt> cp file2 file2.hi

Edit file2.1ow 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 file2.1ow and file2.hi with file1 as follows:

UNIX prompt> cat file2.1ow file1 > file3.1ow

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), mlnlat, maxlat, minlon, maxlon,
seconds(3000), rhead(2228/4)
integer*4 ibuff(3024/4), flag1, flag2, headcnt, recnum,
outnum, dnum, dnum2, dnum3, dnum4,
dndkpass, dncnt, stop1, stop2,
head10, data10, head11, data11, head12, data12, totrecord,

character*80 filename
character*60 aaid

common /dndkdat/ dndata(1500, 26), idndata(1500, 2), idkdata(1500, 2)
common /coeff/ gg(50, 50), ggt(50, 50), ggtt(50, 50), jnum, knum,
theta, phi, elvo, year
common /dndkpass/ dndata(1500, 26), idndata(1500, 26),
common /data/ data(3000, 26), idata(3000, 2)

write (*,*) 'INPUT FIRST DATASET FROM TAPE TO DISC TRANSFER'
read (*,9990) filename
read (*,9990) filename
read (*,9990) filename
read (*,9990) filename
read (*,9990) filename
read (*,9990) filename
read (*,9990) filename
read (*,9990) filename

write (*,*) 'INPUT FIELD MODEL SPHERICAL HARMONIC'
write (*,*) 'COEFFICIENTS (GSFC1283)'
the following lines automatically place the nowant pass numbers in the nowant array, these passes are messed up for one reason or another. I'm sure they could be salvaged, but I'm lazy.

nowant (1) = 909
nowant (2) = 1070
nowant (3) = 1206
nowant (4) = 2602
nowant (5) = 2728
nowant (6) = 2744
nowant (7) = 2791
nowant (8) = 2854
nowant (9) = 3059

nocnt = 9

write (*,9990) filename
open (20, file=filename, form='unformatted', access='DIRECT',
> recl=116)

write (*,9990) filename
open (21, file=filename, form='unformatted', access='DIRECT',
> recl=116)

write (*,9990) filename
open (22, file=filename, form='formatted')

write (*,9990) filename
open (23, file=filename, form='formatted')

READ (13,9926) Jnum, Knum, TtZERO, AsID
9926 FORMAT (21,1x,F6.1,A60)
M1AXN=0
TTEMP=0.
50 READ (13,9928) N, M, GNM, HNM, GTNM, HTNM, GTTIM, HTTIM
9928 FORMAT (213,6F11.4)
IF (N.LT.0) GO TO 80
MNAXN=MAX0 (N, M,MAXN)
Gy(N,M)=GNM
GyT(N,M)=GTNM
GyT(N,M)=GTTIM
TTEMP=MAX0 (TTEMP, ABS(GTNM))
IF (M.LT.0) GO TO 50
Gy(M-1,N)=HN
GyT(M-1,N)=HTNM
GyT(M-1,N)=HTTIM
GO TO 50
80 CONTINUE

dnrec=0
dkrec=0
dtotrecord=0
datacnt=0
headcnt=0
stopl0=0
stopll=0

headl0=0
data10=0
recnum=1

100 num=1
read (10, end=120) rhead
headl0=headl0+1
go to 220

120 stopl0=1

write (*,9) 'done with file one'
write (*,9) 'total headers on file one =',headl0
write (*,9) 'total data sets on file one =',data10
headl0=0
data10=0
write (*,*) 'MINIMUM AND MAXIMUM LATITUDE OF STUDY AREA'
write (*,9) 'INPUT RANGE IS FROM -90.0 TO 90.0'
read (*,*) mlnlat,maxlat
write (*,*) 'MINIMUM AND MAXIMUM LONGITUDE OF STUDY AREA'
write (*,9) 'INPUT RANGE IS FROM -180.0 TO 180.0'
read (*,*), minlon,maxlon

--- the following arrays store the spherical harmonic coefficients that describe the core field ---

READ (13,9926) Jnum, Knum, TtZERO, AsID
9926 FORMAT (21,1x,F6.1,A60)
M1AXN=0
TTEMP=0.
50 READ (13,9928) N, M, GNM, HNM, GTNM, HTNM, GTTIM, HTTIM
9928 FORMAT (213,6F11.4)
IF (N.LT.0) GO TO 80
MNAXN=MAX0 (N, M,MAXN)
Gy(N,M)=GNM
GyT(N,M)=GTNM
GyT(N,M)=GTTIM
TTEMP=MAX0 (TTEMP, ABS(GTNM))
IF (M.LT.0) GO TO 50
Gy(M-1,N)=HN
GyT(M-1,N)=HTNM
GyT(M-1,N)=HTTIM
GO TO 50
80 CONTINUE

dnrec=0
dkrec=0
dtotrecord=0
datacnt=0
headcnt=0
stopl0=0
stopll=0

headl0=0
data10=0
recnum=1

100 num=1
read (10, end=120) rhead
headl0=headl0+1
go to 220

120 stopl0=1

write (*,9) 'done with file one'
write (*,9) 'total headers on file one =',headl0
write (*,9) 'total data sets on file one =',data10
headl0=0
data10=0
write (*,*) 'MINIMUM AND MAXIMUM LATITUDE OF STUDY AREA'
write (*,9) 'INPUT RANGE IS FROM -90.0 TO 90.0'
read (*,*) mlnlat,maxlat
write (*,*) 'MINIMUM AND MAXIMUM LONGITUDE OF STUDY AREA'
write (*,9) 'INPUT RANGE IS FROM -180.0 TO 180.0'
read (*,*), minlon,maxlon
totrecord=totrecord+recnum
recnum=1

use the go to 999 statement if you only want to input one file at a time

if you only want to Input one file at a time go to 999

num=1
read (11, end=150) rhead
headll=headll+1

125

close (11)
write (**, *) 'done with file two'
write (**, *) 'total headers on file two =', headll
write (**, *) 'total data sets on file two =', datall
headl2=0
data12=0
totrecord=totrecord+recnum
recnum=1

155
num=1
read (12, end=999) rhead
headl2=headl2+1

220
continue

if (flagl .eq. 1) then
if (info .eq. 1) then
write (23, *) lhead(4), (rhead(i), i=23,34)
else
write (23, *) (lhead(1), i=1,4), (rhead(1), i=5,8),
> (lhead(1), i=9,10), (rhead(1), i=11,14),
> (lhead(1), i=15,16), (rhead(1), i=17,34),
> (lhead(1), i=35,64), (rhead(1), i=65,67),
> (rhead(1), i=68,557)
endif
write (22,8880) lhead(4), (rhead(1), i=5,8), (lhead(1), i=15,16)

8880
format (ix,15,4el5.7,216)
dndkpass=lhead(4)
duske=rhead(5)
dusk1=rhead(6)
dawn=rhead(7)
dawn1=rhead(8)
headcnt=headcnt+1
recnum=recnum+1

if (flagl .eq. 1) then
write (**, *) 'problem with flagl -', flagl,
> 'in a data record'
stop
endif

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

these do 250 loops reorder the data from column to row oriented data

j=j+1
if (j .eq. 250) then
i=num, num+29
data(1,i)=rbuff(jj)
j=j+1
endif

num=num+30

if (flag2 .eq. 1) then
innum=num-1

B-3
c--------------------------------------- search the nowant array for passes
That just didn't happen, the
following passes are doubled and
are considered not wanted:
 909,1079,1206,2602,2728,2744,2791,
2854,3059

   do i=1,innum
   if (nowant(i) .eq. idata(1,1))
      write (*,*)'removed pass number ',
         > nowant(i),idata(1,1)
   endif
   do 270 i=1,innum
      data(i,26)=seconds(i)
   continue
270
   c
   call nine (innum,outnum)
   innum=outnum
   c
   call dawn (innum,dnnum,dknum)
   c
   if (dnnum .le. 0) go to 310
   call pfligrf (dnnum,1)
   call corering (dnnum,1)
   do 300 i=1,dnnum
      drrec=drrec+1
      write (20,rec=drrec) (idndata(i,j),j=1,2),
         > (outdawn(i,j),j=1,27)
   continue
300
   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)
   continue
350
   c
   endif
   c
   c-------------- these ugly little go to's get back to a
   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
   c-------------- go back and get another data record
   go to 230
   c
   c
   else if (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
   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
   close (12)
   close (14)
   close (20)
   close (21)
   close (22)
   close (23)
   stop
end
subroutine nine (innum,nlncnt)
    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
  nlinct=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
    if (data(i,1) .eq. 99999.0 .or. data(i,2) .eq. 99999.0 .or.
      data(i,3) .eq. 99999.0 .or. data(i,8) .eq. 99999.0 .or.
      data(i,12) .eq. 99999.0) go to 200
    nlinct=nlinct+1
    do 140 j=1,2
      idata(nlinct,j)=idata(i,j)
    140 continue
  do 150 j=1,26
    data(nlinct,j)=data(i,j)
  150 continue
  200 continue
  return
end

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
  200 continue
  return
end

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
  200 continue
100 continue
elseif (data(i,1).gt.data(i+1,1)) then

dncnt=dncnt+1

remove the first "dawn" data point because in reality this point could actually belong to a dusk profile. a look at the longitudes will prove this point. of course with some extra code this point could be saved -- but hey its only one in a thousand!!

if (dncnt.eq.1) go to 160

J=1,2

140 continue

J=1,26

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

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)

dncnt=dncnt-1

return

end

subroutine pfligrf (innum,indk)

common /dndkdat/ dndata(1500,26),dkdata(1500,26),

> idndata(1500,2),idkdata(1500,2)

common /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)

********************************************************************

PROGRAM PFLIGRF

THIS PROGRAM CALCULATES VALUES OF ALL OF THE FOLLOWING ALONG MAGSAT PROFILES CREATED BY STEPI4 FORTRAN AT PURDUE.

INDEX VALUE
1 pass number
2 TOTAL FIELD
3 X COMPONENT
4 Y COMPONENT
5 Z COMPONENT
6 INCLINATION
7 DECLINATION
8 latitude point to assure correct points are compared

YEAR(I)=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 OBSERVATION POINTS. THEN THE GEOMAGNETIC FIELD OVER THE OBSERVATION POINT IS COMPUTED BY SUBROUTINE FIELDG FOR THE EPOCH SPECIFIED BY THE YEAR-INPUT VARIABLE

TAPE UNITS:
4. (U/I)....DATA FILES CREATED BY STEPI4
7. (U/O)....OUTPUT...WATCH THE ORDER OF VARIABLES

revised 25 AUG 90

this subroutine was modified to read the spherical harmonic coefficients in the main program and transfer those coefficients by a common block. with these modifications, the file holding the coefficients is only read once and not a thousand billion times which decreases total run time on the program. (ok ok, so maybe not a thousand billion times, but only the number of dawn and dusk profiles given to the subroutine.)

along the way i've removed some useless code that would write items to file 6 or any of a number of additional places depending on which format the user supplied, so if the original is desired, it can be found in programs named the same as the subroutines in this program.
C*******************************************************************************
C
C if (ldndk .eq. 1) then
  do 50 i-l,lnnum
    ipassl=ldndate(i,1)
    mj=ldndate(i,2)
    secx=ldndate(i,26)
    theta(i)=ldndate(i,1)
    phi(i)=ldndate(i,3)
    elvo(i)=ldndate(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(365)
      FRACYA = SECX / (3600000.*24.*365.)
      YEAR(I) = 1980.0 + FRACY + FRACYA
    ENDIF
  continue
50
else if (ldndk .eq. 2) then
  do 70 i-l,lnnum
    ipassl=ldkdata(i,1)
    mj=ldkdata(i,2)
    secx=dkdate(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(365)
      FRACYA = SECX / (3600000.*24.*365.)
      YEAR(I) = 1980.0 + FRACY + FRACYA
    ENDIF
  continue
70
endif
C
np=lnnum
LQ=1
CALL FIELDG (0.,0.,0.,0.,50,LQ,Q1,Q2,Q3,Q4)
CALL GEOMAG (NP,IPASSI)
C
return
end
C
SUBROUTINE GEOMAG (NPTS,IPASSI)
COMMON /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)
common /malnfld/ fld(1500,8),dawne,dawnl,duske,duskl,dndkpass
INTEGER*4 IPASSI,dndkpass
*******************************************************************************
THIS SUBROUTINE CALCULATES THE MAGNITUDE, INCLINATION, AND
DECLINATION OF THE GEOMAGNETIC FIELD ON A GRID NTHETA BY
NPHI
*******************************************************************************
THETA, PHI ** ORIGIN OF THE GRID (DEG.)
ELV ** ELEVATION OF GRID (KILD. ABOVE SEA LEVEL)
NTHETA,NPHI ** DIMENSIONS OF THE GRID
NF ** UNIT FILE WHICH WILL STORE THE FIELD
*******************************************************************************
SUBROUTINES USED
** FIELDG ** (NASA)
** FIELD ** (NASA)
*******************************************************************************
LL=0
RD=180./3.14159265
DO 100 I=1,NPTS
  ATHETA = THETA(I)
  APHI = PHI(I)
  ELV = ELVO(I)
  TR = YEAR(I)
  CALL FIELDG (ATHETA,APHI,ELV,TR,50,LL,X,Y,Z,FF)
  H=SQRT(X*X+Y*Y)
  T=SQRT(H*H+Z*Z)
100
*** B-7
SUBROUTINE FIELDG (DLAT, DLONG, ALT, TM, NM, L, X, Y, Z, F)
EQUIVALENCE (SIMIT(i,1), TG(i,1))
COMMON /NASA/ TO(50,50)
COMMON /FLDCC_4/ ST, CT, SPH, CPH, R, NMAX, BT, BP, BR, B
COMMON /coeff/ gg(50,50), ggt(50,50), ggtt(50,50), jnum, knum,
> tzero, aaid, mmax, temp
DIMENSION G(50,50), GT(50, 50), SIMIT(50, 50), GTT(50, 50)
_character*60 AID, aaid
DATA A/0.0/.

*************** FOR DOCUMENTATION OF THIS SUBROUTINE AND SUBROUTINE FIELD SEE :
*NATIONAL SPACE SCIENCE DATA CENTER'S PUBLICATION
**COMPUTATION OF THE MAIN GEOMAGNETIC FIELD
*FROM SPHERICAL HARMONIC EXPANSIONS**
*DATA USERS' NOTE, NSSDC 68-11, MAY 1968
*GODDARD SPACE FLIGHT CENTER, GREENBELT, MD.

***************

DLAT ** LATITUDE IN DEGREES POSITIVE NORTH
DLONG ** LONGITUDE IN DEGREES POSITIVE EAST
ALT ** ELEVATION IN KM (POSITIVE ABOVE, NEGATIVE BELOW
EARTH'S SURFACE)
TM ** EPOCH IN YEARS
NM ** SET TO INTEGER GREATER THAN DEGREE OF EXPANSION
L ** SET TO 1 ON INITIAL DUMMY CALL, SET TO 0 ON SUBSEQUENT
CALLS

SUBROUTINE RETURNS GEOMAGNETIC FIELD DIRECTIONS (X,Y,Z), POSI-
TIVE NORTH, EAST AND DOWN, RESPECTIVELY, AND MAGNITUDE OF TOTAL
FIELD, F---ALL VALUES ARE IN GAMMAS

............................................ from the data statement above, A = 0.0 only and only on the first
call to this subroutine from anywhere within the program. after
the first call, it is seen below that A = 6371.2 for all future
calls during the running of the program
tLAST=0.0

IF(A.EQ.6378.139) IF(L) 210,100,110
IF(A.EQ.6371.2) IF(L) 210,100,110

A=6378.139
A = 6371.2
FLAT=-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
110 continue
j=0
k=jnum
lzero=tzero
aaid=aaid
mnum=mnum
temp=temp
do 120 j=1,mmax
be do 120 i=1,mmax
be g(i,j,k)-qq(i,j,k)
g(i,j,k)-qqt(i,j,k)
gt(i,j,k)-gg(i,j,k)
gtt(i,j,k)-ggtt(i,j,k)
do 120 continue
110 READ (3,260) J,K,TZERO, AID

B-8
I=0
WRITE (7,270) J,K,TZERO, AID
MAXX=0
TMAX=0.
120 READ (3,280) N,M,GNM,HNM,GTNM,HTNM,GTTHNM,HTTHNM
WRITE (7,280) N,M,GNM,HNM,GTNM,HTNM
IF (N.EQ.0) GO TO 130
MAXX=MAXX(N,MAXX)
G(N,M)=GNM
GT(N,M)=GTNM
GTTH(N,M)=GTTHNM
DO 120 M=1,N
MI=M-1
IF (M.EQ.1) GO TO 140
WRITE (7,280) N,M,G(N,M),GT(N,M),GTTH(N,M)
GO TO 120
130 CONTINUE
DO 150 N=2,MAXX
DO 150 M=1,N
MI=M-1
IF (M.EQ.1) GO TO 140
WRITE (7,300) N,M,G(N,M),GT(N,M),GTTH(N,M)
GO TO 150
140 CONTINUE
150 CONTINUE
WRITE (7,320)
IF (TEMP.EQ.0.) L=1
REWIND 3
160 IF (K.NE.0) GO TO 190
SHMIT(1,1)=1.
DO 170 N=2,MAXX
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(M-1,N)
JJ=JJ+1
170 CONTINUE
180 CONTINUE
T=TM-TZERO
DO 200 N=1,MAXX
DO 200 M=1,N
TG(N,M)=G(N,M)*T
GT(N,M)=GT(N,M)*T
IF (M.EQ.1) GO TO 200
G(N-1,M)=G(N-1,M)*T
GT(N-1,M)=GT(N-1,M)*T
GTTH(N-1,M)=GTTH(N-1,M)*T
GO TO 200
200 CONTINUE
TIAST=IM
210 DLATR=DLAT/57.2957795
SINA=SN(DLATR)
RLONG=RLONG/57.2957795
CPH=CO(RLONG)
SPH=SN(RLONG)
IF (J.EQ.0) GO TO 220
R=ALT+6371.2
C=SL
GO TO 230
220 SINA2=SL2
COSL2=1.-SINA2
DEN2=A2-A2*B2*SINA2
DEN=SQRT(DEN2)
FAC=(11+(ALT+DEN)+A2)/((ALT+DEN)+B2)**2
C=SL/SL2
R=SQRT((ALT+B2)*DEN)*(1+4*A2+B2)/DEN2
MAXM=MIN(M(9,MAXO))
CALL FIELD
Y=BY
IF (J) 240,250,240
X-BX
Z-BZ
RETURN
TRANSFORMS FIELD TO GEODETIC DIRECTIONS
B-9
250 SIND=SINLA*ST-SQRT(COSLA2)*CT
   COSD=SQRT(1.0-SIND**2)
   X=BT*COSD-BPSIND
   Z=BT*SIND-BR*COSD
   RETURN

C
260 FORMAT (213,1X,F6.1,A60)
270 FORMAT (213,5X,SHPECH,F7.1,5X,A60)
280 FORMAT (213,6F11.4)
290 FORMAT (6H0,N,M,6X,IFG,10X,HH,6X,2HGT,6X,2HHT,6X,3HGTT,6X,3HHTT/)
300 FORMAT (213,6F11.4)
310 FORMAT (213, F11.4,11X, F11.4,11X,F11.4)
320 FORMAT (///)
END
SUBROUTINE FIELD

COMMON /NASA/ G(50,50)
COMMON /FLDCOM/ ST, CT, SPH, CPH, BT, BP, BR,
   NMAX,
DIMENSION P(50, 50), DP(50, 50), CONST(50, 50), SP(50), CP(50),
   FN(50), FM(50)
DATA P(1,1)/0.0/
IF (P(I,I).EQ.0.0) GO TO 120
IF (P(I,I).EQ.1.0) GO TO 120
IF (P(I,I).EQ.2.0) GO TO 120
IF (P(I,I).GT.3.0) GO TO 120

120 IF (N-M) > 150, 140, 150
   IF (N-M) > 140, 150, 150
   IF (N-M) > 130, 140, 150
   IF (N-M) > 120, 130, 140
   IF (N-M) > 110, 120, 130
   IF (N-M) > 100, 110, 120
   IF (N-M) > 90, 100, 110
   IF (N-M) > 80, 90, 100
   IF (N-M) > 70, 80, 90
   IF (N-M) > 60, 70, 80
   IF (N-M) > 50, 60, 70
   IF (N-M) > 40, 50, 60
   IF (N-M) > 30, 40, 50
   IF (N-M) > 20, 30, 40
   IF (N-M) > 10, 20, 30
   IF (N-M) > 0, 10, 20

   IF (P(I,I).EQ.0.0) GO TO 120
   IF (P(I,I).EQ.1.0) GO TO 120
   IF (P(I,I).EQ.2.0) GO TO 120
   IF (P(I,I).GT.3.0) GO TO 120

   120 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,N)-CT*P(N-1, N-1) -CONST(N, M) *P(N-2, M)
   GO TO 160

   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
   GO TO 180

   180 BP=BP*TEMP*DP(N, M) *AR
   BP=BP/ST
   B=SQRT(BT*BT+BP*BP+BR*BR)
   RETURN
END

SUBROUTINE CORERING (INNUM, IWNDK)

** Description **
This subroutine subtracts the field at each data point and calculates the ring current affect as defined by NASA's formula which uses the I and V values for the entire orbit. The current value is also subtracted to yield the 'residual' value. Since the total field value, the core-field value and the ring current values are written to file, any one value can be obtained at the next processing step.

NOTE: the core field subtraction is not a least squares procedure, the least squares removal is done in program massage
c
pie=3.1415927
radius=6371.2

c if (lndk .eq. 1) then
  do 100 i=1,lnnum
    if (dndata(i,1).ne.fld(i,8)) then
      write (*,*), 'no match between latitudes in corering',
      write (*,*), dndata(i,1),fld(i,8)
      stop
    else if (dndkpass.ne.idndata(i,1).or. dndkpass.ne.fld(i,1)) then
      write (*,*), 'no match between pass numbers in corering',
      write (*,*), dndkpass, idndata(i,1), fld(i,1)
      stop
  endif
  totmag=dndata(i,8)-fld(i,2)
tavgmag=dndata(i,12)-fld(i,2)
delbz=(dawne*sin(dip)-(2.0*dawnt*sin(dip)*
       (radius/dndata(i,3))**3.0))
  delbxx=(-1.0*dawne*cos(dip)-(dawnt*cos(dip)*
       (radius/dndata(i,3))**3.0))
ringcur=sqrt(((fld(i,3)+delbxx)**2.0) + (fld(i,4)**2.0) +
       ((fld(i,5)+delbz)**2.0)) - fld(i,2)
resid=totmag-ringcur
resavgmag=tavgmag-ringcur
do 150 J=1,15
  outdawn(i,J)=dndata(i,J)
  continue
  do 170 J=16,21
    J=J-14
    outdawn(i,J)=fld(i,J)
    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)
  continue
  else if (lndk .eq. 2) then
    do 200 i=1,lnnum
      if (dkdata(i,1).ne.fld(i,8)) then
        write (*,*), 'no match between latitudes in corering',
        write (*,*), dkdata(i,1),fld(i,8)
        stop
      else if (dndkpass.ne.idkdata(i,1).or. dndkpass.ne.fld(i,1)) then
        write (*,*), 'no match between pass numbers in corering',
        write (*,*), dndkpass, idkdata(i,1), fld(i,1)
        stop
      endif
      totmag=dkdata(i,8)-fld(i,2)
tavgmag=dkdata(i,12)-fld(i,2)
delbz=(duske*sin(dip)-(2.0*duskt*sin(dip)*
       (radius/dkdata(i,3))**3.0))
  delbxx=(-1.0*duske*cos(dip)-(duskt*cos(dip)*
       (radius/dkdata(i,3))**3.0))
  ringcur=sqrt(((fld(i,3)+delbxx)**2.0) + (fld(i,4)**2.0) +
       ((fld(i,5)+delbz)**2.0)) - fld(i,2)
resid=totmag-ringcur
resavgmag=tavgmag-ringcur
do 250 J=1,15
  outdusk(i,J)=dkdata(i,J)
  continue
  do 270 J=16,21
    J=J-14
    outdusk(i,J)=fld(i,J)
    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)
  continue
  endif

B-11
c------------------------------------------------------------------------
 c return
 end

c------------------------ this is the driver for direct access
 c
 c program subroutine
 c real*4 rbuff(3024/4),minlat, maxlat, minlon, maxlon,
 c seconds(3000)
 c integer*4 ibuff(3024/4),flag1, flag2, headcnt, nowant,
 c outnum, innum, dnum,
 c > ndnpsass, datacnt, stop1, stop11,
 c > head10, data10, head11, data11, head12, data12, totrecord,
 c > dnrec, drncnt, nowant(50)
 c character*80 filename
 c character*4 cbuf(3024/4)
 c character*60 said
 c equivalence (ibuff, rbuff), (cbuff, rbuff), (flag1, ibuff(1)),
 c (flag2, ibuff(2))
 c common /latlon/ minlat, maxlat, minlon, maxlon
 c common /dndkdat/ dndata(1500,24), dndata(1500, 26),
 c > idndata(1500,2), idndata(1500,2)
 c common /coeff/ gg(50,50), gtt(50,50), gttt(50,50), jnum, knum,
 c > tttsc, said, maxn, temp
 c common /mainfld/ fld(1500, 8), dawn, dawln, duske, duskl, dndkpass
 c common /thatstl/ outdawn(1500,27), outdusk(1500, 27)
 c COMMON /NASA/ G(50, 50)
 c COMMON /FLDCOM/ ST, CT, SPH, R, NMAX, BT, BR, B
 c COMMON /megfld/ 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)
 cc
 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)
 cc
 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)
 cc
 c write (*,*), 'INPUT FIELD MODEL SPHERICAL HARMONIC'
 c write (*,*), 'COEFFICIENTS (GSFCI283)'
 c read (*,9990) filename
 c open (13, file-filename, status='old', form='formatted')
 cc
 c----------------------------------------------------------------------
 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----------------------------------------------------------------------
 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.
 cc
 cc nowant(1)=909
 cc nowant(2)=1079
 cc nowant(3)=1260
 cc nowant(4)=1602
 cc nowant(5)=2728
 cc nowant(6)=2744
 cc nowant(7)=2791
 cc nowant(8)=2854
 cc nowant(9)=3059
 cc nocnt=9
 cc----------------------------------------------------------------------
 cc recli=116 for an ibm rs6000
 cc
 cc write (*,*), 'OUTPUT DAWN DATA FILE OF 2-INTS AND 27-REALS'
 cc read (*,9990) filename
 cc open (20, file-filename, form='unformatted', access='DIRECT',
 cc recli=29)
 cc write (*,*), 'OUTPUT DUSK DATA FILE OF 2-1 AND 27-R'
 cc read (*,9990) filename
 cc open (21, file-filename, form='unformatted', access='DIRECT',
 cc recli=29)
 cc write (*,*), 'OUTPUT HEADERS FILE'
 cc read (*,9990) filename
 cc open (22, file-filename, form='formatted')
write (*,*) '0 FOR NO ADDITIONAL INFORMATION'
write (*,*) '1 FOR ONLY Card 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 (*,9950) filename
open (23, file=filename,form='formatted')
endif
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
the following arrays store the spherical harmonic coefficients
that describe the core field
READ (13,9926) Jnum,Knum, TtZERO, AalD
READ (13, 9928) N, M, G_4, H_, GTNM, HTNM, GTTNM, HTTNM
IF (N. LE.0) GO TO 80
MAXN = MAXN(N, M_AXN) 
Gg (N, M) =GNM 
G_T (N, H) =GTNH 
GgTT (N, M) =CTempo (GTNM)
IF (M.Eq.1) GO TO 50
Gg (M-l, N} =HNM 
GgT (M-l, N) =HTNM 
GO TO 50
CONTINUE
dnrec=0
dkrec=0
totrecord=0
datacnt=0
headcnt=0
stop10=0
stop11=0
head10=0
data10=0
recnum=1
num=1
read (10,rec=recnum,err=120) rbuff
go to 220
write (*,*) 'done with file one'
write (*,*) 'total headers on file one =',head10
write (*,*) 'total data sets on file one =',data10
head10=0
data10=0
totrecord=totrecord+recnum
recnum=1
100 num=1
110 read (10,rec=recnum,err=120) rbuff
go to 220
write (*,*) 'done with file two'
write (*,*) 'total headers on file two =',head11
write (*,*) 'total data sets on file two =',data11
head11=0
data11=0
totrecord=totrecord+recnum
recnum=1
125 num=1
130 read (11,rec=recnum,err=150) rbuff
go to 220
150 stop1=0
write (*,*) 'done with file three'
write (*,*) 'total headers on file three =',head20
write (*,*) 'total data sets on file three =',data20
head20=0
data20=0
totrecord=totrecord+recnum
recnum=1
155 num=1
160 read (12,rec=recnum,err=999) rbuff
continuing
write (*,*) 'total headers on file four =',head21
write (*,*) 'total data sets on file four =',data21
head21=0
data21=0
totrecord=totrecord+recnum
recnum=1
175 num=1
180 read (13,rec=recnum,err=999) rbuff
continuing
write (*,*) 'total headers on file five =',head22
write (*,*) 'total data sets on file five =',data22
head22=0
data22=0
totrecord=totrecord+recnum
recnum=1
190 num=1
200 read (14,rec=recnum,err=999) rbuff
continuing
write (*,*) 'total headers on file six =',head23
write (*,*) 'total data sets on file six =',data23
head23=0
data23=0
totrecord=totrecord+recnum
recnum=1
210 num=1
220 continue
if (flag1 .eq. 1) then
if (info .eq. 0) then
write (23,*) 'done with file seven'
write (23,*) 'total headers on file seven =',head30
write (23,*) 'total data sets on file seven =',data30
print *, 'flag1=', flag1
endif
else if flag1 .eq. 2 then
write (23,*) 'done with file eight'
write (23,*) 'total headers on file eight =',head31
write (23,*) 'total data sets on file eight =',data31
endif
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=68,697).
c endlf

c 225 continue

c write (22,8880) ibuff(4), (rbuff(i), i=5,8), (ibuff(i), i=15,16)
c 8880 format (lx,15,4,e15.7,216)
c dndkpaas-lbuff (4)
c duelke-rbuff (5)
c dawne-rbuff (7)
c dawnl-rbuff (8)
c headcnt-leadingcnt+l

c recn_-recn_+1

c if (stop10 .eq. 0 .and. stop11 .eq. 0) then

c headl0-headl0+1

c go to 100

c elseif (stop10 .eq. 1 .and. stop11 .eq. 0) then

c headll-headl1+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

c endif

c elseif (flag1 .eq. 2) then this is a data file

c do 230 i-num, num+29

c

c i-data(i, 1)=ibuff(5)
c i-data(i, 2)=ibuff(3)
c xnum=real(i-num)
c seconds(i)=(real(ibuff(4)) + (rbuff(6)*xnum))
c 230 continue

ccc- these do 250 loops reorder the data from column to row oriented data

cjj=7

c do 250 jj=1,25

c do 250 i=num, num+29

c data(i,j)=rbuff(jj)
c jj=jj+1

c 250 continue

c num=num+30

c if (flag2 .eq. 1) then

c the next record is a header and the data information is complete for this orbit

c innum=num-1

c endif

cif (flag1 .eq. 2 .and. flag2 .eq. 1) then

c endif

c do 1=1, nocnt

c if (nowant(i) .eq. i-data(i,1)) then

c write ('*', '*') 'removed pass number',
c nowant(i), i-data(i,1)
c go to 400

c enddo

c do 270 i=1, innum

c data(i,26)=seconds(i)
c 270 continue

c call nine (innum, outnum)
c innum=outnum

c call area (innum, outnum)
c call dawnl (innum, dnnun, dknun)
c if (dnnun .le. 0) go to 310

c call pfligrf (dnnun, 1)
c call coreing (dnnun, 1)
c do 300 l=1,dnnun

c datarec=datarec+l

c write (20, rec=datarec) (idkdata(l, j), j=1,2),
c (outdawn(l, j), j=1,27)
c 300 continue

c 310 continue

c if (dknun .le. 0) go to 360

c call pfligrf (dknun, 2)
c call coreing (dknum, 2)
c do 350 l=1, dknun

c datarec=datarec+l

c write (21, rec=datarec) (idkdata(l, j), j=1,2),

B-14
c 350 continue
cc 360 continue
cc endi
cc
cc 400 recnum=recnum+1
cc datacnt=datacnt+1
cc if (stop10.eq. 0 .and. stop11.eq. 0) then
cc data10=data10+1
cc go to 110
cc elseif (stop10.eq. 1 .and. stop11.eq. 0) then
cc data11=data11+1
cc go to 130
cc elseif (stop10.eq. 1 .and. stop11.eq. 1) then
cc data12=data12+1
cc go to 160
cc endif
cc elseif (flag1.ne. 1 .or. flag1.ne. 2) then
cc write (*,*) 'HOLD THE FORT MAN, BAD FIRST FLAG NUMBER'
c write (*,*) flag1
c endif
cc
cc 999 continue
cc write (*,*), 'total headers on file three =',headl2
cc write (*,*), 'total data sets on file three =',data12
cc write (*,*), 'total headers on tapes =',headcnt
cc write (*,*), 'total data sets on tapes =',datacnt
cc totrecord=totrecord+recnum
cc write (*,*), 'total records read =',totrecord
cc write (*,*), 'total records written to dawn file =',dnrec
cc write (*,*), 'total records written to dusk file =',dkrec
cc 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
cc
cc------------------- add on all subroutines from here on
program reorder
character*80 filename
integer passno(4000,2),date(400,2),store(2),
> countall, jstop, choice, passno(4000), dndx,
> ptcnt(400,2), shrtpas(3000,2), shrtcnt,
> passrec(4000,2), passrem, cnter, denum, pchoice,
> spknum, mincheck, spkvar, nowant(4000), outnum,
> minchk, coordsm, inrec, innum, olf, inf, onf,
> innum2, ptall(4000,2)
real date(400,27), store(27), east, west, diff, hlat,
> 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), cross
common data(400,27), date(400,2)
common /order1/ aver (4000,2), passno (4000,2), cross
common /order2/ passno (4000,2), pntcnt (4000,2), pntall(4000,2)
common /order3/ savglon (4000,2), cross
common /short/ shrtpas (3000,2), shrtcnt, passrec (4000,2)
common /spike/ desdata (400,27), upper, lower, spkvar
common /intb/ intpdata (400,27)

-----------------------------------------------
program description

this program takes data in the 2-integers and 27-reals format
takes in the variable that the user chooses, that variable is
the longitude average of each individual pass, such that after
reorder2, the dataset will have all passes arranged from LOWEST
average longitude (-179.99) to the HIGHEST average longitude
(+179.99). if the dataset crosses the -180.0 180.0 meridian,
the eastern (negative) longitudes are incremented to a
positive value by adding 360.0 (see write statement with
variable "cross"). the sorting variable can also be the
average elevation or the pass numbers of each individual pass
(sorting by pass numbers = sorting by time). this program takes
at a little time (about 15 minutes on the DECstation 3100). the
program requires DIRECT ACCESS or else it just won't happen.
NOTE: the dataset must be
the same because if found two dusk longitude averages to be
EXACTLY the same with real*4. if two averages are the same
then passno(4000) will have the same pass twice which messes
up subroutine reorder2.
NOTE: i usually try to keep all file reads and writes in the
main program but reorder2 and reorder3 are a deviation
from the rule
NOTE: the dataset must be despiked and interpolated to
correctly calculate the longitude averages of the
extended passes. however, i prefer to not write out
despeak or interpolated data because this program
represents the end of the first processing step, after
which the data is ready for more involved processing
(i.e., correlation filtering, bandpassing ...).
therefore, the output from reorder should be the
original data, only reordered. get it??
NOTE: for direct access on an ibm rs6000, recl=116.
ON a dec1000, recl=29

program date: 16 apr 91
write (*) 'INPUT 21-27R 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 (*, dndx)
write (*) '2 TO AVERAGE AND REORDER ON LONGITUDE'
write (*) '3 TO AVERAGE AND REORDER ELEVATION'
write (*) '4 TO REORDER ON PASS NUMBER'
read (*, choice)
if (choice .eq. 1) then
write (*) 'OUTPUT FILE OF 21-27R 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)
eendif
if (choice .gt. 1) then
write (*) 'OUTPUT FILE OF 21-27R DATASET REORDERED'
read (*,9990) filename
open (20, file=filename, form='unformatted', access='direct',
> recl=116)
C-18

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 (*, *) spkn
if (spkn.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-totmag..25-resav_mag'
write (*, *) 'totavg_enc resid resav_mag rangcur sec'
read (*, *) spkvar
endif
c-18
the following if statement determines if
the study area includes the -180.0 180.0
c-longitude line. for further comments see
subroutine reorder1
c
cross=0.0
crossx=cross
minchk=0
c
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
endif
write (*, *) 'the program has determined that this study'
write (*, *) 'area crosses the 180.0, -180.0 meridian'
write (*, *)

endif
write (*, *) 'NORTHERN AND SOUTHERNMOST 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.
c
c
if (north .gt. 83.0) north = 83.0
c
if (south .lt. -83.0) south = -83.0
c
totlat=abs(north-south)
c
minchk=int(((100.0-percent)/(2*100.0))*totlat+1)*3
percent=((100.0-percent)/(2*100.0))*totlat

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
the main program reads the data to find which 21-27r lines belong to a specific pass number (idata(n,1)) and since it reads one line of the next pass it stores that line in memory.

```
c outrec=0
shrtcnt=0
countall=0
jstop=0
passrem=0
```

```
read (10,rec=inrec) (idata(1,i),i=1,2), (data(1,j),j=1,27)
n=2
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
105 continue
   jetop=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
   countall=countall+1
   innum=innum+i
   lnnumall=lnnum
```

```
fILTERS
```

```
if (pchoice .eq. 0) go to 145
   do 143 ii=1,pchoice
      if (idata(lnnum,1) .eq. nowant(ii)) then
         write (*,*) 'PASS NUMBER REMOVED', nowant(ii), idata(lnnum,1)
      endif
   143 continue
145 continue
```

```
if (lnnum .lt. mlncheck) then
   write (*,9980) idata(lnnum,1),lnnum
9980 format ('PASS REMOVED AT READ -',16,' OBSERV COUNT -',15)
go to 400
endif
```

```
call meridian (lnnum,imerpass)
```

```
if (spknum .eq. 0) go to 190
   cnter=0
150 continue
   lnnum=denum
   do 180 k=1,lnnum
      do 180 kk=1,27
         data(k,kk)=desdata(k,kk)
   180 continue
   if (cnter .lt. spknum) go to 150
190 continue
   if (lnnum .lt. mlncheck) then
      write (*,9981) idata(lnnum,1),lnnum
9981 format ('PASS REMOVED AFTER DESPIKING -',16,' OBSERV COUNT -',15)
go to 400
endif
```

```
call interp (dnuk,lnnum,outnum)
```

```
if (lnnum .lt. mlncheck) then
   write (*,9982) idata(lnnum,1),lnnum
9982 format ('PASS REMOVED AFTER INTERPOLATING -',16,' OBSERV COUNT -',15)
```
passrem=passrem+1
go to 400
endif

if (choice .eq. 1) call reorder1
if (choice .gt. 1) go to 260
do 250 l=1,innum
outrec=outrec+1
write (olf,rec-outrec) (idata(l,j),j=1,2),(data(i,j),j=1,27)
250 continue
do 260 l=1,innum
continue
ntnum=cntsome+l
ptcnt(cntsome,1)=data(l,1)
pntall(cntsome,1)=idata(l,1)
pntall(cntsome,2)=lnnumall

