NASA Technical Memorandum 104612

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

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

Geodynamics Branch
Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. PASS PROCESSING</td>
<td>3</td>
</tr>
<tr>
<td>III. MAP PROCESSING</td>
<td>9</td>
</tr>
<tr>
<td>IV. CONCLUSIONS</td>
<td>15</td>
</tr>
<tr>
<td>V. ACKNOWLEDGEMENTS</td>
<td>15</td>
</tr>
<tr>
<td>VI. REFERENCES CITED</td>
<td>16</td>
</tr>
<tr>
<td>Figure 1: Data Processing Flow Chart</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2: Pass-to-Pass Processing Schematic</td>
<td>19</td>
</tr>
<tr>
<td>APPENDIX A: DATA EDITING AND COMPUTING REQUIREMENTS</td>
<td></td>
</tr>
<tr>
<td>Computing Environment</td>
<td>A-1</td>
</tr>
<tr>
<td>Helpful UNIX Commands</td>
<td>A-2</td>
</tr>
<tr>
<td>Suggested Improvements</td>
<td>A-4</td>
</tr>
<tr>
<td>APPENDIX B: PROFILE PROCESSING</td>
<td></td>
</tr>
<tr>
<td>subcore.f</td>
<td>B-1</td>
</tr>
<tr>
<td>reorder.f</td>
<td>B-17</td>
</tr>
<tr>
<td>massage.f</td>
<td>B-33</td>
</tr>
<tr>
<td>movetrunc.f</td>
<td>B-48</td>
</tr>
<tr>
<td>fourier1d.f</td>
<td>B-59</td>
</tr>
<tr>
<td>combine.f</td>
<td>B-75</td>
</tr>
<tr>
<td>gsf1283 coefficients</td>
<td>B-86</td>
</tr>
<tr>
<td>APPENDIX C: MAP PROCESSING</td>
<td></td>
</tr>
<tr>
<td>collocation.f</td>
<td>C-1</td>
</tr>
<tr>
<td>fourier2d.f</td>
<td>C-10</td>
</tr>
<tr>
<td>avgdifres.f</td>
<td>C-39</td>
</tr>
<tr>
<td>sqrmap.f</td>
<td>C-42</td>
</tr>
<tr>
<td>inversion.f</td>
<td>C-45</td>
</tr>
<tr>
<td>rmagcov</td>
<td>C-67</td>
</tr>
<tr>
<td>APPENDIX D: STATISTICS AND DATA CONVERSIONS</td>
<td></td>
</tr>
<tr>
<td>check.f</td>
<td>D-2</td>
</tr>
<tr>
<td>statmat.f</td>
<td>D-7</td>
</tr>
<tr>
<td>part2.f</td>
<td>D-10</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

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

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

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

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

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

The programs used in this section include: 1) subcore, 2) reorder, 3) massage, 4) movetrunc, 5) 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 \texttt{movetrunc}, four sets of data will exist including upper
and lower altitude dawn data and upper and lower altitude dusk data.

Before \texttt{movetrunc} is run, the program \texttt{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 \texttt{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.

\texttt{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
\texttt{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 \texttt{fourier1d} (Section II.E) can correlate the immediately
adjacent profiles.

\textbf{II.E \texttt{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 \texttt{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 \textit{fourier1d}.

\[
12, 11, \ldots, 2, 1, 2, \ldots, 11, 12, 13, \ldots, 109, 110, 111, \ldots, 120, 121, 121, 120, \ldots, 111, 110
\]

folded data ---I------------- original data ---------------|---------- 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 \textit{fourier1d} program and the user should consider bandpass and correlation coefficient filtering of the data at this time.

\textbf{II.F COMBINE}

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

II.G FOURIER1D

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

II.H COMBINE

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

This concludes the data processing as applied to passes.
III. MAP PROCESSING

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

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

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

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

The following programs are presented in correct running order.

III.A COLLOCATION

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

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

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

III.B FOURIER2D

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

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

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

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

III.C AUXILIARY MAP PROCESSING

The grid processing steps of this section are not necessary for standard map reductions; however, these steps are presented for completeness. The processing of this section removes possible external field influences manifested as coherent differences between the correlated dawn and dusk maps. Also, the standard deviations (ie. energy levels) of the correlated dawn and dusk maps are adjusted to nearly the same level.
III.C.1 AVGDIFRES

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

III.C.2 FOURIER2D

This is the second application of fourier2d on the grids. Here, the difference grid from avgdifres (e.g., low.diff1) 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


**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 -JM 10Mw -IT 3600 shellfile

The -JM 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)
script prompt> output file from massage (e.g., dk.var)

script prompt> 1

script prompt> 200.0  200.0  200.0  -200.0  500.0

( these are suggested values)

script prompt> yes

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

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

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

    script prompt> exit

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

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

        1074  129  75.982  -0.023  -29.244  18.222  1087.460

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

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

    UNIX prompt> cp file2 file2.low

    UNIX prompt> cp file2 file2.hi

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

Finally, combine the edited file2.low and file2.hi with file1 as follows:

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

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

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

A.3 SUGGESTED IMPROVEMENTS

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

1. subcore: Arrays that are in part named "data" could be combined so that the same array is passed to each subroutine.
2. reorder: A better method of finding the average longitude of short passes may decrease total run time.
3. movetrunc and combine:
   a) These two programs are very similar and with some modifications the programs could be combined.
   b) Because the output lat-long-radii files are similar, only one file containing flags indicating the index locations where the passes overlap is necessary for output.

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

1. collocation:
   a) Invert only the symmetric half of the COVM array.
   b) Use a faster sorting routine for finding the closest points to a grid node location.
2. inversion: For arrays larger than the allocated machine memory, an option should be inserted that uses disc space for the matrix inversion.

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

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

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

PROGRAMS
  subcore.f
  reorder.f
  massage.f
  movetrunc.f
  fourier1d.f
  combine.f

DATA FILE
  gsfc1283
program subcore
real*4 rbuff(3024/4), minlat, maxlat, minlon, maxlon,
seconds(3000), rhead(2228/4),
integer*4 ibuff(3024/4), flag1, flag2, headcnt, recnum,
outnum, dnxnum, dnxnum,
ndxpass, datacnt, stop10, stop11,
head10, data10, head1, data1, head2, data2, totrecord,
dinrec, dinrec, nowant(50), ihead(2228/4)
character*80 filename
character*60 aaid
equivalence (ibuff, rbuff), (cbuff, rbuff), (flag1, ibuff(1)),
(flag2, ibuff(2)), (ihead, rhead), (flag1, ihead(1)),
(flag2, ihead(2)), (chead, rhead)
common /latlon/ minlat, maxlat, minlon, maxlon
common /dndkdat/ dndata(1500, 2), dkdata(1500, 26),
idndata(1500, 2), idkdata(1500, 26),
common /coeff/ gg(50, 50), ggt(50, 50), ggtt(50, 50), jnum, knum,
ttzero, aaid, mmaxn, ttemp
common /malnfld/ fld(1500, 8), dawnf, dawni, duske, duskl, dndxpass
common /thatsit/ outdawn(1500, 27), outdusk(1500, 27)
COMMON /NASA/ TG(50, 50)
COMMON /FLDCOM/ ST, CT, SPH, CPH, R, MAX, BT, BF, BR, B
COMMON /magfld/ THETA(1500), PHI(1500), ELVO(1500), YEAR(1500)
common data(3000, 26), idata(3000, 2)

---

program description
subcore reads the 3 NASA INVESTIGATOR-B tapes from disc and obtains
the data for the user defined area. The program performs the following
corrections to the data: 1) reorders the dataset from NASA's column
elements to user manageable row arrays 2) removes all values at a
single sampling point if one of those values is flagged by NASA with
9999 3) selects the data for the user defined area 4) separates
the area into dawn and dusk datasets 5) calculates the core-field
value for every data point along the dawn or dusk profile and saves
that value in an array 6) removes the core-field values from the
data point along a dawn or dusk profile 7) removes NASA's ring-
current correction and 8) writes several values to the output
files - these values can be determined by looking at subroutine
corerlng.

NOTE: output unformatted files are direct access
input unformatted files are sequential access
output formatted files are sequential access

NOTE: use NASA Technical Memorandum No. 82160 for
a complete
description of each variable

program date: 16 apr 91

updates:
4 jun 92, added sequential access driver

NOTE: if you are working on an ibm rs6000 then the record
length for direct access on files 10, 11 and 12 then
remove the /4 from 3024/4.
if you are working on a dec3100:
and keep 3024/4.

write (*, *) 'INPUT FIRST DATASET FROM TAPE TO DISC TRANSFER'
read (*, 9990) filename
9990 format (a80)
open (10, file=filename, status='old', form='unformatted')
write (*, *) 'INPUT SECOND DATASET FROM TAPE TO DISC TRANSFER'
read (*, 9990) filename
write (*, *) 'INPUT THIRD DATASET FROM TAPE TO DISC TRANSFER'
read (*, 9990) filename
write (*, *) 'INPUT FIELD MODEL SPHERICAL HARMONIC'
write (*, *) 'COEFFICIENTS (GSFC1283)'
read (*, 9990) filename
open (13, file=filename, status='old', form='formatted')
write (*, *) '1 to remove certain pass numbers'
write (*, *) '0 do no remove any pass numbers'
read (*, *) ncnt
if (ncnt .eq. 0) then
write (*, *) 'input file of pass numbers not wanted'
read (*, 9990) filename
open (14, file=filename, status='old', form='formatted')
do i=1, 3024
read (14, *, end=10) nowant(i)
enddo

---

B-1
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)=1079
nowant(3)=1206
nowant(4)=2602
nowant(5)=2728
nowant(6)=2744
nowant(7)=2791
nowant(8)=2854
nowant(9)=3059
nocnt=9
write (*,*) 'OUTPUT DAWN DATA FILE OF 2-INTEGERS AND 27-REALS'
read (*,9990) filename
write (*,*) 'OUTPUT DUSK DATA FILE OF 2-I AND 27-R'
read (*,9990) filename
write (*,*) 'OUTPUT HEADERS FILE'
read (*,9990) filename
write (*,*) 'MINIMUM AND MAXIMUM LATITUDE OF STUDY AREA'
write (*,*) 'INPUT RANGE IS FROM -90.0 TO 90.0'
read (*,*) mlnlat,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, TZERO, AsID

FORMAT (211,1X,F6.1,A60) 9926 FOR_T (2II,IX, F6.I,A60)
FmAXN-0
TTEM-0.
50 READ (13,9928) N,M,GnM, HNM, GTM, GTM, GTTm, GTTm
9928 FOR_T (213,6FII.4)
IF (N.LE.0) GO TO 80
FmAXN- (MAX0 (N, MmAXN) )
Gg (N, M) -GNM
GgT (N,M) =GTNM
GgTT (N, M) =GTTTM
TTEM-AMAXI (TTEM, ABS (GTNM) )
IF (M.EQ.1) GO TO 50
Gg (M-1,N) =HNM
GgT (M-1,N) =HTNM
GgTT (M-1, N) =HTTTM
GO TO 50
80 CONTINUE

dnrec=0
dkrec=0
totrecord=0
datacnt=0
headcnt=0
stopI0=0
stopII=0

headI0=0
dataI0=0
recnum=1
100 num=1
read (10,end=120) rhead
headI0=headI0+1
go to 220

120 stopI0=1
close (10)
write (*,*) 'done with file one'
write (*,*) 'total headers on file one =',headI0
write (*,*) 'total data sets on file one =',dataI0
totrecord=totrecord+recnum
recnum=1

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

num=1
read (11,end=150) rhead
head1=head1+1
go to 220

c
150 stop1=1
  close (11)
  write (*,'done with file two')
  write (*,'total headers on file two =',head1)
  write (*,'total data sets on file two =',datall)
  head1=0
datall=0
totrecord=totrecord+recnum
recnum=1

num=1
read (12,end=999) rhead
head2=head2+1

c
220 continue

if (flag1 .eq. 1) then
  if (flag1 .eq. info) then
    write (23,'(head(4),(rhead(i),i=23,34))')
    else
      if (flag1 .eq. info) then
        write (23,'(head(4),(rhead(i),i=5,8))')
        (head(1),i=1,4), (head(1),i=5,8),
        (head(1),i=9,12)
        end
      endif
      endif
  endif
endif
write (2,8880)'(head(4),(rhead(1),i=5,8),(head(1),i=15,16)
8880 format (ix,15,4d15.7,216)
dndkpass=head(4)
duske=head(5)
duski=head(6)
dwne=head(7)
dawn=head(8)
headdnt=headdnt+1
recnum=recnum+1

c
if (flag2 .eq. 2) then
  continue
endif

i=num, num+29
idata(i, 1)=ibuff(5)
idata(i, 2)=ibuff(3)
xnum=real(i-num)
seconds(i)=real((i) + (rbuff(6)*xnum))
enddo
c

these do 250 loops reorder the data from column to row oriented data
j=1
do 250 j=1, 25
  do 250 i=num, num+29
    data(1, j)=rbuff(jj)
     jj=jj+1
  enddo
250 continue

num=num+30

c
if (flag2 .eq. 1) then
  innum=num-1
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. ideta(1,1)) then
      write (*,*) 'removed pass number ', nowant(i), ideta(1,1)
      if (stop10.eq.0 .and. stop11.eq.0) go to 100
      if (stop10.eq.1 .and. stop11.eq.0) go to 125
      if (stop10.eq.1 .and. stop11.eq.1) go to 155
   endif
endo
```

do 270 i=1,innum
   data(i,26)=seconds(i)
   continue
270

call nine (innum, outnum)
   innum=outnum
   call arearea (innum, outnum)
call dawn dusk (innum, dnnum, dknum)

c   if (dnnum .le. 0) go to 310
   call pfllgrf (dnnum,1)
   call corering (dnnum,1)
do 300 i=1,dnnum
   dnrec=dnrec+1
   write (20,rec=dnrec) (idndata(i,J), J=1,2), (outdawn(i,J), J=1,27)
300
   continue
310

c   if (dknum .le. 0) go to 360
   call pfllgrf (dknum,2)
   call corering (dknum,2)
do 350 i=1,dknum
   dkrec=dkrec+1
   write (21,rec=dkrec) (idkdata(i,J), J=1,2), (outdusk(i,J), J=1,27)
350
   continue
360

c----------------------------- these ugly little go to's get back to a header record

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   endif

c----------------------------- go back and get another data record

go to 230

c   endif

celse 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

c999 continue
   write (*,*) 'total headers on file three = ', ihead2
   write (*,*) 'total data sets on file three = ', data12
   write (*,*) 'total headers on tapes = ', headcnt
   write (*,*) 'total data sets on tapes = ', datacnt
   totrecord=totrecord+recnum
   write (*,*) 'total records read = ', totrecord
   write (*,*) 'total records written to dawn file = ', dnrec
   write (*,*) 'total records written to dusk file = ', dkrec
   close (12)
   close (13)
   close (14)
   close (20)
   close (21)
   close (22)
   close (23)
   stop
end
subroutine nine (innum, nlncnt)
    common data(3000,26),idata(3000,2)

----------------------------------------------- subroutine description
this subroutine removes from the data array all variables
associated with a single sampling point if selected variables at
that sampling point are greater than 99999.0

nlncnt=0

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 (data(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

nlncnt=nlncnt+1

do 140 j=1,2
    idata(nlncnt,j)=idata(i,j)
140 continue

do 150 j=1,26
    data(nlncnt,j)=data(i,j)
150 continue

continue

do 100 i=1,innum
continue

return
end

subroutine area(innum, outnum)
    integer*4 innum, outnum
    real*4 minlat, maxlat, mlnlon, maxlon
    common /latlon/ mlnlon, maxlon
    common data(3000,26),idata(3000,2)

----------------------------------------------- subroutine description
this subroutine removes all data outside of the user defined
area.

outnum=0

outnum=0
do 200 i=1,innum
    if (data(i,1) .gt. maxlat .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

outnum+1
continue

return
end

subroutine dawndusk (innum, dncnt, dkcnt)
    integer*4 innum, dncnt, dkcnt, totcnt
    common /dndkdat/ dndata(1500,26),dkdata(1500,26)
    common data(3000,26),idata(3000,2)

----------------------------------------------- subroutine description
this subroutine separates the data array into dawn and dusk data
sets.

data(innum+1,1) = -90.0

dkcnt=0
dncnt=0

do 200 i=1,innum
    if (data(i,1) .lt. data(i+1,1)) then
        dkcnt=dkcnt+1
        do 90 j=1,2
            idndata(dkcnt,j)=idata(i,j)
        enddo
    else
        dncnt=dncnt+1
        do 90 j=1,2
            idndata(dncnt,j)=idata(i,j)
        enddo
    endif
90 continue

return
end
```fortran
100 continue
else if (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
    do 140 J=1,2
       idndata(dncnt-1,J)=idata(i,J)
    140 continue
    do 150 J=1,26
       dndata(dncnt-1,J)=data(i,J)
    150 continue
    160 continue
elseif (data(i,1) .eq. data(i+1,1)) then
  write (*,8880) data(i,1),data(i+1,1),idata(i,1)
  8880 format ('two latitudes are equal therefore program skips',
               'the first latitude ',f9.4,' and reviews the second latitude ',f9.4,' for pass number',i6)
endif
200 continue

c totnct = dkcnt + dncnt
if (totntc .ne. innum) write (*,8881) dkcnt,dncnt,totntc,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,indnkd)
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 STEP14 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 STEP14
  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.

B-6```
If (Idndk .eq. 1) then
      do 50 i=1,innum
         ipass1=Idndata(i,1)
         mj=Idndata(i,2)
         secx=Idndata(i,26)
         theta(i)=Idndata(i,1)
         phi(i)=Idndata(i,3)
         elvo(i)=Idndata(i,3)
         ELVO(i) = ELVO(i) - 6371.2
         IDAYS = 44239 - MJD
         IF (IDAYS .GT. 0) THEN
            FRACY = FLOAT(365-(IDAYS)) / FLOAT(365)
            FRACYA = SECX / (3600000.'24.'365.)
            YEAR(I) = 1979.0 + FRACY + FRACYA
         ELSE
            FRACY = FLOAT(-IDAYS) / FLOAT(366)
            FRACYA = SECX / (3600000.'24.'366.)
            YEAR(I) = 1980.0 + FRACY + FRACYA
         ENDIF
      continue
   50
elseif (Idndk .eq. 2) then
      do 70 i=1,innum
         ipass2=Idndata(i,1)
         mj=Idndata(i,2)
         secx=Idndata(i,26)
         theta(i)=Idndata(i,1)
         phi(i)=Idndata(i,2)
         elvo(i)=Idndata(i,3)
         ELVO(i) = ELVO(i) - 6371.2
         IDAYS = 44239 - MJD
         IF (IDAYS .GT. 0) THEN
            FRACY = FLOAT(365-(IDAYS)) / FLOAT(365)
            FRACYA = SECX / (3600000.'24.'365.)
            YEAR(I) = 1979.0 + FRACY + FRACYA
         ELSE
            FRACY = FLOAT(-IDAYS) / FLOAT(366)
            FRACYA = SECX / (3600000.'24.'366.)
            YEAR(I) = 1980.0 + FRACY + FRACYA
         ENDIF
      continue
 70
endif

C**************************************************************
C
C
   COMMON /magfld/ THETA(1500),PHI(1500),ELVO(1500),YEAR(1500)
C   COMMON /malnfld/ fld(1500,8),dawne,dawnl,duske,duskl,dndkpass
C   INTEGER*4 ipass1,dndkpass
C   ************************************************************
C   THIS SUBROUTINE CALCULATES THE MAGNITUDE, INCLINATION, AND
C   DECLINATION OF THE GEOMAGNETIC FIELD ON A GRID NTHETA BY
C   NPHI
C   *************************************************************
C   SUBROUTINES USED
C      ** FIELDG ** (NASA)
C      ** FIELD ** (NASA)
C   *************************************************************
C
   LL=0
   RD=180./3.14159265
   DO 100 I=1,NPTS
      ATHETA = THETA(I)
      APHI = PHI(I)
      ELV = ELVO(I)
      YR = YEAR(I)
      CALL FIELDG (ATHETA,APHI,ELV,YR,50,LL,X,Y,Z,FF)
      H=SQRT(X*X+Y*Y)
      T=SQRT(H*H+Z*Z)
   100
C
B-7
C***************************************************************
CFOR DOCUMENTATION OF THIS SUBROUTINE AND SUBROUTINE
FIELD SEE:
C 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
GOODRARD SPACE FLIGHT CENTER, GREENBELT, MD.
C***************************************************************
CLAT ** 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
NMK ** 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
C...
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
CTLAST=0.0
C
IF(A.EQ.6378.139) IF(L) 210,100,110
IF(A.EQ.6371.2) IF(L) 210,100,110
C
A=6378.139
A = 6371.2
C
FLAT=1.-1./298.25
FLAT = 1.
A2=A**2
A4=A**4
B2=(A*FLAT)**2
A2B2=A2*(1.-FLAT**2)
A4B4=A4*(1.-FLAT**4)
IF (L) 160,210,110
100 IF (TM-CTLAST) 190,210,190
C
110 continue
1=0
j=1
k=1
num=ttzero
maxn=maxm
temp=temp
Do 120 l=1,maxn
do 120 li=1, maxn
g(li,1)=gq(li,1)
gt(li,1)=gtq(li,1)
gtq(li,1)=gtq(li,1)
gtq(li,1)=gtq(li,1)
120 continue
C
C 110 READ (3,260) J,K,TZERO, AID
B-8
C
L-0
C WRITE (7,270) J,K,TZERO, AID
C TEMP=0.
C 120 READ (3,280) N,M,GNM,HNM,GTNM,HTNM
C WRITE (7,280) N,M,GNM,HNM,GTNM,HTNM
C IF (N.LE.0) GO TO 130
C MAXN=(MAXO(N,MAXN))
C G(N,M)=GNM
C GT(N,M)=GTNM
C GTT(N,M)=GTNM
C IF (M.EQ.1) GO TO 120
C G(N-1,M)=HNM
C GT(N-1,M)=HTNM
C GTT(N-1,M)=HTTNM
C GO TO 120
C 130 WRITE (7,290)
C 130 CONTINUE
C DO 150 M-2,MAXN
C DO 150 M-I,N
C J=M-1
C IF (M.EQ.1) GO TO 140
C WRITE (7,300) N,M,G(N,M),GT(N,M),GTT(N,M),GT(N,M)
C GO TO 150
C 140 WRITE (7,310) N,M,G(N,M),GT(N,M)
C 140 CONTINUE
C 150 CONTINUE
C WRITE (7,320)
C 160 IF (K.NE.0) GO TO 190
C SHMIT(1,1)=1.
C DO 170 N=2,MAXN
C SHMIT(N,1)=SHMIT(N-1,1)*FLOAT(2*N-3)/FLOAT(N-1)
C SHMIT(1,1)=0.
C J=2
C DO 170 M=2,N
C SHMIT(N,M)=SHMIT(N,M-1)*SQRT(FLOAT((N-M+1)*J)/FLOAT(N+M-2))
C SHMIT(M-1,N)=SHMIT(N,M)
C 170 J=J+1
C DO 180 M=1,N
C G(N,M)=G(N,M)*SHMIT(N,M)
C GT(N,M)=GT(N,M)*SHMIT(N,M)
C GTT(N,M)=GTT(N,M)*SHMIT(N,M)
C IF (M.EQ.1) GO TO 180
C G(N-1,M)=G(N-1,M)*SHMIT(M-1,N)
C GT(N-1,M)=GT(N-1,M)*SHMIT(M-1,N)
C GTT(N-1,M)=GTT(N-1,M)*SHMIT(M-1,N)
C 180 CONTINUE
C 190 T=T-ZERO
C DO 200 M=1,MAXN
C DO 200 M=1,N
C T(N,M)=G(N,M)*T*GT(N,M)+GTT(N,M)*T
C IF (M.EQ.1) GO TO 200
C TG(N-1,M)=G(M-1,N)+T*(GT(M-1,N)+GTT(M-1,N)*T)
C 200 CONTINUE
C TIAST=IN
C 210 DIAT=DIAT/57.2957795
C SINLA=SIGN(DIAT)
C RLONG=RLONG/57.2957795
C CPH=CPH
C SPH=SPH
C IF (J.EQ.0) GO TO 220
C R=ALT+6371.2
C CT=SINLA
C GO TO 230
C 220 SINL2=SINLA**2
C COSL2=1.-SINL2
C DEN2=A2-A2*B2*SINL2
C DEN=SQR(DEN2)
C FAC=((ALT+DEN)+A2)/((ALT+DEN)+B2)**2
C CT=SINL2/SQR(FAC*COSL2*SINL2)
C R=SQR((ALT+2.)*DEN+((A4-A4*B4)*SINL2)/DEN2)
C 230 ST=SQR((1.-CT)**2)
C MAX=MIN((WKF,MAX))
C CALL FIELD
C Y=BP
C IF (J) 240,250,240
C 240 X=BT
C Z=BR
C RETURN
C C TRANSFORMS FIELD TO GEODETIC DIRECTIONS

B-9
250 SIND=SINLA*ST-SQRT(COSLA2)*CT
   COSD=SQRT(1.0-SIND**2)
   X--=BT*COSD-BP*SIND
   2*BT*SIND-BR*COSD
   RETURN

260 FORMAT (21I,1X,F6.1,A60)
270 FORMAT (213,5X,SHPOCH,F7.1,5X,A60)
280 FORMAT (213,6F11.4)
290 FORMAT (6H0
300 FORMAT (213,6F11.4)
310 FORMAT (213, FII.4,11X, FII.4,11X,FII.4)
320 FORMAT (///)
END

SUBROUTINE FIELD
COMMON /NASAG/ G(50,50)
COMMON /FLDCOM/ ST, CT, SPH, CPH, BT, BP, BR, B
DIMENSION P(50,50), DP(50,50), CONST(50,50), SP(50), CP(50), FN(50), FM(50)
DATA P(I,I)/0./
IF (P(I,I).EQ.0.0) GO TO 120
P(I,I)=1.
DP(I,I)=0.
SP(I)=0.
CP(I)=1.
DO ii0 N=2,NMAX
   FN(N)=-N
END
DO Ii0 M=1,N
   AM=371.2/R
   AR=AOR**2
   BT=0.
   BP=0.
   BR=0.
   DO 190 M=2,NMAX
      AM=AR+VAR
      DP(M)=DP(M-1)+DP(M-1)
      SP(M)=SP(M-1)+SP(M-1)
      DO 190 N=2,NMAX
         FN(N)=FN(N-1)+FN(N-1)
         DP(N,N)=DP(N-1,N-1)+DP(N-1,N-1)
         DO 190 M=1,N
            IF (N-M) 150,140,150
            DP(N,N)=DP(N-1,N-1)+DP(N-1,N-1)
            GO TO 160
         END
         DP(N,M)=DP(N-1,M)-FN(N,M)*AR
         IF (M.EQ.1) GO TO 170
         IF (N-M) 160,150,160
         PAR=P(N,M)*AR
         IF (M.EQ.1) GO TO 170
         TEMP=G(N,M)**SP(M)**FM(M)**PAR
         BP=BP-G(N,M)**SP(M)**FM(M)**PAR
         DO 190 N=2,NMAX
            BP(BP)+G(N,M)**SP(M)**FM(M)**PAR
            BP=BP+G(N,M)**SP(M)**FM(M)**PAR
            BP=BP+G(N,M)**SP(M)**FM(M)**PAR
            GO TO 180
         END
         BP=BP+G(N,M)**SP(M)**FM(M)**PAR
         GO TO 180
      END
      RETURN
END

SUBROUTINE CORERNG
COMMON /FLDCOM/ FLD(1500,8), DAWN, DAWN, DUSKL, DUSKE, DUSKL, DUSKE, DUSKL, DUSKE
COMMON /DNDKDAT/ DNDK(1500,26), DNDK(1500,26), DNDK(1500,26), DNDK(1500,26)
COMMON /THATISIT/ OUTDAWN(1500,27), OUTDAWN(1500,27)

c------------------- subroutine description
C this subroutine subtracts the core field at each data point and
C calculates the ring current affect as defined by NASA's formula
C which uses the I and V values for the entire orbit. this ring
C current value is also subtracted to yield the 'residual' value.
C since the total field value, the core-field value and the ring
C current values are written to file, any one value can be obtained
C at the next processing step.
C NOTE: the core field subtraction is not a least squares
C procedure. the least squares removal is done in
C program massage
if (idndk .eq. 1) then
  do 100 i=1,innum
  if (dndata(1,1) .ne. fild(1,8)) then
    write (*,*), 'no match between latitudes in corerling',
  write (*,*), dndata(1,1),fild(1,8)
  stop
else (dndkpass.ne.idndata(1,1) .or. dndkpass.ne.fild(1,1)) then
  write (*,*), 'no match between pass numbers in corerling',
  write (*,*), dndkpass, idndata(1,1),fild(1,1)
  stop
end if

if (dndata(1,1),ne.fild(1,8)) then
  write (*,*), 'no match between latitudes in corerling',
  write (*,*), dndata(1,1),fild(1,8)
  stop
else if (dndata(1,1),ne.idndata(1,1) .or. dndata(1,1).ne.fild(1,1)) then
  write (*,*), 'no match between pass numbers in corerling',
  write (*,*), dndata(1,1),idndata(1,1),fild(1,1)
  stop
endif

totmag=dndata(1,8)-fild(1,2)
tavgmag=dndata(1,12)-fild(1,2)
dip=dndata(1,6)*(pie/180.0)
delbzz=(dwnesin(dip)-2.0*dwnesin(dip)*
  ((radius/dndata(1,3))**3.0))
  delbxx=-1.0*dwnecos(dip)+dwnecos(dip)*
  ((radius/dndata(1,3))**3.0))
  ringcur=sqrt(((fild(1,3)+delbxx)**2.0) + (fild(1,4)**2.0) +
  ((fild(1,5)+delbzz)**2.0)) - fild(1,2)
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)=fild(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(1,26)
  continue
endif

if (dndata(1,1) .ne. fild(1,8)) then
  write (*,*), 'no match between latitudes in corerling',
  write (*,*), dndata(1,1),fild(1,8)
  stop
else (dndata(1,1) .ne. idndata(1,1) .or. dndata(1,1) .ne. fild(1,1)) then
  write (*,*), 'no match between pass numbers in corerling',
  write (*,*), dndata(1,1),idndata(1,1),fild(1,1)
  stop
endif

totmag=dndata(1,8)-fild(1,2)
tavgmag=dndata(1,12)-fild(1,2)
dip=dndata(1,6)*(pie/180.0)
delbzz=(duske*sin(dip)-2.0*duske*sin(dip)*
  ((radius/dndata(1,3))**3.0))
  delbxx=-1.0*duske*cos(dip)+duske*cos(dip)*
  ((radius/dndata(1,3))**3.0))
  ringcur=sqrt(((fild(1,3)+delbxx)**2.0) + (fild(1,4)**2.0) +
  ((fild(1,5)+delbzz)**2.0)) - fild(1,2)
resid=totmag-ringcur
resavgmag=tavgmag-ringcur
do 250 j=1,15
  outdusk(i,j)=dndata(i,j)
  continue
  do 270 j=16,21
    j=j-14
    outdusk(i,j)=fild(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)=dndata(1,26)
  continue
endif

B-11
return
end

c---------------------- this is the driver for direct access

  program subcore
  real*4 rbuff(3024/4), minlat, maxlat, minlon, maxlon,
  > seconds (3000)
  integer*4 lbuff(3024/4), flag1, flag2, headcnt, renum,
  > outnum, dnum, dnum, dnum,
  > ndnkpase, datacnt, stop1, stop2, head0, datal0, head1, datal1, head12, datal2, totrecord,
  > dnrec, drec, nowant (50)
  character*80 filename
  character*4 cbuff (3024/4)
  character*60 aaid
equivalence (ibuff, rbuff), (cbuff, rbuff), (flagl, ibuff (1)),
  > (flag2, ibuff (2))
  common /latlon/ minlat, maxlat, minlon, maxlon
  common /dndkdat/ dndsta (1500, 26), dkdata (1500, 26),
  > idndata (1500, 2), idkdata (1500, 2)
  contain /coeff/ gg (50, 50), ggt (50, 50), gqtt (50, 50), jnum, knum,
  > zero, said, n%maxn, ttemp
common /malnfld/ fld(1500, 8), dawne, dawnl, duske, duskl, dndkpass
  common /thatsit/ outdawn (1500, 27), outdusk (1500, 27)
  common /NASA/ TG(50, 50)
  common /NASA/ ST, CT, SPH, CPH, R, _X, BT, BP, BR, B
  common /magfld/ THETA (1500), PHI (1500), ELVO (1500), YEAR (1500)
common data (3000, 26), idata (3000, 2)
write (*,*), 'INPUT FIRST DATASET FROM TAPE TO DISC TRANSFER'
read (*, 9990) filename
format (a80)
open (10, file-filename, status='old', form='unformatted',
  > access='DIRECT', recl=3024/4)
write (*,*), 'INPUT SECOND DATASET FROM TAPE TO DISC TRANSFER'
read (*, 9990) filename
open (11, file-filename, status='old', form='unformatted',
  > access='DIRECT', recl=3024/4)
write (*,*), 'INPUT THIRD DATASET FROM TAPE TO DISC TRANSFER'
read (*, 9990) filename
open (12, file-filename, status='old', form='unformatted',
  > access='DIRECT', recl=3024/4)
write (*, *) 'INPUT FIELD MODEL SPHERICAL HARMONIC'
write (*, *) 'COEFFICIENTS (GSFC1283)'
read (*, 9990) filename
open (13, file-filename, status='old', form='formatted')
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'
c
cc write (*,*), '0 do no remove any pass numbers'
c
cc read (*,*) ncnt
cc if (ncnt.ne. 0) then
cc write (*,*), 'input file of pass numbers not wanted'
c
cc read (*,9990) filename
cc open (14, file-filename, status='old', form='formatted')
c
cc do i=1,5000
cc read (14, *, end=10) nowant(i)
c
cc enddo
cc 10 ncnt=nt-1
cc
cc------------------------ the following lines automatically place the nowant
cc pass numbers in the nowant array. these passes are
cc massed up for one reason or another. i'm sure they
cc could be salvaged, but i'm lazy.
c
cc nowant (1) = 999
cc nowant (2) = 1079
cc nowant (3) = 1206
cc nowant (4) = 2602
cc nowant (5) = 2728
cc nowant (6) = 2744
cc nowant (7) = 2791
cc nowant (8) = 2854
cc nowant (9) = 3059
cc ncnt=9
cc-------------------------- recli=116 for an ibm rs6000
cc
cc write (*,*) 'OUTPUT DAWN DATA FILE OF 2-INTEGERS AND 27-REALS'
c
cc read (*,9990) filename
cc open (20, file-filename, form='unformatted', access='DIRECT',
  > recli=29)
cc write (*,*) 'OUTPUT DUSK DATA FILE OF 2-1 AND 27-R'
c
cc read (*,9990) filename
cc open (21, file-filename, form='unformatted', access='DIRECT',
  > recli=29)
cc write (*,*) 'OUTPUT HEADERS FILE'
c
cc read (*,9990) filename
cc open (22, file-filename, form='formatted')
cc
B-12
* FOR NO ADDITIONAL INFORMATION'
write ('*',*) '1 FOR ONLY Data INDEXES'
write ('*',*) '2 FOR COMPLETE INFORMATION... this is a big file'
read (**,*) info
if (info .gt. 0) then
write ('*',*) 'OUTPUT ADDITIONAL INFORMATION ORBIT FILE'
read (**,9990) filename
open (23, file=filename, form='formatted')
endif
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
FO_4AT (211,IX, F6.1, A60)
MmAXN=0
READ (13, 9928) N, M, G_4, H_, GTNM, HTNM, GTTNM, HTTNM
FOF_4AT (213,6Fli.4)
IF (N. LE.0) GO TO 80
MmAXN= (_%X0 (N, M_AXN) )
Sg (N, M) =GNM
G_T (N, H) =GTNH
GgTT (N, M) -C_TT EMP-A_%XI (TtEMP, (ABS (GTNM))
IF (M.Eq.1) GO TO 50
Gg (M-l, N} =HNM
GgT (M-l, N) -HTNM
GgTT (M-l, N) -HTTNM
GO TO 50
CONTINUE

read (10,rec-recnum,err-120) rbuff
go to 220
stop10-1
write ('*',*) 'done with file one'
write ('*',*) 'total headers on file one =',head10
write ('*',*) 'total data sets on file one =',data10
head10=0
data10=0
recnum=1
num=1
read (10,rec-recnum,err-150) rbuff
num=1
read (11,rec-recnum,err-150) rbuff
go to 220
read (11,rec-recnum,err-150) rbuff
go to 220
read (11,rec-recnum,err-150) rbuff
go to 220
read (12,rec-recnum,err-999) rbuff
read (12,rec-recnum,err-999) rbuff
go to 220
read (12,rec-recnum,err-999) rbuff
go to 220
read (12,rec-recnum,err-999) rbuff
go to 220

if (flagl .eq. 1) then
if (info .eq. 0) then
go to 225
else (info .eq. 1) then
write (23, *) rbuff(1), (rbuff1(1),1=23,34)
else (info .eq. 2) then
write (23, *) (rbuff1(1),1=1,4), (rbuff1(1),1=5,8),