----------------------------- subroutine finds all short passes

if (choice .eq. 1) call shorts (lnnum,hilat,lolat,dndk)

now call reorderl which will find
the average longitude and elevation
(not radius) and store then in
aver(400) as well as storing the
pass number and modified julian day for
the current pass.

ok, now go back and get more passes
to average until done with the file

now reread
the file
in reorder2
and write
it
ordered
according
to the pass numbers
given to reorder2.

if (choice .eq. 1) then
inf=21
ont=20
elseif (choice .gt. 1) then
inf=10
ont=20
endif

call reorder2 (cntsome,inf,ont)

----------------------------- if sorting by average longitude, then must

B-20
if (choice .eq. 1) then
  call reorder3 (cntsome, dndk)
  do 600 i=1, cntsome
    do 610 j=1, shrtcnt
      if (savglon(9,1) .eq. passavg(i,1)) then
        passavg(i,2)=savglon(9,2)
        go to 620
      endif
    continue
    continue
    ra(i)=passavg(i,2)
  enddo
  call sort (cntsome, 4)
  do 640 i=1, cntsome
    if (savglon(9,1) .eq. passavg(i,1)) then
      passavg(i,2)=savglon(9,2)
    endif
  continue
  continue
  continue
  call reorder2 (cntsome, 10, 20)
end if
  extend the shorter passes and calculate a
  new average. see subroutine reorder3 for
  more information.

write (*,87) '('new reordering as follows')
write (*,87) passavg(9,1), int(passavg(9,2)), x22elev
write (*,87) 'short passes as follows'
write (*,87) shrtcnt,' short passes as follows'
call reorder2 (cntsome, 10, 20)
write (*,87) total records read for original input file= ', inrec=1
close (i0)
close (20)
close (21)
close (22)
close (25)
stop
end

subroutine meridian (lnum, passnum)
real data(400,27)
integer idata(400,2), lnum, passnum
common data(400,27), idata(400,2)
do 100 i=1, lnum-1
  if (data(i,1) .lt. data(i+1,1)) then
    write (*,87) 'CROSSES =180.0', data(i,1), data(i+1,1)
  endif
  if (data(i,1)=data(i+1,1)) then
    write (*,87) 'CROSSES =180.0', data(i,1), data(i+1,1)
  endif
  if (data(i,1)=data(i,1) .lt. 0.0) data(i,1)=data(i,1)+160.0
continue
go to 200
end if
100 continue
write (*,87) 'total records read = ', innum
write (*,87) 'total records written = ', cntsome
write (*,87) 'total records removed = ', passrem
write (*,87) 'total records considered to be short = ', shrtcnt
write (*,87) 'total records read for original input file = ', inrec=1
close (i0)
close (20)
close (21)
close (22)
close (25)
stop
end

B-21
subroutine reorder1 (nobs, num)

real data (400, 27)
double precision nobss, along (4000), radd (4000), aalong, aavg, selev,
elev (4000), savg, aaver (4000, 2), cross
integer nobs, num, idata (400, 2), passmjd (4000, 2), mjd,
pasnum
common /order1/ aver (4000, 2), passmjd (4000, 2), cross
common data (400, 27), idata (400, 2)

--- subroutine description ---

reorder1 takes a given set of longitudes and elevations and
finds the average longitude and elevation for the set. Since
some longitudes cross from -180.0 to +180.0 (that is, longitudes
cross unless crossing 180) it necessary to correct
the average to the more usual 360 method. Therefore the
dataset is ordered from westernmost longitude to eastern most.
NOTE: real*8 is necessary since on some rare occasions the
averages can be the same at real*4 precision.
NOTE: when the study area includes the -180.0-180.0 longitude
equator (but does not include all other longitudes) it is
necessary to add 360.0 to the negative (or eastern)
longitudes so that eastern longitudes will be located after
the western longitudes.
NOTE: for datasets that are global (ie. polar datasets or the
whole blasted world) then variable 'west' should be input
as -180.0 and variable 'east' should be input as 180.0.
Input as such will produce a map centered on 0.0
longitude.

nobss=dble (nobs)
pasnum=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

aalong=0.0
do 110 n=1,nobs-1
   if (along(n) .lt. along(n+1)) then
      write (*,*) pasnum, '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

--- subroutine reorder2 ---

integer nrecord (4000), passno (4000), npolnts (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)
the basis for this subroutine was provided quite generously by:
Dr. D.R.H. O'Connell
Dept. of Geologicl Sci.
Ohio State University
determine which pass this point belongs to.

if (inf .eq. 10) then
    do 20 i=1,nlines
        passrec(i,1)=passno(i)
    do 30 i=1,nlines
        if (passno(i) .eq. pntall(ii,1)) then
            npoints(i)=pntall(ii,2)
        go to 35
    enddo
    continue
  35 continue
  20 continue
elseif (inf .eq. 21) then
    do 50 i=1,nlines
        passrec(i,1)=passno(i)
    do 40 i=1,nlines
        if (passno(i) .eq. pntcnt(ll,1)) then
            npolnts(i)=pntcnt(ll,2)
        go to 45
    enddo
    continue
  45 continue
  50 continue
endif

npoints = number of records to allocate
for each pass number. nrecord = the output file
record positions for each pass
nrecord(1)=1
passrec(1,2)=1
do 60 i=2,nlines
    li=1
    nrecord(li)=nrecord(li-1)+npoints(li)
    passrec(li,2)=nrecord(li)
  60 continue
read each data point
rewind (inf)
inrec2=0
70 inrec2=inrec2+1
read (inf,rec=inrec2,err=90) (idata(j),j=1,2),(data(k),k=1,27)
determine the matching pass number and its 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)
        nrecord(i)=nrecord(i)+1
    goto 70
  80 continue
enddo
write (*,*) inrec2 = 1, ' TOTAL RECORDS READ FOR FILE',inf
write (*,*) nrecord(nlines)=1, ' TOTAL RECORDS WRITTEN',
> FOR FILE',onf
read next data point
  goto 70
endif
  80 continue
  go to 70
  90 continue
write (*,*) inrec2 = 1, ' TOTAL RECORDS READ FOR FILE',inf
write (*,*) nrecord(nlines)=1, ' TOTAL RECORDS WRITTEN',
> FOR FILE',onf
read each data point
rewind (inf)
inrec2=0
read (inf,rec=inrec2,err=90) (idata(j),j=1,2),(data(k),k=1,27)
determine the matching pass number and its 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)
        nrecord(i)=nrecord(i)+1
    goto 70
  80 continue
enddo
write (*,*) inrec2 = 1, ' TOTAL RECORDS READ FOR FILE',inf
write (*,*) nrecord(nlines)=1, ' TOTAL RECORDS WRITTEN',
> FOR FILE',onf
read next data point
  goto 70
endif
  80 continue
  go to 70
  90 continue

declare subroutine:
SUBROUTINE SORT(N,choice)
double precision ra(4000),aver(4000,2),ras,cross
integer passmjd(4000,2),n,choice
common /order/ aver(4000,2),passmjd(4000,2),cross

the routine is referred to as "heapsort"

SUBROUTINE SORT(N,choice)

Copyright (C) 1986, 1992 Numerical Recipes Software
10  continue
  elseif (choice .eq. 3) then
    do 30 i=l,n
      ra(i) = dble (passmj((i,1))
    30  continue
  endif
  i=IR/2+1
  IR=N
 100  continue
  IF (L.GT.1) THEN
    L=L-I
    RRA=RA(L)
  ELSE
    RA(IR)=RA(1)
    IR=IR-1
    IF (IR.EQ.1) THEN
      RA(1)=RRA
      RETURN
    ENDIF
    I=I+L
    J=J+L
  200  IF (J.LE.IR) THEN
    IF (J.LT.IR) THEN
      IF (RA(J).LT. RA(J+1)) J=J+1
      IF (RRA.LT. RA(J)) THEN
        RA(J)=RRA
        I=J
        J=J+J
      ELSE
        J=IR+I
      ENDIF
      GO TO 200
    ELSE IF (J.EQ.1) THEN
      RA(1)=RRA
      GO TO 100
    ENDIF
  ENDIF
  return
end

subroutine shorts (innum, hilat, lolat, dndk)
  integer Innum, Idata(400,2), dndk, shrtcnt, shrtpas(3000,2), pass rec(4000,2)
  real data(400,27), hilat, lolat
  common data(400,27), Idata(400,2)
  common /shorty/ shrtpas(3000,2), shrtcnt, pass rec(4000,2)
--- subroutine description
subroutine shorts determines if the pass is short, a short pass is one which does not extend above the northern-most (hilat) or below the southern-most (lolat).
  if (dndk .eq. 0) then
    if (data(I,1).gt. lolat .or. data(Innum,1).lt. hilat) then
      shrtcnt=shrtcnt+1
      shrtpas(shrtcnt,1)=data(I,1)
      shrtpas(shrtcnt,2)=innum
    endif
  elseif (dndk .eq. 1) then
    if (data(Innum,1).lt. hilat .or. data(I,1).gt. lolat) then
      shrtcnt=shrtcnt+1
      shrtpas(shrtcnt,1)=data(I,1)
      shrtpas(shrtcnt,2)=innum
    endif
  endif
  return
end

subroutine reorder3 (allcnt, dndk)
  integer shrtpas(3000,2), shrtcnt, pass rec(4000,2),
  allcnt, passnum, recnum, row, recnt, rrow, frow,
  isdata(400,2), frec, rrec, rec, pass, ifdata(400,2),
  idata(400,2), humpcnt, minrow, stocount, minfrow,
  dndk, font, ront
  real sdata(400,2), fdata(400,2), rdata(400,2), sftdat, sfdata(400,2),
  fdata(400,2), rdata(400,2), tsfdat, totdiff, totdiff,
  fdiff, rdiff, totsft, totsft, srtdat, stdat(400,2),
  double precision avgstart, avgdiff, avgdiff, avgdiff, average(4000,2), cross
  common /shorty/ shrtpas(4000,2), shrtcnt, pass rec(4000,2)
  common /trunc/ sdata(400,2), fdata(400,2), rdata(400,2),
  sfdata(400,2), fdata(400,2), stdat(400,2),
  rdata(400,2)
common /order3/ savglon(4000,2),crosss

-- subroutine description
this subroutine extends all short passes by adding or subtracting
the average longitude difference from the closest full length
pass, then calculates the true average longitude for the short
passes, and stores that true average in an array. the short passes
are found by subroutine short by with a user defined percentage of total
length to be considered as a long pass. as this percentage is
increased the distance to the closest pass also increases such that
there is more chance for error due to a poor fit of the average
longitude. the closest full length pass is either west or east
of the short pass. if west, then the average difference is added
to the full length pass, however, if east then the average difference is subtracted.
the average differences are then added to the
longitudes of the short pass to create a set of full length
c longitudes which are averaged together to get the true average
longitude. the fundamental principle involved here is that passes
are always parallel and do not actually cross over each other.
therefore, the difference between adjacent passes remains almost
or at least pretty dog-gone close to almost constant.

lcnt=1
50 continue
do 100 i-l,allcnt
if (shrtpas(icnt,1) .eq. passrec(i,1)) then
recnum=passrec(i,2)
pasnum=passrec(i,1)
row=shrtpas(icnt,2)
recnt=i
go to 200
endif
100 continue

200 continue

........................... read in the short pass
do 220 i-l,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 (pasnum .lt. pasnum(row,1)) then
write (*,*) 'wrong pass number in reorder3'
stop
endif
............................. add 360.0 to longitudes that cross the
-180.0 to 180.0 meridian so that
averages are correct
do 230 n-l,row-1
if (sdata(n,2) .lt. sdata(n+1,2)) then
do 235 i-l,row
if (sdata(i,2) .lt. 0.0) sdata(i,2)=sdata(i,2)+360.0
235 continue
230 continue
238 continue

............................. find the starting record number for the
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-l,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
260 continue
frec=nrec
270 continue
............................. find the starting record number for the
nearest west pass
frec=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-l,shrtcnt
if (pass .eq. shrtpas(i,1)) then
numcnt=numcnt-1
if (numcnt .lt. 1) go to 300

B-25
nrec=passrec(numcnt,2)
go to 280
endif
continue
rrec=nrec

c------------------------ calculate the average longitude 
300 continue
avgdif=10000.0
if (frec .gt. 0) then
  l=1
  ifdata(l,1), j=1,2)
  i=1+1
  frec=frec+1
  read (20, rec=frec, err=340) (rdata(l,1), j=1,2)
  if (l(fdata(l,1), ne. rdata(l,1)) go to 340
  frec=frec+1
  i=1+1
  go to 320
continue
rrow=i-1
do 344 n=1, rrow-1
  if (l(fdata(n,2), .lt. fdata(n+1,2)) then
    do 342 i=1, rrow
      if (l(fdata(i,2), .lt. 0.0) fdata(i,2)=fdata(i,2)+360.0
    continue
  endif
  344 continue
  345 continue

c------------------------ truncate the short and forward passes 
to the same length

c call truncate (row, rrow, dndk, minnrow, stocount, passnum, 
ifdata(rrow,1), 1)

totrdif=0.0
totsrt=0.0
do 350 l=1, minnrow
  diff=abs(rtdat(l,2)-rtdat(l,2))
totrdif=totrdif+diff
totsrt=totsrt+rtdat(l,2)
350 continue

c------------------------ calculate the average longitude difference 
avgdif=dble(totrdif/real(minnrow))
endif

c------------------------ repeat the process for the closest west pass
avgdif=10000.0 
if (rec .gt. 0) then
  l=1
  ifdata(l,1), j=1,2)
  i=1+1
  rrec=rec+1
  read (20, rec=rrec, err=340) (rdata(l,1), j=1,2)
  if (l(rdata(l,1), ne. rdata(l,1)) go to 380
  rrec=rrec+1
  i=1+1
  go to 360
380 continue
rrow=i-1
do 384 n=1, rrow-1
  if (l(rdata(n,2), .lt. rdata(n+1,2)) then
    do 382 i=1, rrow
      if (l(rdata(i,2), .lt. 0.0) rdata(i,2)=rdata(i,2)+360.0
    continue
  endif
  384 continue
  385 continue

c call truncate (row, rrow, dndk, minnrow, stocount, passnum, 
irdata(rrow,1), 1)

totrdif=0.0
totsrt=0.0
do 390 l=1, minnrow
  diff=abs(ftpdat(l,2)-ftpdat(l,2))
totrdif=totrdif+diff
totsrt=totsrt+ftpdat(l,2)
390 continue
avgdif=dble(totrdif/real(minnrow))
endif

c------------------------ if the east difference is the smallest
then use the east pass to calculate average longitude of the extended short pass
if (avgdif .lt. avgrdiff) then
do 400 l=1, ftrow
  if (l(ftpdat(l,2), .eq. ftpdat(l,2)) then

B-26


```fortran

! cnt=1
! go to 410
! endif
! continue
!
! if (sftdat(1,2) .lt. ftdata(1,2)) then
! if (cnt=1 .eq. 0) go to 425
! do 420 i=1,cnt-1
! totsft=totsft+(ftdata(1,2)-real(avgfdiff))
! continue
! 420 continue
! 425 continue
! elseif (sftdat(1,2) .gt. ftdata(1,2)) then
! if (minfrow+cnt .eq. frow) go to 435
! do 430 i=minfrow+cnt, frow
! totsft=totsft+(ftdata(1,2)-real(avgfdiff))
! continue
! 430 continue
! 435 continue
! endif
!
! avgslon=dble(totsft)/dble(frow)
! c
!
!------------------------------------------------------------------------------
! repeat the process if the west pass is the closest
!
! elseif (avgfdiff .lt. avgfdiff) then
! do 500 i=1,row
! if (rtdata(i,2) .eq. rdata(i,2)) then
! rcnt=i
! go to 510
! endif
! 500 continue
! 510 continue
!
! if (srtdat(i,2) .lt. rtdata(i,2)) then
! if (rcnt-1 .eq. 0) go to 525
! do 520 i=1,rcnt-1
! totsrt=totsrt+(rdata(i,2)-real(avgfdiff))
! continue
! 520 continue
! 525 continue
!
! elseif (srtdat(i,2) .gt. rtdata(i,2)) then
! if (minrrow+rcnt .eq. rrow) go to 535
! do 530 i=minrrow+rcnt, rrow
! totsrt=totsrt+(rdata(i,2)+real(avgfdiff))
! continue
! 530 continue
!
! endif
!
! avgslon=dble(totsrt)/dble(frow)
! endif
!
!------------------------------------------------------------------------------
! store the average in array
!
! if (avgslon .gt. 180.0) avgslon = avgslon - 360.0
! savglon(icnt,1)=dble(passnum)
! savglon(icnt,2)=avgslon + cross
!
! icnt=icnt+1
! if (icnt .gt. shrtcnt) return
! go to 50
!
!
! end
!
!
```

---

**B-27**
c subroutine

a-scrlpt

lon

c truncate compares the input passes and truncates both
passes to the same overlapping length.
do 20 j=1,xrow
do 20 jj=1,2
  xdata(j,jj)=sdata(j,jj)
20 continue
if (fr) 50,50,60
do 55 j=1,yrow
do 55 jj=1,2
  ydata(j,jj)=rdata(j,jj)
55 continue
go to 80
do 65 j=1,yrow
do 65 jj=1,2
  ydata(j,jj)=fdata(j,jj)
65 continue
80 continue
jrowi=xrow
rowinc=yrow
loops from 90 to 200 increment through the
two input passes and truncate the lengths
to the same length
90 continue
xdata=xdata(jj,1)
ydata=ydata(jj,1)
diff=abs(xdata-ydata)
if (rowii .eq. 0 .or. rowinc .eq. 0) then
    if this happens, then the findgap subroutine from
    movertrunc will have to be implemented to remove
    the appropriate pass. so far, there hasn't been
    any problems. another alternative would be to
    use only the east or west pass instead of
    comparing both to the short pass.
    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)
if pass a (1) matches pass b (inc) at
beginning length then write to xdata and
ydata and race to main program
  if (abs .lt. 0.33) then
    if (fr .eq. -1) then
do 110 li=1,minrow
do 110 kk=1,2
  srtdat(li,kk)=xdata(li,kk)
  rtdata(li,kk)=ydata(li,kk)
110 continue
  elseif (fr .eq. 1) then
do 115 li=1,minrow
do 115 kk=1,2
  sfdat(li,kk)=xdata(li,kk)
  ftdata(li,kk)=ydata(li,kk)
115 continue
endif
return
endif
if pass a no match the b data then find new
c a or b depending on whether or not ascending
or descending order of independent variable
  if (abs .ge. 0.33) then
    stocount=stocount+1
    if this is a dusk pass then will count from
    -90.0 lat degrees toward the equator
    if (dndk .eq. 0) then
      if (xdata(jj,1) .lt. 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
    if this is a dawn pass then will count from
    the equator toward the south pole
    that is decreasing independent variable
  endif
endif
if (xdata(jj,1) .lt. ydata(jj,1)) then
  rowinc=rowinc-1
  do 160 mm-l,rowinc
  ydata(mm,kk)=ydata(mm+1,kk)
  continue
else if (xdata(jj,1) .gt. ydata(jj,1)) then
  rowinc=rowinc+1
  do 170 nn-l,rowinc
  ydata(nn,kk)=ydata(nn-1,kk)
  continue
endif
endif

c go to 90
c end

c subroutine interpl (dndk, num, ii)
  real data(400,27), xdata(27), Intpdata(400,27)
  integer num, Idata(400,2)
  common data(400,27),idata(400,2)
  co_n/inta/ xdata(27)
  co_on/Intb/ intpdata(400,27)

  subroutine description
  the basic concept of this subroutine was provided by:
  Dr. D.N. "Tiku" Raver
  Dept. of Geology
  Purdue University
  this subroutine linearly interpolates ALL 27-r variables by
  basing the interpolation on the latitudes which are interpolated
  at every 0.33 degrees of starting latitude.

  li=0
  xlat=real (int(data(i,1))*100.0)/100.0
  if (dndk .eq. 0) then
    if (xlat .lt. data(l,l)) xlat=xlat + 0.33
    l=l+1
  100 if (xlat.ge.data(l,l) .and. xlat.le.data(l+1,l)) then
    call interp2 (i,xlat)
    xdata(2)=real (int(xdata(2)*100.0))/100.0
    li=li+1
    do 150 j=1,27
    Intpdata(i,l,j)=xdata(j)
    continue
    xlat=xlat - 0.33
    if (xlat .lt. data(num,1)) return
    go to 100
  elseif (xlat .lt. data(i+1,l)) then
    xlat=xlat - 0.33
    if (xlat .lt. data(num,1)) return
    go to 180
  endif
  endif
  go to 100
endif
c else if (dndk .eq. 1) then
  if (xlat .lt. data(l,l)) xlat=xlat - 0.33
  l=l+1
  180 if (xlat.ge.data(l,l) .and. xlat.le.data(l+1,l)) then
    call interp2 (i,xlat)
    xdata(2)=real (int(xdata(2)*100.0))/100.0
    li=li+1
    do 200 j=1,27
    Intpdata(i,l,j)=xdata(j)
    continue
    xlat=xlat - 0.33
    if (xlat .lt. data(num,1)) return
    go to 180
  elseif (xlat .lt. data(l+1,l)) then
    xlat=xlat - 0.33
    if (xlat .lt. data(num,1)) return
    go to 180
  endif
  endif
end subroutine

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)
  co_n/inta/ xdata(27)

  this subroutine is also from Tiku and is
the interpolator (not to confused with the terminator)

do 100 i=1,27
   dffdata(l)=data(inum,l)-data(inum+l,1)
   continue
100
   do 120 i=1,27
      xdata(l)=data(inum,l)+(xlat-data(inum,l))*
      (diffdata(l)/diffdata(l))
   continue
120
   return
   end

subroutine despike (npts,outnum)
real
   data(400,27), desdata(400,27), upper, lower
integer ic(400), outnum, var, idata(400,2)

common /spike/ desdata(400,27), upper, lower, var

this subroutine also provided by Tikku

PROGRAM DESPIKE

PROGRAM DESPIKE REMOVES MOST SPIKES FROM THE INPUT DATA SET.
HOWEVER, FOR BEST RESULTS, IT IS SUGGESTED TO RUN DESPIKE
AT LEAST THREE TIMES --- FOR EXAMPLE:

INPUT1 ----DESPIKE----> OUTPUT1

(OUTPUT1 - INPUT2) ----DESPIKE----> OUTPUT2

(OUTPUT2 - INPUT3) ----DESPIKE----> OUTPUT3.

STILL, AFTER RUNNING DESPIKE THREE TIMES, IT FAILS TO ELIMINATE
ORBITS WITH DISCONTINUOUS RESID VS LATITUDE PROFILES.
PROGRAM DEGAP ATTEMPTS TO TAKE CARE OF SUCH PASSES.

PARAMETERS TO CHECK: "UPPER" AND "LOWER" (IN NANOAMTAS):

IF PROGRAM DESPIKE HAS DETERMINED OBSERVATION N TO BE
A GOOD POINT, IT THEN SETS OUT TO DETERMINE IF POINT N+1
IS A GOOD POINT. IT DOES THIS BY CHECKING THE POINTS
N, N+1, AND N+2. OBSERVATION N+1 WILL BE A GOOD POINT
IF THE RESIDUAL DIFFERENCE BETWEEN POINT N+1 AND THE
POINT ABOVE IT (N OR N+2) IS LESS THAN "UPPER" AND
IF THE RESIDUAL DIFFERENCE BETWEEN IT AND THE POINT
BELOW IT (N+2 OR N) IS GREATER THAN "LOWER".

DO 2 I=1,400
   IC(I)=1
   2 CONTINUE

ARE THE FIRST NEW POINTS SPIKES?

NOTE: DATA(U,23) = RESID(U)

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))
   SXL=ABS(SL1+SL2+SL3+SL4)/4.0
   20 IF (ABS(SL1).GT.(3.0*XSL).OR.ABS(SL1).GT.(ABS(SL1)+3.0*XSL)) IC(I)=0
      IF (IC(I).EQ.0) THEN
         I=I+1
         GO TO 15
      ENDIF

ARE THE MID POINTS SPIKES?

DO 20 J=1,NPTS-2
   SL2=(DATA(J+1, var)-DATA(J+2, var))
   IF (SL2.GT.UPPER.AND.SL2.LT.LOWER) IC(J+1)=0
   IF (SL2.LT.LOWER.AND.SL2.GT.UPPER) IC(J+1)=0
   SL2=SL2
   20 CONTINUE

IS THE LAST POINT A SPIKE?

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
NOBS=0
DO 30 I=1,NPTS
  IF (IC(I).EQ.1) NOBS=NOBS+1
  CONTINUE
C
WRITE(6,*) IDATA(I,I), NOBS
C
outnum=0
DO 35 I=1,NPTS
  IF (IC(I).EQ.1) THEN
    outnum=outnum+1
  DO 32 m=1,27
data = data(outnum,m)-data(I,m)
    continue
C
RETURN
END
program massage
character*80 filename
read data(400,27),intpdata(400,27),movdata(400,27),
> datastore(2),denum,upper,lower,mean,
> varmax,mincc,contdata(400)
integer i,denum,center,spknum,spkvar,movvar,choice,
> vary,nowant(400),col,zero,gfchoice,
> col3,mincheck,gftype,passrem,recnum,finct,winlen1,
> winlen2a,winlen2b
common data(400,27)
/common /move/ intpdata(400,27)
common /move/ movdata(400,27)
common /move/ varmax,mincc,contdata(400,1)
common /move/ winlen2,spkvar,gftype,
common /move/ dstore(27),desdata(400,27),upper, lower,mean,
common /move/ dstore(27),despdata(400,27),upper, lower,mean,
common /move/ x(400),y(400)

--- program description ---

ok, first the name of the program. to be frank, i just couldn't
think of shortened name for despike-moving-average-min-max-
cubic-spline-linear-interpolation... so i just called the
program "massage" because this program massages the data a bit!!
this program takes data in the 2-integers and 27-reals format
and after a bit of work writes the worked over data in either
21-27r or the more usual format of a file of longitudes, latitudes
and radii separated by headers and a file of residuals separated
by headers, the bit of work includes, 1) removing bad data points
characterized by a large change in magnitude from surrounding
data points, i.e: removing "spikes". 2) fitting a moving average
to the despiked dataset. 3) interpolating every 0.33 degrees of
latitude on the moving average (guide function). 4) interpolating
the despiked dataset. interpolating schemes are linear for
the initial run through on a pass, then the gf is a spline.
5) fit and remove a least squares core field from the data.

of course there are several variations on the above scheme which

can be figured out by reading the write (*) statements.

NOTE: if certain passes are NOT wanted, this program will
remove them from the processing.

NOTE: i've found that the bandpass filter works better than
removing a guide function or cubic spline, but u can
do as u like.

NOTE: for optimal results, make use of the least squares
core field removal as applied by massage.

NOTE: for ibm rs6000, recl=116, for dec3100, recl=29

program date: 16 apr 91

write (*,*) 'INPUT 21-27R FILE'
read (*,9990) filename
9990 format (a80)
open (10, file=filename,status='old',form='unformatted',
> access='direct',recl=116)
write (*,*) 'TYPE dawn OR dusk AS APPROPRIATE'
read (*,9991) dndk
9991 format (a10)
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
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(6) (12)
write (*,') '1=lat, 2=LONG ...12=bva, ...23=totavgmag'
write (*,') 'lat lon rad ml invlat diplat bs bv x y z'
write (*,') 'bva xk ya za totfld xfld yfld zfld inc dec'
write (*,') 'totavgmag totavgmag resid resavgmag ringcur sec'
write (*,') '0 IF YOU WANT 21-27R OUTPUT'
read (*,*) vary
write (*,*) 'FIT LEAST SQUARES CORE FIELD TO THIS VARIABLE'
write (*,*) '0 DO NOT FIT CORE FIELD'
write (*,') 'choose 1'
read (*,1) 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
if (gfchoice .eq. 0 .or. gfchoice .eq. 2) then
if (vary .eq. 0) then
  write (*) 'OUTPUT 21-27R 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 (*) 'AND INTERP-LATS, LONGS, RADII'
  read (*,9990) filename
  open (21, file=filename,form='unformatted')
endif
continue
write (*) 'O FOR NO DESPIKING OF DATA SET'
write (*) 'I FOR DESPIKING ONCE'
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 (*) 'lat lon rad mit invlat diplat bx by x y z'
  write (*) 'bxa yxa ztotfild yfild zfld incr dec'
  write (*) 'totmag totavgmag resid resavgmag ringcur sec '
  read (*,*) mowar
endif
if (gfcholce .eq. 2) then
  write (*) 'WHICH VARIABLE TO WORK WITH IN CUBIC SPLINE: (12) '
  write (*) 'lat lon rad mit invlat diplat bx by x y z'
  write (*) 'bxa yxa ztotfild yfild zfld incr dec'
  write (*) 'totmag totavgmag resid resavgmag ringcur sec '
  read (*,*) movvar
endif
if (vary .eq. 0) then
  write (*) 'OUTPUT 21-27R 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 (*) '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 STDDEV 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 (*,*) gfctype
if (gfctype .eq. 1) then
  incwen2=0
  winlen2=0
endif
if (gfctype .eq. 2) then
  write (*) 'WHAT IS THE LENGTH OF THE WINDOW:'
  read (*,*) winlen1
  write (*) 'HOW MANY POINTS TO INCREMENT WINDOW LOCATION:'
  read (*,*) incwen1
  elseif (gfctype .eq. 2) then
  write (*) 'LENGTH OF THE FIRST AVERAGING WINDOW:'
  read (*,*) winlen2a
  write (*) 'LENGTH OF THE MIN-MAX AVERAGING WINDOW:'
  read (*,*) winlen2b
  write (*) 'LENGTH OF THE LAST AVERAGING WINDOW:'
  read (*,*) winlen2d
  write (*) 'HOW MANY POINTS TO SEARCH FROM AN ENDPOINT TO FIND'
endif
write (*,*) 'IF THE MIN OR MAX POINT SHOULD BE REMOVED'
read (*,*) incwen
write (*,*) 'IF MINCC CAN NOT BE MATCHED THEN'
write (*,*) 'WHAT IS THE LENGTH OF THE AVERAGING WINDOW'
read (*,*) winlenl
write (*,*) 'IF MINCC CAN NOT BE MATCHED THEN'
write (*,*) 'HOW MANY POINTS TO INCREMENT WINDOW LOCATION:'
read (*,*) winlenl
endif
write (*,*) 'MAXIMUM VARIANCE WITHOUT FITTING A CUBIC SPLINE'
read (*,*) varmax
write (*,*) 'ORIGINAL DATA'
read (*,*) mincc
endif
write (*,*) 'AND FINALLY - WHAT IS THE MINIMUM NUMBER OF'
write (*,*) 'OBSERVATIONS ALLOWABLE FOR EACH PASS (50)'
read (*,*) mincheck
if (choice .eq. I) then
  do 50 kk=1,4000
    read (11,*,end=55) nowant(kk)
    continue
  continue
  choice=kk-1
endif
read the data and find all lines for each individual pass
noc=0
noocf=0
itercnt=0
passrem=0
countall=0
jstop=0
ifirstcnt=0
iseccnt=0
iendcnt=0
i1cnt=0
i14cnt=0
i16cnt=0
i17cnt=0
i18cnt=0
i19cnt=0
i20cnt=0
i21cnt=0
i22cnt=0
i23cnt=0
i24cnt=0
i25cnt=0
i26cnt=0
i27cnt=0
i28cnt=0
i29cnt=0
i30cnt=0
i31cnt=0
i32cnt=0
i33cnt=0
i34cnt=0
i35cnt=0
i36cnt=0
i37cnt=0
i38cnt=0
i39cnt=0
i40cnt=0
i41cnt=0
i42cnt=0
i43cnt=0
i44cnt=0
i45cnt=0
i46cnt=0
i47cnt=0
i48cnt=0
i49cnt=0
i50cnt=0
recnum=1
read (10,rec=recnum) (idata(l,1),l=1,2), (data(l,j),j=1,27)
n=2
recnum=recnum+1
read (10,rec=recnum,err=110) (idata(n,1),l=1,2), (data(n,j),j=1,27)
if (idata(n,1) .eq. idata(n-1,1)) go to 220
n=n+1
go to 105
n=1
n=1
jstop=1
120 continue
do 130 i=1,2
istore(i)=idata(n,i)
130 continue
do 140 j=1,27
dstore(j)=data(n,j)
140 continue
c
  countall=countall+1
100 --remove passes that cross the -180.0 180.0 meridian
  if (passes are NOT wanted, remove them)
    if (choice .eq. 0) go to 146
    do 63 li=1,choice
      if (idata(li,1) .eq. nowant(li)) then
        c
          write (*,*) 'PASS NUMBER REMOVED ',nowant(li),idata(li,1)
          passrem=passrem+1
        go to 400
        endif
        63 continue
        145 continue
c
      innum=innum-1
      if (innum .lt. mincheck) then
        write (*,9980) idate(1,1),innum
        9980 format ('PASS REMOVED AT READ =',16,' OBSERV COUNT =',15)
        passrem=passrem+1
        go to 400
        endif
        c
        search for passes that cross the
        -180.0 180.0 meridian

B-35
call meridian (innum)

150 call despike (innum,denum)

180 do 180 k=1,innum
    data(k,kk)=despike(k,kk)
    continue

190 if (cnter .lt. spknum) go to 150

continue

if (innum .lt. mincheck) then
    write (*,9981) idata(l,l),innum
    format ("PASS REMOVED AFTER DESPIKING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

do 205 =1,innum
    x(k)=intpdata(l,kk)
    y(k)=intpdata(l,kk)

205 continue

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

do 210 =1,outnum
    if (intpdata(m,2).gt.180.0)
        intpdata(m,2)-intpdata(m,2)-360.0
    write (20) idata(l,l),intpdata(l,1),intpdata(l,2)

210 continue

do 250 =1,outnum
    if (intpdata(i,2).gt.180.0)
        intpdata(i,2)-intpdata(i,2)-360.0
    write (20) idata(l,l),intpdata(l,1),intpdata(l,2)

250 continue

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

do 310 =1,innum
    do 310 kk=1,27
        data(k,kk)=intpdata(k,kk)
    continue

310 continue

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

do 300 =1,innum
    do 300 kk=1,27
        data(k,kk)=intpdata(k,kk)
    continue

300 continue

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

de 310 continue

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

do 300 =1,innum
    do 300 kk=1,27
        data(k,kk)=intpdata(k,kk)
    continue

300 continue

call interp1 (dnxk,innum,outnum)

if (innum .lt. mincheck) then
    write (*,9982) idata(l,l),innum
    format ("PASS REMOVED AFTER INTERPOLATING =",16,
    > ' OBSERV COUNT =",15)
    passrem=passrem+1
    go to 400
endif

if (eight .eq. 8888) then
    write (28) outnum, col, zero, mean, idata(1,i,1), eight
    write (23) outnum, col, zero, mean, idata(1,i,1), eight
    write (24) outnum, col, zero, mean, idata(1,i,1), eight
    do m=1, outnum
        write (28) (strintdat(m)-intpdata(m,vary))
    enddo
elseif (eight .eq. 7777) then
    if (eight .eq. 8888) then
        write (28) outnum, col, zero, mean, idata(1,i,1), eight
        write (23) eight
        go to 400
    endif
endif

if (eight .eq. 8888) then
    write (23) eight
    go to 400
endif

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)
    continue
enddo

370 continue

410 continue

iendcnt,' passes were 2nd order'
write (*,*) i3cnt,' passes were 3rd order'
write (*,*) i4cnt,' passes were 4th order'
write (*,*) i6cnt,' passes were 6th order'
write (*,*) i8cnt,' passes were 8th order'
write (*,*) ibigcnt,' passes were larger than 8th order'
write (*,*) ilowcnt,' passes were below cc of',minc
continue

420 continue

if (jstop .eq. 1) go to 999
continue

c----------------------------- set idata(i,i) and data(i,i) to the
400 continue

data previously read values and go back and
get the rest of the values for the pass

410 continue

data(i,i)=dstore(i)

420 continue

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 (*,*) isastart,' second points were removed'
    write (*,*) ilowcnt,' 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 (*,*) i6cnt,' passes were 6th order'
write (*,*) i8cnt,' passes were 8th order'
write (*,*) ibigcnt,' passes were larger than 8th order'
write (*,*) ilowcnt,' passes were below cc of',minc
continue

999 continue
subroutine meridian (innum)
real data(400,27)
integer innum
common data(400,27)

--- subroutine description
this subroutine determines if a pass crosses the -180.0 180.0
meridian. If a pass does cross, then 360.0 is added to the
negative values so that the interpolation scheme does not
try to interpolate from -180.0 to 180.0 every 0.33 degrees

do 100 i=1,innum-1
   if (data(i,2).lt.data(i+1,2)) then
      do 150 li=1,innum
          if (data(il,2).lt.0.0) data(il,2)=data(il,2)+360.0
      continue
      go to 200
  100 continue

continue
return
end

subroutine interp1 (dndk,num,ll)
real data(400,27),xdata(27),intpdata(400,27)
integer num
character*10 dmcLk
common data(400,27)
common /inta/xdata(27)
common /intb/intpdata(400,27)

--- subroutine description
the basic concept of this subroutine was provided by:
Dr. D.N. "Tiku" Ravat
Dept. of Geology
Purdue University
this subroutine linearly interpolates ALL 27-r variables by
beside the interpolation on the latitudes which are interpolated
at every 0.33 degrees of starting latitude.

ii=0
xlat=real(int(data(i,1)*100.0))/100.0
if (dndk.eq.'dusk') then
   if (xlat.gt.data(i,1)) then
      xlat=xlat-0.33
      i=i+1
   endif
endif
elseif (xlat.lt.data(i+l,1)) then
   call interp2 (1,xlat)
   xdata(2)=real(int(xdata(2)*100.0))/100.0
   ii=ii+l
   do 150 j=1,27
      intpdata(ii,j)=xdata(j)
   continue
   xlat=xlat+0.33
   if (xlat.ge.data(num,1)) return
   go to 100
elseif (xlat.gt.data(num,1)) then
   xlat=xlat-0.33
   i=i+1
else (dndk.eq.'dawn') then
   if (xlat.lt.data(i,1)) then
      xlat=xlat+0.33
      i=i-1
   endif
endif
elseif (xlat.ge.data(i+l,1)) then
   call interp2 (i,xlat)
   xdata(2)=real(int(xdata(2)*100.0))/100.0
   ii=ii+l
   do 200 j=1,27
      intpdata(ii,j)=xdata(j)
   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
else
   go to 180
endif
endif
end
subroutine interp2 (lnum, xlat)
real data(400, 27), diffdata(27), xdata(27)
integer inum
common data(400, 27)
common /inta/ xdata(27)

---- this subroutine is also from Tiku and is the interpolator (not to be confused with the terminator!)

do 100 1-1, 27
  > diffdata(1)=data(lnum,1)-data(lnum+1,1)
continue

do 120 1-1, 27
  > xdata(1)=data(lnum,1)+xlat-data(lnum,1))*
  > (diffdata(1)/diffdata(1))
continue

return
end

SUBROUTINE SPLINE (X, Y, N, YP1, YPN, Y2)
PARAMETER (NHAX=100)
DIMENSION X(N), Y(N), Y2(N), U(NHAMAX)

c this subroutine provided by the authors of Numerical Recipes
The Art of Scientific Computing
Cambridge University Press, 1989

Copyright (C) 1986, 1992 Numerical Recipes Software

IF (YP1.GT.99E30) THEN
  Y2(1)=0.
  U(1)=0.
ELSE
  Y2(1)=0.
  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.
  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
  ON=0.
  UN=0.
ELSE
  ON=0.
  UN=(3./(X(N)-X(N-1)))*(YN-(Y(N)-Y(N-1))/(X(N)-X(N-1))
ENDIF

Y2(N)=ON*U(N-1)+UN

12 CONTINUE
RETURN
END

SUBROUTINE SPLINT (XA, YA, Y2A, N, X, Y)
DIMENSION XA(N), YA(N), Y2A(N)

--- subroutine description

Copyright (C) 1986, 1992 Numerical Recipes Software

KLO=1
KHI=N

IF (KHI-KLO.GT.1) THEN
  K=(KHI+KLO)/2
  IF(KA(K).GT.X) THEN
    KHI=K
  ELSE
    KLO=K
  ENDIF
GOTO 1
ENDIF

N=KA(KHI)-KA(KLO)
IF (N.LT.0.) PAUSE 'Bad XA input.'
A=KA(KHI)-K1/H
B=(K-XA(KLO))/H
Y=YA(KLO)+BYA(KHI)+
  ((A**3-A)*Y2A(KLO)+(B**3-B)*Y2A(KHI))**(H**2)/6.
RETURN
END

--- subroutine moving (nobs,inum,passnum,eight,qfcnt,dndk,
> integer winlen, incwen, nobs, minm,
> wwinlen2, incwen2, var, maxm,
> gftype, passnum, eight, qfcont, subwinlen, strnobs,
> gftype, wwinlen1, wwinlen2a, wwinlen2b
real data (400, 27), movdata (400, 27), minvmax, maxvmax,
> varmax, strdata (400, 27), intpdata (400, 27), mincc,
> x (100), y (100), y2 (100),
> adate (400, 27)
character*10 dndk
common data (400, 27)
common /move/ movdata (400, 27), wwinlen2, incwen2, var, gftype,
> varmax, mincc, incwen1, wwinlen1, wwinlen2a,
> wwinlen2b
common /intb/ intpdata (400, 27)
c subroutine description
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 within the user defined correlation
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 Davis, Statistics and Data Analysis in
c Geology, 2nd ed., 1986 pp. 41
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
nobs=float(nobs)
c find corrected sum of squares and mean
c xsumsqr=xsumsqr-(xsum**2)/nobs
c find variance, standard deviation
c xvar=xsumsqr/(nobss-1.0)
xstdev=sqrt(xvar)
c write out this mess for individual pass
c write (25, 9992) passnum, xvar, xstdev, xmin, xmax
9992 format (16,5f14.5)
c if the variance of the pass is below the user
defined limit then race back on to main program
c if (xvar .le. varmax) then
 eight=7777
 return
gftype=gftype
 gfcn=gt$1$
do 15 l=1, nobs
do 15 j=1, 27
 strdata(l, j)=data(l, j)
15 continue
subwinlen=0
c use a moving average fit
17 if (gftype .eq. 1) then
 itercnt=itercnt+1
 winlen=wwinlen1
 incwen=incwen1
 inum=1
 if (subwinlen .gt. 0) then
 if (incwen .gt. 1) then
 incwen=incwen-subwinlen
 go to 25
 elseif (incwen .lt. winlen) then
 incwen=winlen-subwinlen
 if (winlen .lt. 2) stop 1110
 endif
 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
B-40
xwinlen=real(nobs)/real(ldivide)
iwinadd=int(xwinlen-(real(int(xwinlen))))*real(ldivide)
winlen=int(xwinlen)-incwien
i2 strife=winlen
iadd=0
if (iwinadd.gt.0 .and. iadd.lt.iwinadd) then
  iadd=iadd+1
  winlen=winlen+1
endif
i1=(winlen-1)/2
i3=11
i7=0
if (i1.gt.nobs) then
  i1=nobs/2
  i7=1
endif
avgdat=0.0
avgdat=data(j,var)+avgdat
endo
do 20 j=1,27
  movdata(inum, j)=data(1, j)
20 continue
movdata(inum, var)=avgdat/(real(11))
if (i7.eq.1) go to 100
30 continue
winlen=istrwlen
if (iwinadd.gt.0 .and. iadd.lt.iwinadd) then
  iadd=iadd+1
  winlen=winlen+1
endif
i2=incwen
i3=winlen-i
i2=(i2-12)/2
if (i2.gt.hobs-1) go to 100
avgdat=0.0
inum=inum+1
do 50 j=1,13
  avgdat=avgdat+data(j, var)
50 continue
movdata(inum, j)=data(14, j)
70 continue
movdata(inum, var)=avgdat/(13-i2+1)
go to 30
100 continue
inum=inum+1
avgdat=0.0
avgdat=avgdat+data(j, var)
endo
150 do 170 j=1,27
  movdata(inum, j)=data(nobs, j)
170 continue
movdata(inum, var)=avgdat/(real(11))
c----------or use the minimum and maximum values
          within the window length

c
elseif (gftype.eq.2) then
  itercnt=itercnt+1
  incwien=incwien2
  winlen=winlen2
  winlenb=winlen2b
  if (winlenb.gt.nobs .or. winlenb.gt.nobs .or.
      incwien.gt.nobs .or. (winlens+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(1, var)+avgdat
    enddo
    avgdat=avgdat/real(nobs/2)
    movdata(1, var)=avgdat
    avgdat=0.0
    do i=(nobs/2)+1, nobs
      avgdat=data(1, var)+avgdat
    enddo
    avgdat=avgdat/real(nobs-(nobs/2))
    movdata(2, var)=avgdat
  endif
  iendcnt=iendcnt+1
  inum=2
  go to 700

endif
avdat=0.0
500 do 1=1,winlena
avdat-data(1, var)+avdat
avdat=avdat/real (winlena) do 510 i=1,27
movdata(1, var)=avdat 11=winlena+1
12=nobs-winlenb
13=0
14=nobs-winlenb+1
15=nobs
maxval=1.0e10
minval=1.0e10
500 do 510 i=1,12
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
llow=minm= (winlen-1)/2
lhi=minm= (winlen-1)/2
530 do 530 i=llow, lhi
avdatimin=avdatmin+data(1, var)
540 do 540 j=llow, lhi
avdatmax=avdatmax+data(1, var)
570 continue
elseif (minm .gt. maxm) then
590 do 590 j=llow, lhi
movdata(2+i3, j)=data(maxm, j)
596 movdata(4+i3, var)=avdatmax
590 continue
elseif (dndk .eq. 'dusk') then
600 do 600 i=llow, lhi
movdata(inum-1+13, i)-movdata(inum+13, i)
inum=inum-i
endif
elseif (dndk .eq. 'dawn') then
610 do 610 i=llow, lhi
movdata(inum-1+13, i)-movdata(inum+13, i)
inum=inum-i
endif
if (movdata(2+i3,1) .le. (movdata(2+i3,1)+ (real (incwen) *0.33))) then
ifirstcnt=ifirstcnt+1
600 do 600 j=2,3
d0 600 j=2, 3
600 movdata(j+13,1)+movdata(j+13,1)
inum=inum-1
endif
if (movdata(inum-1+13,1) .ge. (movdata(inum-1+13,1)+ (real (incwen) *0.33))) then
iseccnt=iseccnt+1
d0 610 i=1,27
610 movdata(inum+1+13,1)=movdata(inum+1+13,1)
inum=inum-1
endif
elseif (dndk .eq. 'dawn') then
if (movdata(2+i3,1) .le. (movdata(2+i3,1)+ (real (incwen) *0.33))) then
ifirstcnt=ifirstcnt+1
620 do 620 j=2,3
d0 620 j=2, 3
620 movdata(j+13,1)+movdata(j+13,1)
inum=inum-1
endif
if (movdata(inum-1+13,1) .le. (movdata(inum-1+13,1)+ (real (incwen) *0.33))) then
iseccnt=iseccnt+1

do 630 i=1,27
   movdata(inum-1+i3,1)=movdata(inum+i3,1)
   inum=inum-1
endif
endif

continue
if (dndk .eq. 'dusk') then
   do 401 i=1,inum
   x(i)=movdata(i,1)
   y(i)=movdata(i,2)
endif
continue
elseif (dndk .eq. 'dawn') then
   do 404 i=1,inum
   x(i)=movdata(i,1)
   y(i)=movdata(i,2)
endif
continue
endif
continue
if (dndk .eq. 'dawn') then
   call espline(x, y, inum, oneslope, twoslope, y2)
   do 402 i=1,nobs
      sdata(i,j)=intpdata(i,j)
   endif
continue
endif
continue
if (dndk .eq. 'dawn') then
   call espline(x, y, inum, oneslope, twoslope, y2)
   do 406 i=1,nobs
      do 406 j=1,27
         intpdata(i,j)=sdata(i,j)
      endif
continue
endif
endif

calculate the correlation coefficient between the original data and the cubic spline

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

Continue

find corrected sum of products, covariance and corrected sum of squares (x) (y)

find variance, standard deviation for x and y

find correlation coefficient by Davis method

if (corrDxy .lt. mincc .and. gftype .eq. 2) then
   write ('*','(passnum, pass below cc limit, corrDxy, gftype=1')
endif

B-43
lswitch=lswitch+1
    do 450 l=1,nobs
         do 450 j=l,27
             data(l,j)=strdata(l,j)
        450 continue
    go to 17
endif
if (corrDxy .lt. mincc .and. gftype .eq. 1) then
    write (*,*) passnum,' pass below cc limit',corrDxy
    lloment=lloment+1
    go to 999
endif
subwinlen=subwinlen+1
    do 460 l=1,nobs
         do 460 j=l,27
             data(l,j)=strdata(l,j)
        460 continue
    go to 17
endif
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

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
    this subroutine also provided by Tiku
    PROGRAM DESPIKE
    ********************************************************************
    PROGRAM DESPIKE
    PROGRAM REMOVES MOST SPIKES FROM THE INPUT DATA SET.
    HOWEVER, FOR BEST RESULTS, IT IS SUGGESTED TO RUN DESPIKE
    AT LEAST THREE TIMES --- FOR EXAMPLE:
    INPUT1 ---DESPIKE---> OUTPUT1
    (OUTPUT1 - INPUT2) ---DESPIKE---> OUTPUT2
    (OUTPUT2 - INPUT3) ---DESPIKE---> OUTPUT3.
    STILL, AFTER RUNNING DESPIKE THREE TIMES, IT FAILS TO ELIMINATE
    ORBITS WITH DISCONTINUOUS RESID VS LATITUDE PROFILES.
    PROGRAM DEGAP ATTEMPTS TO TAKE CARE OF SUCH PASSES.
    PARAMETERS TO CHECK: "UPPER" AND "LOWER" (IN NANOTESLAS):
    IF PROGRAM DESPIKE HAS DETERMINED OBSERVATION N TO BE
    A GOOD POINT, IT THEN SETS OUT TO DETERMINE IF POINT N+1
    IS A GOOD POINT. IT DOES THIS BY CHECKING THE POINTS
    N, N+1, AND N+2. OBSERVATION N+1 WILL BE A GOOD POINT
    IF THE RESIDUAL DIFFERENCE BETWEEN POINT N+1 AND THE
    POINT ABOVE IT (N OR N+2) IS LESS THAN "UPPER" AND
    IF THE RESIDUAL DIFFERENCE BETWEEN IT AND THE POINT
    BELOW IT (N+2 OR N) IS GREATER THAN "LOWER".
    ********************************************************************
    DO 2 i=1,400
    IC(i)=i
    2 CONTINUE
    ********************************************************************
    ARE THE FIRST NEW POINTS SPIKES?
    NOTE: DATA(U,23) = RESID2(U)
    ********************************************************************
    15 S1L=(DATA[i+1,VAR]-DATA[i,VAR])
    S1L=(DATA[i+2,VAR]-DATA[i,VAR])
    S1L=(DATA[i+3,VAR]-DATA[i,VAR])
    S1L=(DATA[i+4,VAR]-DATA[i,VAR])
    XSL=ABS(S1L+S1L+S1L)/4.0
    S2=(DATA[i+2,VAR]-DATA[i+1,VAR])
IF(ABS(SLI).GT.3.0*XSL).OR.ABS(SLI).GT.3.0*S2) IC(I)=0
IF(IC(I).EQ.0) THEN
I=I+1
GO TO 15
ENDIF

C*******************************************************************
C ARE THE MID POINTS SPIKES?
C
DO 20 J=1,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 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 NORS=0
DO 30 I=1,NPTS
IF(IC(I).EQ.1) NORS=NORS+1
30 CONTINUE
C
WRITE(6,*) IDATA(I,I), NORS
C
outnum=0
DO 32 J=1,NPTS
IF(IC(J).EQ.1) outnum=outnum+1
do 32 m=1,27
desdata(outnum,m)=data(I,m)
32 continue
C
return
END
C
C subroutine track (nf,lnnum, noc,nocgf)
common /trax/ x(400), y(400)
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)
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
C
C subroutine sgrfit (lnnum,x,ifitvar)
real core(400),xmag(400),intpdata(400,27)
common /intb/ intpdata(400,27)
C
this subroutine 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
In a pass so that the core field model matches the observed values closer.

\[ x = 0.0 \]
\[ c_{\text{mean}} = 0.0 \]
\[ f_{\text{mean}} = 0.0 \]
\[ \text{do } i = 1, \text{innum} \]
\[ \text{core}(i) = \text{input}(i, 16) \]
\[ c_{\text{mean}} = c_{\text{mean}} + \text{core}(i) \]
\[ x_{\text{mag}}(i) = \text{input}(i, 11, \text{ifitvar}) \]
\[ f_{\text{mean}} = f_{\text{mean}} + x_{\text{mag}}(i) \]
\[ \text{enddo} \]
\[ c_{\text{mean}} = c_{\text{mean}} / \text{real}(\text{innum}) \]
\[ f_{\text{mean}} = f_{\text{mean}} / \text{real}(\text{innum}) \]
\[ \text{do } i = 1, \text{innum} \]
\[ x_{\text{mag}}(i) = x_{\text{mag}}(i) - f_{\text{mean}} \]
\[ \text{core}(i) = \text{core}(i) - c_{\text{mean}} \]
\[ \text{enddo} \]
\[ c_{\text{tc}} = 0.0 \]
\[ \text{do } 600 \ i = 1, \text{innum} \]
\[ c_{\text{tc}} = (\text{core}(i) * \text{core}(i)) + c_{\text{tc}} \]
\[ c_{\text{tcinv}} = 1.0 / c_{\text{tc}} \]
\[ c_{\text{tf}} = 0.0 \]
\[ \text{do } 700 \ i = 1, \text{innum} \]
\[ c_{\text{tf}} = (\text{core}(i) * x_{\text{mag}}(i)) + c_{\text{tf}} \]
\[ x = c_{\text{tcinv}} * c_{\text{tf}} \]
\[ \text{do } 800 \ i = 1, \text{innum} \]
\[ \text{input}(i, 11, \text{ifitvar}) = x_{\text{mag}}(i) - (\text{core}(i) * x) \]
\[ \text{continue} \]
\[ \text{return} \]
\[ \text{end} \]
program movetrunc
  integer xrow,xcol,zero,eight,xpassno,ypassno,ycol,
  xmean,xrow,x3row,xcol,x3col,x3pass,yrow,y3row,y3col,y3pass,ycol,
  xcount,nowant,notcount,paircount,paircnt,passr,global,
  outto,nopase,type, nocnt,
  x3mean,x3pass,y3mean,y3pass,xmean,ymean,x3data(400,3),y3data(400,3),ydata(400,4),
  xdata(400),ydata(400),x3data(400,3),y3data(400,3),xdata(400),y(400)
character*80 filename
common /tplot/
x(400),y(400)
or_mon
common /nope/
nwant(4000), nocnt
common /,runs,at/
xdata(400,4), ybdata(400,4)
common /rowcol/
x3row, y3row, xrow,yrow,x3col,y3col,xcol,ycol,
  x3mean,x3pass,y3mean,y3pass,xmean,ymean,x3data(400,3),xldata(400),y3data(400,3),ydata(400)
character*80 filename
program description
  this program truncates, in the time domain, two adjacent passes
  to the same length, this program should be used only after the
  passes have been reordered by program reorder. truncation is
  accurate in the time domain rather than in the frequency domain.
  this program is used before program fourier which is used to
  extract the similar wavelengths of adjacent passes.
  program date: 16 apr 91
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 (*,*), 'IF THE DATA IS GLOBAL OR POLAR'
write (*,*), 'IF THE DATA DOES NOT INCLUDE ALL LONGITUDES'
read (*,9990) filename
write (*,*), 'NOTE: FILE Y WILL HAVE THE FIRST PASS MOVED'
write (*,*), 'TO THE BOTTOM OF THE FILE'
write (*,*), 'NOTE: OUTPUT FILE Y WILL NOT INCLUDE THE LAST PASS'
write (*,*), 'FILE Y WILL NOT INCLUDE THE FIRST PASS'
do
  c
  i=1,4000
  read (26,*,end-15) nowant(i)
  continue
enddo
nocnt=nocnt-1
write (*,*),'MINIMUM NUMBER OF OBSERVATIONS FOR EACH PASS'
read (*,*), minobs
write (*,*),'TYPE OF GAP FINDER (2)'
write (*,*),'2 FOR USING THE MINIMUM OBSERVATIONS'
c-48

PROCEEDING PAGE BLANK NOT FILMED
read (*,*) type
write (*,*) '-----------------------------', 'running through dataset to find passes', 'that do not overlap'

--- subroutine findgap locates passes that do not
  have overlapping segments and removes the
  shorter of the two passes. The subroutine
  continues reading and rereading the dataset
  until all non-overlapping segments are removed
---
call findgap (global,dndk,minobs,type)
write (*,*) 'done with run through'
write (*,*).
write (25,*)'XPASS YPASS CCD CCY XVAR YVAR ',
write (25,*)'CONARY XSTDEV YSTDEV',
write (25,*)'XMEAN YMEAN XSLOPE YSLOPE',
paircnt=0
paircnt1=0
tcount=0
strcnt=0
jstop=0
passrem=0
totpass=0
c30 continue
read (10,end=90) y3row,y3col,zero,y3mean,y3pass,eight
do 35 i=1,y3row
read (10) (y3data(i,ii},li=1,y3col)
continue
read (11) yrow,ycol,zero,ymean,ypassno,eight
do 45 i=1,yrow
read (11) yldata(i)
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
continue
55 continue
go to 95
--- this little jump around is used
to get the last and first passes
---
of global datasets truncated
jstop=1
if (global .eq. i) go to 999
c95 continue
strcnt=strcnt+1
--- offset the data file in subroutine
movrun
if (strcnt .eq. 1) call movrun (-1,outto,jstop)
if (strcnt .gt. 1) call movrun (0,outto,jstop)
if (outto) 30,100,100
--- truncate the passes to the same length
100 continue
nopass=0
call truncate (xrow,yrow,dndk,nobs,passcnt1,nopass, xpassno,ypassno)
if (nopass .gt. 0) then
--- if this happens, then subroutine
findgap didn't work just right
write (*,*) 'OH MAN HAVE YOU GOTT TROUBLE NOW'
stop
endif
--- do a little statistical nonsense
call statistics (nobs,xpassno,ypassno,xmean,ymean)
**Subroutine Description**

This subroutine truncates two sets of values to the same length. Truncation is based on the independent variable which is the latitude of each point along a pass. These values may be interpolated to every 0.33 degrees.

```fortran
    subroutine truncate (xrow, yrow, dndk, minrow, stocount, nopass, xpassno, ypassno)
      integer xrow, yrow, stocount, rowli, rowinc, minrow, nopass, xpassno, ypassno
      real xdata(400,4), ydata(400,4), x3data(400,3), xldata(400), y3data(400,3), yldata(400,4), 
        xdata, ydata, diffab, abss, xdata(J,2), ydata(J,2)
      common /trunstat/ xadata(400,4), ybdata(400,4)
      common /nope/ nowant(4000), nocnt
      common /trunstat/ x3data(400,3), xldata(400), y3data(400,3), yldata(400,4)

    !-------------------- subroutine description
    !
    ! this subroutine truncates two sets of values to the same length.
    ! truncation is based on the independent variable which is the latitude
    ! of each point along a pass. These values may be interpolated to every
    ! 0.33 degrees.
    !
    do 70 j=1,xrow
      xdata(J,1)=x3data(J,1)
      xdata(J,2)=xldata(J)
      xdata(J,3)=xdata(J,2)
      xdata(J,4)=x3data(J,3)
    70 continue
    do 75 j=1,yrow
      ydata(J,1)=y3data(J,1)
      ydata(J,2)=yldata(J)
      ydata(J,3)=ydata(J,2)
      ydata(J,4)=y3data(J,3)
    75 continue
    continue
    stocount=0
    j=1
    row1=xrow
    rowinc=yrow

    !------------------------ loops from 90 to 200 increment through the
    ! two input passes and truncate the lengths
    !
    continue
    xdata=xdata(j,j,1)
    ydata=ydata(j,j,1)
    diffab=abs(abss)
    if (row1 .eq. 0 .or. rowinc .eq. 0) then
      minyrow=min(xrow,yrow)
    if (minyrow .eq. row1) then
```
nopass=xpassno
nowant (nocont+1)=xpassno
write (*,*) 'xrows (ii) =',rowii,' yrows (inc) =',rowinc
write (*,*) 'rerunning to remove x pass number =',xpassno
write (*,*) 'xrow =',xrow,' yrow =',yrow
return

elseif _mlnxyrow.eq.yrow) then
nopass=ypassno
nowant (nocont+l)=ypassno
write (*,*) 'xrows (ii) =',rowii,' yrows (inc) =',rowinc
write (*,*) 'rerunning to remove y pass number =',ypassno
write (*,*) 'xrow =',xrow,' yrow =',yrow
return

endif

mlnrow=mln(rowll,rowinc)
c write (*,*) rowll,rowinc,mlnrow
c write (*,*) adata,bdata,abss

....................
if pass a (ii) matches pass b (inc) at
beginning length then write to xdata and
ydata and race to main program
if (abss .lt. 0.33) then
do 110 ll=1,mlnrow
do 110 kk=1,4
xdata(ll,kk)=xdata(ll,1)
ydata(ll,kk)=ydata(ll,1)
c write (*,*),xdata(ll),ydata(ll)
110 continue
return
endif

.....................
if this is a dusk pass then will count from
c the equator toward the south pole
c that is decreasing independent variable
else (dndk .eq. 1) then
if (xdata(jj,ll) .lt. ydata(jj,ll)) then
rowinc=rowinc+1
do 160 nn=1,rowinc
do 160 kk=1,4
ydata(nn,kk)=ydata(nn+1,kk)
c write (*,*) ydata(nn,kk)
160 continue
enddo
endif

....................
if this is a dawn pass then will count from
c the equator toward the south pole
c that is decreasing independent variable
else (dndk .eq. 0) then
if (xdata(jj,ll) .gt. ydata(jj,ll)) then
rowinc=rowinc+1
do 160 nn=1,rowinc
do 160 kk=1,4
ydata(nn,kk)=ydata(nn+1,kk)
c write (*,*) ydata(nn,kk)
160 continue
endif

subroutine statistics (mlnrow,xpassno,ypassno,xmean,ymean)
integer mlnrow,nobs,xpassno,ypassno
real xdata(400,4),ydata(400,4),nobs
common /trunstat/ xdata(400,4),ydata(400,4)
c
the statistical calculations using two
c references:
1) Davis, Statistics and Data Analysis in
genology, 2nd ed., 1986 pp. 41
c 2) Young, Statistical Treatment of Experi-
mental Data, 1962, McGraw Hill, 115-132

---


do 240 J=1,nobs
xsum=xsum+xadata(J,2)
xsmsqr=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

write (*,*) xsum, ysum, xsumsqr, ysumsqr, sumxy

c find corrected sum of products, covariance and corrected sum of squares (x)(y)

xmean=xsum/nobss
ymean=ysum/nobss

 covariance=sumxy*(xsumsqr-(xsum**2)/nobss*(ysum**2)/nobss)

cyrsxy=sqrt( covariance

xvar=(xsumsqr-(xsum**2)/nobss)
yvar=(ysumsqr-(ysum**2)/nobss)
xstdev=sqrt(xvar)
ystdev=sqrt(yvar)

c find correlation coefficient by Davis method

corrDxy= covariance/(xstdev*ystdev)

c find slopes, intercepts and correlation coefficient by Young method

xslope=((nobss*sumxy)-(xsum*ysum))/((nobss*xsumsqr)-xsum**2)
yslope=((nobss*sumxy)-(xsum*ysum))/((nobss*ysumsqr)-ysum**2)

xlntcpt=((ysum*xsumsqr)-(sumxy*xsum))/((nobss*xsumsqr)-xsum**2)
yintcpt=((xsum*ysumsqr)-(sumxy*ysum))/((nobss*ysumsqr)-ysum**2)

corrYxy=sqrt(xslope*yslope)

---

subroutine movtrun (into, outto, Jstop)
integer into, outto, Jstop,
 analogue ypass, ypassno, ytop,
 analogue xrow, yrow, x3col, ycol,
 analogue yrow, y3row, y3col, xcol,
 analogue y3pass, pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
 analogue x3data(400,3), xdata(400), ydata(400),
 analogue y3mean, ymean, y3mean, ymean3, xrowcol,
 analogue x3row, xrow, x3col, xcol,
 analogue x3pass, xpassno,
c 40 continue
do 50 1=1,y3row
   do 55 il=1,y3col
      savey3(i,i1)=ydata(i,i1)
   continue
   savey(i)=ydata(i)
50 continue
sy3row=y3row
syrow=yrow
sy3col=y3col
sycol=ycol
sy3mean=y3mean
symean=ymeann
sy3pass=y3pass
ypassno=ypassno
s
55 continue
saveyl(i)=ydata(i)
50 continue
sy3row=y3row
syrow=yrow
sy3col=y3col
sycol=ycol
sy3mean=y3mean
symean=ymeann
sy3pass=y3pass
ypassno=ypassno
y
  c 70 1=1,y3row
    do 75 il=1,y3col
       x3data(i,il)=ydata(i,il)
    continue
    xldata(i)=ydata(i)
70 continue
x3row=y3row
xrow=yrow
x3col=y3col
xcol=ycol
x3mean=x3mean
xmean=ymeann
x3pass=x3pass
xpassno=ypassno
x
  c outto=-1
return
  c
  c 85 continue
do 90 1=1,y3row
    do 95 il=1,y3col
       storer3(i,i1)=ydata(i,i1)
    continue
    storer(i)=ydata(i)
90 continue
sx3row=sx3row
sxrow=sxrow
sx3col=sx3col
sxcol=sxcol
sx3mean=sx3mean
sxmean=xmean
sx3pass=sx3pass
sxpassno=sxpassno
300 continue
sx3row=sx3row
sxrow=sxrow
sx3col=sx3col
sxcol=sxcol
sx3mean=sx3mean
sxmean=xmean
sx3pass=sx3pass
sxpassno=sxpassno
x
  c if (xrow.ne.x3row .or. xpassno.ne.x3pass .or.
    > yrow.ne.y3row .or. ypassno.ne.y3pass) then
    write (*,*) 'WACKO, TRA-IA-IA, 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=xmean
x3pass=sx3pass
xpassno=xpassno
x
  do 300 il=1,x3row
     do 305 il1=1,x3col
        xdata(i,i1)=storer3(i,i1)
x3data(i,i1)=storer3(i,i1)
305 continue
x3data(i)=storer(i)
300 continue
x
  c outto=-1
return
  c
  c 350 continue
  do 360 il=1,sy3row
     do 365 il1=1,sy3col
        ydata(i,i1)=savey3(i,i1)
365 continue
ydata(i)=savey(i)
360 continue
360 continue
y3row=y3row
yrow=yrow
y3col=y3col
ycol=ycol
y3mean=y3mean
ymean=ymean
y3pass=y3pass
ypass=yypass

continue
y3row, y3col, y3pass, xcnt, ycnt, strmincnt,

outto = 0
return

end

subroutine findgap (global,dndk,mlnobs,type)
integer ze:o,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, abs, alldata(4000,4)

subroutine description
findgap locates two adjacent passes that do not have
any common overlapping segment. If it finds two such
passes, then it removes the shorter of the two, continues
running through the remainder of the data while searching
for non-overlapping passes, and finally runs through
the data set to assure that all non-overlapping passes
have been located.

totpass=0
allcnt=0
strnocnt=nocnt

continue
read through the data only once and
store the pass number, first fat
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
continue
allcnt=allcnt+1
alldata(allcnt,1)=real(y3pass)
alldata(allcnt,2)=real(y3row)
alldata(allcnt,3)=y3data(i,1)
alldata(allcnt,4)=y3data(y3row,1)
go to 30
continue

50 continue

70 continue
Jstop=0

xcnt=1
ycnt=2

the next two if statements check
if one of the two adjacent passes
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
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=allcnt+1
      go to 195
    else
      ycnt=allcnt+1
      go to 100
    endif
  endif
  do 105 j=ycnt,allcnt-1
    do 105 j'=1,4
      alldata(j,j')=alldata(jj+1,j)
    enddo
  enddo
  continue
  allcnt=allcnt+1
  go to 100
endif
110 continue
140 continue
both=0
  abs=abs(alldata(xcnt,3)-alldata(ycnt,3))
  if (abs .lt. 0.33) go to 190
  if (abs .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))
    minxrow=min(xrow,yrow)
    nopass=xpassno
    if (minxrow .eq. yrow) nopass=xpassno
    if (xrow .eq. yrow) both=1
    nocnt=nocnt+1
    c------------------------ if this is a dusk pass then will count from
    -90.0 lat degrees toward the equator
    if (dndk .eq. 0) then
      if (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
        if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
          nocnt=nocnt+1
          nowant(nocnt)=nopass
          endif
        endif
      endif
      else if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
        if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
          nocnt=nocnt+1
          nowant(nocnt)=nopass
          endif
        endif
      endif
    endif
  c------------------------ if this is a dawn pass then will count from
    c that is decreasing independent variable
  else if (dndk .eq. 1) then
    if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
      if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
        nocnt=nocnt+1
        nowant(nocnt)=nopass
        endif
      endif
    else if (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
  endif
  if (nocnt .eq. strnocnt) go to 999
  if (strnocnt .lt. nocnt) then
    strnocnt=nocnt
    go to 70
  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
    go to 100
  endif
  200 continue
  if (nocnt .eq. strnocnt) go to 999
  if (strnocnt .lt. nocnt) then
    strnocnt=nocnt
    go to 70
  endif
  c 400 continue
  mncnt=0
  strmncnt=mncnt

B-55
continue

jcstop=0

---------------------------------------- xcnt and ycnt, see notes above

xcnt=1
ycnt=2
if (int(alldata(xcnt,2)) .lt. minobs) then
do 490 j=xcnt,allcnt-1
do 490 j=1,4
   alldata(jj,j)=alldata(jj+1,j)
490 continue
allcnt=allcnt-1
go to 470
endif

500 continue
if (int(alldata(xcnt,2)) .lt. minobs) then
   if (xcnt-1 .gt. allcnt) then
      ycnt=ycnt+1
go to 595
   elseif (jcstop .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
abs=abs(alldata(xcnt,3)-alldata(ycnt,3))
if (abs .lt. 0.33) go to 590
if (abs .ge. 0.33) then
   xrow=int(alldata(xcnt,2))
   yrow=int(alldata(ycnt,2))
   minxyrow=min(xrow,yrow)
   mincnt2=mincnt

----------------------------------------if this is a dusk pass then will count from

   if (dndk .eq. 0) then
      if (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
         if (alldata(xcnt,4) .lt. alldata(ycnt,3)) mincnt=mincnt+1
      elseif (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
         if (alldata(xcnt,4) .lt. alldata(ycnt,3)) mincnt=mincnt+1
      endif
   endif

----------------------------------------if this is a dawn pass then will count from

   elseif (dndk .eq. 1) then
      if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
         if (alldata(xcnt,4) .lt. alldata(ycnt,3)) 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 this is a dusk pass then count from

   if (mincnt .gt. mincnt2) minobs=minxyrow+1
endif

590 continue
xcnt=ycnt
ycnt=ycnt+1
if (jcstop .eq. 1) go to 600
595 if (ycnt .gt. allcnt) then
   if (global .eq. 1) go to 600
   ycnt=1
   jcstop=1
endif
go to 500

600 continue
if (mincnt .eq. strmincnt) go to 999
if (strmincnt .lt. mincnt) then
   strmincnt=mincnt
endif

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)
subroutine track (nop, noc)
integer nop, noc
real radfac, x(400), y(400), xdata(400,4), ydata(400,4)
common /tplot/ x(400), y(400)
common /trunatat/ xdata(400,4), ydata(400,4)

track stores the lat and long coordinates of each data point along a pass. These coordinates are then used to plot with a graphics package, the track footprint of the satellite.

NOTE: the lat and long values are converted to radian values because the plotting package that I work with utilizes radians.

RADFAC=0.017453293
noc=noc+1

do 300 j=1, nop
  x(j) = 90.0 - xdata(j,1)
  y(j) = xdata(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

return
end

B-57
program fourierld
character*80 filename
real xdata(4096), ydata(4096), xmean, ymean, minccin,
> prent, delta, cutli, cutlo, xlag, mincc, maxcc, short, long,
> maxccin, cxmlag, strxdata(4096)
integer xpassno, ypassno, zero, eight, file, xcol, yrow, xcol,
> tima, nd, cc, trnab, npass, lmean, cnwind, seven,
> nwind, numfile, nout, nyout, match, gfcntsxnobs, ynobs
complex xdata(4096), ydata(4096)
common /rowcol/ xdata, ydata
common /fftfft/ nos, prent, lmean, fold
common /lhbft/ delta, cutli, cutlo, xlag, npass, nwind
common /ccflt/ mincc, maxcc, match, minccin, maxccin, cnwind, cxmlag
common /reals/ xdata, ydata
common /complex/ xdata, ydata

program fourierld
character*80 filename
real xdata(4096), ydata(4096), xmean, ymean, minccin,
> prent, delta, cutli, cutlo, xlag, mincc, maxcc, short, long,
> maxccin, cxmlag, strxdata(4096)
integer xpassno, ypassno, zero, eight, file, xcol, yrow, xcol,
> tima, nd, cc, trnab, npass, lmean, cnwind, seven,
> nwind, numfile, nout, nyout, match, gfcntsxnobs, ynobs
complex xdata(4096), ydata(4096)
common /rowcol/ xdata, ydata
common /fftfft/ nos, prent, lmean, fold
common /lhbft/ delta, cutli, cutlo, xlag, npass, nwind
common /ccflt/ mincc, maxcc, match, minccin, maxccin, cnwind, cxmlag
common /reals/ xdata, ydata
common /complex/ xdata, ydata

NOTE: ANY fourier analysis routine can be inserted
into this program as a subroutine. Maybe I'll put in
such features as upward continuation, etc..

NOTE: The only date variables absolutely necessary as INPUT are
the number of observations of input profile, the remaining
variables: zero, mean, pass-number and eight, are not needed.
But, mean can be an OUTPUT if desired.

updates:
30 Jan 91
This update pertained to removing calls to fft2d
subroutines so that now all calls are to the same fft2d
subroutine, and more importantly, now the fft2d routines
will handle 1 row of data so that the bandpass filter works
correctly, and even more importantly the zero filling option
now zero fills such that the data is located in the middle
of all these wonderful zeros. And for those of you who are
really into this, you can now fold out a percentage of the
image of your data, smooth the folded out part to zero, fft,
filter, and ift such that edge effects are minimized.
Ooh boyy!!

9 Jul 92:
Major revisions changing code from two-dimensionally based
fft to one-dimensional ffts. Revisions include removal of
subroutines transpo and store. Major changes to subroutines
fft2d (which is now known as fft1d), datwnd, bndpas and
window. Now there is no longer a need for transposing the
arrays so that run time should be decreased.

NOTE: I removed a great deal of comments at the beginning of
the subroutines. All removed comments discussed the
two dimensional sense of the routine. I added comments
dealing specifically with the one dimensional changes.