B-13
225 continue
write (22, 8880) ibuff (4), (rbuff (i), i-5,8), (ibuff (i), i-15,16)
write (22,8880) format (lx,15#4e15.7,216)
dndkpaas-lbuff (4)
dunake-rbuff (5)
dunBkl-rbuff (6)
dawnl-rbuff (7)
dawnl-rbuff (8)
headcnt -heedcnt+l
recn_-recn_+1
if (stopl0 .eq. 0 .and. stopll .eq. O) then
headl0-headl 0+i
go to 100
c elseif (stopl0 .eq. 1 .and. stopll .eq. O)
headll-heed/1+1
go to 125
c elseif (stopl0 .eq. i .and. stopll .eq. I)
head12-headl 2+1
go to 155
c endif
cc- if flagl-2 then this is a data file
c elseif (flag1 .eq. 2)
then
do 230 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) )
do 230 continue
cc- these do 250 loops reorder the data from column to row oriented data
cc
do 250 j=1,25
do 250 i=num, num+29
data (l,j)-rbuff (jj)
j=jj+1
250 continue
cc
num-num+30
cc- if flag2=1 then the next record is a header and the data information is complete for this orbit
cc
cc- search the nowant array for passes that just didn't happen. the following passes are doubled and are considered not wanted:
cc
909,1079,1206,2602,2728,2744,2791,2854,3059
cc
do l=1,nocnt
if (nowant(l) .eq. ldate(l,1)) then
write (*,*) 'removed pass number ', nowant(l), ldate(l,1)
go to 400
c endif
cc
do 270 l=1,lnnum
data(l,26)=seconds(l)
do 270 continue
cc
call nine (lnnum, outnum)
innum=outnum
call area (innum, outnum)
icnnum=outnum
call dawndusk (innum,dnnum,dknum)
cc
if (dnnum .le. O) go to 310
call pfligrf (dnnum,1)
call corering (dnnum,1)
do 300 l=1,dnnum
dnrec-dnrec+1
write (20, rec=dnrec) (ldndata(l,j),j=1,2),
(outdawn(l,j),j=1,27)
do 300 continue
cc
do 310 continue
cc
if (dknum .le. O) go to 360
call pfligrf (dknum,2)
call corering (dknum,2)
do 350 l=1,dknum
dkrec=dkrec+1
write (21, rec=dkrec) (ldkdata(l,j),j=1,2),
B-14
c 350 continue
  c 360 continue
  c endif
  cc
  c 400 recnum=recnum+1
  c datacnt=datacnt+1
  c if (stop10.eq.0 .and. stop11.eq.0) then
    c data10=data10+1
    c go to 110
  c elseif (stop10.eq.1 .and. stop11.eq.0) then
    c data11=data11+1
    c go to 130
  c elseif (stop10.eq.1 .and. stop11.eq.1) then
    c data12=data12+1
    c go to 160
  c endif
  cc
  c elseif (flag1.ne.1 .or. flag1.ne.2) then
    c write (*,*) 'HOLD THE FORT MAN, BAD FIRST FLAG NUMBER'
    c write (*,*) flag1
  c endif
  cc
  c 999 continue
  c write (*,*) 'total headers on file three =',headl2
  c write (*,*) 'total data sets on file three =',data12
  c write (*,*) 'total headers on tapes =',headcnt
  c write (*,*) 'total data sets on tapes =',datacnt
  c totrecord=totrecord+recnum
  c write (*,*) 'total records read =',totrecord
  c write (*,*) 'total records written to dawn file =',dnrec
  c write (*,*) 'total records written to dusk file =',dkrec
  c close (10)
  c close (11)
  c close (12)
  c close (13)
  c close (14)
  c close (20)
  c close (21)
  c close (22)
  c close (23)
  c stop
  c end
  cc
  c------------------ add on all subroutines from here on
program reorder
character*80 filename
integer passmaj(4000,2),ddata(400,2),store(2),
> countall,jstop, choice, passno(4000,dndk),
> pntcnt(400,2),shrtpas(3000,2),shrtcnt,
> passrec(4000,2),passrem,ctner,denum, dnum,choice,
> spknum, mincheck, spkvar, nowant(4000), outnum,
> minchk, cmtname, inrec, inum, outrec, olf, inf, onf,
> innumall, pntall(4000,2)
real data(400,27), ddata(27), east, west, diff, hlat,
> lolat, north, south, percent, tollat, upper, lower,
> desdata(400,27), inpdata(400,27)
double precision aver(4000,2), ra(4000), cross, passavg(4000,2),
> savglon(4000,2), cross
common data(400,27), ddata(400,2)
common /order2/ aver(4000,2), passmaj(4000,2), cross
common /order3/ savglon(4000,2), cross
common /hsort/ ra(4000)
common /shorty/ shrtpas(3000,2), shrtcnt, passrec(4000,2)
common /spike/ desdata(400,27), upper, lower, spkvar
common /intb/ inpdata(400,27)
c---------------------------------------------
program description
c this program takes data in the 2-integers and 27-reals format
and reorders the entire dataset into a sorted file according
to the variable that the user chooses. that variable is usually
the average longitude of each individual pass, such that after
reordering 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
elevation average or the pass numbers of each individual pass
(sorting by pass numbers = sorting by time). this program takes
about 15 minutes on the DECStation 3100.
the program requires DIRECT ACCESS or else it just won't happen!
NOTE: as crazy as it may seem, real*8 is necessary for the
accuracy because i 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
c 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
the despiked 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)
on (10, file=filename, status='old', form='unformatted',
> access='direct', recl=116)
write (*,*) '0 IF THIS IS A DUSK DATASET'
write (*,*) '1 IF THIS IS A DAWN DATASET'
read (*, dndk)
write (*,*) '1 TO AVERAGE AND REORDER ON LONGITUDE'
write (*,*) '2 TO AVERAGE AND REORDER ELEVATION'
write (*,*) '3 TO REORDER ON PASS NUMBER'
read (*, choice)
if (choice .eq. 1) then
write (*,*) 'OUTPUT FILE OF 21-27R DATASET REORDERED'
open (20, file=filename, form='unformatted', access='DIRECT',
> recl=116)
write (*,*) 'INTERMEDIATE 1/0 FILE NOT REORDERED'
read (*,9990) filename
open (21, file=filename, form='unformatted', access='direct',
> recl=116)
endif
if (choice .eq. 2) then
write (*,*) 'OUTPUT FILE OF 21-27R DATASET REORDERED'
read (*,9990) filename
open (20, file=filename, form='unformatted', access='direct',
> recl=116)
endif
write (*,*) 'OUTPUT FILE OF PASS NUMBERS AND AVERAGED',
read (*,9990) filename
open (22, file=filename,form='formatted')
cwrite (*,*) '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
cwrite (*,*) '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 (*,*) 'I-IT, 2-LONG, ...23-totavgmag...25-resav_mag...'
write (*,*) 'fat Ion rad mlt Invlat diplat x y z'
write (*,*) 'bva xa ya za totfld yfld zfld inc dec'
write (*,*) 'totmag totavgmag resid resav_mag ringcur sec'
read (*,*) spkvar
endif
read (*,*) cross 0.0
cross=cross
minchk=0
if (choice .eq. 1) then
write (*,*) 'WESTERN MOST LONGITUDE OF STUDY AREA'
write (*,*) 'EASTERN MOST LONGITUDE OF STUDY AREA'
write (*,*) '-180.0 to 180.0 NOT 0.0 to 360.0'
read (*,*) west,east
diffw = west - east
diffw .gt. 0.0 then
write (*,*) 'the program has determined that this study'
write (*,*) 'area crosses the 180.0, -180.0 meridian'
endif
write (*,*) 'NORTHERN AND SOUTHERN MOST LATITUDES'
write (*,*) '90.0 to -90.0 NOT 0.0 to 180.0'
read (*,*) north,south
write (*,*) 'PERCENT OF TOTAL LATITUDE LENGTH TO'
write (*,*) 'TO BE CONSIDERED TO FIND SHORT PASSES (90.0)'
read (*,*) percent
percent is used to calculate the range that
is used in subroutine shorts to determine
if a pass is a short pass or a long pass.
see shorts for more info. also, since no
passes go below or above 83.0 degrees, the
program resets north and south if needed.
if (north .gt. 83.0) north = 83.0
if (south .lt. -83.0) south = -83.0
totlat=abs(north-south)
minchk=int(((100.0-percent)/(2*100.0))*totlat)+1)*3
percent=((100.0-percent)/(2*100.0))*totlat
lomat=south-percent
hilat=north-percent
deendf
if (minchk .gt. mincheck) then
mincheck=minchk
write (*,*) 'minimum observation cut-off increased to',
 Explained: The code is designed to process input data and perform various operations such as reading files, making decisions based on user input, performing calculations, and saving output files. It involves handling data from files, making choices based on user input parameters, and processing the data to output results, which are then written to files for further use. The operations performed include sorting, averaging, and determining if the study area crosses the 180° meridian. The code also calculates latitudes and percentages to determine if a pass should be considered as short or long.
c----------------------------- 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.

  inrec=1
  outrac=0
  shrtcnt=0
  countal=0
  jstop=0
  passrem=0
  cntsame=0

  read (10,rec=inrec) (idata(1,i),i=1,2), (data(1,j),j=1,27)

  read (10,rec=inrec,err=110) (idata(n,1),i=1,2), (data(n,j),j=1,27)
  if (idata(n,1) .ne. idata(n-1,1)) go to 120
  n=n+1
  go to 105

  jstop=1
  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
  lnnum=n-1
  innnum=n

c---------------------------------- if passes are NOT wanted, remove them
  if (pchoice .eq. 0) go to 145
  do 143 ii=1,pchoice
  if (idata(Innum,1) .eq. nowant(ii)) then
    write (*,**,'PASS NUMBER REMOVED ',nowant(ii),
    >     idata(innum,1),
    >     go to 400
  endif
  143 continue
  145 continue

  if (lnnum .lt. mlncheck) then
    write (*,9980) idata(innum,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 from
  -180.0 to 180.0 meridian
  imerpass=idata(1,1)
  call meridian (innnum,imerpass)

c---------------------------------- despike the data if user chooses and
despike the number of times chosen
  if (spknum .eq. 0) go to 190
  cnter=0
  150 call despike (innnum,denum)
  cnter=cnter+1
  innnum=denum
  do 180 k=1,innnum
  do 180 kk=1,27
    data(k,kk)=desdata(k,kk)
  180 continue
  if (cnter .lt. spknum) go to 150
  190 continue
  if (innnum .lt. mlncheck) then
    write (*,9981) idata(innum,1),innnum
    9981 format ('PASS REMOVED AFTER DESPIKING -',16,
    >    ' OBSERV COUNT -',15)
    passrem=passrem+1
    go to 400
  endif

  c---------------------------------- interpolate the dataset
  call interp (dndk,lnnum,outnum)
  innnum=outnum
  do 210 i=1,innnum
  do 210 j=1,27
    data(i,j)=intndata(i,j)
  210 continue
  if (innnum .lt. mlncheck) then
    write (*,9982) idata(1,1),innnum
    9982 format ('PASS REMOVED AFTER INTERPOLATING -',16,
    >    ' OBSERV COUNT -',15)

B-19
passrem=passrem+1
go to 400
endif

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

c
260 continue
innum=cntsome+1
pntcnt(cntsome,1)=innum
pntcnt(cntsome,2)=idata(1,1)
pntcnt(cntsome,1)=idata(1,1)
pntcnt(cntsome,2)=innumall

c-------------------------------------------- subroutine finds all short passes
if (choice .eq. 1) call shorts (innum,hilat,lolat,dndk)
c-------------------------------------------- now call reorder1 which will find
the average longitude and elevation
(not radius) and store them in
aver(4000) as well as storing the
pass number and modified julian day for
the current pass.
c
 call reorder1 (innum,cntsome)
c-------------------------------------------- ok, now go back and get more passes
to average until done with the file

c
400 continue
do 410 i=1,2
idata(1,1)=istore(1)
410 continue
do 420 i=1,27
data(i,1)=dstore(i)
420 continue
if (jstop .eq. 1) go to 500
go to 100

c
500 continue

c-------------------------------------------- now sort the chosen variable
call sort (cntsome,choice)
c
do 550 j=1,cntsome
do 530 i=1,cntsome
if (choice .eq. 1) then
if (aver(j,choice) .eq. ra(i)) then
passno(i)=pasman(j,1)
passavg(i,1)=dble(pasman(j,1))
write (22,*1) (pasman(j,1),i),1
> passno(i)=passavg(i,1)
go to 540
endif
elseif (choice .eq. 2) then
if (aver(j,choice) .eq. ra(i)) then
write (22,*1) (pasman(j,1),i),1
> passno(i)=pasman(j,1)
go to 540
endif
elseif (choice .eq. 3) then
if (pasman(j,1) .eq. int(ra(i))) then
write (22,*1) (pasman(j,1),i),1
> passno(i)=pasman(j,1)
go to 540
endif
530 continue
540 continue
550 continue

c-------------------------------------------- now reread the file in reorder2 and write
it ordered according to the pass numbers
given to reorder2.
c
if (choice .eq. 1) then
inf=21
onf=20
elseif (choice .eq. 1) then
inf=10
onf=20
endif
call reorder2 (cntsome,inf, onf)
c-------------------------------------------- if sorting by average longitude, then must
extend the shorter passes and calculate a
new average. See subroutine reorder3 for
more information.

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)
          endif
      enddo 610
  enddo 600
  continue
  continue
  ra(i) = passavg(i,2)
  continue
  call reorder2 (cntsome, 10, 20)
end

B-21
subroutine reorder1 (nobs,num)
real data(400,27)
double precision nobss,along(4000),radd(4000),aalong,aavg,selev,
elev(4000),savg,aver(4000,2),cross
integer nobs,num,ndata(400,2),passmjd(4000,2),njd,
passnum
common /order1/ aver(4000,2),passmjd(4000,2),cross
common data(400,27),idata(400,2)
c
--- subroutine description ---

c 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
always decreases 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
e line (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.

c nobss-dble (nobs)
pasnum-idata (I, 1 )
mJd-ideta (I, 2)
do 50 n-l,nobs
along (n) -4ble (data (n, 2) )
radd(n) -c_le (data (n, 3) )
50 continue
aalong-0.0
do 110 n-l,nobs
if (along(n) .lt. along(n+l)) then
write (*,*) passnum,'CROSSES -180.0 TO 180.0',>
along(n), along (n+1 )
go to 130
endif
110 continue
daalong=aalong+along
120 continue
aavg=aalong/nobss
130 continue
selev-0.0
do 170 n-l,nobs
elev (n) =radd (n) -6378.140
selev=selev+elev
170 continue
savg=selev/nobss
c if (aavg .gt. 180.0) aavg=savg-360.0
passmjd(num,1)=passnum
passmjd(num,2)=njd
aver(num,1)=aavg + cross
aver(num,2)=savg
200 continue
return
end
c
--- subroutine reorder2 ---
integer nrecord(4000),passno(4000),npolnts(4000),
data(2),pntcnt(4000,2),passrec(4000,2),
shrtpas(3000,2),shrtcnt,inrec2,inf,onf,
pntall(4000,2)
real data(27)
c subroutine reorder2 (nlines,inf,onf)
common /order2/ passno(4000),pntcnt (4000,2),passrec(4000,2),
pntall(4000,2)
common /shorty/ shrtpas(3000,2),shrtcnt,inrec2,inf,onf,
pntall(4000,2)
the basis for this subroutine was provided
quite generously by:
Dr. D.R.H. O'Connell
Dept. of Geologcal 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 li=1,nlines
      if (passno(i) .eq. pntall(li,1)) then
        npoints(i)=pntall(li,2)
        go to 35
      endif
    continue
  continue
20 continue
elseif (inf .eq. 21) then
  do 40 li=1,nlines
    if (passno(i) .eq. pntcnt(li,1)) then
      npolnts(i)=pntcnt(li,2)
      go to 45
    endif
  continue
40 continue
45 continue
50 continue
endif
nponts = 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(i)=nrecord(li)+npoints(li)
passrec(i,2)=nrecord(li)
60 continue
read each data point
rewind (inf)
inrec2=0
70  inrec2=inrec2+1
read (inf,rec=1inrec2,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
80 do 90 i=1,nlines
   if (idata(i) .eq. passno(i)) then
      write (onf,rec=nrecord(i))
      * (idata(j),j=1,2), (data(k),k=1,27)
      nrecord(i)=nrecord(i)+1
50 continue
90 go to 70
write (*,*) inrec2 - 1, ' TOTAL RECORDS READ FOR FILE',inf
write (*,*) nrecord(nlines)-1, ' TOTAL RECORDS WRITTEN',' FOR FILE',onf
read next data point
end
10    continue
   elseif (choice .eq. 3) then
      do 30 i=1,n
         ra(i)=dble(passmjd(i,1))
      30    continue
   endif
C
   L=M/2+1
   IN=-N
   100 CONTINUE
   IF (L.GT.1) THEN
      L=L-1
      RRA=RA(L)
   ELSE
      PA=A(IR)
      RA(IR)=RA(1)
      IR=IR-1
   IF (IR.EQ.1) THEN
      RA(1)=RRA
   ENDIF
   ELSE
      IR=IR+1
   ENDIF
   J=J-L
   200 IF (J.LE.IR) THEN
      IF (J.LT.IR) THEN
         IF (RA(J).LT.RA(J+1) .AND. J=J+J)
         ELSE
            J=IR+I
         ENDIF
         GO TO 200
      ELSEIF
         RA(J)=RA
         GO TO 100
      ENDIF
   END
SUBROUTINE SHORTS (INNUM, HILAT, LOLAT, DNDK)
   INTEGER INNUM, IDATA(400, 2), DNDK, SHRTCNT, SHRTPAS(3000, 2), PASSREC(4000, 2)
   REAL DATA(400, 2), HILAT, LOLAT
   COMMON DATA(400, 2), IDATA(400, 2)
   COMMON /SHORTY/ SHRTPAS(3000, 2), SHRTCNT, PASSREC(4000, 2)

   SUBROUTINE DESCRIPTION
   SHORTS DETERMINES IF THE PASS IS SHORT, A SHORT PASS IS A PASS 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(INUM, 1) .GT. HILAT) THEN
         SHRTCNT=SHRTCNT+1
         SHRTPAS(SHRTCNT,1)=DATA(I,1)
         SHRTPAS(SHRTCNT,2)=INNUM
      ENDIF
      ELSEIF
         IF (DATA(INUM,1) .LT. LOLAT .OR. DATA(I,1) .LT. HILAT) 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, PASSREC(4000, 2),
   > ALLCNT, PASSNUM, RCNUM, ROW, RCNT, ROW, FROW,
   > ISDATA(400, 2), FREC, RREC, RCRC, PASS, IFDATA(400, 2),
   > 1RDATA(400, 2), HUMENT, MMINROW, STOCOUNT, MINROW,
   > DNDK, FNT, RNT, RECTYPE, RECL
   REAL SDATA(400, 2), FDATA(400, 2), RDATA(400, 2), SFTDATA(400, 2),
   > FDATA(400, 2), RDATA(400, 2), TGDFF, TCDFF, TOTDIFF,
   > FDIFF, RDIFF, FSTOT, RSTOT, STDATA(400, 2),
   > SFTDATA(400, 2), FTDATA(400, 2), SRTDATA(400, 2),
   > RTDATA(400, 2)
   DOUBLE PRECISION AVGZION, AVGDIFF, AVGDDIF, AVGQION, AVGQN(400, 2), CROSST
   COMMON /SHORTY/ SHRTPAS(4000, 2), SHRTCNT, PASSREC(4000, 2)
   COMMON /TRUNC/ SDATA(400, 2), FDATA(400, 2), RDATA(400, 2),
   > SFTDATA(400, 2), FTDATA(400, 2), SRTDATA(400, 2),
   > RTDATA(400, 2)
   DOUBLE PRECISION AVGZION, AVGDIFF, AVGDDIF, AVGQION, AVGQN(400, 2), CROSST
   COMMON /TRUNC/ SDATA(400, 2), FDATA(400, 2), RDATA(400, 2),
   > SFTDATA(400, 2), FTDATA(400, 2), SRTDATA(400, 2),
   > RTDATA(400, 2)
common /order3/ savglon(4000,2),cross

--- 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
cross passes, and stores that true average in an array. The short passes
are found by subroutine shorts 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. These values are then added to the
longitudes of the short pass to create a set of full length
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=1,allcnt
  if (shrtpas(i,1) .eq. passrec(i,1)) then
    recnum=passrec(i,2)
    passnum=passrec(i,1)
    row=shrtpas(i,2)
    recnt=i
  go to 200
endif
100 continue

--- read in the short pass
do 220 i=1,row
  read (20,rec-recnum) (isdata(i,j),j=1,2), (sdata(i,j),j=1,2)
  recnum=recnum+1
220 continue
recnum=recnum-1
if (passnum .ne. isdata(row,1)) then
  write (*,*) 'wrong pass number in reorder3'
  stop
endif

--- add 360.0 to longitudes that cross the
--- 180.0 to 180.0 meridian so that averages
--- are correct
do 230 n=1,row-1
  if (sdata(n,2) .lt. sdata(n+1,2)) then
    do 235 i=1,row
      if (sdata(i,2) .lt. 0.0) sdata(i,2)=sdata(i,2)+360.0
   235 continue
  endif
230 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=1,shrtcnt
  if (pass .eq. shrtpas(i,1)) then
    numcnt=numcnt+1
    if (numcnt .gt. allcnt) go to 270
    nrec=passrec(numcnt,2)
  go to 240
260 continue
frec=nrec

--- find the starting record number for the
--- nearest west pass
270 continue
nrec=0
numcnt-recnt-1
if (numcnt .lt. 1) go to 300
nrec=passrec(numcnt,2)
280 continue
read (20,rec=nrec) pass
do 290 i=1,shrtcnt
  if (pass .eq. shrtpas(i,1)) then
    nrec=passrec(numcnt,2)
  go to 240
290 continue

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

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

----------------------------------------------- truncate the short and forward passes
c to the same length
c call truncate (row, frow, dndk, minfrow, stocount, passnum, >
  ifdata(frow,1,1))
totfdiff=0.0
totfst=0.0
do 350 i=1, minfrow
  fdiff=abs(fdata(i,2)-fdata(i,2))
  totdiff=totdiff+fdiff
  totfst=totfst+fdata(i,2)
350 continue

----------------------------------------------- calculate the average longitude difference
avgfdiff=dble(totfdiff/real(minfrow))
endf
c

----------------------------------------------- repeat the process for the closest west pass
avgfwdiff=10000.0
if (rrec .gt. 0) then
  l=1
  read (20, rec=rrec) (irdata(i,j), j=1,2), (rdata(i,j), j=1,2)
  i=i+1
  rrec=rrec+1
  continue
endif
360 read (20, rec=rrec) (irdata(i,j), j=1,2), (rdata(i,j), j=1,2)
if (irdata(i-1,1) .ne. irdata(i,1)) go to 380
rrec=rrec+1
l=l+1
380 go to 360
continue
rrow=i-1
do 384 n=1, rrow
  if (rdata(n,2) .lt. rdata(n+1,2)) then
    do 382 i=1, rrow
      if (rdata(i,2) .lt. 0.0) rdata(i,2)=rdata(i,2)+360.0
    continue
    go to 385
  endif
  continue
384 continue
385 continue
c call truncate (row, rrow, dndk, minrrow, stocount, passnum, >
  irdata(rrow,1,1))
totrdiff=0.0
totrst=0.0
do 390 i=1, minrrow
  rdiff=abs(rdata(i,2)-rdata(i,2))
  totrdiff=totrdiff+rdiff
  totrst=totrst+rdata(i,2)
390 continue
avgfwdiff=dble(totrdiff/real(minrrow))
endf
c

----------------------------------------------- if the east difference is the smallest
c then use the east pass to calculate c average longitude of the extended short c pass
if (avgfwdiff .lt. avgfwdiff) then
do 400 i=1, frow
  if (fdata(i,2) .eq. fdata(i,2)) then
fcnt=1
  go to 410
end if
continue
continue
if (sftdat(1,2) .lt. ftdata(1,2)) then
  if (fcnt .eq. 0) go to 425
do 420 i=1,fcnt
  totsft=totsft+(ftdata(1,2)-real(avgfdiff))
continue
425 continue
else if (sftdat(1,2) .gt. ftdata(1,2)) then
  if (fcnt .eq. 0) go to 425
do 440 i=1,fcnt
  totsft=totsft+(ftdata(1,2)+real(avgfdiff))
continue
445 continue
endif
avgslon=dble(totsft)/dble(frow)
c
else if (avgrdiff .lt. avgfdiff) then
  do 500 i=1,rrow
    if (rtdata(i,2) .eq. rdata(i,2)) then
      rcnt=i
    go to 510
  endif
500 continue
510 continue
if (srtdat(1,2) .lt. rtdata(1,2)) then
  if (rcnt .eq. 0) go to 525
do 520 i=1,rcnt
  totsrt=totsrt+(rdata(i,2)-real(avgrdiff))
continue
525 continue
else if (srtdat(1,2) .gt. rtdata(1,2)) then
  if (rcnt .eq. 0) go to 545
do 540 i=1,rcnt
  totsrt=totsrt+(rdata(i,2)+real(avgrdiff))
continue
545 continue
endif
avgslon=dble(totsrt)/dble(rrow)
c
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

subroutine truncate (xrow,yrow,dndk,mlnrc_,etocount,
  xpassno,ypassno,fr)
  integer xrow,yrow,etocount,rowll,rownex,mlnrow,fr,
  dndk,xpassno,ypassno
  real xdata(400,2),ydata(400,2),sftdata(400,2),ftdata(400,2),
  adata,bdata,dlf,bas,
  rdata(400,2),srtdat(400,2)
  common /trunc/ xdata(400,2),ydata(400,2),sftdata(400,2),ftdata(400,2),
  adata,bdata,dlf,bas,
  rdata(400,2),srtdat(400,2)
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 55 jj=1,2
xdota(j,jj)=sdata(j,jj)
20 continue
if (fr) 50,50,60
do 55 jj=1,2
ydata(j,jj)=rdata(j,jj)
55 continue
go to 80
do 65 j=1,yrow
ydata(j,jj)=rdata(j,jj)
65 continue

80 continue
jrowil=xrow
rowinc=yrow

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

90 continue
a-data=xdata(jj,1)
bdata=ydata(jj,1)
diffab=sdata-bdata
abss=abs(diffab)

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
mlnrow=min(rowii,rowinc)

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
if (fr .eq. -1) then
do 110 li=1,mlnrow
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,mlnrow
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 or b depending on whether or not ascending
or descending order of independent variable

if (abss .ge. 0.33) then
if (dndk .eq. 0) then
if (xdata(jj,1) .gt. ydata(jj,1)) then
rowinc=rowinc-1
do 130 mm=1,rowinc
do 130 kk=1,2
ydata(mm,kk)=ydata(mm+1,kk)
130 continue
elseif (xdata(jj,1) .lt. ydata(jj,1)) then
rowii=rowii-1
do 150 nn=1,rowii
do 150 kk=1,2
xdata(nn,kk)=xdata(nn+1,kk)
150 continue
endif

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) .gt. ydata(jj,1)) then
rowinc=rowinc+1
do 130 mm=1,rowinc
do 130 kk=1,2
ydata(mm,kk)=ydata(mm+1,kk)
130 continue
elseif (xdata(jj,1) .lt. ydata(jj,1)) then
rowii=rowii+1
do 150 nn=1,rowii
do 150 kk=1,2
xdata(nn,kk)=xdata(nn+1,kk)
150 continue
endif

if this is a dawn pass then will count from
the equator toward the south pole
that is decreasing independent variable
else (dndk .eq. 1) then

B-28
if (xdata(jj,1) .lt. ydata(jj,1)) then
  rowinc=rowinc-1
  do 160 mm=rowinc
  do 160 kk=1,2
    ydata(mm,kk)=ydata(mm+1,kk)
  continue
elseif (xdata(jj,1) .gt. ydata(jj,1)) then
  rowinc=rowinc-1
  do 170 mm=rowinc
  do 170 kk=1,2
    ydata(mm,kk)=ydata(nn-1,kk)
  continue
endif
endif
c
end

c subroutine interp2 (dndk, num, ii)
real data(400,27), xdata(27), intpdata(400,27)
integer num, ldata(400,2)
dndk
common data(400,27), ldata(400,2)
common /lna/ xdata(27)
common /lnb/ intpdata(400,27)
c------------------------------ subroutine description
c the basic concept of this subroutine was provided by:
c Dr. D.N. "Tiku" Raver
c Dept. of Geology
c Purdue University
c 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.
c
i=0
xlat=real(int(data(i,1)*100.0))/100.0
if (dndk .eq. 0) then
  if (xlat .lt. data(1,1)) xlat=xlat + 0.33
  i=1
else if (xlat.ge.data(i,1) .and. xlat.le.data(1+1,1)) then
  call interp2 (i,xlat)
  xdata(2)=real(int(xdata(2)*100.0))/100.0
  intpdata(i,1)=xdata(j)
  continue
  xlat=xlat - 0.33
  if (xlat .lt. data(num,1)) return
  go to 100
else if (xlat .gt. data(i+1,1)) then
  i=i+1
  go to 100
endif
endif
c
end

c subroutine interp2 (inum,xlat)
real data(400,27), diffdata(27), xdata(27)
integer inum, ldata(400,2)
diffdata(27), ldata(400,2)
common /lna/ xdata(27)
c------------------------------ this subroutine is also from Tiku and is

B-29
100       do i=1,27
         diffdata(i) = data(inum, 1) - data(inum+1, 1)
         continue
100      enddo
120      do i=1,27
         xdata(i) = data(inum, 1) + (xlat - data(inum, 1)) * 
                     (diffdata(i)/diffdata(i))
         continue
120      enddo
      return
    end

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

PROGRAM DESPIKE

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

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
   CONTINUE
2  CONTINUE

ARE THE FIRST NEW POINTS SPIKES?
NOTE: DATA(U, 23) = RESID(U)

I=1
   SL1 = DATA((I+1, var) - DATA(I, var))
   SL2 = DATA((I+2, var) - DATA(I, var))
   SL3 = DATA((I+3, var) - DATA(I, var))
   SL4 = DATA((I+4, var) - DATA(I, var))
   XSL = ABS(SL1 + SL2 + SL3 + SL4) / 4.0
   S2 = DATA(I+2, var) - DATA(I+1, var))
   IF (ABS(SL1) .GT. (3.0 * XSL) .OR. ABS(SL1) .GT. (3.0 * S2)) 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 = S2
   20 CONTINUE

IS THE LAST POINT A SPIKE?

K=NPTS-2
   IF (IC(K) .EQ. 0) THEN
      K = K - 1
   ENDIF
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*SL3)) IC(NPTS)=0
ENDIF

C NOBS=0
C DO 30 I=1,NPTS
C IF (IC(I).EQ.1) NOBS=NOBS+1
C 30 CONTINUE
C WRITE(6,*) IDATA(I,I), NOBS
C
C outnum=0
DO 35 I=1,NPTS
IF (IC(I).EQ.1) THEN
outnum=outnum+1
DO 32 m=1,27
desdata(outnum,m)=data(i,m)
32 continue
35 ENDIF
35 CONTINUE
C
return
END
program massage
character*80 filename
real data(400,27),intpdata(400,27),movdata(400,27),
> dstore(2),denum,cnter, spknum,
> varmax,mincc,strlntdat(400)
integer Idata(400,27),cntall,lnnum,notnum,winlen2,incpen2,
> vary,nowant(400),col,eight,zero,gfcholce,
> coi,3,match,gotype,passrem,recnum, gen, winlen1,
> winlen2,winlen2b
common data(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)
common /move/ movdata(400,27)

--- program description

ok, the name of the program. to be frank, i just couldn't
think of a 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-integer 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 latitudes, longitudes
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
can 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, rec(1)=56 for dec3100, rec1=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', rec=116)

write (*,*) 'TYPE dawn OR dusk AS APPROPRIATE'
read (*,9991) dndk
9991 format (a10)

write (*,*) 'O IF YOU WANT ALL PASSES'
write (*,*) 'I IF CERTAIN PASSES NEED TO BE REMOVED'
read (*,*) choice
if (choice .eq. 1) then
write (*,*) 'INPUT FILE OF PASSES YOU DO NOT WANT'
read (*,9993) filename
open (11, file=filename,status='old',form='formatted')
endif
write (*,*) 'O FOR CUBIC SPLINE AND DATA OUTPUT'
write (*,*) 'I FOR ONLY CUBIC SPLINE OUTPUT'
write (*,*) 'O FOR ONLY DATA OUTPUT (the usual choice)'
read (*,*) gfcholce
write (*,*) 'WHICH 27R VARIABLE TO BE WRITTEN TO FILE(6) (12)'
write (*,*) 'lat lon rad ml lat dpl at bs by x y z'
write (*,*) 'bva bx ya za tot Alf fid yfid zfid inc dec'
write (*,*) 'totmag totmag resid resavmag ringvar sec'
write (*,*) 'O IF YOU WANT 21-27R OUTPUT'
read (*,*) vary
write (*,*) 'I FIT LEAST SQUARES CORE FIELD TO THIS VARIABLE'
write (*,*) 'O DO NOT FIT CORE FIELD'
write (*,*) 'choose 1'
read (*,9994) 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 (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 (*,*) '0 FOR NO DEPIKING OF DATA SET'
write (*,*) '1 FOR DEPIKING ONCE'
write (*,*) '2 FOR DEPIKING TWICE (this is the usual choice)'
write (*,*) '3 ... AND SO ON'
read (*,*), spkn_num
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 DEPIKE: (23)'
  write (*,*) 'I-LAT, 2-LONG, .., 12-bva, .., 23-totavgmag'
  read (*,9990) filename
  open (23, file=filename, form='unformatted')
endif
if (gfcholce .eq. 2) then
  write (*,*) 'WHICH VARIABLE TO WORK WITH IN CUBIC SPLINE: (12)'
  write (*,*) 'I-LAT, 2-LONG, .., 12-bva, .., 23-totavgmag'
  read (4,9990) filename
  open (23, file=filename, form='unformatted')
  write (*,*) 'OUTPUT FILE OF DATA - CUBIC SPLINE'
  read (*,9990) filename
  open (28, file=filename, form='unformatted')
endif
write (*,*) 'OUTPUT STATISTICS FILE'
read (*,9990) filename
open (25, file=filename, form='formatted')
write (25, *) 'PASS VAR STDEV MIN MEAN MAX'
write (*,*) 'OUTPUT FILE OF TRACKS FITTED WITH A CUBIC SPLINE'
read (*,9990) filename
open (26, file=filename, form='unformatted')
write (*,*) 'OUTPUT FILE OF TRACKS NOT FITTED'
read (*,9990) filename
open (27, file=filename, form='unformatted')
write (*,*) 'TYPE OF CUBIC SPLINE TO APPLY TO DATA: (2)'
write (*,*) '1 FOR A MOVING AVERAGE'
write (*,*) '2 FOR A MIN-MAX-AVERAGE FINDER'
read (*,*) gftype
if (gftype .eq. 1) then
  incen2=0
  winlen2=0
  winlen2a=0
  winlen2b=0
  write (*,*) 'WHAT IS THE LENGTH OF THE WINDOW:'
  read (*,*) winlen1
  write (*,*) 'HOW MANY POINTS TO INCREMENT WINDOW LOCATION:
  write (*,*) 'should be equal to or greater than 1'
  read (*,*) incen1
else (gftype .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 (*,*) winlen2c
  write (*,*) 'HOW MANY POINTS TO SEARCH FROM AN ENDPOINT TO FIND'
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 (*,*), winlen
write (*,*), 'IF MINCC CAN NOT BE MATCHED THEN'
write (*,*), 'HOW MANY POINTS TO INCREMENT WINDOW LOCATION?
read (*,*), winlen
endif
write (*,*), 'MAXIMUM VARIANCE WITHOUT FITTING A CUBIC SPLINE'
read (*,*), varmax
write (*,*), 'MINIMUM CORRELATION COEFFICIENT OF CUBIC SPLINE TO'
write (*,*), 'ORIGINAL DATA'
read (*,*), mincc
endif
write (*,*), 'AND FINALLY - WHAT IS THE MINIMUM NUMBER OF'
write (*,*), 'OBSERVATIONS ALLOWABLE FOR EACH PASS (50)'
read (*,*), mincheck
if (choice .eq. 1) then
  do 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
n=0
noc=0
noocf=0
itercnt=0
passrem=0
countall=0
jstop=0
ifirstcnt=0
iseccnt=0
iendcnt=0
11cnt=0
11cnt=0
11cnt=0
11cnt=0
11cnt=0
iblgcnt=0
iswitch=0
ilowcnt=0
recnum=0
read (10,rec=recnum) (idata(l,1),l=1,2), (data(l,j),j=1,27)
100 n=n+1
105 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. nowant(kk)) then
  go to 120
110 continue
jstop=1
110 continue
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
  c--------------------------------------- if passes are NOT wanted, remove them
  if (choice .eq. 0) go to 145
  do 63 ii=1,choice
    if (idata(ii,1) .eq. nowant(kk)) then
      go to 400
    endif
    63 continue
  continue
  i=1
145 continue
  if (innum .lt. mincheck) then
    write (*,9980) idate(1,1),innum
    9980 format ('PASS REMOVED AT READ = ',16,' OBSERV COUNT = ',15)
    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)
  cnter=1
  innum=denum
  do 180 kk=1,innum
   data(k,kk)=desdata(k,kk)
   continue
  end
190 continue

if (cnter .lt. spknum) go to 150

9981 if (spknum .eq. 0) go to 190

9982 continue

if (innum .lt. mincheck) then
   write (*,9981) idata(l,l),innum
   format ('PASS REMOVED AFTER DESPIKING =',l5, ' OBSERV COUNT =',i5)
   passrem=passrem+1
   go to 400
endif

call interp (dndk,innum,outnum)
innum=outnum
if (innum .lt. mincheck) then
   write (*,9982) idata(l,l),innum
   format ('PASS REMOVED AFTER INTERPOLATING =',l5, ' OBSERV COUNT =',i5)
   passrem=passrem+1
   go to 400
endif

do 205 i=1,innum
   x(i)=intpdata(l,i,1)
y(i)=intpdata(l,i,2)
205 continue

call moving (innum,num,ldata(l,l),eight,gfcnt,dndk,itercnt,> ifirstcnt,lseccnt,iendcnt,13cnt,15cnt,> i6cnt,17cnt,18cnt,bswitch,llowcnt)

300 continue

if (gfchoic .lt. 2) then
   do 310 k=1,innum
    data(k,kk)=intpdata(l,kk)
   end
310 continue

call moving (innum,num,ldata(l,l),eight,gfcnt,dndk,itercnt,> ifirstcnt,lseccnt,lendcnt,13cnt,14cnt,> 16cnt,17cnt,18cnt,19cnt,bswitch,llowcnt)

outnum=innum

if (eight .eq. 7777) call track (27,innum,noc,nocgf)
c

if (vary .eq. 0) then
   col=1
   col3=3
   zero=0
   mean=0.0
   eight=8888
   write (22,*) idata(l,l),xxx
   endif

if (gfchoic .eq. 1) go to 300
if (vary .eq. 0) go to 230

if (gfchoic .eq. 1) then
   call sgrfit(innum,xxx,vary)
   write (22,*) idata(l,l),xxx
   endif

300 continue
if (eight .eq. 8888) then
write (28) outnum, col, zero, mean, idata(1,1), eight
write (23) outnum, col, zero, mean, idata(1,1), eight
write (24) outnum, col3, zero, mean, idata(1,1), eight
enddo
elseif (eight .eq. 7777) then
iseight = 8888
write (23) eight
endif

call track (26, outnum, noc, nocgf)
do 350 m = 1, outnum
if (intpdata(m, 1, 2) .gt. 180.0)
intpdata(m, 1, 2) = intpdata(m, 1, 2) - 360.0
write (28) outnum, col, zero, mean, idata(m, 1), eight
write (24) idata(m, 1), intpdata(m, 1), eight
continue
endif
enddo

b-37
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,1) .lt. data(i+1,1)) then
    do 150 li=1,innum
        if (data(li,2) .lt. 0.0) data(li,2)=data(li,2)+360.0
        continue
      go to 200
    100 continue
    continue
    return
  endif

150 continue
end subroutine Interpl (c_dk,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
besing 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 .lt. data(i,1)) xlat=xlat + 0.33
  i=i+1
  if (xlat.ge.data(i,1) .and. xlat.le.data(i+1,1)) then
    call interp2 (i,xlat)
    xdata(2)=real (int (xdata(2)*100.0))/100.0
    ii=ii+1
    do 150 J=1,27
      intpdata(ii,J)=xdata(J)
     150 continue
    xlat=xlat + 0.33
    if (xlat .gt. data(num,1)) return
    go to 100
  elseif (xlat .lt. data(i+1,1)) then
    i=i+1
    go to 100
  endif
elseif (dndk .eq. 'dawn') then
  if (xlat .gt. data(i,1)) xlat=xlat - 0.33
  i=i+1
  if (xlat.le.data(i,1) .and. xlat.ge.data(i+1,1)) then
    call interp2 (i,xlat)
    xdata(2)=real (int (xdata(2)*100.0))/100.0
    ii=ii+1
    do 200 J=1,27
      intpdata(ii,J)=xdata(J)
     200 continue
    xlat=xlat - 0.33
    if (xlat .lt. data(num,1)) return
    go to 180
  elseif (xlat .lt. data(i+1,1)) then
    i=i+1
    go to 180
  endif
endif
endif
```
subroutine interp2 (lnum,xlat)
    real data(400, 27), diffdata(27), xdata(27)
    integer lnum
    common /inta/ xdata(27)
    this subroutine is also from Tiku and is
    the interpolator (not to be confused with
    the terminator!)
    do 100 l=1, 27
      dl = data(lnum+1, l)
      continue
    do 120 l=1, 27
      xdata(l) = data(lnum, l) + (xlat-data(lnum, l)) * 
               (diffdata(l)/diffdata(1))
      continue
    return
end

SUBROUTINE SPLINE (X, Y, N, YP1, YPN, Y2)
PARAMETER (NHAX=100)
DIMENSION X(N), Y(N), Y2(N), U(NHAX)
    subroutine description
    this subroutine provided by
    the authors of Numerical Recipes
    Numerical Recipes (Fortran)
    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)=-0.5
  U(1)=3. * (X(2)-X(1))*(Y(2)-Y(1))/(X(2)-X(1))-YP1)
ENDIF
DO 11 I=2, N-1
  SIG = (X(I)-X(I-1))/(X(I+1)-X(I-1))
  P = SIG*Y2(I-1)+2. * Y2(I)+SIG*P
  U(I) = (6.* (Y(I+1)-Y(I))/(X(I+1)-X(I))-Y(I)-Y(I-1)
         + (X(I)-X(I-1))/(X(I+1)-X(I-1))-SIG*U(I-1))/P
11 CONTINUE
IF (YPN.GT.99E30) THEN
  ON=0.
  UN=0.
ELSE
  ON=0.5
  UN=(1./X(N)-X(N-1))* (YN- (Y(N)-Y(N-1))/ (X(N)-X(N-1))
ENDIF
Y2(N) = ON-UN*U(N-1).
DO 12 K=N-1, 1,-1
  Y2(K)=Y2(K+1)+U(K)
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
1 IF (KHI-KLO,GT,1) THEN
  K=(KHI+KLO)/2
  IF(XA(K),GT,X) THEN
    KHI=K
  ELSE
    KLO=K
  ENDIF
  GOTO 1
ENDIF
H=XA(KHI)-XA(KLO)
IF (H.EQ.0.) PAUSE 'Bad XA input.'
A= XA(KHI)-KLO
B= 9-KLO
Y=AYA(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,
itercnt, ifirstcnt, isecent, iendcnt,
>                     ibincnt, iswitch, liowcnt,
integer winlen, incwen, hobs, sins,
>                   wwinlen2, linwen2, var, maxm,
>                     gftype, passnum, eight, gifcnt, subwinlen, strnobs,
>                     gftype, wwinlen1, wwinlen2a, wwinlen2b
real data (400, 27), movdata (400, 27), minval, maxval,
>                     varmax, strdata (400, 27), intpdata (400, 27), mincc,
>                     x(100), y(100), y2(100),
>                   adate (400, 27)
character*10 dndk
common data (400, 27),
>                     /move/ movdata (400, 27), wwinlen2, linwen2, var, gftype,
>                     varmax, mincc, linwen1, wwinlen1, wwinlen2a,
>                     wwinlen2b
common /intb/ intpdata (400, 27)

--- subroutine description ---
subroutine moving creates the cubic spline fit of each pass with
a variance above the user defined limit. This cubic spline will
match the original pass to 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:
Davis, Statistics and Data Analysis in
Geology, 2nd ed., 1986 pp. 41

--- loops that sum x, x**2 ---
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=real(nobs)
xsumsqr=xsumsqr-((xsum**2)/nobs)

--- find variance, standard deviation ---
xvar=xsumsqr/(nobss-1.0)
xstdev=sqrt(xvar)

--- write out this mess for individual pass ---
write (25,9992) passnum, xvar, xstdev, xmin, xmax
9992 format (16,5f14.5)

--- if the variance of the pass is below the user
defined limit then race back on to main program ---
if (xvar .le. varmax) then
  eight=7777
  return
endif
gftype=gftype
gftcnt=gftcnt+1
15 do 15 i=1,nobs
  do 15 j=1,27
    etrdta(i,j)=data(i,j)
15 continue
subwinlen=0
--- use a moving average fit ---
if (gftype .eq. 1) then
  itercnt=itercnt+1
  winlen=wwinlen1
  incwen=linwen1
  inum=1
  if (subwinlen .gt. 0) then
    if (incwen .gt. 1) then
      incwen=incwen-subwinlen
      go to 25
    elseif (incwen .lt. winlen) then
      winlen=winlen-subwinlen
    end
    if (incwen .lt. 2) stop 1110
  endif
25 continue
dividexreal(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

---
xwlnlen=real(nobs)/real(idivide)
iwinadd-int(real(xwlnlen))/real(idivide)
winlen=int(xwlnlen)-inowel
istrwlen=winlen
iadd=0
if (iwinadd.gt.0 .and. iadd.lt.iwinadd) then

iadd=iadd+1
endif
i=1
j=1
if (i1.gt.nobs) then

i=nobs/2
j=1
endif
avgdat=0.0
do j=1,11

avgdat=avgdat+data(j, var)
enddo
do 20 j=1,27

movdata(inum, j)=data(1, j)
continue
movdata(inum, var)=avgdat/(real(11))
if (i7.eq.1) go to 100
30 continue
wlnlen=itrwlen
if (iwinadd.gt.0 .and. iadd.lt.iwinadd) then

iadd=iadd+1
endif
i=itrwlen-1/2
j=0
if (j1.gt.nobs) then

i=nobs/2
j=1
endif
avgdat=0.0
do j=1,11

avgdat=avgdat+data(j, var)
enddo
do 20 j=1,27

movdata(inum, j)=data(1, j)
continue
movdata(inum, var)=avgdat/(13-i2+i)
go to 30
100 continue
inum=inum+1
avgdat=0.0
do j=nobs-11+1, nobs

avgdat=avgdat+data(j, var)
enddo
do 150 j=1,27

movdata(inum, j)=data(nobs, j)
170 continue
movdata(inum, var)=avgdat/(13-i2+i)

movdata(inum, var)=avgdat/(real(11))
C----------------------------------------------- or use the minimum and maximum values
C
C else if (gftype .eq. 2) then
itercnt=itercnt+1
inowel=linowel2
winlen=winlen2
winlenb=winlen2b
if (winlena.gt.nobs .or. winlenb.gt.nobs .or.

> inowel.gt.nobs .or. (winlena+1).ge.(nobs-winlenb)) then
do 1=1,27

movdata(1,1)=data(1,1)

movdata(2,1)=data(nobs,1)
enddo
avgdat=0.0
do 1=1, nobs/2

avgdat=avgdat+data(1, var)
enddo
avgdat=avgdat/real(nobs/2)
movdata(1, var)=avgdat
avgdat=0.0
do 1=(nobs/2)+1, nobs

avgdat=avgdat+data(1, var)
enddo
avgdat=avgdat/real(nobs-(nobs/2))
movdata(2, var)=avgdat
iendcnt=iendcnt+1
inum=2
go to 700
endif
avdat=0.0
500 do 1=1,nlenb
510 mavdata(1,:)=data(1,:)
mavdata(1,:)=avdat
520 continue
530 mavdatmin=mavdatmax=data(1,:)
avdatmin=mavdatmin/real(nlenb)
540 continue
550 endif
avdat=0.0
594 do 1=1,nlenb+1
596 mavdata(1,:)=data(1,:)
mavdata(1,:)=avdat
600 continue
609 continue
620 continue
629 else (dndk .eq. 'dusk') then
639 if (mavdata(2+i3,1) .le. (mavdata (i+i3,1) + (real (incwen) *0.33) ) ) then
649 do 420 j=2,3
d650 movdata(j+i3,1)=movdata(j+i3,1)
inum=inum-1
660 enddo
679 endwhile
689 else (dndk .eq. 'dusk') then
699 endif
709 endif
719 endif
729 endif
739 endif
749 endif
Do 630 i=1,27
movdata(inum-1+13,i)=movdata(inum+13,i)
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
elseif (dndk .eq. 'dawn') then
do 404 i=1,inum
x(i)=movdata(i,1)
y(i)=movdata(i,2)
endif
continue
endif
oneslope=(y(2)-y(1))/(x(2)-x(1))
twoeslope=(y(inum)-y(inum-1))/(x(inum-1)-x(inum-1))
call spline (x,y,inum,oneslope,twoeslope,y2)
do 402 i=1,nobs
xint-data(i,1)
if (dndk .eq. 'dawn') xint-data(i,1)
call spline (x,y,1,xint,yint)
do 403 j=1,27
intpdata(j,1)=data(i,j)
if (dndk .eq. 'dawn') intpdata(j,1)=data(i,j)
continue
intpdata(1,2)=yint
continue
if (dndk .eq. 'dawn') then
do 405 i=1,nobs
ii=nobs-i+1
do 405 j=1,27
sdata(i,j)=intpdata(ii,j)
do 406 j=1,27
intpdata(i,j)=sdata(i,j)
continue
do 440 j=1,nobs
xsum=xsum+strdata(j,var)
ysum=ysum+intpdata(j,var)
ysum=ysum+intpdata(j,var)**2
ysumsqr=ysumsqr+intpdata(j,var)**2
sumxy=sumxy+strdata(j,var)*intpdata(j,var)
do 440 continue
sumprod=sumxy-(xsum*ysum)/nobs
covarxy=sumprod/(nobs**2/nobs)
xcsumsqr=xcsumsqr-(xsum**2)/nobs
ycsamsqr=ycsamsqr-(ysum**2)/nobs
continue
find corrected sum of products, covariance
and corrected sum of squares (x) (y)
c
xvar=xcsmsqr/(nobs-1.0)
yvar=ycsamsqr/(nobs-1.0)
xstddev=sqrt(xvar)
ystddev=sqrt(yvar)
c
find correlation coefficient by Davis method
corrDxy=covarxy/(xstddev*ystddev)
c
if (corrDxy .lt. mincc .and. gftype .eq. 2) then
write ('*',*),passnum,' pass below cc limit',corrDxy
gftype=1
iswitch=iswitch+1
   do 450 i=1,nobs
      do 450 j=1,27
         data(i,j)=strdata(i,j)
      continue
   endif
   if (corrDxy .lt. mincc .and. gftype .eq. 1) then
      write (*,*) passnum,' pass below cc limit',corrDxy
      llowcnt=llowcnt+1
      go to 999
   endif
   subwinlen=subwinlen+1
   do 460 i=1,nobs
      do 460 j=1,27
         data(i,j)=strdata(i,j)
      continue
   endif
   if (corrDxy .lt. mincc .and. gftype .eq. 1) then
      write (*,*) passnum,' pass below cc limit',corrDxy
      ilowcnt=ilowcnt+1
      go to 999
   endif
   subwinlen=subwinlen+1
   do 460 i=1,nobs
      do 460 j=1,27
         data(i,j)=strdata(i,j)
      continue
   endif
   if (corrDxy .lt. mincc .and. gftype .eq. 1) then
      write (*,*) passnum,' pass below cc limit',corrDxy
      ilowcnt=ilowcnt+1
      go to 999
   endif
   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
   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
   INPUT2 ----DESPIKE-----> 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 NANOKEAS):
   IF PROGRAM DESPIKE HAS DETERMINED OBSERVATION N TO BE
   A GOOD POINT, IT THEN SEES 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) = RESID2(U)
   15 s1=(data(i+1, var)-data(i, var))
      s2=(data(i+2, var)-data(i, var))
      s3=(data(i+3, var)-data(i, var))
      s4=(data(i+4, var)-data(i, var))
      xsl=abs(s1+s2+s3+s4)/4.0
      s2=(data(i+2, var)-data(i+1, var))
IF(ABS(SLI).GT. (3.0*XSL).OR.ABS(SLI).GT.(ABS(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-I,NPTS-2
SL2=(DATA(J+1, var) -DATA(J+2, var))
IF(IC(J+1).EQ.0) THEN
IC(J+1)=0
ENDIF
IF(IC(J+2).EQ.0) THEN
IC(J+2)=0
ENDIF
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
SL1=ABS(DATA(K, var) -DATA(NPTS-1, var))
SL2=ABS(DATA(NPTS-1, var)-DATA(NPTS, var))
SL3=ABS(DATA(NPTS, var) -DATA(K, var))
IF(IC(NPTS-1).EQ.0) THEN
IF(SL1.(GT. (3.0*SL2)) IC(NPTS)=0
ENDIF
IF(IC(NPTS-1).EQ.1) THEN
IF(SL2.GT. (3.0*SL3)) IC(NPTS)=0
ENDIF
ENDIF
NORS=0
DO 30 I-1,NPTS
IF(IC(I).EQ.1) NORS=NORS+1
30 CONTINUE
WRITE (6,* IDATA(1,1), NORS
outnum=0
DO 35 I-1,NPTS
IF(IC(I).EQ.1) THEN
outnum=outnum+1
32 CONTINUE
35 CONTINUE
return
END
C
C subroutine track (nf,innum, noc,nocgf)
common /trax/ x(400),y(400)
C............................ subroutine description
C this subroutine calculates the lat lon coordinates of each point to be plotted in tplot for a map view of the footprint of the pass
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, NPGF, (X(J),Y(J),J=1,NOP)
return
C
C subroutine sqrfit (innum, x, ifitvar)
real core, xmag (400), intpdata (400,
C................................ subroutine description
c sqrfit fits the core field values to the observed data in a least squares manner. that is, the subroutine finds a value of x that is multiplied by all core field values

B-45
in a pass so that the core field model matches the observed values closer.

x=0.0
cmean=0.0
fmean=0.0

do i=1,innum
   core(i)=intpdata(i,16)
cmean=cmean+core(i)
xmag(i)=intpdata(i,1iftvar)
fmean=fmean+xmag(i)
endo
c
   cmean=cmean/real(innum)
fmean=fmean/real(innum)

do i=1,innum
   xmag(i)=xmag(i)-fmean
core(i)=core(i)-cmean
endo
c
   ctc=0.0
600   ctc=(core(i)*core(i))+ctc
c
   ctcinv=1.0/ctc
c
   ctf=0.0
700   ctf=(core(i)*xmag(i))+ctf
x=ctcinv*ctf
c
800   intpdata(i,1iftvar)=xmag(i)-(core(i)*x)
c
999   continue
return
end
program move trunc
integer xrow, xcol, zero, eight, xpassno, ypassno, ycol,
xrow, xcol, yrow, ycol, ypass, xrow, yrow,
string, nob, dndh, tcount, palcnt, nowant (4000), nocnt,
palcnt, jstop, minobs, passoam, totpas, global,
outto, noppas, type, nocnt
real xmean, ymean, x3data (400, 3), y3data (400, 3), ymean,
xdata (400), ydata (400), xdata (400, 4), ydata (400, 4),
x3mean, xmean, ymean, xrow, yrow,
x3row, x3col, x3pass, y3row, y3col, y3pass,
x3data (400, 3), xdata (400, 3), ydata (400), y(400)
character* 80 filename
common /tplot/ x(400), y(400)
common /nope/ nowant (4000), nocnt
common /runs, at/ xdata (400, 4), ydata (400, 4)
common /rowcol/ x3row, y3row, xrow, yrow, x3col, y3col, xcol, ycol,
x3mean, y3mean, xmean, ymean, x3pass, y3pass,
x3data (400, 3), xdata (400, 3), ydata (400)
character* 80 filename
common /tplot/ x(400), y(400)
common /nope/ nowant (4000), nocnt
common /runs, at/ xdata (400, 4), ydata (400, 4)
common /rowcol/ x3row, y3row, xrow, yrow, x3col, y3col, xcol, ycol,
x3mean, y3mean, xmean, ymean, x3pass, y3pass,
x3data (400, 3), xdata (400, 3), ydata (400)
character* 80 filename

c------

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
e xtract 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)
read (10, file=filename, status='old', form='unformatted')
write (*, *) 'INPUT FILE X OF MAGNETIC VARIABLES'
read (11, file=filename, status='old', form='unformatted')
write (*, *) '0 IF THE DATA IS GLOBAL OR POLAR'
write (*, *) '1 IF THE DATA DOES NOT INCLUDE ALL LONGITUDES'
write (*, *) global
if (global .eq. 0) then
write (*, *) 'NOTE: FILE Y WILL HAVE THE FIRST PASS MOVED'
write (*, *) 'TO THE BOTTOM OF THE FILE'
write (*, *)
elseif (global .eq. 1) then
write (*, *) 'NOTE: OUTPUT FILE X WILL NOT INCLUDE THE LAST PASS'
write (*, *) 'FILE Y WILL NOT INCLUDE THE FIRST PASS'
write (*, *)
endif
write (*, *) 'OUTPUT FILE X OF TRUNCATED LAT-LONG-RAD DATA'
read (*, 9990) filename
read (20, file=filename, form='unformatted')
write (*, *) 'OUTPUT FILE X OF TRUNCATED MAGNETIC VARIABLES'
read (21, file=filename, form='unformatted')
write (*, *) 'OUTPUT FILE Y OF TRUNCATED LAT-LONG-RAD DATA'
read (22, file=filename, form='unformatted')
write (*, *) 'OUTPUT FILE Y OF TRUNCATED MAGNETIC VARIABLES'
read (23, file=filename, form='unformatted')
write (*, *) 'OUTPUT FILE OF TRACK PROFILES TO BE RUN IN TIPLOT'
read (24, file=filename, form='unformatted')
write (*, *) 'OUTPUT FILE OF STATISTICS'
read (25, file=filename, form='formatted')
write (*, *) '0 IF THESE ARE DUSK DATA SETS'
write (*, *) '1 IF THESE ARE DAWN DATA SETS'
read (*, *) dndk
write (*, *) '0 IF ALL PASSES ARE WANTED'
write (*, *) '1 IF SOME PASSES SHOULD BE REMOVED'
read (*, *) nocnt
if (nocnt .eq. 1) then
write (*, *) 'INPUT FILE OF PASS NUMBERS NOT WANTED'
read (*, 9990) filename
do 10 i=1, 4000
read (26, *, end=15) nowant (1)
continue
10 continue
nocnt = nocnt - 1
endif
write (*, *) 'MINIMUM NUMBER OF OBSERVATIONS FOR EACH PASS'
read (*, *) minobs
write (*, *) 'TYPE OF GAP FINDER (2)'
write (*, *) '1 FOR ONLY FINDING GAPS'
write (*, *) '2 FOR USING THE MINIMUM OBSERVATIONS'

B-48
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 ',
> 'CONARKY XSTDDEV YSTDDEV',
write (25,*),
' XINTCPT YINTCPT'
paircnt=0
paircnt1=0
tcount=0
strcnt=0
jstop=0
passrem=0
totpass=0

30 continue
read (10, end=90) y3row, y3col, zero, y3mean, y3pass, eight
do 35 i=1,y3row
read (10) (y3data(i,ii),i=1,y3col)
35 continue
read (11) yrow, ycol, zero, ymean,ypassno, eight
do 45 i=1,yrow
read (11) ydata(i)
45 continue
totpass=totpass+1
if (yrow .lt. minobs .or. y3row .lt. minobs) then
write (*,*)y3pass,ypassno,' PASS REMOVED: ROWS=',y3row,yrow
passrem=passrem+1
go to 30
endif
if (nocnt .eq. 0) go to 55
do 50 i=1,nocnt
if (ypassno .eq. nowant(i)) then
write (*,*)ypassno, y3pass,' PASS REMOVED'
passrem=passrem+1
go to 30
endif
50 continue
55 continue
go to 95

-- this little jump around is used to get the last and first passes of global datasets truncated
90 jstop=1
if (global .eq. 1) go to 999
95 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, paircnt1, 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)
write out the truncated lengths of passes

write (20) nobs,x3col,zero,xmean,xpassno,eight
write (21) nobs,xcol,zero,ymean,ypassno,eight
write (22) nobs,ycol,zero,ybmean,ypassno,eight
write (23)

do 200 j=1,nobs
write (20) xadata(J,1), (xadeta(J,i),i=3,4)
write (21) xadata(J,2)
write (22)
write (23)
continue

call track (nobs,noc)
WRITE (24) NOC, NObs, (X (J),Y(J), J-I,NObs)
if (paircnt .gt. 0) tcount=tcount+1
paircnt=paircnt+paircnt
if (jstop .eq. 1) go to 999
if (outto) 30,999,999
continue
if (global .eq. I) strcnt-strcnt-1
write (*,*) 'corrected',paircnt, 'pairs of latitudes in' 
write (*,*)
write (*,*) 'total passes read - ',tcount
close

subroutine truncate (xrow, yrow, dndk, mlnrow, stocount, nopass, 
> xpassno, ypassno)
integer xrow, yrow, stocount, rowli, rowinc, mlnrow, 
> nopass, xpassno, ypassno
real xdata(400,3),xldata(400), x3data(400,3),xldeta(400), 
> xdata, bdata, diffab, abss, xdata(400,4), ydata(400)
common /trunstat/ xadata(400,4), yadata(400,4)
common /nope/ nowant(4000), nocnt
common x3data(400,3), xldata(400), y3data(400,3), yldeta(400)

--- 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 must be
interpolated to every 0.33 degrees.
do 70 j=1,xrow
xdata(J,1)=xdata(J,1)
xdata(J,2)=xdata(J,2)
xdata(J,3)=xdata(J,3)
xdata(J,4)=xdata(J,4)
continue
70 do 75 j=1,yrow
ydata(J,1)=ydata(J,1)
ydata(J,2)=ydata(J,2)
ydata(J,3)=ydata(J,3)
ydata(J,4)=ydata(J,4)
continue
75 continue
80 continue
stocount=0
jj=1
rowli=xrow
rowinc=yrow
--- loops from 90 to 200 increment through the
--- two input passes and truncate the lengths
to the same length
 continu
nopass=xpassno
nowant(noint+1)=xpassno
write(*,*)'xrows (ii)úmeros yrows (inc)úmeros',rovinc
write(*,*)'rerunning to remove x pass number =',xpassno
write(*,*)'xrow =',xrow,' yrow =',yrow
return
endif
endf
minrow=min(rowli,rowinc)
c write(*,*) rowli,rowinc,minrow
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 li=1,minrow
        xdata(ll,kk)=xdata(ll,1,kk)
ydata(ll,kk)=ydata(ll,1,kk)
c write(*,*) xdata(ll),ydata(ll)
  continue
110 continue
return
endif

------------------------------------------------------------------
if pass a no match 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
  go to 90
  and
c subroutine statistics (minrow,xpassno,ypassno,icmean,ycmean)
integer minrow,lobs,xpassno,ypassno
real xdata(400,4),ydata(400,4),nobs
common /trunstat/ xdata(400,4),ydata(400,4)
c the statistical calculations using two
references:
c 1) Davis, Statistics and Data Analysis in
Geology, 2nd ed., 1986 pp. 41
c 2) Young, Statistical Treatment of Experi-
mental Data, 1962, McGraw Hill, 115-132

- loops that sum x, x^2, y, y^2 and calculate new truncate means

```
    nobs=minrow
    nobss=float(nobs)
    xsum=0.0
    xsumgr=0.0
    ysum=0.0
    ysumgr=0.0
    sumxy=0.0
    do 240 j=1,nobs
      xsum=xsum+xadata(j,2)
      xsumgr=xsumgr+(xadata(j,2))**2
      ysum=ysum+ybdata(j,2)
      ysumgr=ysumgr+(ybdata(j,2))**2
      sumxy=sumxy+(xadata(j,2)*ybdata(j,2))
  240
```

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

```
xmean=xsum/nobss
    ybmean=ysum/nobss
    sumprod=sumxy-((xsum*ysum)/nobss)
    covarxy=sumprod/(nobss-1.0)
```

- find variance, standard deviation for x and y

```
xvar=xsumsqr/(nobss-1.0)
    yvar=ysumsqr/(nobss-1.0)
    xstdev=sqrt(xvar)
    ystdev=sqrt(yvar)
```

- find correlation coefficient by Davis method

```
corrDxy=covarxy/(xstdev*ystdev)
```

- find slopes, intercepts and correlation coefficient by Young method

```
xslope=((nobss*xsumy)-(xsum*ysum))/((nobss*xsumsqr)-xsum**2)
ylope=((nobss*xsumy)-(xsum*ysum))/((nobss*ysumsqr)-ystum**2)
xintcpt=((ysum*xsumsqr)-(sumxy*xsum))/((nobss*xsumsqr}-xsum**2)
yintcpt=((xsum*ysumsqr)-(sumxy*ystum))/((nobss*ysumsqr)-ystum**2)
```

- write this mess for individual pass and overlapping lengths of passes

```
write (25,9992)xpassno,ypassno, xvar,yvar,xstdev,ystdev,
    xmean,ybmean
mixed format ('FOR OVERLAPPING LENGTHS X-','i5',' Y-','i5','/',
    ' X VARIANCE-','f9.3',' Y VARIANCE-','f9.3',' X STDEV-','f9.3',
    ' Y MEAN-','f9.3', ' Y MEAN-','f9.3')
```

```
read (25,9993) covarxy,xslope,ylntcpt,corrYxy
```

```
xslope=((nobss*xsumy)-(xsum*ysum))/((nobss*xsumsqr)-xsum**2)
ylope=((nobss*xsumy)-(xsum*ysum))/((nobss*ysumsqr)-ystum**2)
xintcpt=((ysum*xsumsqr)-(sumxy*xsum))/((nobss*xsumsqr}-xsum**2)
yintcpt=((xsum*ysumsqr)-(sumxy*ystum))/((nobss*ysumsqr)-ystum**2)
```

```
write (25,9995) xpassno,ypassno,corrDxy,corrYxy, xvar,yvar,
    covarxy,xstdev,ystdev
9995 format (215,7(f10.4))
write (25,9996) xmean,ymean,xslope,ylope,xintcpt,yintcpt
9996 format (10x,6(f10.4))
return
end
```

```
   subroutine movtrun (into, outto, jstop)
   integer into, outto, jstop,
   xpass, ypassno, jetop,
   y3row, yrow, ycol, ycol, ypass, ypassno,
   x3row, xrow, xcol, xcol, xpass, xpassno,
   real xdata(400,3), xdata(400), ydata(400),
   y3mean, ymean, yavey3(400,3), yavey3(400,3), ymean,
   x3mean, xmean, x3mean, xmean, storex3(400,3), ystorex3(400,3),
   x3mean, xmean,
   common xdata(400,3), xdata(400), ydata(400,3), ydata(400),
   common y3rowcol, x3row, y3row, xrow, yrow, xcol, ycol, xcol, ycol,
   x3mean, xmean,
   x3mean, xmean
   common /rowcol, x3row, y3row, xrow, yrow, xcol, ycol, xcol, ycol,
   x3mean, xmean,
```

```
   subroutine description
   this subroutine stores one pass so that the offset will occur.
   if (jstop) 10,10,350
   10 if (into) 40,85,290
```

B-52
40 continue
do 50 i=1,y3row
  do 55 il=1,y3col
    savey3(i,i)=y3data(i,i)
  continue
savey1(i)=ydata(i)
50 continue
sy3row=y3row
syrow=yrow
sy3col=y3col
sycol=col
sy3mean=ymean
sy3pass=ypass
sy3passno=ypassno
55 continue
saveyl(i)=yldata(i)
50 continue
sy3row=y3row
syrow=yrow
sy3col=y3col
sycol=col
sy3mean=ymean
sy3pass=ypass
sy3passno=ypassno
70 continue
x3data(i,i)=y3data(i,i)
70 continue
x3row=x3row
xrow=xrow
x3col=x3col
xcol=col
x3mean=xmean
x3pass=xpass
x3passno=xpassno
85 continue
do 90 i=1,y3row
  do 95 il=1,y3col
    storex3(i,i)=y3data(i,i)
  continue
storex1(i)=ydata(i)
90 continue
sx3row=x3row
sxrow=xrow
sx3col=x3col
sxcol=col
sx3mean=xmean
sx3pass=xpass
sx3passno=xpassno
if (xrow.ne.x3row .or. x3pass .ne. xpass)
  then
    write (*,*), 'WACKO, TRA-IA-IA, JOLLY-GOOD, NO MATCH BETWEEN'
    write (*,*), ROWS OR PASSNOS : X', xrow, x3row, x3pass, xpass
    stop
endif
85 continue
outto = -1
return
90 continue
x3row=x3row
xrow=xrow
x3col=x3col
xcol=col
x3mean=xmean
x3pass=x3pass
x3passno=x3passno
do 300 il=1,x3col
  do 305 il=1,x3col
    storex3(i,i)=y3data(i,i)
  continue
x3data(i,i)=storex3(i,i)
300 continue
outto = -1
return
305 continue
y3data(i,i)=savey3(i,i)
305 continue
ydata(i,i)=savey1(i,i)
```fortran
360  continue
     y3row=y3row
     y3col=y3col
     y3pass=y3pass
     return
end

subroutine findgap (global,dndk,mlnobs,type)
  integer zero,eight,minxyrow,xrow,yrow,type,
     y3row,y3col,y3pass,xcnt,ycnt,strmincnt,
     y3pass,nowant,global,both,ncnt2,
     no_ss,strnocnt,allcnt
  real y3data(400,3),y3mean,abs,y3local
  common /no/e/n,nowant(4000),nocnt

 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 re-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 last and 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
     allcnt=allcnt+1
     continue
     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
50  continue

common /no/e/n,nowant(4000),nocnt
---------- subroutine description
------------------------------ depending on the type of gap

finder chosen, the program will proceed as appropriate
70  continue
    jstop=0
--------------------- xcnt and ycnt represent the two adjacent passes
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 (ycnt1 .gt. allcnt) then
         ycnt=ycnt1
         go to 195
      elseif (jstop .eq. 1) then
         ycnt=ycnt1
         go to 100
      endif
      do 105 j=ycnt,allcnt-1
      do 105 j=1,4
         alldata(j,j)=alldata(j+1,j)
      105 continue
      allcnt=allcnt-1
      go to 100
   endif
110 continue

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))
      minxyrow=min(xrow,yrow)
      nopass=xpassno
      if (minxyrow .eq. yrow) nopass=ypassno
      if (xrow .eq. yrow) both=1
      nocnt2=nocnt
   c--------------------------------------if this is a dusk pass then will count from
      -90.0 lat degrees toward the equator
      if (dndk .eq. 0) then
         if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
            if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
               nocnt=nocnt1
               nowant(nocnt)=nopass
            endif
         else if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
            if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
               nocnt=nocnt1
               nowant(nocnt)=nopass
            endif
         endif
      endif
   c--------------------------------------if this is a dawn pass then will count from
      c that is decreasing independent variable
      c
      elseif (dndk .eq. 1) then
         if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
            if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
               nocnt=nocnt1
               nowant(nocnt)=nopass
            endif
         else if (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
            if (alldata(xcnt,4) .lt. alldata(ycnt,4)) then
               nocnt=nocnt1
               nowant(nocnt)=nopass
            endif
         endif
      endif
      endif
   if (nocnt .ge. strnocnt) go to 999
   if (strnocnt .lt. nocnt) then
      strnocnt=nocnt
      go to 70
   endif
   endif
190 continue
   xcnt=ycnt
   ycnt=ycnt1
   if (jstop .eq. 1) go to 200
195 if (ycnt1 .gt. allcnt) then
   if (global .eq. 1) go to 200
   ycnt1=1
   jstop=1
   go to 100
200 continue
   if (nocnt .eq. strnocnt) go to 999
   if (strnocnt .lt. nocnt) then
      strnocnt=nocnt
      go to 70
   endif
400 continue
mincnt=0
strmincnt=mincnt
c 470 continue
jstop=0

c-------------------------- xcnt and ycnt, see notes above
xcnt=1
ycnt=2
if (int(alldata(xcnt,2)) .lt. minobs) then
do 480 j=xcnt,allcnt-1
do 480 j=1,4
alldata(j,j)=alldata(jj+1,j)
490 continue
allcnt=allcnt-1
go to 470
endif

c 500 continue
if (int(alldata(ycnt,2)) .lt. minobs) then
if (ycnt=1 .gt. allcnt) then
ycnt=ycnt+1
go to 595
elseif (jstop .eq. 1) then
ycnt=ycnt+1
go to 500
endif
do 505 j=ycnt,allcnt-1
do 505 j=1,4
alldata(j,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 90.0 let degrees toward the equator
if (dndk .eq. 0) then
if (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
mincnt=mincnt+1
elseif (alldata(xcnt,3) .lt. alldata(ycnt,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 (alldata(xcnt,3) .lt. alldata(ycnt,3)) then
mincnt=mincnt+1
elseif (alldata(xcnt,3) .gt. alldata(ycnt,3)) then
mincnt=mincnt+1
endif
endif

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

if (strmincnt) go to 999
if (strmincnt .eq. strmincnt) go to 999
if (strmincnt .lt. strmincnt) then
strmincnt=strmincnt
go to 470
endif

c 999 continue
write (*,*) 'total passes read = ',totpass
if (nocnt .gt. 0) then
write (*,*) 'will remove the following passes from processing'
do 1010 i=1,nocnt
write (*,*) nowant(i),i
1010 continue
endif

if (type .eq. 2) write (*,*) 'new minimum observation cutoff',
> ' is = ',minobs
rewind (10)

B-56
subroutine track (nop, noc)
integer nop, noc
real radfac, x(400), y(400), xadata(400, 4), ybdata(400, 4)
common /plot/ x(400), y(400)
common /trunatat/ xdata(400, 4), ybdata(400, 4)

--- subroutine description

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,
> prcnt,delta,cuthl,cutlo,lag,mincc,short,long,
> maxccin,clag,strdata(4096)
integer xpassno,ypassno,zero,eight,fill,xcol,xrow,ycol,yrow,
> trnsf,lhb,co,transb,npass,imean,cmwind,seven,
> nwind,numfile,nout,yout,match,gfclt,xnobs,ynobs
complex xdata(4096),ydata(4096)
common /rowcol/ xno,yno
common /fftfft/ nobs,prcnt,imean,fill
common /lhbfft/ delta,cuthl,cutlo,lag,npass,nwind
common /ccfft/ mincc,maxcc,match,minccin,maxccin,cmwind,clag
common /reals/ xdata,ydata
common /comps/ xdata,ydata 

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

fourierld 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 and a correlation coefficient
filter which zeros out wavenumbers according to correlation
coefficients both the bandpass filter and the correlation
coefficient filter provide the user with several windowing options
(as well as no windowing option) to smooth wavenumbers prior
to inverse transformation. both filters use the same subroutine
to window in the TIME (real number data) domain. with respect to
run time considerations: if many different datasets are submitted
at the same time to the program, it will still calculate the same
BANDPASS windowing function for each dataset every time it
counters a new dataset. since this windowing function need only
calculate once, it causes the program to do needless work. i
hope to soon remedy this little time consuming "bug". big note of
cautions: the windowing function will change with each new dataset
for the CORRELATION FILTER and therefore, leave it alone!!

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

NOTE: the only data 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 differing 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 those wonderful zeros. and for those of you who are
really into this, you can now fold out a percentage of the
edge of your data, smooth the folded out part to zero, fft,
filter, and ift such that edge effects are minimized.