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 ('*', '99988')
9988 format ('USE THE FOLLOWING ORDER FOR INPUT FILE/'),
> 'IF VARIABLE DOES NOT APPLY INPUT ANY Bogus NUMBER/',
> 'numfile/',
> 'lhb cc/',
> 'nobs fold prent lmean/',
> 'delta short long npass/',
> 'nwind xlag/',
> 'mincc maxcc match minccin maxccin cnwind cxmlag'/
> 'lsub/)
write ('*', '*') 'INPUT FILE OF VARIABLES'
read ('*', '9990') filename
open (22, file=filename, form='formatted', status='old')
go to 100

B-59
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 TOOOOOO -- to the tune of'
write ('(*)', 'fat albert'
do 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 (/'I FOR ONLY LOW-HIGH-BAND FILTER THE DATA','/','2 IF YOU HAVE TWO FILES TO BE COMPARED')
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.i TO 49.9) '/, '0 DO NOT ADD MEAN TO IFFT DATA'/, '1 ADD MEAN OF INPUT DATA TO OUTPUT IFFT DATA'/, 'choose 2 if 5 or less was chosen above'//, 'lhb cc' )
read (*,* }nobs, fold, prcnt, lmean
c if (nobs .gt. 4096) then
write (*,8999) nobs
8999 format (Ix,'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)
write (*, 9993)
9993 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'/, ' - XLAG..... SMOOTHING PARAMETER FOR WINDOWING IDEAL'/, ' - FILTER IN SPATIAL DOMAIN (95.0) (ls disabled if'/' no window was chosen above)'//, ' - cnwind xlag')
read (*,*) cnwind, 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'/
endif

B-60
> 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 maxim match mincin maxcin cnwind cxlag'
read (*,*) mincc, maxcc, match, mincin, maxcin, cnwind, cxlag
endif
if (numfile .eq. 2) then
 write (**)' AND FINALLY: 0 DO NOT SUBTRACT FILEI-FILEI (use 0)'
 write (**)' I TO WRITE A FILE OF TIME DOMAIN SUBTRACTION'
 write (**)' OF FILE 3 = INPUT FILE 1 - OUTPUT FILE 1'
read (*,*) isub

--------- isub equals 1 if you want to subtract
 the filtered portions of file.I from the
 input of file.I. This option is not
 often used.
endif
goto 200
100 continue
read (22,*) numfile
read (22,*) lhb, cc
read (22,*) nobs, fold, pront, imean
read (22,*) delta, short, long, npass
read (22,*) nwind, xlag
read (22,*) mincc, maxcc, match, mincin, maxcin, cnwind, cxlag
read (22,*) isub

100 continue
cwrite (**)' all input files must have a header with:'
cwrite (**)'row, column, zero, mean, pass number, eight'
cwrite (**)'row... can be bogus but row and col are necessary'
cwrite (**)'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
endif
write (**)' OUTPUT OF FILE 1'
open (20, file-filename, form='unformatted')
if (isub .eq. 1) then
 write (**)' OUTPUT FILE 3 OF SUBTRACTION'
read (**,9990) filename
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
endif
cif (lhb .lt. 6) then
cuthi=.0/short
cutlo=.0/long
RCUTLO=.999999.99
IF(CUTLO .GE. 0.0000001 } RCUTLO=
RCUTHI=.0/CUTHI
WAVLEN=.0/DELTA
FNQI=.0/WAVLEN
WRITE (25,9987) FNQI, MAVLEN, CUTFI, RCUTHI, RCUTLO, CUTOH, WAVLEN,
9987 format ('NYQUIST WAVELENGTH = ',F10.5, ' CYCLES PER DATA INTERVAL/',
 'NYQUIST WAVELENGTH = ',F10.5, ' LENGTH INTERVALS/',
 'LOW WAVE# CUTOFF OF IDEAL FILTER = ',F10.5, '
 'WAVELENGTH EQUIVALENT/',
 'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5, '
 'WAVELENGTH EQUIVALENT/',
endif
c gfont=0
210 continue
read (10, end=999) xrow, xcol, zero, xmean, xpassno, eight
xcol=xrow
do 1=1,xrow

read (10) xdata(1)

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 1-1, yrow
    read (11) ydata(1)
  enddo
endif

c if (isub .eq. 1) then
  do 1-1, xrow
    strxdata(1)=xdata(1)
  enddo
endif

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)

if (numfile .eq. 2) call inverseft (2, ymean, ypassno)
c

500 write (20) xrow, xcol, zero, xmean, xpassno, eight
  do 1-1, xrow
    write (20) xdata(1)
  enddo
endif

530 if (numfile .eq. 2) then
  write (21) yrow, ycol, zero, ymean, ypassno, seven
  if (seven .eq. 7777) then
    gfcnt=gfcnt+1
    go to 570
  endif
  do 1-1, yrow
    write (21) ydata(1)
  enddo

570 if (isub .eq. 1) then
  if (seven .eq. 7777) then
    write (23) xrow, xcol, zero, xmean, xpassno, eight
    do 1-1, xrow
      write (23) xdata(1)
    enddo
  elseif (seven .ne. 7777) then
    write (23) xrow, xcol, zero, xmean, xpassno, seven
    do 1-1, xrow
      write (23) (strxdata(1)-xdata(1))
    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

subroutine forwardft (num, mean, passno)
integer num, xnobs, ynobs, xynobs, nob, passno
real xdata(4096), ydata(4096), prcnt, mean
complex xcdata(4096), ycdata(4096)

B-62
common /fftfft/ nobs,prcnt, lmean, fold
common /rowcol/ xdata,ydata
common /reals/ xdata,ydata
common /comps/ xdata,ydata

COMMON H(4096)
DIMENSION X(2,4096)
COMPLEX H
DOUBLE PRECISION TSL
EQUIVALENCE (X(I,I), H(I))

TSL-0.DO
IF num .eq. 1 then
ynobs-ynobs
DO I-1,ynobs
x[i,i] = xdata(I)
enddo
elseif num .eq. 2 then
ynobs-ynobs
DO I-1,ynobs
ydata(I) = ydata(I)
enddo
eendif

********************************************************************
INT(LOG FLOAT(nobs))/LOG(2.0)+0.01).NE.nobs) THEN
STOP
ENDIF

C....REQUIRE SUBROUTINES :
C....FFTID, FORK, DATWND
C
C....DIMENSIONING REQUIREMENTS :
C
C X(2,4096) ....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 FFTID 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 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 These revisions will almost always be lower case letters.
C revised again (i.e. 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....CALCULATE AND REMOVE THE MEAN
C
XMEAN=TSUM/FLOAT(ynobs)
DO IY-1,ynobs
x[i,iy] = xmeani
enddo
WRITE(25,1020) XMEANI

C....WINDOW THE EDGES VIA DATWND
C
CALL DATWND(PRCCNT, xynobs,nobs, fold)

C....MATRIX IS NOW ZERO FILLED TO NOUT BY NOUT SIZE
C
C....CALCULATE AND REMOVE THE MEAN INTRODUCED BY TAPERING
C
TSUM-0.DO
DO IV=1,nobs
TSUM-+TSUM+x(i,iy)
enddo
XMEAN2=TSUM/FLOAT(nobs)
DO IV=1,nobs
X(I,Y)=X(I,Y)-XMEAN2
enddo
WRITE(25,1020)XMEAN2
XMEAN=XMEAN2+XMEAN1
WRITE(25,1080)XMEAN
write (25,*) passno,xmean1,xmean2,xmean

C.....TRANSFORM DATA TO THE WAVENUMBER DOMAIN
C
CALL FFTID(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('nobs MUST BE A POWER OF 2: SPA2FREQ FATAL')
1080 FORMAT('TOTAL MEAN REMOVED ',F15.7)
C
END
C
************************************************************************
C
SUBROUTINE FFTID (nobs,NSIGN)
C
************************************************************************
C
"FFTID" PERFORMS BOTH A FORWARD OR INVERSE FAST FOURIER TRANSFORM. "FFTID" IS THE DRIVER THAT PASSES THE CORRECT VECTORS TO "FORK" WHICH PERFORMS THE ACTUAL TRANSFORMING.
The dimensioning of "H" must be the same as in the main program.

"nobs" = Number of fft observations in data matrix
"NSIGN" = Direction of desired transformation

-1 INVERSE TRANSFORM (FREQUENCY TO SPATIAL)
-0 FORWARD TRANSFORM (SPATIAL TO FREQUENCY)

************************************************************************
C
COMMON H(4096)
COMPLEX H
C
SIGNI=FLOAT(NSIGN)
if (iabs(signi).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
SUBROUTINE FORK (LXX,CX,SIGNI)
C
************************************************************************
C
FAST FOURIER TRANSFORM, MODIFIED FROM CLAERSBOUT, J.F., FUNDAMENTAL OF GEOPHYSICAL DATA PROCESSING, MCGRAW-HILL, 1976
FORK USES COOLEY-TUKEY ALGORITHM.

"CX" = Data vector to be transformed
"LXX" = Length of data vector "CX" to be transformed, must be a power of 2 (LXX=2**INTEGER)
"SIGNI" = Direction of desired transformation

-1 INVERSE TRANSFORM (FREQUENCY TO SPATIAL)
-0 FORWARD TRANSFORM (SPATIAL TO FREQUENCY)

NORMALIZATION PERFORMED BY DIVIDING BY DATA LENGTH UPON THE FORWARD TRANSFORM.

************************************************************************
C
COMPLEX CX(LXX),CW,CTEMP,CON2
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

102 IF (J.GT.M) THEN
  J=J-M
  M=M/2
ENDIF

103 CONTINUE

104 ISTEP=2*L
CON2=(0.0, 3.14159265)/FLOAT(L)*SIGNI
DO 105 M=I, L
  CX(I)=EXP(CON2*FLOAT(M-1))
  CX((I+L)=CX((I-1)+CTEMP
105 IF (L.LE.LX) GO TO 104
END IF

DO 106 I=1, LX
  CX(I)=CX(I)*SC
106 RETURN
END

C***********************************************************************
C SUBROUTINE DATWND (PRCNT, xynobs, nobs, fold)
integer xynobs
C***********************************************************************
C **DATWND** MULTIPLIES THE INPUT F(I,xynobs) BY A HALF BELL OF A HAMMING*
C TUKEY WINDOW ON ALL EDGES AND ZEROS OUT THE REMAINDER OF THE
C (NX, NY) ARRAY.
C **PRCNT** =PERCENTAGE OF DATA TO BE ALTERED IN SMOOTHING TO ZERO
C 0.0.LT. "PRCNT" .LE. 50.0
C update 2 feb 91
datwnd has been considerably improved such that now the subroutine
performs three (count them, three !!!) functions; one; a percentage
of the input matrix can be folded out. two; after folding out,
a new percentage of the folded out matrix (or regular data if
folding was not performed) can be smoothed to zero. three; the
manipulated data is centered within zeros to finish filling the
matrix to nx by ny size. because the actual data is now centered
within the transformed array, it is necessary to use the
do loops in subroutine inverseft to correctly extract the actual
data
C***********************************************************************
C dimension holdme(4096)
COMMON F(2,4096)
C--------------------------------- fold out the data based on percentage
C
C xyn=xynobs
if (fold.gt.0 .and. fold.lt.100.0) then
  KX=Int(fold*FLOAT(xynobs)/100.0+0.5)
  if (xynobs, gt, nobs) kx=(nobs-xynobs)/2
  do i=1, xynobs
    holdme(i)=f(1,1)
  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.ge.(kx+xynobs)) f(1,i)=holdme(i-kx)
  enddo
  xyn=xynobs*2+kx
C
C if (prcnt.gt.0 .and. prcnt.lt.50.0) then
  KX=IFIX(PRNCNT*FLOAT(100)/100.0+0.5)
  IF (KX.K.E.0) THEN
    RXXPI= 3.14159265/FLOAT(KX)
    DO IX=1, KX
      FACTOR=0.5*(1.0+COS(FLOAT(IX+1)*RXXPI))
    enddo
    F(1,IX)=F(1,IX)*FACTOR
ENDIF

B-65
END

WRITE(25,150) KX, KY
WRITE (25,*} KX, KY
END

C
C WRITE(25,150)
KX, KY
ENDIF
C
C.....center and ZERO OUT REMAINDER OF ARRAY IF NECESSARY
C
nxhalf=(nobs-nxl)/2
DO I=1,nxhalf
holdme(I)=f(I,1)
ENDO
DO I=1,nxhalf
f(I,1)=0.0
ENDO
DO I=nxhalf+1,nxhalf+nxl
f(I,1)=holdme(I-nxhalf)
ENDO
DO I=nxhalf+nxl+1,nobs
f(I,1)=0.0
ENDO
RETURN

C
C SUBROUTINE FILTER (NUM)
INTEGER NUM, NPASS, IMEAN, NIND, NIND
REAL PRCNT, XLAG, DELTA, CUTHI, CUTLO
COMPLEX XCDATA (4096), YCDATA(4096), CDATA (4096)
COMMON /ROWCOL/ XN obs, YN obs
COMMON /FFTIFFT/ N OBS, PRCNT, IMEAN, FOLD
COMMON /COMPS/ XCDATA, YCDATA
COMMON /LHBFLT/ DELTA, CUTHI, CUTLO, XLAG, NPASS, NIND
COMMON H(4096)

COMPLEX H1(2,4096)
DIMENSION D1 (2, N) ..... WHERE N IS THE NUMBER OF OBSERVATIONS OF THE
H(N) INPUT AND OUTPUT TRANSFORMED MATRIX. N MUST BE AN
INTEGRAL POWER OF TWO (2,4,8,16...).
NOTE: DIMENSIONS IN EVERY SUBROUTINE MUST BE
SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.

C
C PROGRAM BANDPASS
C PROGRAM BANDPASS PERFORMS HIGH, LOW, OR BANDPASS WAVENumber
FILTERING OF UNIFORMLY GRIDDED ARRAYS. AN IDEAL FILTER
IS CONSTRUCTED IN THE WAVENUMBER DOMAIN, WINDOWED IN THE SPACE
DOMAIN, THEN TRANSFORMED BACK INTO THE WAVENUMBER DOMAIN TO
BE MULTIPLIED BY THE INPUT TRANSFORM.
C
C.....REQUIRED SUBROUTINES :
BNDPAS, FFT2D, FORK, STORE, WINDOW
C
C.....DIMENSIONING REQUIREMENTS :
D1(2,N) ..... WHERE N IS THE NUMBER OF OBSERVATIONS OF THE
H(N) INPUT AND OUTPUT TRANSFORMED MATRIX. N MUST BE AN
INTEGRAL POWER OF TWO (2,4,8,16...).
NOTE: DIMENSIONS IN EVERY SUBROUTINE MUST BE
SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C.....AUTHOR: ODEN REED, PURDUE UNIVERSITY, DECEMBER 1980.
REVISIONS BY STEVE MATESKON AND JEFF LUCIUS,

this program, like others in the fft series, has been updated
to the DEC workstation system and now the program is actually
usable to just about anybody! revised 21 apr 90
well, like the other programs in this package, this has been
updated on 4 AUG 90. few comments have been removed - mainly
those comments about I/O operations not necessary to this
package have been removed.
update: 2 feb 90, removed need for cstore array

************************************************************************
.
.
CREATE IDEAL CONTINUATION FILTER AND STORE IN ARRAY H
CALL BNDPAS (CUTLO, CUTHI, NPASS, DELTA, nobs)

CALL BNDPAS (CCUTLO, CCUTHI, NPASS, DELTA, nobs)

************************************************************************
.
.
SUBROUTINE BNDPAS (CCUTLO, CCUTHI, NPASS, DELTA, nobs)

************************************************************************
.
.
"BNDPAS" CALCULATES THE WAVE# RESPONSE OF AN IDEAL BANDPASS
FILTER OF A (nobs) MATRIX. ARRAY "H" MUST BE DIMENSIONED THE
SAME AS IN THE MAIN PROGRAM

"CCUTLO" LOWEST WAVE# TO BE PASSED, GE 0.0
"CCUTHI" HIGHEST WAVE# TO BE PASSED, LE NYQUIST
"NPASS" SWITCHES EITHER A PASS OR REJECTION BETWEEN
"CUTLO" & "CUTHI"
   -1 REJECT WAVENUMBERS BETWEEN THE 2 WAVENUMBERS
   -1 PASS WAVENUMBERS BETWEEN THE 2 WAVENUMBERS
"DELTA" DATA GRID INTERVAL, IN MAP UNITS
nobs number of fft observations

************************************************************************
.
.
COMMON H(4096)
COMPLEX H, ZERO, ONE

CUTLI=CCUTLI
CUTLO=CCUTLO
RCUTLO=999999.99
IF(CUTLO.GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
RCUTHI=1.0/CUTLI
WAVLEN=2.0*DELTA
FNQ1=1.0/WAVLEN
WRITE (25,112) FNQ1,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI,NPASS

IF (IBS(NPASS).NE.1) THEN
WRITE(6,151)
STOP
ENDIF
IF (CUTHI.GT.FNQ1.OR.CUTHI.LE.CUTLO.OR.CUTLO.GT.0.0) THEN
WRITE(6,151)
STOP
ENDIF

NXX=nobs+2
NO=(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

B-67
CLOWX = FLOAT (NX2) * WAVLEN * CUTLO
CHIX = FLOAT (NX2) * WAVLEN * CUTHI

C ....... "ZERO" OUT THE entire ARRAY
C DO IX=1,nobs
  H(IX) = ZERO
enddo

C ....... OPERATE ON ROWS WHERE WAVENUMBERS ARE .LE. CUTLO
C MINS=1
MAXS=NX2
IF (CUTLO.GT.0.000001) MINS = int (HLOWX + 2.00001)
IF (MAXS .LE. MAXH) then
  DO IX=MINS,MAXS
    H(IX) = ONE
  enddo
ENDIF

RETURN

C
112 FORMAT (/IX, 'NYQUIST WAVELENGTH = ',F10.5, 'LENGTH INTERVALS'
> /IX, 'LOW WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
> ' CYCLES PER DATA INTERVAL',3X, 'LENGTH INTERVALS'/
> /IX, 'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
> ' CYCLES PER DATA INTERVAL',3X, 'LENGTH INTERVALS'/
> /IX, 'WAVELENGTH EQUIVALENT',/IX, 'HPASS= ', II/)
151 FORMAT (5X, 'IMPOSSIBLE FILTER CONSTRUCTION IS SPECIFIED. FATAL')
C END

C************************************************************************
C SUBROUTINE WINDOW (hObS, XLAG, NWIND)
C************************************************************************
C "WINDOW" PERFORMS 1-DIMENSION WINDOWING OVER THE DATA ARRAY
C EACH QUAD. IS SEPERATELY WINOONED. 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 "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-turkey window
C -4 gives parzen window
C
C************************************************************************
C COMMON H(4096)
COMPLEX H
C IAG=FLOAT(nobs) * XLAG/200.0+0.5
IAG=ANMOD(LAG,2)
F1=3.14159265
NX2=(nobs/2)+1
XNX=FLOAT(NX2)
XNX-1.0/FLOAT(NX2)

C RADIUS=FLOAT(LAG)*NX2
RAD1=1.0/RADIUS
RAD2= RADIUS/RADIUS
C ....... APPLY RECTANGULAR WINDOW TO FILTER
C IF (NWIND .EQ. 1) THEN
  MAX= int (RAD1 *NNXXR + 2.0001)
  DO II=NXX-1,0
    H(NXX-II) = (0.0, 0.0)
    H(II) = (0.0, 0.0)
  enddo
WRITE (25,660) XLAG, IAG
RETURN
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=0.5*(1.0-COS(PIRADI*XI))
H(LL)=H(LL)*FACTOR
MK=NOX-LL
253 H(MK)=H(MK)*FACTOR
ENDIF
C
LL=MAX+1
DO 255 II=LL,NX2
H(NOX-II)=(0.0,0.0)
255 H(II)=(0.0,0.0)
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=1.0-6.0* ((XI*RADI)**2-(XI*KADI)**3)
H(LL)=H(LL)*FACTOR
MX=NOX-LL
353 H(MX)=H(MX)*FACTOR
ENDIF
C
KOUNT=MAX+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=NOX-LL
457 H(MX)=H(MX)*FACTOR
LL=MAX+1
DO 455 II=LL,NX2
H(NOX-II)=(0.0,0.0)
455 H(II)=(0.0,0.0)
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*KADI)**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*KADI)**3)
H(LL)=H(LL)*FACTOR
MX=NOX-LL
453 H(MX)=H(MX)*FACTOR
ENDIF
C
KOUNT=MAX2+1
DO 457 LL=KOUNT,MAX2
XI=FLOAT(LL-1)*XNX
FACTOR=2.0*(1.0-(XI*RADI))**3
H(LL)=H(LL)*FACTOR
MX=NOX-LL
457 H(MX)=H(MX)*FACTOR
LL=MAX2+1
DO 455 II=LL,NX2
H(NOX-II)=(0.0,0.0)
455 H(II)=(0.0,0.0)
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,','LAG= ',I5)
661 FORMAT ( 'BARTLETT WINDOW USED XLAG= ',F7.3,','LAG= ',I5)
662 FORMAT ( 'HAMMING-TUKEY WINDOW USED XLAG= ',F7.3,','LAG= ',I5)
663 FORMAT ( 'PARZEN WINDOW USED XLAG= ',F7.3,','LAG= ',I5)
664 FORMAT ( 'NO WINDOWING HAS BEEN APPLIED ; XLAG= ',F7.3)
C
END
subroutine correlate(xpassno,ypassno)
integer xnobs,ynobe,hobs,match,
powerS,power6,totpwr
real mlncc,maxcc,ccwinout,prcnt,
pctpr3,pctpr4,minccin,maxccin,cxlag
complex h(nobs),*nwind
complex x(nobs),y(nobs),zero,
POWER1,POWER2,POWER3,POWER4,TPOWER
REAL CC0EF,CCIN,CCOUT
DATA ZERO/(0.000000,0.000000)/
common /rowcol/xnobe,ynobe
common /comps/x,y
common /ccflt/mincc,_,axcc,match,mlnccin,maxccin,cnwlnd,cxlag
common /fftifft/nobs,prcnt,imean,
POWER1,POWER2,POWER3,POWER4,TPOWER

-- subroutine description

sub subroutine correlate finds the correlation coefficient between each
wavenumber component of the two input arrays. Each cc is
normalized to range between -1.0 through 0.0 to 1.0.
the cc is the cosine of the phase angle difference between two
wavenumber components.

revisions:
well, by now you know the story... revised 4 AUG 90
i've added the windowing functions available from the
correlation subroutines to this cc-filter. try them if
you like!
update 6 feb 91: change calculation of correlation
coefficient from a summation based formula to the cosine of
the phase angle difference.

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 NBR OF OBSERVATIONS'
write ('*,*') 'CORRELATION COEF MAY NOT BE CORRECT'
write ('*,*') 'FILE 1: OBSERVATIONS =',xnobs
write ('*,*') 'FILE 2: OBSERVATIONS =',ynobs
endif

pi=3.141592654
twopi=6.283185307
POWER1=ZERO
POWER2=ZERO
POWER3=ZERO
POWER4=ZERO
XPPOWER=ZERO
TPPOWER=ZERO

DO 110 i=1,nobs
zerocont(i)=1
--- zerocont array is a flagging array used to
set the windowing array h to equal
(0.0,0.0) or (1.0,0.0). a little inspection
of subroutine BNDPAS will help illuminate
the principle.

POWER1=POWER1+X(i)*CONJG(X(i))
POWER2=POWER2+Y(i)*CONJG(Y(i))
XPPOWER=XPPOWER+X(i)*CONJG(Y(i))

xrad is the phase angle of the x array wavenumber and
yrad is the phase angle of the y array wavenumber.
the
cosine of the minimum phase difference is the correlation
of the two wavenumbers. to find the minimum phase difference
it is necessary to adjust xrad or yrad with integer values
of pi. so... do not change the order of the if statements !!

xrad=atan(imag(x(i))/real(x(i)))
if (real(x(i)).lt.0.0) xrad=xrad+pi
if (imag(x(i)).lt.0.0) xrad=pi+xrad
yrad=atan(imag(y(i))/real(y(i)))
if (real(y(i)).lt.0.0) yrad=yrad+pi
if (imag(y(i)).lt.0.0) yrad=pi+yrad
xrad=xrad-yrad
ccof=cos(xrad)

B-70
IF (CCOEF .GT. maxcc .or. CCOEF .LT. mincc) THEN
  X(I) = ZERO
  Y(I) = ZERO
  zerocnt(I) = 0
ENDIF

! SUM THE POWERS & CROSS PRODUCTS FOR THE OUTPUT MAPS.
POW3 = POW3+ (X(I) * CONJG(X(I)))
POW4 = POW4+ (Y(I) * CONJG(Y(I)))
TPOWER = TPOWER+ (X(I) * CONJG(Y(I)))
CONTINUE

! CALCULATE THE C.C. FOR THE OUTPUT MAPS.
if (power3 .eq. zero .or. power4 .eq. zero) then
  ccout = REAL(TPOWER/SQRT(POWER3*POWER4))
else
  endif

! CALCULATE THE C.C. FOR THE INPUT MAPS.
if (powerl .eq. zero .or. power2 .eq. zero) then
  ccin = REAL(POWER/SQRT(POWER1*POWER2))
else
  endif

! CALCULATE THE PERCENTAGE OF THE POWER RETAINED IN THE FILTERED MAPS.
if (powerl .eq. zero .or. power2 .eq. zero) then
  pctprl = -999.9
  pctpr2 = -999.9
else
  PCTPRI = (POWER3/POWER1)*100.0
  PCTPR2 = (POWER4/POWER2)*100.0
endif

WRITE (6, 444) CCIN
WRITE (6, 555) CCOUT
WRITE (6, 666) PCTPR1, PCTPR2
WRITE (6, 666) PCTPR1, PCTPR2
WRITE (6, 888) xpassno, ypassno, ccin, ccout, pctpr1, pctpr2
888 FORMAT (2i6, 4fi0.3)

if (ccin .lt. minccin) write (*,*) xpassno, ypassno, ccin, " <min'
if (ccin .gt. maxccin) write (*,*) xpassno, ypassno, ccin, " >max"

! The following if statement controls the windowing functions for smoothing the output arrays and calculates a new output correlation coefficient and percents of power retained in the windowed arrays because the data will change slightly with windowing
if (cnwind .ge. 1 .and. cnwind .le. 4) then
  powers = zero
  power5 = zero
  totpw = zero
  do i = 1, nobs
    h(i) = (1.0, 0.0, 0.0)
    if (zerocnt(I) .eq. 0) h(I) = (0.0, 0.0, 0.0)
    enddo
  call fftld(nobs, h)
  call window(nobs, cxlag, cnwind)
  call fftld(nobs, -h)
  do i = 1, nobs
    x(i) = x(i)+h(i)
    y(i) = y(i)+h(i)
    power5 = power5+ (x(i)*CONJG(x(i)))
  enddo
endif
power6 = power6 + y(1) * conjg(y(1))
retotpw = retotpw + x(1) * conjg(y(1))
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(retotpw/sqrt(power5*power6))
pctpr3 = (power5/power1) ** 100.0
pctpr4 = (power6/power2) ** 100.0
340 continue
write (25,888) xpassno, ypassno, ccln, ccwinout, pctpr3, pctpr4
format (2i6, 4f10.3)
endlf
return
end

subroutine inverseft (num, mean, passno)
integer nmsx, yobs, xobs, xynobs, row, col, passno
real cdata(4096), ydata(4096)
complex xdata(4096), ydata(4096)
common /rowcol/ xobs, yobs
common /reals/ cdata, ydata
common /xdata/ xdata, ydata
common /fftwf/ nobs, prcnt, lmean, fold
common /h(4096)
dimension x(2, xynobs), holdme(4096)
colex h

if (num .eq. 1) then
  xynobs = xobs
do l = 1, nobs
    h(l) = xdata(l)
  enddo
elseif (num .eq. 2) then
  xynobs = yobs
do l = 1, nobs
    h(l) = ydata(l)
  enddo
endif
--- subroutine description

--- REQUIRED SUBROUTINES:
--- FFTID, FORK

--- DIMENSIONING REQUIREMENTS:
--- X(2,N) . . . WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
--- H(N) . . . OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
--- POWER OF TWO (2,4,8,256...).  
--- NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
--- SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.

--- AUTHOR : JEFF LUCIUS
--- DEPARTMENT OF GEOLOGY AND MINERALOGY
--- OHIO STATE UNIVERSITY, DECEMBER 1984.

--- revised: 8 AUG 90
--- updated: 2 Feb 91
--- added do loops that find the data portion of the
--- zero centered inverse transformed data.  a look at
--- subroutine datWnd will help figure this out.

--- INVERSE TRANSFORM DATA TO THE SPACE DOMAIN
--- CALL FFTID (nobs,+1)
--- nhalf = (nobs-xynobs)/2
do l = 1, nhalf + 1, nhalf + xynobs
  holdme(l-nhalf) = h(l)
enddo
total = 0.0
do l = 1, xynobs
  x(l,1) = holdme(l)
total = total + x(l,1)
enddo
xmean = total/float(xynobs)
IF (IMean .EQ. 1) THEN
  do i=1,xynobs
    x(i,1)=x(i,1)+mean
  enddo
ENDIF

C
XMIN= 1.0E20
XMAX=-1.0E20
if (num .eq. 1) then
  do i=1,xynobs
    xdata(i) = x(i,1)
  enddo
else (num .eq. 2) then
  do i=1,xynobs
    ydata(i) = x(i,1)
  enddo
endif

DO
  i=1,xynobs
  XMIN=AMINI (XMIN,X(i,1))
  XMAX=AMAXI (XMAX,X(i,1))
  IF(XMAX.EQ.X(i,1)) IMAX=i
  IF(XMIN.EQ.X(i,1)) IMIN=i
  enddo
C
  WRITE (25,1020) XMAX,IMAX,XMIN,IMIN,xmean,passno
  write (25,9980) passno,xmean,xmax,imax,xmin,imin
  9980 format (15,2x,f13.5,2x,f13.5,2x,f13.5,2x,f13.5,2x,f13.5,2x,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,
> xrow, xcol, x3pass, yrow, ycol, y3pass, yrow,
> strct, srow, s3row, s4col, s4col, sobs, noc,
> sypassno, s3pass, dndk, tcount,��rent, minobs,
>��rentpair, choice, toctobs, prime, global, crosscnt,
> passrem, nowant(4000), nocnt, type
real xmean, ymean, xdata(400, 3), ydata(400, 3), x3mean,
> xdata(400, 3), ydata(400, 3), xdata(400, 4), ydata(400, 4),
> x3mean, y3mean, x3mean, x(400), y(400),
> savey(400), savey3(400, 3), xmean,
> ymean, avgdata(400, 4)
character*80 filename
common /trunstat/ x3data(400, 3), xldata(400)
common /aver/ avgdata(400, 4)
common /tplot/ x(400), y(400)
common /nope/ nowant(4000), nocnt

program description

combine is very similar to movetrunc in that both programs truncate adjacent passes to the same over-lapping segments, both programs also provide statistics and track output files. the major difference is that movetrunc has only one file as input whereas combine has two files as input. combine can output one file of two passes averaged together to make the one file or it can output two similar length passes that can be further processed by fourier methods. movetrunc and combine could be cluged together to make one program so why don't you go ahead and jam them together??
good luck!!

program date: 16 apr 91

write (*,*) '1 TO HAVE ONE OUTPUT FILE'
write (*,*) '2 TO HAVE TWO OUTPUT FILES'
read (*,*) ifilenum
if (ifilenum .eq. 2) then
prime = 1
endif
write (*,*) '1 TO AVERAGE A-east AND A-west'
write (*,*) '2 TO AVERAGE A-east AND B-west (choose 2)'
read (*,*) prime
write (*,*) '0 IF THE DATA SET IS GLOBAL OR POLAR'
write (*,*) '1 IF THE DATA SET DOES NOT INCLUDE ALL LONGITUDES'
read (*,*) global
if (prime .eq. 1) then
write (*,*) 'NOTE: FILE ¥ WILL HAVE THE FIRST PASS MOVED'
write (*,*) 'TO THE BOTTOM OF THE FILE'
write (*,*) 'AND THE FIRST PASS WILL NOT BE INCLUDED IN THE PROCESSING'
if (global .eq. 1) then
write (*,*) 'THE PROCESSING, FILE X WILL HAVE THE LAST PASS REMOVED AND THIS PASS WILL NOT BE INCLUDED IN THE PROCESSING'
eendif
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')
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')
else if (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 Z OF VARIABLE'
read (*,9990) filename
open (33, file=filename, form='unformatted')
endf

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 (*,*) 'DO NOT REMOVE THE MEAN FROM THE AVERAGED DATASET'
write (*,*) '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,*), nowant(i)
5 continue
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 (*,*) 'running through dataset to find passes that',
' do not overlap'
c findgap locates adjacent
passes that do not overlap
call findgap (global,dndk,minobs,type)
c write (*,*) 'done with run through'
c
write (25,*) 'XPASS YPASS CCD CCY XVAR YVAR ',
'COVARXY XSTDEV YSTDEV',
'XMEAN YMEAN XSLOPE YSLOPE',
'XINTCPT YINTCPT'
c
paircnt=0
paircnt1=0
tcount=0
totobs=0
noc=0
crosscnt=0
passrem=0
c
continue
read (10, end=999) x3row,x3col,zero,x3mean,x3pass,eight
do 15 i=1,x3row
read (10) (x3data(i,1),i=x3row)
15 continue
read (11) xcol,zero,x3mean,xpassno,eight
do 25 i=1,x3row
read (11) xdata(i)
25 continue
read (12,end=150) y3row,y3col,zero,y3mean,y3pass,eight
do 35 i=1,y3row
read (12) (y3data(i,1),i=y3row)
35 continue
read (13) yrow,ycol,zero,ymean,ypassno,eight
do 45 i=1,yrow
read (13) ydata(i)
45 continue

continue

strcnt=strcnt+1
c-------------------------------------- this if statement offsets the passes

B-76
and saves the offset for the end

if (strcnt .eq. 1 .and. prime .eq. 1) then
do 50 i=1,y3row
do 55 li=1,y3col
  savey3(i,ii)=y3data(i,li)
55 continue
  saveyl(1)=yldata(1)
50 continue
syr3row=y3row
syr3col=y3col
sy3mean=y3mean
sy3pass=y3pass
ypassno=yypassno
50 continue
sy3row-y3row
syrow-yrow
sy3col-sy3col
sycol-ycol
sy3mean-ymean
symean-ymean
sy3pass-sypass
sypassno-sypassno
50 continue
go to 30
endif

go to 190

150 continue
if (global .eq. 1) go to 999
do 160 i=1,sysr3row
do 165 li=1,sys3col
  y3data(i,li)=savey3(i,li)
165 continue
  yldata(1)=saveyl(1)
160 continue
sysr3row-sy3row
syrow-syrow
sys3col-sy3col
sycol-sycol
sys3mean-sy3mean
symean-symean
sy3pass-sy3pass
sypassno-sypassno
160 continue
if (xrow .ne. x3row .or. xpassno .ne. x3pass .or. yrow .ne. y3row .or. ypassno .ne. y3pass) then
  write (*,*) 'WACKO, TBA-LA-LA, JOLLY-GOOD, NO MATCH BETWEEN'
  write (*,*) 'X-',xrow,x3row,
  write (*,*) 'Y-',yrow,y3row,
  go to 999
elseif (xrow .lt. mlnobs .or. yrow .lt. mlnobs) then
  write(*,*)'FILE X PASS',xpassno,' REMOVED: OBSERVATIONS-',xrow
  write(*,*)'FILE Y PASS',ypassno,' REMOVED: OBSERVATIONS-',yrow
  passrem=passrem+1
  go to 10
endif
195 continue
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

call truncate (xrow,yrow,dndk,nobs,paircntl,mlnobs)
call statistics (nobs,xpassno,ypassno,xmean,ymean)
call average (nobs,choice,prime,crsscnt)
endif

!-----------------------------write out the truncated lengths passes
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)
  enddo
  continue
  do 205 j=1,nobs
    if (xadata(j,3) .lt. xadata(j+1,3)) then
      crosscnt=crosscnt+1
      go to 207
    endif
    207 continue
  endif
else (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)
  enddo
endif

!----------------------------- write out the trace of the pass for plotting
in tplot
  call track(nobs,noc,prime)
  WRITE (21) NOC,NObs,(X(J),Y(J),J-1,NObs)
  C
  continue
if (paircnt .gt. 0) tcount=tcount+1
  paircnt=paircnt+paircnt
  totobs=totobs+nobs
  go to 999
continue
write (*,'(a9,a10)') 'corrected','pairs of latitudes in'
write (*,'(a9,a10)') 'total passes read = ',stront
write (*,'(a9,a10)') 'total observations in the written dataset - ',totobs
write (*,'(a9,a10)') 'study area includes','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

! subroutine truncate (xrow,yrow,dnd,xminrow,stocount,mlnobs)
integer xrow,yrow,stocount,rolii,roline,minrow,
      dnd,minobs
real xdata(400,4),ydata(400,4),
      xddata(400,3),xldata(400),ydata(400,3),yldata(400),
      xdata,ydata,diffab,absx,xdata(400,4),ydata(400,4)
common /trunstat/ xdata(400,4),ydata(400,4)
common xdata(400,3),xdata(400),ydata(400,3),ydata(400)

!-------------------------------- subroutinedescription
! truncate locates the overlapping segment between two adjacent passes and stores that segment in the appropriate arrays

  do 70 j=1,xrow
    xdata(j,1)=xdata(j,1)
    xdata(j,2)=xdata(j,2)
    xdata(j,3)=xdata(j,3)
    70 continue
  do 75 j=1,yrow
    ydata(j,1)=ydata(j,1)
    ydata(j,2)=ydata(j,2)
    ydata(j,3)=ydata(j,3)
    75 continue
  stocount=0
  rolii=xrow

B-78
rowinc=yrow

---

loops from 90 to 200 increment through the
two input passes and truncate the lengths
to the same length

90 continue
  adata=xdata(jj,1)
  bdata=ydata(jj,1)
  diffab=edata-bdata
  abs=abs(diffab)
  if (rowii.eq.0 .or. rowlnc.eq.0) then
    write(‘*’,'(6x,’xrows (li) ’,rowii,’ yrows (Inc) ’,rowlnc)
    write(‘*’,'(6x,’xrow ’,xrow,’ yrow ’,yrow)
    stop
  endif
  rowii=min(rowii, rowlnc)
if pass a (li) matches pass b (Inc) at beginning length then write to xdata and ydata and race to main program
if (abs.lt.0.33) then
  do 110 ll=1,minrow
     do 110 kk=1,4
        xdata(ll,kk)=xdata(ll,kk)
        ydata(ll,kk)=ydata(ll,kk)
    110 continue
  return
endif
---

if pass a no matcha the b data then find new a or b depending on whether or not ascending or descending order of independent variable
if (abss.ge.0.33) then
  stocount=stocount+1
---

if this is a dusk pass then will count from -90.0 lat degrees toward the equator
if (dndk.eq.0) then
  if (xdata(JJ, l) .gt. ydata(JJ,l)) then
    rowlnc=rowlnc-1
    do 130 mm=1,rowlnc
       do 130 kk=1,4
          ydata(nn,kk)=ydata(nn+1,kk)
    130 continue
  elseif (xdata(JJ, l) .lt. ydata(JJ,l)) then
    rowii=rowii-1
    do 150 nn=1,rowii
       do 150 kk=1,4
          xdata(nn,kk)=xdata(nn+1,kk)
    150 continue
  endif
endif
---

if this is a dawn pass then will count from the equator toward the south pole
elseif (dndk.eq.1) then
  if (xdata(JJ, l) .lt. ydata(JJ,l)) then
    rowlnc=rowlnc-1
    do 160 mm=1,rowlnc
       do 160 kk=1,4
          ydata(mm,kk)=ydata(mm+1,kk)
    160 continue
  elseif (xdata(JJ, l) .gt. ydata(JJ,l)) 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
---

go to 90
end
---

---

subroutine statistics (minrow,xpassno,ypassno, xmean, ymean)
integer minrow,nobs,xpassno,ypassno
real xdata(400,4),ydata(400,4),nobs
common /trunstat/ xdata(400,4),ydata(400,4)
---

from 200 to write statement of variables is the statistical calculations using two references:
1) Davis, Statistics and Data Analysis in Geology, 2nd ed., 1986 pp. 41
2) Young, Statistical Treatment of Experimental Data, 1962, McGraw Hill, 115-132
---

loops that sum x, x**2, y, y**2 and xy and calculate new truncate means
nobs=minrow
nobs=float(nobs)
---

B-79
```fortran
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

write (*,*) xsum, ysum, xsumsqr, ysumsqr, sumxy

xmean=xsum/nobss
ymean=ysum/nobss

sumprod=sumxy-((xsum*ysum)/nobss)

xvar=xsumsqr-((xsum**2)/nobss)

yvar=ysumsqr-((ysum**2)/nobss)

xstdev=sqrt(xvar)

ystdev=sqrt(yvar)

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)

write (25,9992) xpassno, ypassno, xvar, yvar, xstdev, ystdev, xamean, ybmean, xslope, yslope, xintcpt, yintcpt, corrDxy, corrYxy
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)
9993 format ('COVARIANCE XY-',f9.3,' Davis CORRELATION COEF-',f9.3)
9994 format ('X SLOPE-',f9.3,' Y INTERCEPT-',f9.3,' Y SLOPE-',f9.3)
9995 format (215,7(f10.4))
9996 format (10x,6(f10.4))
return
end

subroutine average (nobs,choice,prime,crosscnt)
real xadata(400,4), ybdata(400,4), avgdatmean, avgdatsum, avgdata(400,4), nobss
integer nobss, choice, prime, crosscnt
common /runstat/ xadata(400,4), ybdata(400,4)
common /aver/ avgdata(400,4)

------ subroutine description ------
average calculates the average magnetic value of the input passes. It will also find the average position of the input passes if so directed.
avgdatsum=0.0
nobss=real(nobs)

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,2)+ybdata(i,2))/2.0
    addxy=abs(xadata(i,3))+abs(ybdata(i,3))
    if (addxy .gt. 270.0) then
      crosscnt=crosscnt+1
      if (ydata(i,3) .gt. 0.0) then
        xadata(i,3)=xadata(i,3)+360.0
      else (ydata(i,3) .lt. 0.0) then
        xadata(i,3)=xadata(i,3)-360.0
    end if
  end do
100 continue
```

ybdata(i,3) = ybdata(i,3) + 360.0
endif
avgdatal(1,2) = (avgdatal(1,3) + ybdata(i,3))/2.0
if (avgdatal(1,2) > 180.0)
  avgdatal(1,2) = avgdatal(1,2) - 360.0
endif
avgdatal(1,3) = (avgdatal(1,4) + ybdata(i,4))/2.0
avgdatal(1,4) = (avgdatal(1,2) + ybdata(i,2))/2.0

continue
avgdatalmean = avgdatalsum/nobs
if (choice .eq. 1) then
do 150 l = 1,nobs
  avgdatal(i,4) = avgdatal(i,4) - avgdatalmean
  continue
endif
return
end

subroutine track (nop, noc, prime)
integer nop, noc, prime
real radfac, avgdatal(400,4), x(400), y(400), xadata(400,4),
> ybdata(400,4)
common /aver/ avgdatal(400,4)
common /tplot/ x(400), y(400)
common /trunstat/ xadata(400,4), ybdata(400,4)

------------------------------ subroutine description
track finds the lat and long coordinates of the observations
along an orbit. These coordinates can be plotted as the
trace of the pass along the earth.
NOTE: lat and long are changed to radians for the plotting
package that I use.