9 jul 92:
major revisions changing code from two-dimensionally based
 ffts to one-dimensional ffts. revisions include removal of
subroutines transpo and store. major changes to subroutines
fft2d (which is now known as fft1d), datwnd, bndpass 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 (*,9988)
format ('USE THE FOLLOWING ORDER FOR INPUT FILE'/,
> 'IF VARIABLE DOES NOT APPLY INPUT ANY BOGUS NUMBER'/,
> 'numfile/',
> 'lhb cc/',
> 'nobs fold prcnt imean/',
> 'delta short long npass/',
> 'nwind xlag/'
> 'mincc maxcc match mincin maxccin cmwind xlag/'
> 'sub/')
write (*,**) 'INPUT FILE OF VARIABLES'
read (*,9990) filename
open (22, file=filename, form='formatted',status='old')
go to 100


B-59
elself (file .eq. 1) then
go to 50
endif

continue
write (*, 9989)
9989 format ('1 IF YOU HAVE ONLY ONE FILE TO BE FOURIERED',/,
> '2 IF YOU HAVE TWO FILES TO BE COMPARED',/,
> '3 FOR ONLY LOW-HIGH-BAND FILTER THE DATA',/,
> '4 FOR LOW-HIGH-BAND THEN C-C FILTER THE DATA',/,
> '5 FOR LOW-HIGH-BAND THEN C-C FILTER THE DATA',/,
> '6 DO NOT FILTER THE DATA',/,
> '1 FOR ONLY CORRELATION COEFFICIENT FILTER THE DATA',/,
> '2 DO NOT C-C FILTER THE DATA',/,
'choose 2 if 5 or less was chosen above',/,
> 'lhb cc' )
read (*,*) lhb, cc
write (*, 9992)
9992 format (*THE FOLLOWING REFERS TO FFT AND IFFT',/,
> 'NUMBER OF OBSERVATIONS FOR FFT ARRAY',/,
> 'AT A POWER OF 2: (256) (2 16 32 64 128 256 etc)',/,
> 'PERCENT OF DATA TO BE FOLDED OUT (0.1 TO 99.9)',/,
> 'PERCENT OF EACH EDGE OF INPUT ARRAY OR FOLDED',/,
> 'OUT ARRAY TO BE SMOOTHED TO ZERO (0.1 TO 49.9)',//,
> '0 DO NOT ADD MEAN TO IFFT DATA',/,
> '1 ADD MEAN OF INPUT DATA TO OUTPUT IFFT DATA',/,
' hobs fold prcnt imean' )
read (*,* }nobs, fold, prcnt, Imean
endif
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',/,
' cmind..... TYPE OF WINDOW TO APPLY',/,
> '0 GIVES NO WINDOW',/,
> '1 FOR PROGRAM TO APPLY WINDOW',/,
> '2 FOR PROGRAM TO APPLY WINDOW',/,
> '3 FOR PROGRAM TO APPLY WINDOW',/,
> '4 FOR PROGRAM TO APPLY WINDOW',/,
' cmind=xlag')
write (*, 9994)
9994 format (' mwind..... TYPE OF WINDOW TO APPLY',/,
> '0 GIVES NO WINDOW',/,
> '1 RECTANGULAR WINDOW',/,
> '2 BARLETT WINDOW (TRIANGULAR)',/,
> '3 HAMMING-TUKEY WINDOW',/,
> '4 PARZEN WINDOW',/,
> ' xlag.....SMOOTHING PARAMETER FOR WINDOWING IDEAL',/,
> ' FILTER IN SPATIAL DOMAIN (95.0) (is disabled if',/,
> 'no window was chosen above)',//,
' mwind=xlag')
endif
endif
I RECTANGULAR WINDOW
- 2 BARTLETT WINDOW (TRIANGULAR)
- 3 HAMMING-TUKEY WINDOW
- 4 PARZEN WINDOW
"CXLAG.....SMOOTHING PARAMETER FOR WINDOWING IDEAL"
"FILTER IN SPATIAL DOMAIN (95.0) (is disabled if"
"no window was chosen above)."
"mincc maxcc match mincin maxcin cnwind cxlag"
read (*,*) mincc, maxcc, match, mincin, maxcin, cnwind, cxlag
\nodef
if (numfile .eq. 2) then
write (*,*) 'AND FINALLY: 0 DO NOT SUBTRACT FILE1-FILE1 (use 0)'
write (*,*) ' 1 TO WRITE A FILE OF TIME DOMAIN SUBTRACTION'
write (*,*) 'OF FILE 3 = INPUT FILE 1 - OUTPUT FILE 1'
read (*,*) isub
\@iawulauikeisi eual 1 if you want to subtract
\the filtered portions of file1 from
\the input of file1. this option is not
generally used.
endif
\go to 200
100 continue
read (22,*) numfile
read (22,*) lhb, cc
read (22,*) nobs, fold, prcnt, imean
read (22,*) delta, short, long, npass
read (22,*) cnwind, xlag
read (22,*) mincc, maxcc, match, mincin, maxcin, cnwind, cxlag
read (22,*) isub
200 continue
write (*,*) 'all input files must have a header with:'
write (*,*) 'row, column,zero,mean,pass number,eight'
write (*,*) 'zero... can be bogus but row and col are necessary'
write (*,*) 'INPUT FILE 1 (do not put guide function file here)'
read (*,9990) filename
9990 format (a80)
open (10, file=filename,status='old', form='unformatted')
if (numfile .eq. 2) then
write (*,*) 'INPUT FILE 2 (or the guide function file)'
read (*,9990) filename
endif
write (*,*) 'OUTPUT OF FILE 1'
read (*,9990) filename
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
write (*,*) 'OUTPUT OF FILE 2'
read (*,9990) filename
open (11, file=filename, status='old', form='unformatted')
endif
write (*,*) 'OUTPUT FILE OF STATISTICS AND INFORMATION'
read (*,9990) filename
open (21, file=filename, form='unformatted')
if (isub .eq. 1) then
write (*,*) 'OUTPUT FILE 3 OF SUBTRACTION'
read (*,9990) filename
endif
write (*,*) 'OUTPUT OF FILE 2'
read (*,9990) filename
open (22, file=filename, form='formatted')
if (isub .eq. 1) then
write (*,*) 'OUTPUT FILE 3 OF SUBTRACTION'
read (*,9990) filename
endif
\if (lhb .lt. 6)
cuthi=1.0/short
cutlo=1.0/long
RCUTLO=99999.99
IF(CUTLO .GE. 0.0000001) RCUTLO= 1.0/CUTLO
RCUTHI=1.0/CUTHI
WAVELEN=2.0/Delta
FNO=1.0/WAVELEN
WRITE (25,9987) FNO, WAVELEN, CUTLO, RCUTLO, CUTHI, RCUTHI
9987 FORMAT('INQUIST WAVENUMBER = ',F10.5, ' CYCLES PER DATA INTERVAL/',
'INQUIST WAVELENGTH = ',F10.5, ' LENGTH INTERVALS/',
'LOW WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
'CYCLES PER DATA INTERVAL ',F15.5,
'WAVELENGTH EQUIVALENT/',
'HIGH WAVE# CUTOFF OF IDEAL FILTER = ',F10.5,
'CYCLES PER DATA INTERVAL ',F15.5,
'WAVELENGTH EQUIVALENT/',)
endif
\gfont=0
210 continue
read (10, end=999) xrow, xcol, zero, xmean, xpassno, eight
xrows=xrow
do i=xrow

read (10) xdata(1)
enddo
230 if (numfile .eq. 2) then
   read (11, end=999) yrow, ycol, zero, ymean, ypassno, seven
   if (seven .eq. 7777) go to 500
   yobs=yrow
   do 1-i, yrow
      read (11) ydata(i)
   enddo
endif
c if (isub .eq. 1) then
   do 1-i, xrow
      strxdata(i) = xdata(i)
   enddo
c xmean=0.0
   call forwardft (1, xmean, xpassno)
   ymean=0.0
   if (numfile .eq. 2) call forwardft (2, ymean, ypassno)
c if (lhb .le. 3) then
      call filter (1)
      if (numfile .eq. 2) call filter (2)
endif
c if (lhb .ge. 2 and. lhb .le. 5 or. cc .eq. 1) then
      call correlate (xpassno, ypassno)
endif
c if (lhb .ge. 3 and. lhb .le. 5) then
      call filter (1)
      if (numfile .eq. 2) call filter (2)
endif
c if (lhb .eq. 5) call correlate (xpassno, ypassno)
c call inverseft (1, xmean, xpassno)
if (numfile .eq. 2) call inverseft (2, ymean, ypassno)
c WRITE (20) xrow, xcol, zero, xmean, xpassno, eight
   do 1-i, xrow
      WRITE (20) xdata(i)
   enddo
   if (numfile .eq. 2) then
      WRITE (21) xrow, xcol, zero, xmean, xpassno, eight
      if (seven .eq. 7777) then
         gfcnt=gfcnt+1
         go to 570
      endif
      do 1-i, yrow
         WRITE (21) ydata(i)
      enddo
   endif
570 if (isub .eq. 1) then
      if (seven .eq. 7777) then
         write (23) xrow, xcol, zero, xmean, xpassno, eight
         do 1-i, xrow
            write (23) xdata(i)
         enddo
      elseif (seven .ne. 7777) then
         write (23) xrow, xcol, zero, xmean, xpassno, seven
         do 1-i, xrow
            write (23) (strxdata(i)-xdata(i))
         enddo
      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
c subroutine forwardft (num, mean, passno)
integer num, xlobs, ylobs, xynobs, yynobs, passno
real xdata(4096), ydata(4096), prcnt, mean
complex xcdat(4096), ycdat(4096)
end
common /fftlfft/ nobs, prcnt, lmean, fold
common /rowcol/ xdata, ydata
common /reals/ xdata, ydata
common /comps/ xdata, ydata

DIMENSION X(2, 4096)

DOUBLE PRECISION TSL

EQUIVALENCE (X(I, I), H(I))

TSL = 0. DO
IF (num .eq. 1) then
xynobs = xynobs
DO I = 1, xynobs
X(1, I) = xdata(I)
TSL = TSL + X(1, I)
ENDDO
ELSEIF (num .eq. 2) then
xynobs = xynobs
DO I = 1, xynobs
X(1, I) = ydata(I)
TSL = TSL + X(1, I)
ENDDO
ENDIF

SUBROUTINE DESCRIPTION

REQUIRED SUBROUTINES:
FFTID, FORK, DATWND

DIMENSIONING REQUIREMENTS:

X(2, N) WHERE N IS THE NUMBER OF COLUMNS AND ROWS OF THE
OUTPUT TRANSFORMED MATRIX. N MUST BE AN INTEGER
POWER OF TWO (2, 4, 8, 16,...).
NOTE: DIMENSIONS IN EVERY SUBROUTINE MUST BE
SET EQUAL TO DIMENSIONS IN MAIN PROGRAM.

AUTHOR: SUBROUTINES FFTID 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 Mineralogy
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.
These revisions will almost always be lower case letters.

Revised again (Judas priest this is getting old) on
1 Aug 90 into this present format of all fourier programs
combined into this program. For a full listing of all
comments in the 6 APR 90 version, see that version. No
kidding.

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

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 XMEAN1 = TSL/FLOAT(xynobs)
DO IF = 1, xynobs
X(1, IF) = X(1, IF) - XMEAN1
ENDDO
WRITE(25,1020) XMEAN1

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

C..THE MATRIX IS NOW ZERO FILLED TO MXOUT BY NYOUT SIZE
C CALCULATE AND REMOVE THE MEAN INTRODUCED BY TAPERING
C TSL = 0. DO
DO IF = 1, nobs
TSL = TSL + X(1, IF)
ENDDO
XMEAN2 = TSL/FLOAT(nobs)
DO IF = 1, nobs

B-63
X(I,IY)=X(I,IY)-MEAN2
endo
WRITE(25,1020) XMEAN2
XMEAN=XMEAN2+MEAN1
WRITE(25,1080) XMEAN
write (25,*) passno,xmean,xmean2,m_ean

C....TRANSFORM DATA TO THE WAVENUMBER DOMAIN
C
CALL FFTID(nobs,-1)
C
mean=xmean
if (num .eq. 1) then
   do ix=1,nobs
       xdata(ix) = h(ix)
   enddo
elseif (num .eq. 2) then
   do ix=1,nobs
       ydata(ix) = h(ix)
   enddo
endif

return

1020 FORMAT('MEAN REMOVED ',F15.7)
1030 FORMAT('MUST BE A POWER OF 2: SPA2FRQ 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.
C "nobs" - NUMBER OF fft observations IN DATA MATRIX
C "NSIGN" - DIRECTION OF DESIRED TRANSFORMATION
C +1 INVERSE TRANSFORM (FREQUENCY TO SPATIAL)
C -1 FORWARD TRANSFORM (SPATIAL TO FREQUENCY)
C*************************************************************************
C COMMON H (4096)
COMPLEX H
C SIGN=FLOAT(NSIGN)
C IF (IABS(NSIGN).NE.1) THEN
WRITE(6,105)
STOP
ENDIF
C
CALL FORK (nobs,H,SIGN)
C
RETURN
C
105 FORMAT(5X,'"NSIGN" MUST EQUAL +1 OR -1 FOR "FFT2D", FATAL')
C
END
C*************************************************************************
C SUBROUTINE FORK (LXX,CX,SIGN)
C*************************************************************************
C FAST FOURIER TRANSFORM, MODIFIED FROM CLAERSOFT, J.F., FUNDAMENTAL OF GEOPHYSICAL DATA PROCESSING, MCGRAW-HILL, 1976
C FORK USES COOLEY-TUYKY ALGORITHM.
C "CX" = DATAVECTOR TO BE TRANSFORMED
C "LXX" = LENGTH OF DATAVECTOR "CX" TO BE TRANSFORMED, MUST BE A POWER OF 2 (LXX=2**INTEGER)
C "SIGN" - 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),CM,CTEMP,CON2
CLX=LXX
LXH=LX/2
J=1
DO 103 I=1,LX

B-64
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)*SIGN
DO 105 I=1,L
CON=CEXP(CON2*FLOAT(M-1))
DO 105 I=M,L,ISTEP
  CTEMP=CM*CX(I)
  CX(I)=CX(I)-CTEMP
105 CX(I)=CX(I)*SC
RETURN
END

C*************************************************************************
C SUBROUTINE DATWND (PRCNT,xynobs,nobs,fold)
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
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
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 if (fold.gt.0.0 .and. fold.lt.100.0) then
  KX=Int(fold*FLOAT(xynobs)/100.0+0.5)
  if (Kx+nobs .gt. nobs) KX=(nobs-nxobs)/2
  do I=1,xynobs
    holdme(I)=f(I,1)
  enddo
C************************************************************** fold out the observations
  do I=1,xynobs+KX
    if (1.0.e.kx) f(I,1)=holdme(KX-I+1)
    if (1.0.e.kx .and. 1.1.e.kx+1) f(I,1)=holdme(KX+1-I)
    if (1.0.e.kx+1) f(I,1)=holdme((2*KX+nobs+1-I))
  enddo
  nxl=xynobs+2*KX
C**************************************************************
if (prcnt.gt.0.0 .and. prcnt.lt.50.0) then
  KK=IFIX(PRCNT*FLOAT(NXI)/100.0+0.5)
  if (KK.NE.0) then
    BKXPI=3.14159265/FLOAT(KK)
    CFACTOR=0.5*CM+COS(KK*KKPI)
    IXX=NX1+1
    F(I,1XX)=F(I,1XX)*CFactor
  endif
PROGRAM BANDPASS

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.

REQUIRED SUBROUTINES:

BNDPAS, FFT2D, FORK, STORE, WINDOW

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.

AUTHOR : ODN REED, PURDUE UNIVERSITY, DECEMBER 1980.
REVISIONS BY STEVE MATESRON 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.

B-66
update: 2 feb 90, removed need for cstore array

*********************************************************
C
C Create ideal continuation filter and store in array H
CALL BNDPAS (CUTLO, CUTHI, NPASS, DELTA, nobs)
C
C Create smoothed filter
C
IF (XLAG.GT.0.0 .AND. XLAG.LE.100.0) THEN
  IF (NWIND.GT.0 .AND. NWIND.LE.4) THEN
    CALL FFTID (nobs,1)
    CALL WINDOW (nobs,XLAG,NWIND)
    CALL FFTID (nobs,-1)
  ENDIF
ENDIF
C
C Write filter (wavenumber domain) onto unit 30 if IOFIL = 1
IF (IOFIL.EQ.1) THEN
  WRITE (30,*) (H(IX),IX-1,nobs)
ENDIF
C
if (num .eq. 1) then
  IF (XLAG.GT.0.0 .AND. XLAG.LE.100.0) THEN
    CALL FFTID (nobs,1)
    CALL WINDOW (nobs,XLAG,NWIND)
    CALL FFTID (nobs,-1)
  ENDIF
ENDIF
C
return
end

*********************************************************
C
SUBROUTINE BNDPAS (CCUTLO, CUTHI, NPASS, DELTA, nobs)
C
"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
C
"CCUTLO" lowest wave# to be passed, ge 0.0
"CUTHI" 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
C
C
COMMON H(4096)
COMPLEX H, ZERO, ONE
C
CCUTHI=CCUTLO
CUTLO=CCUTLO
RCUTLO=999999.99
IF (CUTLO.GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
RCUTHI=1.0/CUTLO
WAVLEN=2.0*DELTA
FNQI=1.0/WAVLEN
WRITE (25,112) FNQI,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI,NPASS
C
IF (IABS(NPASS) .NE. 1) THEN
  WRITE (6,151)
  STOP
ENDIF
IF (CUTHI.GT.FNQI.OR.CUTHI.LE.CUTLO.OR.CUTLO.LT.0.0) THEN
  WRITE (6,151)
  STOP
ENDIF
C
NXX=nobs+2
NOZ=(nobs/2)+1
ZERO = (0.0,0.0)
ONE = (1.0,0.0)
IF (NPASS.NE.1) THEN
  ZERO = (0.0,0.0)
  ONE = (0.0,0.0)
ENDIF
C
B-67
CLOWX = FLOAT (NX2)*WAVLEN*CUTLO
CHIX = FLOAT (NX2)*WAVLEN*CUTHI
C
C .... "ZERO" OUT THE ENtIRE ARRAY
C
DO IX=1,nobs
   H(IX) = ZERO
endo
C
C .... OPERATE ON ROWS WHERE WAVENUMBERS ARE LE. CUTLO
C
MINS = 1
MAXS = NX2
IF (CUTLO.GT.0.00001) MINS = int(clowx + 2.0001)
IF (FNL(CUTHI).GT.0.00001) MAXS = int(chlx + 1.0001)
if (mins .le. maxs) then
   DO IX=MINS,MAXS
      H(IX) = ONE
      R(IX) = ONE
   enddo
ENDIF
C
RETURN
C
C************************************************************************
C
SUBROUTINE WINDOW (nobs,xlag,nwinds)
C************************************************************************
C "WINDOW" PERFORMS 1-DIMENSION WINDOWING OVER THE DATA ARRAY
C EACH QUAD. IS SEPARATELY WINDED. THE 1.0 COEFFICIENT IS ALWAYS
C THE OUTER MOST CORNER OF THE ARRAY.
C
C "nobs" - NUMBER OF OBSERVATIONS IN DATA MATRIX
C "XLAG" - SMOOTHING PARAMETER FOR WINDOWING IDEAL FILTER IN SPATIAL *
C          DOMAIN. DETERMINES WHAT PERCENTAGE OF DATA IS WINDED *
C           (nobs*XLAG/100.0) THE REMAINDER IS SET TO 0.0. I.E. THE *
C           SMALLER "XLAG" THE SMOOTHER THE WINDOWING. *
C           "XLAG" MUST BE >.0 AND. LE. 100.0 FOR WINDOWING *
C           VALUES OUTSIDE OF THIS RESULTS IN NO WINDOWING *
C           THE SMALLER THE "XLAG" THE SMOOTHER THE FILTER. *
C
C "NWIND" = TYPE OF WINDOW TO APPLY *
C -0 GIVES NO WINDOW *
C -1 gives a rectangular window *
C -2 GIVES BARTLETT WINDOW (TRIANGLE WINDOW) *
C -3 GIVES HAMMING-TUKEY WINDOW *
C -4 GIVES PARZEN WINDOW *
C
C************************************************************************
C COMMON H(4096)
COMPLEX H
C
IAG = FLOAT(nobs)*XLAG/200.0+0.5
LAG = ANMOD(IAG,2)
PI = 3.14159265
NX2 = (nobs/2)+1
NX = FLOAT(NX2)
XX01 = 1.0/FLOAT(NX2)
C
RADIUS = FLOAT(LAG)*XX01
RAD1 = 1.0/RADIUS
RAD2 = RAD1*RADIUS
C
C .... APPLY RECTANGULAR WINDOW TO FILTER
C
IF (NWIND.EQ.1) THEN
   MAX = int(RADIUS*XX01+1.0001)
   LL = MAX
   DO II = 1, MAX
      H(NX-IX) = (0.0,0.0)
      H(II) = (0.0,0.0)
   enddo
   NWRITE(25,660) XLAG, IAG
RETURN
C
B-68
C......APPLY BARTLETT WINDOW TO FILTER
C
ELSEIF (NWIN.EQ.2) THEN
  MAX=RADIUS*XNXR+1.0001
  IF (MAX.GE.2) THEN
    DO 253 LL=2,MAX
      XI=FLOAT(LL-1)*XNX
      FACTOR=0.45*(1.0-COS(PIRADI*XI))
      H(LL)=H(LL)*FACTOR
      MK=NOXX-LL
      253 H(MK)=H(MK)*FACTOR
    ENDIF
    LL=MAX+1
    DO 255 II=LL,NX2
      H(II)=(0.0,0.0)
      255 H(II)=(0.0,0.0)
    C C WRITE (25,661) XLAG,LAG
    RETURN
C......APPLY HAMMING-TUKEY WINDOW TO FILTER
C
ELSEIF (NWIN.EQ.3) THEN
  PIRADI=PI*RADI
  MAX=RADIUS*XNXR+1.0001
  IF (MAX.GE.2) THEN
    DO 353 LL=2,MAX
      XI=FLOAT(LL-1)*XNX
      FACTOR=0.5*(1.0+COS(PIRADI*XI))
      H(LL)=H(LL)*FACTOR
      MK=NOXX-LL
      353 H(MK)=H(MK)*FACTOR
    ENDIF
    LL=MAX+1
    DO 355 II=LL,NX2
      H(II)=(0.0,0.0)
      355 H(II)=(0.0,0.0)
    C C WRITE (25,662) XLAG,LAG
    RETURN
C......APPLY PARZEN WINDOW TO FILTER
C
ELSEIF (NWIN.EQ.4) THEN
  MAX=RADIUS*XNXR+1.0001
  MAX2=SQRT(RAD2/4.0)*XNXR+1.0001
  FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
  H(1)=H(1)*FACTOR
  C IF (MAX2.GE.2) THEN
    DO 453 LL=2,MAX2
      XI=FLOAT(LL-1)*XNX
      FACTOR=1.0-6.0*((XI*RADI)**2-(XI*RADI)**3)
      H(LL)=H(LL)*FACTOR
      MK=NOXX-LL
      453 H(MK)=H(MK)*FACTOR
    ENDIF
    COUNT=MAX2+1
    DO 457 LL=COUNT,MAX
      XI=FLOAT(LL-1)*XNX
      FACTOR=2.0*(1.0-(XI*RADI)**3)
      H(LL)=H(LL)*FACTOR
      MK=NOXX-LL
      457 H(MK)=H(MK)*FACTOR
    LI=MAX+1
    DO 455 II=LL,NX2
      H(II)=(0.0,0.0)
      455 H(II)=(0.0,0.0)
    C C WRITE (25,663) XLAG,LAG
    RETURN
C......DO NOT APPLY A WINDOW TO FILTER
C
ELSEIF (NWIN.EQ.0) THEN
  C 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,ynobs,nobs,xpassno,ypassno, 
    > zerocont(4096),cnwind 
    real mincc,maxcc,cwlnout,prcnt, 
    > pctpr3,pctpr4,minccn,maxccn,cxlag 
    complex h(4096),power2,power6,totpw 
    COMPLEX X(4096),Y(4096),zero, 
    > POWER1,POWER2,POWER3,POWER4,POWER5 
    REAL 
    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, 
    XPOWER,POWER3,POWER4,POWER5,POWER6,POWER7,POWER8 
    COMMON h 

    subroutine description 
    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 
    bandpassing subroutines to this cc-filter. try them if 
    you like! 
    updates 1 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 
    POWER5=ZERO 
    POWER6=ZERO 

    DO 110 l=1,nobs 

    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. 

    zerocont(l)=1 

    SUM THE POWERS & CROSS PRODUCTS OF THE INPUT MAPS. 
    POWER1=POWER1+X(I)*CONJG(X(I)) 
    POWER2=POWER2+Y(I)*CONJG(Y(I)) 
    XPOWER=XPOWER+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 (aimag(x(i)))/(real(x(i)))) 
    if (real(x(i)).lt.0.0) xrad=xrad+pi 
    yrad=atan (aimag(y(i)))/(real(y(i)))) 
    if (real(y(i)).lt.0.0) yrad=yrad+pi 
    delrad=abs (xrad-yrad) 
    ccoef=cos(delrad)

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

C SUM THE POWERS & CROSS PRODUCTS FOR THE OUTPUT MAPS.
POWER3 = POWER3 + (X(I) * CONJG(X(I)))
POWER4 = POWER4 + (Y(I) * CONJG(Y(I)))
TPOWER = TPPOWER + (X(I) * CONJG(X(I)))
CONTINUE

C 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(TPOWER/SQRT(POWER1 * POWER2))
endif

C 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

C 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
   PCTPRI = (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 (6,666) PCTPRI, PCTPR2

C WRITE THE POWER PERCENTAGES TO FILE 6.
WRITE (6,666) PCTPRI, PCTPR2

C------ 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
   power5 = zero
   powex = zero
   totpow = zero
   do 1 = 1, nobs
      if (zerocont(I) .eq. 0) h(1) = (0.0, 0.0)
   enddo
   call fftld(nobs, nobs, cnwind)
endif

C-71
power6=power6+ \langle y(1) * \text{conj}(y(1)) \rangle \\
totpwr=totpwr+(k(1) * \text{conj}(y(1))) \\
enddo 
if (power5 .eq. zero .or. power6 .eq. zero) then 
write (*) 'power5 =',power5,xpassno 
write (*) 'power6 =',power6,ypassno 
ccwlnout=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 
ccwlnout=real(totpwr/sqrt(power5*power6)) 
pctpr3=power5/power1 *100.0 
pctpr4=power6/power2 *100.0 
340 
continue 
write (25,888) xpassno,ypassno,ccln,ccwlnout,pctpr3,pctpr4 
format (2i6,4f10.3) 
endlf 
return 
end 

subroutine inverseft (nmn,mean,passno) 
integer numsxnobs,ynobs,xynobs,row,col, pessno 
real xdata(4096),ydata (4096),mean 
complex xcdata(4096},ycdata(4096) 
common /rowcol/ xnobs,ynobs 
common /reals/ xdata, ydata 
common /comps/ xcdata,ycdata 
common /fftlfft/ nobs,prcnt,lmean, fold 
COMMON H(4096) DIMENSION X(2,4096),holdme(4096) 
COMPLEX H EQUIVALENCE (X(1, 1),H(1)) 
c 
if (num .eq. 1) then 
xynobs=xnobs 
do l=1,nobs 
h(l) = xcdata(l) 
endo 
elself (num . eq. 2) then 
xynobs=ynobs 
do i=1,nobs 
h(i) = ydata(l) 
endo 
endif 
--------------
subroutine description 
.. 
**INVERSE TRANSFORM DATA TO THE SPACE DOMAIN** 
CALL FFTID (nobs,+1) 

n-half=(nobs-xynobs)/2 
do l=n-half+1,n-half+xynobs 
holdme(l-n-half)=w(1,1) 
endo 
total=0.0 
do l=1,xynobs 
x(1,1)=holdme(l) 
total=total+x(l,1) 
endo 
xmean=total/float(xynobs)
IF (IHEA_.EQ. I) THEN
  do i-I, xynobs
    x(I, I) = x(I, I) + mean
  enddo
ENDIF
C
XMIN= 1.0E20
XMAX=-1.0E20
if (num .eq. 1) then
  do i-1, xynobs
    xdata(I) = x(I, I)
  enddo
else (num .eq. 2) then
  do i-1, xynobs
    ydata(I) = x(I, I)
  enddo
endif
C
DO I-1, xynobs
  XMIN=AMINI(XMIN, X(I, I))
  XMAX=AMAXI(XMAX, X(I, I))
  IF (XMAX.EQ.X(I, I)) IMAX-I
  IF (XMIN.EQ.X(I, I)) IMIN-I
enddo
C
WRITE(25,1020) XMAX, IMAX, XMIN, IMIN, XMEAN, PASSNO
WRITE(25,9980) PASSNO, XMEAN, XMAX, IMAX, XMIN, IMIN
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

B-73
program combine
  integer xrow,xcol,zero,eight,xpassno,ypassno,ycol,
    > x2row,x2col,x3pass,y3row,y3col,y3pass,xrow,
    > stratc,sxrow,sxcol,sxcol,scols,scols,scols,
    > sy3pass,eq3pass,ncnt,paircnt,choice,
    > prime,global,crosscnt,
    > passrem,nowant(4000),nocnt,type
  real xmean,ymean,x3data(400,3),y3data(400,3),y3mean,
    > xldata(400),ylata(400),xdata(400,4),ydata(400,4),
    > xmean,ymean,ymean,x(400),y(400),
    > savey(400),savex(400,3),xmean,
    > ymean,avdata(400,4)
  character*80
  common /trunstat/ xdata(400,4),ydata(400,4)
  common x3data(400,3),xldata(400),y3data(400,3),yldata(400)
  common /aver/ avdata(400,4)
  common /tplot/ x(400),y(400)
  common /nope/ nowant(4000),nocnt
  c
  program combine
  c 9990 program combine
  c xrow,xcol,zero,eight,xpassno,ypassno,ycol,
  c x2row,x2col,x3pass,y3row,y3col,y3pass,xrow,
  c stratc,sxrow,sxcol,sxcol,scols,scols,scols,
  c sy3pass,eq3pass,ncnt,paircnt,choice,
  c prime,global,crosscnt,
  c passrem,nowant(4000),nocnt,type
  c xmean,ymean,x3data(400,3),y3data(400,3),y3mean,
  c xldata(400),ylata(400),xdata(400,4),ydata(400,4),
  c xmean,ymean,ymean,x(400),y(400),
  c savey(400),savex(400,3),xmean,
  c ymean,avdata(400,4)
  c character*80
  c common /trunstat/ xdata(400,4),ydata(400,4)
  c common x3data(400,3),xldata(400),y3data(400,3),yldata(400)
  c common /aver/ avdata(400,4)
  c common /tplot/ x(400),y(400)
  c common /nope/ nowant(4000),nocnt
  c
  ccombine is very similar to movetrunc in that both
  c programs truncate adjacent passes to the same over-
  c lapping segments, both programs also provide
  c statistics and track output files. the major
  c difference is that movetrunc has only one file
  c as input whereas combine has two files as input.
  c combine can output one file of two passes averaged
  c together to make one file or it can output
  c two similar length passes that can be further
  c processed by fourier methods. movetrunc and combine
  c could be cluged together to make one program
  c so why don't you go ahead and jam them together??
  c good luck!!
  c
  program date: 16 apr 91
  c
  write (*,*)'I TO HAVE ONE OUTPUT FILE'
  write (*,*)'2 TO HAVE TWO OUTPUT FILES'
  read (*,*) ifilenum
  if (ifilenum .eq. 2) then
    goto 1
  endif
  write (*,*)'I 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 (*,*)'i IF THE DATA SET DOES NOT
  INCLUDE ALL LONGITUDES'
  read (*,*) global
  if (prime .eq. 1) then
    write (*,*)'NOTE: FILE Y WILL HAVE THE FIRST PASS MOVED'
    write (*,*)'TO THE BOTTOM OF THE FILE'
    if (global .eq. 1) then
      write (*,*)'AND THE FIRST PASS WILL NOT BE INCLUDED IN'
      write (*,*)'THE PROCESSING. FILE X WILL HAVE THE LAST'
      write (*,*)'PASS REMOVED AND THIS PASS WILL NOT BE INCLUDED'
      write (*,*)'IN THE PROCESSING'
    endif
  endif
  write (*,*)'INPUT FILE X OF LAT-LONG-RAD DATA'
  read (9990) filename
  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 TPL0'
    read (9990) filename
    open (21, file=filename,form='unformatted')
  else if (ifilenum .eq. 2) then
    write (*,*)'OUTPUT FILE X OF LAT-LONG-RAD'
    read (9990) filename
    open (30, file=filename,form='unformatted')
  endif
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')
END*

WRITE ('*',*) 'OUTPUT FILE OF STATISTICS'
READ (*,9990) FILENAME
OPEN (25, FILE=FILENAME, FORM='FORMATTED')
WRITE ('*',*) '0 IF THESE ARE DUSK DATA SETS'
WRITE ('*',*) '1 IF THESE ARE DAWN DATA SETS'
READ (*,*) DINDK
IF (DINDK .EQ. 1) THEN
WRITE ('*',*) 'DO NOT REMOVE THE MEAN FROM THE AVERAGED DATASET'
WRITE ('*',*) '1 REMOVE THE MEAN'
READ (*,*) CHOICE
ENDIF
WRITE ('*',*) '0 IF ALL PASSES ARE WANTED'
WRITE ('*',*) '1 TO REMOVE THE PASSES THAT ARE NOT WANTED'
READ (*,*) NCNT
IF (NCNT .EQ. 1) THEN
WRITE ('*',*) 'INPUT FILE OF PASSES NOT WANTED'
OPEN (14, FILE=FILENAME, FORM='FORMATTED', STATUS='OLD')
DO 5 I=1,4000
READ (14,*,END=6) NOWANT(I)
5 CONTINUE
CONTINUE
NCNT=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 (*,*) MLNOBS
WRITE (25,*) 'XPASS YPASS CCD CCY XVAR YVAR',
WRITE (25,*) 'COVARY XSTDEV YSTDEV',
WRITE (25,*) 'XMEAN YMEAN XSLOPE YSLOPE',
WRITE (25,*) 'KINTCPT YINTCPT',
PAIRCNT=0
PAIRCNT1=0
TICOUNT=0
STRCNT=0
TOCOS=0
NCROSS=0
PASSREM=0
10 CONTINUE
READ (10,END=999) XROW,XCOL,ZERO,XMEAN,XPASS,EIGHT
DO 15 I=1,XROW
READ (10) (X3DATA(I,11),I=1,X3COL)
15 CONTINUE
READ (11) XROW,XCOL,ZERO,XMEAN,XPASSNO,EIGHT
DO 25 I=1,XROW
READ (11) XDATA(I)
25 CONTINUE
READ (12,END=150) YROW,Y3COL,ZERO,YMEAN,YPASS,EIGHT
DO 35 I=1,YROW
READ (12) (Y3DATA(I,11),I=1,Y3COL)
35 CONTINUE
READ (13) YROW,YCOL,ZERO,YMEAN,YPASSNO,EIGHT
DO 45 I=1,YROW
READ (13) YDATA(I)
45 CONTINUE
46 CONTINUE
STRCNT=STRCNT+1
C--------------------------- this if statement offsets the passes
and saves the offset for the end

if (strcnt .eq. 1 .and. prime .eq. 1) then
  do 50 i-l,y3row
    do 55 li-l,y3col
      savey3(i,ii)-y3data(i,li)
  55 continue
  ydata(1)=savey3(1)
  sy3row=y3row
  syrow=yrow
  sy3col=y3col
  sycol=ycol
  sy3mean=y3mean
  sy3mean=ymean
  sy3pass=y3pass
  ypassno=ypassno
  go to 30
endf

go to 190

continue

if (global .eq. 1) go to 999
  do 160 i-l,sy3row
    do 165 li-l,sy3col
      y3data(i,ii)-savey3(i,ii)
    165 continue
  ydata(1)=saveyl(1)
  sy3row-sy3row
  syrow-syrow
  sy3col-sy3col
  ycol-sycol
  sy3mean-sy3mean
  ymean-symean
  y3pass-sy3pass
  ypassno=ypassno
endf

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 (*,*) 'ROWS OR PASSNOS
  X- ',xrow,x3row,
    xpassno,x3pass
  yrow, y3row, ypassno, y3pass
  go to 999
endif

else if (xrow .lt. mlnobs .or. xrow .lt. minobs) then
  write(*,'FILE X PASS',xpassno, 'REMOVED: OBSERVATIONS= ',xrow)
  write(*,'FILE Y PASS',ypassno, 'REMOVED: OBSERVATIONS= ',yrow)
  passrem-passrem+1
  go to 10
endif

do 195 i=1,ncnt
  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
  endif

195 continue

call truncate (xrow,yrow,dndk,nobs,paircntl,mlnobs)

call statistics (nobs,xpassno,ypassno,xmean,ymean)

if (ifilenum .eq. 1) call average (nobs,choice,prime,crosscnt)

if (ifilenum .eq. 2) then
  if (xpassno .ne. ypassno) then
    stop 444
  endif
endif

lthree=3
ione=1
xxavg=0.0
leight=8888
lzero=0
write (30) nobs,lthree,zero,xxavg,xpassno,leight
write (31) nobs,ione,zero,xxavg,xpassno,leight
write (32) nobs,lthree,zero,xxavg,xpassno,leight
write (33) nobs,ione,zero,xxavg,xpassno,leight

do j=1,nobs
  if (xdata(j,3) .gt. 180.0) xdata(j,3)-xdata(j,3)-360.0
  write (30) xdata(j,1), (xdata(j,i),i-3,4)
  write (31) xdata(j,2)
  if (ydata(j,3) .gt. 180.0) ydata(j,3)-ydata(j,3)-360.0
  write (32) ydata(j,1), (ydata(j,i),i-3,4)
  write (33) ydata(j,2)
endd

100 continue

B-77
END

C-------------------------------------------------------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)
      200 continue
   do 205 j=1,nobs-1
      if (xadata(j,3) .lt. xadata(j+1,3)) then
         crosscnt=crosscnt+1
      go to 207
   endif
   205 continue
   207 continue
   elseif (prime .eq. 2) then
      do 220 j=1,nobs
         if (avgdata(j,2) .gt. 180.0) avgdata(j,2)=avgdata(j,2)-360.0
      write (20,*) (avgdata(j,k), k=1,4)
      220 continue
   endif

C------------------------------------------------------- write out the trace of the pass for plotting

   Call track (nobs,noc,prime)
   WRITE (21) NOB,Nobs,(X(J),Y(J),J=1,Nobs)
   continue
   300 continue
   if (paircnt .gt. 0) tcount=tcount+1
   paircnt=paircnt+paircnt
   totobs=totobs+nobs
   go to 10
   999 continue
   write (*,*),'corrected','paircnt',' pairs of latitudes in'
   write (*,*),'tcount',' passes to beginning lengths'
   write (*,*),'total passes read = ',stront
   write (*,*),' removed','passes from the file'
   write (*,*),'total observations in the written dataset = ',totobs
   >      ' study area includes','crosscnt',' pairs of'
   >      ' longitudes that cross -180.0 180.0'
      close (10)
      close (11)
      close (12)
      close (13)
      close (20)
      close (21)
      close (23)
      close (30)
      close (31)
      close (32)
      close (33)
      stop
   end

SUBROUTINE TRUNCATE (xrow,yrow,dndk,minrow,stocount,mlnobs)
   INTEGER xrow,yrow,stocount,rowii,rowinc,minrow,
           dndk,mlnobs
   REAL xdata(400,4),ydata(400,4),
           x3data(400,3),xldata(400),
           y3data(400,3),yldata(400),
           adeta,bdeta,dlffab,abss,xadeta(400,4),
           ybdata(400,4)
   COMMON /trunstat/ xadata(400,4),ybdata(400,4)
   COMMON x3deta(400,3),xldata(400),y3deta(400,3),yldeta(400)
C------------------------------------------------------- subroutine description

   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)=x3data(J,1)
      xdata(J,2)=xldata(J)
      xdata(J,3)=x3data(J,2)
      xdata(J,4)=x3data(J,3)
   70 continue
   DO 75 J=1,yrow
      ydata(J,1)=y3data(J,1)
      ydata(J,2)=yldata(J)
      ydata(J,3)=y3data(J,2)
      ydata(J,4)=y3data(J,3)
   75 continue
   STOCOUNT=0
   JY=J
   ROWII=XROW

B-78
...loops from 90 to 200 increment through the c two input passes and truncate the lengths c to the same length

90 continue
  adatat=xdata(jj,1)
  bdatal=ydata(jj,1)
  diffab=adatat-bdatal
  abs=abs(diffab)
  if (rowii .eq. 0 .or. rowlnc .eq. 0) then
    write (*,*) 'xrowe (li) -',rowil,'
yrows (Inc) -',rowinc
    write (*,*) 'xrow -',xrow,'
yrow -',yrow
    stop
  endif
  minrow=min(rowii,rowlnc)

if pass (li) matches pass b (Inc) at c beginning length then write to xdata and c ydata and race to main program

if (abss .lt. 0.33) then
  do 110 ll-l,minrow
    do 110 kk-l,4
      xadata(ii, kk)=xdata(ll,kk)
      ydata(ii, kk)=ydata(ll,kk)
  110 continue
else (xdata(JJ,1) .lt. ydata(JJ,1)) then
  rowlnc=rowinc-1
  do 130 nn=mm+1,rowinc
    ydata(nn,kk)=ydata(mm+1,kk)
  130 continue
  do 130 kk=ll,1,4
    ydata(nn,kk)=ydata(nn+1,kk)
    xdata(nn,kk)=xdata(nn+1,kk)
  130 continue
  endif
  stocount=stocount+1
  go to 90
else (xdata(JJ,1) .gt. ydata(JJ,1)) then
  rowi=rowii-1
  do 150 nn=ll+1,rowii
    xdata(nn,kk)=xdata(nn+1,kk)
  150 continue
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endi...
xsum=0.0
xsumsq=0.0
ysum=0.0
ysumsq=0.0
sumxy=0.0

DO 240 J=1,NOBS
  xsum=xsum+xadata(J,2)
  xsumsq=xsumsq+(xadata(J,2)**2)
  ysum=ysum+ybdata(J,2)
  ysumsq=ysumsq+(ybdata(J,2)**2)
  sumxy=sumxy+(xadata(J,2)*ybdata(J,2))
240 CONTINUE

WRITE(*,*) xsum, ysum, xsumsq, ysumsq, sumxy

C find corrected sum of products, covariance and corrected sum of squares (x) (y)
Cxmean=xsum/nobss
C
C ymean= ysum/nobss
C
C sumprod=sumxy*((xsum*ysum)/nobss)
C covarxy=covarxy/(nobss-1.0)
C
C xsumsqr=xsumsqr-((xsum**2)/nobss)
C ysumsqr=ysumsqr-((ysum**2)/nobss)
C
C find variance, standard deviation for x and y
Cxvar=xsumsqr/(nobss-1.0)
Cyvar=ysumsqr/(nobss-1.0)
C
C xstdev=sqrt(xvar)
C ystdev=sqrt(yvar)
C
C find correlation coefficient by Davis method
C corrDxy=covarxy/(xstdev*ystdev)
C
C find slopes, intercepts and correlation coefficient by Young method
C xslope=(((nobss*sumxy)-xsum*ysum)/(nobss*xsumsqr)-xsum**2)
C yslope=(((nobss*sumxy)-xsum*ysum)/(nobss*ysumsqr)-ysum**2)
C
C xintcpt=((ysum*xsumsqr)-(xsum*ysum)/(nobss*xsumsqr)-xsum**2)
C yintcpt=((xsum*ysumsqr)-(sumxy*ysum)/(nobss*ysumsqr)-ysum**2)
C
corrYxy=(xslope*yslope)
C
C write out this mess for individual pass and overlapping lengths of passes
C
C write (25,9992) xpassno,ypassno,xvar,yvar,xstdev,ystdev,
C   xamean, ybmean
9992 FORMAT('FOR OVERLAPPING LENGTHS X=',I5,' Y=',I5,/, 'X VARIANCE=',F9.3,' Y VARIANCE=',F9.3,' X STDDEV=',F9.3,' Y STDDEV=',F9.3)
C
C write (25,9993) xslope, yintcpt, xintcpt, corryxy, corrDxy
9993 FORMAT('X SLOPE=',F9.3,' Y INTERCEPT=',F9.3,' Y SLOPE=',F9.3)
C
C write (25,9995) xpassno, ypassno, xslope, yintcpt, xintcpt, yintcpt, corrDxy, corrYxy, xvar, yvar
9995 FORMAT(215,7(F0.4))
C
C write (25,9996) xamean, ybmean, xslope, yslope, xintcpt, yintcpt
9996 FORMAT(10x,6(F0.4))
RETURN
END

SUBROUTINE average (nobs, choice, prime, crosscnt)
REAL xadata(400,4), ybdata(400,4), avgdatmean, avgdatsum,
   avgdata(400,4), nobbs
INTEGER nobbs,choice, prime, crosscnt
COMMON /trunstat/ xadata(400,4), ybdata(400,4)
COMMON /aver/ avgdata(400,4)

C subroutine average (nobs, choice, prime, crosscnt)
C real xadata(400,4), ybdata(400,4), avgdatmean, avgdatsum,> avgdata(400,4), nobbs
C integer nobbs, choice, prime, crosscnt
C common /trunstat/ xadata(400,4), ybdata(400,4)
C common /aver/ avgdata(400,4)
C
C------------------------------- subroutine description
C average calculates the average magnetic value of the C
C input passes. it will also find the average position of C
C the input passes if so directed.
C
C avgdatsum=0.0
C nobbs=real(nobs)
C
C IF (prime .EQ. 2) THEN
C DO 100 I=1,nobbs
C  avgdata(1,1)=(xadata(I,1)+ybdata(I,1))/2.0
C  avgdata(1,2)=(xadata(I,1)+ybdata(I,1))/2.0
C  addxy=abs(xadata(I,1))+abs(ybdata(I,1))
C  IF (addxy .GT. 270.0) THEN
C    crosscnt=crosscnt+1
C ELSE IF (ybdata(I,1).GT.0.0) THEN
C ELSE IF (ybdata(I,1).LT.0.0) THEN
C ELSE IF (xadata(I,1).EQ.0.0) THEN
C ELSE IF (xadata(I,1).NE.0.0) THEN
C elseif (ydata(I,1).LT.0.0) THEN
C ELSEIF (xdata(I,1).EQ.0.0) THEN
C ELSEIF (xdata(I,1).NE.0.0) THEN
C B-80
ybdata(1,3) - ybdata(1,3) + 360.0  
endif  
avgdata(1,2)={avgdata(1,2)+ydata(1,2)}/2.0  
if (avgdata(1,2) .gt. 180.0)  
avgdata(1,2)-avgdata(1,2)-360.0  
endif  
avgdata(1,4)={avgdata(1,4)+ydata(1,4)}/2.0  
avgdatum=avgdatum+avgdata(1,4)  
continue  
avgdatmean=avgdatum/nobs  
if (choice .eq. 1) then  
do 150 i=1,nobs  
avgdata(i,4)-avgdata(i,4)-avgdatmean  
continue  
endif  
elseif (prime .eq. 1) then  
do 200 j=1,nop  
x(j)-xdata(j,1)  
y(j)=ydata(j,2)  
if (y(j) .lt. 0.0) y(j)=y(j)+360.0  
x(j)=x(j)*RADFAC  
y(j)=y(j)*RADFAC  
continue  
elself (prime .eq. 1) then  
do 300 j=1,nop  
x(j) = xdata(j,1)  
y(j) = ydata(j,3)  
if (y(j) .lt. 0.0) y(j) = y(j) + 360.0  
x(j) = x(j) * RADFAC  
y(j) = y(j) * RADFAC  
continue  
endif  
return  
end  

subroutine track (nop, noc, prime)  
integer nop, noc, prime  
real radfac, avgdata(400,4), x(400), y(400), xadata(400,4), ybdata(400,4)  
common /aver/ avgdata(400,4)  
common /tplot/ x(400), y(400)  
common /trunstat/ xadata(400,4), ybdata(400,4)  
subroutine description  
track finds the long and east 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-avgdata(j,1)  
y(j)=avgdata(j,2)  
if (y(j) .lt. 0.0) y(j)=y(j)+360.0  
x(j)=x(j)*RADFAC  
y(j)=y(j)*RADFAC  
continue  
elself (prime .eq. 1) then  
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  
continue  
endif  
return  
end  

subroutine findgap (global,dndx, minobs, type)  
integer zero,elght, strtotpass,x3row,x3col,x3pass,  
y3row,y3col,y3pass, cnt, type, strmincnt,  
dndx,nowant(4000), ncnt, mincnt,  
nowant(4000), nocnt, allcnt, strallcnt  
real y3data(400,3), y3mean, absy, allxdata(400,4),  
y3data(400,4), x3mean, x3data(400,3), stryone(4)  
common /hope/ nowant(4000), ncnt  
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
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 you like.

totpass=0
allcnt=0
strnocnt=nocnt

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

30 continue
strtotpass=totpass
strallcnt=allcnt
totpass=0
allcnt=0
31 read (12, end=50) y3row, y3col, zero, y3mean, y3pass, eight
  do 35 i=1, y3row
    read (12) (y3data(i,1,ii), ii=1, y3col)
  35 continue
  if (y3row .lt. minobs) then
    nowant(nocnt+1)=y3pass
    nocnt=nocnt+1
  endif
  allcnt=allcnt+1
  allydat(allcnt,1)=real(y3pass)
  allydat(allcnt,2)=real(y3row)
  allydat(allcnt,3)=y3data(i,1)
  allydat(allcnt,4)=y3data(y3row,1)
goto 31

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

do 52 ii=1,4
  stryone(ii)=allydat(1,1)
52 continue
do 55 i=1,allcnt
  do 55 ii=1,4
    allydat(i,ii)=allydat(i+1,ii)
  55 continue
do 57 i=1,4
  allydat(allcnt,ii)=stryone(ii)
57 continue
if (global .eq. 1) allcnt=allcnt-1
if (type .eq. 2) go to 400

c---------------------- work a latitude gap finder

70 continue
cnt=1

c70 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

80 continue
if (nocnt .eq. 0) go to 140
do 110 i=1, nocnt
  if (int(allydat(cnt,1)) .eq. nowant(1)) then
    if (cnt+1 .gt. allcnt) go to 190
  do 105 jj=cnt, allcnt-1
B-82
do 105 J=1,4
   allxdat(jj,j)=allxdat(jj+1,j)
   allydat(jj,j)=allydat(jj+1,j)
105 continue
   allcnt=allcnt-1
end
110 continue
140 continue
   abss=abs(allxdat(cnt,3)-allydat(cnt,3))
   if (abss .lt. 0.33) go to 190
   if (abss .ge. 0.33) then
      nopass=int(allxdat(cnt,1))
   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
   c   elseif (dndk .eq. 0) then
      if (allxdat(cnt,3) .gt. allydat(cnt,3)) then
         if (allxdat(cnt,3) .lt. allydat(cnt,4)) then
            nocnt=nocnt+1
            nowant(nocnt)=nopass
         endif
      elseif (allxdat(cnt,3) .lt. allydat(cnt,3)) then
         if (allxdat(cnt,4) .lt. allydat(cnt,3)) then
            nocnt=nocnt+1
            nowant(nocnt)=nopass
         endif
      endif
      endif
   c------------------------if this is a dawn pass then will count from
   c   the equator toward the south pole
   c   elseif (dndk .eq. 1) then
      if (allxdat(cnt,3) .lt. allydat(cnt,3)) then
         if (allxdat(cnt,3) .lt. allydat(cnt,4)) then
            nocnt=nocnt+1
            nowant(nocnt)=nopass
         endif
      elseif (allxdat(cnt,3) .gt. allydat(cnt,3)) then
         if (allxdat(cnt,4) .gt. allydat(cnt,3)) then
            nocnt=nocnt+1
            nowant(nocnt)=nopass
         endif
      endif
   endif
190 continue
   cnt=cnt+1
   if (cnt .gt. allcnt) go to 200
   go to 100
200 continue
   if (nocnt .eq. strnocnt) go to 999
   if (strnocnt .lt. nocnt) then
      strnocnt=nocnt
      go to 70
   endif
400 continue
   mincnt=0
   strmincnt=mincnt
   c
   470 continue
   cnt=1
   c
   500 continue
   if (allxdat(cnt,1) .ne. allydat(cnt,1)) then
      write (*) 'PASSES DO NOT MATCH FOR A-east A-west'
      write (*) 'REVERSE THE ORDER OF INPUT FILES AND RERUN'
      stop
   endif
   if (int(allxdat(cnt,2)) .lt. mlnobs .or.
      int(allydat(cnt,2)) .lt. mlnobs) then
      if (cnt+1 .gt. allcnt) go to 590
      do 505 J=cnt,allcnt-1
         do 505 J=1,4
            allxdat(jj,j)=allxdat(jj+1,j)
            allydat(jj,j)=allydat(jj+1,j)
      505 continue
   c
   500 continue
   allcnt=allcnt-1
   go to 500
510 continue
540 continue
   abss=abs(allxdat(cnt,3)-allydat(cnt,3))
   if (abss .lt. 0.33) go to 590
   if (abss .ge. 0.33) then
      xrow=int(allxdat(cnt,2))
      yrow=int(allydat(cnt,2))
      minxyrow=min(xrow,yrow)
mincnt2-mincnt

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
else (allxdat(cnt,3) .lt. allydat(cnt,3)) then
  mincnt=mincnt+1
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
else (allxdat(cnt,3) .gt. allydat(cnt,3)) then
  mincnt=mincnt+1
endif
endif

if (mincnt .lt. mlncnt) then
  strmincnt=mincnt
  go to 470
endif
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)
1010 continue
endif
if (type .eq. 2) write (*,*), 'new minimum observation cutoff',
  ' is ',mlnobs
rewind (10)
rewind (12)
return
OC 1980, MACSAT12/83 FIELD MODEL

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>dx</th>
<th>dy</th>
<th>dz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0084</td>
<td>-1.5290</td>
<td>2180.5198</td>
<td>-335.5408</td>
<td>-3.6188</td>
</tr>
<tr>
<td>2</td>
<td>3.1459</td>
<td>1662.6899</td>
<td>-199.7370</td>
<td>03.1459</td>
<td>-24.1879</td>
</tr>
<tr>
<td>3</td>
<td>4.2139</td>
<td>26.5047</td>
<td>0</td>
<td>26.5047</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5.3567</td>
<td>1.1335</td>
<td>613.1343</td>
<td>0.6132</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>6.3800</td>
<td>-4.2379</td>
<td>-4.2132</td>
<td>1.2244</td>
<td>-4.876</td>
</tr>
<tr>
<td>6</td>
<td>7.4131</td>
<td>-22.1996</td>
<td>-4.1066</td>
<td>-0.0000</td>
<td>0.4037</td>
</tr>
<tr>
<td>7</td>
<td>8.4790</td>
<td>-27.4258</td>
<td>-4.7606</td>
<td>-1.1335</td>
<td>0.5800</td>
</tr>
<tr>
<td>8</td>
<td>9.5400</td>
<td>-9.6949</td>
<td>-2.4801</td>
<td>-0.3808</td>
<td>0.0000</td>
</tr>
<tr>
<td>9</td>
<td>10.6050</td>
<td>16.0818</td>
<td>1.7210</td>
<td>1.7163</td>
<td>-3.808</td>
</tr>
<tr>
<td>10</td>
<td>11.6950</td>
<td>149.8220</td>
<td>-4.0062</td>
<td>1.2244</td>
<td>-4.876</td>
</tr>
<tr>
<td>11</td>
<td>12.7860</td>
<td>17.1278</td>
<td>9.4401</td>
<td>-3.3002</td>
<td>0.0000</td>
</tr>
<tr>
<td>12</td>
<td>13.8634</td>
<td>-5.0000</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>13</td>
<td>14.9316</td>
<td>-4.9579</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>14</td>
<td>15.1050</td>
<td>-2.2499</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>15</td>
<td>16.1262</td>
<td>-20.8464</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>16</td>
<td>17.1574</td>
<td>-25.0584</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>17</td>
<td>18.1873</td>
<td>-29.0000</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>18</td>
<td>19.2172</td>
<td>-33.0000</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>19</td>
<td>20.2471</td>
<td>-37.0000</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
<tr>
<td>20</td>
<td>21.2770</td>
<td>-41.0000</td>
<td>-3.4208</td>
<td>-6.3179</td>
<td>-0.7318</td>
</tr>
</tbody>
</table>

**B 86**
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>13</td>
<td>-1.378</td>
<td>.4564</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>-2.2049</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>-2.947</td>
<td>-.4816</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>.3725</td>
<td>.3039</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>-2.8791</td>
<td>1.4826</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>-1.829</td>
<td>-.2216</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>1.1048</td>
<td>-.4758</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>-2.404</td>
<td>-.2088</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>.3847</td>
<td>.7985</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>-1.4823</td>
<td>.0535</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>.1867</td>
<td>.8530</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>-1.106</td>
<td>.0633</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>.3455</td>
<td>-.1350</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>-.0851</td>
<td>0.2500</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>.3797</td>
<td>-.3448</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>0</td>
<td>.0</td>
</tr>
</tbody>
</table>
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:
   \[ TP - TI \times COVM \times MI \]
   AND THE STANDARD ERROR OF PREDICTION IS GIVEN BY,
   \[ SEP = \sqrt{VAR - TI \times COVM \times TI} \]
   VAR...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 GIGL & 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) copying all arrays that work with the inversion as
   real*8.
4) removing every blasted 'nbug' statement 1 could
   get my hands on!!
5) reading the data once storing everything in
   memory -ie, not reading the data twice.
6) changing all logical
   true 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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.

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 getllraspc.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 alsdorf@geols.mps.ohio-state.edu

if you need help.
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

C
9991 format (a5)
C

If (EAST.EQ.WEST.OR.NORTH.IE.SOUTH.OR.NY.IE.0.OR.NX.
. IE.0) STOP 9999
C
C CHANGE LATITUDES TO SPHERICAL COORDINATES
C
NORTH=90.0-NORTH
SOUTH=90.0-SOUTH
C

-------------------- the program is checking for the -180.0 180.0 meridian
C
C cross=0.0
If (WEST.GT.EAST) cross=360.0
EAST=EAST + cross
C
C INPUT THE COVARIANCE TABLE
C
I=1
5 read (1,*,end=7) dnum(i),cnn(i)
I=I+1
10 go to 5
7 ncov=i-1
SCALE=CNN(1)
C
GRID SPACING ..ie: interval between grid nodes = dl and dp
C
DP=(NORTH-SOUTH)/FLOAT(NY)
DL=(EAST-WEST)/FLOAT(NX)
C
C .. DETERMINE THE OVERLAPS IN X AND Y-DIRECTIONS
C
YOVLAP=DP/2.
XOVVLAP=DL/2.
C
C .. Nx1,Ny1 = NUMBER OF SORT ELEMENTS IN X AND Y-DIR.
C
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 10 I=1,Nx1
Y(I)=Y(I)-1+YOVVLAP
Do 10 J=1,Ny1
X(J)=X(J)-1+XOVVLAP
10 CONTINUE
C
C DETERMINE THE X,Y COORDINATES OF THE DATA AREA AND
C GRID SPACING
C
Xll=0.
Xup=X(Nx1)+XOVVLAP
Yll=0.
Yup=Y(Ny1)+YOVVLAP
Dxx=Nx1/(Xup-Xll)
Dyy=Ny1/(Yup-Yll)
C
C DETERMINE BOUNDARIES FOR DATA SELECTION
C
THI=SOUTH-YOVVLAP
THIN=NORTH-YOVVLAP
EPHI=EAST-XOVVLAP
WPHI=WEST-XOVVLAP
C
C---------------------- varn is a constant for input to subroutine
C
prdt. varn should be change to an array for
C
Corresponding individual data points if each
C
data point or group of data points need to have
C
Individually different error variances.
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,
> wp =',wp,
write (21,*) 'xll =',xll,' xup =',xup,' yll =',yll,' yup =',yup
write (21,*) 'dxx =',dxx,' dy =',dy
write (21,*) 'error variance =',varn,
> ' standard deviation error =',estd
C
C-3
INPUT ADJUSTED MAGNETIC DATA AND SELECT DATA FOR
THE PREDICTION

np=(nxpl+1)*(nypl+1)
do 15 i=1,np
  ifc(i)=0
continue

AMEAN=0.0
totpts=0
20 read (10,*,end=30) thi,phi,rad,anomaly
totpts=totpts+1
THI=90.0-THI
IF(THI.GE.THIS) GO TO 20
IF (THI.LE.THIN) GO TO 20
IF (PHI.LT.0.0) PHI=PHI+cross
IF(PHI.LE.WPHI) GO TO 20
IF (PHI.GE.EPHI) GO TO 20
NPTS=NPTS + 1
AMEAN=AMEAN + anomaly
RV=PHI-WPHI
RV=THI-THIS
IX AND IY IDENTIFIES THE BLOCK TO WHICH DATA FALL INTO
AND IYJX ASSIGN AN IDENTIFIER TO DATA CORRESPONDING TO
THAT BLOCK

IY=INT((RY-YLL)*DYY)+1
JX=INT((RX-XLL)*DXX)+1
IYJX=(IY-I)*NXPI+JX

IFC = COUNTER VECTOR, STORES NUMBER OF DATA PER
SORT ELEMENT

IFC(IYJX)=IFC(IYJX)+1

thi(npts)=thi
phi(npts)=phi
radd(npts)=rad
anom(npts)=anomaly

GO TO 20
30 continue
if (npts .le. i) stop

AMEAN=AMEAN/FLOAT(NPTS)
WRITE(21,*) 'total points selected =',npts
WRITE (21,*) 'total points read =',totpts
WRITE (21,*) 'mean of selected points =',amean
WRITE (*,*) 'finished reaching data set'

IH = POINTER VECTOR, FOR CORRESPONDING BLOCK ACHIEVES
A VALUE EQUAL TO SUM OF DATA IN PREVIOUS BLOCKS + 1

ND=NXPI*NYP1
IH(1)=1
DO 87 I=2,ND
  1 I=I-1
  IH(I)=IFC(I)+IH(I)
87 CONTINUE

TX,TY,RTHI,RPHI,RAAD,TANO,VARN ARE NUMBERED FOR
CORRESPONDING DATA, IN EACH BLOCK NUMBERING STARTS
WITH IN VALUE FOR THAT BLOCK AND INCREMENTED BY 1 FOR
NEXT DATA IN THE BLOCK

the mean anomaly value is removed here rather than in subroutine prdt.
also the sum of squares is calculated here and transferred to
subroutine prdt.

sumqr=0.0
DO 85 I=1,NPTS
  rx=phl(I)-phi
  ry=thi(I)-this
  IY=INT((RY-YLL)*DYY)+1
  JX=INT((RX-XLL)*DXX)+1
  IYJX=(IY-I)*NXPI+JX
  NUM=IH(IYJX)
  TANO(NUM)=anom(I)-mean
  sumqr=sumqr+DBLE(tano(num))**DBLE(tano(num))
  RTHI(NUM)=thi(I)
  RPFI(NUM)=phi(I)
  RRAAD(NUM)=radd(I)
  IH(IYJX)=IH(IYJX)+1
85 CONTINUE

IH(I) ATTAINS THE VALUE EQUAL TO NUMBER OF SAMPLE
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, NP1, NXP1, X, Y, TP, RTHI, RPHI, DIST,
   . \ VARN, NTH, TANO, NORTH, SOUTH, EAST, WEST, IH, DXX, DYY, DP,
   . \ DL, AMEAN, RDAD, 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 INCREMENTING TO NORTH.

if (yesno .eq. 'y') then
   np=nxp1*nypl
do 380 i=1,np
totgrid=tp(i)+totgrid
   continue
avgrid=totgrid/real (np)
do 390 i=1,np
   tp(i)=tp(i)-avgrid
   continue
write (21,*) 'total mean removed from the grid = ',avgrid
endif

WRITE (21,9600) WEST,DL
write (20,*) np1
write (20,*) nypl
write (20,*) south
write (20,*) west
write (20,*) dl
P=SOUTH
DO 400 i=1,nypl
400 ID=(i-1)*NXP1+1
   ID2=ID+NXP1-1
WRITE (21,9601) P, (TP(J),J=IDI,ID2)
WRITE(21,' (6F13.5)') (TP(J),J=IDI,ID2)
P=P+DP
C
WRITE (21,9602) WEST,DL
write (20,*) np1
write (20,*) nypl
write (20,*) south
write (20,*) west
write (20,*) dl
P=SOUTH
DO 420 i=1,nypl
420 ID=(i-1)*NXP1+1
   ID2=ID+NXP1-1
WRITE (21,' (6F13.5)') (SEP(J),J=IDI,ID2)
P=P+DP
C
9600 FORMAT(/,'PREDICTED Z-AXIS VALUES',/,
   . 'STARTING LONGITUDE=',F9.3,' INCREMENT=',F5.2,/) 9601 FORMAT('SEP=',F8.2,F6.2,F8.2) 9602 FORMAT(/,'PREDICTED STANDARD DEVIATIONS',/,
   . 'STARTING LONGITUDE=',F9.3,' INCREMENT=',F5.2,/) 9603 FORMAT(/,'PREDICTED Z-AXIS VALUES',/,
   . 'STARTING LONGITUDE=',F9.3,' INCREMENT=',F5.2,/) 9604 FORMAT(/,'PREDICTED STANDARD DEVIATIONS',/,
STOP
END
C

SUBROUTINE PRDT(NP1,MY,MX,X,Y,TP,THI,THII,THI3,THII,THI2,THI1,THI0,DXX,DYY,DP,DL,
   . AMEAN,RC,NCPP,ELEV,SEP, sumsqr)
IMPLICIT REAL (A-H,O-Z)
REAL NSEe, LAT, LONG,MI
REAL scale, covm, fact,sumsqr,var,nse,dummy,b,
   t1,t2,tem,tem2,cov
DIMENSION LAT(11),LONG(11),DIST(1),MI(1),RC(1),
   X(1),Y(1),TP(1),IH(1),SEP(1)
DIMENSION COVM(10,10),B(110),T1(110),T2(110),LCC(110)
COMMON/TWO/ SCALE
DATA DMAX/6E6/
EQUIVALENCE (B (i }, T1 (i) )

--- subroutine description
this subroutine is one big do loop that progresses through
the grid nodes to determine the magnitude at each node.

nse=dlile(nse)
CONJ=51,2957795
NO-MY*MX
NAD=0
YLL=0.0
C. Distance to all points in the window from prediction point

D. Not enough data in first window, so consider next window

E. Write the bad point to file

---

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 id1 id2 ',
> 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 J=1,NCPP
LCC(I)=0
IY=INT ((YPP-YLL) *DYY) +I
JX=INT ((XPP-XLL) *DXX) +J
ID1=IND+I
XPP=X (IND)
YPP=Y (IND)
DO 8 I=1,NCPP
LCC (I)=0
IY=INT ((YPP-YLL) *DYY) +I
JX=INT ((XPP-XLL) *DXX) +J
ID1=IND+I
ID2=ID1 +I
IF((ID2-ID1).GE.NCPP) GO TO 100

DO 211 IC=1,MY
NDATA=0
IY1=IY+IC
IY2=IY+IC
IF (IY1.LT. I) IY1=I
IF (IY2.GT.MY) IY2=MY
JX1=X+IC
JX2=X+IC
IF (JX1.LT. J) JX1=J
IF (JX2.GT.MX) JX2=MX
DO 18 IL=IY1, IY2
IYJX1-(IL-I) *MX+JX1
IDI=ID1 +I
IYJX2-(IL-I) *MX+JX2
ID2=IDI +I
NDATA=NDATA+1
CONTINUE
IF (NDATA.GE.NCPP) GO TO 100
NBAD=NBAD+1
write (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-I) *MX+JX1
ID1=IND+I
IYX2=IYX1+JX2-JX1+1
ID2=ID1 +I
IF (ID2.LT.ID1) GO TO 211

106 DO 210 I=1, ID2
DELC=COS(LAT(I)/CON1)*COS(P/CON1)+SIN(LAT(I)/CON1)*
> SIN(P/CON1)*COS((LONG(I)-Q)/CON1)
> PART1=ELEV**2 + RC(I)**2
> PART2=2.0*ELEV*RC(I)
> DARCO-PART1-PART2*DELC
IF (DARGU.LE.0.0) THEN
DIST(I)=0.0

C-6
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=1, NCPP
IF (LCC(J), GT, 0) GO TO 260
DMIN=DMIN
DO 253 IC=IY1, IY2
IF (IY1, EQ, IY2) GO TO 107
IYX1=IC-1
IYX2=IYX1+1
ID1=IN(IYX1)
ID2=IN(IYX2)-1
IF (ID2, LT, ID1) GO TO 253
DO 250 I=ID1, ID2
IF (DIST(I), GT, 250, 250, 250)
DMIN=DIST(I)
LMIN=I
250 CONTINUE
253 CONTINUE
DIST(LMIN) = DMIN
LCC(J) = LMIN
260 CONTINUE
IF (LCC(NCPP), NE, 0) GO TO 258
GO TO 220
258 CONTINUE
C FORM COVARIANCE MATRIX
C
DO 280 I=1, NCPP
M=LCC(I)
DO 280 J=1, M
COVM(I,J) = DBLE(VAR+NSE)
K=1+I
DO 280 J=K, NCPP
N=LCC(J)
DEL0=COS(LAT(M)/CON1)*COS(LAT(N)/CON1)+SIN(LAT(M)/CON1)*SIN(LAT(N)/CON1)*COS((LONG(M)-LONG(N))/CON1)
0.5*PART1=RC(M)**2 + RC(N)**2
0.5*PART2=2.0*RC(M)*RC(N)
DARGU=PART1-PART2+DEL0
IF (DARGU, LE, 0.0) THEN
DIS=0.0
ELSE
DIS=SQRT(DARGU)
END IF
280 CONTINUE
COVM(NCPP, NCPP) = DBLE(VAR+NSE)
C INVERT COVARIANCE MATRIX
COVM(I,I) = 0.0/COVM(I,I)
DO 340 J=1, NCPP
L=1+I
DO 340 J=1, L
B(J) = 0.0
DO 310 K=1, J
B(J) = B(J) - COVM(K, J) * COVM(L, K)
Dummy = COVM(L, L)
DO 310 K=1, L
Dummy = Dummy - COVM(K, J) * Dummy
DO 330 K=1, L
COVM(K, J) = Dummy
340 CONTINUE
DO 410 K=1, J
COVM(J, K) = Dummy
410 CONTINUE

C-7
PART2=2.0*ELEV*RC(N)
DARGU=PART1+PART2*DELC
IF (DARGU.LE.0.0) THEN
  DIS=0.0
ELSE
  DIS=SQRT(DARGU)
END IF

CALL COVINT (DIS,FACT,COV)

DO 430 I=1,NCPP
  TEM=0.0
  DO 420 J=1,NCPP
    TEM=COVM(J,I)*TI(J)+TEM
  420 TEM=TEM

  T2(I)=TEM

  DO 440 I=1,NCPP
    M=LCC(I)
    TEM=REAL(M)*T2(I)+TEM
    TEM2=TI(I)*T2(I)+TEM2
  440 CONTINUE

C COMPUTE ANOMALY ESTIMATE AND STANDARD ERROR OF PREDICTION

TP(IND)=REAL(TEM)+AVERAGE
IF (VAR.LT.TEM2) GO TO 444
SEP(IND)=SQRT(VAR-TEM2)
GO TO 445
444 SEP(IND)=1.5
WRITE (21,446) VAR, TEM2
445 CONTINUE
10 CONTINUE
RETURN
END

SUBROUTINE COVINT (DIS,FACT,COV)
IMPLICIT REAL (A-H,O-Z)
REAL DIS
INTEGER NCPP
DIMENSION CNN(500), DNUM(500)
COMMONE/ CNN, DNUM, NCPP

C IMPLEMENTATION OF COVARIANCES

TCOV=REAL(NCOV-1)
R1=REAL(DIS)
IF (R1.LT.TCOV) GO TO 1
COV=CNN(NCOV)*FACT
WRITE (21,100) R1
RETURN
1 IF (R1.GT.0.0001) GO TO 6
COV=CNN(I)*FACT
RETURN
6 DO 2 I=1,NCPP
  IF (R1.LT.DNUM(I)) GO TO 2
  CONTINUE
2 I=1
  FPINT=I1-DNUM(I)
  FDINT=I1-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,
prect,delte,cuthi,cutlo,klag,mincc,maxcc,short,long,
xcolat,ycolat,xlong,ylong, xgridspc,ygridspc,
minccin,maxccin,xmlag
integer xpassno, ypassno,
llb,cc, npass, lmean, head,spass, swind,
wind,numfile, nxout, nyout, cmwind, cnwind, uwind,
numcc(10),numlhb(10),numd(10),numud(10),numrtp(10),
numderiv(10)
common xdata(512,512), ydata(512,512)
common /rowcol/ xrow, yrow, ycol, xmlag
common /ftfft/ nxout, nyout, prect, lmean, fold, iltypefold
common /lbfft/ delta, cuthi, cutlo, klag, npass, wind
common /ccfft/ mincc, maxcc, minccin, maxccin, cmwind, cxmlag
common /reals/ xdata, ydata
common /stricking/ anl, ang2, spass, wind, xmlag
common /udecont/ uddelta, zcmean, udxlag, udxwind
common /xyzerot/ xymean, ymmean, nxwind, nywind
common /rtp/ xrow, xcol, xrc, xcdata(512,512), ydata(512,512)
common /compa/ xdata, ydata

---program description---

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
different directions, 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
correlation coefficient, however, in 1-d 1 have not yet implemented the continuation,
and array lhb should only have the value 2 (for the
second position). This code was an extensive modification of an earlier code
named format.

write ('*,' 'OUTPUT FILE OF STATISTICS AND INFORMATION'
read (*,9990) filename
open (25, file=filename, form='formatted')
write (*, 'numfile'
9989 format ('1 IF YOU HAVE ONLY ONE FILE TO BE FOURIERED',
'2 IF YOU HAVE TWO FILES TO BE COMPARED')
read (*, numfile)
write (*, 'numfile'
9988 format ('ENTER THE MAXIMUM NUMBER OF TIMES TO RUN EACH FILTER',
'BANDPASS, CORR.COEFF., STRIKE-DIP, UP/DN CONT., RTP, DERIV')
read (*, lhb, lcc, lsd, lsd, ldxlag, ldwind)

if (lhb .eq. 0) then
write (*, 'BANDPASS SECTION:
write (*, 'enter placement values in order ie. 2 3 5'
write (*, 'do not repeat these values elsewhere'
write (*, '1-first, 3-third, 5-fifth etc...'
read (*, numlhb(1),1=1,1, lhb)
write (*, 'numlhb',read (*, 'numderiv(10)
write (*, 'numderiv(10)

9993 format ('DELTA.....GRID INTERVAL IN MAP UNITS (0.0 degrees)'
'SHORT....SHORTEST WAVELENGTH TO BE PASSED')
MUST BE AT LEAST 2*DELTA (2.0 degrees)/
LONG.......LONGEST WAVELENGTH TO BE PASSED/
MUST BE LARGER THAN SHORT'/
NPASS......-1 TO REJECT WAVELENGTHS BETWEEN SHORT'/
AND LONG'/
1 TO PASS WAVELENGTHS BETWEEN SHORT AND'/
LONG'/
NOTE : WAVELENGTH = 1/WEAVELENGTH 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-1.0/SHORT
CUTLO-1.0/LONG
RCUTLO=999999.99
IF(CUTLO .GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
RCUTHI=1.0/CUTHI
WAVLEN=2.0*DELTA
FNQI-1.0/WAVLEN
WRITE (25,9987) FNQI,WAVLEN,CUTLO,RCUTLO,CUTHI,RCUTHI
9987 FORMAT('NYQUIST WAVENUMBER = ',f10.5,
  'CYCLES PER DATA INTERVAL'/,
  'NYQUIST WAVELENGTH = ',f10.5,
  'LENGTH INTERVALS'/,
  'LOW WAVE# CUTOFF OF IDEAL FILTER - ',f10.5,
  'CYCLES PER DATA INTERVAL '/,F15.5,
  'WAVELENGTH EQUIVALENT'/,
  'HIGH WAVE# CUTOFF OF IDEAL FILTER - ',f10.5,
  'CYCLES PER DATA INTERVAL '/,F15.5,
  'WAVELENGTH EQUIVALENT'/)
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 (*,*) (numud(i),i=1,lud)
write (*,9985)
9985 format('udDELTA = GRID INTERVAL IN MAP UNITS (0.0)/,
> 'EDON = 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 (NON), 1 (RECTANGULAR), 2 (TRIANGULAR)/,
> '3 (H-T), 4 (PARZEN)'/,
> 'udXLAG - SMOOTHING PARAMETER FOR WINDOWING FILTER'/,
> ' IN SPATIAL DOMAIN. DETERMINES WHAT PERCENTAGE'/,
> 'delta zcon udnwind udxlag')
read (*,*) uddelta, zcon, udnwind, udxlag
endif
c
if (irtp .gt. 0) then
write (*,*) 'REDUCTION TO POLE SECTION:'
write (*,*) 'enter placement values in order ie. 2 3 5'
write (*,*) 'do not repeat these values elsewhere'
write (*,*) 'l-first, 3-third, 5-fifth etc...
read (*,*) (numrtp(i),i=1,irtp)
write (*,9983)
9983 format ('AZM - AZIMUTH OF Y AXIS MEASURED IN DEGREES'/,
> 'CLOCKWISE FROM TRUE NORTH'/,
> 'DEC - AVERAGE DECLINATION OF THE INPUTTED'/,
> 'ANOMALY DATA'/,
> 'XINC - AVERAGE INCLINATION OF THE INPUTTED'/,
> 'AZM dec xinc')
read (*,**) azm, dec, xinc
endif
c
if (ideriv .gt. 0) then
write (*,*) 'DERIVATIVE SECTION:
write (*,*) 'enter placement values in order ie. 2 3 5'
write (*,*) 'do not repeat these values elsewhere'
write (*,*) 'l-first, 3-third, 5-fifth etc...
read (*,*) (numdariv(i),i=1,ideriv)
write (*,9984)
9984 format ('xyzDELTA - GRID INTERVAL'/,
> 'NTH - ORDER OF SPATIAL DERIVATIVE TO PERFORM'/,
> 'ON THE DATA'/,
> 'NWAY - DIRECTION IN WHICH TO CALCULATE THE'/,
> 'DERIVATIVE'/,
> '0 - VERTICAL DERIVATIVE'/,
> '1 - HORIZONTAL DERIVATIVE IN "X" DIRECTION'/,
> '2 - HORIZONTAL DERIVATIVE IN "Y" DIRECTION'/,
> 'xyzdelta nth nway')
read (*,**) xyzdelta, nth, nway
endif
c
c
write (*,9992)
9992 format ('THE FOLLOWING REFERS TO FFT AND IFFT'/,
> 'NUMBER OF COLUMNS AND ROWS OF FFT ARRAY'/,
> 'AT A POWER OF 2: (256 128) (2 16 32 64 128 256 etc)'/,
> 'TYPE OF INPUT ARRAY INDICATES TYPE OF FOLDING TO BE USED'/,
> '0 IF A POLAR REGION, ie. E AND W EDGES ARE SAME'/,
> '1 IF A NON-POLAR REGION, ie. E AND W EDGES NOT SAME'/,
> 'PERCENT OF EACH EDGE OF INPUT ARRAY TO'/,
> 'BE FOLDED OUT: (0.1 TO 99.9)'/,
> 'PERCENT OF EACH EDGE OF FOLDED OUT OR NORMAL ARRAY'/,
> 'TO BE SMOOTHED TO ZERO: (0.1 TO 49.9)'/,
> '0 DO NOT ADD MEAN TO IFFT DATA'/,
> '1 ADD MEAN TO IFFT DATA'/,
> 'nxout nyout itypefold fold prcnt imean')
read (*,**) nxout, nyout, itypefold, fold, prcnt, imean
endif
c
if (nxout .gt. 512 .or. nyout .gt. 512) then
write (*,8999) nxout, nyout
8999 format (I9,'SORRY',I6,1x,'OR',I6,1x,'IS GREATER THAN 512 THE',
> 'SIDE OF ARRAYS SET'/' IN THE SOURCE CODE ',
> 'YOU NEED TO ACCESS SOURCE CODE AND MAKE CHANGES')
stop
c
c
write (*,*) 'INPUT FILE 1'
read (*,9990) filename
9990 format (A80)
open (10, file=filename, status='old', form='formatted')
if (numfile .eq. 2) then
write (*,*) 'INPUT FILE 2'
read (*,9990) filename
open (11, file=filename, status='old', form='formatted')
endif
read (*,9990) filename
open (20, file=filename, form='formatted')
if (numfile .eq. 2) then
  write (*,*) 'OUTPUT OF FILE 1'
endif
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
endif
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
call forwardft (1, xmean, xpassno)
ymean=0.0
if (numfile .eq. 2) call forwardft (2, ymean, ypassno)

lottime=ihb+icc+isd+ldu+irt+lderlv
i=1,lottime
  do j=1,ihb
    if (numihb(j) .eq. 1) then
      call filter (1)
    endif
    if (numfile .eq. 2) call filter (2)
    goto 888
  enddo
  do j=1,loc
    if (numloc(j) .eq. 1) then
      call correlate(xpassno, ypassno)
    endif
  enddo
  do j=1,isd
    if (numisd(j) .eq. 1) then
      call strkpas(1)
    endif
    if (numfile .eq. 2) call strkpas(2)
    goto 888
  enddo
  do j=1,ldu
    if (numldu(j) .eq. 1) then
      call upcon(1)
    endif
    if (numfile .eq. 2) call upcon(2)
    goto 888
  enddo
  do j=1,irt
    if (numirt(j) .eq. 1) then
      call mag2pol(1)
    endif
    if (numfile .eq. 2) call mag2pol(2)
    goto 888
  enddo
  do j=1,lderlv
    if (numlerlv(j) .eq. 1) then
      call deriva(1)
    endif
    if (numfile .eq. 2) call deriva(2)
    goto 888
  enddo
  continue
enddo

888 continue
subroutine forwardft (num, mean, passno)
  integer num, xrow, xcol, yrow, ycol, nxout, nyout, passno,
          > row, col
  real xdata(361,361), ydata(361,361), prcnt, mean
  complex xcdata(512,512), ycdata(512,512)
  call /fftlfft/ nc, nh, prcnt, xdata, ydata, xcdata, ycdata
  double precision TSUM
  EQUIVALENCE (X(I,I),H(I,I))
  TSUM=0.0D0
  if (num .eq. 1) then
    row=xrow
    col=xcol
    do 50 i=1,row
         do 50 j=1,col
            x(i,j,1) = xdata(j,1)
            tsum=tsum+x(i,j,1)
      50 continue
  elseif (num .eq. 2) then
    row=yrow
    col=ycol
    do 80 i=1,row
         do 80 j=1,col
            x(i,j,1) = ydata(j,1)
            tsum=tsum+x(i,j,1)
  80 continue
  endif
end

*********************************************************************
C     PROGRAM SPA2FRQ
C     PROGRAM SPA2FRQ TRANSFORMS AN N X N MATRIX OF SPACE-DOMAIN
C     AMPLITUDES INTO THE N X N MATRIX OF WAVE NUMBER DOMAIN
C     IN-CORE PROGRAMS TO PERFORM SPECTRAL OPERATIONS (UPCON, MAGPOL,
C     BANDPASS, STRKPASS, DERIV). FUNCTIONS PERFORMED BY THIS PROGRAM
C     INCLUDE:
C     - REMOVAL OF THE MEAN FROM THE DATA
C     - OPTIONAL WINDOWING OF THE EDGES OF THE DATA SET
C     - PADDING OF THE DATA SET WITH ZEROS TO ACHIEVE NECESSARY
C     SIZE (A POWER OF TWO)
C     - FORWARD TRANSFORM OF THE DATA
C     .....REQUIRED SUBROUTINES :
C     FFT2D, 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 Mineralogy
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
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

NXIN=COL
NYIN=ROW
ICOL=NXIN
ICOL=NYIN
XMEAN1=SUM/FLOAT(NXIN*NYIN)
DO 210 IY=1,NYIN
   DO 210 IX=1,NXIN
      X(I, IX, IY) = X(I, IX, IY) - XMEAN1
210 CONTINUE

WRITE (25, 1020) XMEAN1

C ....... WINDOW THE EDGES VIA DATWND

CALL DATWND(PRINT, NXIN, NYIN, NXOUT, NYOUT, fold, itypefold)

C ....... MATRIX IS NOW ZERO FILLED TO NXOUT BY NYOUT SIZE

CALCULATE AND REMOVE THE MEAN INTRODUCED BY TAPERING

TSUM=0.0
DO 214 IY=1, NYOUT
   DO 214 IX=1, NXOUT
      TSUM=SUM=SUM*SUM(1, 1, IX, IY)
214 CONTINUE

XMEAN2=TSUM/FLOAT(NXOUT*NYOUT)
DO 215 IY=1, NYOUT
   DO 215 IX=1, NXOUT
      X(I, IX, IY) = X(I, IX, IY) - XMEAN2
215 CONTINUE

WRITE(25,1020) XMEAN2

WRITE(25,1080) XMEAN

WRITE(25,**) passno, xmean1, xmean2, xmean

C ....... TRANSFORM DATA TO THE WAVENUMBER DOMAIN

NX=NXOUT
NY=NYOUT
CALL FFT2D(NX, NY, -1)

mean=xmean
If (num .eq. 1) then
   do 500 IY=1, NY
      do 500 IX=1, NX
         xdata(IX, IY) = h(IX, IY)
500 continue
else (num .eq. 2) then
   do 580 IY=1, NY
      do 580 IX=1, NX
         ydata(IX, IY) = h(IX, IY)
580 continue
endif
SUBROUTINE FFT2D (NX, NY, NSIGN)

"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.