RADFAC=0.017453293
noc=noc+1
if (prime .eq. 2) then
do 200 J-1,nop
  x(J) = 90.0 - avgdatal(J,1)
  y(J) = avgdatal(J,2)
  if (y(J) .lt. 0.) y(J) = y(J) + 360.
  x(J) = x(J) * RADFAC
  y(J) = y(J) * RADFAC
  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.) y(J) = y(J) + 360.0
    x(J) = x(J) * radfac
    y(J) = y(J) * radfac
  300 continue
endif
return
end

subroutine findgap (global, dndk, minobs, type)
integer zero, elght, strtotpass, x3row, x3col, x3pass,
> y3row, y3col, y3pass, cnt, type, strmincnt,
> dndk, nowant(4000), nocnt, mincnt,
> minobs, totpass, global,
> nopass, strnocnt, strallcnt, strlallcnt
real y3data(400,3), y3mean, abss, allxdat(4000,4),
> allydat(4000,4), x3mean, x3data(400,3), stryone(4)
common /hope/ nowant(4000), nocnt

------------------------------- subroutine description
findgap locates the overlapping segment in each of the
two adjacent passes. This is done by looking at the
first and last latitudes in each pass and comparing
the values. If there are two passes that do not have
overlapping segments, then one of the two passes

B-81
is removed.
NOTE: the difference between a latitude gap finder and a minimum observation gap finder is: a latitude finder allows extremely short passes to be worked with in the processing whereas a minimum observation gap finder removes all short passes. the trade off occurs because a minimum observation finder actually allows a higher number of observations to be worked with in collocation. therefore a minobs gap finder is usually best. experiment to see what u like.

totpass=0
allcnt=0
strn=0

c 10 continue
read (10,end=30) x3row,x3col,zero,x3mean,x3pass,eight
do 15 1=i=1,x3row
read (10) (x3data(i,i),i=1,x3col)
15 continue
 totpass=totpass+1
if (x3row .lt. minobs) then
 want(nocnt+1)=x3pass
 nocnt=nocnt+1
endif
allcnt=allcnt+1
allxdat(allcnt,1)=real(x3pass)
allxdat(allcnt,2)=real(x3row)
allxdat(allcnt,3)=x3data(i,i)
allxdat(allcnt,4)=x3data(x3row,1)
go to 10

c 30 continue
strtotpass=totpass
strallcnt=allcnt

totpass=0
allcnt=0

c 31 continue
read (12,end=50) y3row,y3col,zero,y3mean,y3pass,eight
do 35 1=i=1,y3row
read (12) (y3data(i,i),i=1,y3col)
35 continue
 totpass=totpass+1
if (y3row .lt. minobs) then
 want(nocnt+1)=y3pass
 nocnt=nocnt+1
endif
allcnt=allcnt+1
allydat(allcnt,1)=real(y3pass)
allydat(allcnt,2)=real(y3row)
allydat(allcnt,3)=y3data(i,i)
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 52 do 11 i=1,4
 sryone(i)=allydat(i,i)
11 continue
do 55 i=1,allcnt-1
do 55 i=1,4
 allydat(i,i)=allydat(i,1,i)
55 continue
do 57 i=1,4
 allydat(allcnt,1)=sryone(i)
57 continue
if (global .eq. 1) allcnt=allcnt-1

c 1 if (type .eq. 2) go to 400

c c----------------------------- work a latitude gap finder
70 continue

c 100 continue
if (allydat(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
c 1 if (nocnt .eq. 0) go to 140
do 110 i=1,nocnt
if (int(allydat(cnt,1)) .eq. nowant(i)) then
 if (cnt+1 .gt. allcnt) go to 190
do 110 j=cnt,allcnt-1

B-82
do 105 J=1,4
   allxdat(J,J)=allxdat(J,J+1,J)
   allydat(J,J)=allydat(J,J+1,J)
105   continue
   allcnt=allcnt+1
   continue
   allcnt=allcnt-1
110 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))
   endif
   140 continue
   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
      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
   endif
   c------------------------if this is a dusk 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
      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
   c------------------------if this is a dawn pass then will count from
   c the equator toward the south pole
   c that is decreasing independent variable
   endif
190 continue
   cnt=cnt+1
   if (cnt .gt. allcnt) go to 200
   go to 100
200 continue
   if (nwant .eq. strnwant) go to 999
   if (strnwant .lt. nwant) then
      strnwant=nwant
      go to 70
   endif
   c------------------------ work a minimum observations gap finder
400 continue
   mincnt=0
   strmincnt-strmincnt
   c
470 continue
   cnt=1
   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 .lt. allcnt) go to 590
      do 505 jj=cnt,allcnt-1
         do 505 jj=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
500 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))
      minxrow=min(xrow,yrow)
if this is a dusk pass then will count from -90.0 lat degrees toward the equator

if (dndk .eq. 0) then
   if (allxdat(cnt,3) .gt. allydat(cnt,3)) then
      mincnt = mincnt+1
   elseif (allxdat(cnt,3) .lt. allydat(cnt,3) )
   then
      mincnt = mincnt+1
   endif
endif

if this is a dawn pass then will count from the equator toward the south pole

elseif (dndk .eq. 1) then
   if (allxdat(cnt,3) .lt. allydat(cnt,3) ) then
      mincnt = mincnt+1
   elseif (allxdat(cnt,3) .gt. allydat(cnt,3) )
   then
      mincnt = mincnt+1
   endif
endif

if (mincnt .gt. mlncnt2) minobs = minobs+1
endif

continue 590
if (cnt .gt. allcnt) go to 600

continue 600
if (mincnt .eq. strmlncnt) go to 999
if (strmlncnt .lt. mincnt) then
   stmincnt = mincnt
   go to 470
endif

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)

return
<table>
<thead>
<tr>
<th>OC 1980, MAGSAT/83 FIELD MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

-1  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

THIS PROGRAM READS A FILE OF IRREGULARLY DISTRIBUTED
DATA POINTS (LATITUDE, LONGITUDE, ELEVATION, ANOMALY
VALUE) AND PREDICTS THE GRIDDED ANOMALIES ON A GRID OF
SPECIFIED DENSITY. THE ANOMALY ESTIMATE FOR EACH GRID
LOCATION IS OBTAINED FROM NCPP NEAREST SAMPLE POINTS
USING A LOCAL COVARIANCE MODEL. THE PROCEDURE KNOWN AS
LEAST-SQUARES COLLOCATION INVOLVES THE FOLLOWING STEPS:

1. REMOVE THE MEAN OF THE ANOMALIES IN THE PREDICTION
   AREA TO OBTAIN RESIDUALS CENTERED AROUND MEAN
2. SEARCH FOR THE NCPP CLOSEST DATA POINTS TO THE GRID
   LOCATION AND STORE THEM IN VECTOR (MI)
3. FORM THE COVARIANCE MATRIX (COVM) OF THE NCPP DATA
   POINTS
4. ADD THE ERROR VARIANCE OF THE DATA POINTS TO THE
   DIAGONAL OF 'COVM' MATRIX, TO FORM THE FINAL 'COVM'
   MATRIX
5. INVERT 'COVM' MATRIX AND STORE THE RESULT AGAIN IN
   'COVM'
6. FORM THE CROSS-COVARIANCE VECTOR (TI) BETWEEN THE
   GRID VALUE TO BE PREDICTED AND THE NCPP DATA POINTS
7. BY LEAST SQUARES COLLOCATION, THE ANOMALY ESTIMATE
   IS GIVEN AS:
   \[ TP - TI \cdot COVM^{-1} M1 \]
   AND THE STANDARD ERROR OF PREDICTION IS GIVEN BY,
   \[ SEP = \sqrt{VAR - TI \cdot COVM \cdot T1} \]
   \[ VAR \ldots \text{COVARIANCE AT ZERO SEPARATION (I.E. VARIANCE)} \]

THE ABOVE EQUATIONS INVOLVE THE MATRIX OPERATIONS.

PRELIMINARY SOFTWARE EXPLICITLY DEVELOPED FOR GRAVITY
PREDICTION OVER A SPHERICAL SURFACE WAS MADE AVAILABLE
BY GEODETIC SCIENCE DEPARTMENT AT THE OHIO STATE
UNIVERSITY. IT WAS MODIFIED FOR NASA MAGNETIC SATELLITE
APPLICATION FOR 3-D PREDICTION AND THEREBY ALTITUDE
NORMALIZATION.

MODIFICATIONS BY: HARISH K. GOYAL
DEPT. OF GEO & MIN, OSU
TEL. 422-1434, CAMPUS
MAR. 1986

1. LATITUDES ARE CHANGED TO CO-LATITUDES TO COMPLY
   WITH SPHERICAL COORDINATES.
2. SEPARATION DISTANCES ARE THE RADIAL VECTORS TO
   ACCOUNT FOR THREE DIMENSIONAL VARIABILITY
3. COVARIANCES ARE AUTOMATICALLY SCALED IN THE
   PROGRAM

further modifications 11 may 90
these modifications are all lower case letters
well, just a few more modifications on 8 sep 90
these changes are: 1) removal of unnecessary arrays
2) changing all arrays to real*4 and not real*8
3) duping all arrays that work with the inversion as
   real*8. 4) removing every blasted 'nbug' statement i could
get my hands on!! 5) reading the data once storing everything in
memory -ie, not reading the data twice. 6) changing all logical
test and false statements to user friendly statements.
7) this program could be faster by inverting only the half of
   the symmetric covariance matrix (COVM) and there probably is a
   faster routine for searching for the closest points.

INPUT PARAMETERS ....................................

NORTH...NORTH LATITUDE OF DATA AREA
SOUTH...SOUTH LATITUDE OF DATA AREA
WEST...WEST LONGITUDE OF DATA AREA
EAST...EAST LONGITUDE OF DATA AREA

note: use the following example if you want an equal-
area projection, say you are working at the south
pole from -40S to -83S and including all longitudes.
then instead of choosing north--40, south--83,
west--180 and east=180 (which are appropriate for
non-equal area degrees) choose instead:

north=55, south=55, west=55, east=55. 55 comes
from the following calculation. from -40 to
-83 degrees is 43 points, however, an equal area projection will go all the way to the pole because the pole will be centered at the middle of the grid (unlike the non-equal area degree projection where the pole is at the southern edge of the grid) so, from -40 to -90 is 50 points and you should add a few points for a rind around the edge, say 5 points. 55-50+5. get it? now, if you are going to work with an equal area projection, you must transform the coordinates of the input data points from the degree domain to the spatial domain. program gllraspc.f does this transformation. also, if you want to convert a grid from/to equal-area to/from non-equal degree then use program deg2spc.f to do the coordinate transform and use collocation to get the values at the new grid coordinates. clear as mud, eh? email me at aladorf@geol.mps.osu.edu if you need help.

NX...NO. OF GRID PTS IN THE LONG. DIRECTION, MINUS ONE
NY...NO. OF GRID PTS IN THE LAT. DIRECTION, MINUS ONE
NCOV...NUMBER OF ENTRIES IN COVARIANCE FUNCTION
and is determined by the program
NCPP...NO. OF NEAREST POINTS USED FOR PREDICTION
ELEV...COMMON ALTITUDE FOR GRIDDED ANOMALY DATA
and is in kilometers from --SURFACE-- of the earth
ESTD...STANDARD DEVIATION OF OBSERVATIONAL ERROR
(ERROR VARIANCE-ESTD**2)

GRID DIMENSIONS AS FOLLOWS:
the following arrays = (ny+l)*(nx+l):
 x,y,tp,ifc,ih,sep
the following arrays must be equal to or greater than the maximum number of data points:
 rr, rthi, rphi, tano, dlst, thii, phi, radd, anomal
ARRAYS CNN, DNUM MUST ACCOMMODATE THE NUMBER OF ENTRIES
IN COVARIANCE FUNCTION
CCCC...NO. OF GRID PTS IN THE LONG. DIRECTION, MINUS ONE
CC...NO. OF GRID PTS IN THE LAT. DIRECTION, MINUS ONE
NCOV...NUMBER OF ENTRIES IN COVARIANCE FUNCTION
and is determined by the program
NCPP...NO. OF NEAREST POINTS USED FOR PREDICTION
ELEV...COMMON ALTITUDE FOR GRIDDED ANOMALY DATA
and is in kilometers from --SURFACE-- of the earth
ESTD...STANDARD DEVIATION OF OBSERVATIONAL ERROR
(ERROR VARIANCE-ESTD**2)

GRID DIMENSIONS AS FOLLOWS:
the following arrays = (ny+l)*(nx+l):
 x,y,tp,ifc,ih,sep
the following arrays must be equal to or greater than the maximum number of data points:
 rr, rthi, rphi, tano, dlst, thii, phi, radd, anomal
ARRAYS CNN, DNUM MUST ACCOMMODATE THE NUMBER OF ENTRIES
IN COVARIANCE FUNCTION

COMMON/TWO/ SCALE
DATA RHO, NPTS/57.2957795,0/
c
write (*,*) 'INPUT COVARIANCE MATRIX'
read (*,9990) filename
9990 format (a80)
write (*,*) 'INPUT FILE OF ALL DATA POINTS LAT LON RAD ANOM'
open (i, file=filename, status='old', form='formatted')
write (*,*) 'OUTPUT FILE OF GRIDDED DATA POINTS'
open (20, file=filename, form='formatted')
open (21, file=filename, form='formatted')
c
c
write (*,*) 'BAKF-- 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 WRITING' 
write (*, *) 'y OR n' 
read (*, 9991) yesno 
format (a5} c 
If (EAST.EQ.WEST OR NORTH.I.E. SOUTH.OR.NY.I.E.0.OR.NX. 
. IE.0) STOP 9999 
C CHANGE LATITUDES TO SPHERICAL COORDINATES 
NORTH=90.0-NORTH 
SOUTH=-90.0-SOUTH 
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 INPUT THE COVARIANCE TABLE 
i=1 
read (1,,end=7) dnum(i),cnn(i) 
i=1 
go to 5 
7 ncowv=1 
SCALE=CNN(i) 
C GRID SPACING ..ie: interval between grid nodes = dl and dp 
C DP=(NORTH-SOUTH)/FLOAT(NY) 
DL=(EAST-WEST)/FLOAT(NX) 
C .. DETERMINE THE OVERLAPS IN X AND Y-DIRECTIONS 
C YOVLAP=DP/2. 
XOVLAP=DL/2. 
C .. NX1,NY1 = 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 
NX1=NX + 1 
NY1=NY + 1 
DO IO 1=1,NX1 
Y(i)=DP*(I-1)+YOVLAP 
DO IO J=1,NY1 
Y(J)=TY 
X(J)=DL*(J-1)+XOVLAP 
10 CONTINUE 
C 
DETERMINE THE X,Y COORDINATES OF THE DATA AREA AND 
GRID SPACING 
C XLL=0. 
XUP=X(NX1)+XOVLAP 
YLL=0. 
YUP=Y(NX1,NY1)+YOVLAP 
DXX=NX1/(XUP-XLL) 
DYY=NY1/(YUP-YLL) 
C DETERMINE BOUNDARIES FOR DATA SELECTION 
C THIS = SOUTH-YOVLAP 
THIS=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 
corresponding individual data points if each 
data point or group of data points need to have 
individually different error variances. 
C 
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 INPUT ADJUSTED MAGNETIC DATA AND SELECT DATA FOR
C THE PREDICTION

C
np=(nxpl+1)*(nypl+1)
do 15 i=1,np
ifc(i)=0
continue
C AMEAN=0.0
totpts=0
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
RVI=THI-THIS
C IX AND IY IDENTIFIES THE BLOCK TO WHICH DATA FALL INTO
C AND IYJX ASSIGN AN IDENTIFIER TO DATA CORRESPONDING TO
C THAT BLOCK

IX=INT((RY-YLL)*DYY)+1
JX=INT((RX-XLL)*DXX)+1
IYJX=(IY-I)*NXPI+JX
C IFC - COUNTER VECTOR, STORES NUMBER OF DATA PER
C SORT ELEMENT

IFC(IYJX)=IFC(IYJX)+1
C
C thii(npts)-thi
phii(npts)=phi
raddi(npts)=rad
anom(npts)=anomaly
C GO TO 20
C
continue
if (npts .le. i) 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
IH=NXPI*NYPI
IH(1)=1
DO 87 I=2,NP
11=I-1
IH(I)=IH(11)+IH(I)
87 CONTINUE
C
IX,IX,IXHI,RPHI,RRAD,TANO,VARN ARE NUMBERED FOR
C CORRESPONDING DATA, IN EACH BLOCK NUMBERING STARTS
C WITH IN 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 transferred to
C subroutine prdt.
C
sumsqr=0.0
DO 85 I=1,NPTS
rxiphii(I)=phi
rythii(I)=thi
IX=INT((RY-YLL)*DYY)+1
JX=INT((RX-XLL)*DXX)+1
IYJX=(IY-I)*NXPI+JX
NAM=IH(IYJX)
TANO(NAM)=anom(I)-amean
sumsqr=sumsqr+(dble(tano(nam))*dble(tano(nam)))
RTHI(NAM)=thii(I)
RPHI(NAM)=phii(I)
RRAD(NAM)=raddi(I)
IH(IYJX)=IH(IYJX)+1
85 CONTINUE
C
IH(I) ATTAINS THE VALUE EQUAL TO NUMBER OF SAMPLE
C POINTS IN PREVIOUS BLOCKS + 1

C-4
SUBROUTINE PRDT PREDICTS ANOMALIES AND ERRORS OF STANDARD DEVIATION AT EACH GRID LOCATION

WRITE (*,*) 'calculating anomaly values'

CALL PRDT (NPTS, NYPI, NXPI, X, Y, TP, RTHI, RPHI, DIST, 
VARN, THIS, TANO, NORTH, SOUTH, EAST, WEST, IH, DXX, DYY, DP, 
DL, AMEAN, RAD, NCPP, ELEV, SEP, sumsqr)

WRITE THE VALUES OF THE PREDICTED Z-AXIS VALUES (ANOMALIES) AND THEIR ERROR OF STANDARD DEVIATIONS. THE ROWS ARE LISTED WITH LATITUDES STARTING SOUTH AND INCREASING TO NORTH.

if (yesno .eq. 'y') then
np=np1*nypl
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

WRITE(21,9600) WEST,DL
write (20,*) nxpl
write (20,*) nypl
write (20,*') south
write (20,*') west
write (20,*') dl
P=SOUTH
DO 400 I=1,NYPI
  ID1=(I-1)*NXPI+1
  ID2=ID1+NXPI-1
  WRITE(21,9601) (TP(J),J=IDI,ID2)
400 continue
WRITE (21,9602) WEST, DL
write (20,*) nxpl
write (20,*) nypl
write (20,*) south
write (20,*) west
write (20,*) dl
P=SOUTH
DO 420 I=1,NYPI
  ID1=(I-1)*NXPI+1
  ID2=ID1+NXPI-1
  WRITE(21,'(6F13.5)') (SEP(J),J=IDI,ID2)
420 continue
WRITE (21,*') 'total mean removed from the grid =',avgrid

STOP
END
XLL=0.0

CALCULATE THE SCALING FACTOR FOR THE COVARIANCE TABLE

fact = (sumsqr/dble(nptp))/scale

CALL COVINT(0.0,FACT,VAR)

write (21,*) 'sum of squares =',sumsqr,' scaling factor =',FACT
write (21,*) 'zero separation variance =', var

NCPPM=NCPP-1
IND=0

P AND Q ARE THE LATITUDE AND LONGITUDE OF THE PREDICTED POINT

write (21,*) 'the following data indicate areas where the'
write (21,*) 'prediction was bad'
write (21,*) 'lat lon x grid node y gridnode id 1 id 2 ',
> 'bad point number of total points'

DO 10 IPT=1,MY
IM=IPT-1
P=PHI2+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
LCC(I)=0
DO 7 IY=INT((YPP-YLL)*DYY)+I
DO 7 JX=INT((XPP-XLL)*DXX)+JX
IYJX=(IY-I)*MX+JX
IDI=IH(IYJX)
IYJX2=IYJX+JX1
ID2=IH(IYJX2)
IF((ID2-IDI).GE.NCPP) GO TO I00

NOT ENOUGH DATA IN FIRST WINDOW, 50 CONSIDER NEXT WINDOW

DO 17 IC=IYI,IY2
NDATA=0
IYI=IY-IC
IY2=IY+IC
IF (IYI.LT. I) IYI=I
IF (IY2.GT.MY) IY2=MY
JXI=JX-IC
JX2=JX+IC
IF (JXI.LT. I) JXI=I
IF (JX2.GT.MX) JX2=MX
DO 18 IL=IYI, IY2
IYJXI=(IL-I)*MX+JXI
IDI=IH(IYJXI)
IYJX2=IYJXI+JX2-OXI+I
ID2=IH(IYJX2)
NDATA=NDATA+1
18 CONTINUE
IF (NDATA.GE.NCPP) GO TO 100
NBAD=NBAD+I

write the bad point to file
WRITE (21,800) P, Q, XPP, YPP, ID1, ID2, NBAD, ND

800 FORMAT(4 (ix,F12.5), 416)
17 CONTINUE
100 CONTINUE

DO 211 IC=IY1, IY2
IF (IY1.EQ. IY2) GO TO 106
IYX1=(IC-1)*MX+JX1
ID1=IH(IYX1)
IYX2=IYX1+JX2-JX1+1
ID2=IH(IYX2)
NDATA-NDATA+1
DO 210 IL=IY1, IY2
IYJX1=(IL-I)*MX+JX1
IDI=IH(IYJX1)
IYJX2=IYJX1+JX2-JX1
ID2=IH(IYJX2)
NDATA-NDATA+1
DO 210 IL=ID1, IY2
IF (IL.EQ. ID2) GO TO 211
106
DO 210 IL=IY1, IY2
IF (IL.EQ. ID2) GO TO 106
DELC=COS(LAT(IL)/CON1)*COS(P/CON1)+SIN(LAT(IL)/CON1)*
> SIN(P/CON1)*COS((LONG(IL)-Q)/CON1)
PART1=ELEV**2 + RC(IL)**2
PART2=2.0*ELEV*RC(IL)
DARGU-PART1-PART2*DELC
IF (DARGU.LE.0.0) THEN
DIST(I)=0.0

C-6
\begin{verbatim}
ELSE
   DIST(I) = SQRT(DARGU)
END IF
210 CONTINUE
211 CONTINUE

C SEARCH FOR NCPP NEAREST POINTS TO PREDICTION POINT
C
220 DO 260 J-I, NCPP
   IF (LCC(J).GT.0) GO TO 260
   DMIN=DMAX
   DO 253 IC-IYI, IY2
      IF (IYI.EQ.IY2) GO TO 107
      IIYOMI = (IC-I)*MX + JXI
      IIIXI = IIYOMI - JXI
      ID2=IIYOMI - JXI
      IF (ID2.LT.IDI) GO TO 253
   DO 250 I-IDI, ID2
      IF (DIST(I).LT.DMIN) THEN
         DMIN=DIST(I)
         LMIN=I
      END IF
   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 FORM COVARIANCE MATRIX
C
280 DO 300 I-I, NCPP
   N=LCC(I)
   COVM(1,I)=DBLE(VAR+NSE)
   K=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=DARGU+PART1*PART2+DELC
      IF (DARGU.LE.0.0) THEN
         DIS=0.0
      ELSE
         DIS=SQRT(DARGU)
      END IF
      CALL COVINT(DIS, FACT, COV)
      COVM(J,J)=DBLE(COV)
      CONTINUE
290 CONTINUE
270 COVM(NCPP, NCPP)=DBLE(VAR+NSE)

C INVERT COVARIANCE MATRIX
C
COVM(1,1)=1.0/COVM(1,1)
DO 340 I-I, NCPP
   L=I+1
   DO 300 J=I+1, L
      B(J)=0.0
      DO 310 K=I+1, L
         B(J)=B(J)-COVM(K,J)*COVM(L,K)
      Dummy=COVM(L,L)
      DO 330 J=I+1, L
         Dummy=B(J)*COVM(L,J)+Dummy
      Dummy=1.0/Dummy
      DO 330 J=I+1, L
         COVM(L,J)=B(J)*Dummy
      END DO
      DO 310 K=I+1, L
         COVM(J,K)=COVM(J,K)+B(K)*B(J)*Dummy
      END DO
340 COVM(L,L)=Dummy

C c-------------------------- corm array dimensioned at (ncpp,ncpp) is now inverted
C c-------------------------- corm array dimensioned at (ncpp,ncpp) is the input matrix
C
270 DO 410 J-J, NCPP
   N=LCC(J)
   DELC=COS(LAT(N)/CON1)*COS(P/CON1)+SIN(LAT(N)/CON1)*
      SIN(P/CON1)*COS((LONG(N)-Q)/CON1)*
      PART1=RC(N)**2 + LLEV**2
   END IF
C
C-7
\end{verbatim}
PART2=2.0*ELEV*RC(N)
DARGU=PART1+PART2*DEL
IF (DARGU.LE.0.0) THEN
  DIS=0.0
ELSE
  DIS=SQR(DARGU)
END IF

CALL COVINT(DIS,FACT,COV)

DO 430 I=1,NCPP
  TEM=0.0
  DO 420 J=1,NCPP
    TEM=COVM(J,I)*T1(J)+TEM
  420 CONTINUE
  T2(I)=TEM
  TEM=0.0
  TEM2=0.0
  DO 440 I=1,NCPP
    TEM=CCM(I)*T2(I)+TEM
  440 CONTINUE

DO 410 I=1,NCPP
  TEM=0.0
  DO 400 J=1,NCPP
    TEM=COVM(J,I)*T1(J)+TEM
  400 CONTINUE

CALL COVINT(DIS,FACT,COV)

DO 410 I=1,NCPP
  T1(I)=dble(COV)
  DO 400 J=1,NCPP
    TEM=COVM(J,I)*T1(J)+TEM
  400 CONTINUE
  T2(I)=TEM

DO 410 I=1,NCPP
  TEM=0.0
  DO 400 J=1,NCPP
    TEM=COVM(J,I)*T1(J)+TEM
  400 CONTINUE

CALL COVINT(DIS,FACT,COV)

END

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 ................................... INTERPOLATION OF COVARIANCES
TCOV=dble(NCOV-1)
RL=dble(DIS)
IF(RL.GT.TCOV)GO TO 1
COV=CNN(NCOV)*FACT
WRITE(21,100) RL
RETURN
1 IF(RL.GT.0.0001) GO TO 6
COV=CNN(I)*FACT
RETURN
6 DO 2 I=1,NCOV
  IF(RL.GT.DNUM(I)) GO TO 3
  2 CONTINUE
3 I=I-1
  FPINT=RL-DNUM(I)
  FDINT=DNUM(I+1)-DNUM(I)
  COV=CNN(I)*(CNN(I+1)-CNN(I))*FPINT/FDINT
  COV=COV*FACT
  RETURN
100 FORMAT( 'SEPARATION = ',F10.3,' >500 KM COVARIANCE VALUE FOR',
               ' > 500 KM USED.' )
END
program fourier2d
character*80 filename
real xdata(361,361), ydata(361,361), xmean, ymean,
> prcnt, delta, cuthl, cuto, cxlag, mincc, maxcc, short, long,
> xcolat, ycolat, xlong, ylong, xgrldspc, ygrldspc,
> minccin, maxccin, cxlag
integer xpassno, ypassno,
> lhb, cc, npass, lmean, xmean, ymean, npass, swind,
> nwind, numfile, nout, nyout, cwind, uwind, ndl,
> numcc(10), numlhb(10), numud(10), numrtp(10),
> numudt(10)
complex xcdat(512,512), ycdat(512,512)
common /rowcol/ xrow, yrow, xcolat, ycolat,
> xdata, ydata
common /ftflfl/ lmean, xmean, fold, itypefold
common /lhfilt/ delta, cuthl, cuto, cxlag, npass, swind,
common /lhfilt/ mincc, maxcc, minccin, maxccin, cwind, cxlag
common /reals/ xdata, ydata
common /striking/ ang1, ang2, spass, swind, cxlag
common /udcont/ uddata, xdata, ydata
common /ocfilt/ xcol, xrow, ycol, yrow,
common /lhbflt/ xdata, ydata
common /rtp/ xdata, ydata
common /xyzdelta/ xdata, ydata
common /udlhb/ xcdat, ycdat

!-----------------------------------------------------!
!
!fourier2d is an all encompassing fourier analysis program.
!subroutines include the fft for forward and inverse situations,
!a bandpass filter which can be adjusted to perform low, high and
!bandpass filtering of wave numbers, a correlation coefficient
!filter which zeros out wavenumbers according to correlation
!coefficients, a strike-dip filter to remove wavelengths in
!direction of defined strain, an upward-downward continuation filter,
!a derivative filter for any of the three directions and a
!derivative-to-pole filter for magnetic total field intensity data.
!
!NOTE: the only data variables absolutely necessary as INPUT are
!the number of rows and number of columns, the remaining
!variables; zero, mean, pass-number and eight, are not needed,
!but, mean can be an OUTPUT if desired.
!
!NOTE: fourier2d is for two-dimensional data. If you have a
!one-dimensional data set then use fourierid. However,
in 1-d 1 have not yet implemented the continuation,
!derivative or rtp filters.
!
!program date: 10 jul 92
!
!This code was an extensive modification of an earlier code
!named fourmat.
!
!NOTE: because there are 6 filters, there are about forty
!zillion different combinations of filtering the fft’d
!data. so, to accomodate all of these, the user must
!first state how many times to run the filter (lhb, icc, etc) then state the order where to run each
!filter (arrays numlhb, numcc, num etc... hold these
!user defined positions). note that with this scheme
!any filter can run multiple times. be sure to enter
!only one filter for each position value. the
!following example should clear things up:
!say you want to first bandpass filter, then cc
!filter and then bandpass filter again. then you
!should set lhb=2 and icc=1. then array numcc should
!have the values 1 and 3 (for the first and third positions)
!and array numuc should only have the value 2 (for the
!second position)
!
write (*,*) 'OUTPUT FILE OF STATISTICS AND INFORMATION'
read (*.9990) filename
open (25, file=filename, form='formatted')
write (*,*) numfile
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
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 (*,*) lhb, icc, lsd, lhd, lrd, lderiv
if (lhb .gt. 0) then
write (*,*) 'BANDPASS SECTION:
write (*,*) 'enter placement values in order ie. 2 3 5'
write (*,*) 'do not repeat these values elsewhere'
write (*,*) 'l=first, 3=third, 5=fifth etc...'
read (*,*) numlhb(1), lhd, lderiv
write (*,9993) 9993 format ('DELTA.....GRID INTERVAL IN MAP UNITS (.0 degrees)'/
> 'SHORT.....SHORTEST WAVELENGTH TO BE PASSED'/

C-10
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

cuthi-l.0/short

cutlo=1.0/long

cutlo=999999.99
IF (CUTLO .GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
RCUTHI=1.0/CUTHI
WAVLEN=2.0*DELTA
FNQI=1.0/WAVLEN
write (25,9987) FNQI,WAVLEN,cuthi, cutlo, rcutlo, rcuthi
9987 format ('NYQUIST WAVENUMBER = ',F10.5, ' CYCLES PER DATA INTERVAL'/,

'NYQUIST WAVELENGTH= ',F10.5, ' LENGTH INTERVALS'/,

LOW WAVE# CUTOFF OF IDEAL FILTER - ',FI0.5,' CYCLES PER DATA INTERVAL '/,

HIGH WAVE# CUTOFF OF IDEAL FILTER - ',FI0.5,' WAVELENGTH EQUIVALENT'/,

endlf

if (icc .gt. 0) then
if (numfile .ge. 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 '
read (**, *(numcc(i),i=1,icc)
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'/

XLAGG SMOOTHING PARAMETER FOR WINDOWING IDEAL'/

FILTER IN SPATIAL DOMAIN (is disabled if' /

no window was chosen above)./ 

mminc maxincc minccin maxccin cnwinder cxlag'
read (**, mminc, maxcin, mincin, maxcin, cnwinder, 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,lsd)
9986 format ('ANGLE i: >- 0.0 AND < ANGLE2'/,

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 (lud .gt. 0) then
write (**, 'UP/DOWN CONTINUATION SECTION:'

C-11
write (*,*), 'enter placement values in order ie. 2 3 5'
write (*,*), 'do not repeat these values elsewhere'
read (*,*), (numud(i),i=1,lud)
write (*,9985)
9985 format('udDELTA - GRID INTERVAL IN MAP UNITS (I.0)/,
  'ZCON - DISTANCE TO CONTINUE THE DATA SET/,
  'IN THE SAME LENGTH UNITS AS DELTA'/,
  'NEGATIVE FOR UPWARD CONTINUATION'/,
  'POSITIVE FOR DOWNWARD CONTINUATION'/,
  'udWIND - 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 udwind udxlag')
read (*, *) uddelta, zcon, udwind, udxlag
endif

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 (*,*) 'l-first, 3-third, 5-fifth etc...
read (*,*) (numrtp(i),i=1,irtp)
endif

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 (*,*) 'l-first, 3-third, 5-fifth etc...
read (*,*) (numdariv(i),i=1,ideriv)
endif

c write (*, 9992)
9992 format ('THE FOLLOWING REFERS TO FFT AND IFFT'/,
  'NUMBER OF COLUMNS AND ROWS OF FFT ARRAY'/,
  'TYPE OF INPUT ARRAY INDICATES TYPE OF FOLDING TO BE USED'/,
  'IF A POLAR REGION, i.e. E AND W EDGES ARE SAME'/,
  'IF A NON-POLAR REGION, i.e. E AND W EDGES NOT SAME'/,
  '1 IF EACH EDGE OF INPUT ARRAY TO /
  'BE FOLDED OUT: (0.0 TO 99.5)'/,
  'PERCENT OF EACH EDGE OF FOLDED OUT OR NORMAL ARRAY'/,
  'TO BE SMOOTHED TO ZERO: (0.0 TO 49.9)'/,
  'O 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
endif

c write (*,9999)
8999 format (lx, 'SORRY',16,lx, 'OR',16,lx, 'IS GREATER THAN 512 THE',
  'SIDE OF ARRAYS SET'/ IN THE SOURCE CODE '/,
  'YOU NEED TO ACCESS SOURCE CODE AND MAKE CHANGES')
stop

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
endif
open (11, file=filename, status='old', form='formatted')
C-13

```fortran
endif
write ('*,9990) 'OUTPUT OF FILE 1'
read ('*,9990) filename
open (20, file=filename, form='formatted')
if (numfile .eq. 2) then
  write ('*',9990) 'OUTPUT OF FILE 2'
  read ('*,9990) filename
  open (21, file=filename, form='formatted')
endif

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

xpassno=1
ypassno=2
do i=1,xrow
  read (10,*) (xdata(j,1),j=1,xcol)
enddo
if (numfile .eq. 2) then
  do i=1,yrow
    read (11,*) (ydata(j,1),j=1,ycol)
  enddo
endif

xmean=0.0
ymean=0.0
if (numfile .eq. 2) call forwardft (2,xmean,ypassno)
call forwardft (1,xmean,xpassno)

ltottime=llhb+llcc+llsd+lirtp+lderlv
do i=1,ltottime
  do j=1,llhb
    if (numlhb(j) .eq. 1) then
      call filter (1)
    endif
    if (numfile .eq. 2) call filter (2)
    goto 888
  enddo
  do j=1,llcc
    if (numcc(j) .eq. 1) then
      call correlate(xpassno,ypassno)
    endif
    goto 888
  enddo
  do j=1,llsd
    if (numsd(j) .eq. 1) then
      call strkpas(1)
    endif
    if (numfile .eq. 2) call strkpas(2)
    goto 888
  enddo
  do j=1,lirtp
    if (numrtp(j) .eq. 1) then
      call mag2pol(1)
    endif
    if (numfile .eq. 2) call mag2pol(2)
    goto 888
  enddo
  do j=1,lderlv
    if (numderlv(j) .eq. 1) then
      call deriva(1)
    endif
    if (numfile .eq. 2) call deriva(2)
    goto 888
  enddo
  continue
enddo
```

C-13
SUBROUTINE FORWARDFT (NUM, MEAN, PASSNO)
  INTEGER NUM, XROW, XCOL, YROW, YCOL, NOUT, NYOUT, PASSNO,
  > ROW, COL
  REAL XDATA(361,361), YDATA(361,361), PRCNT, MEAN
  COMPLEX XCDATA(512,512), YCDATA(512,512)
  COMMON /FFTFT/ NOUT, NYOUT, PRCNT, MEAN, FOLD, ITYPEFOLD
  COMMON /ROWCOL/ XROW, XCOL, YROW, YCOL
  COMMON /REALS/ XDATA, YDATA
  COMMON /COMPS/ XCDATA, YCDATA
  DIMENSION X(2,512,512)
  COMPLEX H
  DOUBLE PRECISION TSUM
  EQUIVALENCE (X(I,1,I), H(I,1))

  TSUM=0.0
  IF (NUM .EQ. 1) THEN
    DO 50 I=1,ROW
      DO 50 J=1,COL
        X(I,J,1) = XDATA(J,I)
        TSUM = TSUM + X(I,J,1)
  50 CONTINUE
  ELSEIF (NUM .EQ. 2) THEN
    DO 80 I=1,ROW
      DO 80 J=1,COL
        X(I,J,1) = YDATA(J,I)
        TSUM = TSUM + X(I,J,1)
  80 CONTINUE
  ENDIF

END

PROGRAM SPA2FRQ
PROGRAM SPA2FRQ TRANSFORMS AN N X N MATRIX OF SPACE-DOMAIN
AMPLITUDES INTO THE N X N MATRIX OF WAVE NUMBER DOMAIN
IN-CORE PROGRAMS TO PERFORM SPECTRAL OPERATIONS (UPCON, MAGPOL, BANPASS, STRKPASS, DERIV). FUNCTIONS PERFORMED BY THIS PROGRAM INCLUDE:
- REMOVAL OF THE MEAN FROM THE DATA
- OPTIONAL WINDOWING OF THE EDGES OF THE DATA SET
- PADDING OF THE DATA SET WITH ZEROS TO ACHIEVE NECESSARY SIZE (A POWER OF TWO)
- FORWARD TRANSFORM OF THE DATA

REQUIRED SUBROUTINES:
F1T2D, FORK, DATWND
DIMENSIONING REQUIREMENTS:

X(2,N,N) WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
H(N,N) OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
POWER OF TWO (2,4,8,16...).

NOTE: DIMENSIONS IN EVERY SUBROUTINE MUST BE
SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.

AUTHOR: SUBROUTINES FFT2D AND FORK ARE MODIFIED FROM JON REED,
Purdue University, December 1980.
All other code written by:
JEFFREY E. LUCIUS
Geophysical Interactive Computing Laboratory
Department of Geology and Geophysics
The Ohio State University
Columbus, Ohio 43210

March 25, 1985 (Revised Dec 5, 1986)

Revised once again for DEC workstations on 6 Apr 90 so that
that this beast is actually user friendly!

Revised again (Judas priest this is getting old) on
1 Aug 90 into this present format of all fourier programs
combined into this program.

********************************************************************

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 ..... CALCULATE AND REMOVE THE MEAN
C
nxin=col
nyin=row
ICOL=nxin
ICRY-nyin
XMEAN1=SUM/FLOAT (NXIN*NYIN)
DO 210 IY-I,NYIN
DO 210 IX-I,NXIN
X (I, IX, IY)-X (I, IX, IY)-XMEAN1
210 CONTINUE
C WRITE (25, 1020) XMEAN1

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
C CALCULATE AND REMOVE THE MEAN INTRODUCED BY TAPERING
C
TSUM=0.0
DO 214 IY-I,NYOUT
DO 214 IX-I,NXOUT
TSUM=TSUM+X (I, IX, IY)
214 CONTINUE
XMEAN2=TSUM/FLOAT (NXOUT*NYOUT)
DO 215 IY-I,NYOUT
DO 215 IX-I,NXOUT
X (I, IX, IY)=X (I, 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 WAVELENGTH DOMAIN
C
NX=NXOUT
NY=NYOUT
CALL FFT2D (NX, NY, -1)
C
mean=xmean
IF (num .eq. 1) then
DO 500 ly=1,ny
DO 500 lx=1,nx
XDATA (lx, ly)=H (lx, ly)
500 CONTINUE
ELSEIF (num .eq. 2) then
DO 580 ly=1,ny
DO 580 lx=1,nx
YDATA (lx, ly)=H (lx, ly)
580 CONTINUE
ENDIF

C-15
c return

1020 FORMAT('MEAN REMOVED ',F15.7)
1030 FORMAT(8x,'NXOUT MUST BE A POWER OF 2; SPA2FRQ FATAL!')
1040 FORMAT(8x,'NYOUT MUST BE A POWER OF 2; SPA2FRQ FATAL!')
1080 FORMAT('TOTAL FFc_ILN REMOVED ',F15.7)
C
************************************************************************
C SUBF_()) FFT2D (NX, NY, NSIGN)
C************************************************************************
C "FFT2D" PERFORMS BOTH A FORWARD OR INVERSE FAST FOURIER TRANSFORM. "FFT2D" IS THE DRIVER THAT PASSES THE CORRECT VECTORS TO "FORK" WHICH PERFORMS THE ACTUAL TRANSFORMING. THE DIMENSIONING OF "H" MUST BE THE SAME AS IN THE MAIN PROGRAM.
C "NSIGN" = DIRECTION OF DESIRED TRANSFORMATION
C -1 INVERSE TRANSFORM (FREQUENCY TO SPATIAL)
C +1 FORWARD TRANSFORM (SPATIAL TO FREQUENCY)
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
  CALL FORK (NX,H(I,1),SIGNI)
C C .... OPERATE BY COLUMNS
C DO 104 IX=1,NX
  DO 102 IY=1,NY
    CTEMP (IY)=H (IX, IY)
    CALL FORK (NY, CTEMP,SIGNI)
  DO 103
  H (IX,IY) =CTEMP (IY)
  CONTINUE
C
RETURN
C
105 FORMAT(5X, '"NSIGN" MUST EQUAL +1 OR -1 FOR "FFT2D", FATAL')
C
END
C************************************************************************
C SUBROUTINE FORK (LXX,CX,NSIGN)
C************************************************************************
C FAST FOURIER TRANSFORM, MODIFIED FROM CLAERBOUT, J.F., FUNDAMENTAL OF GEOPHYSICAL DATA PROCESSING, McGraw-Hill, 1976
C FORK USES COOLEY-TUKEY ALGORITHM.
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 "NSIGN" = DIRECTION OF DESIRED TRANSFORMATION
C -1. INVERSE TRANSFORM (FREQUENCY TO SPATIAL)
C +1. FORWARD TRANSFORM (SPATIAL TO FREQUENCY)
C NORMALIZATION PERFORMED BY DIVIDING BY 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=IAX

C-16
102 IF (J.GT.M) THEN
  J=J-M
  M=M/2
ENDIF
J=J+M
103 CONTINUE
104 L=1
ISTEP=2*L
CON2=(0.0, 3.14159265)/FLOAT(L)*SIGNI
DO 105 M=1, L
  CM=CEXP(CON2*FLOAT(M-1))
  DO 105 I=1, LX, ISTEP
    CTEM=CM*CX(I+L)
    CX(I+L)=CX(I+L)-CTEMP
  enddo
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=LX, 1, -ISTEP
  CX(I)=CX(I)*SC
106 RETURN
END

************************************************************************
C
SUBROUTINE DATWND (PRCNT, NXI, NYI, NX, NY, fold, itypefold)
************************************************************************
C *
C "DATWND" MULTIPLIES THE INPUT F(I,×,Y) BY A HALF BELL OF A HAMMING-
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 dimension holdme (512,512)
C COMMON F (2,512,512)
C
nxl=nx11
nyl=ny11
C----------------------------------- fold out the data based on percentage
C if (fold.gt.0.0 .and. fold.lt.100.0) then
C  MX=Int(fold*FLOAT(NX1)/100.0+0.5)
  MY=Int(fold*FLOAT(NY1)/100.0+0.5)
  if (kx=ny1 .gt. nx) kx=(nx-ny1)/2
  if (ky=ny1 .gt. ny) ky=(ny-ny1)/2
  do j=1, nyl
    if (l.le.kx) f(l,i,j)=holdme(kx-l+i,j)
    if (l.gt.kx .and. l.le.(kx+nx1)) f(l,i,j)=holdme(l-kx,j)
    if (l.gt.(kx+nx1)) f(l,i,j)=holdme(2*nx1+kx+l-1,j)
  enddo
C----------------------------------- fold out the columns in each row:
C   if itypefold is 1 then the data is considered to
C   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 symmetric
C   with respect to the individual edge,
C   if (itypefold .eq. 1 ) then
    do j=1, nyl
      if (l.le.kx) f(l,i,j)=holdme(kx-l+i,j)
      if (l.gt.kx .and. l.le.(kx+nx1)) f(l,i,j)=holdme(l-kx,j)
      if (l.gt.(kx+nx1)) f(l,i,j)=holdme(2*nx1+kx+l-1,j)
    enddo
c------------------------------ if typefold 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
else if (typefold .eq. 0) then
    do j=1, nyl
       do i=1, nxl+kx+kx
          if (i.le.kx) f(i,i,J)=holdme(nxl+i-kx,J)
          if (i.gt.kx .and. i.le.kx+nxl) f(i,i,J)=holdme(i-kx,J)
          if (i.gt.kx+nxl) f(i,i,J)=holdme((i-nxl-kx),J)
       enddo
    enddo
endif

    c if (nyl .eq. i) go to 333
    do j=1, nyl
       do i=1, nxl+kx+kx
          holdme(i,J)=f(i,i,J)
       enddo
    enddo

    c fold out the rows in each column
    do i=1, nxl+kx+kx
       do j=1, ny+ky+ky
          if (j.le.ky) f(i,i,J)=f(i,i,J-ky)
          if (j.gt.ky .and. j.le.(ky+nyl)) f(i,i,J)=holdme(i,J)
          if (j.gt.(ky+nyl)) f(i,i,J)=f(i,i,2*nyl+ky-J)
       enddo
    enddo
    nyl=nyl+2*ky
    333 nxl=nxl+2*kx

endf

    c if (prcnt.gt.0.0 .and. prcnt.lt.50.0) then
    mx=ifix(prcnt*float(nx)/100.0+0.5)
    my=ifix(prcnt*float(ny)/100.0+0.5)

    c ..... APPLY WINDOW TO COLUMNS
    c
    kypi=3.14159265/float(ky)
    do iy=1,ky
       factor=0.5* (1.0+cos(float(ky-iy+1)*kypi))
       iy=ny+1-iy+1
       do ix=1,nx
          f(i,iy)=f(i,ix,iy)*factor
       enddo
    enddo
    10

    c ..... APPLY WINDOW TO ROWS
    c
    kxpi=3.14159265/float(kx)
    do ix=1,kx
       factor=0.5* (1.0+cos(float(kx-ix+1)*kxpi))
       ix=ny+1-ix+1
       do iy=1,ny
          f(iy,i)=f(iy,ix,i)*factor
       enddo
    enddo

    c      WRITE(25,150) KX, KY
    write (25,*) kx, ky
endf

    c      c ..... center and ZERO OUT REMAINDER OF ARRAY
    c
    nxhalf=(nx-nxl)/2
    nyhalf=(ny-nyl)/2
    do i=1,nx
       do j=1,ny
          holdme(i,j)=f(i,i,j)
       enddo
    enddo
    do i=1,nx-nxhalf
       do j=1,ny
          if (j.le.nyhalf) f(i,j)=0.0
          if (j.gt.nyhalf .and. j.le.ny+nyhalf) > f(i,j)=holdme(i-nxhalf,j-nyhalf)
          if (j.gt.ny+nyhalf) f(i,j)=0.0
       enddo
    enddo
    do i=nx-nxhalf+1,nx

C-18
do j=1,ny
   f(1,1,j)=0.0
endo

RETURN

C

FORMAT('smoothed',14,' values on both x edges'/,
       ',14,' y'
C
160 FORMAT(1H- ,''PRCNT'' ,',F7.3,' OUTSIDE OF PROPER RANGE',
       > ' NO WINDOWING PERFORMED: ''DATWND''/
C
END

C

C subroutine filter (num)
C
integer num, npass, lmean, nwind, nxout, nyout,
>   row, col
real prcnt, xlag, delta, cuthi, cutlo
complex xcdata(512,512), ycdata(512,512)
common /fftlfft/ nxout, nyout, prcnt, lmean, fold, itypefold
common /comp/ xcdata, ycdata
common /ihbflt/ delta, cuthi, cutlo, xlag, npass, nwind
COMMON H(512,512)
COMPLEX H
C
if (num .eq. 1) then
row=nyout
col=nxout
ny=nyout
elseif (num .eq. 2) then
row=nyout
col=nxout
ny=nyout
endif
C
C **********************************************************************
C
C PROGRAM BANDPAS
C
C PROGRAM BANDPASS PERFORMS HIGH, LOW, OR BANDPASS WAVELENGTH
FILTERING OF UNIFORMLY GRIDDED ARRAYS. INPUT MATRIX IS THE
WAVELENGTH DOMAIN TRANSFORM AS OUTPUT BY SPA2FRQ. AN IDEAL FILTER
IS CONSTRUCTED IN THE WAVELENGTH DOMAIN, WINDOWED IN THE SPACE
DOMAIN, THEN TRANSFORMED BACK INTO THE WAVELENGTH DOMAIN TO BE
MULTIPLIED BY THE INPUT TRANSFORM.
C
C ..... REQUIRED SUBROUTINES :
C
BNDPAS, FFT2D, FORK, STORE, WINDOW
C
C ..... DIMENSIONING REQUIREMENTS :
C
H(N,N) WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
INPUT AND OUTPUT TRANSFORMED MATRIX. N MUST BE AN
INTEGRAL POWER OF TWO (2,4,8,16...).
NOTE : DIMENSIONS IN EVERY SUBROUTINE MUST BE
SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.
C
C ..... AUTHOR : JON REED, PURDUE UNIVERSITY, DECEMBER 1980.
REVISIONS BY STEVE MATESKON AND JEFF LUCIUS,
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
update: 2 feb 90, removed need for cstore array
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
WRITE (30,*) NX,NY,IZERO,XMEAN
DO 200 IT=1,NY
WRITE (30,*) (H(IX, IY),IX=1,NX)
200 CONTINUE
C
WRITE (6, 1040) NX,NY
C
ENDIF
C
IF (num .eq. 1) then
DO 500 =1,row
DO 500 =1,col
xcdata(j,i) = xcdata(j,i)*h(j,i)
500 continue
else if (num .eq. 2) then
DO 580 =1,row
DO 580 =1,col
ycdata(j,i) = ydata(j,i)*h(j,i)
580 continue
end
C
RETURN
END
C
************************************************************************
C
SUBROUTINE BNDPAS (CCUTLO, CCUTHI, NPASS, DELTA, NX, NY)
C
************************************************************************
C
* "BNDPAS" CALCULATES TWO QUADRANTS OF THE WAVE# RESPONSE OF *
C AN IDEAL BANDPASS FILTER OF A (NX, NY) MATRIX. *
C "H" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM *
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 WAVE NUMBERS BETWEEN THE 2 WAVE NUMBERS *
C = 1 PASS WAVE NUMBERS BETWEEN THE 2 WAVE NUMBERS *
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 "ICOL", 16,32,ETC) *
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
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
NX=-NX*2
NY=-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
RNY2 = (FNQ1/(ANY2*CUTHI))**2
NHIY - CUTLO*WAVLEN*ANY2+1.0
NHIY=MIN(NHIY,NY2)
CLOWF=FLOAT(NX2)*WAVLEN*CUTLO
CHIX =FLOAT(NX2)*WAVLEN*CUTHI
C
IF (CUTLO.LE.0.000001) THEN
  WRITE(6,152) RLOWY=0.0
ELSE
  IF (FNQI-CUTHI.LE.0.000001) THEN
    WRITE(6,153) RLOWY2=(FNQI/(ANY2*CUTLO))**2
    NLOWY=CUTLO*WAVLEN*ANY2+1.0
  ELSE
    WRITE(6,154) RLOWY2=(FNQI/(ANY2*CUTLO))**2
    NLOWY=CUTLO*WAVLEN*ANY2+1.0
  ENDIF
ENDIF
C
C....."ZERO" OUT THE PART OF ARRAY TO BE ALTERED
C
DO 35 IX=1,NX2
DO 35 IY=1,NY2
H(IX, IY) = ZERO
35 CONTINUE
C
C.....OPERATE ON ROWS WHERE SOME WAVENUMBERS ARE .LE. CUTLO
C
IF (NLOWY.NE.0) THEN
  MINS=I
  MAXS=NX2
  IF (NLOWY.EQ.1) THEN
    MINS=NX2*CUTLO+1.0
    MAXS=NX2*CUTHI+1.0
  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 (FNQI-CUTHI.GE.0.00001) THEN
      MAXX=CHIX*SQR(1.0-Y2*RHIY2)+2.0001
    ENDIF
    IF (MINX.EQ.1) THEN
      H(I, 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
ELSE
  IF (NLOWY+I.GT.NY2) RETURN
  MINS=I
  MAXS=NLOWY+1
  DO 200 IY=MINS,MHIY
    Y2=FLOAT(IY-1)**2
    MAXX=MAXS
    IF (FNQI-CUTHI.GE.0.00001) THEN
      MAXX=HIX*SQR(1.0-Y2*RHIY2)+2.0001
    ENDIF
    IF (MAXX.GT.1) THEN
      DO 180 IX=2,MAXX
        H(IX, IY)=ONE
      180 H(IX, IY)=ONE
    ENDIF
  200 CONTINUE
ELSE
  IF (NLOWY+1.GT.NY2) RETURN
  MINS=I
  MAXS=NLOWY+1
  DO 215 IY=MINS,MAXS
    H(I, IY)= ONE
  215 CONTINUE
ENDIF
C
RETURN
C
112 FORMAT(//IX, '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',3X,F15.5, ' WAVELENGTH EQUIVALENT'
             ' HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
            C-21
C-22

SUBROUTINE STORE(NNX, NNY)
COMMON H(512,512)
COMPLEX H
IF (NNY.EQ.1) RETURN
NX=NNX
NY=NNY
NXX=NX+2
NYX=NY+2
NYI=NYX-1
DO 15 IY-NYX,NY
NYY=NYX-1-I
H(I,IY)=H(NXX-I, NYY)
DO 10 IX-2,NX
H(IX,IY)=CONJG (H(NXX-IX, NYX))
10 CONTINUE
H(NXX, IY)=H(NXX, NYX)
15 CONTINUE
RETURN
END

SUBROUTINE WINDOW (NX, NY, XLAG, NWIND)

"WINDOW" PERFORMS 2-DIMENSION WINDOWING OVER A 4 QUAD. DATA ARRAY *
EACH QUAD. IS SEPARATELY WINDOWED. THE 1.0 COEFFICIENT IS ALWAYS *
THE OUTER MOST CORNER OF THE ARRAY. *
FOR ONE DIMENSIONAL WINDOW, LET NY=1 *
"NX" = NUMBER OF COLUMNS IN DATA MATRIX *
"NY" = NUMBER OF ROWS IN DATA MATRIX *
"XLAG" = SMOOTHING PARAMETER FOR WINDOWING IDEAL FILTER IN SPATIAL *
DOMAIN. DETERMINES WHAT PERCENTAGE OF DATA IS WINDOWED *
(NX*XLAG/100.0) THE REMAINDER IS SET TO 0.0. I.E. THE *
SMALLER "XLAG" THE SMOOTHER THE WINDOWING. *
"XLAG" MUST BE .GT. 0.0 .AND. LE. 100.0 FOR WINDOWING *
VALUES OUTSIDE OF THIS RESULTS IN NO WINDOWING *
THE SMALLER THE "XLAG" THE SMOOTHER THE FILTER. *
"NWIND" = TYPE OF WINDOW TO APPLY *
-0 GIVES NO WINDOW *
-1 gives a rectangular window *
-2 GIVES BARTLETT WINDOW(TRIANGLE WINDOW) *
-3 GIVES HAMMING-TUKEY WINDOW *
-4 GIVES PARZEN WINDOW *

COMMON H(512,512)
COMPLEX H
IF (XLAG.LE.0.0 .OR. XLAG.GT.100.0) THEN
WRITE (6, 50) XLAG
RETURN
ENDIF
LAG=FLOAT(NX)*XLAG/200.0+0.5
PI=3.14159265
NX=NX+2
NY=NY+2
NXR=FLOAT(NX)+1
NXR=FLOAT(NX)
NXR=1.0/FLOAT(NX)
NY=NY+1
NY=1.0/FLOAT(NY)
RADX=FLOAT(NX)*RADX
RADX=1.0/RADUS
RAD2= RADUS*RADUS
NRAD=FLOAT(NY)*RADUS+1.0001

C-22
C....APPLY RECTANGULAR WINDOW TO FILTER
C
IF (NWIND.EQ.1) THEN
  IF (NRAD.NE.0) THEN
    MAX=RADIUS*XNXR+1.0001
    H(1,1)=(0.0,0.0)
    MAX=MAX+1
  ENDIF
  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
  IF (NRAD.NE.1.AND.NRAD.NE.0) THEN
    DO 102 JY=2,NRAD
      IYY=NYY-JY
      YLEN2=(FLOAT(JY-1)*YNY)**2
      MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
    102 CONTINUE
  ENDIF
  WRITE (25,660) XLAG, LAG
  IF (NRAD.EQ.NY2) RETURN
  LL=NRAD+1
  DO 108 I=LL,NY2
    IYY=NYY-I
    H(I,1)=(0.0,0.0)
    H(I,1YY)=(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....APPLY BARTLETT WINDOW TO FILTER
C
ELSEIF (NWIND.EQ.2) THEN
  IF (NRAD.NE.0) THEN
    MAX=RADIUS*XNXR+1.0001
    X=FLOAT(LL-1)*XNX
    FACTOR=1.0-X*X*RA
    H(LL,1)=H(LL,1)*FACTOR
    MX=NXX-LL
  253 H(MX,1)=H(MX,1)*FACTOR
  ENDIF
  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
  IF (NRAD.NE.1.AND.NRAD.NE.0) THEN
    DO 202 IY=2,NRAD
      IY=FLOAT(IY-1)*YN
      YLEN2= X*X
      MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
      FACTOR=1.0-X*X*RA
      H(I,1YY)=H(I,1YY)*FACTOR
      H(I,Y)=H(I,Y)*FACTOR
  202 CONTINUE
  ENDIF
  IF (NRAD.NE.1.AND.NRAD.NE.0) THEN
    DO 203 IL=2,NX2
      XI=FLOAT(IL-1)*XNX
      YLEN2=XI*X
      MAX=SQRT((FLOAT(IL-1)*XNX)**2+YLEN2)
      FACTOR=1.0-XI*RA
      H(IL,Y)=H(IL,Y)*FACTOR
      H(IL,YY)=H(IL,YY)*FACTOR
      H(LL,YY)=H(LL,YY)*FACTOR
  203 CONTINUE
  ENDIF
C-23
C-24
H(NX-3, IY) = (0.0, 0.0, 0.0)
H(J, I) = (0.0, 0.0, 0.0)
309 H(NX-3, I) = (0.0, 0.0, 0.0)
308 CONTINUE
C . . . APPLY PARZEN WINDOW TO FILTER
C ELSEIF (NRAD .EQ. 4) THEN
C
IF (NRAD, NE, 0) THEN
MAX=RX0R*I*NRR+1.0001
NRAD=FLOAT(NY2)*RADIUS+0.5+1.0001
MAX2=SQRT(RAD2/4.0)*XRNR+1.0001
FACTOR=1.0-6.0*(XI*RAD1)**2-(XI*RAD1)**3
H(I, 1) = H(I, 1)*FACTOR
C
IF (MAX2 .GE. 2) THEN
DO 453 LL=2, MAX2
XI=FLOAT(LL-1)*XNN
FACTOR=1.0-6.0*(XI*RAD1)**2-(XI*RAD1)**3
H(LL, I) = H(LL, I)*FACTOR
MX=MAXX-LL
453 H(MX, 1) = H(MX, 1)*FACTOR
ENDIF
C
KOUNT=MAX2+1
DO 457 LL=KOUNT, MAX
XI=FLOAT(LL-1)*XNN
FACTOR=2.0*(1.0-(XI*RAD1)**3)
H(LL, 1) = H(LL, 1)*FACTOR
MX=MAXX-LL
457 H(MX, 1) = H(MX, 1)*FACTOR
ENDIF
C
IF (NRAD, NE, 1) AND (NRAD, NE, 0) THEN
DO 402 IY=2, NRAD
IY=NY-IY
XI=FLOAT(IY-1)*YNN
YLEN2= XI*XI
MAX=SQRT(RAD2-YLEN2)*XRNR+1.0001
NRAD=FLOAT(NY2)*RADIUS+0.5+1.0001
C
IF (IY. GT. N2RAD) THEN
KOUNT=2
FACTOR=2.0*(1.0-(XI*RAD1)**3)
H(I, IYY) = H(I, IYY)*FACTOR
H(I, IY) = H(I, IY)*FACTOR
ELSE
MAX=SQRT(RAD2/4.0-YLEN2)*XRNR+1.0001
FACTOR=1.0-6.0*(XI*RAD1)**2-(XI*RAD1)**3
H(I, IYY) = H(I, IYY)*FACTOR
H(I, IY) = H(I, IY)*FACTOR
C
IF (MAX2 .GE. 2) THEN
DO 403 LL=2, MAX2
XI=SQRT((FLOAT(LL-1)*XNN)**2+YLEN2)
FACTOR=1.0-6.0*(XI*RAD1)**2-(XI*RAD1)**3
H(LL, IYY) = H(LL, IYY)*FACTOR
H(LL, IY) = H(LL, IY)*FACTOR
MX=MAXX-LL
403 H(MX, IYY) = H(MX, IYY)*FACTOR
ENDIF
C
KOUNT=MAX2+1
ENDIF
C
DO 407 LL=KOUNT, MAX
XI=SQRT((FLOAT(LL-1)*XNN)**2+YLEN2)
FACTOR=2.0*(1.0-(XI*RAD1)**3)
H(LL, IYY) = H(LL, IYY)*FACTOR
H(LL, IY) = H(LL, IY)*FACTOR
MX=MAXX-LL
407 H(MX, IYY) = H(MX, IYY)*FACTOR
ENDIF
C
WRITE(25, 663) XLAG, LAG
IF (NRAD, NE, NY2) RETURN
C
C
```fortran
LL=NRAD+1
DO 408 I=LL,NY2
   IYY=NY2
   H(I,IYY)=(0.0,0.0)
   DO 409 J=2,NX2
      H(J,IYY)=(0.0,0.0)
      H(NXJ-IYY)=(0.0,0.0)
   END DO
409   H(J, I)=(0.0,0.0)
408 CONTINUE
C DO NOT APPLY A WINDOW TO FILTER
C ELSEIF (NWIND. EQ. 0) THEN
C   WRITE (25,664)
C ENDIF
C RETURN
C FORMAT (3X,'INPUTTED XLAG OF ',F5.2,' EXCEEDS PERMISSIBLE ',
C          'RANGE OF 0.0 AND .LE. 100.0, NO WINDOWING PERFORMED')
C FORMAT ('RECTANGULAR WINDOW USED XLAG-',F7.3,4X,'LAG-',I5)
C FORMAT ('HAMEE4-TUKEY WINDOW USED XLAG-',F7.3,4X,'LAG-',I5)
C FORMAT ('NO WINDOWING HAS BEEN APPLIED ; XLAG-',F7.3)
CEND
C subroutine correlate(xpassno,ypassno)
integer xrow,xcol,yrow,ycol,nxout,nyout,xpassno,ypassno,zerocnt(512,512),cnwlnd
real mincc,maxcc,ccwinout,prcnt,pctpr3,pctpr4,minccin,maxccin,cxlag
complex h(512,512),power5,power6, totpwr
REAL  X(512,512),Y(512,512),zero,
POWERI,POWER2,POWER3,POWER4,XPOWER, TPOWER
REAL CCOCF,CCIN,CCOUT
DATA ZERO/(0.000000, 0. 000000 )/
common /rowcol/ xrow,xcol,yrow,ycol
common /comps/ x,y
common /ccflt/ mincc,maxcc,maxccin,ccwlnd,cxlag
common /fftifft/ nxout,nyout,prcnt,imean,foil,itypefoil
common h
c-------------------------- subroutine description
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 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 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
C ny=nyout
C if (xcol.ne. ycol .or. xrow .ne. yrow ) then
C   write (*,*) 'NO MATCH BETWEEN ROW OR COLUMN'
C   write (*,*) 'CORRELATION COEF MAY NOT BE CORRECT'
C endif
C p1=3.141592654
C twpol=6.283185307
C POWERI=ZERO
C POWER2=ZERO
C POWER3=ZERO
C POWER4=ZERO
C XPOWER=ZERO
C TPOWER=ZERO
C DO 110 i=1,NX
C-------------------------------------- zerocont 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
```
the principle.

SUM THE POWERS & CROSS PRODUCTS OF THE INPUT MAPS.

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)))

xrad is the phase angle of the x array wavenumber and
yrad is the phase angle of the y array wavenumber, the
cosine of the minimum phase difference is the correlation
of the two wavenumbers, to find the minimum phase difference
it is necessary to adjust xrad or yrad with integer values
of pl. so...do not change the order of the if statements!!

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
delrad=abs(xrad-yrad)
coef=cos(delrad)

if (CCOEF .GT. maxcc .or. CCOEF .LT. mincc) THEN
  X(I,J)=ZERO
  Y(I,J)=ZERO
  zerocnt(I,J)=0
ENDIF

SUM THE POWERS & CROSS PRODUCTS FOR THE OUTPUT MAPS.

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)))

CALCULATE THE C.C. FOR THE INPUT MAPS.

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

CALCULATE THE C.C. FOR THE OUTPUT MAPS.

if (power3 .eq. zero .or. power4 .eq. zero) then
  write ('*','power3 =',power3,xpassno
  write ('*','power4 =',power4,ypassno
  ccout=9999.9
else
  COUT=REAL( TPOWER/SQRT (POWER3*POWER4))
endif

CALCULATE THE PERCENTAGE OF THE POWER RETAINED IN THE FILTERED
MAPS.

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

WRITE THE C.C. FOR THE INPUT & OUTPUT MAPS TO FILE 6.

WRITE (6,444) CCIN
WRITE (6,555) CCOUT

WRITE THE POWER PERCENTAGES TO FILE 6.

WRITE (6,666) PCTPR1,PCTPR2

444 FORMAT (' ', 'THE CORRELATION COEFFICIENT BETWEEN THE INPUT'
     ' MAPS IS ',F6.3)
555 FORMAT (' ', 'THE CORRELATION COEFFICIENT BETWEEN THE OUTPUT'
     ' MAPS IS ',F6.3)
666 FORMAT (' ', 'THE PERCENTAGE OF THE TOTAL POWER IN MAP ONE',
     ' PASSED IS ',F7.3,'%', ' THE PERCENTAGE OF THE TOTAL POWER',
     ' IN MAP TWO PASSED IS ',F7.3,'%')

write (25,888) xpassno,ypassno,ccin,ccout,pctpr1,pctpr2
888 format (216,4f10.3)
if (ccin .lt. minccin) write ('*','xpassno,ypassno,ccin, ' <min'

C-27
if (ccin > maxccin) write ('*'), xpassno, ypassno, ccin, ' > max'

------------------------------------------------------------------
the following if statement controls the windowing functions for smoothing
the output arrays and calculates a new output correlation coefficient and percent of power retained in the
windowed arrays because the data will change slightly with windowing

write ('*'), '1 = zero cnt, 0 != zero cnt'
read ('*') l
if (l .eq. 1) then
write (50, *) nx
write (50, *) ny
do i = 1, ny
write (50, 9970) (zerocnt(j,i), j = 1, nx)
enddo
format (20(13,1x))
endif
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) = (0, 0, 0, 0)
if (zerocnt(j,i) .eq. 0) h(j,i) = (0, 0, 0, 0)
300 continue
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/power2)*100.0
pctpr4 = (power6/power2)*100.0
340 continue
write (25, 888) xpassno, ypassno, ccin, ccwinout, pctpr3, pctpr4
format (2i6, 4f10.3)
endif
return
end

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, lmean, fold, ltypefold
COMMON H(512,512)
DIMENSION X(2,512,512), holdme(361,361)
COMPLEX X
EQUIVALENCE (X(1,1), H(1,1))

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) = xdata(i, j)
 50 continue
endif
else (num .eq. 2) then
C-28
PROGRAM FRQ2SPA

PROGRAM FRQ2SPA INVERSE TRANSFORMS AN N X N MATRIX OF WAVE NUMBER
DOMAIN COEFFICIENTS INTO THE N X N MATRIX OF SPACE DOMAIN
AMPLITUDES. FUNCTIONS PERFORMED BY THIS PROGRAM INCLUDE:
- INVERSE TRANSFORM OF THE DATA
- RESTORING THE MEAN TO THE DATA
- CALCULATION OF SPACE DOMAIN MAXIMUM AND MINIMUM AMPLITUDES

....REQUIRED SUBROUTINES:
- FFT2D, FORK

....DIMENSIONING REQUIREMENTS:
- X(2,N,N) WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
  OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGRAL
  POWER OF TWO (2, 4, 8, 256...).
- NOTE: DIMENSIONS IN EVERY SUBROUTINE MUST BE
  SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.

....AUTHOR: JEFF LUCIUS
- DEPARTMENT OF GEOLOGY AND MINERALOGY
- OHIO STATE UNIVERSITY, DECEMBER 1984.
- revised: 8 AUG 90
- updated: 2 Feb 91
- added do loops that find the data portion of the
  zero centered inverse transformed data. a look at
  subroutine datwnd will help figure this out.

C..... INVERSE TRANSFORM DATA TO THE SPACE DOMAIN

icol=col
CALL FFT2D (NX, NY, +1)
nhalf=(nx-icol)/2
nyhalf=(ny-icol)/2
DO I=nhalf+1,nxhalf+icol
  DO J=nyhalf+1,nyhalf+1row
    holdme (i-nxhalf,j-nyhalf)=x(i,j)
  ENDDO
ENDDO

DO 210 J=1,irow
  DO 210 I=1,icol
    x(i,j)=holdme(i,j)
  210 CONTINUE

xmean=total/float(irow*jrow)
IF (JMEAN.EQ.1) THEN
  DO 215 J=1,irow
    DO 215 I=1,icol
      x(i,j)=x(i,j)+xmean
  215 CONTINUE
ENDIF

xMIN=1.0E20
xMAX=-1.0E20
DO 220 J=1,irow
  IF (num.eq.1) THEN
    DO 220 I=1,icol
      xdata(i,j)=x(i,j)
  ELSEIF (num.eq.2) THEN
    DO 220 I=1,icol
      xdata(i,j)=x(i,j)
  ENDIF
  DO 220 I=1,icol
    xMIN=MINI (xMIN, x(i,j))
XMAX=XMAX1(OMAX, X(I, I, J))
IF(XMAX.EQ.X(I, I, J)) THEN
IOMAX=I
JOMAX=J
ENDIF
IF(XMIN.EQ.X(I, I, J)) THEN
IMIN=I
JMIN=J
ENDIF
220 CONTINUE
WRITE(25,1020) XMAX, XM, XMIN, XM, XMIN, mean, passno
WRITE(25, 9980) passno, mean, XM, XM, XMIN, XMIN, mean, passno
9980 format (i5,2x,f13.5,2x,f13.5,2x,i4,2x,14,f13.5,2x,i4,2x,14)
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, I)
CALL WINDOW(NX, NY, sLAG, swind)
CALL FFT2D(NX, NY,-I)
else
endif
C
C ********** STRIKE **PERFORMS A STRIKE SENSITIVE FILTERING (FAN FILTER)**********
C ON UNIFORM GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.
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 updates and revisions:
C 23 dec 91: added this strike pass routine to the fourier program. required removal of write statements
C
CALL STRIKE(ANG1,ANG2,spass,NX,NY)
C-------------- CREATE SMOOTHED FILTER
C IF(sLAG.gt.0.0 .and. sLAG lt.100.0) then
C IF(swind.gt.0 .and. swind.le.4) then
C CALL FFT2D(NX, NY, I)
C CALL WINDOW(NX, NY, sLAG, swind)
C CALL FFT2D(NX, NY,-I)
C else
C endif
C-------------- SET UP TO WRITE 30 IF DESIRED
C DO 356 IY-I,NY
C WRITE(30) (H(IY,JX),IY-I,NX)
C 356 CONTINUE
C-------------- ACCESS TRANSFORM OF DATA & MULTIPLY *FILTER (CONVOLVING)
C IF(num .eq. 1) then
C do 500 I=1,row
C do 500 J=1,col
C xcdata(J,I) = xcdata(J,I)*h(J,I)
C 500 continue
C
SUBROUTINE STRIKE (AANGI, AANG2, NNPASS, NNX, NNY)

COMPLEX H, ZERO, ONE
COMMON H(512, 512)
DATA DG2RAD, DG90 /0.017453293, 1.570796327/

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, GE 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 "NX", 16, 32, ETC) MAX-128
C "NY" = NUMBER OF ROWS (POWER OF 2 GE "NY", 16, 32, ETC) MAX-128
C
C
NX=NNX
NY=NNY
ANG1 = AANG1
ANG2 = AANG2
NPASS = NNPASS

IF(ANG1.LT.ANG2 .AND. ANG1.GE.0.0 .AND. ANG2.LE.180.0) GOTO 109
WRITE (6, 125)
125 FOPJ_T(5X, 'ILLEGAL SPECIFICATION OF STRIKE ANGLES, FATAL')
STOP
109 CONTINUE

NX2=NX/2+1
NY2=NY/2+1
NX1=NX+1
NXX=NX+2
NY=Y+2
XY=FLOAT(NX2)/FLOAT(NY2)

ZERO=(0.0,0.0)
ONE=(1.0,0.0)
IF(NPASS.NE.-1) GOTO 160
ZERO=(0.0,0.0)
ONE=(1.0,0.0)
160 CONTINUE

C 'ZERO' OUT ARRARY
C
DO 15 IY=1, NY
DO 15 IX=1, NX
15 H(IX, IY)=ZERO

C COMPUTE PARAMETERS FOR SOUTHWEST QUADRANT OF MATRIX
C
IF(ANG2.GE.90.0) GOTO 20
IYMAX=0
GOTO 200
20 IF(ANG1.GT.90.0) GOTO 114
A1=0.0
TA1=0.0
IYMAX=NY2
ADA1=1.5
GOTO 113
114 CONTINUE
A1=ACAN(TAN((ANG1-90.0)*DG2RAD)*XY)
TA1=TAN(A1)
IYMAX=FLOAT(NX2)/TA1+1.0
IYMAX=AMIN0(IYMAX, NY2)
ADA1=1.0
113 CONTINUE
IF(ANG2.LT.180.0) GOTO 115
A2=0.0
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
GOTO 200

116 CONTINUE

C 200 IF(ANG1.LE.90.0) GOTO 25
IYMAX=0
GOTO 300

C **********************************************************************
C COMPUTE P_TERS FOR SOUTHEAST QUADRANT OF MATRIX
C **********************************************************************
C
C 25 IF(ANG1.GT.0.0) GOTO 45
A1=0.0
TTA1=0.0
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
IYMAX=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
IYMAX=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
IYMAX=ABS(FLOAT(NX2)/TTA2+1.5)
IYMAX=AMIN(IYMAX, NY2)

122 CONTINUE

C ********************************************************************************
C CALCULATE THE FILTER COEFFICIENTS
C ********************************************************************************
C
C 300 NYMAX=AMAX0(IYMAX, IYMAXX)
DO 50 IY=1, NYMAX
Y=FLOAT(IY-1)

C ********************************************************************************
C DEFINE SOUTHWEST QUADRANT
C ********************************************************************************
C
C IF(IYMAX.LT.IY) GOTO 30
MIN=Y*TTA1+ADDA1
MAX=Y*TTA2+ADDA2
MAX=AMIN0(MAX, NX2)
C
C IF(MIN.GT.MAX) GOTO 30
DO 75 IX=MIN, MAX
H(IX,IY)=ONE

75 CONTINUE

C ********************************************************************************
C DEFINE SOUTHEAST QUADRANT
C ********************************************************************************
C
C 30 IF(IYMAX.LT.IY) GOTO 35
MIN=(NX1-(Y*TTA2))+ADDA2
MIN=AMINO(MIN, NX)
MAX=(NX1-(Y*TTA1))+ADDA1
MAX=AMAX0(MAX, NX2+1)
C
C IF(MAX.GT.NX) GOTO 35
DO 275 IX=MAX, MIN
H(IX,IY)=ONE

275 CONTINUE

C ********************************************************************************
C USE ANTI-SYMMETRY TO DEFINE QUADRANTS # 2 & 3
C ********************************************************************************
C
C 35 IF(IY.EQ.1 .OR. IY.EQ.NY2) GOTO 50

C-32
SUBROUTINE UPCON (num)
  integer num, npass, lmean, nwind, nxout, nyout, udnwind,
  > row, col
  complex xcdata(512,512), ycdata(512,512)
  common /fftifft/ nxout, nyout, prcnt, lmean, fold, ltypefold
  common /comps/ xcdata, ycdata
  common /udcont/ delta, zcon, udxlag, udnwind
  COMMON H(512,512)
  COMPLEX H

  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

  CALL CONTIN (DELTA, ZCON, NX, NY)
  CALL STORE (NX, NY)

  smooth the continuation filter
  IF (udxlag.gt.0.0 .and. udxlag.le.100.0) then
    IF (udnwind .eq. 4 .and. udnwind .gt. 0) then
      CALL FFT2D (NX, NY, 1)
      CALL WINDOW (NX, NY, udxlag, udnwind)
      CALL FFT2D (NX, NY, -1)
    endif
  endif

  ACCESS TRANSFORM OF DATA & MULTIPLE FILTER (CONVOLVING)
  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

  return
end

SUBROUTINE CONTIN (DELTA, ZCON, NX, NY)
  COMMON H(512,512)
  COMPLEX H

  "CONTIN" COMPUTES TWO QUADRANTS (NX/2+1 BY NY) OF AN IDEA
  UPWARD OR DOWNWARD CONTINUATION FILTER DIMENSIONED "NX" BY "NY".
  FOR ONE DIMENSION LET NY=1.
  ARRAY "H" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM
  "DELTA" = GRID INTERVAL IN LENGTH UNITS
  "ZCON" = THE DEPTH OR HEIGHT AT WHICH CONTINUATION IS DESIRED.
  (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 *

PI2Z= 3.14159265 *ZCON/DELTA
NX2=NX**2
NY2=NY**2

DO 101 IY=1,NY2
   CON1=(FLOAT (IY-1)*NY2)**2
   S=SQRT (CON1)
   XI=EXP (S*PI2Z)
   H(I,Y)=CMPLX (XI, 0.0)
   DO 101 IX=1,NX2
      S=SQRT ((FLOAT (IX-1)*NX2)**2+CON1)
      XI=EXP (S*PI2Z)
      H(IX,Y)=CMPLX (XI, 0.0)
      H(NXX-IX,Y)=H(IX,Y)
101 CONTINUE
C
RETURN
END

subroutine DERIVA (num)
integer num, npass,imean,nwlnd,nxout, nyout,
> row, col
complex xcdata (512,512),ycdata (512,512)
common /fftlfft/
> nxout, nyout, prcnt, imean, fold, itypefold
common /comps/
> xcdata,ycdata
common /xyzderlv/
> delta, nth,nway

COM_DN H(512,512)
COM_ LEX H

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

"DERIVA" CALCULATES THE "NTH" DERIVATIVE IN THE WAVE# *
DOMAIN OF UNIFORMLY GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.