"NSIGN" = DIRECTION OF DESIRED TRANSFORMATION
  == 1 INVERSE TRANSFORM (FREQUENCY TO SPATIAL)
  ==-1 FORWARD TRANSFORM (SPATIAL TO FREQUENCY)

COMMON H (512,512)
COMMON CTEMP (512)
COMPLEX H, CTEMP

SIGNI=FLOAT (NSIGN)
IF (IABS (NSIGN) .NE. 1) THEN
  WRITE (6,105)
  STOP
ENDIF

C ..... OPERATE BY ROWS
C
DO 101 IY=I, NY
   CALL FORK (NX, H(I, IY), SIGNI)
C
C ..... OPERATE BY COLUMNS
C
DO 104 IX=I, NX
   CALL FORK (NY, CTEMP, SIGNI)
   DO 103 IY=I, NY
      H(IX, IY) = CTEMP(IY)
   103 CONTINUE
104 CONTINUE

RETURN

105 FORMAT (5X, "NSIGN" MUST EQUAL +1 OR -1 FOR "FFT2D", FATAL)

END

SUBROUTINE FORK (LXX, CX, SIGNI)


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)
  == 1. FORWARD TRANSFORM (SPATIAL TO FREQUENCY)

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

COMPLEX CX(LXX), CW, CTEMP, CON2

IX=LXX
LXH=IX/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=1XH
102 IF (J.GT.M) THEN
  J-J-M
  M-M/2
ENDIF
103 CONTINUE
104 L=1
ISTEP=2*
CON2=(0,0,3,14159265)/FLOAT(L)+SIGNI
DO 105 M=1,L
CM=C*EXP(CON2*FLOAT(M-1))
DO 105 I=1,L,ISTEP
CM=CM*CXI(I-2)
CX(I+1)=CX(I+1)-CM
105 CX(I)=CX(I)*ISTEP
DO 106 I=1,L
CX(I)=CX(I)*SC
106 RETURN
END

C************************************************************************
C SUBROUTINE DATWND (PRCNT, NXIL, NYII, NX, NY, fold, itypefold)
C************************************************************************
C *
C "DATWND" MULTIPLIES THE INPUT F(I,×,Y) BY A HALF BELL OF A HAMMIM-
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)
COMMON F(2,512,512)
C
nxl=nxll
nyl=nyll
C-------------------- fold out the data based on percentage
C if (fold.gt.0.0 .and. fold.lt.100.0) then
C  MK=Int(fold*FLOAT(NXIL)/100.0+0.5)
  MY=Int(fold*FLOAT(NYII)/100.0+0.5)
  if (nx=xl .gt. nx) nx=(nx-nxl)/2
  if (ky=yl .gt. ny) ky=(ny-nyl)/2
  do j=1,nyl
    do i=1,nxl
      holdme(i,j)=f(1,i,j)
    enddo
  enddo
C-------------------- fold out the columns in each row:
C if itypefold is 1 then the data is considered to
C 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
    do i=1,nxl+nxk
      if (i.le.kx) f(1,i,j)=holdme(kx-i+1,j)
      if (i.gt.kx .and. i.le.(kx+nxk)) f(1,i,j)=holdme(1-kx,j)
      if (i.gt.(kx+nxk)) f(1,i,j)=holdme((2*nxk+kx+1-i),j)
    enddo
  enddo
C-17
if (itypefold .eq. 0) then
  do j=1, nyl
    do i=1, nxl+xx+xx
      if (i .le. 1) f(i,j) = holdme(i-xl+kk+kx)
      if (i .gt. 1 .and. i .le. (xl+xx)) f(i,j) = holdme(i-kx, j)
      if (i .gt. (xl+xx)) f(i,j) = holdme(i, (2*xx+xx+kx-J))
    enddo
  enddo
endif

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

c------------------------ fold out the rows in each column

do i=1, nxl+xx+kx
  do j=1, nyl+ky+ky
    if (j .le. ky) f(i,j) = holdme(i,ky-J+J)
    if (j .gt. ky .and. j .le. (ky+nyl)) f(i,j) = holdme(i, ky-J)
    if (j .gt. (ky+nyl)) f(i,j) = holdme(i, (2*nyl+ky+ky-J))
  enddo
nyl = nyl + ky+ky
333 nxl = nxl + 2*xx
enddo

if (prcnt.gt.0.0 .and. prcnt.lt.50.0) then
  KX = IFIX (PRCNT*FLOAT (NXI)/100.0+0.5)
  KY = IFIX (PRCNT*FLOAT (NYI)/100.0+0.5)

  IF (NYI .NE. 1 .AND. KY .NE. 0) THEN
    RKYPI = 3.14159265/FLOAT (KY)
    DO IY=1, KY
      FACTOR = 0.5*(1.0+COS (FLOAT (KY-IY+I) *RKYPI))
      IYY = KY-IY+I
      DO IX=1, NXI
        F(I, IX, IY) = F(I, IX, IYY) * FACTOR
      END
    END
  ENDIF

  IF (KX .NE. 0) THEN
    RKXPI = 3.14159265/FLOAT (KX)
    DO IX=1, KX
      FACTOR = 0.5*(1.0+COS (FLOAT (KX-IX+I) *RKXPI))
      IXX = KX-IX+I
      DO IY=1, NYI
        F(I, IXX, IY) = F(I, IXX, IY) * FACTOR
      END
    END
  ENDIF

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

C...... center and ZERO OUT REMAINDER OF ARRAY

nxhalf = (nx-nxl)/2
nyhalf = (ny-nyl)/2

do i=1, nxl
  do j=1, nyl
    holdme(i,j) = f(i, j)
  enddo
endo
do i=1, nxhalf
  do j=1, nyl
    f(i,j) = 0.0
  enddo
endo
do i=nxhalf+1, nxhalf+nxl
  do j=1, nyl
    if (j .le. nyhalf) f(i,1) = 0.0
    if (j .gt. nyhalf .AND. j .le. nyl)
      f(i,j) = holdme(i-nxhalf, j-nyhalf)
    if (j .gt. nxhalf+nyl) f(i,1) = 0.0
  enddo
endo
do i=nxhalf+nxl+1, nx
do i=1,ny
   f(i,1,1)=0.0
enddo
c
RETURN

FORMAT('smoothed','14,' values on both x edges',/,'14,' y')

FORMAT(H..''PRCNT''..',F7.3,' OUTSIDE OF PROPER RANGE',/,' NO WINDOWING PERFORMED: "DATWND"')

END subroutine filter (num)

integer n_m, npass,imean,nwind, nxout, nyout,
        row, col
real prcnt,xlag,delta,cutli,culto
complex xcdata(512,512),ycdata(512,512)
common /fftlfft/ nxout, nyout,prcnt,imean,ifold,itypefold
common /comps/ xcdata,ycdata
common /lhbflt/ delta,culto,xlag, npass, nwind
COMMON H(512,512)

C**********************************************************************
C    *    PROGRAM : BANDPASS
C    *    FILTERS A UNIFORMLY GRIDDED ARRAY OVER A RANGE WAVENUMBER.
C    *    INPUT IS THE WAVENUMBER DOMAIN TRANSFORM AS OUTPUT BY SPA2FRQ.
C    *    AN IDEAL FILTER IS CONSTRUCTED IN THE WAVENUMBER DOMAIN, WINDOWED
C    *    IN THE SPACE DOMAIN, THEN TRANSFORMED BACK INTO THE WAVENUMBER
C    *    DOMAIN TO BE MULTIPLIED BY THE INPUT TRANSFORM.
C    *    REQUIRED SUBROUTINES : BNDPAS, FFT2D, FORK, STORE, WINDOW
C    *    DIMENSIONING REQUIREMENTS :
C    *    H(N,N) WHERE N IS THE NUMBER OF COLLUMNS AND ROWS OF THE
C    *    INPUT AND OUTPUT TRANSFORMED MATRIX. N MUST BE AN
C    *    INTEGRAL POWER OF TWO (2,4,8,16...).
C    *    AUTHOR: JON REED, PURDUE UNIVERSITY, DECEMBER 1980.
C    *    REVISIONS BY STEVE MATESKON AND JEFF LUCIUS,
C    *    OHIO STATE UNIVERSITY, JULY 1984.
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    *    USEABLE TO JUST ABOUT ANYBODY! REVISED 21 APR 90
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 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    CREATE FILTER AND STORE IN ARRAY H
C    CALL BNDPAS (CUTLO,CUTHI, NPASS, DELTA, NX, NY)
C    CALL STORE (NX, NY)
C
C    CREATE SMOOTHED FILTER
C    IF(XLAG.GT.0.0 .AND. XLAG.LE.100.0) THEN
C       IF(NWIND.GT.0.0 .AND. NWIND.LE.4) THEN
C          CALL FFT2D (NX, NY, 1)
C    END
CALL WINDOW (NX, NY, XLAG, NWIND)
CALL FFT2D (NX, NY,-1)
END IF
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)
CONTINUE
END IF
ENDIF

C
C IF (num .eq. 1) then
do 500 I=1,row
do 500 J=1,col
xcdata(J,I) = xcdata(J,I)*h(J,I)
500 continue
elseif (num .eq. 2) then
do 580 I=1,row
do 580 J=1,col
ycdata(J,I) = ycdata(J,I)*h(J,I)
580 continue
endif
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 WAVENUMBERS BETWEEN THE 2 WAVENUMBERS
C = 1 PASS WAVENUMBERS BETWEEN THE 2 WAVENUMBERS
C "DELTA" DATA GRID INTERVAL, IN MAP UNITS
C "NX" NUMBER OF ROWS (POWER OF 2 GE "ICOL", 16,32,ETC)
C "NY" NUMBER OF ROWS (POWER OF 2 GE "IROW", 16,32,ETC)
C
C*********************************************************
C COMMON H(512,512)
COMPLEX H, ZERO, ONE
DIMENSION A(2)
DATA A/4HPASS, 4HCUT /

C CUTHI=CCUTHI
CUTLO=CCUTLO
RCUTLO=999999.99
IF (CUTLO.GE. 0.0000001 ) RCUTLO= 1.0/CUTLO
RCUTHI=1.0/CUTHI
WAVLEN=2.0*DELTA
FNQ1=1.0/WAVLEN
WRITE (25,112) FNQ1, WAVLEN, CUTLO, RCUTLO, CUTHI, RCUTHI, NPASS
C IF (NABS(NPAS.S.NE.1)) THEN
WRITE (6,151)
STOP
ENDIF
IF (CUTHI.GT.FNQ1.OR.CUTHI.LE.CUTLO.OR.CUTLO.LE.0.0) THEN
WRITE (6,151)
STOP
ENDIF
C
NXX=NX+2
NX2=(NX/2)+1
NY2=(NY/2)+1
ANY2=FLOAT(NY2)
ZERO = (0.0,0.0)
ONE = (1.0,0.0)
WAY = A(1)
C
IF (NPASS.NE.1) THEN
ZERO = (1.0,0.0)
ONE = (0.0,0.0)
WAY = A(2)
ENDIF
C
RHLY2 = (FNQ1/(ANY2*CUTHI))**2
C
NHY = CUTHI * WAVLEN * ANY2 + 1.0
NHY = AMINO (NHY, NY2)
CLOB = FLOAT (NX2) * WAVLEN * CUTHI
CHIX = FLOAT (NX2) * WAVLEN * CUTHI

C IF (CUTLO .LE. 0.000001) THEN
C WRITE (6, 152) WAY
RLOWY = 0.0
ENDIF
ELSE
C IF (FNQI - CUTHI .LT. 0.00001) THEN
C WRITE (6, 153) WAY
RLOWY = (FNQI / (ANY2 * CUTLO)) ** 2
ENDIF
C ELSE
C WRITE (6, 154)
RLOWY = (FNQI / (ANY2 * CUTLO)) ** 2
ENDIF
ENDIF

C ..... "ZERO" OUT THE PART OF ARRAY TO BE ALTERED
C
DO 35 IY = 1, NY2
DO 35 IX = 1, NX
H (IX, IY) = ZERO
35 CONTINUE
C
C ..... OPERATE ON ROWS WHERE SOME WAVENUMBERS ARE .LE. CUTLO
C
IF (NLOWY .NE. 0) THEN
MINS = 1
MAXX = MAX2
IF (NLOWY .EQ. 1) THEN
MINS = NX2 * WAVLEN + 2.0001
ENDIF
ENDIF
DO 102 IY = 1, NLOWY
Y2 = FLOAT (IY - 1) ** 2
MINS = MINS
IF (CUTLO .GT. 0.000001) THEN
MINS = CHIX * SQRT (1.0 - Y2 * RLOWY2) + 2.0001
ENDIF
MAXX = MAXX
IF (FNQI - CUTHI .GE. 0.00001) THEN
MAXX = CHIX * SQRT (1.0 - Y2 * RHIY2) + 1.0001
ENDIF
H (MINS, IY) = ONE
MINS = MINS + 1
ENDIF
IF (MINS .EQ. I) THEN
H (I, IY) = ONE
ENDIF
DO 150 IX = MINS, MAXX
H (IX, IY) = ONE
150 H (IX, IY) = ONE
102 CONTINUE
C
C ..... OPERATE ON ROWS WHERE ALL WAVENUMBERS ARE .GT. CUTLO
C
IF (NNIY .NE. NY2) THEN
LL = NLOWY + 1
DO 200 IY = LL, NHY
Y2 = FLOAT (IY - 1) ** 2
MAXX = MAXX
IF (MAXX .LE. MAX2) THEN
MAXX = MAXX
H (MAXX - IX, IY) = ONE
180 H (IX, IY) = ONE
200 CONTINUE
ENDIF
ELSE
IF (NLOWY + 1 .GT. NY2) RETURN
II = NLOWY + 1
DO 215 IX = II, NY2
DO 215 IY = 1, NX
H (IX, IY) = ONE
215 CONTINUE
ENDIF
RETURN
C
112 FORMAT (/IX, 'NYQUIST WAVENUMBER - ', F10.5, ' CYCLES PER DATA INTERVAL'
> /IX, 'NYQUIST WAVELENGTH = ', F10.5, ' LENGTH INTERVALS'/
> /IX, 'LOW WAVE# CUTOFF OF IDEAL FILTER = ', F10.5,
> ' CYCLES PER DATA INTERVAL', X, F15.5, ' WAVELENGTH EQUIVALENT'/
> /IX, 'HIGH WAVE# CUTOFF OF IDEAL FILTER = ', F10.5,
> )
SUBROUTINE STORE (NNX, NNY)

COMMON H(512, 512)

COMPLEX H

IF (NNY.EQ.1) RETURN

NX=NNX
NY=NNY
NX=NX/2+1
NX=NX/2+2
NY=NY/2+2
NY=NY+1

DO 15 IY=NY, NY+1
     H(1, IY)=H(1, NYL)
     DO 10 IX=2, NX
          H(IX, IY)=CONJG (H(NXX-IX, NYL))

10 CONTINUE

H(NXH, IY)=H(NNH, NYL)

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
LAG=AMAX0 (LAG, 2)
PI=3.14159265
NX=NX+2
NY=NY+2
NX=(NX/2)+1

NXR=FLOAT (NX2)
NX=1.0*FLOAT (NX2)
NY=(NY/2)+1

IF (NY.EQ.1) NY2=1
NY1=1.0*FLOAT (NY2)

RADIUS=FLOAT (LAG) *XNX
RADIUS=FLOAT (LAG) *NXR
RAD1=1.0/RADIUS
RAD2= RADIUS*RADIUS
NRAD=FLOAT (NY2) *RADIUS+1.0001

C-22
C.... APPLY RECTANGULAR WINDOW TO FILTER
C
C IF (NWIND.EQ.1) THEN
   C
   IF (NRAD.NE.0) THEN
      MAX=RADIUS*XNXR+1.0001
   ENDIF
   C
   IF (NRAD.EQ.0) THEN
      H(1,1)=(0.0,0.0)
   MAX=MAX+1
   ENDIF
C
   LL=MAX+1
   DO 155 II=LL,NX2
   H(NXX-II,1)=(0.0,0.0)
155   H(II,1)=(0.0,0.0)
   ENDIF
C
   IF (NRAD.NE.1 .AND. NRAD.NE.0) THEN
      DO 102 IY=2,NRAD
      IYY=NYY-IY
      YLEN2=(FLOAT(IY-1)*YNV)**2
      MAX=SQRT(RAD2-YLEN2)*XNXR+1.0001
      C
      IF (MAX.GE.2) THEN
      H(1,IYY)=(0.0,0.0)
      H(IY,IYY)=(0.0,0.0)
      ENDIF
      C
      LL=MAX+1
      DO 105 II=LL,NX2
      H(NXX-II,IYY)=(0.0,0.0)
      H(II,IYY)=(0.0,0.0)
      105 H(NXX-II,1IYY)=(0.0,0.0)
      ENDIF
C
   WRITE (25,660) XLAG, LAG
   IF (NRAD.EQ.NY2) RETURN
   LL=NRAD+1
   DO 108 I=LL,NY2
      IYY=NYY-I
      H(I,IYY)=(0.0,0.0)
      DO 109 J=2,NX2
      H(J,I)=(0.0,0.0)
      H(NXX-J,1)=(0.0,0.0)
      H(J,IYY)=(0.0,0.0)
      109 H(NXX-J,IYY)=(0.0,0.0)
      108 CONTINUE
C.
C.. APPLY BARTLETT WINDOW TO FILTER
C
C ELSEIF (NWIND.EQ.2) THEN
C
   IF (NRAD.NE.0) THEN
      MAX=RADIUS*XNXR+1.0001
   C
   IF (NRAD.NE.1 .AND. NRAD.NE.0) THEN
      DO 202 IY=2,NRAD
      IYY=NYY-IY
      X1=FLOAT(IY-1)*YNV
      YLEN2= X1*X1
      MAX=SQRT((FLOAT(IY-1)*YNV)**2+YLEN2)
      FACTOR=1.0-X1*RADI
      H(1,IYY)=H(1,IYY)*FACTOR
      H(IY,IYY)=H(IY,IYY)*FACTOR
      ENDIF
C
      IF (NRAD.NE.1 .AND. NRAD.NE.0) THEN
         DO 203 IY=2,NRAD
         IYY=NYY-IY
         MAX=SQRT((FLOAT(IY-1)*YNV)**2+YLEN2)
         FACTOR=1.0-X1*RADI
         H(1,IYY)=H(1,IYY)*FACTOR
         H(IY,IYY)=H(IY,IYY)*FACTOR
      ENDIF
C
      C-23
M = MAX+1
DO 202 I = LL, NLX
II = II-1
H (NXX-II, IYY) = (0.0, 0.0, 0.0)
H (II, IYY) = (0.0, 0.0, 0.0)
H (NXX-II, IY) = (0.0, 0.0, 0.0)
H (II, IY) = (0.0, 0.0, 0.0)
ENDIF
C
WRITE (25, 661) XLAG, LAG
IF (NRAD. EQ. NY2) RETURN
LL = NRAD+1
DO 208 I = LL, NY2
IYY = IYY-1
H (I, IYY) = (0.0, 0.0, 0.0)
DO 209 J = 2, NX2
H (J, IYY) = (0.0, 0.0, 0.0)
H (NXX-J, IYY) = (0.0, 0.0, 0.0)
H (J, I) = (0.0, 0.0, 0.0)
H (NXX-J, I) = (0.0, 0.0, 0.0)
209 CONTINUE
C
C ..... APPLY HAMMING-TUKEY WINDOW TO FILTER
C
ELSEIF (NWIND. EQ. 3) THEN
C
IF (NRAD. NE. 0) THEN
PIRAD = PI * RAD
MAX = RADIUS * XKX + 1.0001
C
IF (MAX. GE. 2) THEN
DO 353 LL = 2, MAX
XI = FLOAT (LL-1) * XKX
FACTOR = 0.5 * (1.0 + COS (PIRAD * XI))
H (LL, IYY) = H (LL, IYY) * FACTOR
H (LL, IY) = H (LL, IY) * FACTOR
MX = MAX+1
DO 303 J = LL, MX
H (J, IYY) = H (J, IYY) * FACTOR
303 CONTINUE
C
C WRITE (25, 662) XLAG, LAG
C
IF (NRAD. EQ. NY2) RETURN
LL = NRAD+1
DO 308 I = LL, NY2
IYY = IYY-1
H (I, IYY) = (0.0, 0.0, 0.0)
H (I, IY) = (0.0, 0.0, 0.0)
DO 309 J = 2, NX2
H (J, IYY) = (0.0, 0.0, 0.0)
H (J, IY) = (0.0, 0.0, 0.0)
309 CONTINUE
C
C-25

CONTINUE
C
C ... APPLY PARZEN WINDOW TO FILTER
C
ELSEIF (NMIND.EQ.4) THEN
C
IF (NRAD.NE.2) THEN
MAX=RADI0(XNR+1.0001)
NRAD=FLOAT(NY2)*RADIUS*0.5+1.0001
MAX2=SQR(RAD2/4.0)*XNR+1.0001
FACTOR=1.0-6.0*(XI*RADI)**2-(XI*RADI)**3
H(1,1)=H(1,1)*FACTOR
C
IF (MAX2.GE.2) THEN
DO 453 LL=2,MAX2
XI=FLOAT(LL-1)*XNX
FACTOR=2.0*1.0-(XI*RADI)**3
H(LL,1)=H(LL,1)*FACTOR
MK=MAX-LX
453 H(MK,1)=H(MK,1)*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,1)=H(LL,1)*FACTOR
MK=MAX-LX
457 H(MK,1)=H(MK,1)*FACTOR
ENDIF
C
IF (NRAD.NE.1 .AND. NRAD.NE.0) THEN
DO 402 IY=2,NRAD
IY=NYY+IY
XI=FLOAT(IY-1)*NYY
YLEN2=XI*XJ
MAX=SQR(RAD2-YLEN2)*XNR+1.0001
NY2=FLOAT(NY2)*RADIUS*0.5+1.0001
C
IF (IY.GT.N2RAD) THEN
KOUNT=2
FACTOR=2.0*1.0-(XI*RADI)**3
H(I,IY)=H(I,IY)*FACTOR
H(I,IY)=H(I,IY)*FACTOR
ELSE
MAX=SQR(RAD2/4.0-YLEN2)*XNR+1.0001
FACTOR=1.0-6.0*(XI*RADI)**2-(XI*RADI)**3
H(I,IY)=H(I,IY)*FACTOR
H(I,IY)=H(I,IY)*FACTOR
ENDIF
C
IF (MAX2.GE.2) THEN
DO 403 LL=2,MAX2
XI=SQR((FLOAT(LL-1)*XNX)**2+YLEN2)
FACTOR=1.0-6.0*(XI*RADI)**2-(XI*RADI)**3
H(LL,IY)=H(LL,IY)*FACTOR
H(LL,IY)=H(LL,IY)*FACTOR
MK=MAX-LX
403 H(MK,IY)=H(MK,IY)*FACTOR
ENDIF
C
KOUNT-MAX+1
ENDIF
C
DO 407 LL=KOUNT,MAX
XI=SQR((FLOAT(LL-1)*XNX)**2+YLEN2)
FACTOR=2.0*1.0-(XI*RADI)**3
H(LL,IY)=H(LL,IY)*FACTOR
H(LL,IY)=H(LL,IY)*FACTOR
MK=MAX-LX
407 H(MK,IY)=H(MK,IY)*FACTOR
L=L+MAX+1
DO 405 II=LL,MAX2
H(NOX-II,IY)=H(NOX-II,IY)+H(I,IY)
H(NOX-II,IY)=H(NOX-II,IY)+H(I,IY)
405 H(I,IY)=H(I,IY)+H(I,IY)
402 CONTINUE
ENDIF
C
WRITE(25,663) XLAG,LAG
IF (NRAD.EQ.NY2) RETURN
DO 408 I=LL,NY2
  H(I, IY)= (0.0,0.0)
  H(IY, I)= (0.0,0.0)
DO 409 J-2,NX2
  H(J, IY)= (0.0,0.0)
  H(NX-J, IY) - (0.0,0.0)
  H(J, I) - (0.0, 0.0)
CONTINUE
C......DO NOT APPLY A WINDOW TO FILTER
CELSEIF (NWIND. EQ. 0) THEN
  WRITE (25,664)
ENDIF
RETURN
FORMAT(3X,'INPUTTED XLAG OF ',F5.2,' EXCEEDS PERMISSIBLE ',
       ' RANGE OF 0.0. AND. ,LE. 100.0, NO WINDOWING PERFORMED')
FORMAT ('RECTANGULAR WINDOW USED XLAG- ',F7.3,4X,'LAG- ',I5)
FORMAT ('BARTLETT WINDOW USED XLAG- ',F7.3,4X,'LAG- ',I5)
FORMAT ('HAMMING-TUKEY WINDOW USED XLAG- ',F7.3,4X,'LAG- ',I5)
FORMAT ('NO WINDOWING HAS BEEN APPLIED ; XLAG- ',F7.3)
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 CC0EF, CCIN, CCOUT
DATA ZERO/(0.000000, 0.000000 )/
common /rowcol/ xrow, xcol, yrow, ycol
common /comps/ x, y
common /ccflt/ mincc, maxcc, minccin, maxccin, cnwlnd, cxlag
common /fftifft/ nxout, nyout, prcnt, imean, fold, itypefold
common h
-- subroutine description
-- 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.
-- revised 4 aug 90: i've added the windowing functions
-- available from the bandpassing subroutines to this cc-
-- filter. try them if you like.
-- updates 1 feb 91: change calculation of correlation
-- coefficient from a summation based formula to the cosine of
-- the phase angle difference.
-- nx=nnxout
-- ny=nyout
if (xcol .ne. ycol .or. xrow .ne. yrow ) then
  write (*,*) 'NO MATCH BETWEEN ROW OR COLUMN'
  write (*,*) 'CORRELATION COEF MAY NOT BE CORRECT'
  write (*,*) 'PASSNUMBERS=', xpassno, ypassno
  write (*,*) 'FILE 1: ROW COL = ', xrow, xcol
  write (*,*) 'FILE 2: ROW COL = ', yrow, ycol
endif
pi=3.141592654
two=6.283185307
POWER1=ZERO
POWER2=ZERO
POWER3=ZERO
POWER4=ZERO
XPOWER=ZERO
TPPOWER=ZERO
DO 110 J=1, NY
  DO 110 I=1, NX
-- zeroct 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
c the principle.
c SUM THE POWERS & CROSS PRODUCTS OF THE INPUT MAPS.
c POWER1=POWER1+(X(I,J)*CONJG(X(I,J)))
POWER2=POWER2+(Y(I,J)*CONJG(Y(I,J)))
XPOWER=XPOWER+(X(I,J)*CONJG(Y(I,J)))

C xrad is the phase angle of the x array wavenumber and
yc rad is the phase angle of the y array wavenumber. the
C cosine of the minimum phase difference is the correlation
C of the two wavenumbers, to find the minimum phase difference
C it is necessary to adjust xrad or yrad with integer values
C of 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))<0.0) xrad=xrad+pi
if (aimag(x(i,j))<0.0) xrad=xrad+twopi
yrad=atan(aimag(y(i,j))/real(y(i,j))
if (real(y(i,j))<0.0) yrad=yrad+pi
if (aimag(y(i,j))<0.0) yrad=yrad+twopi
delrad=abs(xrad-yrad)
ccoe cos(delrad)

C IF (CCOE .GT. maxcc .or. CCOE .LT. mincc) THEN
X(I,J)=ZERO
Y(I,J)=ZERO
zerocnt(i,j)=0
ENDIF

C CALCULATE THE POWERS & CROSS PRODUCTS FOR THE OUTPUT MAPS.
C POWER3=POWER3+(X(I,J)*CONJG(X(I,J))
POWER4=POWER4+(Y(I,J)*CONJG(Y(I,J)))
XPOWER=XPOWER+(X(I,J)*CONJG(Y(I,J)))

C CALCULATE THE C.C. FOR THE INPUT MAPS.
C if (power1 .eq. zero .or. power2 .eq. zero) then
write (*,*5)'power1=',power1,xpassno
write (*,*)'power2=',power2,ypassno
ccin=9999.9
else
ccin=REAL(XPOWER/SQRT(POWER1*POWER2))
endif

C CALCULATE THE C.C. FOR THE OUTPUT MAPS.
C if (power3 .eq. zero .or. power4 .eq. zero) then
write (*,*)'power3=',power3,xpassno
write (*,*)'power4=',power4,ypassno
ccout=9999.9
else
ccout=REAL(TPOWER/SQRT(POWER3*POWER4))
endif

C CALCULATE THE PERCENTAGE OF THE POWER RETAINED IN THE FILTERED MAPS.
C if (power1 .eq. zero .or. power2 .eq. zero) then
pctpr1=9999.9
pctpr2=9999.9
else
pctpr1=(POWER3/POWER1)*100.0
pctpr2=(POWER4/POWER2)*100.0
endif

C WRITE THE C.C. FOR THE INPUT & OUTPUT MAPS TO FILE 6.
C WRITE (6,444) CCIN
C WRITE (6,555) CCOUT

C WRITE THE POWER PERCENTAGES TO FILE 6.
C WRITE (6,666) PCTPR1,PCTPR2
C 444 FORMAT ('THE CORRELATION COEFFICIENT BETWEEN THE INPUT ', 'MAPS IS ',F6.3)
C 555 FORMAT ('THE CORRELATION COEFFICIENT BETWEEN THE OUTPUT ', 'MAPS IS ',F6.3)
C 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,'%')
c write (25,8885)xpassno,ypassno,ccin,ccout,pctpr1,pctpr2
888 format (216,4f10.3)
if (ccin .lt. minccin) write (*,*)xpassno,ypassno,ccin,' <min'

C-27
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
      write (*,*) '1 - zerocnt, 0 !- zerocnt'
      read (*,*) 1
      if (i .eq. 1) then
        write (50,*) nx
        write (50,*) ny
        do i=1,ny
          write (50,9970) (zerocont(j,i),j=1,nx)
        enddo
      endif
      if (cnwind.ge.1 .and. cnwind.le.4) then
        power5=zero
        power6=zero
        totpwr=zero
        do 300 j=1,ny
          do 300 i=1,nx
            h(i,j) = (0.0,0.0)
          enddo
          if (zerocont(j,i) .eq. 0) h(i,j) = (0.0,0.0)
        enddo
        continue
      endif
      call store (nx,ny)
      call fft2d (nx,ny,1)
      call window (nx,ny,0,0.0,0.0)
      call fft2d (nx,ny,-1)
      do 330 iy=1,ny
        do 330 ix=1,nx
          x(ix,iy) = x(ix,iy)*h(ix,iy)
          y(ix,iy) = y(ix,iy)*h(ix,iy)
          power5=power5*(x(ix,iy)*conjugate(x(ix,iy)))
          power6=power6*(y(ix,iy)*conjugate(y(ix,iy)))
          totpwr=totpwr+(x(ix,iy)*conjugate(y(ix,iy)))
        enddo
        continue
      enddo
      if (power5.eq.0 .or. power6.eq.0) then
        write (*,*) 'power5 -',power5,xpassno
        write (*,*), 'power6 -',power6,ypassno
        ccwlnout=9999.9
        go to 340
      endif
      if (power1 .eq. 0 .or. power2 .eq. 0) then
        pctpr3=9999.9
        pctpr4=9999.9
        go to 340
      endif
      ccwlnout=real(totpwr/sqrt(power5*power6))
      pctpr3=(power5/power1)*100.0
      pctpr4=(power6/power2)*100.0
      go to 340
    enddo
    continue
    write (25,888) xpassno,ypassno,ccin,ccwlnout,pctpr3,pctpr4
    format (2i6,4fi0.3)
    return
  end
  subroutine inverseft(num,mean,passno)
    integer n_, xrow, xcol, yrow, ycol, row, col, passno
    real xdata(361,361), ydata(361,361), mean
    complex xdata(512,512), ydata(512,512), mean
    common /rowcol/ xrow, xcol, yrow, ycol
    common /real/ xdata, ydata
    common /comps/ xdata, ydata
    common /fftifft/ nxout, nyout, prcnt, lmean, fold, ltypefold
    common H(512,512)
    DIMENSION X(2,512,512), holdme(361,361)
    COMPLEX W
    EQUIVALENCE (X(1,1,1),H(1,1))
  c
    if (num .eq. 1) then
      ny=nyout
      nx=nxout
      row=xrow
      col=xcol
      do 50 j=1,ny
        do (i=1,nx
          h(i,j) = xdata(i,j)
        enddo
        continue
      endif
      num = 2
      if (num .eq. 1) then
        nx = nxout
        row = xrow
        col = xcol
        do 50 i = 1,nx
          do (j = 1,ny
            h(i,j) = ydata(i,j)
          enddo
          continue
        endif
      endif
    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
N(N,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 AUS 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.