CREATE DERIVATIVE FILTER
CALL DERIV(NX, NY, NTH, NWAY, DELTA)

ACCESS TRANSFORM OF DATA & MULTIPLE FILTER (CONVOLVING)
if (num .eq. 1) then
   do 500 i=1,row
      do 500 j=1,col
         xdata(j,i) = xdata(j,i)*h(j,i)
   500 continue
elseif (num .eq. 2) then
   do 500 i=1,row
      do 500 j=1,col
         ydata(j,i) = ydata(j,i)*h(j,i)
   500 continue
endif

"DERIVA" CALCULATES THE VALUES OF 2 QUADRANTS FOR THE *
"NTH" DERIVATIVE OF A (NX, NY) MATRIX. THESE VALUES ARE TO *
BE MULTIPLIED BY THE WAVE# SPECTRUM OF THE GIVEN FIELD.

FOR A 1 DIMENSIONAL ARRAY SET "NY"=1

ARRAY "C" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM.

"NNTH" = THE ORDER OF DERIVATIVE DESIRED

"NWAY" = THE DIRECTION THE DERIVATIVE IS TO TAKEN

0 VERTICAL DIRECTION
1 HORIZONTAL DIRECTION (X)
2 HORIZONTAL DIRECTION (Y)

"DELTA" = GRID INTERVAL IN LENGTH UNITS

NX=NXK
NY=NYK
NTH=NNTH
NXX=NXK+2
NYY=NYK+2
CON=(6.2831853, 0.0)

IF (NWAY.GE.1) CON=(0.0, 6.2831853)

NXDEL=1.0/(FLOAT (NX)*DELTA)
NYDEL=1.0/(FLOAT (NY)*DELTA)

NX-NNX
NY-NNY
NTH-NNTH
NXX-NX+2
NX2-NX/2+I
NYY-NY+2
NY2-NY/2+I

C(NXX-IX, IYY) = (CON*FX2)**NTH
C(I, IYY) = (CON*FY2)**NTH
C(IX, NYY-IY) = CON2
C(NXX-IX, IY) = CON2
C(IX, IYY) = CON2

C = (NXX-IX, IYY) = CON3
C(I, IYY) = CON2
DO 305 IY=1,NY
305 C(I, IYY) = CON2
RETURN

C

C TAKE VERTICAL DERIVATIVE IN WAVE# DOMAIN

C******************************************************************

DO 105 IX=2,NX
FX2=(FLOAT (IX-1)*NXDEL) C(IX,1) = (CON*FX2)**NTH
105 C(NXX-IX, 1) = (CON*FX2)**NTH
C(1,1) = (0.0,0.0)

DO 110 IY=2,NY2
FY2=(FLOAT (IY-1)*NYDEL) C(I,1) = (CON*FY2)**NTH
110 C(I,1) = (0.0,0.0)

DO 205 IX=2,NX
C(IX, I) = (0.0,0.0)
205 C(IX, I) = (0.0,0.0)

DO 210 IY=2,NY2
FY2=(FLOAT (IY-1)*NYDEL)*CON
CON2=CONJG (CON)
CON3=CON2
210 C(I, IY) = CON3
C(IX, NYY-IY) = CON3
C(IX, IY) = CON3
C(NXX-IX, IY) = CON3
RETURN

C

C TAKE HORIZONTAL(X) DERIVATIVE IN WAVE# DOMAIN

C******************************************************************

DO 305 IY=1,NY
305 C(I, IY) = (0.0,0.0)

DO 310 IX=2,NX
IXX=NXX-IX
CON2=(FLOAT (IX-1)*NXDEL*CON)**NTH
CON3=CON2
310 C(IX, IY) = CON3
C(IX, NYY-IY) = CON3
C(IX, IYY) = CON3
C(IX, NYY-IY) = CON3
RETURN
else
    write (*,*) 'nway is not equal to 0,1 or 2'
    stop
end
end

subroutine MAG2POL (num)
    integer num, npass, imean, nwind, nxout, nyout,
    > row, col
    complex xdata(512,512), ydata(512,512)
    common /fftifft/ nxout, nyout, prcnt, imean, fold, itypefold
    common /comps/ xdata, ydata
    common /rtp/ azm, xinc, dec
    COMMON H(512,512)
    COMPLEX H
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"MAG2POL" APPROXIMATELY CALCULATES THE CORRESPONDING MAGNETIC
ANOMALY MAP DUE TO AN EARTH'S FIELD VECTOR OF 0.0 DECLINATION,
AND 90.0 INCLINATION FROM AN INPUTTED MAGNETIC ANOMALY MAP WITH
A KNOWN FIELD VECTOR.
C**********************************************************************
CDEC-DEC+AZM
CALL MAGPOL(XINC,DEC,NX,NY)
CACCESS TRANSFORM OF DATA & MULTIPLE FILTER (CONVOLVING)
if (num .eq. 1) then
    do 500 i=1,row
        do 500 j=1,col
            xdata(j,i) = xdata(j,i)*h(j,i)
    500 continue
elseif (num .eq. 2) then
    do 580 i=1,row
        do 580 j=1,col
            ydata(j,i) = ydata(j,i)*h(j,i)
    580 continue
endif
C
return
end

SUBROUTINE MAGPOL(AINC, DDEC, NX, NY)
COMMON X(512,512)
COMPLEX X, CONA, CONB
C************************************************************************
C"MAGPOL" CREATES A WAVE\# REDUCTION-TO-MAGNETIC-POLE FILTER
ONTO THE ARRAY "X" IN BLANK COMMON
THE DIMENSIONS OF THE ARRAY "X" MUST BE IDENTICAL TO THAT IN
THE MAIN PROGRAM.
"AINC" THE AVERAGE MAGNETIC INCLINATION OF THE AREA IN DEGREES.
"DDEC" THE AVERAGE MAGNETIC DECLINATION OF THE AREA IN DEGREES.
************************************************************************
C
Rx=3.14159265/180.0
Nx2=NX/2+1
Nx1=NX/2
Nx=NX+2
Ny2=NY/2+1
Ny=NY+2
Rx1=1.0/FLOAT(NX)
Ny1=1.0/FLOAT(NY)
SINI=SIN(AINC*R8)
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)

program description
avgdifres is a bashing of matrix manipulation.  the program
will find the average of two input data sets, the difference
between them, and will resample (hence avg-dif-res ain't science
great?) every other point, every third, fourth etc., point.
If you want the average and difference of the input matrices
make sure you choose to resample every data point, ie choose
1 for ixl.
write (*,*) 'I FOR ONE DATA SET'
write (*,*) '2 FOR TWO DATA SETS'
read (*,*) set
write (*,*) 'INPUT FILE OF DAWN GRIDDED ANOMALIES OR ONE DATA SET'
read (*,*) 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')
write (*,*) 'OUTPUT AVERAGED AND/OR RESAMPLED GRIDDED FILE'
read (*,9990) filename
open (20, file-filename, form='formatted')
if (set .eq. i) go to 20
write (*,*) 'OUTPUT DIFFERENCE GRID OF (DUSK) - (DAWN)'
read (*,9990) filename
open (22, file-filename, form='formatted')
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
read (10,*) coln
read (10,*) rown
read (10,*) xcolat
read (10,*) xlong
read (10,*) xgridspc
40 do 50 i=1,rown
   do 50 j=1,coln
      read (10,*) (dndata(i,j),j=1,coln)
50 continue
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
40 continue
write (*,*) 'THE GRIDS ARE THE SAME SIZE NOW
read (10,*) row
read (10,*) col
read (10,*) xgridspc
do 80 i=1,rown
   do 80 j=1,coln
      read (12,*) (dndata(i,j),j=1,coln)
80 continue
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
avgrms-totrms/(real (rown)*real (coln))

find the average and difference matrices also
calculate the total and average RMS difference.
c
170 if (set .eq. 1) then
   do 180 l=1,rown
      do 180 j=1,coln
         avgdata(i,j)=dndata(i,j)
180 continue
   endif
170 continue
if (set .eq. 1) then
   do 180 l=1,rown
      do 180 j=1,coln
         avgdata(i,j)=dndata(i,j)-dndata(i,j)
180 continue
   endif

this is the section that resamples
row=1
1=1
200 jj=1
1=1
col=1
250 ddata(i,j)-avgdata(row,col)
col=col+var
if (col.gt. coln) go to 300
jj=jj+1
300 row=row+var
if (row .gt. rown) go to 400
ll=ll+1
go to 200
400 continue

write (*) 'new rows=', ll, ' new cols=', jj
write (20, *) jj
write (20, *) jcol
write (20, *) xlong
write (20, *) real (var)
do 450 k=1, ll
write (20, 9992) (ddata(k,l), l=1, jj)
450 continue

9992 format (6(f11.4,1x))
c
if (set .eq. i) go to 999
write (22, *) col
write (22, *) row
write (22, *) ycolat
write (22, *) ylong
write (22, *) ygridspc
do 500 m=1, row
write (22, 9992) (dlfdata(m,n), n=1, coln)
500 continue
write (*, '*') 'total rms difference=', totrms
write (*, '*') 'average rms difference=', avgrms
999 continue

continue
close (10)
close (12)
close (20)
close (22)
stop
de}
program sqmap
character*80 filename

dimension xcore(100000),xxmag(100000),
> core(361,361),xxmag(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
$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
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,*) lrow
read (10,*) south
read (10,*) west
read (10,*) gridspc

$c do 1=l,lrow
read (10,*) (core(l,J),J-l,icol)
do =j,l,icol
  cmean=cmean+core (i,j)
enddo
enddo

$c $read (Ii,*) imcol
read (II,*) imrow
read (II,*)
south
read (II,*) west
read (II,*)
grldspc

$c do i-l, imrow
read (ii,*) (xmag(i,J),J-l,imcol)
do J-l, imcol
  $mean=$mean+xxmag(i,j)
enddo
endif
if (irow.ne.imrow .or. icol.ne.imcol ) then
write (*,*) 'rows cor mag ',irow,
write (*,*) 'cols cor mag ',icol,lmcol
stop 10
endif
$c cmean=cmean/real(irow*icol)
$fmean=$mean/real(irow*icol)

$c------------------------------------------ remove the mean values
$c do 500 j-l, lrow
$c do 500 J-l, icol
xxmag(i,J)=xxmag(i,J)-$mean
core(i,j)=core(i,j)-cmean
enddo
enddo
$c------------------------------------------ turn the 2-D arrays into 1-D (cheater)
cdo i-l, imrow
  cdo j-l, imcol
    li=(i-1)*icol+j
    xxmag(li,xxmag(li,j))=core(li,icol)
    xcore(li)=core(li,j)
enddo
enddo
$c------------------------------------------ find a scalar value of c transpose c
ctc=0.0
$do 600 j-l,ii
  ctc=(xcore(j)*xcore(j))+ctc
$do 600 j-l,ii
ctcinv=1.0/ctc
$c------------------------------------------ find (c transpose c) inverse
$ctf=0.0
$do 700 j-l,ii
  ctf=(xcore(j)*xxmag(j))+ctf
$do 700 j-l,ii
x=ctcinv*ctf
write (*,*) 'the value of x = ', x
$c
write (20,*) imcol
write (20,*) imrow
write (20,*) south

C-42
write (20,*) west
write (20,*) gridspc
do i-l, irow
  do j-l,icol
    xmultmag(j) = xmag(i, j) - x(i, j) * (core(i, j))
  enddo
write (20,9991) (xmultmag(j), j-l,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, XORO, YORDO, NGX, NYO, RX, RXO, 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), II, DI, FI, MYEAR, MI, MD,
• MIL, MDL, MF1, MUXORO, MYORO, MDXO, MGYO, MELVO
REAL FLDI, FLDX, INC, INCX, DEC, DECX
DIMENSION FLDI (44, 181), FLDX (44, 361), INC (44, 181), INCX (44, 181),
• DEC (44, 181), DECX (44, 361)
COMMON /GMF/ FLDI, FLDX, INC, INCX, DEC, DECX
CHARACTER*80 FILENAME
INTEGER CHOICE
************************************************************************
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
C 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 (NXD), NGX, NYO, S (NIJ) WHERE NIJ = (NXO*NYO) + (NXD*NYD+I)/2, FLD (NXD),
C INC (NXD), D1 (NXD), FLD (NXO), I (NXO), D (NXO)
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 i0 or ii are necessary as all arrays are stored in
C memory.
C modifications made 15 may 90
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 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******** DATA INPUT SEQUENCE *************
C WRITE (*,*) 'O IF YOU HAVE A FILE FOR THE VARIABLES'
C WRITE (*,*) 'I IF YOU WANT TO TYPE VARIABLES INTERACTIVELY'
READ (*,*) CHOICE
C IF (CHOICE .EQ. 0) GO TO 10
C WRITE (*, 9991)
C 9991 FORMAT ('NFLD=O DO NOT CALCULATE EQUIVALENT SOURCE M-FIELD'/
C 1 'NFLD=1 CALCULATE EQUIVALENT SOURCE M-FIELD'/
C 2 'NFLD=2 GIVEN THE CGS-SUSCEPTIBILITIES THE PROGRAM WILL CALCULATE THE EQUV SRC M-FILD'/
C 1 'NIO=O DO NOT WRITE A FILE SUSCEPTIBILITIES'/
C 2 'NIO=1 WRITE OUT A SEPARATE FILE OF SUSCEPTIBILITIES'/
C 'NFLD(1) NIO(0)')
READ (*,*) NFLD, NIO
C IF (NFLD .EQ. 1) THEN
C WRITE (*,*) 'the following refers to calculation of the',
C WRITE (*,*) 'susceptibilities'
C ENDIF
WRITE (*, 9992)
9992 FORMAT ('NXO= NUMBER OF LONGITUDINAL COLS OF OBSERVATION GRID'/
C 'NYO= NUMBER OF LATITUDINAL ROWS OF OBSERVATION GRID'/
C 'XORO=WESTERN-MOST LONGITUDINAL COORDINATE OF OBSERVATION'/
C 'YORD=SOUTHERN-MOST LATITUDINAL COORDINATE OF OBSERVATION'/
C 'GRID IN -180.0 TO 180.0 DEGREES'/
C 'DXO=LONGITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/
C 'DYO=LATITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/
C 'ELVO=ELEVATION OF OBSERVATION GRID IN KILOMETERS (350.0)'/
C 'MYEAR MYORO 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'/
C 'NYD= NUMBER OF LATITUDINAL ROWS OF SOURCE GRID'/
C 'XORD=WESTERN-MOST LONGITUDINAL COORDINATE OF SOURCE'/
C 'YORD=SOUTHERN-MOST LATITUDINAL COORDINATE OF SOURCE'/
C 'GRID IN -180.0 TO 180.0 DEGREES'/
C 'DXD=LONGITUDINAL STATION SPACING OF SOURCE GRID IN DEGS'/
C 'DYD=LATITUDINAL STATION SPACING OF SOURCE GRID IN DEGS'/
C 'DYS=LATITUDINAL STATION SPACING OF SOURCE GRID IN DEGS'
`ELEVATION OF SOURCE GRID IN KILOMETERS (-50.0)`

- `*` read (*, *) na, nd, xord, yord, dxd, dyd, elvd

- `c`

- `write (*, 9994)`

```fortran
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, I, D) AND GEOMAGNETIC FIELD (I, D) OVER'/
OBSERVATION GRID'/
F1= SCALAR MAGNETIC INTENSITY IN GAMMAS OF SOURCE POLARIZATION'/
FIELD.'/
1= INCLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
D= DECLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
NOTE---IF (F1+I+D).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 (600.0) I (90.0) D (0.0)'
read (*, *) year, f1, i, d
```

- `write (*, *) 'ERROR FACTOR FOR VARIANCE (fak) (0.10E-7)'`

```fortran
read (*, *) fak
```

- `if (nfld .eq. 0)`

```fortran
go to 15
```

- `write (*, 9996)`

```fortran
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, I, D) AND GEOMAGNETIC FIELD (I, D) OVER'/
OBSERVATION GRID'/
F1= SCALAR MAGNETIC INTENSITY IN GAMMAS OF SOURCE POLARIZATION'/
FIELD.'/
1= INCLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
D= DECLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
NOTE---IF (mF1+m1+mD).EQ.0.0, THEN THE SOURCE POLARIZING'/
FIELD IS COMPUTED BY SUBROUTINE FIELDG FOR EPOCH'/
SPECIFIED BY THE YEAR-INPUT VARIABLE'/
M YEAR (0.0) mF1 (6000.0) mI (90.0) mD (0.0)'
read (*, *) myear, mF1, mI, mD
```

- `write (*, 9997)`

```fortran
9997 format ('mI= INCLINATION 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 (0.0) mD (0.0)'
read (*, *) mi, md
```

- `if (isub .eq. 0)`

```fortran
go to 15
```

- `write (*, 10)`

```fortran
10 write (*, *) 'INPUT FILE OF NUMBERS FOR VARIABLES'
read (*, 9990) filename
open (17, file=filename, status='old', form='formatted')
```

C -46
READ (17,*) NFLD, NIO
READ (17,*) NKO, NYO, XORO, YORO, DXO, DYO, ELVO
READ (17,*) NKO, NYO, XORO, YORO, DXO, DYO, ELVO
READ (17,*) YEAR, FI, II, DI, I, D
read (17,*) xk, yk, mnxo, mnyo, mxoro, myoro, mdxo, mdyo, melvo
read (17,*) myear, mfl, mfl, ml, ml, md
READ (17,*) csub

write (*,*) 'INPUT FILE OF GRIDDED ANOMALY DATA OR SUSC DATA'
write (*,*) 'DATA SHOULD BE WEST TO EAST AND SOUTH TO NORTH'
write (*,9990) filename
9990 format (a80)
open (I3, file=filename, status='old', form='formatted')
write (*,*) 'INPUT FILE OF SPHERICAL HARMONIC COEFFICIENTS SUCH AS GSFC1283'
read (*,9990) filename
open (I3, file=filename, status='old', form='formatted')
if (nfld .eq. I .or. nfld .eq. 2) then
write (*,*) 'OUTPUT FILE OF EQUIVALENT SOURCE M-FIELD'
read (*,9990) filename
open (I30, file=filename, form='formatted')
endif
if (nio .eq. 1) then
write (*,*) 'OUTPUT FILE OF SUSCEPTIBILITIES'
read (*,9990) filename
open (I33, file=filename, form='formatted')
endif
write (*,*) 'OUTPUT INFORMATION FILE OF A BUNCH OF STUFF!!'
read (*,9990) filename
open (I6, file=filename, form='formatted')
if (nfld .eq. 2) go to 155
WRITE (6,520) NKO, NYO, XORO, YORO, DXO, DYO, ELVO
WRITE (6,540) NXD, NYD, XORD, YORD, DXD, DYD, ELVD
read (13,*) few
read (13,*) ins
read (13,*) scolat
read (13,*) west
read (13,*) gridspc
DO 130 JY=I,NYO
READ (13,*) (DF(IX, JY),IX=I,NXO)
130 continue
C COMPUTE MAXIMUM, MINIMUM AND AVERAGE AMPLITUDE VALUES FOR M-FIELD INPUT DATA
DSUM=0.0
DMIN=DF(I,1)
DMAX=DMIN
DO 140 IX=I,NXO
DSUM=DSUM+DF(IX,JY)
IF (DMIN.LT.DF(IX,JY)) DMIN=DF(IX,JY)
IF (DMAX.GT.DF(IX,JY)) DMAX=DF(IX,JY)
140 continue
DSUM=DSUM/FLOAT(NKO*NYO)
WRITE (6,640) DMAX, DMIN, DSUM
save values for the reduction-to-pole

istrxd=nxd
istryd=nymd
strxord=xord
stryord=yord
strxd=dxd
stryd=dyd
strelvd=elvd
if (nfld .eq. 2) go to 840

CONVERT LAT AND LONG TO RADIANS AND ELEVATIONS TO EARTH RADII

PI=3.1415926536
FACT=PI/180.0

XORD=XORD*FACT
YORD=(90.0-YORD)*FACT
XORD=(90.0-YORD)*FACT
DXO=DXO*FACT
DYD=DYD*FACT
DYD=DYD*FACT

I=1*FACT
D=-D*FACT
I=I*FACT
D=D*FACT

REARTH=6371.0
RD=ELVD+REARTH
RD=ELVD
E=RD**2*RD**2
G=2.0*RD*RD
NP=(XORD+YORD)
DO 150 JY=1,NP
   P(YJ)=0.0
   DM(YJ)=0.0
   CONTINUE

NJ=NP*(NP+1)/2
DO 160 JY=1,NJ
   S(JY)=0.0
   CONTINUE

DO 170 IX=1,NXO
   PHI=XORD+(FLOAT(IX)-I.0)*DXO
   AD=DEC(IX)
   CALL MG5S1(AI,AD)
   DY=DF(IX,JY)-YD
   LLL=0
   DO 170 L=1,LLP
      DM(L)=DM(L)+TMAG(L)*DY
      LLL=LLL+1
   DO 170 K=1,LLL
      S(LLL)=S(LLL)+TMAG(L)*TMAG(K)
   CONTINUE
   GO TO 240

DO 190 IX=1,NXO
   PHI=XORD+(FLOAT(IX)-I.0)*DXO
   AI=INC(IX)
   AD=DEC(IX)
   CALL SMG5S1(AI,AD)
   DY=DF(IX,JY)-YD
   LLL=0
   DO 190 L=1,LLP
      DM(L)=DM(L)+TMAG(L)*DY
      LLL=LLL+1
   DO 190 K=1,LLL
      S(LLL)=S(LLL)+TMAG(L)*TMAG(K)
   CONTINUE
   GO TO 240

DO 190 JY=1,NJ
   S(JY)=0.0
   CONTINUE

C -48
COST = COS(THETA)
SINT = SIN(THETA)
DO 190 IX = 1, NXO
PHI = XORO + (FLOAT (IX) - 1.0) * DXO
C
CALL MAGS2 (I, D, FI, II, DI)
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
COMPUTE A (TRANSPOSE) A MATRIX FOR CASE WHERE (I, D) ARE DERIVED
FROM FIELDG AND USER SUPPLIES (FI, II, DI)
C
200 DO 210 JY = 1, NYO
THETA = XORO + (FLOAT (JY) - 1.0) * DYO
C
COST = COS (THETA)
SINT = SIN (THETA)
DO 185 L = 1, NXO
fld (L) = fld (L) * DYO
inc (L) = inc (L) * DYO
de (L) = de (L) * DYO
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, FI, II, DI)
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
COMPUTE A (TRANSPOSE) A MATRIX FOR CASE WHERE USER SUPPLIES (I, D)
AND (FI, II, DI) ARE DERIVED FROM FIELDG
C
220 DO 230 JY = 1, NYO
THETA = XORO + (FLOAT (JY) - 1.0) * DYO
C
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
C
C COMPUTE INVERSE OF S-ARRAY
C
C ------------------------------ If a transpose a without error
C variance is wanted then use
C write (20).
C
C WRITE (20) (S (J), J = 1, LLL)
C
C ------------------------------ If a transpose observations without
C error variance is wanted then use
C write (25).
C
C WRITE (25) (DW (L), L = 1, NP)
C
C ------------------------------ the following loop adds fak to the diagonal
C of ATa
C
II = 0
DO 888 J = 1, NP
II = II + J
S (II) = S (II) + FAK
888 CONTINUE
C
C COMPUTE INVERSE OF S-ARRAY
CALL SPPCO (S, NP, RCOND, DP, INFO)

WRITE (6, 480) RCOND
IF (INFO .NE. 0) WRITE (6, 490) INFO
DO 250 IX = 1, NP

250 DP(IX) = DM(IX)

CALL SPPSL (S, NP, DP)

WRITE OUT COORDINATE CHARACTERISTICS OF M-DIPOLES

WRITE (6, 590)
L = 0
DO 260 JY = 1, NYD

260 CONTINUE
DO 270 JY = 1, NP

270 P(JY) = DP(JY)

C IF N10 = 1 WRITE SUSCEPTIBILITIES ONTO USER DEFINED UNIT 33

C IF (N10 .EQ. 1) then
WRITE (33, *) iew
WRITE (33, *) ins
WRITE (33, *) ncolat
WRITE (33, *) west
WRITE (33, *) gridspc
DO 375 JY = 1, NYD

375 CONTINUE
END IF
C IF (NFLD .EQ. 0) GO TO 410
WRITE (6, 620)
C COMPUTE EQUIVALENT SOURCE M-FIELD

840 continue
WRITE (6, 630)

END IF
C IF (mYEAR .LT. 1.E-9) GO TO 880
CALL FIELDSG (0., 0., 0., 55.1, q1, q2, q3, q4)
IF (mfi .GT. .9D1) GO TO 870
CALL GEOMAG (myear, elvd, yord, xord, dyd, dxd, nyd, nxd, 11)

870 IF (mfl .GT. 0.0) GO TO 880
CALL GEOMAG (myear, elvd, yord, xord, dyd, dxd, nyd, nxd, 10)

880 continue
C PI = 3.1415926536
FACT = PI / 180.0
C
nxordo = nxordo * FACT
nxord = nxord * FACT
nyordo = (90.0 - nyordo) * FACT
yord = (90.0 - yord) * FACT
mdyo = dyo * FACT
dxo = dxo * FACT
mdyo = dyo * FACT
dyo = dy * FACT

C -50
C

m1=m1*FACT
mD=mD*FACT
m1l=m1l*FACT
mD1=mD1*FACT

C REARTH=6371.0
RD=ELVD+REARTH
E=RD+2*R0*R0**2
G=2.0*RD

C IF (mYEAR.LT.1.E-9) GO TO 300
IF (mF1+m1l+mD1.GT.0.0 .AND. m1+mD.LT.1.E-9) GO TO 330
IF (mF1+m11+mD1.LT.1.E-9 .AND. m1+mD.GT.0.0) GO TO 360

C COMPUTE M-FIELD FOR CASE WHERE (m1,mD) AND (mF1,m11,mD1)
C ARE DERIVED FROM FIELDG

DO 290 JY=1,mNYO
   THETA=mYORO+(FLOAT(JY)-1.0)*mDYO
   COST=COS(THETA)
   SINT=SIN(THETA)
   DO 195 IX=1,mNXO
      FLD(IX)=FIDO(JY,IX)
      INC(IX)=INCO(JY,IX)
      DEC(IX)=DECO(JY,IX)
   195 CONTINUE
   CALL MAGSGL(a1,aD)
   280 DFS(IX,JY)=YC
   290 WRITE(6,580) (DFS(IX,JY),IX=1,mNXO)
   GO TO 390

C COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (m1,mD)
C AND (mF1,m11,mD1)

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
      AI=INC(IX)
      AO=DEC(IX)
   CALL_4KGS2(m1,mD,mF1,m11,mD1)
   310 DFS(IX,JY)=YC
   320 WRITE (6,580) (DFS(IX,JY),IX=1,mNXO)
   GO TO 390

C COMPUTE M-FIELD FOR CASE WHERE (m1,mD) ARE DERIVED FROM
C FIELDG AND USER SUPPLIES (mF1,m11,mD1)

DO 350 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
      AI=INC(IX)
      AO=DEC(IX)
   CALL_4KGS2(a1,aD,mF1,m11,mD1)
   370 DFS(IX,JY)=YC
   350 WRITE (6,580) (DFS(IX,JY),IX=1,mNXO)
   GO TO 390

C COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (m1,mD) AND
C (mF1,m11,mD1) ARE DERIVED FROM FIELDG

DO 380 JY=1,mNYO
   THETA=mYORO+(FLOAT(JY)-1.0)*mDYO
   COST=COS(THETA)
   SINT=SIN(THETA)
   DO 390 IX=1,mNXO
      PHI=mXORO+(FLOAT(IX)-1.0)*mDXO
   CALL_4KGS2(a1,aD,mF1,m11,mD1)
   390 DFS(IX,JY)=YC
   380 WRITE (6,580) (DFS(IX,JY),IX=1,mNXO)
   GO TO 390
CALL F_GSI (mI, mD)
370 DFS(IX, JY)=YC
380 WRITE(6,580) (DFS(IX, JY), IX=1,mNXO)
390 CONTINUE

C COMPUTE MAXIMUM, MINIMUM AND AVERAGE AMPLITUDE VALUES FOR EQUIVALENT SOURCE M-FIELD
C
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) DM_X,DMIN,DSt_4
C
if (isub .eq. 0) ds_m=0.0
dmax-dfs (i, i )
dmin-dmax
write (30,*) mnxo
write (30,*)(mnyo/fact)
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
DMAX-MAX (DMAX, (DFS (IX, JY)-dsum) )
DMIN-MIN (DMIN, (DFS (IX, JY)-dsum) )
385 CONTINUE
if (isub .eq. I) 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(' ST_ OF SQUARES - ',E15.10)
901 CONTINUE
STOP
C
420 FORMAT (/,,/X,,3HFI-, FI0.3, 5X, 3HI1-, F6.2, 5X, 3HDI-, F6.2, 5X, 2HI-, F6.2, 15X, 2HD-, F6.2,/) 430 FORMAT (/,,/X,,5HYEAR-, FI0.5, 5X, 2HI-, F6.2, 5X, 3HDI-, F6.2,/) 440 FORMAT (/,,/X,,5HYEAR-, FI0.5, 5X, 3HI1-, FI0.3, 5X, 3HDI-, FI0.3,/) 450 FORMAT (//,2X,25HEQUIVALENT SOURCE M-FIELD,/) 460 FORMAT (//, 'CGS-SUSCEPTIBILITIES ', 5X, 'E-LONGITUDE ', 5X, 'N-LATITUDE ', 5X, 'KM-LEVEL ',/) 470 FORMAT (IX, E20.5,5X, FIO.4,5X, FIO.4,5X,/) 480 FORMAT (///) 490 FORMAT (///)
C END SUBROUTINE MAGSI (I,D)
C************************************************************************
C************************************************************************
C SUBROUTINE MAGSI CALCULES 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(I) AT SOURCE GRID COORDINATES
C (RI,THETAI,PHII). POLARIZING FIELD CHARACTERISTICS (FI,II,DI)
C ARE READ FROM TAPEII.
************************************************************************
C
REAL I, Ii, JR, JTHETA, JPHI,
INCI (181),FLDI (181),DEC1 (181)
YC=0.0
L=0
DO 110 JY=1,NXD

Theta=THETA+FLOAT(JY-1)*DYD
COSTI=COS(Theta)
SINTI=SIN(Theta)
CTIT=COSTI*COST
STIT=SINTI*SINT
CSIT=COSTI*SINT

C DO 730 K=1,NXD
FDD(K)=FDD(JY,K)
INC(K)=INC(JY,K)
DEC(K)=DEC(JY,K)
CONTINUE

730 DO 110 IX=1,NXD
L=1

C COMPUTE POINT DIPOLE POLARIZATION VECTOR IN SPHERICAL ORTHONORMAL
C UNIT VECTORS (ER,ETHETA,EPHI) WITH SUCEP(L)=1.0

F1=FDD(IX)
I1=INC(IX)
D1=DEC(IX)

JR=F1*SIN(I1)
JTHETA=F1*COS(I1)*COST
JPHI=F1*COS(I1)*SINT

C COMPUTE MAGNETIC FIELD VECTOR (U,V,W) AT THE POINT (R,THETA,PHI)
C DUE TO POINT DIPOLES AT (RI,THETAI,PHI1)

PHI1=PHI1+FLOAT(IX-1)*DXD
SINP=SIN(PHI-PHI1)
COSP=COS(PHI-PHI1)
A=CTIT*STIT*COSP
B=CTIT*STIT*COSP
C=CINT*SINP
R2=E+G*A
K1=R2**1.5
K2=R2**2.5
XX=JR*(RO-RD*A)+JTHETA*B-JPHI*C
ATHETA=CSIT*STIT*COSP
BTHETA=STIT-CTIT*COSP
CTHETA=COSTI*SINT
BPHI=COSTI*SINT

C CALCULATE THE COMPONENT OF THE ANOMALOUS FIELD IN THE DIRECTION
C OF THE GEOMAGNETIC FIELD AT THE OBSERVATION POINT---I.E.,
C ((U,V,W)* (UR,UTHETA,UPHI) ) * P(L) = TOTAL MAGNETIC FIELD

UR=SIN(I)
UTHETA=COS(I)*COST
UPHI=COS(I)*SINT

TMAG(L)=JR*UR*JTHETA*B*JPHI*C
VC=YC=P(L)*TMAG(L)

110 CONTINUE
RETURN
END

SUBROUTINE MAGS2 (I,D,FI,II,DI)
************************************************************************
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 -53
COMMON XORD, YORD, DWD, DWD, MYO, RO, RD, E, G, COST, SINT, YC, THETA, PHI,
> (3982), TMXG (3982), DF (361, 44), S (7930153)
REAL 1, 11, JR, JTHETA, JPHI
YC=0, 0
L=0
DO 110 JY=1, MYO
THETA=PI-YORD*FLOAT(JY-1) *DYD
COST1-COST(THETAI)
SINT1-SINT(THETAI)
CTIT-COSTI*COST
STIT-SINTI*SINT
CTST-SINTI*COST
CSIT-COSTI*SINT
110 CONTINUE

RETURN
END

SUBROUTINE GEOMAG (YEAR, ELV, THETA, PHI, HTMTHETA, HPHI, NTHETA, NPHI, NFC
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 /gmn/ fldd, fldo, inco, deco, inco, deco
C
***********************************************************************

THIS SUBROUTINE CALCULATES THE MAGNITUDE, INCLINATION, AND
DECLINATION OF THE GEOMAGNETIC FIELD ON A GRID MTHETA BY

NPHI
***********************************************************************
REAL F (361), INC (361), DEC (361)

write (6, *), ' ' 
if (nf .eq. 10) then
  write (6, *) 'the following is the geomagnetic field'
elseif (nf .eq. 11) then
  write (6, *) 'the following is the geomagnetic field'
endif
write (6, *) 'over the grid'
endlf
write (6, *) ' ' 
c LL-O YYYY-TETA
XXXYX-55-5

DO 120 I-1, NTHETA
PHI-00000

DO 110 J-1, NPHI

CALL FIELDG (THETA, PHI, ELV, YEAR, 50, LL, X, Y, Z, F (J))

H-SQRT (X**2+Y**2)
INC (J) =ATAN2 (Z, H)
DEC (J) =ATAN2 (Y, X)
PHI-PHI+HPHI

ii0 CONTINUE

if (nf .eq. ii) then
  do 710 l-1, nphi
    fldd (i, l)-f (i)
    incd (i, 1) -Inc (i)
    deco (i, 1) -dec (i)
  710 continue
else if (nf .eq. I0) then
  do 720 l-1, nphi
    fido (i, l)-f (i)
    incod (i, 1) -Inc (i)
    deco (i, 1) -dec (i)
  720 continue
endif
THETA-THETA+ HTHETA