PROGRAM FRQ2SPA

CALL FFT2D (NX, NY, +I)

nxhalf= (nx-icol)/2
nyhalf= (ny-irow)/2

DO I=nxhalf-I, nxhalf+icol
DO J=nyhalf+1, nyhalf+1row
HOLDME(I-nxhalf,J-nyhalf)=X(I,J)
enddo
enddo

IROW=IROW
icol=icol

nxhalf=(nx-icol)/2
nyhalf=(ny-irow)/2

DO I=nxhalf-I, nxhalf+icol
DO J=nyhalf+1, nyhalf+1row
HOLDME(I-nxhalf,J-nyhalf)=X(I,J)
enddo
enddo

DO 210 J-IROW
DO 210 I-icol
X(I,J)=HOLDME(I,J)
enddo
enddo

total=0.0

DO 210 J-IROW
DO 210 I-icol
X(I,J)=X(I,J)+mean
enddo
enddo

XMIN=1.0E20
XMAX=-1.0E20

DO 220 I=1, IROW
IF(NUM.EQ. 1) THEN
DO 215 J=1, IROW
X(I,J)=X(I,J)+mean
ENDDO
ELSEIF(NUM.EQ. 2) THEN
DO 215 J=1, IROW
X(I,J)=X(I,J)+mean
ENDDO
ENDIF
DO 220 I=1, IROW
XMIN=MINI(XMIN, X(I,J))
ENDDO

C-29
IF(XMAX.EQ.X(I,I,J)) THEN
  IMAX=I
  JMAX=J
ENDIF
IF(XMIN.EQ.X(I,I,J)) THEN
  IMIN=I
  JMIN=J
ENDIF
220 CONTINUE
C WRITE (25,1020) XMAX, XMIN, JMIN, JMAX, IMIN, IMAX, passno
WRITE (25, 9980) passno, mean, JMAX, IMAX, JMIN, IMIN
9980 format (i5,2x,E15.7,2x,i3,2x,E15.7,2x,i4,2x,E15.7,2x,i4,2x,E15.7,2x,i4)
C
C subroutine STRKPAS (num)
integer num,xrow,xcol,yrow,ycol,spass,swind,
  imean,nxout,nyout,row,col
real prcnt,slag
complex xcdata(512,512),ycdata(512,512)
common /fftlfft/ nxout,nyout,prcnt,imean,fold,itypefold
common /comps/ xcdata,ycdata
common /striking/ angl,ang2,spass,swind,slag
COMMON H(512,512)
COMPLEX H
C if (num .eq. 1) then
  row=nyout
  col=nxout
  nx=nxout
  ny=nyout
else (num .eq. 2) then
  row=nyout
  col=nxout
  nx=nxout
  ny=nyout
endif
C **********************************************************************
C * "STRIKE" PERFORMS A STRIKE SENSITIVE FILTERING (FAN FILTER) *
C * ON UNIFORMLY 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 * 0 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 * updates and revisions: *
C * 23 dec 91: added this strike pass routine to the fourier *
C * program. required removal of write statements *
C **********************************************************************
C CALL STRIKE (ANG1,ANG2,spass,nx,ny)
C---------------------- CREATE SMOOTHED FILTER
C IF(slag.gt.0.0 .and. slag.lt.100.0) then
  IF(swind.gt.0.0 .and. swind.le.4) then
    CALL FFT2D (nx,ny)
    CALL WINDOW (nx,ny,slag,swind)
    CALL FFT2D (nx,ny)
  endif
endif
C---------------------- SET UP TO WRITE 30 if desired
C DO 356 i=1,row
  DO 356 j=1,col
    xcdata(j,i) = xcdata(j,i)*h(j,i)
  356 CONTINUE
C---------------------- ACCESS TRANSFORM OF DATA & MULTIPLY *FILTER (CONVOLVING)
C
500         continue
      elseif (num .eq. 2) then
         do 580 1=1,row
            do 580 j=1,col
               ycdata(j,1) = ycdata(j,1)*h(j,1)
            enddo
         enddo
      continue
      endif
      return
  end SUBROUTINE

SUBROUTINE STRIKE (AANG1,AANG2,NNPASS,NNX,NNY)

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

"STRIKE" CREATES A STRIKE SENSITIVE FILTER (FAN FILTER)
FOR 2 QUADRANTS OF THE (NX, NY) MATRIX
ARRAY "H" MUST BE DIMENSIONED THE SAME AS IN THE MAIN PROGRAM
ANGLES ARE MEASURED IN DEGREES CLOCKWISE FROM NORTH
"ANG1" = SMALLEST ANGLE, GE 0.0 .AND. LT ANG2
"ANG2" = LARGEST ANGLE, LT ANG1 .AND. LE 180.0
"NPASS" = STATES IF DATA IS PASSED OR REJECTED BETWEEN THE 2 ANGLES
--1 REJECT AZIMUTHS BETWEEN ANGLES "ANG1" & "ANG2"
1 PASS AZIMUTHS BETWEEN ANGLES "ANG1" & "ANG2"
"NX" = NUMBER OF ROWS (POWER OF 2 GE "NXI", 16, 32, ETC) MAX-128
"NY" = NUMBER OF ROWS (POWER OF 2 GE "NY1", 16, 32, ETC) MAX-128

NX=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
NXN=NX+2
NYN=NY+2
XY=FLOAT (NX2)/FLOAT (NY2)

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

DO 15 IY=1,NY
   DO 15 IX=1,NX
      H(IX, IY)=ZERO
15   CONTINUE

COMPUTE PARAMETERS FOR SOUTHWEST QUADRANT OF MATRIX

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
   ADA=1.5
   GOTO 113
114   CONTINUE
   A1=ACOT(TAN((ANG1-90.0)*DG2RAD)*XY)
   TA1=TAN(A1)
   IYMAX=FLOAT (NX2)/TA1+1.0
   IYMAX=MIN(IYMAX, NY2)
   ADA=1.0
113   CONTINUE

C-31
IF(ANG2.LT.180.0) GOTO 115
A2=DG90
TA2=0.0
ADA2=FLOAT(NX2)+0.5
GOTO 116
115 CONTINUE
A2=ATAN(TAN((ANG2-90.0)*DG2RAD)*XY)
TA2=TAN(A2)
ADA2=1.0
116 CONTINUE
C
200 IF(ANG1.LE.90.0) GOTO 25
IYMAXX=0
GOTO 300
C
25 IF(ANG1.GT.0.0) GOTO 45
A1=0.0
TTAI=1.0/DG20
ADDAI=FLOAT(NX2)-2.5
GOTO 60
45 IF(ANG1.NE.90.0) GOTO 55
A1=0.0
A2=0.0
TTA2=0.0
ADDA2=0.0
ADDA2=0.0
IYMAXX=NY2
GOTO 122
55 A1=ATAN(TAN(ANG1*DG2RAD)/XY)
TTAI=1.0/TAN(A1)
ADDA1=1.0
C
60 IF(ANG2.LT.90.0) GOTO 121
IYMAXX=NY2
TTA2=0.0
ADDA2=0.0
GOTO 122
121 CONTINUE
A2=ATAN(TAN(ANG2*DG2RAD)/XY)
TTA2=1.0/TAN(A2)
ADDA2=1.0
IYMAXX=ABS(FLOAT(NX2)/TTA2+1.5)
IYMAXX=AMIN(IYMAXX, NY2)
122 CONTINUE
C
C ********************************************************************************
C  CALCULATE THE FILTER COEFFICIENTS
C ********************************************************************************
C
300 NYMAX=AMAX0(IYMAX, IYMAXX)
DO 50 IY=1,NYMAX
Y=FLOAT(IY-1)
C
C ********************************************************************************
C  DEFINE SOUTHWEST QUADRANT
C ********************************************************************************
C
30 IF(IYMAXX.LT.IY) GOTO 35
MI=IY/TTAI+ADAI
MAX=(NX-IY*TTA2)+ADDA2
MAX=AMIN(MAX, NX2)
C
IF(MIN.GT.MAX) GOTO 30
DO 75 IX=MIM, MAX
H(IX,IY)=ONE
C
C ********************************************************************************
C  DEFINE SOUTHEAST QUADRANT
C ********************************************************************************
C
35 IF(IY.EQ.1.OR. IY.EQ.NY2) GOTO 50
C-32
subsection UPCON (num)
integer num, npass, imean, nwind, nxout, nyout, udnwind, 
row, col
complex xcdata(512,512), ycdata(512,512)
common /fftifft/ nxout, nyout, prcnt, imean, fold, itypefold 
common /comps/ xcdata, ycdata 
common /udcont/ delta, zcon, udxlag, udnwind
common H(512,512)
COMPLEX H

if (num .eq. 1) then
row=nyout
col=nxout
nx=nxout
ny=nyout
else if (num .eq. 2) then
row=nyout
col=nxout
nx=nxout
ny=nyout
endif

************************************************************************
** "UPCON" PERFORMS UPWARD OR DOWNWARD CONTINUATION ON UNIFORMLY
** GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.
************************************************************************

CREATE CONTINUATION FILTER
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 .le. 4 .and. udnwind .gt. 0) then
CALL FFT2D (NX, NY)
CALL WINDOW (NX, NY, udxlag, udnwind)
CALL FFT2D (-NX, -NY)
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
else if (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 OF 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
NX2-1.0/FLOAT (NX2)
NY2=NY/2+1
NY2=1.0/FLOAT (NY2)

DO 101 IY=1,NY2
   CON1=(FLOAT (IY-1)*NY2)**2
   S=SQRT (CON1)
   X1=EXP (S*PI2Z)
   H(1,IY)=CMPLX (X1,0.0)
   DO 101 IX=2,NX2
      S=SQRT ((FLOAT (IX-1)*NX2)**2+CON1)
      X1=EXP (S*PI2Z)
      H(IX,IY)=CMPLX (X1,0.0)
   H(NXX-IX, IY)=H(IX, IY)
101 CONTINUE
H(1,1)=(1.0,0.0)
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
  COMPLEX H(512,512)
  CALL DERIV (NX, NY, NTH, NWAY, DELTA)
  IF (hUm .eq. I) 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

SUBROUTINE DERIV (NNX, NNY, NNTH, NWAY, DELTA)
  COMMON C (512, 512)
  COMPLEX C, CON, CON2, CON3

  "DERIVA" CALCULATES THE "NTH" DERIVATIVE IN THE WAVE# DOMAIN OF UNIFORMLY GRIDDED ONE OR TWO DIMENSIONAL DATA SETS.
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 BE 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)
RNXDEL=1.0/(FLOAT(NX)*DELTA)
RNYDEL=1.0/(FLOAT(NY)*DELTA)

C
C******************************************************************
C TAKE VERTICAL DERIVATIVE IN WAVE# DOMAIN
C******************************************************************
C
IF (NWAY.EQ.0) THEN
  DO 105 IX=2,NX2
  FX2=(FLOAT(IX-1)*RNXDEL)
  C(NXX-IX,1)=(CON*FX2)**NTH
  105 C(I,1)=(0.0,0.0)

  DO 205 IY=2,NY2
  IYY=NYY-IY
  FY2=(FLOAT(IY-1)*RNYDEL)
  C(I,IYY)=(CON*FY2)**NTH
  205 CON2=CON2

  DO 305 IX=2,NX2
  IXX=NXX-IX
  CON2=FLOAT(IX-1)*RNXDEL*CON**NTH
  CON3=CONJG(CON2)
  DO 310 IY=2,NY2
  C(IXX,NYY-IY)=CON3
  310 C(IY,IY)=CON3

  110 C(IX, IY)=CON2

C******************************************************************
C TAKE HORIZONTAL(Y) DERIVATIVE IN WAVE# DOMAIN
C******************************************************************
C
ELSEIF (NWAY.EQ.2) THEN
  DO 205 IX=1,NK
  C(IX,1)=(0.0,0.0)

  DO 210 IY=2,NY2
  IYY=NYY-IY
  CON2=FLOAT(IY-1)*RNYDEL*CON**NTH
  CON3=CONJG(CON2)
  DO 310 IX=1,NK
  C(IX, IYY)=CON3
  310 CON2=CON2

  210 C(IX, IY)=CON2

C******************************************************************
C TAKE HORIZONTAL(X) DERIVATIVE IN WAVE# DOMAIN
C******************************************************************
C
ELSEIF (NWAY.EQ.1) THEN
  DO 305 IY=1,NY
  C(I, IYY-1)=(0.0,0.0)

  DO 310 IX=2,NX2
  IXX=NXX-IX
  CON2=FLOAT(IX-1)*RNXDEL*CON**NTH
  CON3=CONJG(CON2)
  C(IXX,1)=CON3
  C(I, I)=CON2
  DO 310 IY=2,NY2
  C(IXX,NYY-IY)=CON3
  310 C(IY,IY)=CON3

  310 C(IX, IY)=CON2

RETURN

C
else
  write (*,*) 'nway is not equal to 0,1 or 2'
  stop
endif
end

subroutine Mag2pol (num)
  integer num, npass, imean, nwind, nxout, nyout,
           row, col
  complex xdata(512,512), ydata(512,512)
  common /fftfft/ 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 "F_G2POL" APPROXIMATLY CALCULATES THE CORRESPONDING MAGNETIC
C ANOMALY MAP DUE TO AN EARTH'S FIELD VECTOR OF 0.0 DECLINATION,
C AND 90.0 INCLINATION FROM AN INPUTTED MAGNETIC ANOMALY MAP WITH
C A KNOWN FIELD VECTOR.
C**************************************************************
C DEC-DEC+AZM
  CALL MAGPOL(XINC, DEC, NX, NY)
C ACCESS TRANSFORM OF DATA & MULTIPLE *FILTER (CONVOLVING)
  if (num .eq. 1) then
    do 500 l = 1, row
      do 500 j = 1, col
         xdata(j, l) = xdata(j, l) * h(j, l)
      500 continue
  elseif (num .eq. 2) then
    do 580 l = 1, row
      do 580 j = 1, col
         ydata(j, l) = ydata(j, l) * h(j, l)
      580 continue
  endif
  return
end

SUBROUTINE Mag2pol (xinc, dec, nx, ny)
  common x(512,512)
  complex x, xinc, xdec
 ************************************************************************
  "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.
  "XINC" THE AVERAGE MAGNETIC INCLINATION OF THE AREA IN DEGREES.
  "XDEC" THE AVERAGE MAGNETIC DECLINATION OF THE AREA IN DEGREES.
************************************************************************
  RE-3.14159265/180.0
  NX2 = NX/2+1
  NXX = NX+2
  NYY = NY+2
  NY = NY+2
  RNX = 1.0/FLOAT(NX)
  RNY = 1.0/FLOAT(NY)
  SINI = SIN(XINC*RR)

C-36
COSI = COS(AINC * RR)
SIND = SIN(DDEC * RR)
COSD = COS(DDEC * RR)

C
CON3 = COSI * COSD
CON2 = COSI * SIND

C
CONA = 1.0 / CMPLX(SINI, CON2) ** 2
CONB = CONJG(CONA)
X(1,1) = (0.0, 0.0)
DO 30 IX = 2, NX1
   X(IX, 1) = CONA
30
X(NX2 - IX, 1) = CONB
   X(NX2, 1) = CONA

C
CONA = 1.0 / CMPLX(SINI, CON3) ** 2
CONB = CONJG(CONA)
DO 50 IY = 2, NY2
   IYY = IYY - IY
   FY = FLOAT(IY - 1) * RNY
   FY2 = FY * FY
   CFY = CON3 * FY
   X(1, IYY) = CONB
   X(1, IY) = CONA
   DO 40 IX = 2, NX1
      IXX = IX = IX - IX
      FX = FLOAT(IX - 1) * RNX
      CON4 = SQRT(FX ** 2 + FY2)
      X(IX, IYY) = 1.0 / CMPLX(SINI, (CFY * FX + CON2) / CON4) ** 2
      X(IX, IY) = CONJG(X(IX, IYY))
      X(IX, IY) = 1.0 / CMPLX(SINI, (FX * CON2 + CFY) / CON4) ** 2
40
X(IX, IYY) = CONJG(X(IX, IY))
   CON4 = SQRT(0.25 + FY2)
   X(NX2, IYY) = 1.0 / CMPLX(SINI, (0.5 * CON2 - CFY) / CON4) ** 2
50
RETURN
end
program avgdifres
character*80 filename

integer coln, rown, colk, rowk, var, row, col, set

dimension dndata(550,550), dkdata(550,550), avgdata(550,550),
>            difdata(550,550), ddata(550,550)

program description
e avgdifres is a bashing of matrix manipulation. the program
c will find the average of two input data sets, the difference
between them, and will resample (hence avg-dif-res aint science
great?) every other point, every third, fourth etc., point.
c If you want the average and difference of the input matrices
make sure you choose to resample every data point, ie choose
c 1 for 1x1.
c
write (*,*) '1 FOR ONE DATA SET'
write (*,*) '2 FOR TWO DATA SETS'
read (*,*) set

write (*,*) 'INPUT FILE OF DAWN GRIDDED ANOMALIES OR ONE DATA SET'
read (*,9990) filename
9990 format (a80)
open (10, file-filename, status='old', form='formatted')
if (set .eq. 1) go to 10
write (*,*) 'INPUT FILE OF DUSK GRIDDED ANOMALIES'
read (*,9990) filename
open (12, file-filename, status='old', form='formatted')
write (*,*) 'OUTPUT AVERAGED AND/OR RESAMPLED GRIDDED FILE'
read (*,9990) filename
open (20, file-filename, form='formatted')
if (set .eq. 1) go to 20
write (*,*) 'OUTPUT DIFFERENCE GRID OF (DUSK) - (DAWN)'
read (*,9990) filename
open (22, file-filename, form='formatted')
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,*) scolat
read (10,*) xlong
read (10,*) xgridspc
40 do 50 l=1,rown
    read (10,*) (dndata(l, 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

c .................. find the average and difference matrices also
c calculate the total and average RMS difference.
c
read (12,*) colk
read (12,*) rowk
read (12,*) ycolat
read (12,*) ylong
read (12,*) ygridspc
80 do 100 l=1,rown
    read (12,*) (dkdata(l, J), J=1,coln)
100 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
c
40 coninue

totrms=0.0
do 150 l=1,rown
    do 150 j=1,coln
        avgdata(l, j)=((dndata(l, j)+dkdata(l, j))/2.0)
        ddata(l, j)=dkdata(l, j)-dndata(l, j)
        totrms=totrms+(((dndata(l, j)-dkdata(l, j))**2)/2.0)
150 continue
avgrms=totrms/(real(rown)*real(coln))

170 if (set .eq. 1) then
    do 180 l=1,rown
        do 180 j=1,coln
            avgdata(l, j)=dndata(l, j)
180 continue
endif

c this is the section that resamples
row=1
1=1
200 1=1
1=1
col=1
250 ddata(l, j)=avgdata(row, col)
col=col+var
if (col.gt.col) go to 300
jj=jj+1

300 row=row+var
if (row.gt.row) go to 400
ii=ii+1
go to 200

400 continue
c
write (*,*) 'new rows=',ii,' new cols=',jj
write (20,*) jj
write (20,*) if
write (20,*) xcolat
write (20,*) xlong
write (20,*) ylong
write (20,9992) (ddata(k,l),l=1, JJ)
write (20,9992) (ddata(k,l),l=1, JJ)
c
if (set.eq.1) go to 999
write (22,* ) col
write (22,* ) row
write (22,* ) ycolat
write (22,* ) ylong
write (22,* ) ygridpsc
do 500 m=1,rown
write (22,9992) (dlfdata(m,n),n=1,coln)
continue
write (*,*) 'total rms difference=', totrms
write (*,*) 'average rms difference=', avgrms

c
999 continue
close (10)
close (12)
close (20)
close (22)
stop
end
program sqmap
character*80 filename
dimension xcore(100000),xmag(100000),
> core(361,361),xmag(361,361),xmultmag(400)

c---------------------------------------- program description
cc this program will find a least squares value of x that
cc can be multiplied by the difference matrix so that the
cc difference closer fits the dawn or dusk matrix. the
cc value of x is multiplied by the difference and subtracted
cc from the dawn or dusk grid to make an output grid.
ccc NOTE: i realize i'm making the 2-D arrays
cc into 1-D arrays
cc
cc write (*,*) 'input difference matrix'
read (*,9990) filename

9990 format (a80)
open (10, file=filename,form='formatted',status='old')
write (*,*) 'input dawn or dusk matrix of total field values'
read (*,9990) filename
open (10, file=filename,form='formatted',status='old')
write (*,*) 'output (total field) - (x)(difference)'
read (*, 9990) filename
open (20, file=filename,form='formatted')

ccmean=0.0
fmean=0.0
read (10,*) icol
read (10,*) irow
read (10,*) south
read (10,*) west
read (10,*) gridspc
do i=1,irow
  read (10,*) (core(i,j),j=1,icol)
do j=1,icol
  cmean=cmean+core(i,j)
enddo
endo
c
read (10,*) imcol
read (10,*) imrow
read (10,*) south
read (10,*) west
read (10,*) gridspc
do i=1,imrow
  read (10,*) (xmag(i,j),j=1,imcol)
do j=1,imcol
  fmean=fmean+xmag(i,j)
endo
endif
if (irow.ne.imrow.or.icol.ne.imcol) then
write (*,*) 'rows cor mag ',irow,irow
write (*,*) 'cols cor mag ',icol,icol
stop 10
endif
ccmean=cmean/real (irow*icol)
fmean=fmean/real (irow*icol)
c
cc---------------------------------------- remove the mean values
cc do 500 j=1,irow
cc do 500 j=1,icol
cc xmag(i,j)=xmag(i,j)-fmean
cc core(i,j)=core(i,j)-cmean
cc
cc---------------------------------------- turn the 2-D arrays into 1-D (cheater)
c
cc do 500 j=1,irow
cc do 500 j=1,icol
cc
xmag(100000)=xmag(i,j)+j
xcore(100000)=core(i,j)+j
endo
cc
cc---------------------------------------- find a scalar value of c transpose c
cc do 600 j=1,111
cc ctc=(xcore(i)*xcore(i))+ctc
cc
cc---------------------------------------- find (c transpose c) inverse
cc ctcinv=1.0/ctc
cc do 700 j=1,111
cc ctf=(xcore(i)*xmag(i))+ctf
cc
cc---------------------------------------- find x
cc x=ctcinv*ctf
write (*,*) 'the value of x = ', x
cc
write (20,*) imcol
write (20,*) imrow
write (20,*) south

C-42
write (20,*) west
write (20,*) gridspc
do i=1,1row
  do j=1,1col
    xmultmag(j) = xmag(1, j) - (x)(core(i, j))
  enddo
  write (20,9991) (xmultmag(j), j=1,1col)
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, YORO, DXD, DYD, NXD, NYD, E, G, COST, SINT, YC, THETA, PHI, P(3982), TMAG(3982), DF(361,44), S(7930153)
REAL I, INC(361), D, DEC(361), FLD(361), II, DI, FI, MYEAR, MI, MD, MILO, MDLO, MFLO, MIRO, MDRO, FLD(44,361), INCD(44,361), DECD(44,361), DECO(44,361)
COMMON //gmf// FLD(44,361), INC(44,361), DECD(44,361), DECO(44,361)
REAL FLD(44,361), INC(44,361), DECD(44,361), DECO(44,361)
CHARACTER*80 FILENAME
INTEGER CHOICE
************************************************************************
C
PROGRAM NVERTSM CALCULATES A SET OF CGS-SUSCEPTIBILITIES FOR AN NXD-BY-NYD SPHERICAL ARRAY OF POINT DIPOLES SUCH THAT THE RESULTANT EQUIVALENT SOURCE FIELD IS A LEAST-SQUARES BEST FIT TO MAGNETIC DATA OBSERVED OVER AN NXO-BY-NYO SPHERICAL GRID. OUTPUT CONSISTS OF LISTINGS AND/OR PUNCHED DECKS OF EQUIVALENT SOURCE SUSCEPTIBILITIES AND/OR ANOMALY VALUES (SEE DATA CARD 1 BELOW).

DIMENSIONING REQUIREMENTS.....
DIMENSION DW(NXD*NYD), DP(NXD*NYD), P(NXD*NYD), TMAG(NXD*NYD), DF(NXD*NYO), INC(NXD) WHERE N1=NXD*NYD+1/2, FLD(NXD), INC(1), DI(1), FLD(1), I(1), D(1)

this program has been slightly modified from a viewpoint of lines of code. these changes are all lower case. however from the point of view of run time it should now take less than 1/4 of the time that it took in the past. this is because no reads from files i0 or ii are necessary as all arrays are stored in memory.

modifications made 15 may 90

further changes on 22 sep 90
these changes are mostly removal of unnecessary write statements and general cleanup of the program.

more changes on 2 jan 91
this update included the addition of a few lines of code that allows for input of a file of susceptibilities and output of a magnetic field map.

************************************************************************ DATA INPUT SEQUENCE ****************************

write ('*','0 IF YOU HAVE A FILE FOR THE VARIABLES'
write ('*','1 IF YOU WANT TO TYPE VARIABLES INTERACTIVELY'
read ('*','*') choice
if (choice .eq. 0) go to 10
write ('*','*') nfld, nio
if (nfld .eq. 1) then
write ('*','*') nxo, nyo, xoro, yoro, dxo, dyo, elvo
endif
write ('*','*') nxo, nyo, xoro, yoro, dxo, dyo, elvo
read ('*','*') nxo, nyo, xoro, yoro, dxo, dyo, elvo
write ('*','*') nxo, nyo, xoro, yoro, dxo, dyo, elvo
read ('*','*') nxo, nyo, xoro, yoro, dxo, dyo, elvo
'ELVD- ELEVATION OF SOURCE GRID IN KILOMETERS (-50.0)'
read (*,*) nxd,nyd,xord,yord,dxd,dyd,elvd

write (*, 9994)
9994 format ('YEAR- EPOCH IN YEARS AND DECIMAL FRACTION YEARS'/
'E.G., 1965.75 = 1 OCT 65 FOR WHICH THE'/
'GEOMAGNETIC REFERENCE FIELD IS TO'/
'BE COMPUTED AT SOURCE AND/OR OBSERVATION POINTS'/
'USER SUPPLIES CHARACTERISTICS OF SOURCE POLARIZATION'/
'FIELD (F1,I,D) AND GEOMAGNETIC FIELD (I,D) OVER'/
'OBSERVATION GRID'/
'F1= SCALAR MAGNETIC INTENSITY IN GAMS OF SOURCE POLARIZATION'/
'I= 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 (0.0) I (0.0) D (0.0)'
read (*,*) year,f1,i,d

write (*, 9995)
9995 format ('I- INCLINATION IN DEGREES OF THE GEOMAGNETIC FIELD'/
'OVER THE OBSERVATION POINTS'/
'NOTE----IF (I+D).EQ.0.0, THEN THE GEOMAGNETIC FIELD OVER THE'/
'OBSERVATION POINT IS COMPUTED BY SUBROUTINE FIELDG'/
'FOR THE EPOCH SPECIFIED BY THE YEAR-INPUT VARIABLE'/'
'I (0.0) D (0.0)'
read (*,*) i,d

write (*,*) 'ERROR FACTOR FOR VARIANCE (fak) (0.1OE-7)'
read (*,*) fak

if (nfld .eq. 0)
go to 15
write (*) 'the following refers to calculation of the'
write (*) 'equivalent source magnetic field'
write (*)
9998 format ('r_XO- NUMBER OF LONGITUDINAL COLS OF OBSERVATION GRID'/'mNYO- NUMBER OF LATITUDINAL ROWS OF OBSERVATION GRID'/'mXORO- WESTERN-MOST LONGITUDINAL COORDINATE OF OBSERVATION'/'GRID IN -180.0 to 180.0 DEGREES'/'mYORO- SOUTHERN-MOST LATITUDINAL COORDINATE OF OBSERVATION'/'GRID IN -90.0 to 90.0 DEGREES'/'mDXO- LONGITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/'mDYO- LATITUDINAL STATION SPACING OF OBSERVATION GRID IN DEGS'/'mELVO- ELEVATION OF OBSERVATION GRID IN KILOMETERS (350.0) '/
'mNXO \_TYO mXORO mYORO mDXO mDYO mELVO ')
read (*, t ) mnxo,mnyo,maoro,myoro,mdxo,mdyo,melvo

write (*, 9996)
9996 format ('mYEAR- EPOCH IN YEARS AND DECIMAL FRACTION YEARS'/
'E.G., 1965.75 = 1 OCT 65 FOR WHICH THE'/
'GEOMAGNETIC REFERENCE FIELD IS TO'/
'BE COMPUTED AT SOURCE AND/OR OBSERVATION POINTS'/
'USER SUPPLIES CHARACTERISTICS OF SOURCE POLARIZATION'/
'FIELD (F1,I,D) AND GEOMAGNETIC FIELD (I,D) OVER'/
'OBSERVATION GRID'/
'F1= SCALAR MAGNETIC INTENSITY IN GAMS OF SOURCE POLARIZATION'/
'I= INCLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
'D= DECLINATION IN DEGREES OF SOURCE POLARIZING FIELD'/
'NOTE----IF (mFI+mII+mDI).EQ.0.0, THEN THE SOURCE POLARIZING'/
'FIELD IS COMPUTED BY SUBROUTINE FIELDG FOR EPOCH'/
'SPECIFIED BY THE YEAR-INPUT VARIABLE'/'
'mYEAR (0.0) mFI (60000.0) mII (90.0) mDI (0.0)'
read (*,*) myear,mfi,mii,midi

write (*, 9997)
9997 format ('mI= INCLINATION IN DEGREES OF THE GEOMAGNETIC FIELD'/
'OVER THE OBSERVATION POINTS'/
'mD= DECLINATION IN DEGREES OF THE GEOMAGNETIC FIELD OVER THE'/
'OBSERVATION POINTS'/
'NOTE----IF (mI+mD).EQ.0.0, THEN THE GEOMAGNETIC FIELD OVER THE'/
'OBSERVATION POINT IS COMPUTED BY SUBROUTINE FIELDG'/
'FOR THE EPOCH SPECIFIED BY THE YEAR-INPUT VARIABLE'/'
'mI (90.0) mD (0.0)'
read (*,*) mi,md

write (*,*) '0 DO NOT SUBTRACT MEAN OF FINAL R-T-P MAP'
write (*,*) '1 SUBTRACT THE MEAN'
read (*,*) isub
go to 15

write (*,*) 'INPUT FILE OF NUMBERS FOR VARIABLES'
read (*,9990) filename
open (17, file-filename, status='old',form='formatted')
READ (17,*) Nfld, Nio
READ (17,*) Nxo, NyO, Xoro, Yoro, DXo, Dyo, elvo
READ (17,*) Nxo, NyD, Xord, Yord, DXd, Dyd, elvd
READ (17,*) Year, Fl, II, I, D
READ (17,*) fak
READ (17,*) mxo, myo, nxoro, myoro, mdxo, mdyo, melvo
READ (17,*) myear, mfl, mil, mdl, mi, md
READ (17,*) Isub

WRITE (*,*) 'INPUT FILE OF GRIDDED ANOMALY DATA OR SUSC DATA'
WRITE (*,*) 'DATA SHOULD BE WEST TO EAST AND SOUTH TO NORTH'
WRITE (*,*) 'ANOMALY DATA IN GGGAMS OR NANO-TESLAS WITH THE INPUT STARTING WITH THE SOUTHERNmost LATITUDE AT THE WESTERNmost LONGITUDINE.'

READ (*, 9990) filename
9990 format (80A)
OPEN (13, file=filename, status='old', form='formatted')
WRITE (*,*) 'INPUT FILE OF SPHERICAL HARMONIC COEFFICIENTS SUCH AS GSFC1283'
READ (*,9990) filename
OPEN (3, file='../../data/mgst1283', status='old')
IF (nfld .eq. I .or. nfld .eq. 2) THEN
WRITE (*,*) 'OUTPUT FILE OF EQUIVALENT SOURCE M-FIELD'
READ (*,9990) filename
OPEN (30, file=filename, form='formatted')
ENDIF
IF (nlo .eq. I) THEN
WRITE (*,*) 'OUTPUT FILE OF SUSCEPTIBILITIES'
READ (*,9990) filename
OPEN (33, file=filename, form='formatted')
ENDIF
WRITE (*,*) 'OUTPUT INFORMATION FILE OF A BUNCH OF STUFF!!'
READ (*,9990) filename
OPEN (6, file=filename, form='formatted')
C
C IF (YEAR.LT.I.E-9) GO TO 120
C CALL FIELDG (O.,O.,O.,O.,55, LQ, Q1, Q2, Q3, Q4)
C IF (F1+II+DI.GT.0.0) GO TO 110
C CALL GEOMAG (YEAR, ELVD, YORD, XORD, DYO, DXD, NYD, NXD, 10)
C 120 CONTINUE
C C COMPUTE MAXIMUM, MINIMUM AND AVERAGE AMPLITUDE VALUES FOR M-FIELD INPUT DATA
C DSUM=0.0
DMIN=DF(I, I)
DMAX=DMIN
DO 140 JY=1, NYO
    READ (13,*) (DF(IX, JY), IX=1, NVO)
C 130 CONTINUE
C 140 CONTINUE
C 140 DMIN=DF(MIN=(NIK))
C 140 DMAX=DF(MAX=(NIK))
WRITE (6, 640) DMAX, DMIN, DSUM
C
save values for the reduction-to-pole

ISTRNX=NXD
ISTRNY=NYD
STRXORD=XORD
STRYORD=YORD
STRDXD=DXD
STRDYD=DYD

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=180.0-YORD)*FACT
XORD=90.0-YORD)*FACT
YORD=90.0-YORD)*FACT

DXD=DXD*FACT
DYD=DYD*FACT

I=I*FACT
D=D*FACT

KEARTH=6371.0
RO=ELVD+KEARTH
RD=ELVD+KEARTH
E=RD**2+RO**2
G=2.0*RD*RO
NP=(NXD*NYD)

DO 150 JY=1,NP
  P(JY)=0.0
  DN(JY)=0.0
  CONTINUE

DO 160 JY=1,NIJ
  S(JY)=0.0
  CONTINUE

DO 170 IY=1,NYD
  THETA=YORD+ (FLOAT (JY) -1.0)*DYD
  COST=COS (THETA)
  SINT=SIN

  DO 175 IX=1,NXD
    FLD(IX)=FLD(IX,JY)
    INC(IX)=INC(IX,JY)
    DEC(IX)=DEC(IX,JY)
  CONTINUE

DO 170 IX=1,NXD
  PHI=XORD+ (FLOAT (IX) -1.0)*DXD
  AI=INC(IX)
  AD=DEC(IX)

  CALL MAGS1 (AI,AD)

  DY=DFI(IX,JY)-YC

DO 170 L=1,NP
  DM(L)=DM(L)+TMAG(L)*DY

170 CONTINUE

DO TO 240

DO 180 JY=1,NYD
  THETA=YORD+ (FLOAT (JY) -1.0)*DYD

C-48
**COST-COS(THETA)**  
**SINT-SIN(THETA)**  
**DO 190 IX=1,NXO**  
**PHI=XSIND(IX) (FLOAT (IX) -1.0) *DXO**  
**CALL MAGS2 (I,D, FI, II, DI)**  
**DO 190 JY=1,NYO**  
**THETA=YORDO (FLOAT (JY) -1.0) *DYO**  
**CALL MAGS1 (I,D)**  
**C**  
**CONTINUE**  
**GO TO 240**  
**C**  
**C COMPUTE A (TRANSPOSE) A MATRIX FOR CASE WHERE (I,D) ARE DERIVED FROM FIELDG AND USER SUPPLIES (FI, II, DI)**  
**DO 210 JY=1,NYO**  
**THETA=YORDO (FLOAT (JY) -1.0) *DYO**  
**CALL MAGS2 (AI,AD, FI, II, DI)**  
**C**  
**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
YLAT = 90.0 - (YORD + FLOAT (JY - 1) * DYD) / FACT
DO 260 IX = 1, NXD
XLON = (XORD + FLOAT (IX - 1) * DXD) / FACT
WRITE (6, 600) DP (L), XLON, YLAT, ELVD
260 CONTINUE
DO 270 JY = 1, NP
270 P (JY) = DP (JY)
C IF NIO = 1 WRITE SUSCEPTIBILITIES ONTO USER DEFINED UNIT 33
C IF (NIO .EQ. 1) THEN
WRITE (33, *)lew
WRITE (33, *)ins
WRITE (33, *)scolat
WRITE (33, *)west
WRITE (33, *)gridspc
IJK = 1
DO 375 JY = 1, NYD
IJK = IJK + 1
WRITE (33, ' (4 (£18.8, Ix) ) ') (DP (L), L-IJK, IJK+1)
375 CONTINUE
ENDIF
C IF (NFLD .EQ. 0) GO TO 410
WRITE (6, 620)
C COMPUTE EQUIVALENT SOURCE M-FIELD
840 CONTINUE
WRITE (6, 630)
xord = strxord
yord = stryord
dx = strdx
dyd = strdyd
e1v = strelvd
IF (NFLD .EQ. 2) THEN
READ (13, *)lew
READ (13, *)ins
READ (13, *)scolat
READ (13, *)west
READ (13, *)gridspc
DO 375 JY = 1, NYD
READ (13, *) (DF (IX, JY), IX = 1, NXD)
375 CONTINUE
ENDIF
380 CONTINUE
C IF (MYEAR.LT.1.E-9) GO TO 880
CALL FIELDG (0., 0., 0., 0., 55.1, q1, q2, q3, q4)
IF (MFL+MIL+MDL .GT. 0.0) GO TO 870
CALL GEOMAG (MYEAR, E1V, YORD, XORD, D1V, D2V, D3V, D4V, D5V, D6V, D7V, D8V, D9V, D10V, D11V)
870 IF (MFL+MDL .GT. 0.0) GO TO 880
CALL GEOMAG (MYEAR, MEIVO, MYORO, MXORO, MDYO, MXDO, MNXO, MNW, MNNXO, 10)
880 CONTINUE
C PI = 3.1415926536
FACT = PI / 180.0
MXORO = MXORO * FACT
MYORO = MYORO * FACT
MXDVO = MXDVO * FACT
MYRD = (90.0 - MYORO) * FACT
MDYO = MDYO * FACT
MXDO = MXDO * FACT
D1V = D1V * FACT
D2V = D2V * FACT
D3V = D3V * FACT
D4V = D4V * FACT
D5V = D5V * FACT
D6V = D6V * FACT
D7V = D7V * FACT
D8V = D8V * FACT
D9V = D9V * FACT
D10V = D10V * FACT
D11V = D11V * FACT
D12V = D12V * FACT
D13V = D13V * FACT
D14V = D14V * FACT
D15V = D15V * FACT
D16V = D16V * FACT
D17V = D17V * FACT
D18V = D18V * FACT
D19V = D19V * FACT
D20V = D20V * FACT
.fx 250
C

mI=mI*FACT
mD=mD*FACT
mI1=mI1*FACT
mD1=mD1*FACT
C
C REARTH=6371.0
RO=ELVD+REARTH
RD=ELVD+2.0*RO**2
C
C IF (mYEAR.LT.I.E-9) GO TO 300
IF (mF1+mI1+D1.GT.0.0 .AND. mI+mD.LT.1.1.E-9) GO TO 330
IF (mFI+mII+mDI.LT.0.0 .AND. mI+mD.GT.0.0) GO TO 360
C
C COMPUTE M-FIELD FOR CASE WHERE (mI,mD) AND (mF1,mI1,mD1)
C ARE DERIVED FROM FIELDG
C
DO 290 JY=1,mNYO
THETA=MYORD+ (FLOAT(JY)-1.0)*mDYO
COS= (FLOAT(JY)-1.0)*mDYO
SIN= (FLOAT(JY)-1.0)*mDYO
C
DO 280 IX=1,mNXO
PHI=MXORD+ (FLOAT(IX)-1.0)*mDXO
aI=INC(IX)
aD=DEC(IX)
C
CALL D@KGS2(aI,aD)
C
DFS(IX,JY)+YC
GO TO 390
C
C COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (mI,mD)
C AND (mF1,mI1,mD1)
C
DO 320 JY=1,mNYO
THETA=MYORD+(FLOAT(JY)-1.0)*mDYO
COS= (FLOAT(JY)-1.0)*mDYO
SIN= (FLOAT(JY)-1.0)*mDYO
C
DO 310 IX=1,mNXO
PHI=MXORD+(FLOAT(IX)-1.0)*mDXO
aI=INC(IX)
aD=DEC(IX)
C
CALL D@KGS2(aI,aD)
C
DFS(IX,JY)+YC
GO TO 390
C
C COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (mI,mD)
C AND (mF1,mI1,mD1) ARE DERIVED FROM FIELDG
C
DO 350 JY=1,mNYO
THETA=MYORD+(FLOAT(JY)-1.0)*mDYO
COS= (FLOAT(JY)-1.0)*mDYO
SIN= (FLOAT(JY)-1.0)*mDYO
C
DO 340 IX=1,mNXO
PHI=MXORD+(FLOAT(IX)-1.0)*mDXO
aI=INC(IX)
aD=DEC(IX)
C
CALL D@KGS2(aI,aD,mF1,mI1,mD1)
C
DFS(IX,JY)+YC
GO TO 390
C
C COMPUTE M-FIELD FOR CASE WHERE USER SUPPLIES (mI,mD) AND
C (mF1,mI1,mD1) ARE DERIVED FROM FIELDG
C
DO 380 JY=1,mNYO
THETA=MYORD+(FLOAT(JY)-1.0)*mDYO
COS= (FLOAT(JY)-1.0)*mDYO
SIN= (FLOAT(JY)-1.0)*mDYO
C
C-51
CALL MAGSI (mI,mD)
C
370  DFS(IX, JY)=YC
380  WRITE(6, 580) (DFS(IX, JY), IX=1,mNXO)
390  CONTINUE
C
C COMPUTE MAXIMUM, MINIMUM AND AVERAGE AMPLITUDE VALUES FOR EQUIVALENT SOURCE M-FIELD
C
DSUM=0.0
DMIN=DFS(1,1)
DMAX=DMIN
DO 400 JY=1,mNYO
DO 400 IX=1,mNXO
DSUM=DSUM+DFS(IX, JY)
IF (DMAX.LT.DFS(IX, JY) ) DMAX=DFS(IX, JY)
IF (DMIN.GT.DFS(IX, JY) ) DMIN=DFS(IX, JY)
400  CONTINUE
DSUM=DSUM/FLOAT (mNXO*mNYO)
WRITE (6,640) 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,*)
 myoro/fact
 write (30,*) mxoro/fact
 write (30,*)
 mdyo/fact
 DO 385 JY=1,mNYO
 WRITE (30,580) ( (DFS (IX, JY)-dsum), IX=1,mNXO)
 do 385
 ix=1,mnxo
 dmax=max(dmax,(dfs (ix, Jy)-dsum) )
 dmin=min (dmin,(dfs (ix, Jy)-dsum) )
385  CONTINUE
if (isub .eq. 1) write (6,*), new max-,dmax, new min-,dmin
C
410  CONTINUE
SSUM=0.0
DO 900 II=1,mNXO
DO 900 J=1,mNYO
SSUM=SSUM+ (DF (II, J) -DFS (II, J) ) **2
900  CONTINUE
WRITE (6,790) SSUM
790  FORMAT(' St_ OF SQUARES - ',E15.10)
901  CONTINUE
STOP
C
420  FORMAT (/, IX, 3HFI-,
FI0.3, 5X, 3HI1-,
F6.2, 5X, 3HDI-, F6.2,
15X, 2H-,
F6.2,
/)
430  FORMAT (/, IX, 3HFI-,
FI0.5, 5X,
/)
440  FORMAT (/, IX, 5HYEAR-,
FI0.5, 5X,
/)
450  FORMAT (/, IX, 5HYEAR-,
FI0.5, 5X,
/)n
480  FORMAT (//, 2X, 7HRCOND -,E20.8)
490  FORMAT (//,
2X, 26HATA LEADING MINOR OF ORDER, I5, IX,
24HPIS NOT POSITI
12X,
/)
520  FORMAT (//, 'CGS-SUSCEPTIBILITIES ', 5X, 'E-LONGITUDE ', 5X, 'N-
•', 'LATITUDE ', 5X, 'KM-LEVAT ION ',
/)
600  FORMAT (IX, E20.5,5X, FIO.4,5X, FIO.4,5X, FIO.4)
620  FORMAT (///)
630  FORMAT (//,2X,25HEQUIVALENT SOURCE M-FIELD,/)n
640  FORMAT (///)
C
END
SUBROUTINE MAGSI (I,D)
C
************************************************************************
C
SUBROUTINE M_GSI CALCS THE TOTAL MAGNETIC FIELD AT A SPHERICAL OBSERVATION POINT (R,THETA,PHI) DUE TO A SPHERICAL ARRAY OF POINT DIPOLES WITH CGS-SUSCEPTIBILITIES P (L) AT SOURCE GRID COORDINATES C
POLARIZING FIELD CHARACTERISTICS (FI,II,DI) ARE READ FROM TAPE1.
************************************************************************
C
REAL
fld0, lncd, lnco, decd, deco
dimension fldd (44,181),
LDI (181),DEC1 (181)
YC=0.0
I=0
C
C SUBROUTINE MAGSI CALCULATES THE TOTAL MAGNETIC FIELD AT A SPHERICAL OBSERVATION POINT (R,THETA,PHI) DUE TO A SPHERICAL ARRAY OF POINT DIPOLES WITH CGS-SUSCEPTIBILITIES P (L) AT SOURCE GRID COORDINATES POLARIZING FIELD CHARACTERISTICS (FI,II,DI) ARE READ FROM TAPE1.
C
COMMON
XORD, YORD, DXD, DYO, NXD, NYD, RO, RD, E, G, COST, SINT, YC, THETA, PHI,
>P (3982), TMAG (3982), DF (361,44), S (7930153)
real fldd, fldo, lncd, lnco, decd, deco
dimension fldd(44,181), lncd(44,181), lnco(44,361),
decd(44,181), deco(44,361)
common /gmf/fldd, fldo, lncd, lnco, decd, deco
REAL 1,11, JR, JTHETA, JPHI, IXD1(181), IXDI(181), DECD1(181)
YC=0.0
I=0
C
C -52
DO 110 JY=1,NXD
   THETA=FORD+FLOAT(JY-1)*DYD
   COST1=COS(THETA)
   SINT1=SIN(THETA)
   CTIT=COST1*COST1
   SINTIT=SINT1*SINT1
   CTITIT=COST1*COST1

   DO 730 K=1,NXD
      FLDL(K)=FLDD(JY,K)
      INCL(K)=INCD(JY,K)
      DECL(K)=DECL(JY,K)
   730 CONTINUE

DO 110 IX=1,NXD
   L=L+1

   C CPUTE POINT DIPOLE POLARIZATION VECTOR IN SPHERICAL ORTHONORMAL
   UNIT VECTORS (ER,ETHETA,EPHI) WITH SUCP(L)=1.0
   C
   J=FLDL(IX)
   I=INCL(IX)
   D=DECL(IX)

   JR=J*SIN(I)
   JTHETA=J*COST(I)*COST(D)
   JPHI=J*SINT(I)*SINT(D)

   PHII=J*(RO+R0)*SQR(D)+J*(RO-R0)*SQR(D)
   ATHETA=COST(I)*SINT1*COST(D)
   BTHETA=SINT1*COST1*COST(D)
   CTHETA=SINT1*COST1*SINT(D)
   APHI=SINT1*COST1*SINT(D)
   BPHI=COST1*COST1*SINT(D)
   CPHI=COST1*COST1
   C
   U=JR*ATHETA+BPHI-JPHI*CPHI
   U2=3.0*XX*RO*ATHETA
   U3=U/RI5
   U4=U2/R25
   U=U3+U4

   V=JR*JTHETA+BTHETA-JPHI*CPHI
   V2=3.0*XX*RO*BTHETA
   V3=V/RI5
   V4=V2/R25
   V=V3+V4

   W=JR*APHI+BPHI-JPHI*JPHI
   W2=3.0*XX*RO*APHI
   W3=W1/R15
   W4=W2/R25
   W=W3+W4

   CALCULATE THE COMPONENT OF THE ANOMALOUS FIELD IN THE DIRECTION
   OF THE GEOMAGNETIC FIELD AT THE OBSERVATION POINT---I.E.,
   ((U,V,W) *(UR,UTHETA,UPHI)) \* P(L) = TOTAL MAGNETIC FIELD
   C
   UR=UR1*UR
   UTHETA=UTHETA1*UTHETA
   UPHI=UPHI1*UPHI

   TMAGL=JR*JTHETA+BTHETA-JPHI*JPHI
   YC=YC+TMAGL

110 CONTINUE

C RETURN
C
END

SUBROUTINE MAGS2 (I,D,FI,II)
C************************************************************************
C SUBROUTINE MAGS2 CALCULATES THE TOTAL MAGNETIC FIELD AT A SPHERICAL
C OBSERVATION POINT (R,THETA,PHI) DUE TO A SPHERICAL ARRAY OF POINT
C DIPOLES WITH CGS-SUSCEPTIBILITIES P(L) AT SOURCE GRID COORDINATES
C************************************************************************
C-53
COMMON XORD, YORD, DXD, DYD, MXD, MYD, RO, RD, E, G, COST, SINT, YC, THETA, PHI,
T (3982), TNDG (3982), DF (44, 44), S (7930153)
REAL 1,1, JR, JTHETA, JPHI
YC=0.0
L=0
DO 110 JY=1,NYD
THETA=FLOAT(JY-1)*DYD
COST=FLOAT(COST)+COST
SINT=FLOAT(SINT)+SINT
CTIT=COST1*COST
STIT=SINT1*SINT
SCIT=SINT1*COST
CSIT=COST1*SINT
110 CONTINUE
RETURN
END

SUBROUTINE GEOMAG (YEAR, ELV, THETA, PHI, HTHETA, HPHI, NTHETA, NPHI, NF)
REAL FLDG, FLOD, INCD, INCO, DECD, DECO
COMMON /GMF/ FLDG, FLOD, INCD, INCO, DECD, DECO

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

** FIELDG ** (NASA)  
** FIELD ** (NASA)  
** WRITEB ** (SYSTEM)  

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 source grid'

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

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

RETURN

END
**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, P---ALL VALUES ARE IN GAMMAS

**EQUIVALENCE (SHMIT(I,I),TG(I,I))**  
**COMMON /NASA/ TS(55,55)**  
**COMMON /FLDCOM/ CPH, SPH, R, CT, ST, BT, BP, BR, B**  
**COMMON /MAX/ MAXX**  
**DIMENSION G(55,55), GT(55,55), SHMIT(55,55), AID(55), GTT(55,55)**  
**DATA A/0./**

**TLAST=0.0**

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

**A=6378.16**  
**FLAT=1./298.25**

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

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

**READ (3,280) N,M,GNM, HNM, GTNM, HTNM, GTTNM, HTT**  
**IF (N.LE.0) GO TO 130**  
**MAXN=MAX(N, MAXN)**

**G(N,M)=GNM**

**GT(N,M)=GTNM**

**TEMP=MAX1(TEMP,ABS(GT(N,M)))**

**IF (M.EQ.I) GO TO 120**  
**G(M-I,N)=HNM**

**GT(M-I,N)=HTNM**

**GTT(M-I,N)=HTT**

**GO TO 120**

**130 continue**

**rewind (3)**

**c 130 WRITE(6,290)**

**c DO 150 N-2,MAXN**

**c DO 150 M=1,N**

**c MI=M-I**

**c IF (M.EQ.I) GO TO 140**

**c WRITE(6,300) N,M,G(N,M),G(M-I,N), GT(N,M),GT(M-I,N), GTT(N,M),GTT(M-I, N)**

**c GO TO 150**

**c 140 WRITE(6,310) N,M,G(N,M),GT(N,M),GTT(N,M)**

**c 150 continue**

**WRITE(6,320)**

**c IF (TEMP.EQ.0.) L=1**

**c 160 IF (K.NE.0) GO TO 190**

**SHMIT(I,1)=1.**

**DO 170 N=2,MAXN**

**SHMIT(N,1)=SHMIT(N,1)*FLOAT(2*N-3)/FLOAT(N-1)**

**SHMIT(I,1)=0.**

**L=2**

**DO 170 M=N,N**

**SHMIT(N,M)=SHMIT(N,M-1)*SQRT(FLOAT(N-M+1)*JJ)/FLOAT(N+M-2))**

**SHMIT(M,1)=SHMIT(M-1,N)**

**170 J=1**

**DO 180 H=1,N**

**G(N,M)=G(N,M)*SHMIT(N,M)**

**GT(N,M)=GT(N,M)*SHMIT(N,M)**

**GT(N,M)=GT(N,M)*SHMIT(N,M)**

**IF (M.EQ.I) GO TO 180**

**G(M-I,N)=G(M-I,N)*SHMIT(M-I,N)**

**GT(M-I,N)=GT(M-I,N)*SHMIT(M-I,N)**

**GT(M-I,N)=GT(M-I,N)*SHMIT(M-1,N)**

**180 continue**

**190 T=TM-TZERO**

**DO 200 N=1,MAXN**

**DO 200 M=1,N**

**TG(N,M)=TG(N,M)*T*GT(N,M)*GTT(N,M)**

**IF (M.EQ.1) GO TO 200**

**TG(M-I,N)=TG(M-I,N)*T*GT(M-I,N)*GTT(M-I,N)**

**C -56**
200 CONTINUE
TLM=TM
210 DLAT=DLAT/57.2957795
SINLA=SIN(DLATR)
RLONG=RLONG/57.2957795
COS=CO5(RLONG)
SIN=SI5(RLONG)
IF (.EQ.0) GO TO 220
N=ALT+6371.0
CT=SINLA
GO TO 230
220 SINLA2=SINLA**2
DEN2=A2-A2B2*ST
DEN=SQRT(DEN2)
FAC=(((ALT*DEN)+A2)/((ALT*DEN)+B2))**2
CT=SINLA/SQRT(FAC*COSLA2+DEN)
R=SQRT(ALT*(ALT+2.*DEN)+(A4-A4B4*SINLA2)/DEN2)
230 X=CT**2
NMAX=MIND(NMAX,MAXN)
CALL FIELD
Y=BP
IF (.EQ.0) GO TO 240
240 X=BT
Z=BR
RETURN
C TRANSFORMS FIELD TO GEODETIC DIRECTIONS
250 SIND=SINLA*ST-SQRT(COSLA2)*CT
COSD=SQRT((1.0-SIND)**2)
X=CT*COSD+BR*SIND
Z=ST*SIND-CT*BR
RETURN
C FORMATTING
260 FORMAT (11X,F6.1,10A6,A3)
270 FORMAT (11X,F5.1,10A6,A3)
280 FORMAT (11X,F7.1,A4)
290 FORMAT (11X,6H0,6H ,I8,6X,10A6)
300 FORMAT (11X,F7.1,A4)
310 FORMAT (11X,6F11.4,11X,F11.4,11X,F11.4)
320 FORMAT (11X,6F11.4,11X,F11.4,11X,F11.4)
END
SUBROUTINE FIELD
COMMON /NASA/ G(55,55)
COMMON /FLDCAM/ CPH,SPH,CT,ST,BT,BR,B
COMMON /MAX/ DIMENSION P(55,55), DP(55,55), CONST(55,55), SP(55), CP(55),
> FN(55), FM(55)
DATA P(I,I)/0./
IF (P(I,I).EQ.1.0) GO TO 120
P(I,I)=1
DO 110 M=I,N
DO 110 N=I,M-1
FN(M,N)=0
FM(N,M)=0
110 CONST(M,N)=FLOAT((N-2)**2-(M-1)**2)/FLOAT((2*N-3)*(2*N-1))
120 CP(2)=SPH
CP(2)=CPH
DO 130 M=2,NMAX
DO 130 N=2,M-1
130 CP(M)=CP(2)*CP(M-1)+CP(2)*SP(M-1)
AR=6371.0/R
IF (N-M) 150,140,150
PP(P(N,M))=AR
IF (M.I.EQ.1) GO TO 160
DO 140 P(N,M)=ST*DP(N-1,1)+CT*P(N-1,1)
DO 140 N=2,NMAX
140 IF (N-I) 150,140,150
DP(N,M)=CT*DP(N-1,1)-ST*P(N-1,1)
G0 TO 160
150 IF (N-M) 170,160,150
160 PAR=PAR+P(N,M)*AR
IF (M.EQ.1) GO TO 170
TO COMPUTE ON ENTRY
\[
\begin{align*}
&\text{TEMP} = G(N,M) \cdot CP(M) + G(M-1,N) \cdot SP(M) \\
&\text{BP} = \text{FP} - G(N,M) \cdot SP(M) - G(M-1,N) \cdot CP(M) \cdot FM(M) \cdot \text{PAR}
\end{align*}
\]
GO TO 180

\[
\begin{align*}
&\text{TEMP} = G(N,M) \cdot CP(M) \\
&\text{BP} = \text{FP} - G(N,M) \cdot SP(M) - G(M-1,N) \cdot CP(M)
\end{align*}
\]

RETURN

END

SUBROUTINE SPPCO (AP, N, RCOND, Z, INFO)
INTEGER N, INFO
REAL AP(1), Z(1)
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**2) - 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, SDOT, SSCAL, 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 FIND NORM OF A
C
J1 = 1
DO 30 J = 1, N
Z(J) = SASUM(J, AP(JI), I)
J1 = J1 + 1
JMI = J - 1
IF (JMI .LT. I) GO TO 20
DO 10 I = I, JMI
Z(I) = Z(I) + ABS(AP(IJ))
I0 CONTINUE
20 CONTINUE
30 CONTINUE
ANORM = 0.0E0
DO 40 J = 1, N
ANORM = MAX(ANORM, Z(J))
40 CONTINUE

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.

SOLVE TRANS(R)*W = E
EK = 1.0E0
DO 50 J = 1, N
Z(J) = 0.0E0
50 CONTINUE

KK = 0
DO 60 K = 1, N
KK = KK + K
IF (Z(K) .NE. 0.0E0) EK = SIGN(EK, Z(K))
60 CONTINUE

WK = EK
WKM = -EK
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
Z(J) = Z(J) + WK*AP(KJ)
S = S + ABS(Z(J))
KJ = KJ + J
70 CONTINUE

T = WKM
WK = WK
KJ = KK + K
DO 80 J = KPI, N
Z(J) = Z(J) + T*AP(KJ)
KJ = KJ + J
80 CONTINUE

SOLVE R*Y = W

DO 130 KB = 1, N
K = N + 1 - KB
IF (ABS(Z(K)) .LT. AP(KK)) GO TO 120
S = AP(KK) / ABS(Z(K))
120 CONTINUE

C SOLVE R*Y = W
Z(K) = Z(K)/AP(KK)
KK = KK - K
T = -Z(K)
CALL SAXPY(K-1,T,AP(KK+1),1,Z(1),1)

130 CONTINUE
S = 1.0E0/SASUM(N,Z,1)
CALL SSCAL(N,S,Z,1)

YNORM = 1.0E0

SOLVE TRANS(R)*V = Y

DO 150 K = 1, N
  Z(K) = Z(K) - SDOT(K-1,AP(KK+1),1,Z(1),1)
  KK = KK + K
  IF (ABS(Z(K)) .LE. AP(KK)) GO TO 140
  S = AP(KK)/ABS(Z(K))
  CALL SSCAL(N,S,Z,1)
  YNORM = S*YNORM
140 CONTINUE

SOLVE R*Z = V

DO 170 K = 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)
170 CONTINUE

MAKE YNORM = 1.0
S = 1.0E0/SASUM(N,Z,1)
CALL SSCAL(N,S,Z,1)
YNORM = S*YNORM

IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0

180 CONTINUE

RETURN

END

SUBROUTINE SPPFA(AP,N,INFO)
INTEGER N, INFO
REAL AP(N)

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
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
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 = 1, N
   INFO = J
   S = 0.0E0
   JMI = J - 1
   KJ = JJ
   KK = 0
   IF (JMI .LT. 1) GO TO 20
   DO 10 K = 1, JMI
      KJ = KJ + 1
      T = AP(KJ) - SDOT(K-1,AP(K+1),1,AP(J+1),1)
      KK = KK + K
      T = T/AP(KK)
      AP(KJ) = T
      S = S + T*T
10  CONTINUE
20  CONTINUE
   JJ = JJ + J
   S = AP(JJ) - S
C ...... EXIT
   IF (S .LE. 0.0E0) GO TO 40
   AP(JJ) = SQRT(S)
30  CONTINUE
40  CONTINUE
RETURN
END

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

C
C
C ...

COMPUTER - CDC/SINGLE
LATEST REVISION - JANUARY 1, 1978
PURPOSE - COMPUTE A CONSTANT TIMES A VECTOR PLUS A VECTOR, ALL SINGLE PRECISION
USAGE - CALL SAXPY (N, SA, SX, INCX, SY, INCY)
ARGUMENTS N - LENGTH OF VECTORS X AND Y. (INPUT)
SA - REAL SCALAR. (INPUT)
SX - REAL VECTOR OF LENGTH MAX(N*IABS(INCX),1). (INPUT)
SX - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1). (INPUT)
INCX - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
SY - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1). (INPUT)
SY - REAL VECTOR OF LENGTH MAX(N*IABS(INCY),1). (INPUT)
SAXPY REPLACES Y(I) WITH SA*X(I)+Y(I)
FOR I=1,...,N.
X(I) AND Y(I) REFER TO SPECIFIC ELEMENTS OF SX AND SY, RESPECTIVELY. SEE INCX AND
INCY ARGUMENT DESCRIPTIONS.
INCY - DISPLACEMENT BETWEEN ELEMENTS OF SY. (INPUT)
SY(I) IS DEFINED TO BE...
SY(I) IS DEFINED TO BE...
PRECISION/HARDWARE - SINGLE/ALL
REQD. IMSL ROUTINES - NONE REQUIRED
NOTATION - INFORMATION ON SPECIAL NOTATION AND
CONVENTIONS IS AVAILABLE IN THE MANU-
INTEGER N, INCX, INCY
REAL SX(I), SY(I), SA

SPECIFICATIONS FOR LOCAL VARIABLES

IF (N.LE.0 OR SA.EQ.0.E0) RETURN
IF (INCX.EQ.INCY) IF (INCX-1) 5,15,35
5 CONTINUE

CODE FOR NONEQUAL OR NONPOSITIVE INCREMENTS.

IX = 1
IY = 1
IF (INCX.LT.0) IX = (-N+1)*INCX+1
IF (INCY.LT.0) IY = (-N+1)*INCY+1
DO 10 I=1,N
SY(IY) = SY(IY)+SA*SX(IX)
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 4.

15 M = N-(N/4)*4
IF (M.EQ.0) GO TO 25
DO 20 I=1,M
SY(I) = SY(I)+SA*SX(I)
20 CONTINUE
IF (M.LT.4) RETURN
25 MPI = M+1
DO 30 I=MPI,N,4
SY(I) = SY(I)+SA*SX(I)
SY(I+1) = SY(I+1)+SA*SX(I+1)
SY(I+2) = SY(I+2)+SA*SX(I+2)
SY(I+3) = SY(I+3)+SA*SX(I+3)
30 CONTINUE
RETURN

CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.

35 CONTINUE
NS = N*INCY
DO 40 I=NS+1,NS+INCY
SY(I) = SA*SX(I)+SY(I)
40 CONTINUE
RETURN

REAL FUNCTION SDOT(N, SX, INCX, SY, INCY)

COMPUTER - CDC/SINGLE
LATEST REVISION - JANUARY 1, 1978
PURPOSE - COMPUTE SINGLE PRECISION DOT PRODUCT
USAGE - 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.
N - LENGTH OF VECTORS X AND Y. (INPUT)
SX - REAL VECTOR OF LENGTH MAX(N*INCS(INCX),1). (INPUT)
INCX - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
SY - REAL VECTOR OF LENGTH MAX(N*INCS(INCY),1). (INPUT)
INCY - DISPLACEMENT BETWEEN ELEMENTS OF SY. (INPUT)
PRECISION/HARDWARE - SINGLE/ALL

REQU. IMSL Routines - NONE REQUIRED

-62
SUBROUTINE SSCAL (N, SA, SX, INCX)

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

INTEGER I, M, MP1, NS, IX, IY

FIRST EXECUTABLE STATEMENT

SDOT = 0.0
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 (INCY.LT.0) IX = (-N+I)*INCX+I
IF (INCY.LT.0) IY = (-N+I)*INCY+I
DO 10 I = I, 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.

15 M = N-(N/5)*5
IF (M.EQ.0) GO TO 25
DO 20 I = 1, M
SDOT = SDOT+SX(I)*SY(I)
20 CONTINUE
IF (N.LT.5) RETURN
25 NS = M+1
DO 30 I = M+1, N
SDOT = SDOT+SX(I)*SY(I)
SDOT = SDOT+SX(I)*SY(I)
SDOT = SDOT+SX(I)*SY(I)
30 CONTINUE
RETURN

CODE FOR POSITIVE EQUAL INCREMENTS

40 CONTINUE
RETURN
END

SUBROUTINE SSCAL (N, SA, SX, INCX)

C
C PURPOSE - COMPUTE A SINGLE PRECISION CONSTANT TIMES A SINGLE PRECISION VECTOR
C
C USAGE - CALL SSCAL (N, SA, SX, INCX)
C
C ARGUMENTS
N - LENGTH OF VECTOR X. (INPUT)
SA - REAL SCALAR. (INPUT)
SX - REAL VECTOR OF LENGTH N*INCX. (INPUT/OUTPUT)
C SSCAL REPLACES X(I) WITH SA*X(I) FOR I=1,...,N.
C X(I) REFERS TO A SPECIFIC ELEMENT OF SX.
C SEE INX ARGUMENT DESCRIPTION.
C INCX - DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
C X(I) IS DEFINED TO BE SX(I+(I-1)*INCX).
C INCX MUST BE GREATER THAN ZERO.
C
C PRECISION/HARDWARE - SINGLE/ALL
C REQD. IMSL ROUTINES - NONE REQUIRED
C
C NOTATION - INFORMATION ON SPECIAL NOTATION AND CONVENTIONS IS AVAILABLE IN THE MANUAL
C INTRODUCTION OR THROUGH IMSL ROUTINE UHELP

C -63
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 INCX, N
REAL SA, SX(I)

SPECIFICATIONS FOR LOCAL VARIABLES

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.

CODE FOR INCREMENTS EQUAL TO 1.

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

I0 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

M01 = M+1
DO 25 I=M01,NS,INCX
SX(I) = SA*SX(I)
SX(I+I) = SA*SX(I+I)
SX(I+2) = SA*SX(I+2)
SX(I+3) = SA*SX(I+3)
SX(I+4) = SA*SX(I+4)
25 CONTINUE
RETURN

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)
S X(I) REFERS TO A SPECIFIC ELEMENT OF SX. (INPUT)
N = LENGTH OF VECTOR X. (INPUT)
SX = REAL VECTOR OF LENGTH N*INCX. (INPUT)
INCX = DISPLACEMENT BETWEEN ELEMENTS OF SX. (INPUT)
X(I) IS DEFINED TO BE SX(I+(I-1)*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

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
REAL SX(I)

SPECIFICATIONS FOR LOCAL VARIABLES

INTEGER I, M, MPI, NS

FIRST EXECUTABLE STATEMENT

SASUM = 0.0E0
IF (N.LE.0) RETURN
IF (INCX.EQ.1) GO TO 10

C-64
C

NS = N*INCX
DO 5 I=1,NS,INCX
      SASLM = SASUM+ABS(SX(I))
5 CONTINUE
RETURN
C

CODE FOR INCREMENTS NOT EQUAL TO 1.
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 1=1,M
      SASLM = SASUM+ABS(SX(I))
15 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
   N
   B
ON RETURN
   B
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
      CALL SPPSL(AP,N,C(1,J))
10 CONTINUE
LINPACK. THIS VERSION DATED 06/14/78.
CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
SUBROUTINES AND FUNCTIONS
BLAS SAXPY, SDOT
INTERNAL VARIABLES
REAL SDOT, T
INTEGER K, KB, KK
C
KK = 0
DO 10 K = 1, N
   K = KK + K
   B(K) = T
   CALL SPPSL(AP(K+1),1,B(1),1)
10 CONTINUE
RETURN
END
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 password(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), passmind(4000, 2)
common /hsort/ ra

ccc- program description
ccc check locates all passes with a minimum below a user defined
ccc value, a maximum above a user defined value, a variance above
ccc a user defined maximum, a mean value beyond a user defined value
ccc and a difference between adjacent observations greater than a
ccc user defined value. If a pass is selected, then the pass number
ccc and the above values are written to the screen. this program
ccc can be used on direct access 21-27r files or on files with a
ccc header and one variable per pass. this program is used to find
ccc passes which are influenced by external fields as noted by
ccc their variance properties.
ccc
ccc program date: 22 apr 91
ccc
ccc updates: 6 jun 92; added sort subroutine
ccc NOTE: check now removes the checked passes (i.e. the high
ccc variance passes) from the ordered pass number file
ccc written by reorder.
ccc NOTE: now check also orders the output file according to
ccc the average elevation of the pass;
ccc NOTE: these new options are not available for 21-27r input.
ccc 20 jul 92; added output file on unit 21
ccc NOTE: this update simplifies the usage of check
ccc
ccc write (*,*) '0 IF INPUT FILE IS 21-27R'
ccc write (*,*) '1 IF INPUT FILE IS HEADER AND VARIABLE'
ccc read (*,*) type
ccc if (type) 50,50,200
ccc 50 write (*,*) 'INPUT 21-27R FILE:'
ccc read (*,9990) filename
ccc format (a80)
ccc open (i0, file-filename, status-'old', form-'unformatted',
ccc access='direct', recl-116)
ccc 9990 format(a80)
ccc c--------------------------- program description
ccc
ccc check locates all passes with a minimum below a user defined
ccc value, a maximum above a user defined value, a variance above
ccc a user defined maximum, a mean value beyond a user defined value
ccc and a difference between adjacent observations greater than a
ccc user defined value. If a pass is selected, then the pass number
ccc and the above values are written to the screen. this program
ccc can be used on direct access 21-27r files or on files with a
ccc header and one variable per pass. this program is used to find
ccc passes which are influenced by external fields as noted by
ccc their variance properties.
ccc
ccc program date: 22 apr 91
ccc
ccc updates: 6 jun 92; added sort subroutine
ccc NOTE: check now removes the checked passes (i.e. the high
ccc variance passes) from the ordered pass number file
ccc written by reorder.
ccc NOTE: now check also orders the output file according to
ccc the average elevation of the pass;
ccc NOTE: these new options are not available for 21-27r input.
ccc 20 jul 92; added output file on unit 21
ccc NOTE: this update simplifies the usage of check
ccc
ccc write (*,*) '0 IF INPUT FILE IS 21-27R'
ccc write (*,*) '1 IF INPUT FILE IS HEADER AND VARIABLE'
ccc read (*,*) type
ccc if (type) 50,50,200
ccc 50 write (*,*) 'INPUT 21-27R FILE:'
ccc read (*,9990) filename
ccc format (a80)
ccc open (i0, file-filename, status-'old', form-'unformatted',
ccc access='direct', recl-116)
ccc 9990 format(a80)
ccc c--------------------------- program description
ccc
ccc check locates all passes with a minimum below a user defined
ccc value, a maximum above a user defined value, a variance above
ccc a user defined maximum, a mean value beyond a user defined value
ccc and a difference between adjacent observations greater than a
ccc user defined value. If a pass is selected, then the pass number
ccc and the above values are written to the screen. this program
ccc can be used on direct access 21-27r files or on files with a
ccc header and one variable per pass. this program is used to find
ccc passes which are influenced by external fields as noted by
ccc their variance properties.
ccc
ccc program date: 22 apr 91
ccc
ccc updates: 6 jun 92; added sort subroutine
ccc NOTE: check now removes the checked passes (i.e. the high
ccc variance passes) from the ordered pass number file
ccc written by reorder.
ccc NOTE: now check also orders the output file according to
ccc the average elevation of the pass;
ccc NOTE: these new options are not available for 21-27r input.
ccc 20 jul 92; added output file on unit 21
ccc NOTE: this update simplifies the usage of check
ccc
ccc write (*,*) '0 IF INPUT FILE IS 21-27R'
ccc write (*,*) '1 IF INPUT FILE IS HEADER AND VARIABLE'
ccc read (*,*) type
ccc if (type) 50,50,200
ccc 50 write (*,*) 'INPUT 21-27R FILE:'
ccc read (*,9990) filename
ccc format (a80)
ccc open (i0, file-filename, status-'old', form-'unformatted',
ccc access='direct', recl-116)
ccc 9990 format(a80)
c 9992 format (215,5(1x,f12.6))
c continue

data(n, var)=dummy
c continue
do 28 i=1,2
idata(1,1)=data(n,1)
do 29 j=1,2
idata(1,j)=data(n,j)
continue
if (jstop .eq. 1) go to 999
go to 22

200 continue
c 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 VARIANCE'
read (*, meanmax, diffmax, rmax, mlnmin, 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
if (choice.eq.'yes' .or. choice.eq.'YES') then
write (*,*) 'INPUT FILE OF ORDERED PASS NUMBERS'
read (*,9990) filename
open (11, file=filename, form='formatted',status='old')
write (*,*) 'OUTPUT FILE = PASS NUMBERS - CHECKED NUMBERS'
read (*,9990) filename
open (20, file=filename, form='formatted')
icnt=0
do i=1,10000
read (11,9991) test
icnt=icnt+1
totai_.0
xsumsqr=.0
minval=10e10
maxdiff=10e-10
maxval=10e10
read (11,*) (passmjd(i,j),j=1,2), (passord(i,j),j=1,2)
data(row+1, var)=data(row, var)
write (*,*)'PASSNO CNT MAX MEAN MIN',
write (*,*)'DIFF MAXVAR'
c 210 read (10, end=500) row, col, zero, xmean, pass, eight
do 220 i=1, row
read (10) (data(i,j),j=1, col)
continue
countall=countall+1
 totals=0.0
 xsumsqr=0.0
minval=10e10
maxval=10e10
maxdiff=10e-10
data(row+1, var)=data(row, var)
do 240 l=1,row
  total=total+data(l,var)
xsumsqr=xsumsqr+(data(l,var)**2)
  maxval=max(maxval,data(l,var))
minval=min(minval,data(l,var))
diff=data(l,var)-data(l+1,var)
  maxdiff=max(maxdiff,abs(diff))
240 continue
  mean=total/row
  xvar=(xsumsqr-((total)**2)/real((row))/real((row-1))
  absmean=abs(mean)
  if (absmean.gt.meanmax .or. maxval.gt.maxmax .or. 
      minval.lt.minmin .or. maxdiff.gt.diffmax .or. 
      xvar.gt.mxvar) then
    write (*,9992) pass,row,maxval,mean,minval,maxdiff,xvar
    if (choice.eq.'yes' .or. choice.eq.'YES') then
      write (21,9992) pass,row,maxval,mean,minval,maxdiff,xvar
  endif
  count=count+1
  passnum(pass)=count
  endif
  go to 210
Continued
500 continue
  if (choice.eq.'yes' .or. choice.eq.'YES') then
    cnt=0
    do l=1,cnt
      do j=1,1
        if (passmjd(l,j) .eq. passnum(j)) go to 510
      enddo
    cnt=cnt+1
    enddo
      ra(cnt)=passord(l,2)
510 continue
  endif
  call sort (cnt)
  do l=1,cnt
    do j=1,cnt
      if (ra(l) .eq. passord(j,2) ) then
        write (20,') (passmjd(j, j),j=1,2),passord(j,1),
        real(passord(j,2))
        go to 520
      endif
    enddo
  enddo
520 continue
  endif
  enddo
999 continue
  close (10)
close (11)
close (20)
write (*,* )'total passes counted = ',count
write (*,* )'total passes checked = ',count
write (*,* )'total passes written to file = ',cnt
stop
end
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

L=N/2+1
IR=N
100 CONTINUE
  IF (L.LT.1) THEN
    L=1
    RRA=RA(L)
  ELSE
    RRA=RA(IR)
    RA(IR)=RA(L)
    IR=IR-1
  IF (IR.EQ.1) THEN
    RA(1)=RRA
    RETURN
  ENDIF
  ENDIF
  L=L+L
200 IF (J.LE.IN) THEN
  IF (J.EQ.IN) THEN

D-4
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)

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

write (*,*), 'INPUT MATRIX WITH TPILOT HEADER'
read (*,9990) filename(1)
9990 format (a70)
open (10, file=filename(i), 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 i0
50 count-i
write (*,*) 'total number of input data sets -',count
do 60 i=1,count-i
if (col(1) .ge. col(i+1) .or. row(i) .ge. 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
60 continue

c-------------------from 200 to write statement of variables is
the statistical calculations using two

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

do 400 i=1,count-i
   do 400 j=1,nobs
      sdata(i)=data(j,i)
data(i)=data(i)(1,1,n)
do 400 k=1,col(i)
j=(col(i)*j-1)+k
   ydata(i)=ydata(i)(1,1,k)
data(i)=data((a),k,1)
ydata(i)=ydata(i)(1)
max=xdata(i)
min=ydata(i)

100 continue

D-7
ysumsqr=ysumsqr+(ydata(j4))^2
sumxy=sumxy+(xdata(j4)*ydata(j4))

max=max(max(xmax,xdata(j4))
min=min(min(xmin,xdata(j4))
ymax=max(ymax,ydata(j4))
ymin=min(ymin,ydata(j4))

240 continue

c-------------------find corrected sum of products, covariance
and corrected sum of squares (x) (y)
c
xmean=xsum/nobss
ymean=ysum/nobss
sumprod=sumxy-((xsum*ysum)/nobss)
covarxy=sumprod/(nobss-1.0)
xsumsq=xsumsq-((xsum^2)/nobss)
ysumsq=ysumsqr-((ysum^2)/nobss)

240 continue

c-------------------find variance, standard deviation for x and y
c
xvar=xsumsq/(nobss-1.0)
yvar=ysumsqr/(nobss-1.0)
xstdev=sqrt(xvar)
ystdev=sqrt(yvar)

c-------------------find correlation coefficient by Davis method

corrdxy=covarxy/(xstdev*ystdev)

c-------------------find slopes, intercepts and correlation

coefficient by Young method

gxslope=((nobss*sumxy)-(xsum*ysum))/((nobss*xsumsq)-xsum^2)

yslope=((nobss*sumxy)-(xsum*ysum))/((nobss*ysumsq)-ysum^2)
xintcpt=(ysum*xsumsq)-(sumxy*xsum)/((nobss*xsumsq)-xsum^2)
yintcpt=(xsum*ysumsqr)-(sumxy*ysum)/((nobss*ysumsq)-ysum^2)
corrxy=sqrt(nxlope*yslope)

240 continue

c-------------------write out this mess for individual pass and
overlapping lengths of passes

c
write (25,*) 'X- ',filename(icnt)
write (25,*) 'Y- ',filename(jcnt)
write (25,9992) xmean,ymean,xvar,yvar,xstdev,ystdev
9992 format('X MEAN-',f9.3,' Y MEAN-',f9.3,/,> 'X VARIANCE-',f9.1,' Y VARIANCE-',f9.1,' X STDEV-',f8.3,' Y STDEV-',f8.3)
write (25,9993) covarxy,corrdxy
9993 format ('COVARIANCE XY-',f9.1,' Davis CORRELATION COEF-',f8.3)
write (25,9994) xslope,xintcpt,yslope,yintcpt,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) xmax,xmin,ymax,ymin
write (*,*),corrDxy
9995 format('X-MAX=',f9.3,' X-MIN=',f9.3,' Y-MAX=',f9.3,>
> 'Y-MIN=',f9.3,/)
program part2
  c**** convert magsat text from ebcidc to ascii, reorder integer bytes,
  c**** and translate ibm real to dec real
  c**** editorial note: this code was supplied quite generously by Dr. Gary P.
  c**** Murdock
  c
  implicit none

  c**** parameter storage:
  integer reflen
  parameter (relen=3024)

  c**** common storage:
  character*1 ascconv(256)
  common /elacm/ ascconv
  integer*4 recnum,position
  common /xxyyzz/recnum,position

  c**** equivalence storage:
  integer*4 inbufl(relen/4),outbufl(relen/4)
  character*1 inbufc(relen),outbufc(relen)
  equivalence (inbufl,inbufc),(outbufl,outbufc)

  c**** local storage:
  integer il,i2
  character*80 f1lename

  c**** data: (0-no translate, 1=real*4, 2=integer*4, 3=ebcdic)
  integer*2 headtyp(557) /4*2,4*1,1*3,2*2,6*0,12*1,30*3,3*2,>
  490*1/
  integer*2 datatyp(756) /5*2,691*1,30*2,30*0/

  c**** functions:
  integer*4 realconv

  c**** constants:
  cl - char(1)
  c
  write (*,*) 'input file:'
  read (*,99901) f1lename

  99901 format (a80)
  open (21, file=f1lename,status='old',
       access='direct',form='formatted',recl=relen)
  write (*,*) 'output file:'
  read (*, 99901)
  filename
  open (31, file=f1lename,
       access='direct',form='formatted',recl=relen)

  recnum = 1
  read (21,92101,rec=recnum,err=200) inbufc
  92101 format (50000a)
  if (inbufc(4).eq.cl) then
    do position = 1,557
      goto (104,101,102,103),headtyp(position)+1
      outbufl(position) = realconv(inbufc(position))
    goto 104
  101 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
  102 il = position*4
      do il-3,il-1
        outbufc(il) = ascconv(char(inbufc(il))+1)
      end do
  103 end do
  104 else
    do position = 1,756
      goto (108,105,106,107),datatyp(position)+1
      outbufl(position) = realconv(inbufl(position))
    goto 108
  105 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
  106 il = position*4
      do il-3,il-1
        outbufc(il) = ascconv(char(inbufc(il))+1)
      end do
  107 end do
  108 end if
  write (31,92101,rec=recnum) outbufc
  recnum = recnum+1
  goto 100
INTEGER*4 FUNCTION KEALCONV (IBM)

C
C    IMPLICIT NONE
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
C    KEEP COUNT
COUNT = 0
100 IDEC = ISHIFT(IIBM,1)
IF (CDEC(4).NE.CO) GOTO 110
COUNT = COUNT+1
GOTO 100
C
C    EXTRACT AND CLEAR SIGN BIT
110 SIGNFLAG = BTEST(IIBM,7)
IIBM = INCLR(IIBM,7)
C
C    CALCULATE NEW EXPONENT
CIBM(2) = CO
CIBM(3) = CO
CIBM(4) = CO
IIBM = IBM**4-COUNT-130
IF (IIBM.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
C----- MERGE DEC SIGN, EXPONENT AND MANTISSA
CDEC(4) = CIBM(1)
IDEC = ISHFT(IDEC,-1)
IF (SIGNFLAG) IDEC = IBSET(IDEC,31)

120  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

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

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 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.

**Subject Terms:**
Magsat, Fourier, Geomagnetic

**Number of Pages:** 196

**Price Code:** Unlimited