120 CONTINUE

C ****************************************
C END SUBROUTINE FIELDG (DLAT, DLONG, ALT, TM, NGRID, L, X, Y, Z, F)
C
C SUBROUTINE FIELDG (NASA)
C SUBROUTINE FIELDG (NASA)
C SUBROUTINE FIELDG (SYSTEM)
C SUBROUTINES USED
C***************************************************************
FOR DOCUMENTATION OF THIS SUBROUTINE AND SUBROUTINE FIELD SEE:
NATIONAL SPACE SCIENCE DATA CENTER'S PUBLICATION
**COMPUTATION OF THE MAIN GEOMAGNETIC FIELD
FROM SPHERICAL HARMONIC EXPANSIONS**
DATA USERS' NOTE, NSSDC 68-11, MAY 1968.
GOODARD SPACE FLIGHT CENTER, GREENBELT, MD.
***************************************************************
**LONGITUDE IN DEGREES POSITIVE EAST**

**ELEVATION IN KM (POSITIVE ABOVE, NEGATIVE BELOW EARTH'S SURFACE)**

**EPOCH IN YEARS**

**SET TO INTEGER GREATER THAN DEGREE OF EXPANSION**

**SET TO 1 ON INITIAL DUMMY CALL, SET TO 0 ON SUBSEQUENT CALLS**

**SUBROUTINE RETURNS GEOMAGNETIC FIELD DIRECTIONS (X,Y,Z), POSITIVE NORTH, EAST AND DOWN, RESPECTIVELY, AND MAGNITUDE OF TOTAL FIELD, F—ALL VALUES ARE IN GAMMAS**

**EQUIVALENCE (SPLIT(I,I),TG(I,I))**

**COMMON /NASA/ TS(55,55)**

**COMMON /FLDCOM/ CPH,SPH,R,CT,ST,RT,BP,BR,B**

**COMMON /MAX/ MAX**

**DIMENSION G(55,55), GT(55,55), SPLIT(55,55), AID(55), GTT(55,55)**

**DATA A/0.0.6378.16**

**IF (A.EQ.6378.16) IF (L) 210,100,110**

**A-6378.16**

**FLAT-1./298.25**

**A2-A**

**A4-A**

**B2- (A*FLAT)**

**A2B2-A2* (I-.FLAT**

**A4B4-A4* (I-.FLAT**

**IF (L) 160,160,110**

**100 IF (TM-TLAST) 190,210,190**

**110 READ (3,260) J,K,TZERO, (AID(I),I-I,II)**

**L-0**

**WRITE(6,270) J,K,TZERO, (AID(I),I-I,II)**

**MAXN=0**

**TEMP=0.**

**120 READ (3,280) N,M,GNM,HNM,GTNM,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=MAX1 ( TEMP, ABS (GTNM) )**

**IF (M.EQ.1) GO TO 120**

**G (M-1,N)=HNM**

**GT (M-1,N)=HTTNM**

**GO TO 120**

**130 continue**

**REWIND (3)**

**130 WRITE (6,290)**

**DO 150 N-2,MAXN**

**150 M=1,N**

**150 M=1,N**

**DO 150 N=2,MAXN**

**150 M=1,N**

**DO 150 M=1,N**

**DO 170 M=1,N**

**170 J=1**

**DO 180 M=1,N**

**180 M=1,N**

**DO 200 M=1,N**

**200 M=1,N**

**DO 200 M=1,N**

**DO 200 M=1,N**

**DO 200 M=1,N**

**DO 300 M=1,N**

**WRITE (6,310) N,M,G(N,M),G(T(N,M),G(MT(N,M),GTT(N,M))**

**C-56**
200 CONTINUE
TLAST=TH
210 DLAT=DLAT/57.2957795
SINLA=SIN(DLAT)
RLONG=LONG/57.2957795
COS= COS (RLONG)
COS= COS (RLONG)
IF ((J.EQ.0) GO TO 220
R=ALT+6371.0
CT=SINLA
GO TO 230
220 SINLA2=SINLA**2
COSL2=2-2*SINLA2
DEN=A2-A2B2*SINLA2
DEN=SQR(T(DEN))
FAC=(((ALT*DEN)+A2)/((ALT*DEN)+B2))**2
CT=SINLA/SQRT(FAC*COSLA2+SINLA2)
R=SQR ((ALT+2.*DEN)+(A4-A4B4*SINLA2)/DEN)
230 ST=SQR (1.-CT**2)
NMAX=MIND (NMAX,MAX)
C 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-SQR (COSLA2)*CT
COSD=SQR (1.-SIND**2)
X=BT*COSD-BR*SIND
Z=BT*SIND-BR*COSD
RETURN
C 260 FORMAT (211,1X,F6.1,10A6,A3)
270 FORMAT (213,5X,6DEPOCH, ,F7.1,5X,10A6,A3)
280 FORMAT (213,6F11.4)
290 FORMAT (6HG N M,6X,1HG,10X,1HG,9X,2HHT,9X,2HHTT,8X,3HHTT)
300 FORMAT (213,6F11.4)
310 FORMAT (213,F11.4,11X,F11.4,11X,F11.4)
320 FORMAT (///)
C C SUBROUTINE FIELD
C COMMON /NASA/ G(55,55)
COMMON /FLDCAM/ CPH, SPH, R, CT, ST, BT, BP, BR, B
COMMON /MAX/ 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)=0
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)
AGR=6371.0/R
AR=AG**2
BT=0.
BP=0.
BR=0.
DO 190 N=2,NMAX
AR=AOMAR
DO 190 M=1,N
190 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
C C 57
TO COMPUTE C(AMPUTE ON ENTRY
TEMP-G (N, M) *CP (M) +G (M-1, N) *SP (M) *FM (M) *PAR
GO TO 180
TEMP-G (N, M) *CP (M) +G (M-1, N) *SP (M) *FM (M) *PAR
BP = BP-ST
B = SORT (ST * BT + BP * BP + BR * BR)
RETURN
C END

SUBROUTINE SPPCO (AP, N, RCOND, Z, INFO)
INTEGER N, INFO
REAL AP (1), Z (N)
REAL RCOND
SPPCO FACTORS A REAL SYMMETRIC POSITIVE DEFINITE MATRIX
STORED IN PACKED FORM
AND ESTIMATES THE CONDITION OF THE MATRIX.
IF RCOND IS NOT NEEDED, SPPFA IS SLIGHTLY FASTER.
TO SOLVE A*X = B , FOLLOW SPPCO BY SPPSL.
TO COMPUTE INVERSE(A)*C , FOLLOW SPPCO BY SPPSL.
TO COMPUTE DETERMINANT(A) , FOLLOW SPPCO BY SPPDI.
TO COMPUTE INVERSE(A) , FOLLOW SPPCO BY SPPDI.
ON ENTRY
AP REAL (N*(N+1)/2)
THE PACKED FORM OF A SYMMETRIC MATRIX A . THE
COLUMNS OF THE UPPER TRIANGLE ARE STORED SEQUENTIALLY
IN A ONE-DIMENSIONAL ARRAY OF LENGTH N*(N+1)/2 .
SEE COMMENTS BELOW FOR DETAILS.
N INTEGER
THE ORDER OF THE MATRIX A .
ON RETURN
AP AN UPPER TRIANGULAR MATRIX R , STORED IN PACKED
FORM, SO THAT A = TRANS(R)*R . IF INFO .NE. 0 , THE FACTORIZATION IS NOT COMPLETE.
RCOND REAL
AN ESTIMATE OF THE RECIPROCAL CONDITION OF A
FOR THE SYSTEM A*X = B , RELATIVE PERTURBATIONS
IN A AND B OF SIZE EPSILON MAY CAUSE
RELATIVE PERTURBATIONS IN X OF SIZE EPSILON/RCOND .
IF RCOND IS SO SMALL THAT THE LOGICAL EXPRESSION
1.0 + RCOND .EQ. 1.0
IS TRUE, THEN A MAY BE SINGULAR TO WORKING
PRECISION. IN PARTICULAR, RCOND IS ZERO IF
EXACT SINGULARITY IS DETECTED OR THE ESTIMATE
UNDERFLOWS. IF INFO .NE. 0 , RCOND IS UNCHANGED.
Z REAL (N)
A WORK VECTOR WHOSE CONTENTS ARE USUALLY UNIMPORTANT.
IF A IS SINGULAR TO WORKING PRECISION, THEN Z IS
AN APPROXIMATE NULL VECTOR IN THE SENSE THAT
NORM(A*Z) = RCOND*NORM(A)*NORM(Z) . IF INFO .NE. 0 , Z IS UNCHANGED.
INFO INTEGER
= 0 FOR NORMAL RETURN.
= K SIGNALS AN ERROR CONDITION, THE LEADING MINOR
OF ORDER K IS NOT POSITIVE DEFINITE.
PACKED STORAGE
THE FOLLOWING PROGRAM SEGMENT WILL PACK THE UPPER
TRIANGLE OF A SYMMETRIC MATRIX.
K = 0
DO 20 J = 1, N
DO 10 I = 1, J
K = K + 1
AP (K) = A(I,J)
10 CONTINUE
20 CONTINUE

LINPACK. THIS VERSION DATED 08/14/78.
CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.

SUBROUTINES AND FUNCTIONS
LINPACK SPPFA
BLAS SAXPY, SSDOT, SSYAL, SASUM
FORTRAN ABS, AMAX1, REAL, SIGN

INTERNAL VARIABLES

C-58
REAL SDOT, EK, T, WK, WKM
REAL ANORM, S, SASUM, SM, YNO_M
INTEGER I, J, JMI, J, K, KB, KJ, KK, KPI

C

C

C

FIND NORM OF A

C

J = 1
DO 30 J = 1, N
   Z(J) = SASUM(J, AP(JI), IJ - J1)
   J1 = J1 + J
   JMI = J - 1
   IF (JMI .LT. I) GO TO 20
   DO 10 I = IJ - J1, JMI + 1
      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 = MAX(ANORM, Z(J))
40 CONTINUE

C

C

FACTOR

CALL SPPFA(AP,N,INFO)
IF (INFO .NE. 0) GO TO 180

RCOND = 1/(NORM(A)*ESTIMATE OF NORM(INVERSE(A)))
ESTIMATE = NORM(Z)/NORM(Y) WHERE A*Z = Y AND A*Y = E
THE COMPONENTS OF E ARE CHOSEN TO CAUSE MAXIMUM LOCAL
GROWTH IN THE ELEMENTS OF W WHERE TRANS(R)*W = E
THE VECTORS ARE FREQUENTLY RESCALED TO AVOID OVERFLOW.

C

C

SOLVE TRANS(R)*W = E

EK = 1.0E0
DO 50 J = 1, N
   Z(J) = 0.0E0
50 CONTINUE

KK = 0
DO 100 K = 1, N
   KK = KK + K
   IF (Z(K) .NE. 0.0E0) EK = SIGN(EK,-Z(K))
   IF (ABS(EK-Z(K)) .LE. AP(_)) 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)
KPI = K + 1
KJ = KK + K
IF (KPI.GT. N) GO TO 100
   DO 70 J = KPI, 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
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

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)
   ELSE
      DO 120 J = 1, N
         Z(J) = Z(J) + T*AP(KJ)
      KJ = KJ + J
120 CONTINUE
SUBROUTINE SPPFA(AF, N, INFO)
  REAL AF(1)
  INTEGER N, INFO

  SPPFA FACTORS A REAL SYMMETRIC POSITIVE DEFINITE MATRIX
  STORED IN PACKED FORM.

  SPPFA IS USUALLY CALLED BY SPPCO, BUT IT CAN BE CALLED
  DIRECTLY WITH A SAVING IN TIME IF RCOND IS NOT NEEDED.
  (TIME FOR SPPCO) = (1 + 18/N)*(TIME FOR SPPFA).

  ON ENTRY
  AP    REAL (N*(N+1)/2)
  THE PACKED FORM OF A SYMMETRIC MATRIX A. THE
  COLUMNS OF THE UPPER TRIANGLE ARE STORED SEQUENTIALLY
  IN A ONE-DIMENSIONAL ARRAY OF LENGTH N*(N+1)/2.
  SEE COMMENTS BELOW FOR DETAILS.
  N    INTEGER
  THE ORDER OF THE MATRIX A.

  ON RETURN
  AP    AN UPPER TRIANGULAR MATRIX R, STORED IN PACKED
  FORM, SO THAT A = TRANS(R)*R.
  INFO    INTEGER
  = 0 FOR NORMAL RETURN.
  = K IF THE LEADING MINOR OF ORDER K IS NOT
  POSITIVE DEFINITE.

  PACKED STORAGE
  THE FOLLOWING PROGRAM SEGMENT WILL PACK THE UPPER
  TRIANGLE OF A SYMMETRIC MATRIX.
  K = 0
  DO 20 J = 1, N

  Z(K) = Z(K)/AP(KK)
  KK = KK - K
  T = -Z(K)
  CALL SAXPY(K-1,T,AP(KK+1),1,Z(1),1)
  CONTINUE

  YNORM = 1.0E0

  CALL TRANS(R)*V = Y
  DO 150 K = 1, N
    Z(K) = Z(K) - DDOT(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 SCAL(N,S,Z,1)
    YNORM = S*YNORM
  CONTINUE
  Z(K) = Z(K)/AP(KK)
  CONTINUE
  S = 1.0E0/SASUM(N,Z,1)
  CALL SCAL(N,S,Z,1)
  YNORM = S*YNORM

  CALL R*Z = V
  DO 170 K = 1, N
    K = N + 1 - KB
    IF (ABS(Z(K)) .LE. AP(KK)) GO TO 160
    Z(K) = Z(K)/AP(KK)
    KK = KK - K
    T = -Z(K)
    CALL SAXPY(K-1,T,AP(KK+1),1,Z(1),1)
  CONTINUE
  MAKE YNORM = 1.0
  S = 1.0E0/SASUM(N,Z,1)
  CALL SCAL(N,S,Z,1)
  YNORM = S*YNORM
  IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
  IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0

  CONTINUE
  RETURN
END

SPPFA FACTORS A REAL SYMMETRIC POSITIVE DEFINITE MATRIX
STORED IN PACKED FORM.

SPPFA IS USUALLY CALLED BY SPPCO, BUT IT CAN BE CALLED
DIRECTLY WITH A SAVING IN TIME IF RCOND IS NOT NEEDED.
(TIME FOR SPPCO) = (1 + 18/N)*(TIME FOR SPPFA).

ON ENTRY

AP      REAL (N*(N+1)/2)
  THE PACKED FORM OF A SYMMETRIC MATRIX A. THE
  COLUMNS OF THE UPPER TRIANGLE ARE STORED SEQUENTIALLY
  IN A ONE-DIMENSIONAL ARRAY OF LENGTH N*(N+1)/2.
  SEE COMMENTS BELOW FOR DETAILS.

N      INTEGER
  THE ORDER OF THE MATRIX A.

ON RETURN

AP      AN UPPER TRIANGULAR MATRIX R, STORED IN PACKED
  FORM, SO THAT A = TRANS(R)*R.

INFO      INTEGER
  = 0 FOR NORMAL RETURN.
  = K IF THE LEADING MINOR OF ORDER K IS NOT
  POSITIVE DEFINITE.

PACKED STORAGE
  THE FOLLOWING PROGRAM SEGMENT WILL PACK THE UPPER
  TRIANGLE OF A SYMMETRIC MATRIX.

  K = 0
  DO 20 J = 1, N
DO 10 I = 1, J
    K = K + 1
    AP(K) = A(I,J)
  10 CONTINUE
20 CONTINUE

LINPACK. THIS VERSION DATED 08/14/78.
CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.

SUBROUTINES AND FUNCTIONS

BLAS SDOT

FORTRAN SORT

INTERNAL VARIABLES

REAL SDOT, T
REAL S
INTEGER J, JJ, JMI, K, KJ, KK

BEGIN BLOCK WITH ...EXITS TO 40

JJ = 0
DO 30 J = I, N
   INFO = J
   S = 0.000
   JMI = J - 1
   KJ = JJ
   KK = 0
   IF (JMI .LT. I) GO TO 20
   DO 10 K = I, JMI
      KJ = KJ + 1
      T = AP(KJ) - SDOT(K-I,AP(KK+I),I,AP(JJ+I),I)
      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. 0.000) GO TO 40
   AP(JJ) = SQRT(S)
30 CONTINUE
40 CONTINUE
RETURN
END

SUBROUTINE SAXPY (N, SA, SX, INCX, SY, INCY)

C
C - CDC/SINGLE
C
C LATEST REVISION - JANUARY 1, 1978
C
C PURPOSE - COMPUTE A CONSTANT TIMES A VECTOR PLUS
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)
SA - REAL SCALAR. (INPUT)
SX - REAL VECTOR OF LENGTH MAX(N*IABS(INCX),I). (INPUT)
SX - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1). (INPUT)
INCY - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
C X(I) IS DEFINED TO BE...
C SX(I+(I-I)*INCX) IF INCX.GE.0 OR
C SX(I+(I-N)*INCY) IF INCY.LT.0.
SY - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1). (INPUT/OUTPUT)
C SAXPY REPLACES Y(I) WITH SA*X(I)+Y(I) FOR I=1,...,N.
C SAXPY REPLACES Y(I) WITH SA*X(I)+Y(I) FOR I=1,...,N.
C FOR I=1,...,N.
C X(I) AND Y(I) REFER TO SPECIFIC ELEMENTS
C OF SX AND SY, RESPECTIVELY. SEE IN CX AND
C INCY ARGUMENT DESCRIPTIONS.
C
C ARGS N - LENGTH OF VECTORS X AND Y. (INPUT)
C SX - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1). (INPUT)
C INCY - DISPLACEMENT BETWEEN ELEMENTS OF SY. (INPUT)
C Y(I) IS DEFINED TO BE...
C SY(I+(I-I)*INCY) IF IN CX AND
C SY(I+(I-N)*INCY) IF INCY.LT.0.
C PRECISION/HARDWARE - SINGLE/ALL
C REQD. IMSL ROUTINES - NONE REQUIRED
C
C -------------------------------

C -61
FUNCTION SDOT (N, SX, INCX, SY, INCY)

N - LENGTH OF VECTORS X AND Y. (INPUT)
SX - REAL VECTOR OF LENGTH MAX(N*ABS(INCX),1). (INPUT)
INCX - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
SY - REAL VECTOR OF LENGTH MAX(N*ABS(INCY),1). (INPUT)
INCY - DISPLACEMENT BETWEEN ELEMENTS OF SY. (INPUT)

USAGE - COMPUTE SINGLE PRECISION DOT PRODUCT
        - FUNCTION SDOT (N, SX, INCX, SY, INCY)

ARGUMENTS SDOT - SUM FROM I=1 TO N OF X(I)*Y(I). (OUTPUT)
X(I) AND Y(I) REFER TO SPECIFIC ELEMENTS OF SX AND SY, RESPECTIVELY. SEE INCX AND INCY ARGUMENT DESCRIPTIONS.

PRECISION/HARDWARE - SINGLE/ALL
REQD. IMSL ROUTINES - NONE REQUIRED
NOTATION - INFORMATION ON SPECIAL NOTATION AND
CONVENTIONS IS AVAILABLE IN THE MANUAL

COPYRIGHT - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.

WARRANTY - IMSL WARRANTS ONLY THAT IMSL TESTING
HAS BEEN APPLIED TO THIS CODE. NO OTHER WARRANTY,
EXPRESSED OR IMPLIED, IS APPLICABLE.

---

**Specifications for Arguments**

- **INTEGER** N, INCX, INCY
- **REAL** SX(I), SY(I)

**Specifications for Local Variables**

- **INTEGER** I, M, MPI, NS, IX, IY

**First Executable Statement**

SDOT = 0.0E0
IF (N.LE.0) RETURN
IF (INCX.EQ.INCY) IF (INCX-I) 5,15,35
5 CONTINUE

**Code for unequal increments or nonpositive increments.**

IX = 1
IY = 1
IF (INCX.LE.0) IX = (-N+1)*INCX+1
IF (INCY.LE.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

**Code for both increments equal to 1**

CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.

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.NE.1) RETURN
MPI = M+I
DO 30 I=MPI,N,5
SDOT = SDOT+SX(I)*SY(I-I)+SX(I+1)*SY(I+1)+SX(I+2)*SY(I+2)+SX(I+3)*SY(I+3)+SX(I+4)*SY(I+4)
30 CONTINUE
RETURN

**Code for positive equal increments**

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)**

COMPUTER - CDC/SINGLE

LATEST REVISION - JANUARY 1, 1978

PURPOSE - COMPUTE A SINGLE PRECISION CONSTANT TIMES A SINGLE PRECISION VECTOR

USAGE - CALL SSCAL (N, SA, SX, INCX)

**Arguments**

- **N** - LENGTH OF VECTOR X. (INPUT)
- **SA** - REAL SCALAR. (INPUT)
- **SX** - REAL VECTOR OF LENGTH N*INCX. (INPUT/OUTPUT)
  - SSCAL REPLACES X(I) WITH SA*X(I) FOR I=1,...,N.
  - X(I) REFERS TO A SPECIFIC ELEMENT OF SX.
  - SEE INX ARGUMENT DESCRIPTION.
- **INX** - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
  - X(I) IS DEFINED TO BE SX(I+I-I)*INCX).
  - INCX MUST BE GREATER THAN ZERO.

PRECISION/HARDWARE - SINGLE/ALL

REQD. IMSL ROUTINES - NONE REQUIRED

**Notation** - INFORMATION ON SPECIAL NOTATION AND
CONVENTIONS IS AVAILABLE IN THE MANUAL
INTRODUCTION OR THROUGH IMSL ROUTINE UHELP

C-63
C-COPYRIGHT - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C-WARRANTY - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
APPLIED TO THIS CODE. NO OTHER WARRANTY, EXPRESSED OR IMPLIED, IS APPLICABLE.

INTEGER INCX, N
REAL SA, SX(I)

INTEGER I, M, MPI, NS

FIRST EXECUTABLE STATEMENT

IF (N.LE.0) RETURN
IF (INCX.EQ.1) GO TO 10

NS = N*INCX
DO 5 I=1, NS, INCX
SX(I) = SA*SX(I)
5 CONTINUE
RETURN

CODE FOR INCREMENTS NOT EQUAL TO 1.

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 MPI = M+I
DO 25 I=MPI, 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)

COMPUTER - CDC/SINGLE
LATEST REVISION - JANUARY 1, 1978
PURPOSE - COMPUTE SINGLE PRECISION SUM OF ABSOLUTE VALUES
USAGE - FUNCTION SASUM (N, SX, INCX)
ARGUMENTS SASUM - SUM FROM I=1 TO N OF ABS(X(I)). (OUTPUT)
X(I) REFERS TO A SPECIFIC ELEMENT OF SX.
SEE INX ARGUMENT DESCRIPTION.
N - LENGTH OF VECTOR X. (INPUT)
SX - REAL VECTOR OF LENGTH N*INX. (INPUT)
INX - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
X(I) IS DEFINED TO BE SX(I+(I-1)*INX).
INX MUST BE GREATER THAN ZERO.

PRECISION/HARDWARE - SINGLE/ALL
REPD. IMSL ROUTINES - NONE REQUIRED
NOTATION - INFORMATION ON SPECIAL NOTATION AND
CONVENTIONS IS AVAILABLE IN THE MANUAL
INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
COPYRIGHT - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
WARRANTY - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
APPLIED TO THIS CODE. NO OTHER WARRANTY, EXPRESSED OR IMPLIED, IS APPLICABLE.

INTEGER N, INCX
REAL SX(I)

INTEGER I, M, MPI, NS

FIRST EXECUTABLE STATEMENT

SASUM = 0.0
IF (N.LE.0) RETURN
IF (INXC.EQ.1) GO TO 10

C-64


```c
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.
CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 6.
10 M = N-(N/6)*6
   IF (M.EQ.0) GO TO 20
   DO 15 I=M+1,M
       SASUM = SASUM+ABS(SX(I))
   15 CONTINUE
   IF (N.LT.6) RETURN
   MPI = M+1
   DO 25 I=MPI,N,6
       SASUM = SASUM+ABS(SX(I))
          +ABS(SX(I+1))+ABS(SX(I+2))+ABS(SX(I+3))+ABS(SX(I+4))+ABS(SX(I+5))
25 CONTINUE
RETURN
END

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

C -65
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), passno(4000,2)
common /hsort/ ra

c---------------------------------------- program description

check locates all passes with a minimum below a user defined
value, a maximum above a user defined value, a variance above
a user defined value, if a pass is selected, then the pass number
and the above values are written to the screen. This program
can be used on direct access 21-27r files or on files with a
header and one variable per pass. This program is used to find
cases which are influenced by external fields as noted by
their variance properties.

c program date: 22 apr 91

c updates: 6 jun 92; added sort subroutine
NOTE: check now removes the checked passes (ie. the high
variance passes)from the ordered pass number file
written by reorder.
NOTE: now check also orders the output file according to
the average elevation of the pass.
NOTE: these new options are not available for 21-27r input.
20 jul 92; added output file on unit 21
NOTE: this update simplifies the usage of check

c write (*,**) '0 IF INPUT FILE IS 21-27R'
write (*,**) '1 IF INPUT FILE IS HEADER AND VARIABLE'
read (*,**) type
if (type) 50,50,200

50 write (*,**) 'INPUT 21-27R FILE:'
read (*,9990) filename
format (a80)
open (i0, file-filename, status='old', form='unformatted',
> access='direct', recl=116)

write (**,*) 'WHICH VARIABLE DO YOU WANT FROM THE 27'
write (**,*) 'bva xa ya za totfld yfld zfld inc dec'
write (**,*) 'totmag totavgmag resid resav_mag ringcur sac'
write (**,*) 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, maxvar, minmin, maxvar

write (**,*) 'PASSNO CNT MAX MEAN MIN',
> 'DIFF MAXVAR'
c
21 jstop=0
recnum=1
countall=0
count=0
read (10, rec=recnum) (idata(i,1),i=1,2), (data(i,j),j=1,27)
n=2
22 recnum=recnum+1
read (10, rec=recnum, err=24) (idata(n,1),i=1,2), (data(n,j),j=1,27)
if (idata(n,1) .ne. idata(n-1,1)) go to 25
n=n+1
24 go to 23
25 continue

continue
c
continue
c
continue
c
countall=countall+1
total=0.0
xsumsq=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,1)
xsumsq=xsumsq+data(i,1)*data(i,1)
maxval=max(maxval, data(i,1))
minval = min(minval, data(l, var))
diff = data(l, var) - data(l+1, var)
maxdiff = max(maxdiff, abs(diff))

continue
mean = total/(n-1)
xvar = (xsumsq - ((total)**2)/real(n-1))/real(n-2)
absmean = abs(mean)
if (absmean > .or. maxval > maxmax .or. 
    minval > minmin .or. maxdiff > .or. 
    xvar > maxvar) then
  write (*, 9992) idata(n-1, n-1), maxval, mean, minval, 
  maxdiff, xvar

9992 format (215,5l(x, f12.6))
  count = count + 1
endif
data(n, var) = dummy
continue
28 continue
  do 28 i = 1, 2
  idata(i, 1) = idata(n, 1)
  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
continue
200 continue
  countall = 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'
  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 1 l = 1, 10000
      read (11, 9991) test
      icnt = icnt + 1
      if (test .eq. ' new') then
        test = 11, 9991)
        test = test + 205
        icnt = icnt - 2
        write (*, *) 'read ', lcnt, ' passes from pass file'
        write (*, *) 'OUTPUT FILE OF CHECKED PASSES'
        read (*, 9990) filename
        open (21, file = filename, form = 'formatted')
        do 1 l = 1, lcnt
          (passmjd(l, j), j = 1, 2), (passord(l, j), j = 1, 2)
        end do
      endif
      1 continue
      icnt = icnt - 2
      write (*, *) 'read ', lcnt, ' passes from pass file'
      write (*, *) 'OUTPUT FILE OF CHECKED PASSES'
      read (*, 9990) filename
      open (21, file = filename, form = 'formatted')
      do 1 l = 1, lcnt
        read (11, *) (passmjd(l, j), j = 1, 2), (passord(l, j), j = 1, 2)
      end do
    1 continue
    write (*, *) 'PASS NO CNT MAX MEAN MIN DIFF MAXVAR'
  210 read (10, end = 500) row, col, zero, xmean, pass, eight
    do 220 l = 1, row
      read (10) (data(l, j), j = 1, col)
    220 continue
    countall = countall + 1
    total = data(row, l) - data(l, var) + data(row, var)
SUBROUTINE SORT(N)
    double precision ra,rra
    common /hsort/ ra(4000)
    this subroutine is written by the authors of: Numerical Recipes (fortran);
    The Art of Scientific Computing
    Cambridge University Press
    1989, p. 230
    the routine is referred to as "heapsort"
    Copyright (C) 1986, 1992 Numerical Recipes Software

100 CONTINUE
    IF (L.GT.1) THEN
        L=L-1
        RRA=RA(L)
    ELSE
        RRA=RA(I)
        IR=1H+1
        IF (IR.EQ.1) THEN
            RETURN
        ENDIF
    ENDIF
    I=L
    J=L+L
200 IF (J.LE.IR) THEN
    IF (J.GT.1) THEN
        IF (J.GT.1) THEN
            L=L+L
        ELSE
            L=L-1
        ENDIF
    ENDIF
    CONTINUE
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+1
ELSE
   J = J+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)

program description
statmat defines the basic statistics of the input grids, see
the write statements for the specific values calculated, also,
the code loops through all input grids and calculates the
correlation coefficients between all combinations of input
data.

write (*,*) 'OUTPUT STATISTICS FILE'
read (*,9990) statfile
open (25, file=statfile, form='formatted')

read all data into array (row, col, layer or map number)
i=1
write (*,*) 'INPUT MATRIX WITH TPILOT HEADER'
read (*,9990) filename(1)
9990 format (a70)
open (10, file=filename(1), status='old', form='formatted')

read (i0,*) col(1)
read (i0,*) row(i)
read (i0,*) colat(i)
read (i0,*) long(i)
read (i0,*) space(i)
read (i0,*) ((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=1+1
go to 10

maximum number of input data sets = count
loops from line 80 to 400 increment through all
combinations of map to map
50 count=1
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

from 200 to write statement of variables is
the statistical calculations using two
references:
1) Davis, Statistics and Data Analysis in
Geology, 2nd ed., 1986 pp. 41
2) Young, Statistical Treatment of Experimental Data, 1962, McGraw Hill, 115-132

80 continue
do 400 icnt=1,count
do 400 jct=1-icnt,count

do 100 j=1,row(icnt)
do 100 k=1,col(icnt)
j=(col(icnt)*{}-1)+k
xdate(i)=data(j,k,icnt)
ydate(i)=data(j,k,jcnt)
100 continue

loops that sum x, x**2, y, y**2 and xy
nobs=row(icnt)*col(icnt)
if (nobs .ne. 1) stop 0002
nobs=float(nobs)
xsum=0.0
xsumsqr=0.0
ysum=0.0
ysumsqr=0.0
sumxy=0.0
xmin=xmax
ymin-ymax
xdata(j)=data(j,k,jcnt)
ydata(j)=data(j,k,jcnt)
do 240 j=1,nobs
xsum=xsum+xdata(j)
ysum=ysum+ydata(j)
xsumsqr=xsumsqr+xdata(j)**2
ysumsqr=ysumsqr+ydata(j)**2
240 continue

D-7

PROCEEDING PAGE BLANK NOT FILMED
ysumqr = ysumqr + (ydata(J4))^2
sumxy = sumxy + (xdata(J4) * ydata(J4))
xmax = ymax = (xdata(J4) * ydata(J4))

max = max(ydata(J4))
ymin = min(ydata(J4))

continue

------------- find corrected sum of products, covariance
            and corrected sum of squares (x) (y)

xmean = sum/nobss
ymean = ysum/nobss
sumprod = sumxy - ((xsum * ysum) / nobss)
covary = sumprod / (nobss - 1.0)
xsumqr = xsumqr - (xsum * xsum) / nobss
ysumqr = ysumqr - (ysum * ysum) / nobss

-------------- find variance, standard deviation for x and y

xvar = xsumqr / (nobss - 1.0)
yvar = ysumqr / (nobss - 1.0)
xstdev = sqrt(xvar)
ystdev = sqrt(yvar)

c---------------- find slopes, intercepts and correlation coefficient by Young method
xslope = ((nobss * sumxy) - (xsum * ysum)) / ((nobss * xsumqr) - xsum * xsum)
yslope = ((nobss * sumxy) - (xsum * ysum)) / ((nobss * ysumqr) - ysum * ysum)

-------------- write out this mess for individual pass and overlapping lengths of passes
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) covary, corrxy
9993 format('COVARIANCE XY-', f9.3, ' Davis CORRELATION COEF-', f8.3)
write (25, 9994) xsumqr, ysumqr, xintcpt, yintcpt, corrxy
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) xmean, xmin, ymax, ymin
           ' Y-MIN=', f9.3,)

c----------------- increment to next set of passes
continue

c999 continue
close (10)
close (25)
stop
eend
program part2

c==== convert magsat text from ebcdic to ascii, reorder integer bytes,

and translate ibm real to dec real

c editorial note: this code was supplied quite generously by Dr. Gary P.
Murdock

implicit none

c==== parameter storage: integer rellen

parameter (rellen=3024)

c==== common storage:

character*1 ascconv(256)

common /asacom/ascconv

integer*4 recnum,position

common /xyyyzz/recnum,position

c==== equivalence storage:

integer*4 inbufl(rellen/4),outbufl(rellen/4)

character*1 inbufc(rellen),outbufc(rellen)

equivalence (inbufl,inbufc), (outbufl,outbufc)

c==== local storage:

integer il,i2

character*80 fllename

character*1 cl

c==== data: (0=no translate, 1=real*4, 2=integer*4, 3=ebcdic)

integer*2 headtyp(557)

/14*2,4*1,2*3,2*4,2*3,2*6,0*12,1*30,3*3,2,
> 490*/

integer*2 datatyp(756)

/5*2,6*19*1,30*2,30*0/

c==== functions:

integer*4 realconv

c==== constants:

cl = char(1)

c write (*,*),'input file:

read (*,99901) filename

99901 format (a80)

open (21, file-filename, status='old',
> access='direct', form='formatted', recl-rellen)

write (*,*) output file:

read (*,99901)

open (31, file-filename,
> access='direct', form='formatted', recl-rellen)

recnum = 1

read (21,92101, rec=recnum, err=200) inbufc

92101 format (50000a)

if (inbufc(4).eq.cl) then

do position = 1,557

headtyp(position)+1

goto 104

101 outbufi(position) = realconv(inbufi(position))

goto 104

102 il = position+4

outbufc(il-3) = inbufc(il)

outbufc(il-2) = inbufc(il-1)

outbufc(il-1) = inbufc(il-2)

outbufc(il) = inbufc(il-3)

goto 104

103 il = position+4

do 12 = il-3,il

outbufc(12) = ascconv(char(inbufc(12))+1)

end do

104 end do

else do position = 1,756

headtyp(position)+1

goto 108

105 outbufi(position) = realconv(inbufi(position))

goto 108

106 il = position+4

outbufc(il-3) = inbufc(il)

outbufc(il-2) = inbufc(il-1)

outbufc(il-1) = inbufc(il-2)

outbufc(il) = inbufc(il-3)

goto 108

107 il = position+4

do 12 = il-3,il

outbufc(12) = ascconv(char(inbufc(12))+1)

end do

108 end do

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, 31995120,
> -202946443, -1886250728, 522067228, 45449852,
> -193891735, -1827237488, 76977556, -168436992,
> 446567970, -1566466016, -1499093853, 77758887, 556476476,
> -1414878938, -850986597, 610120112, 1580935466, -1280168147,
> -1371776072, 746371512, 1061052197, 1117046466, -1044332610,
> -591028418, 574433088, 1667391939, 1734763876, -976987304,
> -909588538, 1818979018, 1886350957, -890862127, -791687475,
> -1953726161, 2021095029, -741180807, -67487432, -404298268,
> -539042340, -404298268, 1128415611, 1195787598,
> -70652856, -303240214, 1280002685, 1347374669, -269594031,
> -202182160, 1414766428, 1482118741, -168535463, -101124158,
> -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 IBM
CHARACTER*1 CIBM (4)
EQUIVALENCE (IBM, CIBM)
INTEGER*4 IDC
CHARACTER*1 CDEC (4)
EQUIVALENCE (IDC, CDEC)
C
C...... common storage:
integer*4 recnum, position
COMMON /xxyyzz/recnum, position
C
C...... LOCAL STORAGE:
INTEGER COUNT
LOGICAL SIGNFLAG
CHARACTER*1 CO
C
C...... "CONSTANTS"
CO = CHAR(0)
C
C...... MOVE ARGUMENT TO EQUIVALENCE AREA
IBM = IBM
C
C...... SWITCH MANTISSA BYTES INTO dec
CDEC(1) = CIBM(4)
CDEC(2) = CIBM(3)
CDEC(3) = CIBM(2)
CDEC(4) = CO
C
C...... ZERO NON-MANTISSA BYTE
CDEC(4) = CO
C
C...... CHECK FOR 0.0
IF (CDEC(1).EQ.CO .AND. CDEC(2).EQ.CO .AND. CDEC(3).EQ.CO)
GOTO 120
C
C...... SHIFT MANTISSA BITS LEFT UNTIL A 1 IS FOUND, DISCARD THE 1,
C...... KEEP COUNT
COUNT = 0
100 IDC = ISHFT(IDC,1)
IF (CDEC(4).NE.CO) GOTO 110
COUNT = COUNT+1
GOTO 100
C
C...... EXTRACT AND CLEAR SIGN BIT
110 SIGNFLAG = BTEST(IBM, 7)
IBM = IBCLR(IBM, 7)
C
C...... CALCULATE NEW EXPONENT
CIBM(2) = CO
CIBM(3) = CO
CIBM(4) = CO
IBM = IBM*COUNT-130
IF (IBM.GT.255 .OR. IBM.LT.0) THEN
WRITE (*,99901) recnum, position
99901 format ("IBM value out of range in", i6, ",", i3)
CDEC(1) = CO
cDEC(2) = CO
cDEC(3) = CO
cDEC(4) = CO
GOTO 120
END IF

D-11
C----- MERGE DEC SIGN, EXPONENT AND MANTISSA
  CDEC(4) = CIBM(1)
  IDEC = ISHFT(IDEC,-1)
  IF (SIGNFLAG) IDEC = IBSET(IDEC,31)
  REALCONV = IDEC
  RETURN
END
FORTRAN Programs to Process Magsat Data for Lithospheric, External Field, and Residual Core Components

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

Goddard Space Flight Center
Greenbelt, Maryland 20771

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

Unclassified-Unlimited

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

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.

Magsat, Fourier, Geomagnetic

Unclassified

Unclassified

Unclassified

Unclassified

Unlimited